FGML - 2000 Tagungsplan*

	Montag 18.9.00		Dienstag 19.9.00		Mittwoch 20.9.00
			Vorträge II (Bayes)		**Vorträge V**
	Tutorials	9.15	Paaß & Kindermann (18)	9.15	Fick & Keller (126)
9.45	Thorsten Joachims: SVM (6)	9.45	Tautz & Althoff & Nick (30)	9.45	Scheffer &Wrobel (134)
		10.15	Kersting & De Raedt (37)	10.15	Schmid & Sinha & Wysotzki (139)
		10.45	- - Pause - -	10.45	Henze (141)
11.15	- - Pause - -		**Vorträge III (SVM)**	11.15	- - Pause - -
		11.00	Rychetsky & Ackermann & Glesner(45)		**Vorträge VI (Text)**
11.30	Alexander Mädche: Ontologien (6)	11.30	Joachims (48)	11.30	Dilger & Zeidler (142)
		12.00	Kindermann & Leopold &Paaß (56)	12.00	Ester & Groß (148)
		12.30	Joachims & Klinkenberg (65)	12.30	Heydemann (150)
13.00	Mittagessen	13.00	Mittagessen	13.00	Wohner (152)
	Softwarepräsentationen I		**Projektberichte**	13.30	Mittagessen
13.30	SPSS: NN	13.30	Kietz & Zücker & Vaduva: Mining Mart(73)		Fare Well Spaziergang
14.30	MIT GmbH: DataEngine	14.15	May: SPIN! (86)		
		14.45	Seeberg et al.: MediBook (96)		
		15.30	- - Pause - -		
15.30	- - Pause -		**Vorträge IV (Clustering)**		
15.45	Thinking Networks AG: b2brain	15.45	Hinneburg (106)		
		16.15	Kirsten & Wrobel (108)		
16.45	Data-Mining-Cup 2000, Bericht	16.45	- - Pause -		
	Vorträge I (Logik)		**Softwarepräsentationen II**		
17.00	Haselmann & Kókai & Sander (7)	17.00	Prudential Systems Software GmbH		
17.30	Müller (13)	17.30	Dialogis GmbH: D-Miner		
18.00	FGML-Fachgruppentreffen	18.00	IBM: Intelligent Miner		
20.00	Rheinlust: Abendessen		- - Ende - -		

* Die Zahlen in Klammern verweisen auf die Seite im Proceedings Band

Liste der Beiträge auf der FGML-2000

Tutorials

Softwaredemos

Projektberichte

Vorträge

Tutorial

Support Vector Machines

Thorsten Joachims
GMD - Institut für Autonome intelligente Systeme
Knowledge Discovery Team

Wie, wann und warum kann man in einem Raum mit 100.000 Merkmalen aus nur 1.000 Beipielen eine gute Klassifikationsregel lernen? Diese Fragen beantwortet dieses Tutorium für Support Vector Machines (SVMs), eine neuartige Lernmethode aus der statistischen Lerntheorie. SVMs vereinen lerntheoretisches Verstaendnis mit flexibler Ausdruckskraft und effizienten Trainingsalgorithmen. Sie zeichnen sich dadurch aus, dass sowohl die Trainingszeit als auch die Generalisierungsgenauigkeit nicht notwendigerweise von der Dimension des Merkmalsraumes abhängen. Zudem sind SVMs sehr flexibel und können auch nicht-lineare Klassifikationsregeln wie RBF-Netze, Polynomklassifikatoren und neuronale Netze lernen. Diese Eigenschaften machen SVMs zu einer Lernmethode speziell fuer hochdimensionale Daten (z. B. Bilderkennung, Textklassifikation).

Dieses Tutorium verfolgt zwei Ziele. Zum einen bietet es eine Einfuehrung in die lerntheoretischen und algorithmischen Grundlagen von SVMs. Zum anderen demonstiert es deren praktische Relevanz und erläutert am Beispiel der Textklassifikation, welche Eigenschaften der Anwendung zu einem erfolgreichen Einsatz von SVMs führen.

Tutorial

Entwicklung und Anwendungen von Ontologien
(Development and Applications of Ontologies)

Alexander Maedche
Institut AIFB, Universität Karlsruhe
http://www.aifb.uni-karlsruhe.de/WBS/ama

Ontologie ist ursprünglich eine philosophische Disziplin zur Untersuchung und Beschreibung der Realität, die Wissenschaft vom Seienden. In der Informatik dienen Ontologien der Konzeptualisierung von Domänen, sie beschreiben eine "explizite Spezifikation einer geteilten Konzeptualisierung". Ontologien haben sich erfolgreich in verschiedenen Anwendungsgebieten wie z.B. dem Wissensmanagement, WWW Suchmaschinen und der Informationsintegration gezeigt.

Das Tutorial gliedert sich in zwei Teile: im ersten Teil werden nach einer kurzen Definition des Terms Ontologie, Mechanismen zur Entwicklung von Ontologien vorgestellt. Dies beinhaltet Fragen zur RRepräsentation von Ontologien zum Engineering und zum Maintenance von Ontologien. Im zweiten Teil wird dann auf aktuelle, ontologiebasierte Anwendungen eingegangen. Dies umfasst das "Semantic Web", die Sprachverarbeitung (NLP) und E-Commerce. Darauffolgend werden noch Anwendungsgebiete und Forschungsfragestellungen in Bezug auf Ontologien im Bereich des Maschinellen Lernens skizziert.

GeLog - Ein genetisch logisches Programmiersystem

Ralf Haselmann*, Gabriella Kókai* und Knut Sander*

*Friedrich-Alexander-Universität Erlangen-Nürnberg,
Lehrstuhl für Programmier- und Dialogsprachen sowie ihre Compiler,
Martensstr. 3, D-91058 Erlangen,
{rfhaselm,ktsander}@cip.informatik.uni-erlangen.de, kokai@informatik.uni-erlangen.de

Abstract. *GeLog* ist ein automatisches Programmiersystem, das die Ansätze der induktiv logischen und der genetischen Programmierung miteinander kombiniert. Dieses Verfahren hat schon in anderen Projekten gezeigt, dass klassische Probleme aus der induktiv logischen Programmierung auch bei wachsender Problemgröße mit Hilfe von genetischen Algorithmen effizient gelöst werden können.
Bei der Konzeption des Systems wurde vor allem Wert auf weitreichende Parametrisierbarkeit und einfache Erweiterbarkeit gelegt, um ein möglichst breites Spektrum von Problemen bearbeiten zu können, was bei bereits existierenden Systemen oft nur schwer möglich ist. Durch entsprechende Formulierung des Grundwissens, Kombination und Konfiguration der zur Verfügung stehenden genetischen Operatoren können verschiedene Problemklassen auf *GeLog* modelliert werden.

Keywords. Genetic Programming, Genetic Algorithm, Inducitve Logic Programming

1 Einleitung

Bei der automatischen Programmerzeugung werden zwei unterschiedliche Lösungsansätze, induktiv logische Programmierung (ILP) [8] und genetisches Programmieren (GP) [3], verfolgt. Beide haben ihre Eignung bei verschiedenen Problemklassen bereits bewiesen [6]. Zur Kombination der Stärken dieser Ansätze schlagen Whigham und McKay [10] vor, bei der Entwicklung der Rekombinations- und Mutationsoperatoren des genetischen Algorithmus, die aus der induktiv logischen Programmierung stammenden Techniken der Generalisierung und Spezialisierung zu berücksichtigen, um eine 'gerichtete Breitensuche' realisieren zu können. Gleichzeitig soll durch die Verwendung von positiven und negativen Beispielen zur Lösung des Problems die Fitnessbewertung der erzeugten Individuen verbessert werden.
Zur Zeit entsteht im Rahmen des *GeLog* Projektes ein genetisch logisches Programmiersystem, welches unter anderem die bereits erwähnten Mechanismen umsetzt.

Zu Beginn eines Evolutionslaufes wird eine initiale Population erzeugt, welche dann durch Rekombinations- und Mutationsoperatoren verändert wird. Die Individuen dieser Population repräsentieren jeweils ein eigenständiges PROLOG-Programm, welches eine potentielle Lösung des gegebenen Problems darstellt. Individuen der ersten Generation werden hierbei aus dem gegebenen Grundwissen generiert [7]. Durch eine geschickte Formulierung des Grundwissens und des Algorithmus zur Individuenerzeugung lässt sich eine günstige Anfangsverteilung erreichen. Hierdurch kann der Verlauf der Evolution bereits in dieser Phase beeinflusst werden. Zur evolutionären Veränderung einer Population stehen unterschiedliche, parametrisierbare Rekombinations- und Mutationsoperatoren zur Verfügung, welche an den Problemtyp angepasst werden können. Die entstehenden Individuen werden anhand der positiven und negativen Beispiele aus dem ILP Problem bewertet. Die besten Vertreter der neu erzeugten Individuen und eventuell der Elterngeneration bilden die Basis des darauffolgenden Evolutionsschrittes [5].

2 Das *GeLog* System

Im Verlauf dieses Kapitels geben wir eine kurze Einführung zum Aufbau des Systems, wobei wir uns zu Beginn mit der Struktur der Individuen beschäftigen.

Folgendes Beispiel soll zur Veranschaulichung dienen. Für das Prädikat

```
Tochter(X0, X1)
```

müssen entsprechende PROLOG-Programme gefunden werden.

Als Grundwissen liegen konkrete Informationen über Personen und deren Verwandtschaftsverhältnisse vor:

```
Weiblich('Christine').
Weiblich('Gabi').
Weiblich('Giesela').
Weiblich('Johanna').

Maennlich('Hans').
Maennlich('Helmut').
Maennlich('Peter').

Mutter('Gabi', 'Johanna').
Mutter('Gabi', 'Peter').
Mutter('Giesela', 'Christine').

Vater('Hans', 'Johanna').
Vater('Hans', 'Peter').
Vater('Helmut', 'Christine').

Eltern('Gabi', 'Hans', 'Johanna').
Eltern('Gabi', 'Hans', 'Peter').
Eltern('Gisela', 'Helmut', 'Christine').
```

Durch Generalisierung erhält man das von *GeLog* verwendete Hintergrundwissen:

```
Weiblich(X0)
Maennlich(X0)
Mutter(X0, X1)
Vater(X0, X1)
Eltern(X0, X1, X2)
```

Mögliche (korrekte) Lösungen des Problems sind unter anderem:

```
Tochter(X0, X1)
   :- Weiblich(X0), Mutter(X1, X0).
Tochter(X0, X1)
   :- Weiblich(X0), Vater(X1, X0).
```

Zur Bewertung gefundener Lösungen werden die folgenden positiven

```
Tochter('Johanna', 'Hans').
Tochter('Johanna', 'Gabi').
Tochter('Christine', 'Giesela').
Tochter('Christine', 'Helmut').
```

und negativen Beispiele verwendet:

```
Tochter('Peter', 'Hans').
Tochter('Peter', 'Gabi').
Tochter('Christine', 'Hans').
Tochter('Christine', 'Gabi').
```

Angelehnt an die genetische Programmierung werden Individuen in einer genotypischen Repräsentation (C++-Objektgraph) gespeichert, aus der sich der Phänotyp (PROLOG-Programmcode) erzeugen lässt (Abb.**??**).

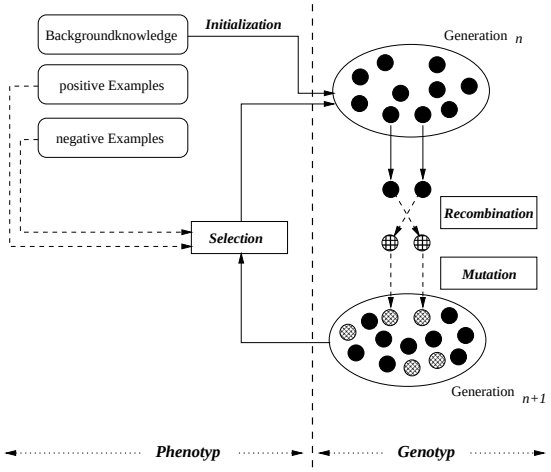


Abbildung 1 Evolutionsablauf bei *GeLog*

2.1 Evolutionsalgorithmus

Der folgende Evolutionsalgorithmus [5] fand bei der Implementierung von *GeLog* Verwendung:

$$\lambda_1 \left(\mu_0 \,/\, \rho_0 \overset{+}{,} \lambda_0 \,\|\, \Omega_0 \right)^{\gamma_0}$$

Jede der λ_1 Populationen enthält μ_0 Individuen. Für den Reproduktionszyklus werden jeweils ρ_0 mal λ_0 Elternindividuen ausgewählt und je ρ_0 miteinander rekombiniert. Die hieraus resultierenden λ_0 Nachkommen werden durch Mutationsoperatoren verändert und gemäß der gewählten Plus- oder Kommastrategie in die nächste Generation übernommen.

Bei der Kommastrategie setzt sich die neue Generation ausschließlich aus den λ_0 Nachkommen des Reproduktionszyklusses zusammen. Im Gegensatz hierzu werden durch die Plusstrategie sowohl die Nachkommen als auch Teile der Elternindividuen in die nächste Generation aufgenommen. Da die Populationsgröße in der Regel über den gesamten Evolutionsverlauf konstant bleiben soll, ergibt sich die Zahl der überlebenden Elternindividuen aus der Differenz zwischen μ_0 und λ_1.

Jede Population durchläuft bis zu γ_0 Generationen.

Der Parameter Ω_0 beschreibt die Wirkungsweise der Operatoren und kann während des Evolutionsablaufs verändert werden.

2.2 Phänotyp

Im Verlauf der Evolution stellt der Phänotyp die Grundlage für die Bewertung eines Individuums dar. Das Programmiersystem muss in der Lage sein, die einzelnen Individuen einer Generation in ihrer 'realen Umwelt' auf ihre Tauglichkeit hin zu bewerten.

Hierzu werden die PROLOG-Programme interpretiert und die berechneten Ergebnisse mit den vorhandenen positiven und negativen Beispielen verglichen. Erzielt ein Individuum einen hohen Anteil an positiven Beispielen und einen niedrigen an negativen, erhöht sich dadurch die Wahrscheinlichkeit, beim folgenden Selektionslauf in die nächste Generation aufgenommen zu werden und seine Erbinformationen weiterzugeben. Es finden außerdem Kriterien wie die Größe (Programmlänge) und die Komplexität (Anzahl der unterschiedlichen Literale) des Individuums bei der Bestenauswahl Beachtung.

2.3 Genotyp

Der Genotyp stellt die eigentliche Repräsentation des Individuums dar und wird als Objektgraph gespeichert. Aus diesem kann der Phänotyp abgeleitet und bewertet werden.

Ein Individuum besteht aus einem Prädikat, welches wiederum mehrere rechte Seiten besitzen kann. Diese setzen sich aus logisch verknüpften Basisprädikaten des Hintergrundwissens zusammen (Abb. **??**).

```
Tochter(X0, X1)

   :- Weiblich(X0), Mutter(X1, X0).

                        ⋮

   :- Mutter(X0), Vater(X1, X0),
      Maennlich(X1).
```

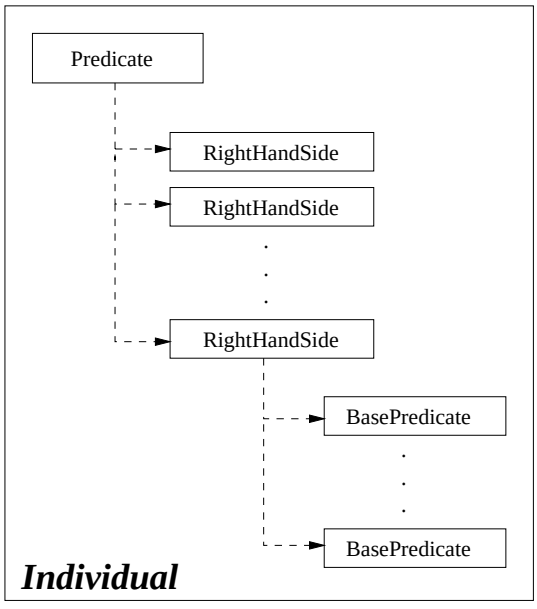


Abbildung 2 Objektgraph eines Individuums

Bei der initialen Erzeugung eines Individuums werden Teile des Grundwissens zu rechten Seiten eines Prädikats zusammengefügt, wobei die Auswahl dieser Basisprädikate durch anzugebende Wahrscheinlichkeiten geschieht. Auf diesem Wege ist es möglich, die Häufigkeit und Verteilung der einzelnen Komponenten des Grundwissens innerhalb der Individuen der ersten Generation (Seeding) zu beeinflussen.

2.4 Generationen und Populationen

Individuen werden zu einer Generation zusammengefasst, die wiederum einer Population angehören. Durch Bewertung einer Generation, Selektion, Rekombination und Mutation entstehen neue Individuen, die wiederum die Nachfolgegeneration innerhalb der Population bilden.

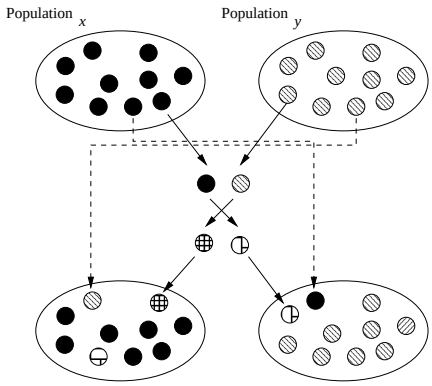


Abbildung 3 Metaevolution

Mehrere Populationen können dabei unabhängig nebeneinander im selben Suchraum existieren. Mittels Metaevolution (Abb. **??**) [5] ist ein regelmäßiger Wissensaustausch zwischen diesen Populationen möglich. Die restliche Zeit entwickeln sie sich getrennt voneinander (Niching). Auf diese Weise können sich die Populationen (besonders gegen Ende der Evolution) auf unterschiedliche Bereiche im Suchraum konzentrieren und somit unterschiedliche Lösungsmöglichkeiten verfolgen.

2.5 Operatoren

Unterschiedliche Rekombinationsoperatoren (Abb. **??**) erlauben es, sowohl gesamte als auch nur Teile von rechten Seiten zwischen Individuen auszutauschen (Crossover). Auf diese Weise lässt sich der Transfer von ‘Wissen’ (Programmcode) zwischen zwei Individuen mit verschiedener Granularität realisieren.

Im Gegensatz dazu arbeiten Mutationsoperatoren jeweils nur auf einem Individuum (Abb. **??**).

Partielles Crossover zweier rechter Seiten:

```
Tochter1(X0, X1) :- Vater(X0, X1), Weiblich(X0).
Tochter2(X0, X1) :- Mutter(X1, X0).
                    |
                    v
Tochter1(X0, X1) :- Mutter(X1, X0), Weiblich(X0).
Tochter2(X0, X1) :- Vater(X0, X1).
```

Crossover zweier rechter Seiten:

```
Tochter1(X0, X1) :- Vater(X0, X1), Weiblich(X0);
                    Maennlich(X0).
Tochter2(X0, X1) :- Weiblich(X0), Mutter(X1, X0);
                    Vater(X0, X1).
                    |
                    v
Tochter1(X0, X1) :- Vater(X0, X1), Weiblich(X0);
                    Weiblich(X0), Mutter(X1, X0).
Tochter2(X0, X1) :- Maennlich(X0);
                    Vater(X0, X1).
```

Abbildung 4 Rekombinationsoperatoren

Prinzipiell lassen sich Mutationsoperatoren in drei Kategorien einteilen, jenachdem ob sie neues Wissen (neuen Programmcode) hinzufügen, Wissen entfernen oder bestehendes Wissen modifizieren. Konkret werden Basisprädikate hinzugefügt, entfernt oder deren Variablen verändert.

Entfernen von Wissen:

```
Tochter(X0, X1) :- Maennlich(X0), Weiblich(X0).
                   |
                   v
Tochter(X0, X1) :- Weiblich(X0).
```

Neues Wissen einbringen:

```
Tochter(X0, X1) :- Weiblich(X0).
                   |
                   v
Tochter(X0, X1) :- Vater(X0, X1), Weiblich(X0).
```

Wissen mutieren:

```
Tochter(X0, X1) :- Vater(X0, X1), Weiblich(X0).
                   |
                   v
Tochter(X0, X1) :- Vater(X1, X0), Weiblich(X0).
```

Abbildung 5 Mutationsoperatoren

Hierzu wurden auch spezielle Operatoren zur Generalisierung bzw. Spezialisierung des erlernten Wissens implementiert. Eine generelle Zugehörigkeit eines Operators zu einer der beiden Klassen lässt sich aber im Allgemeinen nicht vorhersagen, da es sich beispielsweise bei der Umbennenung einer Variablen sowohl um eine Generalisierung als auch um eine Spezialisierung handeln kann.

Die Anwendungswahrscheinlichkeit und Intensität eines Operators sowie deren Veränderungen im Verlauf der Evolution werden zu Beginn festgelegt. So kann die Sprungweite beim Abtasten des Suchraumes reguliert und problemspezifisch angepasst werden. Damit kann verhindert werden, dass die Population am Anfang durch zu kleine Mutationssprünge auf einem Nebenmaxima ‘hängen’ bleibt und am Ende bereits erreichte Maximas durch einen zu großen Mutationssprung wieder verlassen werden.

2.6 Selektion und Fitness

Das Ziel der Selektion ist die Auswahl der besten Individuen einer Generation, um deren erlerntes Wissen in die nächste Generation weiterzugeben. Eine Fitnessfunktion wird verwendet, um die Überlebensfähigkeit der einzelnen Phänotypen miteinander vergleichen zu können.

Zur Bewertung werden die als Objektgraphen repräsentierten Genotypen in den entsprechenden Phänotyp umgesetzt und durch den PROLOG-Interpreter ausgewertet. Ein Kriterium für die Fitness eines Individuums stellt hierbei die Anzahl der erzielten positiven Lösungen dar. Ein Individuum gilt für dieses Kriterium als ‘ideal’, wenn alle positven und kein negatives Beispiel mit diesem berechnet werden können.

Desweiteren fließen sowohl die Länge des Programmcodes als auch die Ausführungsdauer zur Berechnung der Lösungen des Prädikats in den Selektionsalgorithmus ein. Bei *GeLog* wurde zudem das aus [5] bekannte Verfahren implementiert, welches zwischen der Selektion aus den Kinderindividuen, bzw. aus den Eltern- und Kinderindividuen unterscheidet (Komma-, Plus- strategien).

Für die Auswahl der Individuen zum Reproduktionszyklus stehen verschiedene Selektionsmechanismen zur Verfügung:

- rangbasierte Auswahl
- fitnessproportionale Auswahl
- Eliteauswahl
- Bestenauswahl

Wobei es sich bei den ersten drei um, für genetische Algorithmen üblich, probabilistische Auswahlverfahren handelt. Die Bestenauswahl dagegen findet vorallem Verwendung zur Auswahl der überlebenden Elternindividuen bei Plusstrategien.

2.7 Konfigurierbarkeit des Systems

Da *GeLog* nicht für eine bestimmte Problemstellung entwickelt werden sollte, wurde bereits bei der Kon-

zeption des Systems auf weitgehende Konfigurierbarkeit geachtet. Hierdurch wird eine Anpassung der Evolution ermöglicht ohne das System selbst neu übersetzen zu müssen.

In der Konfigurationsdatei lassen sich neben den Parametern der Evolutionsstrategie (λ_1, μ_0, ρ_0, $^+_,$, λ_0, Ω_0, γ_0) ferner die Gewichte der Bewertungskriterien für die Fitnessfuktion und die Selektionsmethode selbst festlegen. Neue entwickelte Operatoren können durch Angabe in dem Kofigurationsfile in das System integriert, bzw. nicht benötigte entfernt werden.

Zur initialen Erzeugung der Individuen kann mit Hilfe von Konfigurationsparametern die Form der Phänotypen bestimmt werden (maximale Anzahl der rechten Seiten je Individuum und maximale Anzahl von Literalen je rechter Seite des Individuums).

3 Beispiel

Im Folgenden sollen noch für ein konkretes Beispiel die von *GeLog* dazu gelieferten Ergebnisse gezeigt werden. Dabei handelt es sich um das Problem, Pilze anhand von Merkmalen eindeutig als giftig zu erkennen. Das Beispiel stammt aus der *Mushrooms Database* aus dem *UCI Machine Learning Repository* [12]. Dieses Problem wurde von uns als Arbeitsbeispiel gewählt, da es einen sehr großen Suchraum aufspannt, das Hintergrundwissen auch nach der Generalisierung relativ umfangreich ist und eine große Menge positiver sowie negativer Beispiele gegeben sind.

Gegeben sind eindeutig als essbar oder giftig klassifizierte Beschreibungen von Pilzen. Ein Pilz wird durch einen Vektor von nominalen Werten für seine 22 Merkmale beschrieben. Zudem ist seine Klassifizierung, essbar oder giftig, bekannt. Insgesamt besteht das Beispiel aus 8124 verschiedenen Pilzbeschreibungen. Davon sind $48,2\%$ als giftig klassifiziert.
Es gibt mehrere einfache Lösungen für dieses Problem, die nahezu alle Beispiele richtig erkennen.

Als Hintergrundwissen erhält *GeLog* 126 verschiedene Basisprädikate, die einen Pilz auf genau einen Wert für ein bestimmtes Merkmal testen und entweder richtig oder falsch liefern.

Aus diesen soll das Prädikat `poisonous/1` entwickelt werden, wobei der Parameter für einen Vektor mit der Beschreibung eines Pilzes steht.

Der Evolutions-Algorithmus wurde mit fitnessproportionaler Auswahl und folgenden Parametern verwendet:

$$10\,(30\,/\,2\,+\,20\,\|\,\Omega_0)^{80}$$

Das bedeutet, es werden zehn unabhängige Populationen mit jeweils 80 Generationen berechnet. Eine Generation besteht aus 30 Individuen, wobei immer die zehn besten Individuen in die nächste Generation übernommen werden und 20 neue Individuen als Nachkommen aus je zwei Elternindividuen erzeugt werden.

Auf Abbildung **??** ist die Entwicklung der prozentualen Fitness über den Evolutionsablauf gezeigt. Dargestellt sind die Gesamtfitness der Generation und die des aktuell besten Individuums, jeweils für die beste Population sowie für das Mittel aus zehn Durchläufen.

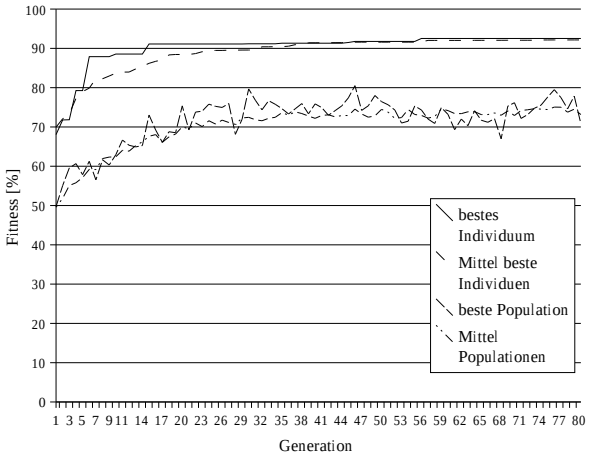


Abbildung 6 Fitnesswerte über den Evolutionsablauf

Das beste Individuum in diesem Evolutionslauf wurde in der 56. Generation seiner Population erzeugt und überlebte bis zum Ende:

```
poisonous1149(X0)
  :- gill_size_n(X0),
     stalk_surface_above_ring_y(X0).
  :- gill_size_n(X0), stalk_shape_e(X0).
  :- odor_f(X0).
  :- ring_type_n(X0).
  :- gill_size_n(X0), gill_spacing_c(X0).
```

Es erreicht eine Gesamtfitness von 92.50%, wobei es 93.16% Beispiele richtig erkennt.

Für das Berechnen der 80 Generationen einer Population wurden duchschnittlich etwa 5 Minuten auf einem PentiumIII/600 unter LINUX benötigt.

4 Schlussbemerkung

Mit *GeLog* steht ein universelles Werkzeug zur Bearbeitung von Problemstellungen aus dem ILP Bereich zur Verfügung. Zu diesem Zweck wurde eine unabhängige Wissensrepräsentation implementiert, auf der mit allgemein formulierten genetischen Operatoren und Selektionsfunktionen gearbeitet werden kann.

Weiterführende Arbeiten müssen sich mit einer statistischen Analyse von *GeLog* befassen. Vergleiche zu bestehenden Systemen aus dem Bereich der induktiv logischen und der genetischen Programmierung, sowie zu anderen, die wie *GeLog* diese beiden Ansätze miteinander kombinieren, sollten angestellt werden.

Zudem ist die Möglichkeit der Einbindung anderer Phänotypräume, zum Beispiel LISP oder JAVA an Stelle von PROLOG, und deren Auswirkung auf die Effizienz des Systems und der entstehenden Programme zu betrachten.

5 Literatur

1. T. Bläck: *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, 1996

2. Botta M., Giordana A.: *SMART+: A Multi-Strategy Learning Tool*, Proc of the IJCAI-93, 1993

3. L. Davis: *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991

4. Hekanaho, J.: *DOGMA: A GA-Based Relational Learner*, TUCS Technical Report, 1997, Turku, Finland

5. Ch. Jacob: *Principia Evolvica - Simulierte Evolution mit Mathematica*, dpunkt Verlag, 1. Auflage, 1997

6. L.R. Tang, M.E. Califf, R.J. Mooney: *An Experimental Comparison of Genetic Programming and Inductive Logic Programming on Learning Recursive List Functions*, TR AI98-271, Artificial Intelligence Lab, University of Texas at Austin, May 1998.

7. N. Lavrac, S. Dzeroski: *Inductive Logic Programming - Techniques and Applications*, Ellis Horwood Limited, 1. Auflage, 1994

8. S. Muggleton: *Inductive Logic Programming - Techniques and Applications*, Academic Press, 1992

9. S.-W. Ninhuys-Cheng, R. de Wolf: *Foundations of Inductive Logic Programming*, Springer Verlag, 1996

10. Whigham P.A., McKay R.I.: *Genetic programming and inductive logic*, Technical Reports CS14/94, University College, University of New South Wales, 1994

11. Man L.W., M.L., Leung K.S.: *Genetic Logic Programming System: Inducing Logic Programs With Genetic Algorithms*, IEEE Expert/Intelligent Systems and Their Applications Vol. 10, No. 5, October 1995

12. Blake, C.L. and Merz, C.J. (1998): *UCI Repository of machine learning databases*, [www.ics.uci.edu/˜mlearn/MLRepository.html], Irvine, CA: University of California, Department of Information and Computer Science.

Inductive Logic Programming for Learning User Models

Martin E. Müller
Institute for Semantic Information Processing
University of Osnabrück, D-49069 Osnabrück, Germany
Martin.Mueller@cl-ki.uni-osnabrueck.de

Abstract

User modeling systems infer user models from user interaction, store user models and induce new assumptions by reasoning about the models. In course of the OySTER project[1] our goal is to induce lucid conceptual user models by means of inductive logic programming methods. With growing knowledge about a single user we need to refine induced user models. With a growing set of users we will also have to refine the underlying ontology in order to learn a suitable representation formalism. In this paper we give an outline of how both user models and representation language can be learned simultaneously.

Keywords. Inductive Logic Programming, Machine Learning for User Modeling, Conceptual User Models, Web Search.

1 Introduction

In this paper we describe the formal framework of a strongly biased ILP approach within the restricted domain of user modeling for web search. Currently, we are working on an implementation of the approach presented in this paper.
User models within OySTER are used to describe a user's interest in web documents. Instead of representing a document d as a word vector (c.f. [Bal98, BP97, JFM97, PMB96]), we use two classifying algorithms in order to determine a *document type* and a *document category*. The former describes a text type $t \in \mathcal{T}$ while the latter describes a content based text class $c \in \mathcal{C}$.

2 Concepts, Clauses and User Models

$\langle \mathcal{T}, \sqcup_{\mathcal{T}} \rangle$ is a semi-lattice of document types t with an ordering relation $\sqsubseteq_{\mathcal{T}}$. Similarly, $\mathcal{C}$ is a semi-lattice of document categories c with an ordering relation $\sqsubseteq_{\mathcal{C}}$. We extend $\mathcal{C}$ by an artificial bottom element $\bot_{\mathcal{C}}$ and define $c \sqcap_{\mathcal{C}} c' = \bot_{\mathcal{C}}$ for any c, c' with $c \sqcap c' \notin \{c, c'\}$ to yield a lattice $\langle \mathcal{C}, \sqcup_{\mathcal{C}}, \sqcap_{\mathcal{C}} \rangle$. For a document d, $\mathbf{A}_{\mathcal{C}}(d)$ and $\mathbf{A}_{\mathcal{T}}(d)$ deliver the category c and the type t of d.
Now, we interpret $\mathcal{T}$ as sorts over expressions in terms of $\mathcal{C}$. In other words, the sorted term $t::c$ with sort (i.e. type) $t \in \sqcup$ and category $c \in \mathcal{C}$ denotes a concept that "contains" any document for which $\mathbf{A}_{\mathcal{T}}(d) = t'$ with $t \sqsubseteq t'$ and $\mathbf{A}_{\mathcal{C}}(d) = c'$ with $c \sqsubseteq c'$. Thus, a user model containing information about the user's interest can be represented by a set of sorted expressions over $\mathcal{T} \times \mathcal{C}$; or—more informal—as shaded parts of a directed acyclic graph which initially is a tree. A user model M_u consists of two parts representing what the user is interested in (M_u^+) and what the user is *not* interested in (M_u^-). Each of those sets again consists of sets of sorted expressions called *aspects*. Aspects are used to formalize different and independent facets of a user's interest. One could as well introduce a set of user models $M_u, M_{u'}, \ldots$ for a single user, but aspects still allow for more compact representation which is more lucid. Now, given a predicate `a_TxC`$(d, t::c)$ which for a document d delivers $\mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d) = \mathbf{A}_{\mathcal{T}}(d)::\mathbf{A}_{\mathcal{C}}(d)$ we define M_u by two 3–ary predicates `pinterest` and `ninterest` as follows:

```
pinterest(u, 01, D) :-
  a_TxC( D, 13:'machine learning'),     (1.1)
  a_TxC( D, 12:'user modeling'),        (1.2)
  a_TxC( D, 19:'cognitive science').    (1.3)
pinterest(u, 01, D) :-
  a_TxC( D, 13:'user modeling'),        (2.1)
  a_TxC( D, 19:'tutoring systems'),     (2.2)
  a_TxC( D, 12:'machine learning').     (2.3)
pinterest(u, 02, D) :-
  a_TxC( D, 11:'diving'),               (3.1)
  a_TxC( D, 15:'submarine robotics').   (3.2)
ninterest(u, 01, D) :-
  a_TxC(D, 13:'language acquisition'),  (4.1)
  a_TxC(D, 12:'student modeling'),      (4.2)
  a_TxC(D, 19:'cognitive science').     (4.3)
ninterest(u, 01, D) :-
  a_TxC(D, 13:'student modeling'),      (5.1)
  a_TxC(D, 19:'tutoring systems'),      (5.2)
```

[1] see http://mir.cl-ki.uni-osnabrueck.de/oyster/

```
a_TxC(D, 12:'human_comp interact').   (5.3)
```

For an explanation of the sorts, see [Mül00b]. Using a sorted variant of SLD resolution [Hed89], we can define several levels of "interestingness":

1. u is interested in a document d, if $\exists a : M_u^+ \vdash \texttt{pinterest}(u,a,d)$. We denote this case by writing $M_u \Vdash_a \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d)$.
2. d could be interesting for u, if $\forall a : M_u^+ \nvdash \texttt{ninterest}(u,a,d)$. We denote this case by writing $M_u \Vdash \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d)$.
3. u is indifferent about d, if $\forall a : M_u^+ \cup M_u^- \nvdash \texttt{pinterest}(u,a,d) \vee \texttt{ninterest}(u,a,d)$. We denote this case by writing $M_u \nVdash \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d)$.

$M_u \nVdash \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d)$ and $M_u \Vdash_a \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d)$ can be constructed as the dual case to 1. and 2.

3 Inducing User Models

By relevance feedback the system receives a labeled sample which is used to induce a user model. An example is a 3–tuple $\langle d, \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d), [(a_i, l_i, m_i)]_{i\in I}\rangle$ where $[(a_i, l_i, m_i)]_{i\in I}$ is a list of triples of aspects a_i , labels $l_i \in \{\oplus, \ominus\}$ and model references $m_i \in \{\mathsf{p}, \mathsf{n}\}$. This way, one document can be associated with different classifications. Furthermore, each document–classification pair can simultaneously be labeled as negative or positive with respect to several aspects. For example, $\langle \mathtt{d}, \mathtt{t}::\mathtt{c}, [(a, \oplus, \mathsf{p})]\rangle$ means, that d (if classified as t::c) is a positive ($\oplus$) example for the user interest (p) aspect a. Similarly, $\langle \mathtt{d}', \mathtt{t}'::\mathtt{c}', [(b, \ominus, \mathsf{n})]\rangle$ means, that d′ (if classified as t′::c′) is a negative ($\ominus$) example for the user's non-interest (n) aspect b. In consequence, we would like to learn $M_u = \langle M_u^+, M_u^-\rangle$ such that:

$$\begin{array}{lll} M_u^+ & \Vdash_a & \mathtt{t}::\mathtt{c} \text{ , that is:} \\ M_u^+ & \vdash_{\text{SLD}} & \texttt{pinterest}(u,a,\mathtt{d}) \end{array}$$

and

$$\begin{array}{lll} M_u^- & \nVdash_b & \mathtt{t}'::\mathtt{c}' \text{ , that is:} \\ M_u^- & \vdash_{\text{SLD}} & \texttt{ninterest}(u,b,\mathtt{d}') \end{array}$$

for some aspects a, b. Using conceptual user models poses a well defined learning task for inductive logic programming techniques.

3.1 Learning initial User Models

A standard bootstrapping method for obtaining initial feedback data is to collect the user's "hot links" from his homepage or link list, c.f. [CSB00]. All those documents d build a sample $S = \{\langle d, \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d), [(a, \oplus, \mathsf{p})]\rangle \,|i \in I\}$. Note, that all examples refer to the same aspect a. On this sample, we can simply construct a complete consistent hypothesis:

$$M_u^+ = \left\{ \begin{array}{c} \texttt{pinterest}(u,a,d)\texttt{:-} \\ \texttt{a\_TxC}(d, t::c). \end{array} \;\middle|\; \begin{array}{c} \forall\, \langle d, t::c, [(a, \oplus, \mathsf{p})]\rangle \\ \in S \end{array} \right\}$$

Since at this point $M_u^- = \emptyset$, M_u most likely is not correct.

3.2 Learning User Models

In [Mül95] we presented a calculus for order sorted inverse resolution that was based upon order sorted resolution as, e.g., described in [Hed89]. The initial user model as described in the last section can be taken as input for a truncation operator. As an example, imagine $\mathcal{C}$ with $c = \bigsqcup_{\mathcal{C}}\{c_1, c_2, c_3, c_4\}$. Furthermore, let $\langle d_i, t_i::c_i, [(a, \oplus, \mathsf{p})]\rangle \in S$ for $i = 1, 2, 3$ and neither $\langle d_4, t_4::c_4, [(a, \ominus, \mathsf{p})]\rangle \in S$ nor $\langle d_4, t_4::c_4, [(a, \oplus, \mathsf{n})]\rangle \in S$.[2] Then,

(1) $\quad H_1 = \texttt{pinterest}(u,i,d)\texttt{:-a\_TxC}(d, t::c).$

would be a consistent hypothesis with $t = \bigsqcup_{i=1}^{3} t_i$, i.e. t is the lub of all types of documents in the sample under consideration and $\mathcal{T}h(M_u^+) \subset \mathcal{T}h(M_u^+ \cup \{H_1\})$. Inducing M_u^- is a dual case with respect to labels and model references. The interesting part is to induce new predicates describing a user's interest: Intra–Construction leads to new concept definitions. Sticking to (simple) generalizations, one would derive

$$\begin{array}{ll} (2) & H_2 = \texttt{pinterest}(u, 01, D)\texttt{:-} \\ & \qquad \texttt{a\_TxC}(D, (t_1 \sqcup_{\mathcal{T}} t_2)::c_1), \\ & \qquad \texttt{a\_TxC}(D, (t_1 \sqcup_{\mathcal{T}} t_2)::c_2), \\ (*) & \qquad \texttt{a\_TxC}(D, (t_3 \sqcup_{\mathcal{T}} t_4)::(c_3 \sqcup_{\mathcal{C}} c_4)). \end{array}$$

given a constellation like clauses 1 and 2 in the program listed above. But since $\sqcup$ would lead to over-generalizations, one would rather try fanning operators like intra–construction thus replacing $(*)$ by:

$$(*') \qquad \texttt{p}(u, 01, D).$$

and

$$\begin{array}{ll} (3) & \texttt{p}(u, 01, D)\texttt{:-a\_TxC}(D, t_3::c_3). \\ & \texttt{p}(u, 01, D)\texttt{:-a\_TxC}(D, t_4::c_4). \end{array}$$

[2] This means: u has not labeled $t_4::c_4$ as a counterexample for his interest nor has he labeled this evidence as an example for his non–interest.

A simple lgg on the clauses in (3) would lead to a user model that is equivalent to adding H from (1). Thus, we need to find a definition H for $\mathtt{p}(u, \mathtt{01}, D)$, such that (1) entails H and H entails (3). In other words, we need to invent a new class. Using a folding like operator which is similar to inter-construction one would derive a hypothesis that is symmetric to (2) in some sense:

(4) $H_3 =$ `pinterest`$(u, \mathtt{01}, D)$ `:-`
$\mathtt{p}(u, \mathtt{01}, D)$,
(4a) $\mathtt{a\_TxC}(D, t_3 :: c_3)$.
and
`pinterest`$(u, \mathtt{01}, D)$ `:-`
$\mathtt{p}(u, \mathtt{01}, D)$,
(4b) $\mathtt{a\_TxC}(D, t_4 :: c_4)$.

and

(5) $\mathtt{p}(u, \mathtt{01}, D)$ `:-`
$\mathtt{a\_TxC}(D, (t_1 \sqcup_{\mathcal{T}} t_2) :: c_1)$,
$\mathtt{a\_TxC}(D, (t_1 \sqcup_{\mathcal{T}} t_2) :: c_2)$.

This operation is just a natural way of expressing different aspects of an interest: Changing the rule heads in H_3 to new aspect identifiers `01a` and `01b`, lines 4a and 4b define the aspects of the code in 5. Note, that due to the strong language bias on predicate names and the use of atomic terms only application of an absorption operator becomes is a tractable task; though it does not seem to be very helpful.

3.3 Learning Ontologies

Given the last example of the preceeding section, we also learned new categories. The concept defined by rule 5 corresponds to the body literal conjunctions `(1.1),(1.2)` and `(2.1),(2.3)` of the example clauses in the sample program. Actually, this conjunction thus represents an intersection of the both concepts. Satisfiability of such a clause would force $\mathbf{A}_{\mathcal{T}\times\mathcal{C}}$ to be *non*–deterministic: A document d must be belong to both c_1 and c_2. This leads to two different conclusions: First, one demands $\mathbf{A}_{\mathcal{T}\times\mathcal{C}}$ to deliver an ordered sequence of classifications $\mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d) = [\langle t::c, p\rangle, \langle t'::c', p'\rangle, \langle t''::c'', p''\rangle, \ldots]$ together with a probability score—or, say, a distance measure—of successively decreasing quality. In consequence, we would need to extend the unification and subsumption such that:

1. $\langle t::c, p\rangle$ subsumes $\langle t'::c', p'\rangle$ if $t \sqsubseteq_{\mathcal{T}} t'$ and $c \sqsubseteq_{\mathcal{C}} c'$ and $p' - p \leq \vartheta$, where ϑ is a threshold which determines a lower confidence boundary for p' relative to p.
2. $\langle T::C, P\rangle$ unifies $\langle t::c, p\rangle$ if $T \sqsubseteq_{\mathcal{T}} t$ or T is a variable, and $C \sqsubseteq_{\mathcal{C}} c$ or C is a variable, and $p - P \leq \vartheta$ with ϑ as above.

Probabilistic derivation then also gives rise to defining an according inverse operation with a further bias ϑ^{-1}. Second, one would actually invent a new class $c_1 \sqcap_{\mathcal{C}} c_2$. This would require to define a new $\mathbf{A}_{\mathcal{C}}$ which only makes sense if this process could be automated by means of learning new classifiers as well. At this point, discrimination of classification and user modeling tasks turns out to be a severe drawback of this approach (see conclusion). Concerning disjunctive new classes as in the clauses in program 3, a new class can be canonically defined by $c_3 \cup c_4$, but certainly *not* by $c_3 \sqcup c_4$. The latter method for defining can be justified only by a generalization threshold which determines a maximum allowed distance in $\mathcal{C}$ which can be defined as the maximum number n of intermediate concepts: $\max_n(c \sqsubset c'_1 \sqsubset \cdots \sqsubset c'_n \sqsubset c_{i,j})$. We consider defining ϑ_n in relation to an information gain measure with respect to the existing user model and the labeled data set. In both cases, user feedback for class or aspect definitions should be requested for a better guidance, i.e. bias, in the induction process.

4 Acquiring Labeled Data

The WWW as application domain has a crucial drawback: Labeled training data is very hard to obtain from a user. Only few users will be provide at least partial feedback about delivered search results. On the other hand [Bal98] presented Slider, an interface which helps in interpreting user interactions as feedback. Within the Slider, search results can be ordered and grouped such that the different actions can be interpreted as feedback on the results. In context of web search with respect to user models and interest aspects, the Slider idea can be carried over to this application. Search results are presented together with their classifications $t::c$, as well as an explanation why they have passed the filtering process. This enables the user to "drag" a "misplaced" document d somewhere else or even to delete it—thus performing a labeling of d as positive for some other aspect or as an instance of `ninterest`. When learning user models, we focus on aspects a_i. This helps in increasing the amount of labeled data: By mutual exclusive sets, positive examples for aspects a_i can be used as negative examples for a_j and vice versa. The same applies even for the dual case. Furthermore we can make use of user communities, i.e. by use of collaborative filtering. The set of positive examples can be expanded by examples that were labeled by other users with

"similar" interests. Similarity can be defined in a completely discrete manner using our notion of interestingness. Since semantic entailment of different user models like $M_u \models\!\!\approx M_v$ is hard to show, we make use of our relations $\Vdash\!\!\!\vdash$ and $\Vdash$. Here, the choice of a set of documents d for which the following holds determines the notion of similarity:

1. If $M_u \Vdash\!\!\!\vdash \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d)$ implies $M_v \Vdash\!\!\!\vdash \mathbf{A}_{\mathcal{T}\times\mathcal{C}}(d)$, u's interest is similar to v's interest: $u \approx v$. Note, that $\approx$ is not symmetric!
2. Similarity is weaker, if we replace $\Vdash\!\!\!\vdash$ by $\Vdash$. Accordingly, we denote this case by $u \sim v$.

More labeling information can be extracted from other users where E_v^- coincides with E_u^+. This way, we can define a sequence of labeled data sets:

1. Initially, we use the set of documents u labeled as interesting with respect to aspect a_i: $\mathbf{E}_0^+ = E_u^+(a_i)$ $(\mathbf{E}_0^- = E_u^-)$.
2. Positive feedback concerning *different* aspects a_j can be regarded as negative feedback for a_i: $\mathbf{E}_1^- = \mathbf{E}_0^- \cup E_u^+(j)$.
3. Similar users v contribute to both sample sets: $\mathbf{E}_1^+ = \mathbf{E}_0^+ \cup E_v^+$, $\mathbf{E}_2^- = \mathbf{E}_1^- \cup E_v^-$.
4. Different users v contribute to both dual sample sets: $\mathbf{E}_2^+ = \mathbf{E}_1^+ \cup E_v^-$, $\mathbf{E}_3^- = \mathbf{E}_2^- \cup E_v^+$.

Introducing new concepts into $\mathcal{C}$ can be triggered by several events. $\mathcal{C}$ obviously is too coarse, if for two users u and v their respective models are similar (according to the notion above) although their labeled samples are heterogeneous (which can be determined extensionally or by explicit requests for label data). Another key for coarseness is the size of peak concept extensions: If classifiers do not distribute documents widely over the classification tree we need to split 'crowded' concepts. Now, imagine some clauses $m_u^i = m_v^j$; i.e. equivalent user model aspects for different users.

1. If for a sufficiently number of users their respective labeled data sets form a significantly large intersection, the union of all data can be used to derive a group model, while the disjoint unions can be used for collaborative labeling.
2. If the labeled data sets are nearly completely disjoint, it is quite reasonable, that m_u^i and m_v^j should not be equal and we need to learn a new concept class.

In the latter case, we can use the example sets for mutual exclusive learning tasks in order to learn new concepts.

5 Conclusion and Prospects

5.1 Conclusion

In this paper, we have formally described a machine learning based approach for user modeling which makes use of the rather exotic methods of inductive logic programming. Due to its application domain, we can define a strong bias which shall help in implementing efficient variants of order sorted inverse resolution. The choice of an ILP approach was motivated by working on user model representations that are transparent to the user (namely Horn clauses where predicates are "tagged" to concepts in a concept hierarchy) and by providing means for a lucid information filtering process (which we can define through different levels of interestingness). The disadvantage of this approach is that performance depends on *both* classification and user modeling. Thus, we presuppose *perfect* classifiers. Even worse, induction of new concepts presupposes *perfect* learning of new perfect classifiers each time a new concept has been invented. In the recent past it has been argued a lot that many approaches for machine learning based user modeling violate theoretical preconditions in application of the theory as well. Nevertheless, domain restriction ensures a pretty good performance. In this tradition, our future evaluation work will show if deliberate violation of our assumptions causes worse results in practice.

5.2 Current work and prospects

The approach presented so far also gives rise to several interesting open research questions. Especially, section 3 poses interesting questions about defining biases that determine when to introduce new classes in terms of intersecting label data. More theoretically, it is an interesting question how many examples are needed in relation to the coarseness of an initial concept hierarchy in order to be able to induce sufficiently precise user models.
Among implementational issues, these questions are in focus of current work.
A recent detailed description of the system can be found in [Mül00c].

References

[Bal98] M. Balabanovic. An interface for learning multi-topic user profiles from implicit feedback. In *AAAI-98 Workshop on Recommender Systems*, Madison, Wisconsin, July 1998.

[BP97] Daniel Billsus and Michael Pazzani. Learning probabilistic user models. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Sixth International Conference, UM97.*

Springer Wien New York, Vienna, New York, 1997. Workshop Paper.

[CSB00] Christoph Clodo, Eric Schwarzkopf, and Mathias Bauer. Einsatz adaptiver Techniken für die UM2001. In Müller [Mül00a].

[Hed89] U. Hedstück. A calculus for order–sorted predicate logic with sort literals. In K. H. Bläsius, U. Hedstück, and C. R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, volume 418 of *Lecture Notes in Artificial Intelligence*, pages 61–72. Springer, Berlin, 1989.

[JFM97] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of IJCAI 97*, August 1997.

[Mül95] M. E. Müller. Eine Erweiterung der inversen Resolution auf sortierte Hornklauselmengen. In K. Morik and J. Herrmann, editors, *Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen*, number 580 in Forschungsberichte der Universität Dortmund, Fachbereich Informatik, 1995.

[Mül00a] M. E. Müller, editor. *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*. Institut für Semantische Informationsverarbeitung, Universität Osnabrück, 2000.

[Mül00b] M. E. Müller. Learning comprehensible conceptual user models for user adaptive meta web search. In *AAAI Spring Symposium on Adaptive User Interfaces*, Stanford, CA, 2000.

[Mül00c] M. E. Müller. OySTER: Web Search as a playground for User Modeling techniques. In *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen* [Mül00a].

[PMB96] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting Web Sites. In *Proc. of the National Conference on Artificial Intelligence*, Portland, OR, 1996.

Explaining Complex Classification Models for Credit Scoring

Gerhard Paass and Jörg Kindermann
GMD – Forschungszentrum Informationstechnik D-52754 Sankt Augustin
paass,kindermann@gmd.de

Abstract. Complex classification models like neural networks usually have lower errors than simple models. They often have very many interdependent parameters, whose effect no longer can be understood by the user. For many applications, especially in the financial industry, it is vital to understand the reasons why a classification model arrives at a specific decision. We propose to use the full model for the classification and explain its results by an explanation model capturing its main functionality. For a real world credit scoring application we investigate a spectrum of explanation models of different type and complexity. We assess them in terms of transparency and difference to the full model. It turns out that decision trees, scorecards and newly developed scorecards within decision trees show the best compromise in this respect and perform better than nonlinear variants.

Keywords. credit scoring, decision trees, scorecards, explanation of complex models

1 Introduction

During the last years the availability of business process data allowed the utilization of sophisticated prediction and classification models in enterprises. A prominent example is credit scoring, where a customer is classified as solvent or insolvent based on a vector $\mathbf{x}$ of features. The unknown model parameters are trained using data on credit histories of customers.

There is a plethora of models starting from simple linear models to complex neural networks, support vector machines and multivariate adaptive regression splines (MARS) to name a few. Experimental evaluations during the last years showed that more complex procedures in general yield better results in terms of classification errors than traditional approaches. These methods usually have many – sometimes thousands – intimately interdependent parameters, and the meaning and effects of single parameters no longer can be understood by the user.

There is a tradeoff between the classification reliability and transparency of approaches. The necessity to understand the result often is more important than a gain in precision. In addition the banking industry in the U.S. has to explain a credit decision to the customer: "If you are denied credit, the Equal Credit Opportunity Act requires that the creditor gives you a notice that tells you the specific reasons your application was rejected . . . . Indefinite and vague reasons for denial are illegal . . . . Acceptable reasons include: 'Your income was low' or 'You haven't been employed long enough'."[Cen00].

However, it seems possible to use a complex *full model* optimized with respect to predictive power if we develop a simplified *explanation model*, which approximates the predictions of the full model as well as possible. If the difference between both predictions is small we can explain the results of the full model by evaluating the simple structure of the explanation model. A user may directly see under which conditions a different recommendation would occur.

There was some research on the explanation of models in the neural network literature. Here the task was to represent the black-box model by a nearly equivalent rule system. A good overview is contained in the proceedings of a workshop [AD96]. Craven [Cra96] proposed to use tree models for the same task.

In 1997 we developed a new credit scoring procedure for the DSGV, a large group of German banks. We employed complex Bayesian decision trees[PK98] and bootstrap approaches to estimate the probability that an enterprise would repay a credit. These procedures generated several 100 different models with many thousand parameters whose average gives a mean probability of insolvency. This paper describes the results of a follow-on project funded by DSGV where we analyzed which type of explanation model is best suited to explain the predictions of the full model.

In section 2 we discuss criteria for selecting an expla-

nation model, both in terms of comprehensibility as well as prediction quality. ??? etc

2 Criteria for Selecting an Explanation Model

2.1 Notation

As full model for our credit scoring task we used different variants of *Bayesian models*

$$p(y = 1|\mathbf{x}, \theta) = f(\mathbf{x}, \theta) \tag{1}$$

which predict the *credit risk* for $\mathbf{x}$, i.e. the probability that a customer with feature vector $\mathbf{x}$ will not repay a credit ($y = 1$). The parameter vector θ completely determines the result. Instead of estimating an *'optimal'* parameter we use the posterior distribution of parameters given some data $\mathbf{y}, \mathbf{X}$ and the prior distribution $p(\theta)$

$$p(\theta|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \theta)p(\theta)}{\int p(\mathbf{y}|\mathbf{X}, \theta)p(\theta)d\theta} \tag{2}$$

The posterior distribution describes the plausibility of different parameters. It is approximated by the empirical distribution of a sample $\Theta = \{\theta_1, \ldots, \theta_B\}$, which is generated by Markov Chain Monte Carlo or related methods [PK98].

For a new input $\mathbf{x}_0$ we may calculate the prediction values $p(y = 1|\mathbf{x}_0, \theta_b)$ with respect to all $\theta_b \in \Theta$. These values follow the *predictive distribution* describing the uncertainty of the credit risk. If we state the consequences of decisions in terms of a loss function we should classify the case $\mathbf{x}_0$ such that the average expected loss is minimal. This implies that $\mathbf{x}_0$ is classified as insolvent if the mean prediction is larger than some threshold π_{dec}

$$E(y = 1|\mathbf{x}_0) \approx \frac{1}{B} \sum_b p(y = 1|\mathbf{x}_0, \theta_b) > \pi_{dec} \tag{3}$$

A change in the loss function (i.e. interest rates) merely leads to a shift in π_{dec}.

In this framework an *explanation model* can be considered as a simplified model $p(y = 1|\mathbf{x}, \psi) = g(\mathbf{x}, \psi)$, which on the one hand should be easy to understand, and on the other hand should approximate the predictions of the full model as good as possible. To keep the situation simple only an optimal model $\hat{\psi}$ is derived, e.g. by maximizing the posterior density (2) with respect to ψ. Criteria for comprehensibility and prediction quality are developed in the following sections.

2.2 Comprehensibility

The explanation of a classification or prediction model is based on the following lines of reasoning:

- The agreement of the results with common *factual rationalizations*. This is mainly based on a more or less elaborate intuitive knowledge about effects. For example a higher debt usually causes higher credit risk.
- Insight into the *internal "mechanics"* of the model. In this case the user can determine which mathematical operations in the model generate a specific prediction or classification. In addition the mathematical operations causing differences between two results may be identified.
- The *empirical validation* of results by simple statistics like counts and means. By counting, for example, the number of solvent and insolvent firms in a subpopulation the credit risk can be estimated in a transparent way, immediately evident to the user. This argumentation is familiar from opinion polls and sampling surveys.

Undoubtedly an explanation by factual rationalizations is most convincing. However intuitive knowledge about effects usually is of qualitative nature, and it is difficult to summarize contradicting trends to an overall result. In this case the other two aspects offer a good explanation of the results.

The internal mechanics of a model is comprehensible only for models with simple structural features:

Analysis of Subpopulations. Here a population is partitioned into disjoint subsets according to the feature values using simple decision criteria. Each of these subsets is characterized by a single prediction or a simple "sub-model". From the view of psychological research subsets are advantageous, as they divide complex decision problems into small, easy subproblems [Hog87, p.185].

If-Then Rules. A special type of local models are rules which make a statement concerning a subset of the feature space. These subsets, however, usually are not disjoint. Rules are especially evident, if the conclusions correspond to the common lines of reasoning in the application.

Linear Relations. The predictive behavior of linear models is especially simple to understand, as all changes are proportional. If the direction of changes corresponds to factual rationalizations, they are highly convincing.

Independent Variables. Even nonlinear models are easier to understand, if each feature adds an independent term to the result, i.e. interactions may be ignored.

Subsets of Variables. If we generate a prediction model using only a subset of variables, we may attribute the difference to the "full" prediction with all variables to the specific subset. If, for instance, the credit risk with respect to debt figures is higher than the credit risk with respect profit features then the debt figures somehow 'cause' the higher risk which would result with 'normal' adepts. Comparison of the results for different subsets gives a picture of the perhaps contradictory effects of the subsets in the specific case.

A prominent aspect for understanding a problem is the sheer number of factors. "As a rule of thumb, eight dimensions is plenty, and fifteen is too many [Hog87, p.185]. Therefore Explanation models should be as small as possible.

To compare the comprehensibility of different explanation models, we use mainly use the *number of parameters* of the explanation model. This criterion subsequently is increased or decreased according to the *transparency* (complexity of the internal mechanics of the model) as well as the *empirical validity*. Obviously the latter quantities are highly subjective and give only a rough indication of the relative performance of procedures.

In this paper we will consider the following *types of explanation models* for the credit scoring task:

- *Linear models* like linear discriminant analysis, linear regression and logit models.
- *Scorecards* which assign numeric scores to feature categories or to intervals of feature values. If the sum of these scores is above a threshold, the case is classified as insolvent. *Generalized additive models* [CH92, p.249ff] are a generalized version which automatically determine an optimal transformation of the feature values.
- *Decision trees*, which successively partition the input space according to the values of single features. Within the leaves the result is constant.
- *If-Then rules* of general shape, which are more flexible than decision trees but often more difficult to interpret.
- *Clustering algorithms* grouping the input feature values.
- *Subset-of-variables models* which indicate the effect of different feature groups on the credit risk.
- *Combinations* of different approaches.

2.3 Classification Similarity and Quality

It is the task of the explanation model to approximate the predicted credit risk $p(y = 1|\mathbf{x}, \theta)$ of the full model (1). As the loss functions vary only to a small extent, the threshold π_{dec} is restricted to a small range. Only in this range the value of the decision risk $p(y = 1|\mathbf{x}, \theta)$ affects the decision (c.f. (3)).

OIP: As main criterion for classification *dissimilarity* of the full model and the explanation model we use the percentage OIP of cases, where the credit risk predicted by the explanation model is *not* between the 5%-quantile and the 95%-quantile of the predictive distribution (section 2.1). Predictions within this interval are fairly well in accordance with the full model predictions. A low OIP value indicates a satisfactory correspondence between the full model and the explanation model.

DPWR: As an additional criterion we assess the *classification error* of the explanation model as a classification model of its own. We determine the percentage of rejected 'good' loans when bad loans are accepted with a fixed probability value. Because of disclosure regulations we only were allowed to publish the scaled difference DWPR of these numbers for the full and explanation model. A low DPWR-number indicates good quality.

3 Comparing Explanation Models in a Real World Credit Scoring Task

3.1 Credit Scoring Data and Baseline Classifications

Explanation models should imitate the behavior of the full model as good as possible. This should also hold in areas where only few training data observations are located. To improve this fit we created *synthetic data* by calculating the credit risk for 11000 unclassified genuine input vectors. In addition we derived minimal spanning trees with respect to Euclidean distances between input vectors and formed convex combinations between close neighbors. Together with the predicted credit risk this produced 50000 new plausible data points. Together with the 6000 original training instances this was used as training input for the explanation models. Each case was used twice for training with output 1 and 0 and weights equal to the credit risk and its complement.

Table 1 **Results for Stepwise Linear Logistic Regression.**

var set	no. of significant ... vars	interactions	param.	error DWPR %	dissim. OIP %
$\mathcal{M}_6$	4		5	25.6	13.7
	3	6 of 15	10	24.6	12.9
$\mathcal{M}_{17}$	10		11	22.8	13.2
	4	11 of 136	16	19.8	13.1
$\mathcal{M}_{39}$	25		26	17.2	12.1
	9	16 of 741	26	18.8	12.5

We investigated three different sets of input features selected by domain experts: The set $\mathcal{M}_6$ of six most relevant variables, the set $\mathcal{M}_{17}$ containing additional medium important variables and a comprehensive set $\mathcal{M}_{39}$. These and similar figures are discussed in [Uth97, p. 135ff]. We must not publish the variable names because of non-disclosure reasons.

3.2 Linear Logistic Models

For a given feature vector $\mathbf{x} = (x_1, \ldots, x_k)$ these procedures approximate the credit risk by [Bis95, p.82f]

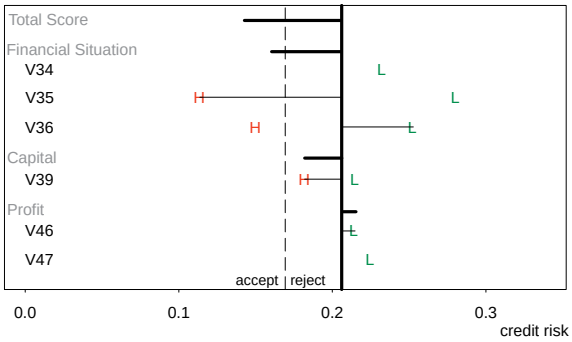
$$g(\mathbf{x}, \theta) = \frac{\exp(\eta)}{1 + \exp(\eta)} \qquad \eta = \theta_0 + \sum_{i=1}^{k} \theta_i x_i \tag{4}$$

This is a classical credit scoring model used in many analyses [Cam96]. It is closely related to linear discriminant analysis [Lus96] which currently is used by every second German bank for credit scoring [GG97].

Transparency: The coefficients θ_i indicate the quantitative and qualitative influence of attribute x_i on the credit risk. If it is positive the risk grows as x_i is increased. Psychological experiments show that linear relations are far simpler to comprehend than nonlinear relations [Hog87, p.129]. Often, however, the features x_i are strongly correlated and a negative parameter value θ_i is offset by some positive θ_j. This reduces the interpretability of parameters. In addition the parameters are determined by complex optimization procedures and their empirical validation, the relation to the data is not evident.

Classification Similarity: We used stepwise regression in an exploratory way to identify the most important parameters. Even for the large synthetic training set only four of the six most important variables get significant (see table 1). In comparison with later results the dissimilarity and the classification error are rather high. This is only marginally improved by using additional variables or interactions. In summary this indicates the presence of non-linearity in the data and in the full model. Therefore linear models are not useful for explaining credit risk.

Figure 1 **Graphical representation of a scorecard for six variables with up to three categories: low (L), medium (baseline) and high (H). The scores are the positive or negative differences to the baseline representing a_0. Only significant scores are shown.**



3.3 Scorecards

The starting point of the approach is that each feature has an additive effect on the output, which is independent of the level of the remaining features. We may then divide the domain of each feature into subsets, each of which gets a specific point *score*. The sum of these scores for a specific case is the overall score. For each of the k features in $\mathbf{x} = (\mathbf{x}_1, \ldots, x_k)$ suppose that m different subsets are defined. Then we need $m - 1$ indicators such that $x_{ij} = 1$ in the j-th subset and 0 elsewhere. The scorecard corresponds to

$$y = a_0 + \sum_{i=1}^{k} \sum_{j=1}^{m-1} a_{ij} x_{ij} \tag{5}$$

with scores a_0 and a_{ij}. The scores can be estimated by linear regression or similar methods from training data. The cut-off point is calibrated by some validation data, such that the loss function gets optimal. Often two cutoff points are identified with an intermediary class of 'questionable' cases. Scorecards are widely used in the financial industry [Asc95, Fos96, ZF96].

An example with six features is shown in figure 1. Each feature is divided into three intervals (low, medium, and high). To identify significant scores a stepwise regression [VR97, p.220] was performed. For the medium category no indicator was defined as its effects are subsumed by a_0 and depicted by the bold line. The scores for high (less than 33% percentile) and low (more than 67% percentile) values are denoted by characters 'L' and 'H' with the difference to the baseline indicating the score value a_{ij}. All scores add up to arrive at the final result. The procedures can be extended to define categories in an automated way and to maintain monotony relations

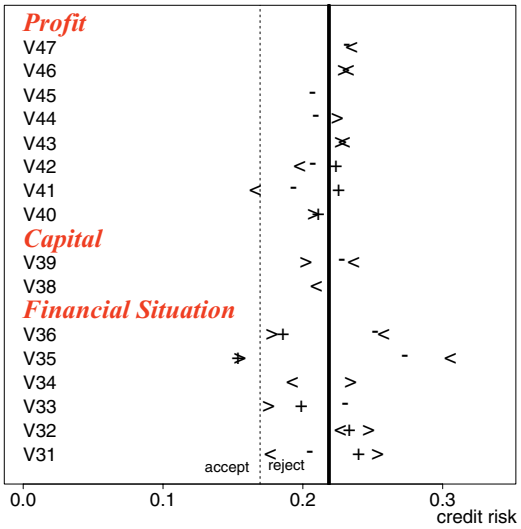


Figure 2 **Scores for $\mathcal{M}_{17}$ features using 5 categories denoted by very low (<), low (−), high (+) and very high (>).**

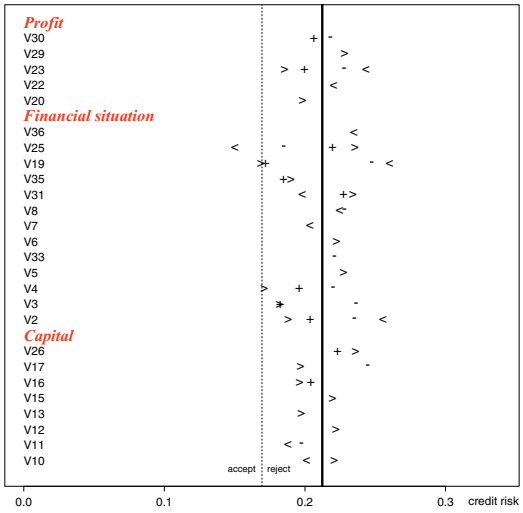


Figure 3 **Scores for $\mathcal{M}_{39}$ features using 5 categories denoted by very low (<), low (−), high (+) and very high (>).**

between categories [Thi88, p.221f]. Figures 2 and 3 show scorecards with the variables in $\mathcal{M}_{17}$ and $\mathcal{M}_{39}$. Again the scores were defined by percentiles of the features. Note that usually not all categories are significant and in a few cases the effects of categories are non-monotonic.

Transparency The transparency of decisions and the simplicity of the 'mechanics' of scorecards is very high. The user can determine immediately under which condition the decision would be different. Psychological research shows that the use of scores corresponds to our intuitions [Hog87, S.189]. In summary scorecards are good explanation models. In the case of many features, however, these are often highly correlated and the relative score values have high variance. Therefore the significance of a score must not be overestimated. It would help a lot if the correlation of scores could be graphically shown in a satisfactory way.

Table 2 **Results for Scorecard estimated by Stepwise Regression.** The features are divided into 3 or 5 categories with equal proportions of cases.

var set	no. of . . . categ.	signif. param.	error DWPR	dissim. OIP
$\mathcal{M}_6$	3	10	17.4	12.9
	5	20	15.5	12.4
$\mathcal{M}_{17}$	3	21	11.5	11.0
	5	40	9.9	10.2
$\mathcal{M}_{39}$	3	31	6.9	9.4
	5	53	5.9	8.2

We may split the input features into groups the according to their meaning. Then for a new case we may add up the scores of all features in a group to the score corresponding to that group. This on the one hand explains the result in terms of broader concepts. On the other hand the scores of usually highly correlated features are aggregated yielding summary scores which have lower variance. An example of such a scorecard with aggregated scores is shown in figure 9.

Classification Similarity: In spite of the simple model structure scorecards have a surprisingly good classification similarity and quality, as shown in table 2. With 39 features in $\mathcal{M}_{39}$ only 8.2% of the explanations are outside of the full model prediction interval, compared to an OIP-value of 12.1% for the linear model. This shows a marked improvement in terms of classification similarity. The classification error DWPR decreased even more from 17.2% to 5.9%. Note that there is a additional drop if we either increase the number of features considered as well as the number of categories within features. One reason for the better performance seems to be that scorecards can capture nonlinearities in the single variables by assigning arbitrary scores to the subsets. Another reason is probably that outliers have no effect in scorecards.

We investigated several generalizations of scorecards. First we analyzed scorecards with interactions by adding new features $z_{ij} = x_i x_j$ for each pair of features x_i and x_j. Subsequently the domain of z_{ij} was divided into three subsets of equal probability yielding two new indicator variables, which were added to the explanation model. Using stepwise regression as

before the large number of parameters in the resulting model were reduced. In no case these models had a better performance than scorecards without interactions. This is another indication that interactions are not very important for this domain.

Instead of assigning scores to subsets, which induces a step-function for each feature we used *generalized additive models*, which determine an optimal smooth transformation for each feature. The resulting model of the following form

$$p(y = 1|\mathbf{x}) = a_0 + \sum_{i=1}^{k} a_i\, h_i(x_i) \qquad (6)$$

may be determined by the back-fitting algorithm [CH92, S.249ff]. With variable set $\mathcal{M}_{17}$ the classification dissimilarity was 11%, which was no improvement compared to the simple score card. The same holds for the classification error which had a value of 11.5%. Therefore these models offer no improvement compared to the simple scorecard and in addition are less transparent.

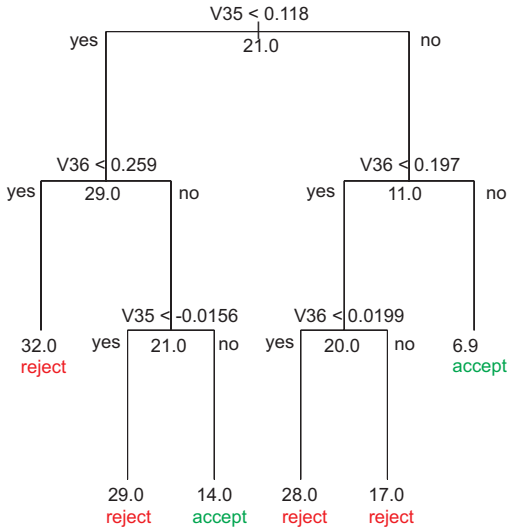


Figure 4 **Simple decision tree for credit scoring. The numbers are the scaled credit risk for the subsets. The decisions result from $\pi_{\mathbf{dec}} = 16.8$.**

3.4 Decision Trees

Decision trees are an established model in data mining. By successively dividing the input space according to the values of selected single features a partition of the input space into disjoint sets is generated. On each set the predicted value gets a fixed value. An example is show in figure 4. We used the tree algorithm of SPLUS [CH92, p.377ff] which has a complexity parameter `mindev`, which usually is determined by

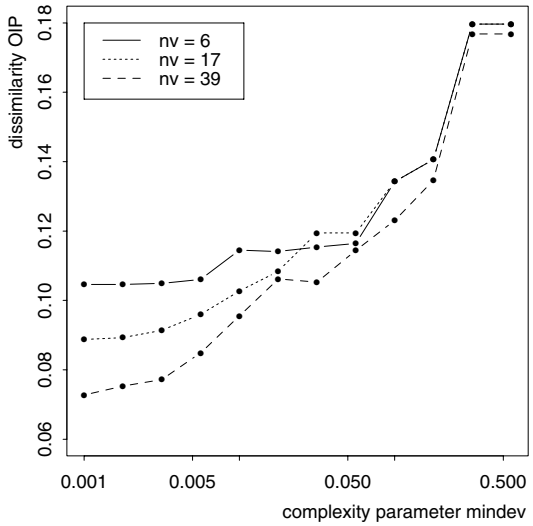


Figure 5 **Dissimilarity OIP in relation to the number `nv` of features and the complexity parameter `mindev`.**

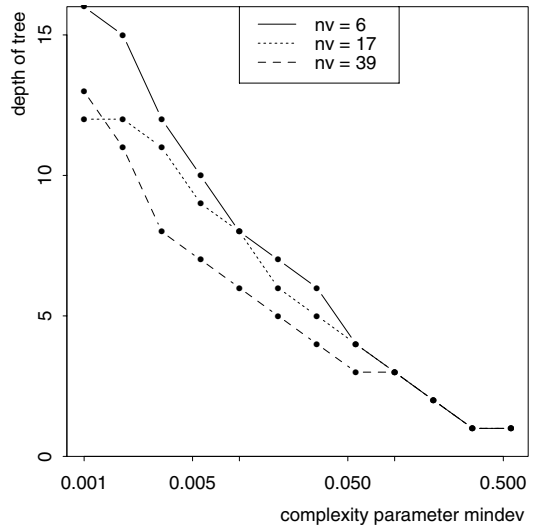


Figure 6 **The depth of resulting tree in relation to the number `nv` of features and the complexity parameter `mindev`.**

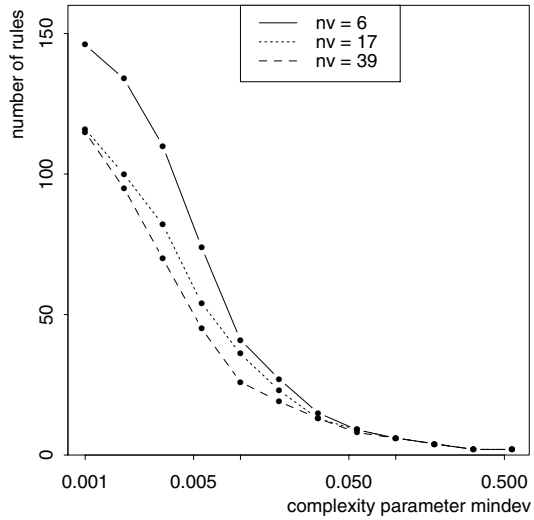


Figure 7 **Number of decision nodes (rules) in relation to the number `nv` of features and the complexity parameter `mindev`.**

cross-validation from the data. We changed it in small steps to get decision trees with desired complexity.

In spite of their simplicity decision trees have a valuable property: if they are large enough they can represent any functional relation arbitrary well. This is a marked difference to linear models or score cards, which have the inherent assumption of independent features. Nevertheless decision trees have an inherent representation bias, as they can reproduce relations especially well if they have a grid-like structure. A problem, however, are continuously growing relations which have to be approximated by a large number of steps. In the US decision trees are used frequently for credit scoring [RG94].

Transparency Decision trees have a double advantage in terms of transparency. First the internal mechanics of the method is simple as the credit risk within subsets defined by a few splits is constant. Second the credit risk itself is determined as the empirical frequency of 'bad' cases in this subset. If in addition the level of this credit risk corresponds to the intuitive knowledge of effects of the split variables, then there is also a factual rationalization of the model. This, however, has to be checked in detail.

As shown in figure 7 the number of rules (decision nodes) in the tree can be controlled by the complexity parameter mindev. Setting its value to zero generates rules which separate each different input vector. Of cause such large trees no longer give any explanation, as the user can only cope with a limited number, say 20 rules. An additional problem arises if there are repeated splits with respect to a single feature, as can be seen in figure 4 as it is more difficult to understand the interlocking definition of subsets in this case. An important aspect is the depth of the tree, which determines the number of conditions defining a subset. Figure 7 shows how it grows with falling complexity parameter. Obviously a user can understand only the joint effect of a small number of up to 5-8 conditions.

Table 3 **Results for Decision Trees of Varying Size for Different Sets of Features.**

variable set	no. of ... rules	classif. error DWPR	dissimilarity OIP
$\mathcal{M}_6$	27	20.8	11.4
	146	13.5	10.4
$\mathcal{M}_{17}$	23	19.2	10.8
	116	8.3	8.8
$\mathcal{M}_{39}$	26	10.1	9.5
	115	5.6	7.2

Classification Similarity The prediction dissimilarity ranges from 11.4% for six features and 26 rules to 7.2% for 39 inputs and 115 rules. This is roughly the range of figures for scorecards, especially if we take into account the number of free parameters. For instance a score card with 31 free parameters yields 9.4% while a tree with 26 rules reaches 9.5%. In terms of classification errors this tree is with 10.1% inferior to the scorecard with 6.9%. The figures 5 and 7 show the dissimilarity which is attainable with a specific number of rules. In summary the scorecards have slight advantages compared to decision trees.

3.5 Rule Based Models

We considered inference systems which use logic based rules of the form **IF** $A \wedge B \wedge C$ **THEN** D. Here A, B, C, D are logical propositions about the features, e.g $x_i \in [a, b]$. Many of these rules may be combined forming a rule system. A number of solution systems have been developed which derive the conclusions for a given set of facts (e.g. a new input case). American express uses a rule-based system for consumer credit scoring which has been developed by hand [Did95].

Logical rule systems are difficult to create and maintain as the rules have to be consistent. Therefore systems are required which automatically generate a rule system from data. We analyzed the performance of two such systems as generators for explanation models.

RULEX [AG96] tries to represent a feed-forward neural network by logical rules. The underlying network has bell-shaped activation functions and inputs with active links are used as antecedents. We applied the

procedure to our credit data. It turned out that the generated model had nearly twice as much classification errors than the full model. Therefore it was not acceptable for our purposes.

FOIL [Qui93] is another system which generates new logical relations for a collection of facts (i.e. examples and counterexamples of a class), which imply these facts and have a lower overall complexity. The procedure generated a large number of rules which were able to describe the insolvent cases quite well. If no rule applies a case can be considered as solvent. Because of the large number of rules this offers no explanation about 'why' the case is solvent. Because of this asymmetry we did not further consider this approach.

3.6 Dimension Reduction Approaches

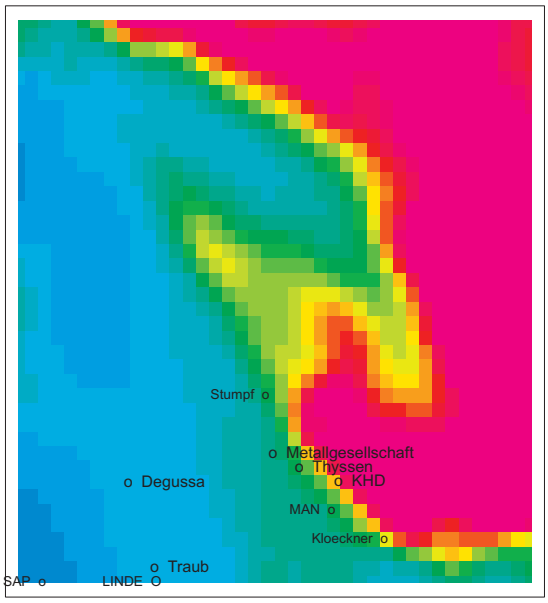


Figure 8 **Two-dimensional Kohonen-map using variables V35,V36, V39, V46 and V47.** The shades indicate the predicted credit risk from low (lower left) to intermediate (bright) and high (upper right).

The procedures project the input space into two or three dimensions such that the functional relation between inputs and outputs get clear. After experimenting with a number of methods we got the best results with two-dimensional Kohonen maps [Koh95]. The input vectors of our training sets are placed in a two-dimensional area such that vectors with a small Euclidean distance of the input vectors are close to each other in the area. In figure 8 such a map is shown where the credit risk of the cases predicted by the full model is indicated by shades. It turn out that the map is remarkably smooth showing the strong relation between credit risk and input features. A new case is corresponds to some point in the map corresponding to its distance to neighboring cases. It can be judged in relation to known cases of German enterprises added to the map. An 'explanation' then states that the case in question is similar to, say SAP. This sort of explanation is heuristic, but nevertheless max be valuable.

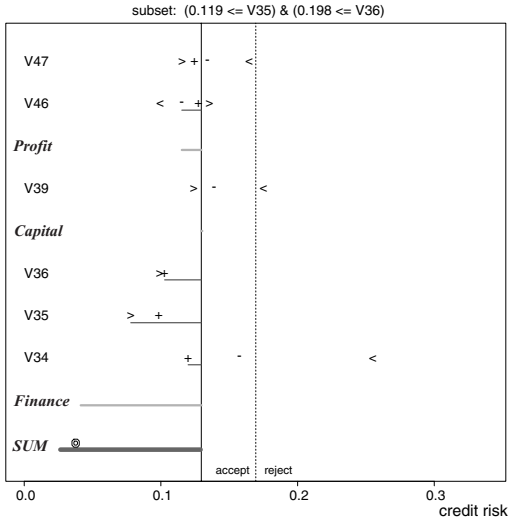


Figure 9 **Scorecard in a subset for the $\mathcal{M}_6$-features with five categories of equal mass. The symbols indicate the scores of the very low (<), low (-), high(+) and very high (>) category. Lines indicate the scores of an actual case.**

3.7 Explanation Models in Leaves of Decision Trees

A natural extension of the procedures is the combination of approaches. Decision trees are especially suitable as they do not assume that the features are independent but are able to approximate arbitrary relations. In addition they partition the input space into subsets where arbitrary alternative explanation models may be used. In this section we combine decision trees with the following 'leaf'-models:

- Scorecards
- One- and two-dimensional nonlinear loess models.

With respect to the combination of decision trees with scorecards we first developed an integrated approach which simultaneously determined the trees as well as the scorecards with the aim to reduce the overall error. It turned out that this algorithm didn't offer any advantages. Therefore we simply generated a tree of the desired size and subsequently estimated scorecards within the leaves using stepwise regression.

In figure 9 a scorecard within a subset is shown. Each feature was divided into 5 intervals covering 20% of the mass. The medium category is depicted by the bold line.The symbols indicate the scores of the very low (<), low (-), high(+) and very high (>) category, always as difference to the bold line. For a specific new case all scores add up to arrive at the final result. The horizontal lines if the figure indicate the scores of an actual case. We grouped the variables, e.g. V34-V36 form the group 'Finance' The score of a group is just the sum of the individual scores and is denoted by a line. The overall score for that case is the line close to 'SUM' yielding a scaled credit risk of about 0.03. The two circles indicate the actual prediction of the full model.

Transparency In our opinion a scorecard is easier to understand than a decision tree with the same number of parameters. As scorecard merely is the sum of independent feature scores. A decision tree usually develops highly interlocked subset definitions, whose multi-dimension relation is difficult to comprehend. On the other hand small decision trees are easy to understand. The combination of small trees and scorecards also is relatively transparent and on the other hand may be able to capture some interactions present in the data.

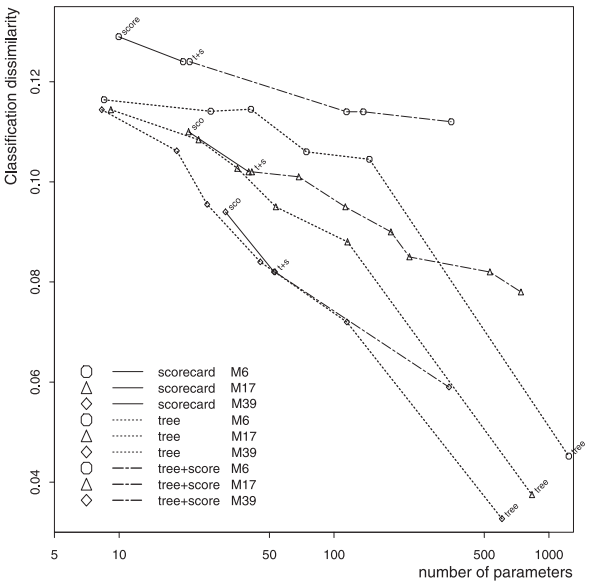


Figure 10 **Classification dissimilarity of scorecards, decision trees and scorecards in the leaves of decision trees in relation to the number of parameters.**

Classification Similarity The results for classification similarity and error of scorecards within leaves are collected in table 4. For some of the scorecards interactions were used. With respect to classification

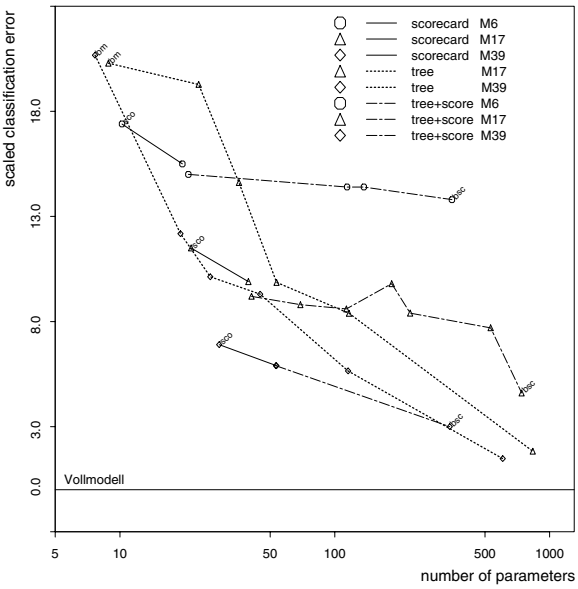


Figure 11 **Classification error of scorecards, decision trees and scorecards within decision trees in relation to the number of parameters.**

Table 4 **Results for Scorecards with 5 categories of equal mass within the leaves of a decision tree for different sets of inputs.**

var	number of		error	dissim.
set	leaves	param.	DWPR %	OIP %
$\mathcal{M}_6$	1	$21^{(*)}$	15.0	12.4
	6	114	14.4	11.4
	6	$137^{(*)}$	14.4	11.4
	27	351	13.8	11.2
$\mathcal{M}_{17}$	1	69	8.8	10.1
	1	$41^{(*)}$	9.2	10.2
	2	113	8.6	9.5
	4	184	9.8	9.0
	6	224	8.4	8.5
	15	$531^{(*)}$	7.7	8.2
	23	740	4.6	7.8
$\mathcal{M}_{39}$	6	343	3.0	5.9

$^{(*)}$: with 2-factor interactions

error the combination of scorecards within trees performs similar to score cards alone. This indicates that there are no marked interactions which can only be recovered by trees.

In figure 10 the dissimilarities of scorecards, trees and scorecards within trees are compared, always in relation of the number of parameters of the model. It turns out that for the six variables in $\mathcal{M}_6$ trees work better than scorecards and scorecards in leaves. For the 17 variables in $\mathcal{M}_{17}$ all tree approaches have similar performance up to about 40 parameters. For a larger number of parameters the trees have some advantages. Finally for the large set $\mathcal{M}_{39}$ of variables all three procedures are nearly identical. Taking into account that scorecards are easier to understand than pure trees this gives scorecards and scorecards within trees an advantage.

As a second model we used loess model within the leaves of trees. Loess is a nonparametric approximation approach which is generated by the superposition of many simple local models defined in a small subset [Cle93, p.93ff]. For each leaf subset we determined a number of loess models with respect to one or two variables and selected the best two or three. In case of a prediction these are shown to the user. Each alternative shows the prediction with respect to that variable. The form of the curve gives a good idea of the resulting changes in credit risk if the feature is modified. However, in terms of classification error and similarity this approach performed worse than decision trees or scorecards.

3.8 Subset-of-variables models

The credit risk may be predicted by a model which includes only the variables of a specific group, e.g. profit. Then the corresponding prediction entirely depends on the variables in this group. By comparing the prediction based on different variable groups we get a picture on the perhaps antagonistic effects of the groups. The model for this explanation is used as a black box. Therefore we may use the best model at hand, in our case the Bayesian procedure.

Figure 12 shows in its upper part the predicted density of the full model for a specific case. The classification is determined by the mean value of the credit risk (vertical line), which is below the decision threshold (dotted line), indicating that the case is classified as ‘solvent’. Below are the mean predictions of four subset-of-variables models corresponding to the groups called finance, capital, profit and liquidity. It is evident that the finance as well as the liquidity variables indicate a higher credit risk, whereas the capital variables and the profit variables tend to a lower risk.

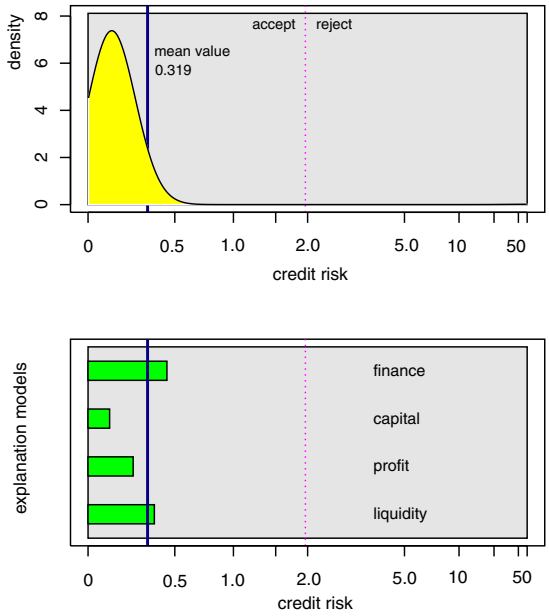


Figure 12 **Density of the scaled credit risk and mean predictions of the subset-of variables models for a new case.**

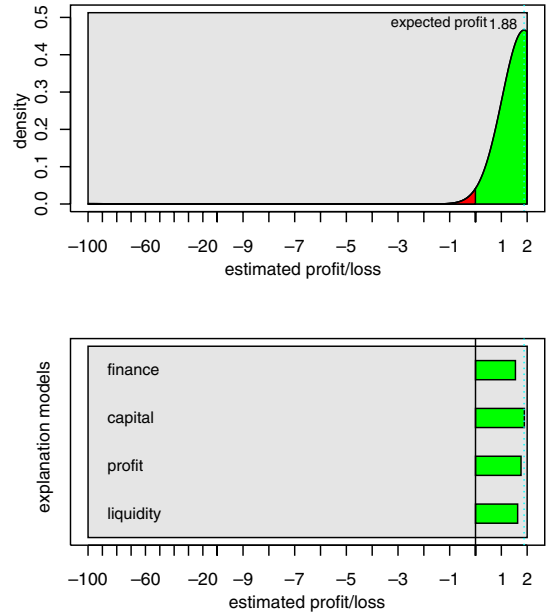


Figure 13 **Density of the projected profit/loss and corresponding mean predictions of the subset-of variable models for the same case.**

In that sense the height of the risk is 'caused' by finance and liquidity. The fact that the prediction is performed by an optimized model adapted to the training data gives a good empirical validation of the approach.

4 Summary

We judged the transparency of explanation models by the following factors: The number of parameters, the insight in the internal 'mechanics' of the model, and the empirical validation. The second main aspect was the similarity to the full model judged by the dissimilarity and the classification error.

As a mayor result there was always some dissimilarity between the full Bayesian model and the simple explanation models for our data. Obviously the interaction of many 100 different models can be captured only with difficulty. Beside of this fact decision trees, scorecards and scorecards within trees had roughly the same similarity to the full model.

Linear models in our opinion have a transparency similar to decision trees. For our data, it turned out that linear models have only limited modeling capacities. If the number of parameters exceeds a certain level scorecards seem to be easier to understand than decision trees, as the scores are added independently. They have, however, a lower empirical validity, as the scores are the result of complex matrix operations while trees are based on counts in subsets. Scorecards within the leaves of a tree are easier to understand than trees, as the latter require a far more detailed structure.

Especially valuable are subset-of-variables models. They ignore detailed insight into the mechanics take explanatory power only from the variables used as inputs. They are a very valuable explanatory approach and are applicable to complex situations. A prerequisite is, however, that the user forms sensible groups of input variables.

The relative ranking of procedures may be different for a new domain, exhibiting other nonlinearity pattern. It is therefore best to experiment with some of these models and use the best ones simultaneously. This can be achieved in a flexible way with the interactive graphical interfaces.

In the next phase of the project we will adapt the learning procedures for the explanation models in such a way that they directly approximate the predictive distribution of the full model, e.g. by using the variance of full-model predictions in generalized regression approaches. This should lead to a marked increase of similarity.

References

[AD96] R. Andrews and J. Diederich, editors. *Rules and Networks*, Brisbane, 1996. Queensland University of Technology, Neurocomputing Research Centre.

[AG96] R. Andrews and S. Geva. Rules and local function networks. In R. Andrews and J. Diederich, editors, *Rules and Networks*, pages 1–15, Brisbane, 1996. Queensland University of Technology, Neurocomputing Research Centre.

[Asc95] Latimer Asch. How the rma/fair, isaac credit-scoring model was built. *Journal of Commercial Lending*, 77:10–16, 1995.

[Bis95] C. Bishop. Real-time control of a tokamak plasma using neural networks. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, volume 7, pages 1007–1014. The MIT Press, 1995.

[Cam96] Steven V. Campbell. Workouts: Reorganize or liquidate? prediction model gives insights for small-business loans. *Commercial Lending Review*, 11:78–85, 1996.

[Cen00] Consumer Resonse Center. Federal Trade Comission, Washington DC 20580. http://www.ftc.gov/bcp/conline/pubs/credit/scoring.htm, 23.Feb. 2000.

[CH92] J. M. Chambers and T. J. Hastie. *Statistical Models in S*, volume Monograph 35. Wadsworth & Brooks, Pacific Grove, California, 1992.

[Cle93] W. S. Cleveland. *Visualizing Data*. Hobart Press, Summit, New Jersey, 1993.

[Cra96] M. W. Craven. *Extracting Comprehensible Models From Trained Neural Networks*. PhD thesis, University of Wisconsin - Madison, 1996.

[Did95] R. S. Didner. Intelligent systems at american express. In S. Goonatilake and P. Treleaven, editors, *Intelligent Systems for Finance and Business*, pages 111–134. Wiley, 1995.

[Fos96] Beverly Foster. Credit-scoring as a part of small business lending process development: A case study. *Journal of Lending & Credit Risk Management*, 79(4):61–66, 1996.

[GG97] Thomas Günther and Michael Grüning. Utilization of credit scoring procedures in pratice in credit institues (in german). Technical report, Fakultä für Wirtschaftswissenschaften, Technische Uni Dresden, 1997.

[Hog87] R. M. Hogarth. *Judgement and Choice*. Wiley, Chichester, 2nd edition, 1987.

[Koh95] T. Kohonen. Learning vector quantization. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 537–540. The MIT Press, Cambridge, MA., 1995.

[Lus96] Robert N. Lussier. A startup business success versus failure prediction model for the retail industry. *Mid-Atlantic Journal of Business*, 32(2):79–92, 1996.

[PK98] G. Paass and J. Kindermann. Bayesian classification trees with overlapping leaves applied to credit scoring. In X. Wu, R. Kotagiri, and K. B. Korb, editors, *Research and Development*

in Knowledge Discovery and Data Mining, pages 234–245. Springer Verlag, 1998.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[RG94] E. Rosenberg and A. Gleit. Quantitative methods in credit management: A survey. *Operations Research*, 42:589–613, 1994.

[Thi88] R. A. Thisted. *Elements of Statistical Computing*. Chapman and Hall, London, 1988.

[Uth97] C. Uthoff. *Erfolgsoptimale Kreditwürdigkeitsprüfung auf der Basis von Jahresabschlüssen und Wirtschaftsauskünften mit Künstlichen Neuronalen Netzen*. M & P Verlag für Wissenschaft und Forschung, Stuttgart, 1997.

[VR97] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-Plus*. Springer Verlag, New York, 2nd edition, 1997.

[ZF96] Mark Zmiewski and Beverly Foster. Credit-scoring speeds small business loan processing. *Journal of Lending & Credit Risk Management*, 79(3):42–47, 1996.

Learning from Project Experience - An Experience Factory Case Study

Carsten Tautz, Klaus-Dieter Althoff, Markus Nick

Fraunhofer Institute for Experimental Software Engineering (IESE)
Sauerwiesen 6
67661 Kaiserslautern, Germany
{tautz, althoff, nick}@iese.fhg.de

Abstract

The explicit management of project experience (lessons learned on project planning and execution) has become of strategic importance to many organizations. This paper describes how to set-up and run a repository, implemented with case-based reasoning technology, for sharing project experience using an industrial-strength case study. The case study shows that it is both effective and efficient for the information seeker to have an experience repository at his or her disposal. In addition, lessons learned regarding the organization and population of such an experience repository are reported. An outlook describes the potential of additionally planned exploitations of experiences learned from projects.

Introduction

Organization-wide sharing of qualitative experiences (i.e., lessons learned) has been recognized as a means to optimize business performance (Gresse von Wangenheim, von Wangenheim, and Barcia 1998). However, the successful set up of a lessons learned program is still a challenge. At our institute, we have started such a program. It is part of a more ambitious project called COIN (Corporate Information Network) which has the main objective of capturing, distributing, and exploiting the business knowledge of the institute. This paper describes how we proceeded in setting up the lessons learned program and reports on the experiences we gained thereby.

The paper is structured as follows. In the following section, principles and benefits of sharing project experience are discussed. The principles are interpreted as requirements for the structure, which is described in the next section. As the name "COIN" suggests, there are two sides to sharing experience: Utilizing and populating a repository. Hence, a separate section is devoted to the topic of populating. Based on the thus captured experiences, a vision for future exploitation possibilities is sketched. The paper ends with a summary and conclusion section.

Fraunhofer IESE´s Corporate Information Network (COIN)

To understand why the project "COIN" was launched, one has to take a look at the business goals and the situation of our institute: The institute was created to transfer innovative software engineering technologies into practice through applied research. Software engineering technologies are tailored to company-specific needs and are constantly evolving. During the transfer, different people in several departments and groups of the institute gain project experience. People learn what Fraunhofer IESE´s customers need, how the people can transfer and tailor software technologies more effectively and efficiently as well as how to improve the institute's business processes.

Expectations are that, with time, Fraunhofer IESE will transfer more technologies that are increasingly better tailored to companies' needs (using the same effort). To do so, the institute must find better ways to search for, reuse, and tailor technology effectively. Therefore, the sharing of experience is a very immediate and obvious problem. This fact led to the launch of "COIN".

Sharing of Project Experience

The effort required for setting up and running an experience transfer program (especially on the side of the project members for providing experience) forced us to start with a limited scope. In our situation, we basically had the choice between two types of experience to share: (a) experience on the applicability and limitations of software engineering technologies (product experience) or (b) experience on the planning and execution of projects (project experience). We decided to start with the latter, because the former was already covered partly by project reports and scientific publications.

Project experience is always related to a particular business process of an organization (in our case, the planning or execution of a project). In addition, it may be related to a particular situation in a particular project (situation-specific experience), a project in general (project-specific experience), a customer (customer-specific experience), or a topic (technology/competence-

specific experience). Alternatively, it may be related to a combination of projects, customers, and/or topics. We refer to anything an experience is related to as the context of the experience. Two types of contexts can be distinguished: The context in which the experience was gained (root context) and the context in which the experience is applicable (application context).

Consequently, it is insufficient to store experience without its context. Without its root context, the application context of the experience cannot be deduced systematically (see below). Without the application context, it is not clear under which circumstances the experience can be taken advantage of.

Over time, the application context can be deduced from the root context(s). A conservative approach is to use the root context as the application context. If the same experience is gained in several different root contexts, a single abstract application context can be constructed.

Experience can also be applied in contexts that are not explicitly documented. However, this decision usually requires a human expert. Since it is not possible to document all aspects of the root context (neither effort-wise nor intellectually since much of it is tacit knowledge), it is important to know a contact person that can provide further information. Hence, experience should never be stored anonymously.

Personalized experience also leads to an increased motivation of utilizing the stored experience. An experiment comparing the effectiveness and efficiency of the traditional human-based experience transfer ("ask your colleagues") and the repository-based experience transfer ("query the repository with personalized experience without asking the originators") clearly demonstrated this (Tautz et al. 2000). The experiential results showed that both approaches complement each other in terms of the project experience found. For example, the effectiveness of the repository-based approach is improved by the human-based approach by at least 50% on average with an error probability of less than 0.1%. Here, the improvement is measured in the number of experience items found by the human-based approach but not by the repository-based approach divided by the number of useful experience items found by the repository-based approach. In addition, the participants of the experiment were asked which of the two approaches they would apply in "real" projects. 28 out of 29 participants answered "both". Many participants added they would use the repository-based approach first to identify interview partners for the human-based approach.

Benefits of Sharing Explicitly Stored Experience

So far we assumed that experience should be stored explicitly (at least partly) to make it available to others. However, we have not discussed the alternative: Set up and use an expert network. For this purpose, we originally conducted the experiment (Tautz et al. 2000).

The results show that the repository-based approach is more efficient than the human-based approach for the seeker of project experience on a statistically significant level of well below 1%. In the experiment, the repository-based approach took 12.5 minutes or less to retrieve one useful experience item in 90% of the cases. In 50% of the cases, it took 3.8 minutes or less. In contrast, the human-based approach took 18.5 and 10 minutes for 90% and 50% of the cases respectively. These numbers are based on the effort needed by the information seeker to get the experience items.

If the availability of the experts is taken into account, the repository-based approach is far more efficient. In the experiment, it took 29.3 hours or less in 90% of the cases and 4.6 hours or less in 50% of the cases to get a useful experience item using the human-based approach. Using the repository-based approach, it took only 12.5 minutes and 3.9 minutes respectively.

In addition, the repository-based approach improves the effectiveness of the human-based approach by at least 50% on average with an error probability of less than 0.5%.

Organizing a Best Practice Repository

Project experience (lessons learned) can take on different forms (Birk and Tautz 1998). The two most important forms are observations and guidelines. Observations are facts that are of interest to future projects, often expressing some baseline (e.g., "it took 10% of the total effort to manage the project") or some positive effect (e.g., "the customer was happy because we provided him a ready-to-use tutorial"). Guidelines are recommendations how a particular business process should be performed. For example, a guideline could be the following: "Interact with the customer frequently, at least twice a month." In contrast to observations, a guideline is always based on some problem that has occurred in the past (e.g., "the expectations of the customer were not met."). In this way, relevant guidelines are restricted: Only guidelines that are really needed (to avoid the recurrence of a problem) are stored.

Another form of a lesson learned is the sequence of immediate countermeasures taken by a project team in response to a recognized problem. We will refer to such a sequence of countermeasures as a *correction*. While a guideline aims at preventing a problem from occurring in the first place, a correction is applied *after* a problem has already occurred.

To store and retrieve the various kinds of qexperience, we use INTERESTS (intelligent retrieval and storage system; Althoff, Tautz, and Bomarius 2000). INTERESTS consists of three major parts (Tautz 2000): Application tools for recording and updating experience over the web, an Experience Base Server for synchronizing access to the experience base, and CBR-Works from tec:inno GmbH, Germany. The latter is a case-based reasoning tool allowing similarity-based, context-sensitive retrieval. Case-based reasoning has been recognized as a suitable technology for implementing knowledge management applications (Gresse von Wangenheim and Tautz 1999, Göker and Roth-Berghofer 1999). The various experience forms are

represented as different case concepts. We defined the following concepts: Observation, guideline, problem, correction, business process, and project. The latter two concepts are used to store the root and application context of the experience. Semantic relationships between cases are represented by references. For example, an observation references the business process for which it is relevant and the project in which it was gained (root context). All cases are represented by attribute value pairs. Values can take on the special value "don't care." This allows us to store *abstract* contexts. For example, a project case with all attribute values set to "don't care" except for the attribute "customer" represents a customer context. Experiences referencing this project case are thus customer-specific. Similarly, topic/competence-specific experience can be represented.

Using this structure, the state-of-the-practice is represented by the descriptions of the business processes, which are complemented by context-specific observations, guidelines, and corrections. The similarity-based querying facility of INTERESTS/CBR-Works allows finding experience that was captured in contexts similar to a context at hand. Thus, potentially applicable experiences are identified (even if the application context has not been generalized from the root context yet). The personalized experience enables a project member to ask the original experience provider for more detailed information if necessary.

Figure 1 shows a (simplified) exemplary guideline with its context. The example indicates that not all experience is stored explicitly. For instance, the fact that Spearmint™ is an innovative process modeling tool developed in-house is not stored because it is known to everybody at the institute. The guideline is applicable whenever a project is set up for projects similar to "CAP" (application context). The root context is given by the problem underlying the guideline. If a project manager (responsible for setting up a project) is in doubt whether to apply the guideline, he or she can look at the underlying problem and decide whether the problem might also occur in the project at hand. Even if the guideline is not applied and the problem recurs (later in the project), the stored correction will probably lead to a quick solution. The project depicted in Figure 1 acts as the root context for the problem. It also further restricts the application context of the guideline (see dotted line). A more abstract project description may be used as the application context for the shown guideline (only "Spearmint™" is relevant for the applicability of the guideline). In fact, for the customer X the guideline needs not to be applied, because the answer is already known. The description of a guideline for customer X should state "do not use Spearmint™ for X".

Populating the Case Base

Experience needs to be explicated and stored before the repository-based approach can be applied. To do so requires the performance of the following steps (Althoff et al. 1999):

- Collect: In a first step, the experience must be collected. In our case, this was done using project analysis interviews (Figure 2). At our institute, such interviews are conducted either at the end of a project (Collier, DeMarco, and Fearey 1996) or — in case the project has a duration of more than nine months — periodically, that is, every six months. The interview results are documented as *project analysis reports* (PARs) (Figure 2). A PAR contains an updated characterization of the project, things to watch out for in similar projects, things that went well, and things that the interviewed project team would do differently if it had to do the same project again.
- Store: In the next step, the collected experience is stored

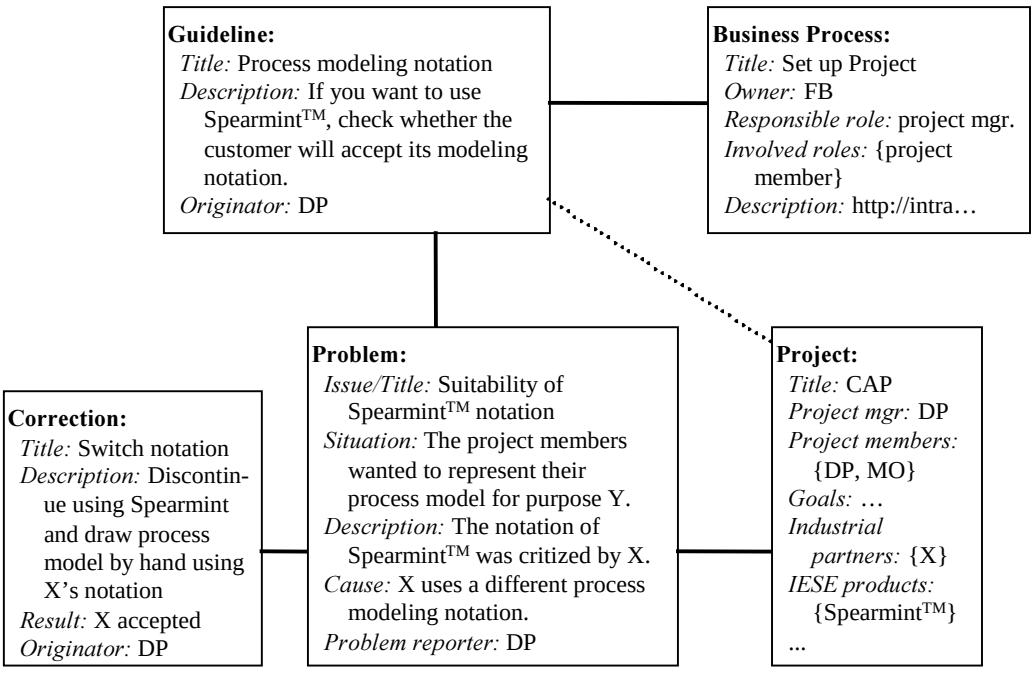


Figure 1: Excerpt of the case base for an exemplary guideline

by copying it into the repository, splitting the experience into reusable parts (see Figure 3), and initially characterizing each reusable part. In our case, the PARs are split into individual cases: The project characterization in the case base is updated; problems, corrections, guidelines, and observations are extracted out of the PARs and entered into the case base (Figures 2 and 3).

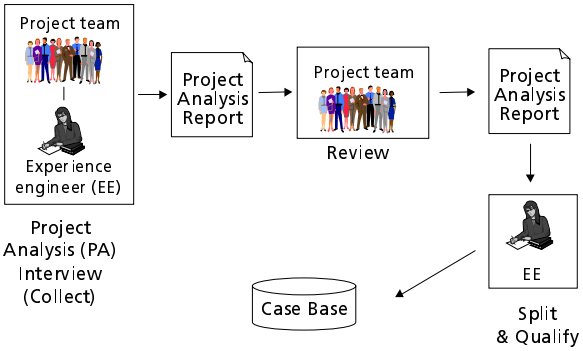


Figure 2: Collecting cases through project analysis

- Qualify: Each case is qualified by analyzing its quality (e.g., its comprehensibility) and checking whether a similar case is already stored in the case base. If a similar case is already stored, the new case may be rejected, be merged with the already stored case — possibly generalizing its context —, or replace the stored case.
- Publish: After the new experience has been qualified, it is made available for retrieval, thus enabling the sharing of the new experience.
- Inform: Finally, everybody who may be interested in the new experience is informed.

At our institute, the COIN team is responsible for populating the case base. This includes making the appointment for the interview, conducting the interview, writing the PAR, and all follow-up activities from storing to informing. Such a dedicated team for populating is necessary, because providing experience to future projects is not one of the (prime) objectives of a project (Basili, Caldiera, and Rombach 1994). Therefore, it cannot be expected that project teams provide experience on a voluntary basis. In fact, many of the interviewees said they are willing to participate in future interviews, but would not organize such a meeting on their own.

Although the population seems quite effort-intensive at first, it is important to recognize that only a systematic process guarantees high quality experience. In return, high quality experience encourages its usage, because it is perceived as useful. Thus, experience returned as the result of a query should exhibit a minimal quality.

At Fraunhofer IESE, interviews take at most two hours. 83% of the interviews are one hour long or shorter. The total effort involved on side of the COIN team is approximately four person hours per interview, whereas a project member spends only about 1.7 hours per project and interview including preparation for the interview and proofreading the PAR. This effort is negligible for projects running several months.

The particular process of capturing experience (as described above) also has the positive side effect that most project members view the collection step already as being beneficial (without the utilization of the repository!). During the interviews, the project members learn how their colleagues view(ed) the project and thus have a chance to reflect on the project — something that is rarely done during everyday business. Consequently, they get feedback they would not have gotten otherwise. In addition, they learn what to look out for in future projects. This is reflected by the results of the survey we conducted. In this survey, 34 project members were asked to rate the importance of project analysis interviews. 60% responded with “very important”, while 31% responded with “important”. Only 2% answered with “unimportant”.

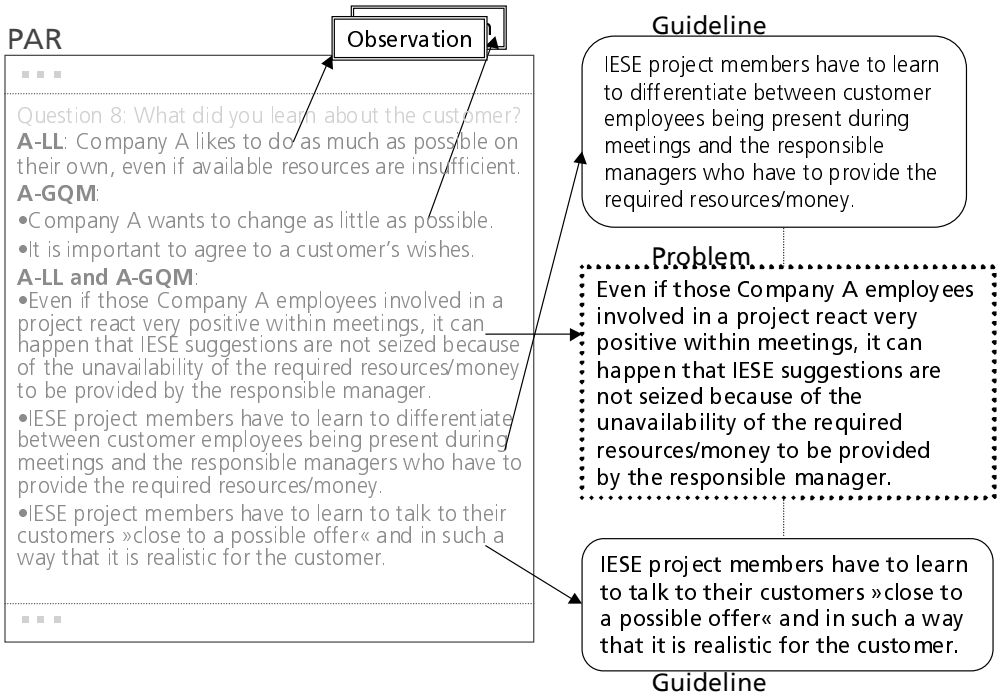


Figure 3: Splitting the project analysis report into cases

Envisioned Usage of Lessons Learned

Up to now, the sharing of project experience has been the focus of COIN. However, the potential of the captured project experiences goes far beyond that. This section outlines future usage scenarios of the captured observations, guidelines, and problems.

Organizational Problem Solving

During the population of the case base, critical or suboptimal project situations are stored as problems in the case base. A project-independent improvement squad meets periodically (e.g., once a month), scans the problem list, and assigns so-called *creativity teams* to solve clusters of similar problems. In this way, solutions that are optimal across the institute can be developed as opposed to the locally optimized corrections developed within the projects. Recurring problems can be avoided by providing guidelines or by changing the description of business processes.

In the context of COIN, we developed an improvement process with the following outstanding characteristics:

- Identification of small problems: Due to the active capturing of problems during the project analyses, a constant supply of problems is guaranteed. Without such an active "problem collection procedure", only serious problems that deeply move project members would be reported. In the past, problems that make project members feel uneasy (but are not serious with respect to project results) were typically not reported.
- Effective problem solving: Due to the systematic problem-solving process, problems are not forgotten (or ignored). Problem reporters are kept informed at every stage of the problem-solving process. This shows employees that their problems are taken seriously and thus encourages problem reporting.
- Efficient problem-solving: The effort to solve a problem depends on the complexity of the problem. A typing error is corrected by a single person, more complex problems are discussed by the improvement squad and solved by an individual person, whereas real complex problems are solved by an assigned creativity team.

This improvement process will be put into place mid 2000.

Consolidation of Project Experience

As already outlined in the beginning of this paper, the context of project experience can and should be generalized. Here further machine learning techniques maybe a very helpful support, especially from the perspective of looking at all observations, guidelines, and problems related to a particular business process. Interesting questions to be answered here are, for instance, what percentage of projects were completed on time within budget? Or, what parts of the process description are hard to comprehend?

Context-specific guidelines for problems, which have occurred often in various contexts, may be "upgraded" to general guidelines and thus become part of the process description.

The performance of such analyses requires a considerable number of stored lessons learned. For our institute, such a cross-project analysis is planned in the second half of the year. In addition, the kind of mentioned analysis tasks are related to the issues of evaluating the case base according to the underlying goals and maintaining it (Nick et al. 1999, Nick & Althoff 2000, Althoff & Nick 2000).

Roll-Out - The Need for More Technological Support

The management of project experience is of prime importance to our institute (project planning and execution are knowledge-intensive processes). However, the experience management processes described in this paper can (and will) also be applied for business processes other than project planning and execution.

The dealing with these various kinds of experiences in the processes of capturing and extracting as well as analyzing and evolving will require further technological support. CBR technology could, for instance, play the role of an intelligent platform for (other) machine learning procedures. Such techniques should support the learning of case structure/contents from databases and/or textual descriptions (e.g., textual CBR, text mining) or the learning of user-adapted access structures (n-step CBR: Weibelzahl & Weber 1999, Bartlmae 1999, Richter 2000).

Conclusion

This paper describes how project experience (lessons learned about project planning and execution) can be shared across projects using a case-based reasoning approach. One of the fundamental underlying principles of our approach is that project experience is never stored without its context. Case-based reasoning enables us to retrieve experiences that were captured in a context similar to the one at hand. Experiences on implementing such a lessons learned program at Fraunhofer IESE are reported. On this basis, scenarios for future exploitations of the captured experiences are presented and discussed.

The case study described in this paper must be seen as a first step towards a comprehensive experience management program. Such a comprehensive experience management should not only include the management of process experience (a generalization of project experience to all business processes). Instead, the management (i.e., systematic definition and capturing) of products, competencies, and skills available at the institute should be an integral part of comprehensive experience management. Currently, we are developing an infrastructure to support this kind of experience management.

References

Althoff, K.-D. (2000). Case-Based Reasoning. Submitted to S. K. Chang (ed.), *Handbook of Software Engineering and Knowledge Engineering*

Althoff, K.-D.; Birk, A.; Hartkopf, S.; Müller, W.; Nick, M.; Surmann, D; and Tautz, C. 1999. Managing Software Engineering Experience for Comprehensive Reuse. *Proceedings of the Eleventh Conference on Software Engineering and Knowledge Engineering*, Kaiserslautern, Germany, June 1999. Skokie, Illinois, USA: Knowledge Systems Institute.

Althoff, K.-D.; Bomarius, F.; and Tautz, C. 2000. Knowledge Management for Building Learning Software Organizations. *Information Systems Frontiers*. To appear.

Althoff, K.-D. & Nick, M. (2000). *Evaluating Case-Based Reasoning Systems.* Springer Verlag (to appear)

Basili, V.R.; Caldiera, G.; and Rombach, D. Experience Factory. In Marciniak, J.J. ed. 1994, *Encyclopedia of Software Engineering*, vol 1, 469–476. John Wiley & Sons.

Bartlmae, K. (1999). An Experience Factory Approach for Data Mining. In Proc. LWA-1999, University of Magdeburg, Germany, 5-14

Birk, A. and Tautz, C. 1998. Knowledge Management of Software Engineering Lessons Learned. *Proceedings of the Tenth Conference on Software Engineering and Knowledge Engineering*, San Francisco Bay, USA. Skokie, Illinois, USA: Knowledge Systems Institute.

Collier, B.; DeMarco, T.; and Fearey, P. 1996. A Defined Process for Project Postmortem Review. *IEEE Software* 13(4): 65–72.

Funk , P., Minor, M., Roth-Berghofer, T. & Wilson, D. (eds.) (2000*). Flexible Strategies for Maintaining Knowledge Containers*. Proc. of a Workshop at the 14th European Conference on Artificial Intelligence (ECAI 2000)

Göker, M. and Roth-Berghofer, T. 1999. Workshop on "The Integration of Case-Based Reasoning in Business Processes". In: *Proceedings of the ICCBR '99 Workshops.* Technical Report, LSA-99-03E, Department of Computer Science, University of Kaiserslautern, Germany: Centre for Learning Systems and Applications.

Gresse von Wangenheim, C. and Tautz, C. 1999. Workshop on "Practical Case-Based Reasoning Strategies for Building and Maintaining Corporate Memories". In: *Proceedings of the ICCBR '99 Workshops.* Technical Report, LSA-99-03E, Department of Computer Science, University of Kaiserslautern, Germany: Centre for Learning Systems and Applications.

Gresse von Wangenheim, C.; von Wangenheim, A.; and Barcia, R.M. 1998. Case-based reuse of software engineering measurement plans. *Proceedings of the Tenth Conference on Software Engineering and Knowledge Engineering*, San Francisco Bay, USA. Skokie, Illinois, USA: Knowledge Systems Institute.

Nick, M. & Althoff, K.-D. (2000). Systematic Evaluation and Maintenance of Experience Bases. In *Funk et al. (2000)*

Nick, M., Althoff, K.-D. & Tautz, C. (1999). Facilitating the Practical Evaluation of Organizational Memories Using the Goal-Question-Metric Technique. Proc. Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW'99), Banff, Oct. 16-21

Richter, M. M. (2000). Personal communication

Tautz, C.; Althoff, K.-D.; Nick, M.; and Ochs, M. 2000. An Empirical Investigation on the Effectiveness and Efficiency of Utilizing Repositories for Qualitative Experience. IESE-Report No. 016.00/E, Fraunhofer IESE, Kaiserslautern, Germany.

Tautz, C. 2000. *Customizing Software Engineering Experience Management Systems to Organizational Needs.* Ph. D. diss., Dept. of Computer Science, University of Kaiserslautern, Germany. To appear.

Weibelzahl, S. & Weber, G. (1999). Benutzermodellierung von Kundenwünschen durch Fallbasiertes Schließen. . In Proc. LWA-1999, University of Magdeburg, Germany, 295-300

tigate these issues.

8 Conclusions

In this paper, we presented Bayesian logic programs as an intuitive and simple extension of Bayesian nets to first-order logic. Given Prolog as a basis, Bayesian logic programs can easily be interpreted using a variant of a standard meta-interpreter. We also indicated parallels to existing algorithms for learning the numeric entries in the CPTs and gave some promising suggestions for the computer-supported specification of the logical component.

9 Acknowledgements

We would like to thank Daphne Koller, Manfred Jaeger, Peter Flach and James Cussens for discussions and encouragement.

References

[1] H. Blockeel and L. De Raedt. ISSID : an interactive system for database design. *Applied Artificial Intelligence*, 12(5):385–420, 1998.

[2] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Contect-specific independence in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-1996)*, 1996.

[3] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transaction on Knowledge and Data Engineering*, 1999. (to appear).

[4] L. De Raedt and L. Deshape. Clausal discovery. *Machine Learning*, (26):99–146, 1997.

[5] P. Haddawy. An overview of some recent developments on Bayesian problem solving techniques. *AI Magazine - Special Issue on Uncertainty in AI*, Summer 1999. (to appear).

[6] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, March 1995.

[7] M. Jaeger. Relational Bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-1997)*, 1997.

[8] K. Kersting. Bayes'sche-logische Programme. Master's thesis, Albert-Ludwigs-University, Freiburg, Germany, 2000.

[9] K. Kersting and L. De Raedt. Bayesian Logic Programs. In J. Cussens and A. Frisch, editors, *Work-in-Progress Reports of the Tenth International Conference on Inductive Logic Programming (ILP -2000)*, London,U.K, 2000.

[10] K. Kersting and L. De Raedt. Bayesian Logic Programs. In F. Furukawa, S. Muggleton, D. Michie, and L. De Raedt, editors, *Proceedings of the Seventeenth Machine Intelligence Workshop*, Bury St. Edmunds, Suffolk, U.K., 2000.

[11] K. Kersting, L. De Raedt, and S. Kramer. Interpreting Bayesian Logic Programs. In L. Getoor and D. Jensen, editors, *Working Notes of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data (SRL)*, Austin, Texas, July 2000.

[12] D. Koller. Probabilistic relational models. In S. Dzeroski and P. Flach, editors, *Proceedings of Ninth International Workschop on Inductive Logic Programming (ILP-1999)*, number 1634 in LNAI, Bled, Slovenia, June 1999. Springer.

[13] D. Koller and A. Pfeffer. Learning probabilities for noisy first-order rules. In *Proceedings of the Fifteenth Joint Conference on Artificial Intelligence*, pages 1316–1321, August 1997.

[14] J. W. Lloyd. *Foundation of Logic Programming*. Springer, Berlin, 2. edition, 1989.

[15] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 1994.

[16] L. Ngo and P. Haddawy. Answering queries form context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.

[17] J. Pearl. *Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2. edition, 1991.

[18] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1995.

[19] L. Sterling and E. Shapiro. *The Art of Prolog: Advanced Programming Techniques*. The MIT Press, 1986.

Bayesian Logic Programs*

Kristian Kersting and Luc De Raedt
Institute for Computer Science, Machine Learning Lab
Albert-Ludwigs-University, Georges-Köhler-Allee, Gebäude 079,
D-79085 Freiburg i. Brg., Germany
{kersting, deraedt}@informatik.uni-freiburg.de

Abstract

Various proposals for combining first order logic with Bayesian nets exist. We introduce the formalism of Bayesian logic programs, which is basically a simplification and reformulation of Ngo and Haddawys probabilistic logic programs. However, Bayesian logic programs are sufficiently powerful to represent essentially the same knowledge in a more elegant manner. The elegance is illustrated by the fact that they can represent both Bayesian nets and definite clause programs (as in "pure" Prolog) and that their kernel in Prolog is actually an adaptation of an usual Prolog meta-interpreter.

1 Introduction

A Bayesian net [17] specifies a probability distribution over a fixed set of random variables. As such, Bayesian nets essentially provide an elegant probabilistic extension of propositional logic. However, the limitations of propositional logic, which Bayesian nets inherit, are well-known. These limitations motivated the development of knowledge representation mechanisms employing first order logic, such as e.g. in logic programming and Prolog. In this context, it is no surprise that various researchers have proposed various first order extensions of Bayesian nets: e.g. probabilistic logic programs [16], relational Bayesian nets [7] and probabilistic relational models [12]. Many of these techniques employ the notion of *Knowledge-Based Model Construction* [5] (KBMC), where first-order rules with associated uncertainty parameters are used as a basis for generating Bayesian nets for particular queries.

We tried to identify a formalism that is as simple as possible. While introducing Bayesian logic programming we employed one key design principle. The principle states that the resulting formalism should be as close as possible to both Bayesian nets and to some well-founded first order logic knowledge representation mechanism, in our case, "pure" Prolog programs. Any formalism designed according to this principle should be easily accessible and usable by researchers in both communities.

This paper does not explicitly cover how to learn Bayesian logic programs, but effective representation and inference is the first step toward effective learning and is still a major issue in this research area.

The paper is laid out as follows. The next three sections present the authors' solution: Bayesian logic programs. Section 4 reveals their relations to Bayesian nets and "pure" Prolog programs. Section 5 shows a kernel implementation of Bayesian logic programs in Prolog. Before concluding this paper we give in Section 7 suggestions for learning Bayesian logic programs.

We assume some familiarity with Prolog or logic programming (see e.g. [19]) as well as with Bayesian nets (see e.g. [18]).

2 Bayesian logic programs

Bayesian logic programs consist of two components. The first component is the logical one. It consists of a set of Bayesian clauses (cf. below) which captures the qualitative structure of the domain and is based on"pure" Prolog. The second component is the quantitative one. It encodes the quantitative information about the domain and employs – as in Bayesian nets – the notions of a conditional probability table (CPT) and a combining rule.

A *Bayesian predicate* is a predicate r to which a finite domain D_r is associated. We define a *Bayesian definite clause* as an expression of the form $A \mid A_1, \ldots, A_n$ where the $A, A_1, \ldots, A_n$ are atoms and

* Slightly different versions of the paper appear as technical report [11] of the AAAI-2000 workshop on Learning Statistical Models from Relational Data (SRL), as work-in-progress report of the Tenth International Conference on Inductive Logic Programming (ILP-2000) [9] and in the Proceedings of the Seventeenth Machine Intelligence Workshop2000 (MI-17) [10].

all variables are (implicitly) universally quantified. When writing down Bayesian definite clauses, we will closely follow Prolog notation (with the exception that Prolog's `:-` is replaced by `|`). So, variables start with a capital, constant and functor symbols start with a lowercase character. The main difference between Bayesian and classical clauses is that Bayesian atoms represent classes of similar random variables. More precisely, each ground atom in a Bayesian logic program represents a random variable. Each random variable can take on various possible values from the (finite) domain D_r of the corresponding Bayesian predicate r. In any state of the world, a random variable takes exactly one value. E.g., we paraphrase that James' house is not burglarized with $burglary(james) = false$. Therefore, a logical predicate r is a special case of a Bayesian one with $D_r = \{true, false\}$. An example of a Bayesian definite clause inspired on [16] is `burglary(X) | neighborhood(X).` where the domains are $D_{burglary} = \{true, false\}$ and $D_{neighbourhood} = \{bad, average, good\}$. Roughly speaking, a Bayesian definite clause specifies that for each substitution θ that grounds the clause the random variable $A\theta$ depends on $A_1\theta, \ldots, A_n\theta$. For instance, let $\theta = \{X \leftarrow james\}$, then the random variable $burglary(james)$ depends on $neighbourhood(james)$.

As for Bayesian nets there is a table of conditional probabilities associated to each Bayesian definite clause[1].

$neighbourhood(X)$	$burglary(X)$ *true*	$burglary(X)$ *false*
bad	0.6	0.4
average	0.4	0.6
good	0.3	0.7

The CPT specifies our knowledge about the conditional probability distribution[2] $\mathbf{P}(A\theta \mid A_1\theta, \ldots, A_n\theta)$ for every ground instance θ of the clause. We assume total CPTs, i.e. for each tuple of values $\mathbf{u} \in D_{A_1} \times \ldots \times D_{A_n}$ the CPT specifies a distribution $P(D_A \mid \mathbf{u})$. For this reason we write $\mathbf{P}(A \mid A_1, \ldots, A_n)$ to denote the CPT associated to the Bayesian clause $A \mid A_1, \ldots, A_n$. For instance, the above Bayesian definite clause and CPT together imply that $P(burglary(james) = true \mid neighbourhood(james) = bad) = 0.6$. Each Bayesian predicate is defined by a set of definite Bayesian clauses, e.g.

```
alarm(X) | burglary(X).
alarm(X) | tornado(X).
```

[1] In the examples, we use a naive representation as a table, because it is the simplest representation. We stress, however, that other representations are possible and known [2].

[2] We denote a single probability with P and a distribution with $\mathbf{P}$.

If a ground atom A is influenced directly (see below) only by the ground atoms of the body of one ground instance of one clause, then the associated CPT specified a conditional probability distribution over A given the atoms of the body. But, if there are more than one different ground instances of rules which all have A as head, we have multiple conditional probability distribution over A – in particular this is the normal situation if a Bayesian atom is defined by several clauses. E.g., given the clauses for *alarm*, the random variable $alarm(james)$ depends on both $burglary(james)$ and $tornado(james)$. However, the CPT for *alarm* do not specify $\mathbf{P}(alarm(james) \mid burglary(james), tornado(james))$. The standard solution to obtain the desired probability distribution from the given CPTs is to use a so called combining rule. Theoretically speaking, a combining rule is any algorithm which maps every finite set of CPTs $\{\mathbf{P}(A \mid A_{i1}, \ldots, A_{in_i}) \mid 1 \leq i \leq m,\ n_i \geq 0\}$ over ground atoms onto one CPT, called combined CPT, $\mathbf{P}(A \mid B_1, \ldots, B_n)$ with $\{B_1, \ldots, B_n\} \subseteq \bigcup_{i=1}^{m} A_{i1}, \ldots, A_{in_i}$. The output is empty iff the input is empty. Our definition of a combining rule is basically a reformulation of the definition given in [16][3]. As an example we consider the combining rule *max*. The functional formulation is

$$\mathbf{P}(A \mid \cup_{i=1}^{n} A_{i1}, \ldots, A_{in_i}) = \max_{i=1}^{n} \{\mathbf{P}(A \mid A_{i1}, \ldots, A_{in_i})\}$$

It is remarkable that a combining rule has full knowledge about the input, i.e., it knows all the appearing ground atoms or rather random variables and the associated domains of the random variables.

We assume that for each Bayesian predicate there is a corresponding combining rule and that the combined CPT still specifies a conditional probability distribution. From a practical perspective, the combining rules used in Bayesian logic programs will be those commonly employed in Bayesian nets, such as e.g. noisy-or, max.

3 Semantics

Following the principles of KBMC, each Bayesian logic program essentially specifies a propositional Bayesian net that can be queried using usual Bayesian net inference engines. This view implicitly assumes that all knowledge about the domain of discourse is encoded in the Bayesian logic program (e.g. the persons belonging to a family). If

[3] It differs mainly in the restriction of the input set to be finite. We make this assumption in order to keep things simple.

the domain of discourse changes (e.g. the family under consideration), then part of the Bayesian logic program has to be changed. Usually, these modifications will only concern ground facts (e.g. the Bayesian predicates "person", "parent" and "sex").

The structure of the corresponding Bayesian net follows from the semantics of the logic program, whereas the quantitative aspects are encoded in the CPTs and combining rules.

The set of random variables specified by a Bayesian logic program is the least Herbrand model of the program[4]. The least Herbrand model $LH(L)$ of a definite clause program L contains the set of all ground atoms that are logically entailed by the program[5], it represents the intended meaning of the program. By varying the evidence (some of the ground facts) one also modifies the set of random variables. Inference for logic programs has been well-studied (see e.g. [14]) and various methods exist to answer queries or to compute the least Herbrand model. All of these methods can essentially be adapted to our context. Here, we assume that the computation of $LH(L)$ relies on the use of the well-known T_L (cf. [14]) operator[6]. Let L be a Bayesian logic program.

$$T_L(\mathcal{I}) = \{A\theta \mid \text{there is a substitution } \theta \text{ and a clause } A \mid A_1, \ldots, A_n \text{ in } L \text{ such that } A\theta \mid A_1\theta, \ldots, A_n\theta \text{ is ground and all } A_i\theta \in \mathcal{I}\}$$

The least Herbrand model $LH(L)$ of L is then the least fixed point of $T_L(\emptyset)$. It specifies the set of relevant random variables. For instance, if one takes as Bayesian logic program the union of all Bayesian clauses written above together with *neighbourhood*(*james*) then $LH(L)$ consists of *neighbourhood*(*james*), *burglary*(*james*) and *alarm*(*james*). Notice that the least Herbrand model can be infinite when the logic program contains structured terms. This is not necessarily problematic as we will see later.

Given two ground atoms A and $B \in LH(L)$, we write that A *is directly influenced by* B if and only if there is a clause $A' \mid B_1, \ldots, B_n$ in L and a substitution θ that grounds the clause such that $A = A'\theta$ and $B = B_i\theta$ for some i and all $B_i\theta \in LH(L)$. The relation *influences* is then the recursive closure of the relation *directly influences*. Roughly speaking, a ground atom A influences B whenever there exists a proof for B that employs A. For instance, *alarm*(*james*) is influenced by *neighbourhood*(*james*) and directly influenced by *burglary*(*james*). Using the *influenced by* relation we can now state a conditional independency assumptions: let $A_1, \ldots, A_n$ be the set of all random variables that directly influence a variable A. Then each other random variable B not influenced by A, is conditionally independent of A given $A_1, \ldots, A_n$, i.e. $\mathbf{P}(A \mid A_1, \ldots, A_n, B) = \mathbf{P}(A \mid A_1, \ldots, A_n)$. E.g. given the propositional Bayesian logic program (the famous example due to Pearl)

```
earthquake.
burglary.
alarm | earthquake, burglary.
johncalls | alarm.
marycalls | alarm.
```

the random variable `johncalls` is conditionally independent of `marycalls` given `alarm`.

Obviously, the relation *influenced by* should be acyclic in order to obtain a well-defined Bayesian net. The network can only be cyclic when there exists an atom A that influences itself. In this case executing the query ?- A (using e.g. Prolog) is also problematic – the SLD-tree (see below) of the query will be infinite and the query may not terminate. Thus the logical component of the Bayesian logic program is itself problematic. Additional simple considerations lead to the following proposition:

Proposition 1. *Let B be a Bayesian logic program and $LH(B)$ the least Herbrand model of B. If B fulfills the following conditions:*

1. *the influenced by relation over $LH(B)$ is acyclic and*
2. *each random variable in $LH(B)$ is only influenced by a finite set of random variables,*

then it specifies a distribution P over $LH(B)$ which is unique in the sense that for each finite subset $S \subset LH(B)$ the induced distribution $\mathbf{P}(S)$ is unique.

A proof can be found in [8]. The conditions still allow infinite least Herbrand models but account

[4] Formally it is the least Herbrand model of the logical program L', which one gets from L by omitting the associated CPTs and combining rules as well as interpreting all predicates as classical, logical predicates. For the benefit of greater readability, in the sequel we do not distinguish between L and L'.

[5] If we ignore termination issues, these atoms can – in principle – be computed by a theorem prover, such as e.g. Prolog.

[6] For simplicity, we will assume that all clauses in a Bayesian logic program are range-restricted. This means that all variables appearing in the conclusion part of a clause also appear in the condition part. This is a common restriction in computational logic. When working with range-restricted clauses, all facts entailed by the program are ground. Also, the pruned and-or trees and graphs (cf. below) will only contain ground facts. This in turn guarantees that the constructed Bayesian net for any query contains only proper random variables.

for Bayesian nets: they are acyclic graphs and each node has a finite set of predecessors. Let us have a look at a program which violates the conditions, more exactly said, the properties of the random variable $r(a)$ together with the *directly influenced by* relation violates them:

```
r(a). s(a,b).
r(X) | r(X).
r(X) | s(X,f(Y)).
s(X,f(Y)) | s(X,Y).
```

Given this Program the random variable $r(a)$ is directly influenced by itself and by $s(a, f(b)), s(a, f(f((b))), \ldots$.

```
s(a).
r(X)    | r(f(X)).
r(f(X)) | s(f(X)).
s(f(X)) | s(X).
```

Given this Program the random variable $r(a)$ is influenced (not directly) by $r(f(a)), r(f(f((a))), \ldots$ though it has a finite proof. In this paper, we assume that the Bayesian logic program is unproblematic in this respect[7].

To summarize, the least Herbrand model of a Bayesian logic program specifies the random variables in the domain of discourse. These random variables can then in principle[8] be represented in a Bayesian net where the parents of a random variable v are all facts directly influencing v. Any algorithm solving the inference problem for Bayesian nets can now be applied. At this point it should be clear how Bayesian nets are represented as Bayesian logic programs. We only encode the dependency structure as a propositional Bayesian logic program. Everything else remains the same.

4 Query-answering procedure

In this section, we show how to answer queries with Bayesian logic programs. A probabilistic query or shortly a query is an expression of the form

```
?- Q | E1=1,...,EN=eN
```

and asks for the conditional probability distribution $\mathbf{P}(Q \mid E_1 = e_1, \ldots, E_n = e_n)$. We first consider the case where no evidence is given, and then show how to extend this in the presence of evidence.

[7] This is a reasonable assumption if the Bayesian logic program has been written by anyone familiar with Prolog.

[8] We neglect the finiteness of Bayesian nets for the moment.

4.1 Querying without evidence

First, we show how to compute the probability of the different possible values for a ground atom (a random variable) Q. Given a Bayesian logic program,

```
lives_in(james,yorkshire).
burglary(james).
lives_in(stefan,freiburg).
tornado(yorkshire).
alarm(X)    | burglary(X).
alarm(X)    | lives_in(X,Y), tornado(Y).
```

the query `?- alarm(james)` asks for the probabilities of $alarm(james) = true$ and $alarm(james) = false$. To answer a query ?- Q we do not have to compute the complete least Herbrand model of the Bayesian logic program. Indeed, the probability of Q only depends on the random variables that influence Q, which will be called *relevant* w.r.t. Q and the given Bayesian logic program. The relevant random variables are themselves the ground atoms needed to prove that Q is true (in the logical sense).

The usual execution model of logic programs relies on the notion of SLD-trees (see e.g. [14, 19]). For our purposes it is only important to realize that the succeeding branches in this tree contain all the relevant random variables. Furthermore, due to the range-restriction requirement all succeeding branches contain only ground facts. Instead of using the SLD-tree to answer the query in the probabilistic sense, we will use a *pruned and-or tree*, which can be obtained from the SLD-tree. The advantage of the pruned and-or tree is that it allows us to combine the probabilistic and logical computations. An *and-or tree* represents all possible partial proofs of the query. The nodes of an and-or tree are partitioned into *and* (black) and *or* (white) nodes. An *and* node for a query ?- $Q_1, \ldots, Q_n$ is proven if all of its successors nodes ?- Q_i are proven. An *or* node ?- Q is proven if at least one of its successors nodes is proven. There is a successor node ?- $A_1\theta, \ldots, A_n\theta$ for an *or* node ?- A if there exists a substitution θ and a Bayesian definite clause $A'|A_1, \ldots, A_n$ such that $A'\theta = A\theta$. Since we are only interested in those random variables used in successful proofs of the original query, we prune all subtrees which do not evaluate to true. A *pruned* and-or tree thus represents all proofs of the query. One such tree is shown in Figure 1. It is easy to see that each ground atom (random variable) has a unique pruned and-or tree. On the other hand, for some queries and Bayesian logic programs it might occur that a ground fact A occurs more than once in the pruned and-or tree. Given the uniqueness of

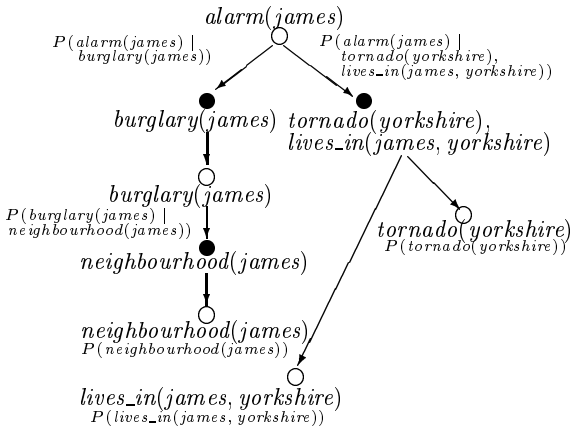


Figure 1: The pruned and-or tree (with associated CPTs) of the query `?- alarm(james)`.

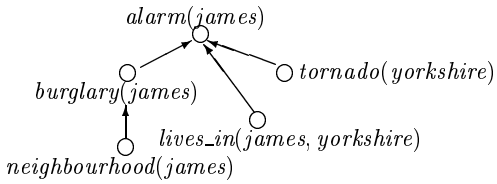


Figure 2: The dependency structure of the resulting Bayesian net of the query `?- alarm(james)`.

pruned and-or trees for ground facts, it is necessary to turn the pruned and-or tree into an and-or graph by merging any nodes for A. This can actually be achieved by storing all ground atoms proven so-far in a look-up table, and using this table to avoid redundant computations.

The resulting pruned and-or graph compactly represents the dependencies between the random variables entailed by the Bayesian logic program. E.g. the tree in Figure 1 says that *burglary(james)* is influenced by *neighbourhood(james)*. Furthermore, the and-or graph reflects the structure of the quantitative computations required to answer the query. To perform this computation, we store at each branch from an *or* node to an *and* node the corresponding CPT (cf. Figure 1). The combined CPT for the random variable v in the *or* node is then obtained by combining the CPTs on v's sub-branches using the combining rule for the predicate in v. It is always possible to turn the and-or graph into a Bayesian net. This is realized by (1) deleting each *and* node n and redirecting each successor of n to the parent of n (as shown in Figure 2), and (2) by using the combined CPT at each *or* node.

4.2 Querying with evidence

So far, we have neglected the evidence. It takes the form of a set of ground random variables $\{E_1, \ldots, E_n\}$ and their corresponding values $\{e_1, \ldots, e_n\}$. The Bayesian net needed to compute the probability of a random variable Q given the evidence consists of the union of all pruned and-or graphs for the facts in $\{Q, E_1, \ldots, E_n\}$. This Bayesian net can be computed incrementally, starting by computing the graph (and the look-up table as described above) for Q and then using this graph and look-up table when answering the logical query for E_1 in order to guarantee that each random variable occurs only once in the resulting graph. The resulting graph is then the starting point for E_2 and so on. Given the corresponding Bayesian net of the final and-or graph, one can then answer the original query using any Bayesian net inference engine to compute

$$\mathbf{P}(Q \mid E_1 = e_1, \ldots, E_n = e_n).$$

The qualitative dependency structure of the resulting Bayesian net for the query `?- alarm(james)` is shown in Figure 1. Normally the resulting Bayesian net is not optimal and can be pruned.

Retrospectively we can say that a probabilistic query `?- Q | E1=e1,..., EN=eN` is legal if the union of all and-or graphs of `Q, E1,..., EN` is finite. In other words, the SLD trees of `Q, E1,..., EN` must be finite.

5 Special cases

In this section, we illustrate the representational power and elegance of Bayesian logic programs by demonstrating that both Bayesian nets and definite clause programs (as in "pure" Prolog) can directly and straightforwardly be encoded as Bayesian logic programs. Furthermore, we also give examples of the use of Bayesian logic programs that involve structured terms (and have an infinite least Herbrand model).

5.1 Bayesian nets

Any Bayesian net (over finite domains) directly translates to a Bayesian logic program in the following manner. Any node n in the Bayesian net will be defined by a single Bayesian clause of the form $n \mid p_1, \ldots, p_n$, where $\{p_1, \ldots, p_n\}$ is the set of all parents of n. The CPT associated to the node in the net becomes the CPT of the predicate n. As an illustration, consider the famous standard example of [17] about burglary alarms at home. It translates to the following program:

```
burglary.
```

A_1	...	A_n	A *true*	A *false*
true	...	*true*	1.0	0.0
...	...	...	0.0	1.0
false	...	*false*	0.0	1.0

Table 1: CPT which is associated to a "logical" clause.

```
earthquake.
alarm      | burglary, earthquake.
johncalls  | alarm.
marycalls  | alarm.
```

where the associated CPTs are identical to the CPTs of the Bayesian net. It is easy to see that any program which get using is the described translation fullfils the conditions of proposition 1.

5.2 "Pure" Prolog programs and structured terms

Another interesting subclass of Bayesian logic programs are "pure" Prolog programs. The Bayesian logic program

```
father(jef,paul).
mother(an,paul).
parent(X,Y)      | father(X,Y).
parent(X,Y)      | mother(X,Y).
```

defines the parent predicate in terms of father and mother. Let us now define the domains of all predicates as $\{true, false\}$ and associate to each clause $A \mid A_1, \ldots, A_n$ the CPT of Table 1 which assigns a probability 1.0 to $A = true$ only when all of the A_i are *true*, otherwise $A = false$ with probability 1.0. If one then uses a combining rule like max or *noisy-or* one obtains the same semantics and behaviour as for "pure" Prolog. This also motivates us to use Prolog's notation `:-` instead of `|` (and the CPTs) as a short hand for these assumptions. Consider the following Bayesian logic program:

```
even(0).
even(s(X)) | odd(X).
odd(s(X))  | even(X).
```

It is easy to see that we can compute the probability distribution of any ground atom for the predicates *even* and *odd* with or without evidence. First, consider the case where there is no evidence. Then the SLD tree for the query will be finite and as a consequence also the and-or graph will be. This should allows us to compute the desired answer. Second, consider the case where there is evidence, e.g. $even(s(s(0))) = true$ and we want to compute the probability of $odd(s(0))$. Then we need to compute the Bayesian net, which is $even(0) \rightarrow odd(s(0)) \rightarrow even(s(s(0)))$, feed it into a propositional Bayesian net engine and compute the result. One can easily see that despite the presence of structured terms and an (countable) infinite number of random variables, the required computations are finite. This will - of course - only be the case when the corresponding Prolog program behaves well w.r.t. the query and evidence. In other words it should fullfil proposition 1.

6 Implementation

The following Prolog code enables one to compute the structure of the pruned and-or graph of a random variable as a set of ground facts of the predicate *imply*, assuming that the logical structure of a Bayesian logic program is given as a Prolog program. The and-or graph is represented as a set of ground atoms of `imply(or:X,and:Y)` and `imply(and:X,or:Y)`. The use of the Prolog's own query procedure proves for two reasons as efficient: (1) it implements the desired search and (2) it is efficient and uses an efficient hash table. We do not present the entire source code, because the remaining program parts follow directly from the previous discussion.

```
build_AOG(Goal) :-
  clause(Goal, Body),imply(or:Goal,and:Body),!.
build_AOG(Goal) :-
  clause(Goal, Body), build_AOG_Body(Goal,Body),
  assert(imply(or:Goal,and:Body)).
build_AOG_Body(_,true):- !.
build_AOG_Body(_,(Body,Bodies)) :- !,
  build_AOG(Body),
  build_AOG_Conj((Body,Bodies),Bodies),
  assert(imply(and:(Body,Bodies),or:Body)).
build_AOG_Body(_,(Body)) :-
  build_AOG(Body), assert(imply(and:Body,or:Body)).
build_AOG_Conj((Goal,Goals),(Body,Bodies)) :- !,
  build_AOG(Body),
  build_AOG_Conj((Goal,Goals),Bodies),
  assert(imply(and:(Goal,Goals),or:Body)).
build_AOG_Conj((Goal,Goals),Body) :- !,
  build_AOG(Body),
  assert(imply(and:(Goal,Goals),or:Body)).
```

The pruned and-or graph is the component containing the root node as the following example clarifies. On the query `?- alarm(stefan).` the code asserts

```
imply(or:lives_in(stefan,freiburg), and:true).
```

The reason for that is that the and-or graph of a ground atom g, which comes in a body of a rule before a ground atom g', is asserted without proving the truth of g'. To extract the right component one can use the following code:

```
extract_pruned_AOG([]).
extract_pruned_AOG([_:true|Rest]) :-
  extract_pruned_AOG(Rest).
extract_pruned_AOG([Goal|Rest]) :-
  findall(Body,(imply(Goal,Body),
                not marked(Goal,Body),
                assert(marked(Goal,Body))),
                Successors),
  append(Successors,Rest,NewRest),
  extract_pruned_AOG(NewRest).
```

Calling `extract_pruned_AOG([or: alarm(stefan)])` it marks all nodes of the component containing the root node. After marking we can use

```
findall((X,Y),(imply(X,Y),
              not marked(X,Y),
              retract(imply(X,Y))),_).
```

to delete all irrelevant nodes and arcs. Furthermore, the code typifies the reason why "pure" Prolog programs as well as structured terms can be elegantly handled with Bayesian logic programs: it describes essentially an usual Prolog meta-interpreter. Moreover it should make the definitions of legal queries clearer.

7 Learning?

So far we have merely introduced a framework that combines Bayesian nets with first order logic. In this section, we provide some initial ideas on how Bayesian logic programs might be learned.

The inputs to a system learning Bayesian logic programs should consist of a set of cases, where each case describes a set of random variables as well as their states. One complication that often arises while learning Bayesian nets and that is also relevant here is that some random variables or the corresponding states may not be fully observable.

In the literature on learning Bayesian nets [6, 3] one typically distinguishes between: (1) learning the structure of a net (model selection) and/or (2) learning the associated CPTs. This distinction also applies to Bayesian logic programs, where one can separate the clausal structure from the CPTs. In addition, the combining rules could be learned[9]. Let us address each of these in turn.

For what concerns learning the underlying logic program of a Bayesian logic program, it is clear that techniques from the field of inductive logic programming [15] could be helpful.

To give an idea of how this might work, we merely outline one possibility for learning the structure of the Bayesian logic program from a set of cases in which the relevant random variables are specified (though their values need not be known). This means that for each case we know the least Herbrand model. One technique for inducing clauses from models (or interpretations) is the clausal discovery technique by De Raedt and Dehaspe [4]. Basically, this technique starts from a set of interpretations (which in our case corresponds to the Herbrand models of the cases) and will induce all clauses (within a given language bias) for which the interpretations are models. E.g. given the single interpretation $\{female(soetkin), male(maarten), human(maarten), human(soetkin)\}$ and an appropriate language bias the clausal discovery engine would induce $human(X) \leftarrow male(X)$ and $human(X) \leftarrow female(X)$. The Claudien algorithm essentially performs an exhaustive search through the space of clauses which is defined by a language $\mathcal{L}$. Roughly speaking, Claudien keeps track of a list of candidate clauses Q, which is initialized to the maximally general clause in $\mathcal{L}$. It repeatedly deletes a clause c from Q, and test whether all given interpretations are a model for c. If they are, c is added to the final hypothesis, otherwise all maximally general specializations of c in $\mathcal{L}$ are computed (using a so-called *refinement operator* [15]) and added back to Q. This process continues until Q is empty and all relevant parts of the search-space have been considered. A declarative bias hand-written by the user determines the type of regularity searched for and reduces the size of the space in this way. The pure clausal discovery process as described by De Raedt and Dehaspe may induce cyclic logic programs. However, extensions as described in [1] can avoid these problems.

If we assume that the logic program and the combining rules are given, we may learn the associated CPTs. Upon a first investigation, it seems that the work of [13] can be adapted towards Bayesian logic programs. They describe an EM based algorithm for learning the entries of CPTs of a probabilistic logic program in the framework of [16] which is strongly related to our framework as is shown in [8]. The approach makes two reasonable assumptions: (1) different data cases are independent and (2) the combining rules are decomposable, i.e., they can be expressed using a set of separate nodes corresponding to the different influences, which are then combined in another node. As Koller and Pfeffer note, all commonly used combining rules meet this condition.

To summarize, it seems that ideas from inductive logic programming can be combined with those from Bayesian learning in order to induce Bayesian logic programs. Our further work intends to inves-

[9] For our suggestions we assume that the rules are determined by a user because learning the rules results in an explosion of complexity.

tigate these issues.

8 Conclusions

In this paper, we presented Bayesian logic programs as an intuitive and simple extension of Bayesian nets to first-order logic. Given Prolog as a basis, Bayesian logic programs can easily be interpreted using a variant of a standard meta-interpreter. We also indicated parallels to existing algorithms for learning the numeric entries in the CPTs and gave some promising suggestions for the computer-supported specification of the logical component.

9 Acknowledgements

We would like to thank Daphne Koller, Manfred Jaeger, Peter Flach and James Cussens for discussions and encouragement.

References

[1] H. Blockeel and L. De Raedt. ISSID : an interactive system for database design. *Applied Artificial Intelligence*, 12(5):385–420, 1998.

[2] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Contect-specific independence in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-1996)*, 1996.

[3] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transaction on Knowledge and Data Engineering*, 1999. (to appear).

[4] L. De Raedt and L. Deshape. Clausal discovery. *Machine Learning*, (26):99–146, 1997.

[5] P. Haddawy. An overview of some recent developments on Bayesian problem solving techniques. *AI Magazine - Special Issue on Uncertainty in AI*, Summer 1999. (to appear).

[6] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, March 1995.

[7] M. Jaeger. Relational Bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-1997)*, 1997.

[8] K. Kersting. Bayes'sche-logische Programme. Master's thesis, Albert-Ludwigs-University, Freiburg, Germany, 2000.

[9] K. Kersting and L. De Raedt. Bayesian Logic Programs. In J. Cussens and A. Frisch, editors, *Work-in-Progress Reports of the Tenth International Conference on Inductive Logic Programming (ILP -2000)*, London,U.K, 2000.

[10] K. Kersting and L. De Raedt. Bayesian Logic Programs. In F. Furukawa, S. Muggleton, D. Michie, and L. De Raedt, editors, *Proceedings of the Seventeenth Machine Intelligence Workshop*, Bury St. Edmunds, Suffolk, U.K., 2000.

[11] K. Kersting, L. De Raedt, and S. Kramer. Interpreting Bayesian Logic Programs. In L. Getoor and D. Jensen, editors, *Working Notes of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data (SRL)*, Austin, Texas, July 2000.

[12] D. Koller. Probabilistic relational models. In S. Dzeroski and P. Flach, editors, *Proceedings of Ninth International Workschop on Inductive Logic Programming (ILP-1999)*, number 1634 in LNAI, Bled, Slovenia, June 1999. Springer.

[13] D. Koller and A. Pfeffer. Learning probabilities for noisy first-order rules. In *Proceedings of the Fifteenth Joint Conference on Artificial Intelligence*, pages 1316–1321, August 1997.

[14] J. W. Lloyd. *Foundation of Logic Programming*. Springer, Berlin, 2. edition, 1989.

[15] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 1994.

[16] L. Ngo and P. Haddawy. Answering queries form context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.

[17] J. Pearl. *Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2. edition, 1991.

[18] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1995.

[19] L. Sterling and E. Shapiro. *The Art of Prolog: Advanced Programming Techniques*. The MIT Press, 1986.

A Multi-scale Support Vector Approach to Classification

Matthias Rychetsky, Kurt Ackermann and Manfred Glesner*

*Darmstadt University of Technology, Institute for Microelectronic Systems, Karlstrasse 15, 64283 Darmstadt, Germany, Email rychetsky@mes.tu-darmstadt.de

Abstract. This paper describes a multi-resolution approach for support vector classification. A first machine is doing a standard classification, whereas a second (regression) machine corrects false classifications by correcting the margin of the first machine. Both machines are trained simultaniously within one optimization problem.

Keywords. Support Vector Machine, Multi-resolution, Local Learning

1 Introduction

Support vector machines have proven to be well performing classification and regression learning machines. Nevertheless there are some draw-backs. E.g. there is no possibility to model local and global properties of a data set in the standard Support Vector Machine. This is desirable to get a more condensed representation and improve the performance for regions with higher data density. To start the support vector machine and it's learning algorithm will be sketched. Than a multi-resolution approach is described which over-comes the disadvantages described above.

2 Support Vector Machines - A short overview

Let us first define a training data set, which is given by

$$\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subset \mathbb{R}^M \times \{+1, -1\}. \quad (1)$$

Furthermore we define that a *kernel matrix* $\mathbf{K}$ used in this context should be symmetric and positive definite. The matrix $\mathbf{K}$ consists of elements $k(\boldsymbol{x}_1, \boldsymbol{x}_2)$ (where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are some data vectors) which can be e.g., of the form

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^T \boldsymbol{x}_2 + 1)^p \quad (2)$$

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2} \quad (3)$$

Using the first kernel one can construct polynomial classifiers of order p and the second one gives a radial-basis-function machine with variance σ. Now a *kernel classifier* is given by [3, 4] as

$$h(\boldsymbol{x}) = \text{sign}(f(\boldsymbol{x})) = \text{sign}\left(\sum_{i=1}^{N} \beta_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b\right) \quad \boldsymbol{\beta} \in \mathbb{R}^N. \quad (4)$$

This is a linear classifier in a high dimensional kernel space (Hilbert space), using a bias b. Further on we will use $\beta_i = \alpha_i * y_i$. The function f can be expressed as an inner product between the mapped point $\boldsymbol{x}$ and a vector $\boldsymbol{w} \in \mathcal{F}$ (where $\mathcal{F}$ is a *feature space*)

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle. \quad (5)$$

Where the weight vector $\boldsymbol{w}$ can be expressed in terms of $\phi(\boldsymbol{x})$ (The kernel function is $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \phi(\boldsymbol{x}_1)\phi(\boldsymbol{x}_2)$)

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(\boldsymbol{x}_i). \quad (6)$$

For recall the support vector approach is sketched here (for a more detailed introduction see e.g. [1] or [4]). In the SVM approach one wants to maximize the decision margin (which is given by $1/\|\boldsymbol{w}_{SVM}\|_2$). This can be done by solving the following optimization problem (primal problem)

$$min \frac{1}{2}\|\boldsymbol{w}_{SVM}\|^2, \quad (7)$$

subject to

$$y_i \cdot (\boldsymbol{w}_{SVM} \cdot \boldsymbol{\phi}(\boldsymbol{x_i}) - b) \geq 1, i = 1, \ldots, N. \quad (8)$$

This leads to a quadratic optimization problem using the Lagrange parameters α. To optimize a support vector machine with non-linear decision function in the

non separable case one has to minimize (dual problem)

$$Q_1(\boldsymbol{\alpha}) = -\sum_{i=1}^{N}\alpha_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{9}$$

subject to

$$\sum_{i=1}^{N}\alpha_i y_i = 0, \tag{10}$$

$$0 \leq \alpha_i \leq C \qquad \text{for } i = 1, 2, \ldots, N. \tag{11}$$

Where the parameter C controls the trade off between errors of the SVM on training data and margin maximization ($C = \infty$ leads to hard margin SVM).

3 A Multi-scale Approach

A draw-back of the support vector machine described above is that there is no possibility to describe local properties of the data. The kernel function for the SVM is choosen once and for the whole data set. In this context the multi-resolution analysis, first introduced in signal processing, is very attractive. For example a Wavelet-based multi-resolution analysis of a signal provides a very powerful tool to identify local and at the same time global properties of the signal. Such a representation can model local or transient features (e.g. edges) better and simultaniously leads to a more compact representation. Both is also very desirable for the support vector machine for classification. Shao and Cherkassky [2] developed an approach with these features, but only for regression. Here we present a method for constructing a support vector classifier using a method similar to the one described in this approach.

The decision function for which we would like to construct the parameters has the form:

$$\begin{aligned} h(\boldsymbol{x}) = \text{sign}(&\sum_{i=1}^{N}(\alpha_i y_i k_1(\boldsymbol{x}_i, \boldsymbol{x}) \\ &+\hat{\alpha}_i y_i k_2(\boldsymbol{x}_i, \boldsymbol{x})) + b) \end{aligned} \tag{12}$$

As it can be seen the decision consists of a superposition of two kernel decision functions: The first one using a (wider) kernel function k_1 and the second one using a (narrow) kernel function k_2. The first part is built up using a normal classification SVM as described above. In the second part the residuum of the first machine to the training data set is learned. This is done by learning the margin deviation of the first machine to the training labels y_i. To do this a regression machine is learned which corrects only the negative margin deviations (if $(y_i \cdot f(\boldsymbol{x}_i)) < 0$). The primal optimization problem for a regression machine which approximates only negative values is

$$J(\boldsymbol{w}, \xi) = C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

subject to

$$\begin{aligned} \boldsymbol{w}^T\phi(\boldsymbol{x}_i) - y_i &\leq \epsilon + \xi_i, \qquad \text{for } i = 1, 2, \ldots, N \\ \xi_i &\geq 0 \qquad \text{for } i = 1, 2, \ldots, N \end{aligned}$$

This can be transformed into the dual optimization problem

$$\begin{aligned} Q_2(\hat{\boldsymbol{\alpha}}) = &\sum_{i=1}^{N}\hat{\alpha}_i\hat{y}_i + \epsilon\sum_{i=1}^{N}\hat{\alpha}_i \\ &+ \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\hat{\alpha}_i\hat{\alpha}_j k_2(\boldsymbol{x}_i, \boldsymbol{x}_j), \end{aligned} \tag{13}$$

subject to

$$0 \leq \hat{\alpha}_i \leq \hat{C} \qquad \text{for } i = 1, 2, \ldots, N. \tag{14}$$

Where $\hat{\boldsymbol{\alpha}}$ denotes the parameter vector of the regression problem. Using the margin of the classification machine

$$\hat{y}_i = y_i \cdot f(\boldsymbol{x}_i) \tag{15}$$

equation 13 can be extended to

$$\begin{aligned} Q_2(\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}) = &\epsilon\sum_{i=1}^{N}\hat{\alpha}_i \\ &+ \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\hat{\alpha}_j y_i y_j k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &+ \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\hat{\alpha}_i\hat{\alpha}_j k_2(\boldsymbol{x}_i, \boldsymbol{x}_j), \end{aligned} \tag{16}$$

Since a closed optimization procedure is searched for both machines, we have to minimize:

$$Q(\alpha, \hat{\alpha}) = Q_1(\alpha) + Q_2(\alpha, \hat{\alpha}) \tag{17}$$

So finally the following optimization problem has to

be solved:

$$\begin{aligned} Q(\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}) =& \epsilon \sum_{i=1}^{N} \hat{\alpha}_i - \sum_{i=1}^{N} \alpha_i \\ &+ \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \hat{\alpha}_j y_i y_j k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &+ \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_i \hat{\alpha}_j k_2(\boldsymbol{x}_i, \boldsymbol{x}_j), \end{aligned} \tag{18}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0, \qquad \text{and} \qquad \sum_{i=1}^{N} \hat{\alpha}_i y_i = 0, \tag{19}$$

$$\begin{aligned} 0 \leq \alpha_i \leq C \qquad & \text{for } i = 1, 2, \ldots, N, \\ 0 \leq \hat{\alpha}_i \leq \hat{C} \qquad & \text{for } i = 1, 2, \ldots, N. \end{aligned} \tag{20}$$

In figure 1 a sample plot for a binary classification problem (a subset of the iris data set) can be seen.

4 Conclusion and outlook

We presented an algorithm for finding a multi-resolution support vector machine for classification. Currently extensive studies are going to determine the performance of this approach. The method has the nice property that in one closed optimization problem a coarse approximation of the decision boundary and a finer approximation of the local residuum are performed. A possible extension of the presented structure is to introduce more levels of approximation.

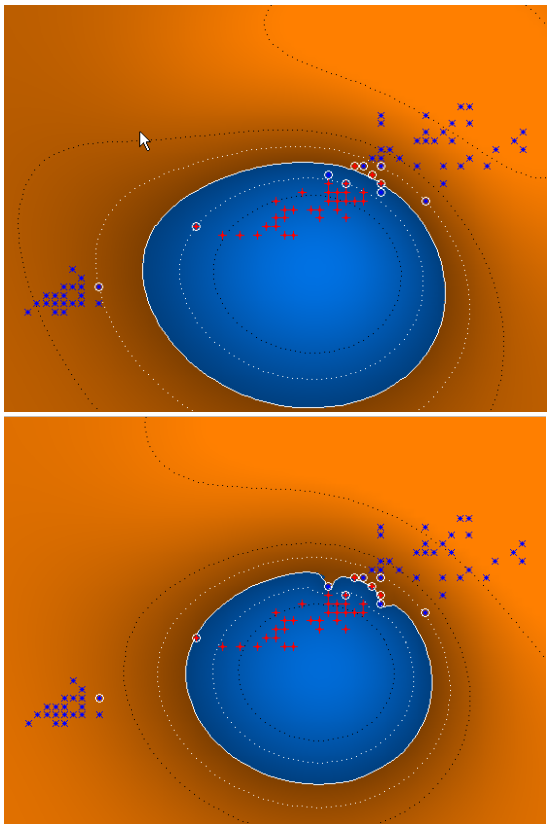


Figure 1 Iris data set classification using a standard SVM (top) with a RBF kernel ($\sigma = 0.5$) and a multi-resolution SVM using two RBF kernels ($\sigma_1 = 0.5, \sigma_2 = 0.03$).

References

1. C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.
2. X. Shao and V. Cherkassky. Multi-resolution support vector machine. In *International Joint Conference on Neural Networks (IJCNN 99)*, Washington DC, USA, July 1999.
3. Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, NY, 1995.
4. Vladimir Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, NY, 1998.

Estimating the Generalization Performance of an SVM Efficiently

Thorsten Joachims JOACHIMS@LS8.INFORMATIK.UNI-DORTMUND.DE
Informatik LS VIII, Universität Dortmund, Baroper Str. 301, 44221 Dortmund, Germany

Abstract

This paper proposes and analyzes an efficient and effective approach for estimating the generalization performance of a support vector machine (SVM) for text classification. Without any computation-intensive resampling, the new estimators are computationally much more efficient than cross-validation or bootstrapping. They can be computed at essentially no extra cost immediately after training a single SVM. Moreover, the estimators developed here address the special performance measures needed for evaluating text classifiers. They can be used not only to estimate the error rate, but also to estimate recall, precision, and F_1. A theoretical analysis and experiments show that the new method can effectively estimate the performance of SVM text classifiers in an efficient way.

1. Introduction

From a practical perspective, learning theory should provide accurate and efficient methods for predicting how well a learner can handle the task at hand. Given a set of training examples, a practitioner will ask how well a particular learner will generalize using this data. With this information it is possible to address the question of selecting between different representation, different models, and different learning parameters. Similarly, it provides the basis for deciding when to apply the learned rule, since real-world tasks usually require a certain minimum performance.

The following presents an approach to answering these questions in the context of text classification with Support Vector Machines (SVMs) (Joachims, 1998; Dumais et al., 1998). The aim is to develop operational performance estimators that are of actual use when applying SVMs. This requires that the estimators are both effective and computationally efficient. Devroye et al. (1996) give an overview of error estimation. While some estimators (e.g., uniform convergence bounds) are powerful theoretical tools, they are of little use in practical applications, since they are too loose. Others (e.g., cross-validation, bootstrapping) give good estimates, but are computationally inefficient. This paper describes new estimators that overcome these problems, extending results of Vapnik (1998) and Jaakkola and Haussler (1999) to general SVMs. The new estimators are both accurate and efficient to compute.

While the results presented here are general enough to apply to arbitrary classification tasks, special emphasis is put on evaluation measures commonly used in text classification. In particular, the approach is not limited to estimating the error rate. It also covers precision and recall, as well as combined measures like F_1. A theoretical analysis as well as experiments show that the new estimators accurately reflect the actual behavior of SVMs on text classification tasks.

2. Text Classification

Text classification is the primary application domain for the methods developed here. The goal of text classification is the automatic assignment of documents to a fixed number of semantic categories. Using a binary classifier for each such category, this leads to the following type of supervised learning problem. Given is an i.i.d. training sample S_n

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n) \tag{1}$$

of size n from a fixed but unknown distribution $\Pr(\vec{x}, y)$ describing the learning task. $\vec{x}_i$ represents the document content and $y_i \in \{-1, +1\}$ indicates the class. The learner $\mathcal{L}$ aims to find a decision rule $h_{\mathcal{L}} : \vec{x} \longrightarrow \{-1, +1\}$ based on S_n that classifies new documents as accurately as possible.

Documents, which typically are strings of characters, have to be transformed into a representation suitable for the learning algorithm and the classification task. Here the so called "bag-of-words" representations following the setup of Joachims (1998) is used. Each distinct word corresponds to a feature with its TFIDF

score (Salton & Buckley, 1988) as the value. Neither stemming nor stopword-removal are used. To abstract from different document lengths, each document feature vector $\vec{x}_i$ is normalized to unit length.

Unlike for other applications of machine learning, error rate $Err(h)$ alone is not necessarily a good performance measure in text classification, since very often the examples from one class vastly outnumber those from the other class. Instead, scores based on precision and recall are used widely. I define the recall $Rec(h)$ of a decision rule h as the probability that a document with label $y = 1$ is classified correctly. The precision $Prec(h)$ of a decision rule h is the probability that a document classified as $h(\vec{x}) = 1$ is indeed classified correctly. Between high precision and high recall exists a trade-off. A popular method to combine both into a single performance measure is the F_1. It is defined as the geometric mean of precision and recall.

3. Support Vector Machines

The new estimators exploit a particular property of Support Vector Machines that is identified in section 4. SVMs (e.g., Vapnik, 1998) were developed based on the structural risk minimization principle from statistical learning theory. In their basic form, SVMs learn linear decision rules $h(\vec{x}) = sign\{\vec{w} \cdot \vec{x} + b\}$ described by a weight vector $\vec{w}$ and a threshold b. For a given training sample S_n, the SVM finds the hyperplane with maximum soft-margin. Computing this hyperplane is equivalent to solving the following optimization problem (Vapnik, 1998), **(OP1)**:

$$\text{minimize:} \quad V(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum_{i=1}^{n} \xi_i \quad (2)$$

$$\text{subj. to:} \quad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \quad (3)$$

$$\forall_{i=1}^{n} : \xi_i > 0 \quad (4)$$

The constraints (3) require that all training examples are classified correctly up to some slack ξ_i. If a training example lies on the "wrong" side of the hyperplane, the corresponding ξ_i is greater or equal to 1. Therefore, $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of training errors. The factor C in (2) is a parameter that allows one to trade off training error vs. model complexity. Instead of soving OP1 directly, it is easier to solve the following Wolfe dual of OP1 (Vapnik, 1998), **(OP2)**:

$$\text{minimize: } W(\vec{\alpha}) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \vec{x}_j) \quad (5)$$

$$\text{subj. to: } \sum_{i=1}^{n} y_i \alpha_i = 0 \quad \wedge \quad \forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \quad (6)$$

In this paper, SVM^{Light} (Joachims, 1999b) is used for computing the solution of OP2[1]. All training examples with $\alpha_i > 0$ at the solution are called support vectors. To differentiate between those with $0 < \alpha_i < C$ and those with $\alpha_i = C$, the former will be called *unbounded support vectors* while the latter will be called *bounded support vectors*. From the solution $\vec{\alpha}$ of OP2 the decision rule $h(\vec{x})$ and the solution of OP1 can be computed using $\vec{w} \cdot \vec{x} = \sum_{i=1}^{n} \alpha_i y_i (\vec{x}_i \cdot \vec{x})$, the threshold $b = y_{usv} - \vec{w} \cdot \vec{x}_{usv}$, and the training losses $\xi_i = max(1 - y_i \left[\vec{w} \cdot \vec{x}_i + b\right], 0)$. The training example $(\vec{x}_{usv}, y_{usv})$ for calculating b must be an unbounded support vector. While it is highly unlikely in practice that one gets a solution of OP2 with only bounded support vectors, it is theoretically possible (see Burges & Crisp, 1999; Rifkin et al., 1999, for a thorough discussion). In this case the solution of the SVM will be called *unstable*, since the hyperplane is not uniquely determined. If there is at least one unbounded support vector, the solution is called *stable*. While SVMs are an essentially linear method, they can easily be generalized to non-linear decision rules by replacing the inner-products $(\vec{x}_i \cdot \vec{x}_j)$ with a kernel function $\mathcal{K}(\vec{x}_i, \vec{x}_j)$ (Vapnik, 1998) in the formulas above.

4. Efficient Performance Estimators for SVMs

The estimators developed in the following are based on the idea of leave-one-out estimation (Lunts & Brailovskiy, 1967). The leave-one-out estimator of the error rate proceeds as follows. From the training sample $S_n = ((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n))$ the first example $(\vec{x}_1, y_1)$ is removed. The resulting sample $S^{\setminus 1} = ((\vec{x}_2, y_2), \cdots, (\vec{x}_n, y_n))$ is used for training, leading to a classification rule $h_{\mathcal{L}}^{\setminus 1}$. This classification rule is tested on the held out example $(\vec{x}_1, y_1)$. If the example is classified incorrectly it is said to produce a leave-one-out error. This process is repeated for all training examples. The number of leave-one-out errors $\sum_{i=1}^{n} L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i), y_i)$ divided by n is the leave-one-out estimate $Err_{loo}^{n}(h_{\mathcal{L}})$ of the generalization error.

A shortcoming of the leave-one-out estimator is its computational inefficiency. The learner needs to be invoked n times for a training set of size n. For most practical problems, this is prohibitively expensive. The estimators proposed in the following are based on the leave-one-out method, but require an order of magnitude less computation time due to particular properties of the SVM. In particular, they do not require actually performing resampling and retraining,

[1] www-ai.informatik.uni-dortmund.de/svm_light

but can be applied directly after training the learner on S_n. The inputs to the estimators are the vector $\vec{\alpha}$ solving the dual SVM training problem OP2 and the vector $\vec{\xi}$ from the solution of the primal SVM training problem OP1. Due to this dependence, they will be called $\xi\alpha$-estimators in the following. Both $\vec{\alpha}$ and $\vec{\xi}$ are available at no extra cost after training the SVM.

4.1 Error Rate

Based on the solution $\vec{\alpha}$ of the dual SVM training problem and the vector of training losses $\vec{\xi}$, the $\xi\alpha$-estimator of the error rate $Err^n_{\xi\alpha}(h_{\mathcal{L}})$ is defined as follows.

Definition 1 *For stable soft-margin SVMs, the $\xi\alpha$-estimator of the error rate is*

$$Err^n_{\xi\alpha}(h_{\mathcal{L}})=\frac{d}{n} \quad with \quad d=|\{i:(\rho\alpha_i R^2_\Delta+\xi_i)\geq 1\}| \quad (7)$$

with ρ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems OP1 and OP2 on the training set S_n. R^2_Δ is an upper bound on $\mathcal{K}(\vec{x},\vec{x})-\mathcal{K}(\vec{x},\vec{x}')$ for all $\vec{x},\vec{x}'$.

$R^2_\Delta=1$ for the linear kernel $\mathcal{K}(\vec{x}_i,\vec{x}_j)=\vec{x}_i\cdot\vec{x}_j$ on normalized document vectors. The definition introduces the parameter ρ. While the theoretical results that are derived below assume $\rho=2$, we will see that $\rho=1$ is a good choice for text classification. The key quantity in definition 1 is d. d counts the number of training examples for which the inequality $(\rho\alpha_i R^2+\xi_i)\geq 1$ holds. But how does one come to this definition of d and what exactly does d count?

The key idea to the $\xi\alpha$-estimator is a connection between the training examples for which the inequality $(\rho\alpha_i R^2+\xi_i)\geq 1$ holds and those training examples that can produce an error in leave-one-out testing. In particular, if an example $(\vec{x}_i,y_i)$ is classified incorrectly by a SVM trained on the subsample $S_n^{\setminus i}$, then example $(\vec{x}_i,y_i)$ *must* fulfill the inequality $(\rho\alpha_i R^2+\xi_i)\geq 1$ for a SVM trained on the full sample S_n. This implies that d is an upper bound on the number of leave-one-out errors. The following lemma establish this result formally.

Lemma 1 *The number of leave-one-out errors $\sum_{i=1}^n L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i),y_i)$ of stable soft-margin SVMs on a training set S_n is bounded by*

$$\sum_{i=1}^{n} L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i),y_i)\leq|\{i:(2\alpha_i R^2+\xi_i)\geq 1\}| \quad (8)$$

$\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems OP1 and OP2 on the training set S_n. R^2 is an upper bound on $\mathcal{K}(\vec{x},\vec{x})$ and $\mathcal{K}(\vec{x},\vec{x}')\geq 0$.

Proof (sketch, more details in (Joachims, 1999a)) An error on a left-out example $(\vec{x}_t,y_t)$ occurs when at the solution of

$$W_t(\vec{\alpha}^t)=max_{\vec{0}\leq\vec{\alpha}^t\leq\vec{C}}\,1^T\vec{\alpha}^t-\frac{1}{2}\vec{\alpha}^{tT}H^t\vec{\alpha}^t\wedge\vec{y}^{tT}\vec{\alpha}^t=0 \quad (9)$$

where $\vec{\alpha}^t$, $\vec{y}^t$, and H^t have the t-th example removed, the expression $y_t\left[\sum_{i\neq t}\alpha_i^t y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b^t\right]>0$ is false. What follows in this proof are conditions for when this expression must be true based on the soft-margin SVM solution

$$W(\vec{\alpha})=max_{\vec{0}\leq\vec{\alpha}\leq\vec{C}}\,1^T\vec{\alpha}-\frac{1}{2}\vec{\alpha}^T H\vec{\alpha}\wedge\vec{y}^T\vec{\alpha}=0 \quad (10)$$

that involves all n training examples. Three cases can occur based on the optimal value of α_t:

Case $\alpha_t=0$: Example $(\vec{x}_t,y_t)$ is not a support vector. Then $W_t(\vec{\alpha}^t)=W(\vec{\alpha})$ and $y_t\sum_{i\neq t}y_i\alpha_i^t\mathcal{K}(\vec{x}_t,\vec{x}_i)=y_t\sum_{i\neq t}y_i\alpha_i\mathcal{K}(\vec{x}_t,\vec{x}_i)=1>0$.

Case $0<\alpha_t<C$: Example $(\vec{x}_t,y_t)$ is a support vector. From the solution $\vec{\alpha}^t$ of $W_t(.)$, the following construction produces a feasible point $\vec{\beta}$ for $W(.)$.

$$\beta_i=\begin{cases}\alpha_i^t & if \quad \alpha_i^t=0\vee\alpha_i^t=C\\ \alpha_i^t-y_i y_t\nu_i & if \quad i\in SV^t\\ \alpha_i & if \quad i=t\end{cases} \quad (11)$$

ν_i has to fulfill the following constraints. Let SV^t be the set of indices corresponding to support vectors of the solution $W_t(\vec{\alpha}^t)$ that are not at the upper bound C (that is $0<\alpha_i^t<C$). Then $\nu_i=0$ for all $i\notin SV^t$. For $i\in SV^t$ the ν_i are chosen to be non-negative and so that $\sum_{i\in SV^t}\nu_i=\alpha_t$ and $0\leq\beta_i\leq C$. Finding such ν_i is always possible, if there are at least two support vectors not at the upper bound C. The existence of such two vectors follows from the assumption that the SVMs solution is stable. From the construction of the ν_i it follows that $\vec{y}^T\vec{\beta}=0$ and $0\leq\beta_i\leq C$. So $\vec{\beta}$ is a feasible point of $W(.)$. It is now possible to transform (10) into

$$\begin{aligned}\alpha_t y_t\left[\sum_{i\in SV^t}\alpha_i^t y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b^t\right]=\\ -W(\vec{\beta})+W_t(\vec{\alpha}^t)-\frac{1}{2}\alpha_t^2\mathcal{K}(\vec{x}_t,\vec{x}_t)+\alpha_t \quad (12)\\ -\frac{1}{2}\sum_{i\in SV^t}\sum_{j\in SV^t}\nu_i\nu_j\mathcal{K}(\vec{x}_i,\vec{x}_j)+\alpha_t\sum_{i\in SV^t}\nu_i\mathcal{K}(\vec{x}_i,\vec{x}_t)\end{aligned}$$

A similar construction produces a feasible point $\vec{\gamma}$ of $W_t(.)$ based on the solution $\vec{\alpha}$ of $W(.)$.

$$\gamma_i=\begin{cases}\alpha_i & if \quad \alpha_i=0\vee\alpha_i=C\\ \alpha_i+y_i y_t\mu_i & if \quad i\in SV\setminus\{t\}\end{cases} \quad (13)$$

SV is the set of indices corresponding to support vectors not at the upper bound for the solution $\vec{\alpha}$ (that is $0 < \alpha_i < C$). $SV\backslash\{t\}$ excludes the index t corresponding to the left-out example. μ_i is chosen non-negative for all $i \in SV\backslash\{t\}$ such that $\sum_{i\in SV\backslash\{t\}} \mu_i = \alpha_t$ and $0 \leq \gamma_i \leq C$. From the construction of the μ_i it follows that $\vec{y}^T\vec{\gamma} = 0$ and so $\vec{\gamma}$ is a feasible point of $W_t(.)$. Exploiting that $W(\vec{\alpha}) \geq W(\vec{\beta})$ by definition, and $W_t(\vec{\alpha}^t) \geq W^t(\vec{\gamma})$, substitution into (12) leads to

$$\alpha_t y_t \left[\sum_{i\in SV^t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t,\vec{x}_i) + b^t\right] \geq \alpha_t y_t \left[\sum_{i\in SV\backslash\{t\}} \alpha_i y_i \mathcal{K}(\vec{x}_t,\vec{x}_i) + b\right]$$

$$-\frac{1}{2}\sum_{i\in SV\backslash\{t\}}\sum_{j\in SV\backslash\{t\}} \mu_i\mu_j\mathcal{K}(\vec{x}_i,\vec{x}_j) + \alpha_t\sum_{i\in SV\backslash\{t\}} \mu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) \quad (14)$$

$$-\frac{1}{2}\sum_{i\in SV^t}\sum_{j\in SV^t} \nu_i\nu_j\mathcal{K}(\vec{x}_i,\vec{x}_j) + \alpha_t\sum_{i\in SV^t} \nu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) \quad (15)$$

The term $\alpha_t \sum_{i\in SV^t} \nu_i\mathcal{K}(\vec{x}_i,\vec{x}_t)$ is non-negative, since all ν_i and $\mathcal{K}(\vec{x}_i,\vec{x}_t)$ are non-negative. The same holds for $\alpha_t \sum_{i\in SV\backslash\{t\}} \mu_i\mathcal{K}(\vec{x}_i,\vec{x}_t)$. Furthermore, $\frac{1}{2}\sum_{i\in SV\backslash\{t\}}\sum_{j\in SV\backslash\{t\}} \mu_i\mu_j\mathcal{K}(\vec{x}_i,\vec{x}_j) \leq \frac{1}{2}\alpha_t^2R^2$ and $\frac{1}{2}\sum_{i\in SV^t}\sum_{j\in SV^t} \nu_i\nu_j\mathcal{K}(\vec{x}_i,\vec{x}_j) \leq \frac{1}{2}\alpha_t^2R^2$, since the $K(\vec{x}_i,\vec{x}_j)$ form a positive semi-definite matrix with the diagonal elements bounded from above by R^2. Using these inequalities and dividing by α_t, it is possible to write $y_t\left[\sum_{i\in SV^t} \alpha_i^t y_i\mathcal{K}(\vec{x}_t,\vec{x}_i) + b^t\right] \geq y_t\left[\sum_{i\in SV\backslash\{t\}} \alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i) + b\right] - \alpha_t R^2$. After adding $y_t\alpha_t y_t\mathcal{K}(\vec{x}_t,\vec{x}_t)$ and exploiting that $y_t\left[\sum_{i\in SV} \alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i) + b\right] = 1$ for support vectors one can see that a leave-one-out error can occur only when $2\alpha_t R^2 \geq 1$. Note that $\xi_i = 0$ for support vectors.

Case $\alpha_t = C$: Example $(\vec{x}_t, y_t)$ is a bounded support vector. The argumentation follows that in the case of unbounded support vectors. The only difference is that $y_t\left[\sum_{i\in SV} \alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i) + b\right] = 1-\xi_t$ for bounded support vectors, leading to the condition $2\alpha_t R^2 + \xi_t \geq 1$. ∎

The idea of connecting the leave-one-out error with properties of the solution vector $\vec{\alpha}$ goes back to Vapnik (1998, pages 418-421). Unlike the work presented here, Vapnik's result is limited to the case where (a) the training data is separable, (b) the special case of hyperplanes passing through the origin, and (c) it is used to derive bounds on the expected error, not estimators. Jaakkola and Haussler (1999) present a generalized bound for inseparable data that is similar to that of Lemma 1. Nevertheless, like Vapnik's bound it is restricted to hyperplanes passing through the origin and does not apply to regular SVMs. Approximations to the leave-one-out error of SVMs without guaranteeing an upper bound were recently proposed by Wahba (1999) and Opper and Winther (in press). An additional difference is that their approach requires computing the inverse for part of the Hessian making it computationally more expensive.

While Lemma 1 is valid for all kernel functions that return positive values, it is tightest when the minimum value is zero. The following lemma shows that this can always be achieved.

Lemma 2 *The soft-margin SVM is invariant under addition of a real value c to the kernel function.*

The proof is given in (Joachims, 1999a). The previous two lemmas complete the tools needed to characterize the bias of the $\xi\alpha$-estimator of the error rate.

Theorem 1 *The $\xi\alpha$-estimator of the error rate is pessimistically biased in the following sense*

$$\mathcal{E}(Err^n_{\xi\alpha}(h_\mathcal{L})) \geq \mathcal{E}(Err^{n-1}(h_\mathcal{L})) \quad (16)$$

The expectation on the left hand side is over training sets of size n, the one on the right hand side is over training sets of size $n-1$.

Proof A theorem of Lunts and Brailovskiy (1967) shows that the leave-one-out estimator $Err^n_{loo}(h_\mathcal{L})$ of the error rate on training sets of size n gives an unbiased estimate of the error rate after training on $n-1$ examples. After adding a constant c to the kernel function so that $min\mathcal{K}(\vec{x}_i,\vec{x}) = 0$, the theorem follows directly from the bound in Lemma 1 using Lemma 2. Lemma 1 establishes that $Err^n_{\xi\alpha}(h_\mathcal{L}) \geq Err^n_{loo}(h_\mathcal{L})$ with $R^2 = c + max\mathcal{K}(\vec{x},\vec{x})$ and therefore $\mathcal{E}(Err^n_{\xi\alpha}(h_\mathcal{L})) \geq \mathcal{E}(Err^{n-1}(h_\mathcal{L}))$. ∎

In other words, the theorem states that the $\xi\alpha$-estimator tends to overestimate the true error rate. This means that "on average" the estimate is higher than the true error. Given a low variance of the estimate it is now possible to guarantee with a certain probability that the true error is lower than the estimate. Nevertheless, known bounds on the variance depend on further assumptions about the learner and/or the learning task. Bounds in (Devroye & Wagner, 1976) require that the probability $\Pr(h^n_\mathcal{L}(\vec{x}) \neq h^{\backslash i}_\mathcal{L}(\vec{x}))$ is small. For SVMs this quantity depends on the learning task. Bounds on the variability of the leave-one-out estimator presented by Kearns and Ron (1997) are independent of the learning task. Nevertheless, their bounds would be too loose to be of practical importance. Therefore, the variability of the $\xi\alpha$-estimator will be assessed empirically in section 5.

4.2 Recall, Precision, and F_1

With similar arguments as for the error rate, one can derive $\xi\alpha$-estimators of the recall, the precision, and the $F1$.

Definition 2 *For stable soft-margin SVMs, the $\xi\alpha$-estimators of the recall, the precision, and the $F1$ are*

$$Rec^n_{\xi\alpha}(h_{\mathcal{L}}) = 1 - \frac{d_+}{n_+} \quad (17)$$

$$Prec^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{n_+ - d_+}{n_+ - d_+ + d_-} \quad (18)$$

$$F1^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{2\,n_+ - 2\,d_+}{2\,n_+ - d_+ + d_-} \quad (19)$$

$$\textit{using}\quad d_+ = |\{i : y_i = 1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \quad (20)$$

$$d_- = |\{i : y_i = -1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \quad (21)$$

$$n_+ = |\{i : y_i = 1\}| \quad (22)$$

with ρ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of OP1 and OP2 on the training set S_n. R^2_Δ is an upper bound on $\mathcal{K}(\vec{x},\vec{x}) - \mathcal{K}(\vec{x},\vec{x}')$ for all $\vec{x},\vec{x}'$.

d_+ (d_-) is the number of positive (negative) training examples for which the inequality $(\rho\alpha_i R^2_\Delta + \xi_i) \geq 1$ holds. n_+ is the number of positive examples in the training sample. Using the same arguments as in Lemma 1, d_+ (d_-) is an upper bound on the number of positive (negative) examples that produce a leave-one-out error.

Using the common definition of bias to characterize the $\xi\alpha$-estimator of the recall is difficult. The recall estimate depends on the number of positive training examples in a non-linear way. The situation is even worse for the precision. Given a decision rule that classifies all examples into the negative class with probability one, the precision is not defined at all. Researchers in information retrieval have worked around this problem by differentiating between micro-averaging and macro-averaging. Macro-expectation, the analog of macro-averaging, corresponds to the conventional expected value. In micro-averaging the arguments (i.e. the elements of the contingency tables) are averaged before the function is applied. This removes the artifacts discussed above. The micro-expected value $\mathcal{E}_{micro}(f(X))$ of a function $f(x)$ is defined as $\mathcal{E}_{micro}(f(X)) = f(\mathcal{E}(X))$. The random variable X can be a vector. The bias of the $\xi\alpha$-estimators in terms of a micro-expectation is characterized by the following theorem.

Theorem 2 *The $\xi\alpha$-estimators of the recall, the precision, and the $F1$ are pessimistically biased in the following sense:*

$$\mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(Rec^{n-1}(h_{\mathcal{L}})) \quad (23)$$

$$\mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(Prec^{n-1}(h_{\mathcal{L}})) \quad (24)$$

$$\mathcal{E}_{micro}(F1^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(F1^{n-1}(h_{\mathcal{L}})) \quad (25)$$

The proof is given in (Joachims, 1999a). The theorem shows that the $\xi\alpha$-estimates are "on average" lower than the true recall, precision, and $F1$ in terms of a micro-average. A similar result for the strong (macro) definition of bias requires further assumptions.

5. Experiments

The following experiments explore how well the $\xi\alpha$-estimators work in practice. The evaluation is done on the three text classification tasks Reuters, WebKB, and Ohsumed. Due to space constraints, here only the results for Reuters are discussed in detail. The results for WebKB and Ohsumed can be found in (Joachims, 1999a).

The Reuters-21578 dataset[2] was collected from the Reuters newswire in 1987. The "ModApte" subset is used, leading to a corpus of 12,902 documents. Of the 135 potential topic categories only the most frequent 10 are used, while keeping all documents.

Two values for the parameter ρ are evaluated in the following, namely $\rho = 2$ and $\rho = 1$. The setting $\rho = 2$ is a direct consequence of Lemma 1, while the setting $\rho = 1$ is suggested as a better choice for text classification by the following argument. The factor $\rho = 2$ in the $\xi\alpha$-estimates was introduced to upper bound the expressions (14) and (15) in the proof of Lemma 1. In the worst case, each expression can be $\frac{1}{2}\alpha_t^2 R^2$ as argued above. For this worst case to happen, it is necessary that all support vectors have identical feature vectors $\vec{x}_i$. For text classification problems the opposite is true. Many support vectors are almost orthogonal. Consequently the expressions in (14) and (15) can be kept close to zero for the linear kernel and suitable $\vec{\nu},\vec{\mu}$, leading to a $\xi\alpha$-estimate with $\rho \approx 1$ instead of $\rho = 2$.

Unless noted otherwise, the following results are averages over 10 random test/training splits. Training and test set are designed to be of equal size to be able to compare variance estimates. The test set is used to get a holdout estimate as an approximation to the true parameter. For simplicity reasons, all results in this section are for linear SVMs with $C = 0.5$. This value of C is chosen since it was selected by the $\xi\alpha$-estimates in model selection experiments (Joachims, 2000). Note that $R_\Delta = 1$, since document vectors are normalized to unit length.

[2] www.research.att.com/~lewis/reuters21578.html

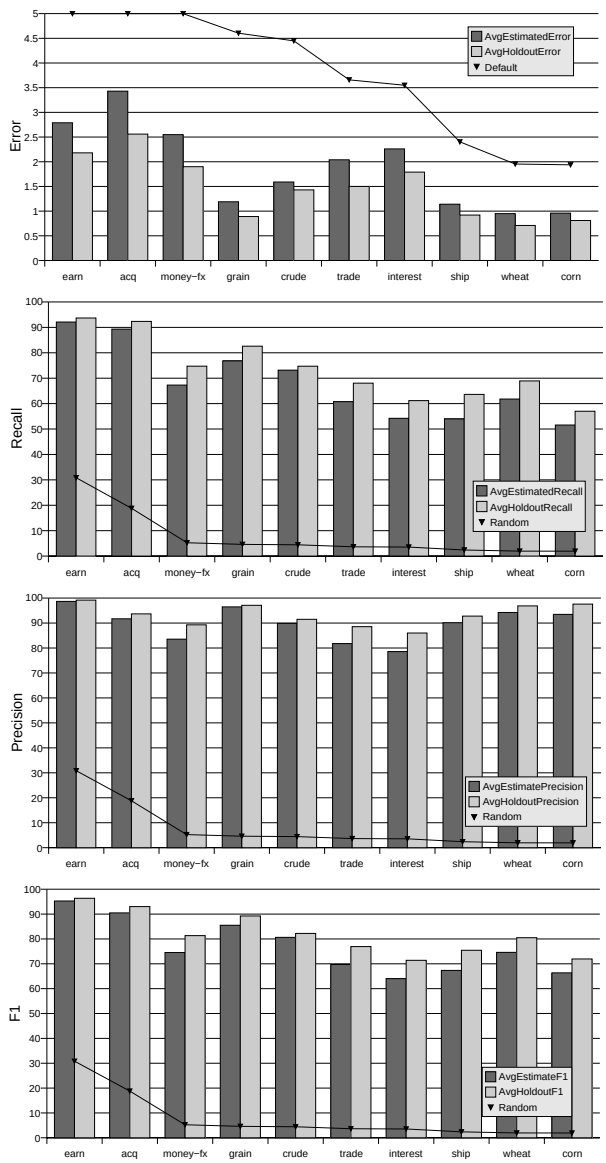


Figure 1. Diagrams comparing average $\xi\alpha$-estimate ($\rho = 1$) of the error, the recall, the precision, and the $F1$-measure with the average true error, true recall, true precision, and true $F1$ measured on a holdout set for the ten most frequent Reuters categories.

5.1 How Large are Bias and Variance of the $\xi\alpha$-Estimators?

Figure 1 illustrates the results for the Reuters dataset with $\rho = 1$. The findings are as expected. The $\xi\alpha$-estimators slightly overestimate the true error and underestimate precision, recall, and $F1$. Nevertheless, they accurately reflect the relative performance between categories. Table 1 gives additional details on the results. The top half of the table contains the $\xi\alpha$-estimates with $\rho = 1$, the lower half with $\rho = 2$. For $\rho = 2$ the $\xi\alpha$-estimates are substantially more biased than for $\rho = 1$. After each average the table includes an estimate of the standard deviation. The standard deviation of the $\xi\alpha$-estimates is very similar to that of the holdout estimates, especially for $\rho = 1$. This shows that in terms of variance the $\xi\alpha$-estimates are as good as holdout testing without requiring an additional test set of the same size as the training data.

5.2 What is the Influence of the Training Set Size?

The results presented so far were for a large training set. Do the estimators work for smaller training sets as well? Figure 2 shows learning curves for the Reuters categories "earn", "acq", and "money-fx". To save space, only error rate and $F1$ are plotted, since precision and recall behave similar to $F1$. The top two curves of each graph show the average $\xi\alpha$-estimate and the average holdout-estimate. The averages are over 20 random training/test splits. Except for very small training sets, the graphs show no strong systematic connection between bias (i.e. the difference between the top two curves of each graph) and the training set size. For "acq" and "money-fx" with small training sets, the SVM behaves almost like the default classifier that assigns all test examples to the more populous class. In this situation the average $\xi\alpha$-estimate and the holdout-estimate are almost equal. In terms of variance, the training set size has a strong influence on both the holdout-estimate and the $\xi\alpha$-estimate. The bottom two curves of each graph show the empirical standard deviation of each estimator. As expected, the variance increases with decreasing training set size. Nevertheless, when moving to very small training sets, the variance decreases again. This is a consequence of the SVM behaving more and more like the default classifier. Interestingly, the variance curves of the $\xi\alpha$-estimator are very similar to those of the holdout-estimator. This confirms that the $\xi\alpha$-estimators have approximately the same variance as holdout testing.

5.3 Do the Findings Transfer to Other Text Classification Tasks?

To make sure that the $\xi\alpha$-estimator are not tailored to the properties of the Reuters dataset, but apply to a wide range of text classification tasks, similar experiments were also conducted for the WebKB and the Ohsumed data (reported in (Joachims, 1999a)). For both collections, the results are qualitatively the same as for Reuters. The $\xi\alpha$-estimates with $\rho = 1$ are preferable over those with $\rho = 2$. The $\xi\alpha$-estimates with $\rho = 1$ exhibit a moderate pessimistic bias and a relatively low variance.

Table 1. Table comparing average $\xi\alpha$-estimates with the average true performance for the ten most frequent Reuters categories. The upper half shows the estimates for $\rho = 1$, the lower half for $\rho = 2$. The "true" values are estimated from a holdout set of the same size as the training set (6451 examples each). All values are averaged over 10 random test/training splits exhibiting the standard deviation printed after each average.

$\rho = 1$	$Err_{\xi\alpha}(h_{\mathcal{L}})$	$Err(h_{\mathcal{L}})$	$Rec_{\xi\alpha}(h_{\mathcal{L}})$	$Rec(h_{\mathcal{L}})$	$Prec_{\xi\alpha}(h_{\mathcal{L}})$	$Prec(h_{\mathcal{L}})$	$F1_{\xi\alpha}(h_{\mathcal{L}})$	$F1(h_{\mathcal{L}})$
earn	2.68 ± 0.08	1.87 ± 0.15	92.3 ± 0.3	94.8 ± 0.5	98.8 ± 0.1	99.1 ± 0.1	95.4 ± 0.1	96.9 ± 0.2
acq	3.54 ± 0.16	2.53 ± 0.20	88.1 ± 0.8	92.5 ± 0.5	92.1 ± 0.4	93.6 ± 0.7	90.0 ± 0.5	93.1 ± 0.4
money-fx	2.67 ± 0.12	1.92 ± 0.14	64.9 ± 2.0	73.6 ± 2.4	83.4 ± 1.4	89.8 ± 1.9	73.0 ± 1.4	80.9 ± 1.4
grain	1.23 ± 0.09	0.82 ± 0.11	74.4 ± 1.7	82.7 ± 2.1	98.2 ± 0.8	98.5 ± 0.5	84.7 ± 1.2	89.9 ± 1.2
crude	1.58 ± 0.08	1.30 ± 0.10	72.5 ± 1.7	76.6 ± 2.6	90.9 ± 1.0	92.7 ± 1.2	80.6 ± 1.0	83.9 ± 1.3
trade	1.99 ± 0.11	1.42 ± 0.10	60.4 ± 2.1	70.0 ± 2.4	83.5 ± 1.3	89.0 ± 1.5	70.0 ± 1.7	78.3 ± 1.6
interest	2.20 ± 0.18	1.67 ± 0.20	54.0 ± 5.2	63.8 ± 4.3	80.2 ± 2.9	87.2 ± 2.8	64.5 ± 4.4	73.6 ± 3.4
ship	1.23 ± 0.10	0.96 ± 0.13	47.5 ± 5.6	61.2 ± 3.9	93.3 ± 1.9	93.8 ± 2.5	62.7 ± 5.0	74.0 ± 2.9
wheat	0.91 ± 0.07	0.65 ± 0.08	62.8 ± 1.8	71.1 ± 2.3	95.0 ± 1.5	97.8 ± 1.0	75.6 ± 1.2	82.3 ± 1.4
corn	0.90 ± 0.05	0.71 ± 0.06	52.4 ± 4.1	61.8 ± 1.7	97.3 ± 1.8	99.1 ± 0.6	68.1 ± 3.6	76.1 ± 1.3
$\rho = 2$	$Err_{\xi\alpha}(h_{\mathcal{L}})$	$Err(h_{\mathcal{L}})$	$Rec_{\xi\alpha}(h_{\mathcal{L}})$	$Rec(h_{\mathcal{L}})$	$Prec_{\xi\alpha}(h_{\mathcal{L}})$	$Prec(h_{\mathcal{L}})$	$F1_{\xi\alpha}(h_{\mathcal{L}})$	$F1(h_{\mathcal{L}})$
earn	6.79 ± 0.21	1.87 ± 0.15	84.4 ± 0.4	94.8 ± 0.5	92.7 ± 0.5	99.1 ± 0.1	88.3 ± 0.4	96.9 ± 0.2
acq	12.99 ± 0.28	2.53 ± 0.20	59.3 ± 1.4	92.5 ± 0.5	66.0 ± 1.2	93.6 ± 0.7	62.5 ± 1.2	93.1 ± 0.4
money-fx	5.38 ± 0.22	1.92 ± 0.14	38.4 ± 2.0	73.6 ± 2.4	52.1 ± 1.8	89.8 ± 1.9	44.2 ± 1.8	80.9 ± 1.4
grain	3.20 ± 0.19	0.82 ± 0.11	48.1 ± 2.0	82.7 ± 2.1	72.8 ± 1.8	98.5 ± 0.5	57.9 ± 2.0	89.9 ± 1.2
crude	3.69 ± 0.20	1.30 ± 0.10	45.3 ± 2.1	76.6 ± 2.6	62.8 ± 2.4	92.7 ± 1.2	52.6 ± 2.1	83.9 ± 1.3
trade	3.80 ± 0.15	1.42 ± 0.10	34.7 ± 1.8	70.0 ± 2.4	51.2 ± 2.2	89.0 ± 1.5	41.4 ± 1.9	78.3 ± 1.6
interest	4.19 ± 0.26	1.67 ± 0.20	27.5 ± 4.2	63.8 ± 4.3	40.8 ± 5.0	87.2 ± 2.8	32.8 ± 4.6	73.6 ± 3.4
ship	2.21 ± 0.08	0.96 ± 0.13	18.2 ± 2.6	61.2 ± 3.9	49.4 ± 5.2	93.8 ± 2.5	26.6 ± 3.5	74.0 ± 2.9
wheat	1.79 ± 0.14	0.65 ± 0.08	43.2 ± 1.8	71.1 ± 2.3	65.8 ± 2.7	97.8 ± 1.0	52.2 ± 1.9	82.3 ± 1.4
corn	1.72 ± 0.12	0.71 ± 0.06	28.1 ± 2.1	61.8 ± 1.7	57.1 ± 3.2	99.1 ± 0.6	37.6 ± 2.2	76.1 ± 1.3

6. Summary and Conclusions

This paper proposed an approach to estimating the generalization performance of a SVM without any computation-intensive resampling. The new estimators are much more efficient than cross-validation or bootstrapping, since they can be computed immediately after training a single SVM. Moreover, the estimators developed here address the special measures used to evaluate text classification performance.

The theoretical analysis of the estimators shows that they tend to be conservative. This is a desirable property for practical applications, since they are less likely to falsely predict a high generalization performance. In addition to the theoretical analysis, the bias and the variance of the estimates are evaluated experimentally. As predicted by the theory, the empirical results show a conservative bias for all $\xi\alpha$-estimators. Typically, the bias is acceptably low and the variance of the $\xi\alpha$-estimates is essentially as low as that of a holdout estimator which has access to twice as much labeled data. The $\xi\alpha$-estimators are therefore a suitable method for estimating the performance of SVMs on text classification tasks. They make efficient use of the data and they are computationally efficient.

Currently, the $\xi\alpha$-estimators are applied to model selection for text classification (Joachims, 2000). They can effectively select between different preprocessing steps (e.g. stemming or no stemming), kernel parameters, and good values for C. Similarly, they can be used to detect concept drift (Klinkenberg & Joachims, 2000). An open questions is whether the bias can be removed with only a modest increase of computational expense. One approach could be to integrate actual leave-one-out testing into the optimization process considering only those examples for which $\rho\alpha R_{\Delta}^2 + \xi > 1$.

Acknowledgments

This work was supported by the DFG Collaborative Research Center on Complexity Reduction in Multivariate Data (SFB475).

References

Burges, C., & Crisp, D. (1999). Uniqueness of the SVM solution. *Proceedings of the Twelfth Conference on Neural Information Processing Systems*. Cambridge, MA: MIT Press.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. Springer.

Devroye, L., & Wagner, T. (1976). A distribution-free performance bound in error estimation. *IEEE Transactions on Information Theory, 22*, 586–587.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *Proceedings of the Eighth Conference on Information and Knowledge Management* (pp. 148–155). New York: ACM Press.

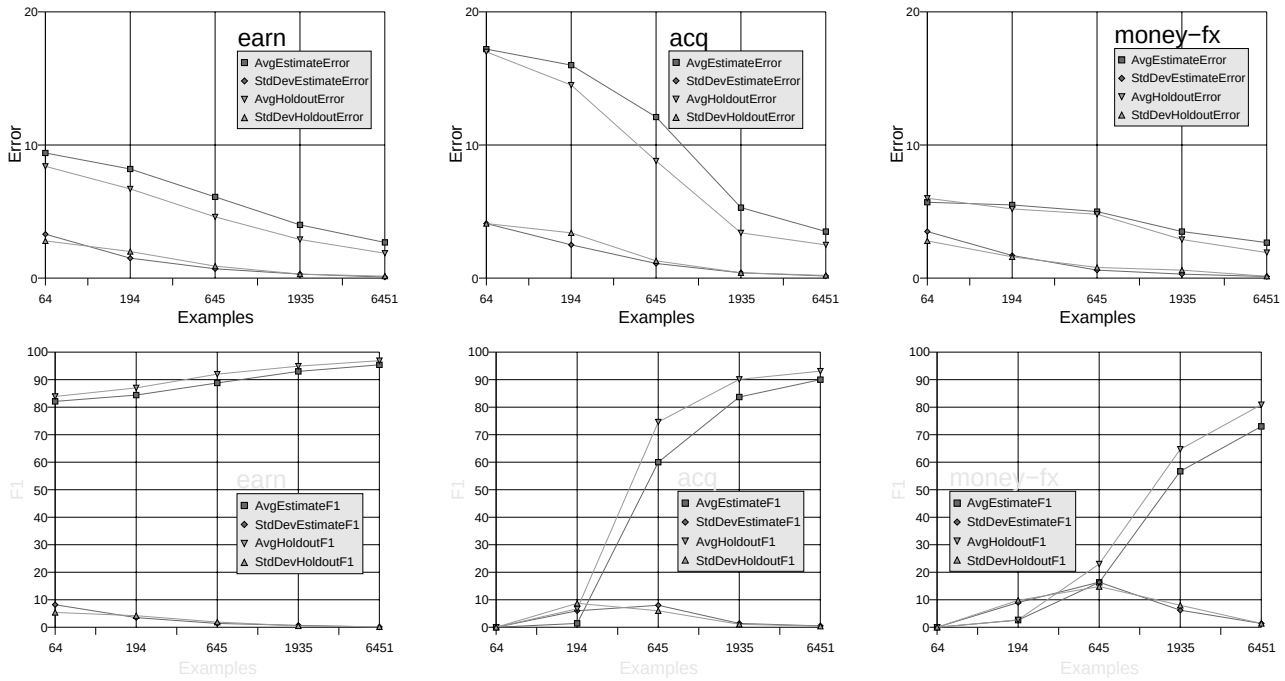


Figure 2. Learning curves for the Reuters categories "earn", "acq", and "money-fx" comparing the $\xi\alpha$-estimator of the error rate (upper row) and the $F1$ (lower row) with holdout testing. The x-axis denotes the size of the training set. Each test set for holdout testing contains as many examples as the corresponding training set. All values are averages over ten random test/training splits. The upper curves show the average, the lower curves show the standard deviation.

Jaakkola, T., & Haussler, D. (1999). Probabilistic kernel regression models. *Proceedings of the Seventh Workshop on AI and Statistics*. San Francisco: Morgan Kaufman.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proceedings of the Tenth European Conference on Machine Learning* (pp. 137 – 142). Berlin: Springer.

Joachims, T. (1999a). *Estimating the generalization performance of a SVM efficiently* (LS VIII-Report 25). Universität Dortmund, Germany.

Joachims, T. (1999b). Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods - Support vector learning*. Cambridge, MA: MIT Press.

Joachims, T. (2000). *Efficient model selection for text classification with support vector machines* (LS VIII-Report 26). Universität Dortmund, Germany.

Kearns, M., & Ron, D. (1997). Algorithmic stability and sanity-check bounds for leave-one-out cross validation. *Proceedings of the Tenth Conference on Computational Learning Theory* (pp. 152–162). New York: ACM Press.

Klinkenberg, R., & Joachims, T. (2000). Detecting concept drift with support vector machines. *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufman.

Lunts, A., & Brailovskiy, V. (1967). Evaluation of attributes obtained in statistical decision rules. *Engineering Cybernetics, 3*, 98–109.

Opper, M., & Winther, O. (in press). Gaussian process classification and SVM: Mean field results and leave-one-out estimator. In P. Bartlett, B. Schölkopf, D. Schuurmans, & A. Smola (Eds.), *Large margin classifiers*. Cambridge, MA: MIT Press.

Rifkin, R., Pontil, M., & Verri, A. (1999). A note on support vector machine degeneracy. *Proceedings of the Tenth Conference on Algorithmic Learning Theory*. Berlin: Springer.

Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management, 24*, 513–523.

Vapnik, V. (1998). *Statistical learning theory*. Chichester, UK: Wiley.

Wahba, G. (1999). Support vector machines, reproducing kernel hilbert spaces, and randomized gacv. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods - Support vector learning*. Cambridge, MA: MIT Press.

Multi-class Classification with Error Correcting Codes

Jörg Kindermann and Edda Leopold and Gerhard Paass
GMD – German National Research Center for Information Technology
D-52754 Sankt Augustin

Abstract. Automatic text categorization has become a vital topic in many applications. Imagine for example the automatic classification of Internet pages for a search engine database. The traditional 1-of-n output coding for classification scheme needs resources increasing linearly with the number of classes. A different solution uses an error correcting code, increasing in length with $\mathcal{O}(log_2(n))$ only. In this paper we investigate the potential of error correcting codes for text categorization with many categories. The main result is that multi-class codes have advantages for classes which comprise only a small fraction of the data.

Keywords. multi-class classification, error correcting codes, support vector machine, text categorization

1 Introduction

With the advent of the Internet, automatic text categorization has become a vital topic in many applications. Imagine for example the automatic classification of Internet pages for a search engine database. There exist promising approaches to this task, among them, support vector machines (SVM) [Joa98] are one of the most successful solutions. One remaining problem is however that SVM can only separate two classes at a time. Thus the traditional 1-of-n output coding scheme applied in this case needs resources increasing linearly: n classes will need n classifiers to be trained independently. Alternative solutions were published by Vapnik [Vap98, p438], Guermeur et al. [GEPM00], Nakajima et al. [NPP00], and Dietterich and Bakiri [DB95].

Dietterich and Bakiri use a distributed output code. Because the output code has more bits than needed to represent each class as unique pattern, the additional bits may be used to correct classification errors. In this paper we investigate the potential of error correcting codes for text categorization with many categories. As a benchmark data set we use the Reuters-21578 dataset. We took Reuters texts from 31 different topics and classified them, using a number of error correcting codes of various lengths. The results are compared to the standard 1-of-31 approach to classification.

2 Methods

2.1 Error correcting codes

Classification with error correcting codes can be seen "as a kind of communications problem in which the identity of the correct output class for a new example is being transmitted over a channel which consists of the input features, the training examples, and the learning algorithm" ([DB95, p266]). The classification of a new set of input features can no longer be determined from the output of one classifier. It is coded in a distributed representation of l outputs from *all* classifiers. Table 2.1 shows an example of an error correcting code for $n = 8$ classes with $l = 5$ code words. The code words are the columns of the table. Each classifier has to learn one of the code words. This means, that the classifier should output a 1 for all input data belonging to one of the classes which are assigned 1 in the classifier's code word, and 0 in all other cases. The code for a specific class is to be found in the row of the table which is assigned to the class. The code length in bits is the number of code words, 5 in our example.

Table 1 **Error correcting code for $n = 8$ classes with $l = 5$ code words.**

	code word				
class	1	2	3	4	5
1	0	0	0	1	1
2	0	0	1	0	1
3	0	1	0	0	1
4	0	1	1	0	0
5	1	0	0	1	0
6	1	1	0	1	1
7	1	1	1	0	1
8	1	1	1	1	0

Noise is introduced by the learning set, choice of features, and flaws of the learning algorithm. Noise may induce classification errors. But if there are more code words than needed to distinguish n classes, i.e. $l > log_2(n)$, we can use the additional bits to correct errors: If the output code does not match exactly one of the classes, take the class with minimal Hamming distance. If the minimum Hamming distance between class codes is d, we can in this way correct at least $\frac{d-1}{2}$ single bit errors (see [DB95, p.266]). It is therefore important to use codes with maximized Hamming distance for the classes.

There are several potential advantages in this approach: The number of classifiers needed increases with $\mathcal{O}(log_2(n))$ only, and additional bits can be used for error correction.

Dietterich and Bakiri used codes with $l >> n$. The codes therefore were much longer than actually needed to produce n different bit strings. Because we wanted to investigate the properties of codes with $log_2(n) < l \leq n$, we did not use the optimization criteria given by Dietterich and Bakiri to find optimal codes. We used simulated annealing instead to optimize a mix of Hamming distances of class codes, and of code words. We wanted to distinguish each of n classes against the other classes *plus* other data not belonging to any of the classes. Therefore we also optimized the codes with respect to all-zero outputs, representing the latter data. This is the reason why we used $31 = 2^5 - 1$ classes in our experiments.

Table 2 **Minimal inter-row and inter-column Hamming distances for our codes.**

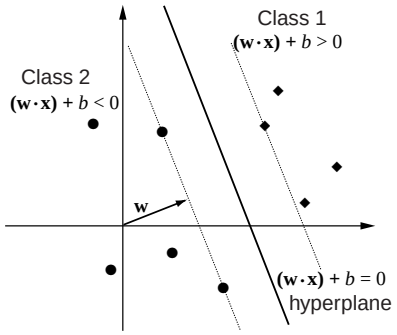
# code words	min. distance btw. class codes	code words
8	2	14
10	3	16
12	4	14
16	6	14
24	10	14
1-of-n	1	1

See table 2.1 for the Hamming distances of a set of codes we used. In a code matrix of size 2^l, i.e. a matrix with 2^l code words, there are always 31 class codes plus the 0-code. For larger numbers of code words, i.e. increasing l, the hamming distances grow. The reason is that the number of class codes to be generated remains constant ($= 32$) and therefore we have more degrees of freedom to place the bits in the class codes. The Hamming distances of code words decrease with increasing l. Here, we have constant length of code words ($= 32$), but increasing numbers of them. Therefore we have decreasing degrees of freedom to design the code words.

2.2 Support vector machines

Support Vector Machines (SVM) recently gained popularity in the learning community [Vap98]. In its simplest linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum interclass distance, the *margin*. Figure 1 shows such a hyperplane with the associated margin.

Figure 1 **Hyperplane with maximal margin generated by a linear SVM**



The formula for the output of a linear SVM is

$$u = w \cdot x + b \tag{1}$$

where w is the normal vector to the hyperplane, and x is the input vector. The margin is defined by the distance of the hyperplane to the nearest of the positive and negative examples. Maximizing the margin can be expressed as an optimization problem:

$$\text{minimize } \frac{1}{2} \|w\|^2 \text{ subject to } y_i(w \cdot x_i + b) \geq 1, \forall_i \tag{2}$$

where x_i is the i-th training example and $y_i \in \{-1, 1\}$ is the correct output of the SVM for the i-th training example. Note that the hyperplane is only determined by the training instances x_i on the margin, the *support vectors*. Of course, not all problems are linearly separable. Cortes and Vapnik [CV95] proposed a modification to the optimization formulation that allows, but penalizes, examples that fall on the wrong side of the decision boundary.

The SVM can be extended to nonlinear models by mapping the input space into a very high-dimensional feature space chosen a priori. In this space the optimal separating hyperplane is constructed [BGV92] [Vap98, p.421].

The distinctive advantage of the SVM for text categorization is its ability to process many thousand different inputs. This opens the opportunity to use all *words* in a text directly as features. For each word w_i the number of times of occurrence is recorded. Joachims [Joa98] used the SVM for the classification of text into different topic categories. Dumais et al. [DPHS98] use linear SVM for text categorization because they are both accurate and fast. They are 35 times faster to train than the next most accurate (a decision tree) of the tested classifiers. They apply SVM to the Reuters-21578 collection, e-mails and web pages. Drucker at al. [DWV99] classify e-mails as spam and non spam. They find that boosting trees and SVM have similar performance in terms of accuracy and speed, but SVM train significantly faster.

Table 3 **Table of tested combinations of frequency transformations and SVM kernels**

#	abbreviation	coding; kernel
1	relFreq0	rf & L_1; linear
2	relFreq1-d 2	rf & L_1; quadr. poly.
3	relFreq1-d 3	rf & L_1; cubic poly.
4	relFreq2-g 1	rf & L_1; rbf
5	relFreqImp0	rf & L_1 & imp; linear
6	relFreqImp1-d 2	rf & L_1 & imp; quadr. poly.
7	relFreqImp1-d 3	rf & L_1 & imp; cubic poly.
8	relFreqImp2-g 1	rf & L_1 & imp; rbf
9	relFreqL20	rf & L_2; linear
10	relFreqL21-d 2	rf & L_2; quadr. poly.
11	relFreqL21-d 3	rf & L_2; cubic poly.
12	relFreqL22-g 1	rf & L_2; rbf
13	relFreqL2Imp0	rf & L_2 & imp; linear
14	relFreqL2Imp1-d 2	rf & L_2 & imp; quadr. poly.
15	relFreqL2Imp1-d 3	rf & L_2 & imp; cubic poly.
16	relFreqL2Imp2-g 1	rf & L_2 & imp; rbf
17	logRelFreq0	log rf & L_1; linear
18	logRelFreq1-d 2	log rf & L_1; quadr. poly.
19	logRelFreq1-d 3	log rf & L_1; cubic poly.
20	logRelFreq2-g 1	log rf & L_1; rbf
21	logRelFreqImp0	log rf & L_1 & imp; linear
22	logRelFreqImp1-d 2	log rf & L_1 & imp; quadr. poly.
23	logRelFreqImp1-d 3	log rf & L_1 & imp; cubic poly.
24	logRelFreqImp2-g 1	log rf & L_1 & imp; rbf
25	logRelFreqL20	log rf & L_2; linear
26	logRelFreqL21-d 2	log rf & L_2; quadr. poly.
27	logRelFreqL21-d 3	log rf & L_2; cubic poly.
28	logRelFreqL22-g 1	log rf & L_2; rbf
29	logRelFreqL2Imp0	log rf & L_2 & imp; linear
30	logRelFreqL2Imp1-d 2	log rf & L_2 & imp; quadr. poly.
31	logRelFreqL2Imp1-d 3	log rf & L_2 & imp; cubic poly.
32	logRelFreqL2Imp2-g 1	log rf & L_2 & imp; rbf
33	tfidf0	tfidf & L_1; linear
34	tfidf1-d 2	tfidf & L_1; quadr. poly.
35	tfidf1-d 3	tfidf & L_1; cubic poly.
36	tfidf2-g 1	tfidf & L_1; rbf
37	tfidfL20	tfidf & L_2; linear
38	tfidfL21-d 2	tfidf & L_2; quadr. poly.
39	tfidfL21-d 3	tfidf & L_2; cubic poly.
40	tfidfL22-g 1	tfidf & L_2; rbf

2.3 Transformations of frequency vectors

As lexical units scale to different order of magnitude in larger documents, it is interesting to examine how term frequency information can be mapped to quantities which can efficiently be processed by SVM. For our empirical tests we use different transformations of type frequencies: *relative frequencies* and *logarithmic frequencies*.

As the simplest "frequency-transformation" we use the term-frequencies $f(w_k, d_i)$ themselves. The frequencies are multiplied by one of the importance weights described below and normalized to unit length. Relative frequencies $f(w_k, d_i)$ with importance weight idf normalized with respect to L_2-norm have been used by Joachims [Joa98] and others. We also tested other combinations as for example relative frequencies with no importance weight normalized with respect to L_1-norm, which is defined by $F_{rel_1}(w_k, d_i) = \frac{f(w_k,d_i)}{f(d_i)}$

The second transformation is the logarithm. We consider this transform because it is a common approach in quantitative linguistics to consider logarithms of linguistic quantities rather than the quantities themselves. Here we also normalize to unit length with respect to L_1 and L_2. We define $F_{log}(w_k, d_i) = \log\big(1+f(w_k, d_i)\big)$, combine F_{log} different importance weights, and normalize the resulting vector with respect to L_1 and L_2.

2.4 Importance weights

Importance weights are often used in order to reduce dimensionality of a learning task. Common importance weights like inverse document frequency (idf) originally have been designed for identifying index words.

However, since SVM are capable to manage a large number of dimensions ([KL00] used more than 400000 input features), reduction of dimensionality is not necessary and importance weights can be used to quantify how a specific given type is to the documents of a text collection. A type which is evenly distributed across the document collection will be given a low importance weight because it is judged to be less specific for the documents it occurs in, than a type which is used in a only a few documents. The importance weight of a type is multiplied by its transformed frequency of occurrence. So each of the importance weights described below can be combined with each of the frequency transformations described in section (2.3). We examined the performance of the following settings *no importance weight*, *inverse document frequency (idf)*, and *redundancy*. Inverse document frequency (idf) is commonly used when SVM is

applied to text classification. Idf is defined by $\log \frac{n}{df_k}$, where df_k is the number of those documents in the collection, which contain the term w_k and n is the number of all documents in the collection.

Redundancy quantifies the skewness of a probability distribution. We consider the empirical distribution of each type over the different documents and define an importance weight by

$$R = H_{max} - H = \quad (3)$$

$$\log n + \sum_{i=1}^{n} \frac{f(w_k, d_i)}{f(w_k)} \log \frac{f(w_k, d_i)}{f(w_k)} \quad (4)$$

where $f(w_k, d_i)$ is the frequency of occurrence of term w_k in document d_i and n is the number of documents in the collection. R is a measure of how much the distribution of a term w_k in the various documents deviates from the uniform distribution.

2.5 Kernel functions

It is well known that the choice of the kernel function is crucial to the efficiency of support vector machines. Therefore the data transformations described above were combined with three different kernel functions:

- linear $K(\vec{x}, \vec{x}') = \vec{x} \cdot \vec{x}'$
- 2nd ord. poly. $K(\vec{x}, \vec{x}') = (\vec{x} \cdot \vec{x}')^2$
- 3rd ord. poly. $K(\vec{x}, \vec{x}') = (\vec{x} \cdot \vec{x}')^3$
- Gaussian $K(\vec{x}, \vec{x}') = e^{-\|\vec{x} - \vec{x}'\|/2\sigma^2}$

In our experiments we tested 40 combinations of kernel functions, frequency-transformations, importance-weights, and text-length-normalization. Table 3 summarizes the combinations of kernels. Its numbering and abbreviations are used throughout the rest of the paper.

In the following list we show the frequency transformations which were used.

(1)–(4) relative frequency normalized with respect to L_1

$$F_{rel,L1} = \frac{F(w_k, d_i)}{\sum_k F(w_k, d_i)}$$

(5)–(8) relative frequency with redundancy normalized with respect to L_1

$$F_{rel,red,L1} = \frac{F_{rel,red}(w_k, d_i) R(w_k)}{\sum_k F_{rel,red}(w_k, d_i) R(w_k)}$$

(9)–(12) relative frequency normalized with respect to L_2.

$$F_{rel,L_2} = \frac{F(w_k, d_i)}{\sqrt{\sum_k F(w_k, d_i)^2}}$$

(13)–(16) relative frequency with redundancy normalized with respect to L_2

$$F_{rel,red,L_2} = \frac{F(w_k, d_i) R(w_k)}{\sqrt{\sum_k \Big(F(w_k, d_i) R(w_k)\Big)^2}}$$

(17)–(20) logarithmic frequencies normalized with respect to L_1.

$$F_{log,L1} = \frac{\log(1 + F(w_k, d_i))}{\sum_k (1 + F(w_k, d_i))}$$

(21)–(24)logarithmic frequencies with redundancy normalized with respect to L_1.

$$F_{log,red,L1} = \frac{\log(1 + F(w_k, d_i)) R(w_k)}{\sum_k \Big(\log(1 + F(w_k, d_i)) R(w_k)\Big)}$$

(25)–(28) logarithmic frequencies normalized with respect to L_2.

$$F_{log,L2} = \frac{\log(1 + F(w_k, d_i))}{\sqrt{\sum_k \Big(\log\big(1 + F(w_k, d_i)\big)\Big)^2}}$$

(39)–(32) logarithmic frequencies with redundancy normalized with respect to L_2

$$F_{log,L1} = \frac{\log(1 + F(w_k, d_i)) R(w_k)}{\sqrt{\sum_k \Big((1 + F(w_k, d_i)) R(w_k)\Big)^2}}$$

(33)–(36) tfidf normalized with respect to L_1

$$F_{idf,L1} = \frac{F(w_k, d_i) \log \frac{n}{df_k}}{\sum_k F(w_k, d_i) \log \frac{n}{df_k}}$$

(37)–(40) tfidf normalized with respect to L_2

$$F_{idf,L2} = \frac{F(w_k, d_i) \log \frac{n}{df_k}}{\sqrt{\sum_k \Big(F(w_k, d_i) \log \frac{n}{df_k}\Big)^2}}$$

3 The experiments

3.1 The corpus

We use the Reuters-21578 dataset http://www.research.att.com/~lewis/reuters21578.html compiled by David Lewis and originally collected by the Carnegie group from the Reuters newswire in 1987.

We selected texts from the reuters collection along the following criteria: (i) The text length is 50 or more

running words. (ii) The text category is indicated and there is exactly one category.

In total, we selected 6242 Texts which were categorized into 64 different topics. We furthermore selected 31 most frequent topics, summing up to 6024 texts. The remaining 218 texts were categorized as "NN", regardless of their true category (which of course was different from our 31 selected categories). The names of the categories and number of texts in each category is displayed in table 4. Frequent categories such as "corn", or "wheat" do not show up in our data set because they only appear in multi-category texts, which were excluded.

To exploit the training set S_0 of 6242 texts in a better way, we used the following *cross-testing* procedure. S_0 was randomly divided into 3 subsets $S_1, \ldots, S_3$ of nearly equal size. Then five different SVM were determined using $S_0 \setminus S_i$ as training set of size ≈ 4161 and S_i was used as test set of size ≈ 2081. The numbers of correctly and wrongly classified documents were added up yielding an effective test set of all 6242 documents.

Table 4 **Names of text categories and numbers of texts for each category.**

category	data	category	data	category	data
earn	1820	gnp	72	rubber	38
acq	1807	cpi	65	veg-oil	37
crude	349	cocoa	52	tin	28
trade	330	iron-steel	45	cotton	23
money-fx	230	grain	43	bop	22
interest	164	jobs	42	gas	22
ship	151	alum	41	wpi	22
sugar	127	nat-gas	41	livestock	20
coffee	107	reserves	40	pet-chem	20
gold	97	copper	39	NN	218
mon.-supp.	91	ipi	39		

3.2 Definition of loss functions

We consider the task to decide if a text previously unknown belongs to a given category or not. The specific setup is reflected in the loss function $L(i)$, which specifies the loss that incurs for the classification of o test set where i is the true class. The loss for the misclassification of a text was set to 1. We cannot decide a-priori whether it is worse to categorize a wrong text to the topic in question than not to recognize a text which actually is in the category.

Let $c_{\text{tar}}(i)$ and $e_{\text{tar}}(i)$ respectively denote the number of correctly and incorrectly classified documents of the target category i and let $e_{\text{oth}}(i)$ and let $c_{\text{oth}}(i)$ be the same figures for the other categories in the alternative class. p_{tar} is the prior probability that an unknown text belongs to a given class. Hence the loss for a the classification of a test set is

$$L(i) = p_{tar} \frac{e_{\text{tar}}(i)}{e_{\text{tar}}(i) + c_{\text{tar}}(i)} \tag{5}$$

$$+(1 - p_{tar}) \frac{e_{\text{oth}}(i)}{e_{\text{oth}}(i) + c_{\text{oth}}(i)} \tag{6}$$

We assume that new texts to be classified arrive at the same ratio as in our corpus, i.e. p_{tar} is computed as the fraction of texts in our corpus which belong to a given class. The mean number of texts in the 31 classes is ≈ 194, and there are 6242 texts, i.e. the mean prior probability would be $p_{tar} = 194/6242 \approx 0.031$. But the fractions of texts in the 31 classes range from $p_{tar} = 0.302$ for the largest class to $p_{tar} = 0.0033$ for the smallest class. Therefore we compute the losses for two different groups of classes. The first group includes the four large classes "earn", "acq", "crude", and "trade" with more than 5% of all texts. The mean prior is $p_{tar} = 0.17$. The second group includes all other classes with mean $p_{tar} = 0.01$. To allow comparisons we always report the loss for 1000 new classifications.

In addition we used the precision $p_{\text{prc}}(i)$ and the recall $p_{\text{rec}}(i)$ to describe the result of an experiment.

$$p_{\text{prc}}(i) = \frac{c_{\text{tar}}(i)}{c_{\text{tar}}(i) + e_{\text{oth}}(i)}, \tag{7}$$

$$p_{\text{prc}}(i) = 0 \text{ if } c_{\text{tar}}(i) = 0 \tag{8}$$

$$p_{\text{rec}}(i) = \frac{c_{\text{tar}}(i)}{c_{\text{tar}}(i) + e_{\text{tar}}(i)} \tag{9}$$

Precision is the probability that a document predicted to be in a class truly belongs to this class. Recall is the probability that a document having a certain topic is classified into this class. A tradeoff exists between large recall and large precision.

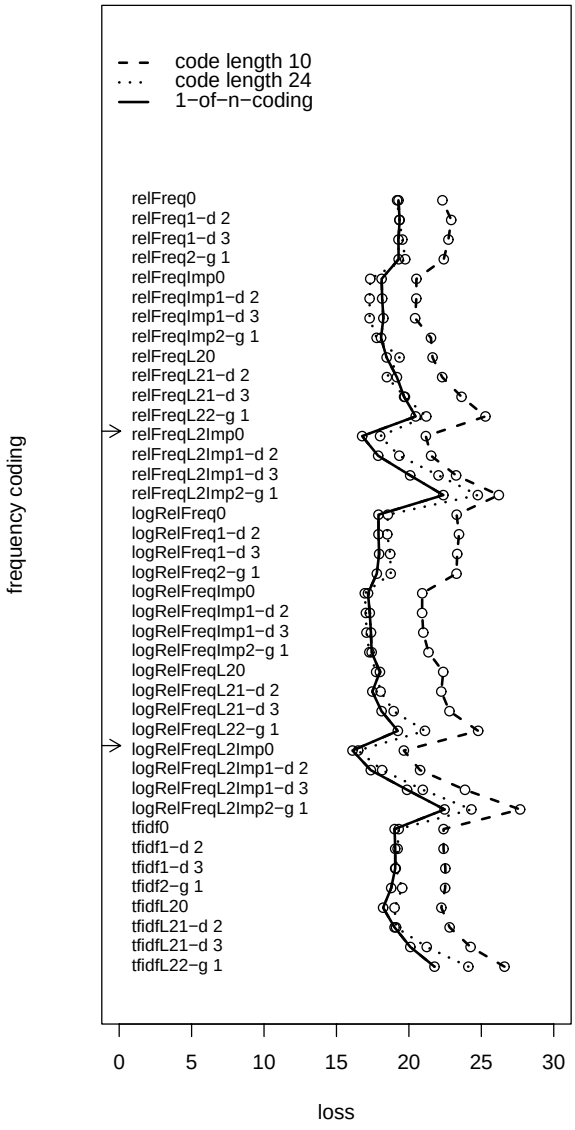
4 Results

4.1 Performance of frequency codings and kernels

As already stated in ([KL00]), which examined the performance of SVM on Reuters and two German newspaper corpora, there are only small differences in performance for different SVM kernels. Figures 2 and 3 show the mean performances for 10-bit and 24-bit multi-class codes as compared to the 1-of-n-coding. Each of the 40 frequency-kernel combinations

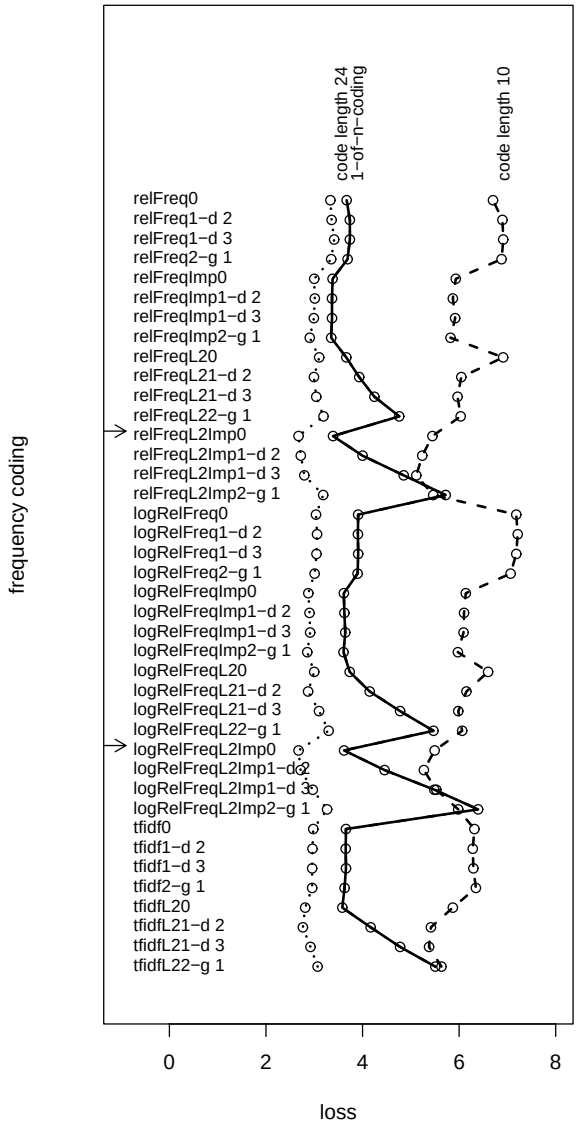
listed in section 2.5 is displayed. Figure 2 shows the mean performance of all classes in group 1 (see section 3.2). Figure 2 shows the mean performance of group 2 classes.

Figure 2 **Mean loss of classes of group 1 for all frequency codings.**



Importance weights improve the performance significantly. Redundancy defined in equation (3) is a better importance weight than the common inverse document frequency. We see that the transformations (logarithmic) relative frequency with redundancy normalized with respect to L_2 combined with the linear kernel (i.e. combinations number 12 and 28) perform best. In figures 2 and 3 these combinations are marked by arrows.

Figure 3 **Mean loss of classes of group 2.**



4.2 Small classes

Table 5 shows the mean performance over all frequency codes and SVM kernels in terms of the loss function defined in equation 5. This time we show results for the 10-bit, 16-bit, and 24-bit multi-class codings. The classes of group 1 with $p_{tar} = 0.17$ are separated from those of group 2 with $p_{tar} = 0.01$. The percentages of classes with better loss for the multi-

class codes than for the 1-of-n code is indicated below both groups in the table.

For group 1 (i.e. large) classes, 1-of-n coding and multi-class codings perform equally well. But for small classes, with less than 5% of texts, the picture changes. Here, the loss of 16-bit and 24-bit multi-class codes is in many cases smaller than that of 1-of-n coding.

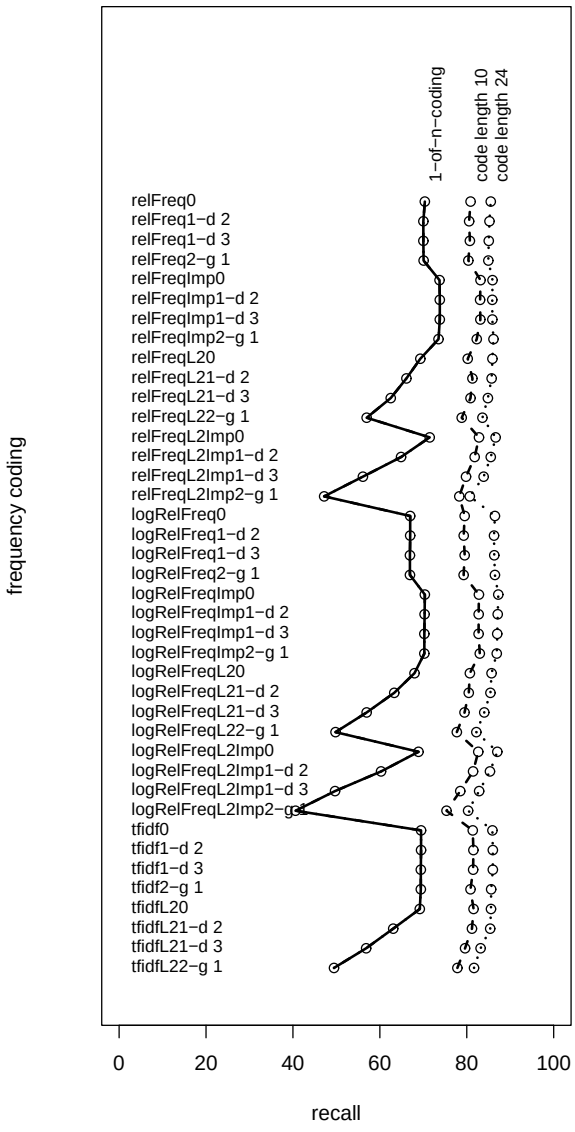
Table 5 **mean loss over all frequency codings for each code length and each category**

category	num. texts	loss of class code length . . . 10	16	24	1-of-n
earn	1820	22.18	23.91	20.769	16.32
acq	1807	35.21	39.37	35.182	23.30
crude	349	20.60	17.33	15.296	28.94
trade	330	23.60	17.80	16.536	34.19
% better than 1-of-n		**50.0**	**50.0**	**50.0**	–
money-fx	230	12.73	9.18	11.978	7.26
interest	164	13.43	6.05	6.667	5.33
ship	151	10.11	6.07	3.439	4.26
sugar	127	4.56	2.18	1.242	1.33
coffee	107	3.10	1.41	1.056	1.08
gold	97	5.41	1.52	1.849	2.48
money-supply	91	4.49	3.22	2.644	2.66
gnp	72	6.28	6.04	3.826	4.48
cpi	65	3.68	3.87	2.353	3.09
cocoa	52	1.76	3.48	0.923	2.32
iron-steel	45	6.00	3.74	3.652	5.94
grain	43	9.69	6.71	4.455	6.89
jobs	42	3.72	2.41	1.479	2.89
alum	41	5.16	4.16	3.085	6.86
nat-gas	41	6.35	3.90	3.597	5.90
reserves	40	5.35	3.39	2.577	4.15
copper	39	3.89	3.92	2.376	4.19
ipi	39	5.42	4.08	3.436	5.37
rubber	38	5.89	2.60	1.425	4.01
veg-oil	37	8.50	4.58	3.973	5.15
tin	28	2.22	1.70	0.933	5.05
cotton	23	4.26	3.24	2.833	4.82
bop	22	10.68	6.75	6.170	7.27
gas	22	13.93	4.40	5.008	5.64
wpi	22	10.18	3.78	3.811	4.92
livestock	20	7.12	7.04	5.298	6.58
pet-chem	20	11.20	9.25	9.149	9.68
% better than 1-of-n		**18.5**	**63.0**	**92.6**	–

An explanation for the better results of multi-class codes for small classes can be seen the over-fitting tendency of the learning algorithm. Classifiers like SVM tend to ignore a few cases of class “1”, when everything else is class “0”. Figures 4 and 5 support the claim: 1-of-n coding is especially strong at precision (eqn 7) on the cost of relatively weak recall (eqn 9).

But if we combine several small classes so that the union has to be recognized, the overall fraction of the training set becomes larger, thus avoiding the adverse effects of very small classes. Another possibility to avoid over-fitting on small classes would be to change the cost function of the SVM training algorithm. We are also planning to investigate this option.

Figure 4 **Mean recall of classes of group 2.**



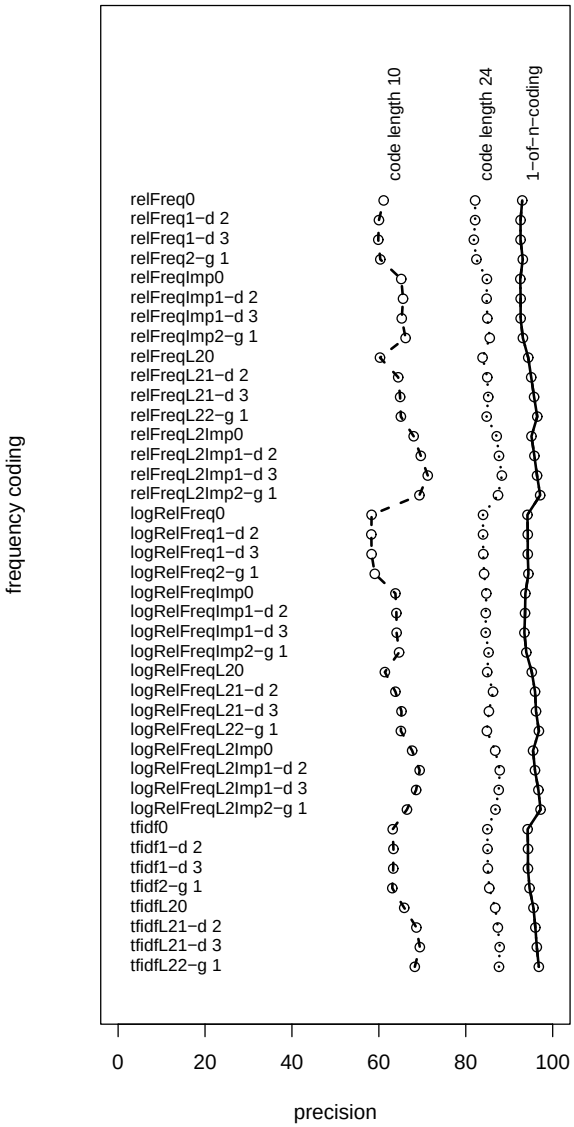
5 Conclusion

We performed experiments for text categorization on a subset of the Reuters corpus. We investigated the influences of multi-class error correcting codes on the performance over a wide variety of frequency code and SVM kernel combinations. The main result is that multi-class codes have advantages over 1-of-n coding for classes which comprise only a small percentage of the data. Advantages were seen most clearly in the range of codes with $50\% \ldots 75\%$ size of the number of classes to distinguish.

In our further work we will investigate the influence of different multi-class codes of the same length on error rates and test different optimization schemata for code generation. One option here is to introduce optimization weights into the code generation algorithm, which depend on statistical class properties. This could yield codes which help to classify small as well as large classes with the same accuracy.

We also will investigate a categorization task with a much larger number of classes (≥ 200), because a larger number of classes implies that most of them will be small.

Figure 5 **Mean precision of classes of group 2.**



References

[BGV92] B.E.Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D.Haussler, editor, *Proc. 5th ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[CV95] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[DB95] T.G. Dietterich and G. Bakiri. Solving multiclass learning via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[DPHS98] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *7th International Conference on Information and Knowledge Managment*, 1998.

[DWV99] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[GEPM00] Y. Guermeur, A. Eliseeff, and H. Paugam-Moisy. A new multi-class svm based on a uniform convergence result. In S.-I. Amari, C.L. Giles, M. Gori, and V. Piuri, editors, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN 2000*, pages IV–183 – IV–188, Los Alamitos, 2000. IEEE Computer Society.

[Joa98] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nedellec and C. Rouveirol, editors, *European Conference on Machine Learning (ECML)*, 1998.

[KL00] J. Kindermann and E. Leopold. Classification of texts with support vector machines. An examination of the efficiency of kernels and data-transformations. Paper accepted at 24th Annual Conference of the Gesellschaft für Klassifikation; 15 - 17 March, 2000 in Passau., 2000.

[NPP00] C. Nakajima, M. Pontil, and T. Poggio. People recognition and pose estimation in image sequences. In S.-I. Amari, C.L. Giles, M. Gori, and V. Piuri, editors, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural*

Networks IJCNN 2000, pages IV–189 – IV–196, Los Alamitos, 2000. IEEE Computer Society.

[Vap98] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

Detecting Concept Drift with Support Vector Machines

Ralf Klinkenberg klinkenberg@ls8.cs.uni-dortmund.de
Thorsten Joachims joachims@ls8.cs.uni-dortmund.de
Informatik VIII, Universität Dortmund, Baroper Str. 301, 44221 Dortmund, Germany
http://www-ai.cs.uni-dortmund.de/

Abstract

For many learning tasks where data is collected over an extended period of time, its underlying distribution is likely to change. A typical example is information filtering, i.e. the adaptive classification of documents with respect to a particular user interest. Both the interest of the user and the document content change over time. A filtering system should be able to adapt to such concept changes. This paper proposes a new method to recognize and handle concept changes with support vector machines. The method maintains a window on the training data. The key idea is to automatically adjust the window size so that the estimated generalization error is minimized. The new approach is both theoretically well-founded as well as effective and efficient in practice. Since it does not require complicated parameterization, it is simpler to use and more robust than comparable heuristics. Experiments with simulated concept drift scenarios based on real-world text data compare the new method with other window management approaches. We show that it can effectively select an appropriate window size in a robust way.

1. Introduction

Machine learning methods are often applied to problems, where data is collected over an extended period of time. In many real-world applications this introduces the problem that the distribution underlying the data is likely to change over time. For example, companies collect an increasing amount of data like sales figures and customer data to find patterns in the customer behaviour and to predict future sales. As the customer behaviour tends to change over time, the model underlying successful predictions should be adapted accordingly.

The same problem occurs in information filtering, i.e. the adaptive classification of documents with respect to a particular user interest. Information filtering techniques are used, for example, to build personalized news filters, which learn about the news-reading preferences of a user or to filter e-mail. Both the interest of the user and the document content change over time. A filtering system should be able to adapt to such concept changes.

This paper proposes a new method for detecting and handling concept changes with support vector machines. The approach has a clear theoretical motivation and does not require complicated parameter tuning. After reviewing other work on adaptation to changing concepts and shortly describing support vector machines, this paper explains the new window adjustment approach and evaluates it in three simulated concept drift scenarios on real-world text data. The experiments show that the approach effectively selects an appropriate window size and results in a low predictive error rate.

2. Concept Drift

Throughout this paper, we study the problem of concept drift for the pattern recognition problem in the following framework. Each example $\vec{z} = (\vec{x}, y)$ consists of a feature vector $\vec{x} \in \mathbf{R}^N$ and a label $y \in \{-1, +1\}$ indicating its classification. Data arrives over time in batches. Without loss of generality these batches are assumed to be of equal size, each containing m examples.

$$\vec{z}_{(1,1)}, \ldots, \vec{z}_{(1,m)}, \vec{z}_{(2,1)}, \ldots, \vec{z}_{(2,m)}, \cdots, \vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)}, \vec{z}_{(t+1,1)}, \ldots, \vec{z}_{(t+1,m)}$$

$\vec{z}_{(i,j)}$ denotes the j-th example of batch i. For each batch i the data is independently identically distributed with respect to a distribution $\Pr_i(\vec{x}, y)$. Depending on the amount and type of concept drift, the example distribution $\Pr_i(\vec{x}, y)$ and $\Pr_{i+1}(\vec{x}, y)$ between batches will differ. The goal of the learner $\mathcal{L}$ is to sequentially predict the labels of the next batch. For example, after

batch t the learner can use any subset of the training examples from batches 1 to t to predict the labels of batch $t+1$. The learner aims to minimize the cumulated number of prediction errors.

In machine learning, changing concepts are often handled by time windows of fixed or adaptive size on the training data (Mitchell et al., 1994; Widmer & Kubat, 1996; Lanquillon, 1997; Klinkenberg & Renz, 1998) or by weighting data or parts of the hypothesis according to their age and/or utility for the classification task (Kunisch, 1996; Taylor et al., 1997). The latter approach of weighting examples has already been used for information filtering in the incremental relevance feedback approaches of Allan (1996) and Balabanovic (1997). In this paper, the earlier approach maintaining a window of adaptive size is explored. More detailed descriptions of the methods described above and further approaches can be found in Klinkenberg (1998).

For windows of fixed size, the choice of a *"good" window size* is a compromise between fast adaptivity (small window) and good generalization in phases without concept change (large window). The basic idea of *adaptive window management* is to adjust the window size to the current extent of concept drift.

The task of learning drifting or time-varying concepts has also been studied in computational learning theory. Learning a changing concept is infeasible, if no restrictions are imposed on the type of admissible concept changes,[1] but drifting concepts are provably efficiently learnable (at least for certain concept classes), if the rate or the extent of drift is limited in particular ways.

Helmbold, & Long (1994) assume a possibly permanent but slow concept drift and define the *extent of drift* as the probability that two subsequent concepts disagree on a randomly drawn example. Their results include an upper bound for the extend of drift maximally tolerable by any learner and algorithms that can learn concepts that do not drift more than a certain constant extent of drift. Furthermore they show that it is sufficient for a learner to see a fixed number of the most recent examples. Hence a window of a certain minimal fixed size allows to learn concepts for which the extent of drift is appropriately limited.

[1] E.g. a function randomly jumping between the values one and zero cannot be predicted by any learner with more than 50% accuracy.

While Helmbold and Long restrict the extend of drift, Kuh et al. (1991) determine a maximal *rate of drift* that is acceptable by any learner, i. e. a maximally acceptable frequency of concept changes, which implies a lower bound for the size of a fixed window for a time-varying concept to be learnable, which is similar to the lower bound of Helmbold and Long.

In practice, however, it usually cannot be guaranteed that the application at hand obeys these restrictions, e.g. a reader of electronic news may change his interests (almost) arbitrarily often and radically. Furthermore the large time window sizes, for which the theoretical results hold, would be impractical. Hence more application oriented approaches rely on far smaller windows of fixed size or on window adjustment heuristics that allow far smaller window sizes and usually perform better than fixed and/or larger windows (Widmer & Kubat, 1996; Lanquillon, 1997; Klinkenberg & Renz, 1998). While these heuristics are intuitive and work well in their particular application domain, they usually require tuning their parameters, are often not transferable to other domains, and lack a proper theoretical foundation.

Syed et al. (1999) describe an approach to incrementally learning support vector machines that handles *virtual* concept drift implied by incrementally learning from several subsamples of a large training set, but they do not address the problem of (*real*) concept drift addressed here.

3. Support Vector Machines

The window adjustment approach described in this paper uses support vector machines (Vapnik, 1998) as their core learning algorithm. Support vector machines are based on the *structural risk minimization* principle (Vapnik, 1998) from statistical learning theory. In their basic form, SVMs learn linear decision rules

$$h(\vec{x}) = sign\{\vec{w} \cdot \vec{x} + b\} = \begin{cases} +1, & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ -1, & \text{else} \end{cases} \quad (1)$$

described by a weight vector $\vec{w}$ and a threshold b. The idea of structural risk minimization is to find a hypothesis h for which one can guarantee the lowest probability of error. For SVMs, Vapnik (1998) shows that this goal can be translated into finding the hyperplane with maximum soft-margin.[2] Computing this hyperplane is equivalent to solving the following optimization problem.

[2] See Burges (1998) for an introduction to SVMs.

Optimization Problem 1 (SVM (primal))

$$\textit{minimize:} \quad V(\vec{w},b,\vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i \quad (2)$$

$$\textit{subject to:} \quad \forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1-\xi_i \quad (3)$$

$$\forall_{i=1}^{n} : \xi_i > 0 \quad (4)$$

In this optimization problem, the Euclidean length $||\vec{w}||$ of the weight vector is inversely proportional to the soft-margin of the decision rule. The constraints (3) require that all training examples are classified correctly up to some slack ξ_i. If a training example lies on the "wrong" side of the hyperplane, the corresponding ξ_i is greater or equal to 1. Therefore $\sum_{i=1}^{n}\xi_i$ is an upper bound on the number of training errors. The factor C in (2) is a parameter that allows trading-off training error vs. model complexity.

For computational reasons it is useful to solve the Wolfe dual (Fletcher, 1987) of optimization problem 1 instead of solving optimization problem 1 directly (Vapnik, 1998).

Optimization Problem 2 (SVM (dual))

$$\textit{minimize: } W(\vec{\alpha}) = -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i\vec{x}_j) \quad (5)$$

$$\textit{subject to:} \sum_{i=1}^{n} y_i\alpha_i = 0 \quad (6)$$

$$\forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \quad (7)$$

In this paper, SVM^{light} (Joachims, 1999) is used for computing the solution of this optimization problem.[3] Support vectors are those training examples $\vec{x}_i$ with $\alpha_i > 0$ at the solution. From the solution of optimization problem 2 the decision rule can be computed as

$$\vec{w}\cdot\vec{x} = \sum_{i=1}^{n}\alpha_i y_i (\vec{x}_i\cdot\vec{x}) \quad \text{and} \quad b = y_{usv} - \vec{w}\cdot\vec{x}_{usv} \quad (8)$$

The training example $(\vec{x}_{usv}, y_{usv})$ for calculating b must be a support vector with $\alpha_{usv} < C$. Finally, the training losses ξ_i can be computed as $\xi_i = max(1 - y_i\left[\vec{w}\cdot\vec{x}_i + b\right], 0)$.

For both solving optimization problem 2 as well as applying the learned decision rule, it is sufficient to be able to calculate inner products between feature vectors. Exploiting this property, Boser et al. introduced the use of kernels $K(\vec{x}_1, \vec{x}_2)$ for learning non-linear decision rules. Depending on the type of kernel function, SVMs learn polynomial classifiers, radial basis function (RBF) classifiers, or two layer sigmoid neural nets. Such kernels calculate an inner-product in some feature space and replace the inner-product in the formulas above.

[3] SVM^{Light} is available at http://www-ai.informatik.uni-dortmund.de/svm_light

4. Window Adjustment by Optimizing Performance

Our approach to handling drift in the distribution of examples uses a window on the training data. This window should include only those example which are sufficiently "close" to the current target concept. Assuming the amount of drift increases with time, the window includes the last n training examples. Previous approaches used similar windowing strategies. Their shortcomings are that they either fix the window size (Mitchell et al., 1994) or involve complicated heuristics (Widmer & Kubat, 1996; Lanquillon, 1997; Klinkenberg & Renz, 1998). A fixed window size makes strong assumptions about how quickly the concept changes. While heuristics can adapt to different speed and amount of drift, they involve many parameters that are difficult to tune. Here, we present an approach to selecting an appropriate window size that does not involve complicated parameterization. They key idea is to select the window size so that the estimated generalization error on new examples is minimized. To get an estimate of the generalization error we use a special form of $\xi\alpha$-estimates (Joachims, 2000). $\xi\alpha$-estimates are a particularly efficient method for estimating the performance of a SVM.

4.1 $\xi\alpha$-Estimators

$\xi\alpha$-estimators are based on the idea of leave-one-out estimation (Lunts & Brailovskiy, 1967). The leave-one-out estimator of the error rate proceeds as follows. From the training sample $S = ((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n))$ the first example $(\vec{x}_1, y_1)$ is removed. The resulting sample $S^{\setminus 1} = ((\vec{x}_2, y_2), \cdots, (\vec{x}_n, y_n))$ is used for training, leading to a classification rule $h_{\mathcal{L}}^{\setminus 1}$. This classification rule is tested on the held out example $(\vec{x}_1, y_1)$. If the example is classified incorrectly it is said to produce a leave-one-out error. This process is repeated for all training examples. The number of leave-one-out errors divided by n is the leave-one-out estimate of the generalization error.

While the leave-one-out estimate is usually very accurate, it is very expensive to compute. With a training sample of size n, one must run the learner n times. $\xi\alpha$-estimators overcome this problem using an upper bound on the number of leave-one-out errors instead of calculating them brute force. They owe their name

to the two arguments they are computed from. $\vec{\xi}$ is the vector of training losses at the solution of the primal SVM training problem. $\vec{\alpha}$ is the solution of the dual SVM training problem. Based on these two vectors — both are available after training the SVM at no extra cost — the $\xi\alpha$-estimators are defined using the following two counts. With R^2_Δ being the maximum difference of any two elements of the Hessian (i.e. $R^2_\Delta \geq \max_{\vec{x},\vec{x}'}(\mathcal{K}(\vec{x},\vec{x}) - \mathcal{K}(\vec{x},\vec{x}'))$),

$$d = |\{i : (\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \tag{9}$$

counts the number of training examples, for which the quantity $\alpha_i R^2_\Delta + \xi_i$ exceeds one. Since the document vectors are normalized to unit length in the experiments described in this paper, here $R^2_\Delta = 1$. It is proven in Joachims (2000) that d is an approximate upper bound on the number of leave-one-out errors in the training set. With n as the total number of training examples, the $\xi\alpha$-estimators of the error rate is

$$Err^n_{\xi\alpha}(h_\mathcal{L}) = \frac{|\{i : (\alpha_i R^2_\Delta + \xi_i) \geq 1\}|}{n} \tag{10}$$

The theoretical properties of this $\xi\alpha$-estimator are discussed in Joachims (2000). It can be shown that the estimator is pessimistically biased, overestimating the true error rate on average. Experiments show that the bias is acceptably small for text classification problems and that the variance of the $\xi\alpha$-estimator is essentially as low as that of a holdout estimate using twice as much data. It is also possible to design similar estimators for precision and recall, as well as combined measures like $F1$ (Joachims, 2000).

4.2 Window Adjustment Algorithm

A window adjustment algorithm has to solve the following trade-off. A large window provides the learner with much training data, allowing it to generalize well given that the concept did not change. On the other hand, a large window can contain old data that is no longer relevant (or even confusing) for the current target concept. Finding the right size means trading-off the quality against the number of training examples.

To answer this question the window adjustment algorithm proposed in the following uses $\xi\alpha$-estimates in a particular way. At batch t, it essentially tries various window sizes, training a SVM for each resulting training set.

$$\vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)} \tag{11}$$

$$\vec{z}_{(t-1,1)}, \ldots, \vec{z}_{(t-1,m)}, \vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)} \tag{12}$$

$$\vec{z}_{(t-2,1)}, \ldots, \vec{z}_{(t-2,m)}, \vec{z}_{(t-1,1)}, \ldots, \vec{z}_{(t-1,m)}, \vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)} \tag{13}$$

$$\vdots$$

For each window size it computes a $\xi\alpha$-estimate based on the result of training. In contrast to the previous section, the $\xi\alpha$-estimator used here considers only the last batch, that is the m most recent training examples $\vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)}$.

$$Err^m_{\xi\alpha}(h_\mathcal{L}) = \frac{|\{i : 1 \leq i \leq m \wedge (\alpha_{(t,i)} R^2_\Delta + \xi_{(t,i)}) \geq 1\}|}{m} \tag{14}$$

This reflects the assumption that the most recent examples are most similar to the new examples in batch $t+1$. The window size minimizing the $\xi\alpha$-estimate of the error rate is selected by the algorithm.

The algorithm can be summarized as follows:

- `input:` S `training sample consisting of` t `batches containing` m `examples each`
- `for` $h \in \{0, \ldots, t-1\}$
  - `train SVM on examples` $\vec{z}_{(t-h,1)}, \ldots, \vec{z}_{(t,m)}$
  - `compute` $\xi\alpha$`-estimate on examples` $\vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)}$
- `output: window size which minimizes` $\xi\alpha$`-estimate`

5. Experiments

5.1 Experimental Setup

Each of the following *data management approaches* is evaluated in combination with the SVM:

- *"Full Memory"*: The learner generates its classification model from all previously seen examples, i.e. it cannot "forget" old examples.
- *"No Memory"*: The learner always induces its hypothesis only from the most recent batch. This corresponds to using a window of the fixed size of one batch.
- Window of *"Fixed Size"*: A window of the fixed size of three batches is used.
- Window of *"Adaptive Size"*: The window adjustment algorithm proposed in the previous section adapts the window size to the current concept drift situation.

The experiments are performed in an information filtering domain, a typical application area for learning drifting concept. Text documents are represented as attribute-value vectors (*bag of words* model), where each distinct word corresponds to a feature whose

Table 1. Relevance of the categories in the concept change scenarios A, B, and C.

Scenario	Category	Probability of being relevant for a document of the specified category at the specified time step (batch)																			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.6	0.4	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.4	0.6	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

value is the "ltc"-TF/IDF-weight (Salton & Buckley, 1988) of that word in that document. Words occurring less than three times in the training data or occurring in a given list of stop words are not considered. Each document feature vector is normalized to unit length to abstract from different document lengths.

The performance of a classifier is measured by the three metrics prediction error, recall, and precision. *Recall* is the probability, that the classifier recognizes a relevant document as relevant. *Precision* is the probability, that a document classified as relevant actually is relevant. All reported results are estimates averaged over ten runs.

The experiments use a subset of 2608 documents of the data set of the *Text REtrieval Conference (TREC)* consisting of English business news texts. Each text is assigned to one or several categories. The categories considered here are 1 (Antitrust Cases Pending), 3 (Joint Ventures), 4 (Debt Rescheduling), 5 (Dumping Charges), and 6 (Third World Debt Relief). For the experiments, three concept change scenarios are simulated. The texts are randomly split into 20 batches of equal size containing 130 documents each.[4] The texts of each category are distributed as equally as possible over the 20 batches.

Table 1 describes the relevance of the categories in the three concept change scenarios A, B, and C. For each time step (batch), the probability of being relevant (interesting to the user) is specified for documents of categories 1 and 3, respectively. Documents of the classes 4, 5, and 6 are never relevant in any of these scenarios. In the first scenario (*scenario A*), first documents of category 1 are considered relevant for the user interest and all other documents irrelevant. This changes abruptly (concept shift) in batch 10, where documents of category 3 are relevant and all others irrelevant. In the second scenario (*scenario B*), again first documents of category 1 are considered relevant for the user interest and all other documents irrelevant. This changes slowly (concept drift) from batch 8 to batch 12, where documents of category 3 are relevant and all others irrelevant. The third scenario (*scenario C*) simulates an abrupt concept shift in the user interest from category 1 to category 3 in batch 9 and back to category 1 in batch 11.

[4]Hence, in each trial, out of the 2608 documents, eight randomly selected texts are not considered.

5.2 Results

Figure 1 compares the prediction error rates of the adaptive window size algorithm with the non-adaptive methods. The graphs show the prediction error on the following batch. In all three scenarios, the full memory strategy and the adaptive window size algorithm essentially coincide as long as there is no concept drift. During this stable phase, both show lower prediction error than the fixed size and the no memory approach. At the point of concept drift, the performance of all methods deteriorates. While the performance of no memory and adaptive size recovers quickly after the concept drift, the error rate full memory approach remains high especially in scenarios A and B. Like before the concept drift, the no memory and the fixed size strategies exhibit higher error rates than the adaptive window algorithm in the stable phase after the concept drift. This shows that the no memory, the fixed size, and the full memory approaches all perform suboptimally in some situation. Only the adaptive window size algorithm can achieve a relatively low error rate over all phases in all scenarios. This is also reflected in the average error rates over all batches given in Table 2. The adaptive window size algorithm achieves a low average error rate on all three scenarios. Similarly, precision and recall are consistently high.

The behavior of the adaptive window algorithm is best explained by looking at the window sizes it selects. Figure 2 shows the average training window ranges. The bottom of each graph depicts the time and extent of concept drift in the corresponding scenario. For

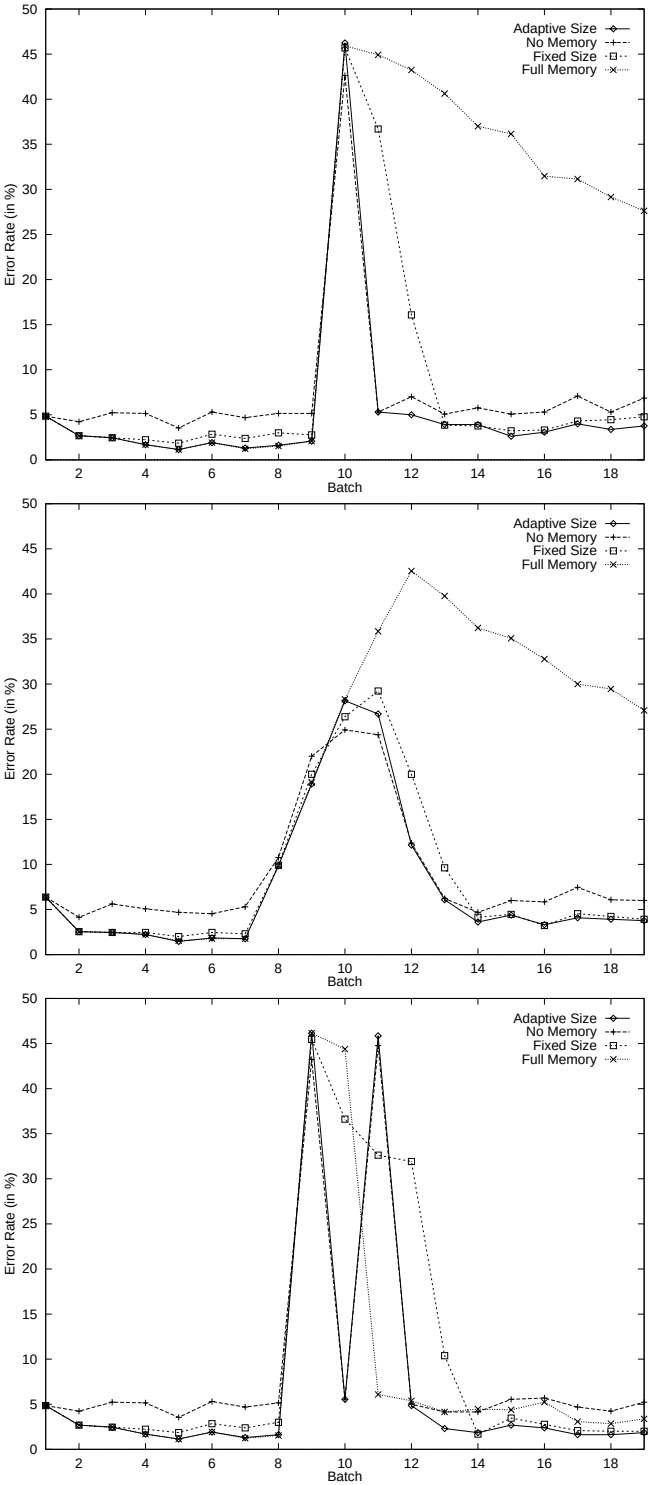


Figure 1. Comparison of the prediction error rates for scenario A (top), B (middle), and C (bottom). The x-axis denotes the batch number and the y-axis the average prediction error.

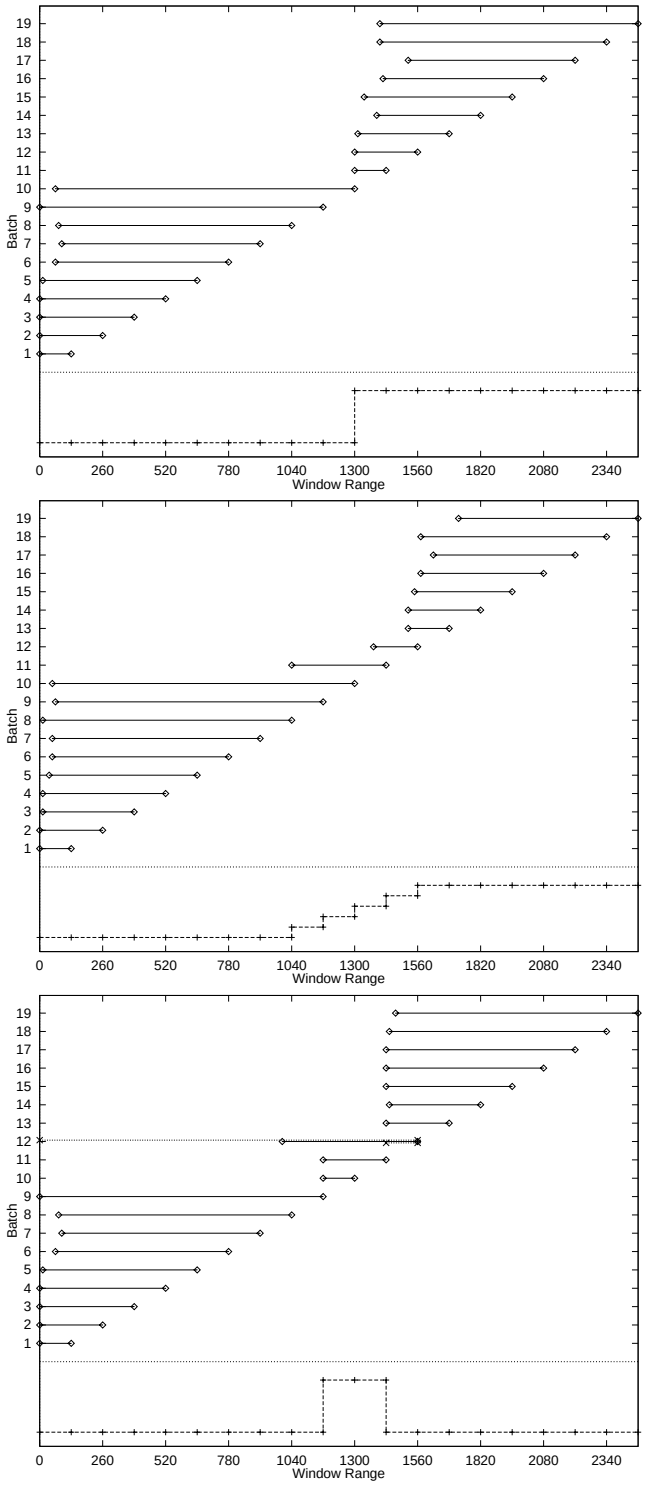


Figure 2. Window size and range for scenario A (top), B (middle), and C (bottom). The y-axis denotes the batch number. Each horizontal line marks the average training window range selected at that batch number. The bottom part of each graph depicts the location and type of the concept shift.

Table 2. Error, accuracy, recall, and precision of all window management approaches for all scenarios averaged over 10 trials with 20 batches each (standard sample error in parentheses).

	Full Memory	No Memory	Fixed Size	Adaptive Size
Scenario A:				
Error	20.36%	7.30%	7.96%	5.32%
	(4.21%)	(1.97%)	(2.80%)	(2.29%)
Recall	51.69%	74.42%	77.64%	85.35%
	(8.37%)	(4.61%)	(6.07%)	(4.93%)
Precision	64.67%	91.29%	87.73%	91.61%
	(8.38%)	(5.10%)	(5.93%)	(5.11%)
Scenario B:				
Error	20.25%	9.08%	8.44%	7.56%
	(3.56%)	(1.57%)	(2.00%)	(1.89%)
Recall	49.35%	67.22%	73.85%	76.70%
	(7.01%)	(5.04%)	(5.51%)	(5.42%)
Precision	65.09%	88.86%	87.19%	88.48%
	(6.80%)	(3.67%)	(4.18%)	(3.89%)
Scenario C:				
Error	7.74%	8.97%	10.17%	7.07%
	(3.05%)	(2.84%)	(3.30%)	(3.16%)
Recall	76.54%	63.68%	68.18%	78.17%
	(6.26%)	(5.27%)	(7.05%)	(6.34%)
Precision	83.15%	87.67%	79.00%	87.38%
	(6.69%)	(7.06%)	(8.09%)	(6.99%)

scenario A the training window increases up to the abrupt concept change after batch 10, covering almost all examples available for the current concept. Only in batches 5 to 10 the average training set size is slightly smaller than maximally possible. Our explanation is that for large training sets a relatively small number of additional examples does not always make a "noticeable" difference. After the concept change in batch 10 the adaptive window size algorithm now picks training windows covering only those examples from after the drift as desired. A similar behavior is found for scenario B (Figure 2, middle). Since the drift is less abrupt, the adaptive window size algorithm intermediately selects training examples from both concepts in batch 11. After sufficiently many training examples from the new distribution are available, those earlier examples are discarded. The behavior of the adaptive window size algorithm in scenario C is reasonable as well (Figure 2, bottom). A particular situation occurs in batch 12. Here the window size exhibits a large variance. For 8 of the 10 runs the algorithm selects a small training set size of one batch, while for the remaining 2 runs it selects all available training examples starting with batch 1. Here there appears to be a borderline decision between accepting 2 (out of 12) batches of "bad" examples or just training on a single batch.

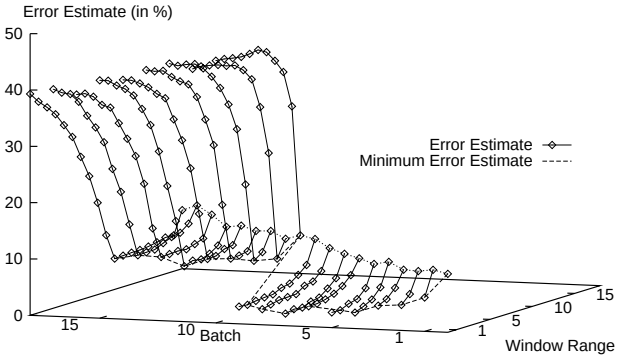


Figure 3. Average $\xi\alpha$-estimates at different batches and for varying training window sizes for scenario A. The dashed curve marks the beginning of the window with the lowest error estimate.

Further insight on how the algorithm selects the window size is gained from figure 3. It plots the average $\xi\alpha$-estimate in scenario A over all batches and for varying window size. The x_1-axis denotes the number of the current batch (increasing from right to left) and the x_2-axis the batch of the window start. The dashed line indicates the beginning of the window with the lowest estimate in the batch. The graph shows that the error estimate decreases with growing window size in batches 1 to 10. After batch 10, the estimate accurately reflects the concept change. The error estimate decreases with training windows growing towards the abrupt concept change. If the window is enlarged beyond this change, the estimated error increases steeply as expected.

6. Summary and Conclusions

In this paper, we proposed a new method for handling concept drift with support vector machines. The method directly implements the goal of discarding irrelevant data with the aim of minimizing generalization error. Exploiting the special properties of SVMs, we adapted $\xi\alpha$-estimates to the window size selection problem. Unlike for the conventional heuristic approaches, this gives the new method a clear and simple theoretical motivation. Furthermore, the new method is easier to use in practical applications, since it involves less parameters than complicated heuristics. Experiments in an information filtering domain show that the new algorithm achieves a low error rate and selects appropriate window sizes over very different concept drift scenarios.

An open questions is how sensitive the algorithm is to the size of individual batches. Since in the current version of the algorithm the batch size determines the estimation window, the variance of the window size is

likely to increase with smaller batches. It might be necessary to select the estimation window size independent of the batch size. A shortcoming of most existing algorithms handling concept drift (an exception is Lanquillon (1999)) is that they can detect concept drift only after labeled data is available. That is, after the learning algorithm starts making mistakes. While this appears unavoidable for concept drift with respect to $\Pr(y|\vec{x})$, it might be possible to detect concept drift in $\Pr(\vec{x})$ earlier by using transductive support vector machines.

Acknowledgments

This work was supported by the DFG Collaborative Research Center on Complexity Reduction in Multivariate Data (SFB 475) and by the DFG Collaborative Research Center on Computational Intelligence (SFB 531).

References

Allan, J. (1996). Incremental relevance feedback for information filtering. *Proceedings of the Nineteenth ACM Conference on Research and Development in Information Retrieval* (pp. 270–278). New York: ACM Press.

Balabanovic, M. (1997). An adaptive web page recommendation service. *Proceedings of the First International Conference on Autonomous Agents* (pp. 378–385). New York: ACM Press.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*, 121–167.

Fletcher, R. (1987). *Practical methods of optimization* (2nd edition). New York: Wiley.

Helmbold, D. P., & Long, P. M. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning, 14*, 27–45.

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods - Support vector learning*. Cambridge, MA, USA: MIT Press.

Joachims, T. (2000). Estimating the generalization performance of a SVM efficiently. *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufman.

Klinkenberg, R. (1998). *Maschinelle Lernverfahren zum adaptiven Informationsfiltern bei sich verändernden Konzepten*. Masters thesis, Fachbereich Informatik, Universität Dortmund, Germany.

Klinkenberg, R., & Renz, I. (1998). Adaptive information filtering: Learning in the presence of concept drifts. *Workshop Notes of the ICML-98 Workshop on Learning for Text Categorization* (pp. 33–40). Menlo Park, CA, USA: AAAI Press.

Kuh, A., Petsche, T., & Rivest, R. (1991). Learning time-varying concepts. *Advances in Neural Information Processing Systems* (pp. 183–189). San Mateo, CA, USA: Morgan Kaufmann.

Kunisch, G. (1996). *Anpassung und Evaluierung statistischer Lernverfahren zur Behandlung dynamischer Aspekte in Data Mining*. Masters thesis, Fachbereich Informatik, Universität Ulm, Germany.

Lanquillon, C. (1997). *Dynamic Neural Classification*. Masters thesis, Fachbereich Informatik, Universität Braunschweig, Germany.

Lanquillon, C. (1999). Information filtering in changing domains. *Working Notes of the IJCAI-99 Workshop on Machine Learning for Information Filtering* (pp. 41–48). Stockholm, Sweden.

Lunts, A., & Brailovskiy, V. (1967). Evaluation of attributes obtained in statistical decision rules. *Engineering Cybernetics, 3*, 98–109.

Mitchell, T., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience with a learning personal assistant. *Communications of the ACM, 37*, 81–91.

Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management, 24*, 513–523.

Syed, N. A., Liu, H., & Sung, K. K. (1999). Handling concept drifts in incremental learning with support vector machines. *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*. New York: ACM Press.

Taylor, C., Nakhaeizadeh, G., & Lanquillon, C. (1997). Structural change and classification. *Workshop Notes of the ECML-97 Workshop on Dynamically Changing Domains: Theory Revision and Context Dependence Issues* (pp. 67–78).

Vapnik, V. (1998). *Statistical learning theory*. Chichester, GB: Wiley.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning, 23*, 69–101.

MINING MART: Combining Case-Based-Reasoning and Multistrategy Learning into A Framework for Reusing KDD-Applications*

Jörg-Uwe Kietz† Regina Zücker† Anca Vaduva‡

† Swiss Life
IT Research & Development
CH-8022 Zurich, Switzerland
{Uwe.Kietz,Regina.Zuecker}@swisslife.ch

‡ University of Zurich
Department of Information Technology
CH-8057 Zurich, Switzerland
vaduva@ifi.unizh.ch

Abstract

One of the most time consuming steps for the process of knowledge discovery in databases (KDD) consists in preparing the source data. On the one hand, problems are raised by the large amounts of heterogeneous data of doubtful quality to be processed. On the other hand, since no universal tool is nowadays available to deal with all the various analysis tasks required by a real application, many different types of data mining algorithms and tools have to be employed. Typically, they have very strict and specialized input requirements. In this paper, we propose a case-based-reasoning (CBR) framework for the pre-processing step of the KDD-process. The aim is less to entirely automatize pre-processing and mining tool selection, but mainly to support the reusing of work done for one KDD-task to similar ones. As a side effect, the integration of pre-processing operations supported by multiple machine learning (or data minig) will make the case adaption at least partially automatic.

1 Introduction

The entire (information) society is in a somewhat paradox situation: we are starving for knowledge while drowning in data. Traditional approaches fail to release the knowledge from the masses of available data. Two technologies are emerging to reduce the gap:

1. data warehousing and on-line analytical processing (OLAP) of data for the verification of hypotheses and
2. knowledge discovery in databases (KDD) for discovering new hypotheses.

Practical experience with these techniques have proven their value. However, it is also apparent that using a data warehouse for decision support or applying tools for knowledge discovery are difficult and time-consuming tasks. Therefore, it is currently still quite difficult to get a high return of investment from using them. Definitely, their main drawback is the pre-processing of data. If we inspect real-world applications of knowledge discovery, we realize that 50 - 80% of the efforts are spent on finding an appropriate transformation of the given data, finding appropriate sampling of the data, and specifying the proper target of data mining, etc. Furthermore, pre-processing is not only time-consuming, but also requires profound data mining and database know-how. As a result, pre-processing cannot be done by the common business people, but only by a few highly skilled power users.

A further problem is the mass of data to be pre-processed. Even if existing KDD-tools offer facilities to pre-process data so far, they fail in achieving this task for real applications because they cannot deal with large amounts of data. The reason is that data has to be first loaded from databases into the KDD-environment and then internally processed by making use of extensive temporary storage which becomes costly or even impossible for large amounts of data.

To overcome the shortcomings of the currently existing support for KDD, this paper addresses the following objectives:

1. Create a user-friendly KDD support environment (KDDSE) for the non-expert user. In particular, the

*A version of this paper appeared in the Proceedings of the 5th Int. Conference on Multi-Strategy Learning, Porto, May 2000.

focus lies on extending the already existing tools for data mining with a user-friendly environment for *data pre-processing*.

2. Provide reusable components for the expert-user within the pre-processing environment. These components may be either easily configured or effortlessly extended for implementing new data mining applications.

3. Avoid the limitations imposed by todays tools where the whole amount of data that has to be loaded, kept and internally handled within the pre-processing environment.

4. Speed up the discovery process by reducing the number and the complexity of trial and error preprocessing and analysis cycles.

In order to reach these objectives, we propose an application development framework called Mining Mart integrating techniques of CBR and multistrategy learning. The framework provides a collection of *cases* which may be either directly used and executed or reused for developing new ones. A case consists of the specification of a business problem, the data to be analyzed and a chain of pre-processing operators that are based on clever multistrategy learning. This framework may be used both, by experts and non-experts: the end-user retrieves one of the prepared cases, makes some simple adaption if required (e.g., the selection of another target segment) and initiates the case execution.

In contrast, the highly skilled data mining power-user uses the framework for creating new cases. The cases already available provide useful building blocks and analysis chains for reusing, which should speed-up the creation of new business cases. One of the particularities of our approach is the realization of the framework: unlike in other KDDSE, we let the data to be stored further on in databases and implement the pre-processing framework beyond the database management system(DBMS). In this way, we can make use of the inherent mechanisms provided by DBMS in handling large amounts of data. That means, the specification of the business problem and the pre-processing operators belonging to a case are (intensionally) stored in the form of metadata [Staudt *et al.*, 1999b; Staudt *et al.*, 1999a]. In particular, transformation specifications, database structures, configuration parameters and statistical information are stored with an appropriate granularity and in accordance with a suitable metadata schema in a repository. At runtime, the specifications are read, interpreted, merged into an operation chain and finally executed. The advantages are the *reusability* of metadata and the handling of *large amounts of data*: cases may be either directly used or easily edited and re-applied and - most importantly to tackle objective 3 - they may make use of the facilities offered by DBMS to execute the data transformations at runtime. In this case, the system is called *metadata-driven* and is based on a similar principle as the modern Extract/Transform/Load-tools like PowerMart (Informatica, http://www.informatica.com) or Datastage (Ardent, http://www.ardentsoftware.com).

	Time	Imp.
Business understanding	20%	80%
a) Exploring the problem	10%	15%
b) Exploring the solution	9%	14%
c) Implementation specification	1%	51%
Data preparation & mining	80%	20%
a) Data preparation	60%	15%
b) Data surveying	15%	3%
c) Modeling (data mining)	5%	2%

Table 1: Steps of a KDD-project with time to complete and importance to success

The remainder of the paper is structured as follows: Section 2 generally discusses aspects related to business cases, KDD-processes and their relation to data warehousing. In Section 3 we present our approach for data pre-processing. Section 4 presents related work while Section 5 concludes the paper.

2 Business Cases, KDD-projects and Data Warehousing

Ideally, a KDD-project starts with a business case, which could be solved or optimized by analyzing available data. The typical steps of such a project are given in table 1 from [Pyle, 1999]. First of all, it should be noted that the specification of *how to use* the expected mining results plays a decisive role for the success of a project and should be established in accordance with the company management at the very beginning; the best mining results are worth nothing, if they are not used. Furthermore, considering that the most time-consuming step is the preparation of data for mining (see table 1), both problems could be solved in a related manner. The management support for the use of the mining results as well as the justification of the high data preparation costs are both most easily reached if the KDD-project is integrated into a strategic and repeatedly occurring business case which

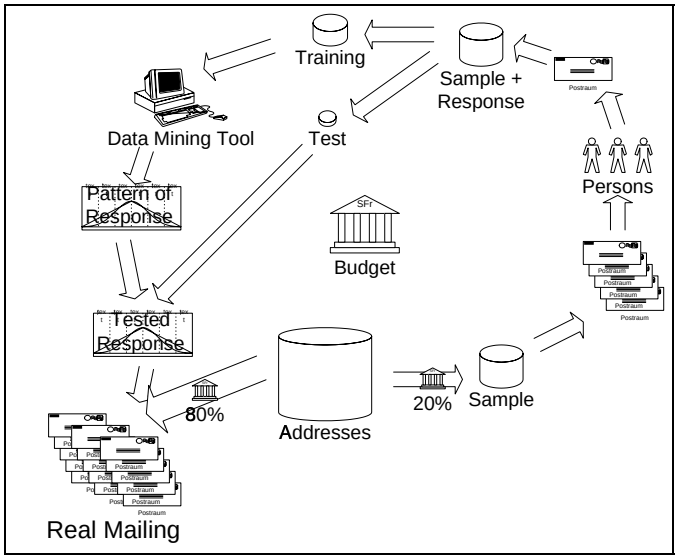


Figure 1: The Business Case Mailing Action

may then reuse the pre-processing efforts each time it is executed.

For Swiss Life, a leading life insurance company of Switzerland, there are several application areas where central business cases could be supported by data mining [Staudt *et al.*, 1998], especially:

- Marketing
- Product development and controlling
- Business reporting

In this paper we will use the optimization of responses to mailing actions in direct marketing as an illustrating and well known example of such a problem. The aim is to perform mailing actions to those addresses with the highest probable rate of return. This business case[1] is illustrated in Figure 1. On a more technical level it may be described by the following steps:

0. Use an existing data warehouse (DWH) as base for extracting the needed data.
1. Construct a household view on this DWH, which provides all relevant information in a form, that allows the next step to be done by an end-user.
2. Select the target segment, e.g. households,
   (a) which have a child with an age below 2 and which are not mailed since the birthday of this child, or
   (b) which have already bought a single-premium insurance, but this is more than two years ago, or
   (c) ...
3. Select a random-sample from the target segment for training and test, let's say with size proportional to 20% of the budget. That means, if the budget is X and a single letter cost Y a total of X/Y letters may be sent. Spending 20% of the budget to get the training/test data means selecting X/Y*0.2 household records from the target segment.
4. Export the addresses of this sample, do the first mailing, and store the responses, i.e. label the sample.
5. Split the sample into training and test set.
6. Select/construct the relevant attributes for the current response prediction task.
7. Train the selected mining-tool, which output could be used to order (not just classify) the data.
8. Apply the data transformations done in step 6 to the test data (as the mined pattern relies on these data transformations)
9. Test the mined pattern on the test set, i.e get an estimated response rate for the mined pattern.
10. Apply the pre-processing done in step 6 to the target segment; the mined pattern relies on this preprocessing.
11. Select the best records (i.e., ordered by the mined reponse pattern) from the target segment of step 2. In accordance with the assumption in step 3, the selection set should be proportional to 80% of the budget.
12. Export the addresses of this selection and do the real mailing.
13. Compute a final evaluation, and store all the mailing-information (date, (non-) responses, segment, product, mined pattern, data-transformations, evaluation, ...) in the metadata repository, such that it could be used as background knowledge for further actions.

[1] Ling and Li [1998] describe another problem and solution analysis of this business case. However their analysis is based on the assumption that existing customers have been acquired by mailings, (i.e., they all are persons answering to mailings), which is not the case in our setting, where most contracts are still sold by insurance-agents and not by mailing-actions.

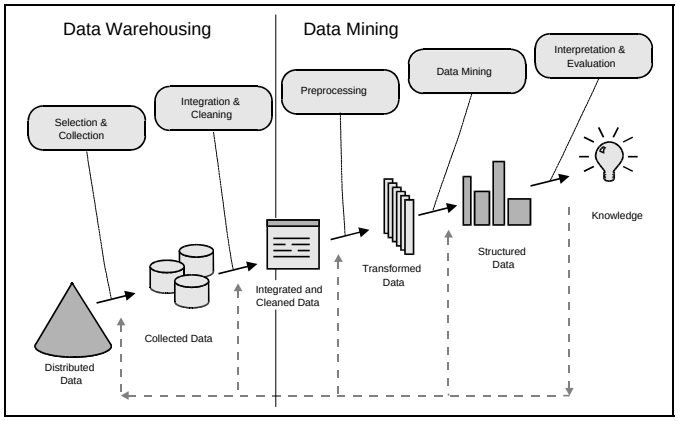


Figure 2: The KDD Process

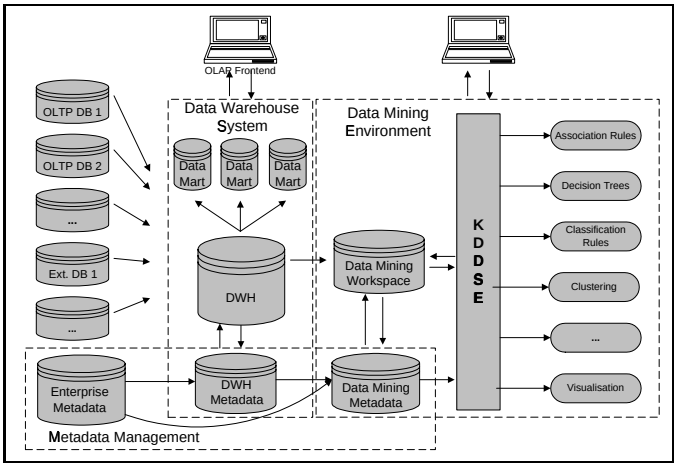


Figure 3: Architecture of a Data Warehousing and KDD Environment

Considering the standard KDD-process [Brachman and Anand, 1996] in Figure 2 consisting of five phases, step 0 corresponds to the end of second phase. The building of a data warehouse (DWH) covers the first two phases (selection & collection and integration & cleaning). Our approach relies on an already built DWH, which is, in fact, an ideal first step in setting up a KDD-process [Inmon, 1996]. The interplay between data warehousing and KDD is depicted in Figure 3. The data warehouse is build by extracting, transforming, integrating, cleaning, and finally loading the data from the operative systems. In the figure, these data sources are labeled as OLTP (on-line transaction processing) systems. While DWH is still a relational, normalized and a mostly general-purpose database, a data mart[2] contains application-specific and aggregated data for an OLAP(on-line analytical processing) application. Thus, data marts are not very useful for data mining, since they usually do not contain the data at the necessary level of detail needed for mining. For example, the typical data mart for the monthly product sales (number, value, ...) stored by region, product and time, could not be used to analyze customer behaviour with mining methods, as it does not contain any reference to specific customers, even if it is an aggregation of customers buying behaviour[3]. As a consequence, a data mining workspace has to be built on top of the data warehouse, independently of the other data marts. Note in Figure 3 that metadata of both data warehouse system and data mining environment have to be integrated into a (logically) central enterprise repository in order to manage the whole meta-information in a unified and consistent way. The repository stores the connections between the data warehouse and the data mining environment through corresponding links between metadata elements.

As far as relevant for this business case, the basic structure of our data warehouse is shown in Figure 4[4].

When we start the pre-processing phase on top of such a DWH, we are faced with

- a normalized, multi-relational database as data source, whereas most existing data mining algorithms are single-table based,
- many features and tables which are only partially relevant for the current business case,
- a data representation optimized for maintenance and not for optimal use within a specific data mining tool for a specific task (e.g., the DWH stores the birthdate but for data mining the age is required).

For our mailing-case, this means that the mapping between DWH data representation and the specification of a customer segment (step 2) is far from being trivial. So the first step (1) has to generate a more application-oriented view on top of the DWH schema. Inherently, it strongly depends on the business case to solve. Since we are interested in households in our business case, the corresponding view will reflect this. In contrast, the base-level for product development would not be households, but insurance contracts.

[2]Note that no agreement exists regarding the distinction between DWH and data mart in the DWH-literature.

[3]However, the hierarchical dimensions, e.g. for product, time and region, of the data mart could be very useful background knowledge for mining.

[4]Further information on this data warehouse, in particular a data extract, are available on our Web for further experiments. It can be obtained from http://research.swisslife.ch/kdd-sisyphus/.

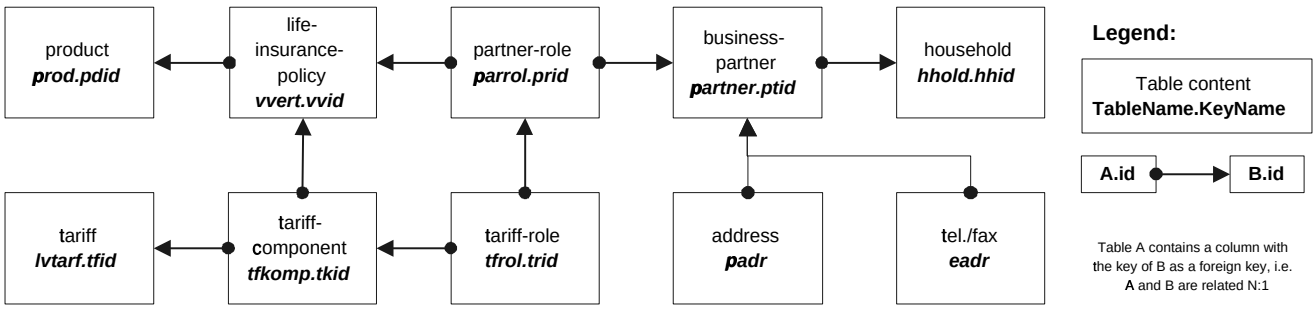


Figure 4: Schema excerpt from our DWH

The construction of the view already specifies the first group of pre-processing steps to be handled (i.e., stored, retrieved and adapted) by our case-based-reasoning system. The other group is the one described in step 6. For the same business case, step 1 does not need much adaption for reuse since the same schema may be applied to another target segment (i.e. to 2.b instead of 2.a). However, this is not true for step 6 as different target segments may have very different properties, resulting in different attributes relevant for predicting behaviour: as a general rule it could be stated, e.g., that most households addressed by 2.a are not ensured so far, whereas for segment 2.b., the past insurance behaviour is very likely to be important for predicting future behaviour.

3 A Case-Based-Reasoning System for Pre-Processing

This section discusses our approach [5] for supporting preprocessing and implicitly data mining. As already mentioned, our aim is to build a system that supports the design, storage and management of business cases that may be directly used by end-users when solving a specific mining task. In addition, support for power-users is needed to describe and store new mining tasks.

Figure 5 illustrates the use of the CBR-system. An end-user comes up with a specific business task he wants to solve with data mining (e.g., doing the mailing action as described in the last section) and asks the system, if there exists a case for this task. If one of the cases fits user's intention, he can apply it, otherwise he has to contact a power-user to set up a new case.

[5] The work is embedded in the Mining Mart project. Information may be found at http://www-ai.cs.uni-dortmund.de/FORSCHUNG/PROJEKTE/MININGMART/index.eng.html

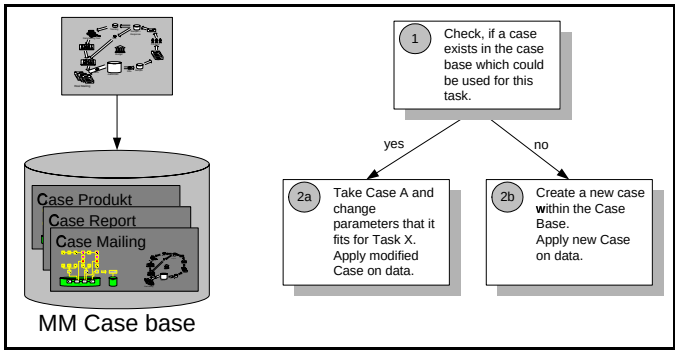


Figure 5: The CBR Approach

To meet these requirements, a case must have two associated parts:

a) a business-case description for retrieval

b) a re-usable technical specification/implementation of the KDD-process for this business case (e.g., the machine-processable representation of the steps 1-12 of the mailing-case)

Since case-retrieval is not an issue of this paper, we assume that a case is manually picked from a given list[6], presented to the user by name or by a graphic like the one in Figure 1. In the following, we will limit our focus on the representation and re-use of the preprocessing operations for data transformations, i.e. the pre-processing steps 1, 2, 3, 5 and 6, and the related data transformation operations 8 and 10 of the mailing-case. We will neither cover the data mining (modeling) step itself (step 7 in the mailing-case) nor post-processing (testing/evaluation/use, steps 9 and 11) in this paper. However, at least clustering,

[6] As long as we are limited to the mining business cases for the DWH of a single company or business unit this will probably be sufficient. For the whole Mining Mart project there will be a work-package concerned with business case description for retrieval.

- no 'unknown' (NULL) values are allowed for specific attributes
- scalar and ordinal attributes have to be numeric
- nominal attributes must have character values or be represented as sets of boolean values
- no numeric or no non-numeric attributes are admissible
- not more than N different values are allowed for nominal attributes
- always the same scale for numeric attributes is required
- no key attributes are respected
- input data must consist of a single flat table

Figure 6: Input restrictions of data mining tools

classification learning and regression could be formalized as pre-processing operations constructing new attributes, i.e., the new class, the predicted class and the predicted value (see sec. 3.1.2) within our framework.

To start with, the goals of pre-processing are:

1. to provide the most relevant data for a certain task,
2. to provide the data in a form most suitable for mining,
3. to fulfill the input restrictions of data mining tools (Figure 6 lists some typical restrictions), and
4. to generate useful and necessary background knowledge to be used for future tasks

Inherently, pre-processing is embedded in a certain case. Important is, however, that the selection of the data mining tool is part of the case as well. This frees the end-user from the difficult task of selecting a data mining tool (see Section 4.2) and ensures at the same time that pre-processing exactly matches the input restrictions imposed by the tool (see Figure 6). That means, the tool is selected and integrated into a case when the case is designed. The use of the tool during case execution is transparent for the end-user.

In the following, we discuss how to adapt a case to reuse it successfully on the new data. As we already showed, even just the selection of another target segment from the same DWH possibly requires different relevant attributes to be considered. In our framework design, we integrate the approach of multistrategy learning (MSL) [Michalski, 1991; Michalski and Kaufman, 1998] to solve this problem of case-adaption to new data. In particular, we will combine constructive induction [Mehra *et al.*, 1989], with feature selection [Liu and Motoda, 1998b; Liu and Motoda, 1998a]. Additionally, several base feature construction operations are based on learning methods, e.g. discovering optimal discretizations and groupings.

At a first sight, MSL seems to be a very promising approach to total automation, as it does not need large amounts of currently unknown knowledge, but relies on systematic or heuristic exploration on the space of possible data transformations. However, pre-processing and mining real-world data using solely this approach is not practicable; even heuristic MSL approaches will possibly get lost in the huge search space of possible data transformations, and the advantage of not needing domain knowledge becomes the disadvantage of not being able to use such domain knowledge to restrict the search space to a manageable size. Our case-based approach to integrate pre-processing and data mining can be seen as an attempt to set up an environment where available knowledge can be used to specify some transformations and especially the structure of the case manually, and where multistrategy learning can be successfully used:

- to automatically solve manageable sub-problems where corresponding knowledge is not available, and
- to locally optimize the adaptation of the case to new data.

3.1 Atomic Operations of Pre-Processing

The base-operations of pre-processing investigated in this paper are sampling and segmentation, feature construction within a table and over related tables, and feature selection[7]. In our approach, feature construction and dropping of base-features are separated, as features may be needed for more than one feature construction operation (e.g. the date of birth of a person is needed to construct the age of the person, the entry-age into a contract, and the end-age of a contract).

3.1.1 Sampling & Segmentation

Sampling and segmentation are used to reduce the number of data records within the training data. The underlying goals of these operations are the following:

- Improve speed and reduce memory requirements of the mining tool
- Focus on rare or special cases

[7] In [Morik, 2000] the pre-processing of time data is investigated, which will be part of the future Mining Mart, too.

- Rebalance the class distribution
- Specify a target group
- Use only clean data

These operations leave the number of attributes and the data records unchanged, however the statistical properties of the population may change dramatically. Sampling can only be applied to one data table.

Examples

- Random sampling or splitting
  Extracts data records out of the set of training data by random generation.
  Parameters: number of records to extract.
- Sampling based on typical cases / atypical cases
  Extracts data records which fulfill the (a)typical case within the training data.
  Parameters: Condition of the (a)typical case.
- Segmentation
  Extracts data records which fulfill a special segmentation-pattern.
  Parameters: Segmentation condition (e.g. 2.a and 2.b).
- Data quality-motivated sampling
  Extracts only data records with values having a certain degree of quality.
  Parameters: Quality-condition (e.g. all records which have less than 2 unknown values).
- Rebalancing
  Noise-tolerant mining tools cannot be used to build a classification for very unbalanced class distributions (e.g. 95% – 5%, with 5% being an optimistic estimate for responses of a mailing.), a sampling favoring the members of the minority class has to be applied first.
  Parameters: class distribution (e.g. 55% – 45%) after rebalancing.

3.1.2 Attribute Construction within a Relation

Simple attribute construction creates a new attribute within one data table or view. The new attribute is based on one or more base attributes and groups their values into a more general form. The total number of data records is the same as before the operation. However, if base attributes are replaced with newly created ones, the number of distinguishable data records is possibly reduced (exception make relativation and rescaling).

Goals

- Improve data-coding relative to the capabilities of the mining tool
- Create a new attribute which can be better used for the mining task

Examples

- Discretization
  Is applied to attribute(s) of the type ordinal or scalar and creates a new attribute of the type nominal ordered
  (ordinal/scalar → nominal ordered).
  Parameters: base attribute(s), output attribute, number of created intervals for the output attribute (e.g. values of the income-attribute(s) are grouped into i0, i1..., i10).
- Grouping
  Is applied to attribute(s) of the type nominal which has/have no hierarchy and creates a new nominal unordered attribute.
  (nominal unordered → nominal unordered).
  Parameters: base attribute(s), output attribute, number of created groups, number of data records in one group (e.g. profession descriptions are grouped into groups of professions).
- Abstraction
  Is applied to attribute(s) of the type nominal or scalar and creates a new attribute of the type nominal ordered.(nominal, scalar → nominal ordered).
  Parameters: base attribute(s), output attribute, hierarchy, output of hierarchy level (e.g. looking at the household level instead of person level).
- Relativation
  Puts one attribute in relation to another attribute. This works only on numeric or date attributes and doesn't change the number of different data records.(numeric, date → numeric ordered).
  Parameters: base attributes, output attribute, operation between the base attributes (e.g. calculating the age from Sysdate and birthdate, calculating the quotient of income and premium sum).
- Cleaning
  Eliminates rare values of data records by creating a new attribute.(any type → any type).
  Parameters: base attributes, output attribute, which value of the base attribute shall be replaced by which new value (e.g. replacing the entry age of a person smaller than one by one).

- Unknown elimination
  Replaces unknown values with a specified new value.(any type → any type).
  Parameters: base attributes, output attribute, specified replacing value (e.g. replacing the unknown values of attribute age by the mean value or most frequent value).

- Scaling
  For all distance-based mining tools (e.g. clustering and instance based learning) the scale of the numeric attributes is very important, i.e. attributes with larger values are more influential on the result. To avoid this usually unintended weighting of attributes, all attributes have to be rescaled, e.g. to a fixed standard deviation or interval. (scalar→ scalar).
  Parameters: standard deviation or interval

3.1.3 Multi-Relational Attribute Construction

Joins and aggregations are used to put information from several related tables into one base table. To avoid unwanted changes of the target population distribution, the object identity within the base table has to remain unchanged (the number of different data records is still the same after the operations). To avoid the loss of base-table record-joins, outer-joins should generally be used, and to avoid the duplication of base-table records, simple joins must not be used for 1:N or N:M related tables (as this would duplicate records in the base table; e.g., it is not a good idea, to send duplicated mail to a household, for every person and contract involved). Instead the aggregation operations of this section have to be used[8].

Goals

- Fit the single table requirement of most DM-tools
- Reduce the complexity for ILP-DM-tools
- Avoid unwanted changes of the population distribution.

[8]The design of these operations is in a way inspired by theoretical results in Inductive Logic Programming (ILP) on how to translate determinate hornclauses into propositional logic [Džeroski *et al.*, 1992; Kietz and Džeroski, 1994], which are also used in the ILP-system DINUS [Lavrač and Džeroski, 1994] for propositionalisation. Newer and more general propositionalisation approaches like [Alphonse and Rouveirol, 1999] are not used, as they are based on duplications of records in the base-table, which does not seem to be adequate in this context. Instead, we use operations inspired by Description Logics (DL) [Brachman and Schmolze, 1985] and the constructive induction operations of the DL-learning system KLUSTER [Kietz and Morik, 1994] to generate determinate features from indeterminate relations. These relations will also be further investigated in a future Mining Mart workpackage.

Examples

- Sum
  Creates a scalar attribute, e.g. premium sum of a product, income of a household.

- Min, Max
  Creates a scalar ordered attribute, e.g. smallest, highest premium sum of a product, smallest age of a member of the household.

- Count different
  Creates a numeric attribute, e.g. how many persons a household has, how many insurance contracts a household has.

- Count X = V
  Creates a numeric attribute, e.g. how many children has a household, how many different values has an attribute.

- Exist X = Y
  Creates a binary attribute, e.g. exists an insurance contract of type 3a for one household.

- All X = Y
  Creates a binary attribute, e.g. are all insurance contracts of a household of type 3a, are all persons of a household adults.

3.1.4 Attribute Selection

Attribute selection drops attributes which should not be in the mining input and/or result, since they are, e.g., clearly uninteresting, not usable or difficult or expensive to measure for new data. The number of different data records and also the number of the total data records remains the same.

Goals

- Drop attributes which do not fit the input requirements of a data mining tool
- Drop attributes which are strongly dependent on or the base of attribute construction for other attributes
- Improve processing speed
- Guide the build-in attribute selection process of the data mining tool

Examples

- Feature / attribute selection
  Only the important attributes are chosen as input for the data mining tool. Selecting the attributes can be done manually, through input-restrictions associated with the mining tool or by feature selection methods [Liu and Motoda, 1998b; Liu and Motoda, 1998a].

3.2 Construction of Pre-Processing Cases

The development process of pre-processing cases is performed on training data. Each case has to be represented by a chain of several pre-processing operations having specific parameters and a defined order of their execution. An operation is either user-defined (so-called manual operation) or supported by an existing MSL tool. In order to develop an optimal pre-processing chain, the power-user has to execute several iterations to find the best fitting (global optimized) chain of pre-processing operations and MSL tools; the integrated MSL tools help him to automatically find a locally optimized solution. After all iterations are completed, several pre-processing chains with mining results exist. The chain with the best result has to be chosen as the best fitting one[9]. Then, this chain is executed on the test and application data as well (e.g., steps 8 and 10 of the mailing case).

A single iteration on the training data consists of the following steps:

1. Chain specification. MSL tools are selected and manual operations are either newly defined[10] or existing ones are reused. Operations have to be linked to build the chain: each takes as input the result of the previous operation and passes the output to the next operation.

2. SQL generation. Manual pre-processing operations are generated as SQL statements.

3. Chain execution. The execution of SQL statements is interwoven with calls of MSL tools which are standalone executables. The whole chain performs the required data transformations; at the end of execution, the data is ready to be loaded by the mining tool.

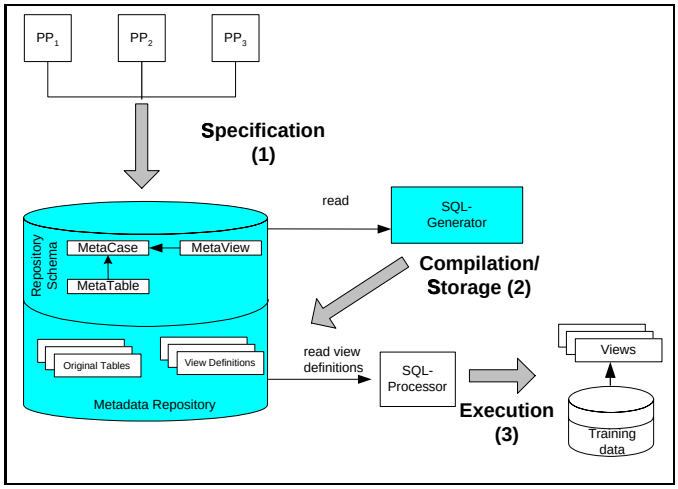


Figure 7: Development and execution of manual operations

3.2.1 Handling Manual Operations

In Figure 7, the three steps are depicted for manual operations. During specification (1), the operation descriptions (PP) are read, decomposed and mapped into reusable SQL code parts (i.e., metadata elements) which are stored in a repository according to a suitable metadata schema. SQL statements are generated through the execution of a Java program which takes as input the metadata elements and brings them together in the required order, with the required syntax. For example, when creating a view, the metadata to be used includes the view name, view attributes, transformation functions applied to original attributes, constraints, the original tables with their description elements (name, attributes, primary and foreign keys), etc. The SQL generation, also called compilation (2), has as output an SQL statement which is stored into the repository as well. The execution (3) of the statement creates a view and either writes the definition of the view into the data dictionary of the training database or lets the view definition in the repository and only builds the (net) connection to the database to get the required data when the view is used (remote access).

Inherently, the repository also contains informations needed to run the MSL tools (paths, configuration files, etc). In this way, whole chains are stored into the repository; parts of them may be reused. Besides reusability, the

[9] For future versions the other ones could be stored as variants in the case base which help the adaption of a case to a new target segment.

[10] For supporting operator specification, a visual programming environment is planed in the Mining Mart project. At present, SQL is used.

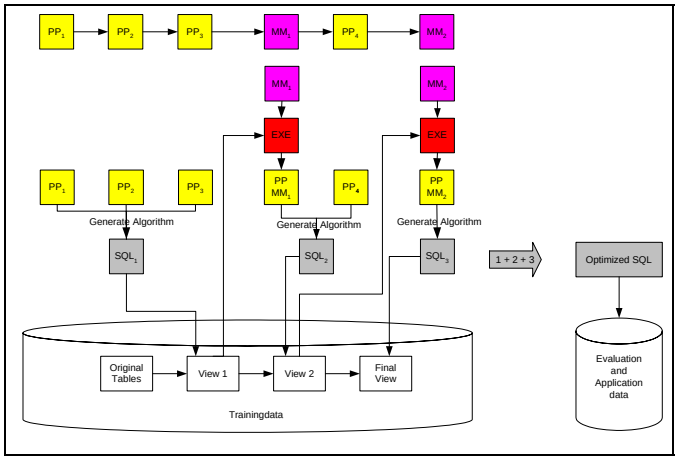


Figure 8: Chain of pre-processing operations

use of the metadata repository for development and execution provides a better documentation. Operations, chains and tools may be accompanied by extensive descriptions of their tasks, executions, required parameters, etc. The fact that operations and their descriptions are managed together in an uniform way increases the software quality and implicitly its maintenance. Additional statistical information regarding the used data is also available in the repository, e.g., nominal and cardinal attributes, response-rate, etc. Furthermore, the repository stores descriptions of the business cases and the links to the operation chains implementing them. This information is needed by the end-user to choose the right business case for execution.

3.2.2 Integrating MSL Tools into the Chain

Figure 8 shows a complete chain of pre-processing operations and MSL tools and also how SQL-statements are generated from that. The chain contains different manual pre-processing operations (PP) and MSL tool supported pre-processing operations (MM) in a specific order. The MSL tools are used to discover the parameters of the associated manual pre-processor operations (i.e., of their successors in the chain). Manual pre-processing operations are translated into SQL-code. MSL tools are stand-alone executables. The result of a SQL-statement is a view in the database which acts as an intermediate result, i.e. the view is needed as the input for a following MSL tool. In turn, the result of a MSL tool is used as input parameters of the associated SQL-code.

As an example consider the discovery of discretization parameters as the task of a MSL tool. For a given data set (i.e., the training data) and a number of discrete values (e.g., 2) the MSL tool finds the best mapping of intervals of the base-attribute (e.g., AGE) to nominal values of the new attribute (e.g., AGE_CATEGORY). The meaning of the mapping may be {if $AGE < 18$ then $AGE\_CATEGORY$ = 'CHILD', else $AGE\_CATEGORY$ = 'ADULT'}. That means, the next operation takes as input the old attribute, AGE, the automatically found parameter value, in our case 18, and the two discrete values, 'CHILD' and 'ADULT' (see Figure 9) and generates an (intermediate or final) view that has a new attribute AGE_CATEGORY taking only the two discrete values.

```
CREATE VIEW New_View (name,..., age_category) AS
SELECT name,..., Discretization(age,18,'ADULT','CHILD')
FROM Partner;

FUNCTION Discretization(age IN NUMBER, bound IN
NUMBER, value1 IN VARCHAR, value 2 IN VARCHAR2)
RETURN VARCHAR2 AS decoded VARCHAR2(50) :=NULL;
BEGIN
decoded:= DECODE(SIGN(age - bound), 1, value1,
-1, value2, value1);
RETURN (decoded);
END;
```

Figure 9: View with an attribute on which discretization has been applied

After the whole chain is executed on the training data, the final view and several SQL-statements exist. Since the results of the MSL tools are now available, they may be embedded into the SQL statements. These may be merged and optimized to one statement (see Figure 8). This statement or the single generated SQL statements are executed on the test data (step 8 of the mailing case) or the application data (step 10 of the mailing case).

4 Related Work

Precondition for the success of the Mining Mart framework is the existence of a user-friendly and efficient KDD environment. We intend to achieve this precondition through a KDD support environment (KDDSE, [Brachman and Anand, 1996]) that advances the state of the art of current KDDSEs (Section 4.1). Such an advanced KDDSE will utilize research results for semi-automatic process planning support as discussed in Section 4.2.

4.1 KDD Support Environments

The KDD-process itself is nicely defined in the CRISP-DM process model [Reinartz *et al.*, 1998]. The pre-

processing phase is the most time consuming task for practical applications. Therefore we discuss the support given by KDDSEs in this process. Data mining environments that support pre-processing generally only support internal pre-processing of data already loaded into the KDDSE. It is problematic, if not impossible, to force the content of a data warehouse into the KDDSE before applying the first pre-processing operator. However, with the announcement of the SPSS Clementine-Server and the better integration of DB2 and IBM's Intelligent Miner this situation is currently changing. Another problem is, that the result of a pre-processing operation is often stored extensionally. This makes the reuse of operations difficult and furthermore, if a problem compounded of a series of pre-processing operations is executed, several nearly identical copies of the original data set must be stored to prevent the loss of the intermediate steps required for further documentation and re-application purposes.

As an alternative, most KDDSEs allow the use of manually specified SQL expressions. However, specifying pre-processing operations in SQL is difficult, and the user is forced to select a sample of the database (via SQL) to be able to conduct the necessary pre-processing inside the KDDSE. The pre-processing environment to be developed in this project introduces support for in-database pre-processing operations. Particular research issues address the introduction of operations suited for multi-relational databases. The importance of support for multi-relational pre-processing is increasing with the introduction of commercial KDDSEs supporting multi-relational analysis (Kepler and Clementine after completion of the Aladin project). Multi-relational analysis is the next step in the evolution of KDDSEs towards allowing analysis of complex, large, and most importantly, relational company data warehouses.

4.2 Semi-automatic process planning support

Beginning with the MLT-Consultant [Sleeman *et al.*, 1989] there was the idea of having a knowledge based system supporting the selection of a machine learning method for an application. The MLT-Consultant succeeded in differentiating the nine MLT learning methods with respect to specific syntactic properties of the input and output languages of the methods. However, there was little success in describing and differentiating the methods on an application level that went beyond the well known classification of machine learning systems into classification learning, rule learning, clustering, and sloppy modeling. Also, the STATLOG ESPRIT-Project [Michie *et al.*, 1994], which systematically applied classification learning systems to various domains, did not succeed in establishing criteria for the selection of the best classification learning system. It was concluded that some systems have generally acceptable performance; and in order to select the best system for a certain purpose, they must each be applied to the task and the best be selected through a test method such as cross-validation. Theusinger and Lindner [1998] are in the process of re-applying this old idea of searching for statistical dataset characteristics necessary for the successful applications of DM-tools. An even more demanding approach was started by Engels [1997]. This approach not only attempted to support the selection of DM-tools, but built a knowledge-based process planning support for the entire KDD-process. Until today this work has not led to an usable system [Engels *et al.*, 1997]. The European project MetaL now aims at learning how to combine learning algorithms and datasets. We do not believe that this top-down knowledge-based approach will lead to an usable environment in the short run, as it requires a large amount of very application-specific knowledge. Furthermore, it is widely agreed upon that even the manual KDD-process cannot be planned ahead of time in detail. The utility of an operation can often only be determined after a large number of further operations is executed. It is apparent that not enough knowledge is available to propose the correct combination of pre-processing operations. However, it is possible to collect knowledge to exclude illegal, meaningless and unsuccessful combinations of operations. Therefore, we propose to use case-based semi-automatic process planning. Our goal is a system which supports a group of experienced data mining and domain experts in creating initial cases of the KDD-process for a specific application (e.g. the mailing action) and a specific type of data (e.g. a company's data warehouse). The system then offers these cases to domain experts, and supports them in repeating the KDD-process on new data of the same type (e.g. the same data warehouse, updated with new data, the result of the last mailing, etc.) for a similar application (the next mailing action). Hence, only the first use/creation of a case will require substantial effort and DM-experience, whereas all further uses of this case will be much quicker and more inexpensive. Additionally, a larger number of data mining applications can be executed with a limited number of available DM-experts. This case-base may even be useful to acquire knowledge about the KDD-process itself, e.g. by the Meta-Level Learning Methods developed in MetaL. This information could be utilized for more sophisticated process-planning support of initial cases for new applications.

5 Conclusion

The Mining Mart framework described in this paper builds on the insight that current approaches for achieving the objectives described above tend to ignore theoretical results, which have been proven that no algorithm can claim to be systematically better than any other on every problem [Wolpert and Macready, 1995], and that nobody has yet been able to identify reliable rules for predicting, that one algorithm should be superior to others, i.e. a total automation of the KDD-process is not possible.

A constraint based graphical user interface based on the KDDSE Kepler [Wrobel *et al.*, 1996] utilizing metadata shall guide users through the knowledge discovery task. The highest possible degree of automation for this process will be the aim of this project. However, as reasoned above, it cannot be expected that the user simply asks a high level question and selects a data set to be analyzed and everything else is done automatically. In particular, the task of proper transformation of the given data into a format that can be successfully analyzed by the available algorithms is difficult. As discussed above, testing of all possible approaches through pure multistrategy learning is currently not practical because the required computational power is not accessible for any single user. However, user that have access to the Mining Mart can search the case-base for suitable solutions to their task at hand. If no proper solution is found, the task will be posted as a new challenge to the knowledge discovery experts.

The main innovation of this project will be the deep integration of the different research directions currently accessible only to experts into an uniform environment usable also by data mining non-experts.

Acknowledgements: This work has been partially found by the Swiss Government under the contract-no "BBW Nr.99.0158" as part of the European commision Research Project IST-1999-11993 (Mining Mart). We thank Cèline Rouveirol and Ulrich Reimer for very helpful comments on a draft of this paper and all partners of the mining mart project with whom we had fruitful discussions about the framework described in this paper.

References

[Alphonse and Rouveirol, 1999] E. Alphonse and C. Rouveirol. Selective propositionalization for relational learning. In *Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*. Springer Verlag, 1999.

[Brachman and Anand, 1996] R. Brachman and T. Anand. The process of knowledge discovery in database. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[Brachman and Schmolze, 1985] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171 – 216, 1985.

[Džeroski *et al.*, 1992] S. Džeroski, S. H. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proc. Fifth ACM Workshop on Computational Learning theory*, pages 128–135. ACM Press, New York, 1992.

[Engels, 1997] R. Engels. Planning tasks for knowledge discovery in databases; performing task-oriented user-guidance. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 170. AAAI Press, 1997.

[Engels *et al.*, 1997] R. Engels, G. Lindner, and R. Studer. A guided tour through the data mining jungle. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 163. AAAI Press, 1997.

[Inmon, 1996] W. H. Inmon. The data warehouse and data mining. *Communications of the ACM*, 39(11):49–50, November 1996.

[Kietz and Džeroski, 1994] J.-U. Kietz and S. Džeroski. Inductive logic programming and learnability. *SIGART Bulletin*, 5(1), 1994.

[Kietz and Morik, 1994] J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2):193–217, 1994.

[Lavrač and Džeroski, 1994] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, Chichester, England, 1994.

[Ling and Li, 1998] C. X. Ling and C. Li. Data mining for direct marketing: Problems and solutions. In R. Agrawal and P. Stolorz, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 73–79. AAAI Press, 1998.

[Liu and Motoda, 1998a] H. Liu and H. Motoda. *Feature Extraction Construction and Selection, A Data Mining Perspective*. Kluwer Academic Publishers, 1998.

[Liu and Motoda, 1998b] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers, 1998.

[Mehra *et al.*, 1989] P. Mehra, L. Rendell, and B. Wah. Principled constructive induction on decision trees. In *Proc. of the Eleventh Int. Joint Conf. on AI (IJCAI-89)*, pages 651–656. Morgan Kaufman, Los Altos, California, 1989.

[Michalski, 1991] R. S. Michalski. Inferential learning theory as a basis for multistrategy task-adaptive learning. In *Proceedings of the first International Workshop on Multistrategy Learning*, pages 3 – 18. George Mason University, 1991.

[Michalski and Kaufman, 1998] R. Michalski and K. Kaufman. Data mining and knowledge discovery: A review of issues and a multistrategy approach. In R. Michalski, I. Bratko, and M. Kubat, editors, *Machine Learning and Data Mining Meathods and Applications.* John Wiley & Sons LTD, Chichester, England, 1998.

[Michie *et al.*, 1994] D. Michie, D. Spiegelhalter, and C. Taylor, editors. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, Chichester, England, 1994.

[Morik, 2000] K. Morik. The representation race – preprocessing for handling time phenomena. In *Proc. of the European Conference on Machine Learning, ECML-2000.* LNAI, Springer Verlag, 2000.

[Pyle, 1999] D. Pyle. *Data Preparation for Data Mining.* Morgan Kaufmann Publishers, San Francisco, California, 1999.

[Reinartz *et al.*, 1998] T. Reinartz, R. Wirth, R. Clinton, T. Khabaza, J. Hejlesen, and P. Chapman. The current crisp-dm process model for data mining. In F. Wysotzki, P. Geibel, and K. Sch"adler, editors, *Beitr"age zum Treffen der GI-Fachgruppe 1.1.3 Machinelles Lernen (FGML-98).* Technical Report 98/11, Technical University Berlin, 1998.

[Sleeman *et al.*, 1989] D. Sleeman, R. Oehlman, and R. Davidge. Specification of consultant-0 and a comparision of several learning algorithms. Mlt-deliverable d5.1, Machine Learning Toolbox Esprit Project P2154, 1989.

[Staudt *et al.*, 1998] M. Staudt, J.-U. Kietz, and U. Reimer. A data mining support environment and its application on insurance data. In R. Agrawal and P. Stolorz, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 105–111. AAAI Press, 1998.

[Staudt *et al.*, 1999a] M. Staudt, A. Vaduva, and T. Vetterli. Metadata management and data warehousing. Technical report, Information Systems Research, SwissLife, 1999. http://research.swisslife.ch/Papers/papers.htm.

[Staudt *et al.*, 1999b] M. Staudt, A. Vaduva, and T. Vetterli. The role of metadata for data warehousing. Technical report, Information Systems Research, SwissLife, 1999. http://research.swisslife.ch/Papers/papers.htm.

[Theusinger and Lindner, 1998] C. Theusinger and G. Lindner. Benutzerunterstützung eines kdd-prozesses anhand von datencharackteristiken. In F. Wysotzki, P. Geibel, and K. Schädler, editors, *Beiträge zum Treffen der GI-Fachgruppe 1.1.3 Machinelles Lernen (FGML-98).* Technical Report 98/11, Technical University Berlin, 1998.

[Wolpert and Macready, 1995] D. Wolpert and W. Macready. No free lunch theorems for search. Techinal Report SFI-TR-95-02-010, Santa F Institute, Santa F, CA., 1995.

[Wrobel *et al.*, 1996] S. Wrobel, D. Wettschereck, E. Sommer, and W. Emde. Extensibility in data mining systems. In E. Simoudis and J. Han, editors, *Proc. of the 2nd Int. Conf. On Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA, USA, 1996.

also appears in: Fullerton, K. (ed.) Proceedings of the 6th EC-GIS Workshop, Lyon, 28-30th June, European Commission, JRC, Ispra

SPIN! – an Integrated Spatial Knowledge Discovery Platform

Michael May
GMD - German National Research Center for Information Technology
Institute for Autonomous intelligent Systems
Knowledge Discovery Team

Abstract
The overall objective of the European IST SPIN!-project is to develop a web-based spatial data mining system by integrating state of the art Geographic Information System (GIS) and data mining functionality in a closely coupled open and extensible system architecture. The state of the art in data mining will be advanced by adapting methods from machine learning, esp. inductive logic programming, and Bayesian statistics, esp. Markov Chain Monte Carlo, to spatial data analysis. The state of the art in GIS will be advanced by developing new methods for the visualization of spatial and temporal information.

Spatial Knowledge Discovery

The GIS revolution in the early 1980s has brought about an explosion of geographically referenced information, wherein much of the data is also temporally referenced. This poses great challenges to commercial enterprises and government organizations that have been left with few tools to extract and to disseminate useful information from these huge amounts of data.

Internet enabled GIS

In the last few years a new generation of Geographic Information Dystems has been emerging that enable interactive, dynamic maps to be disseminated via the Internet (Andrienko and Andrienko 1999b, Dykes 1997, Gitis et al. 1998, Openshaw et al. 1998). While being an exciting development for automating cartography, most of these systems are confined to projecting descriptive statistical displays, such as histograms or pie charts, onto geographical space (maps). Using these projected map displays, more advanced decision making and inference is not always straight-forward. Although the human visual system is highly effective in identifying patterns, this process is subjective and can be influenced by systematic errors (cf. e.g. Diaconis 1985). Secondly, it is very hard to visualize attribute interaction on a map having more than a few dimensions. Hence, complex multi-variate dependencies are easily overlooked.

Data Mining

Searching for multi-variate dependencies is where *data mining* promises great benefits. Data mining is the partially automated search for hidden patterns in typically large and multi-dimensional databases. Data mining techniques have been developed in areas such as machine learning, statistics and database theory (Fayyad et al. 1996; Klösgen & Zytkow 2000). Some of these techniques, such as *k*-nearest neighbor, are extensions of statistical techniques known for a long time. Others, especially from the area of machine learning and inductive logic programming (ILP), are essentially new (cf. Mitchell 1997 for a good introduction).

These techniques have been packaged in so-called *data mining platforms*. A data-mining platform is a software environment providing support for the application of one or more data-mining algorithms. Some platforms provide user friendly interfaces and visual programming environments that a non-expert can also use. Interest in data mining has boomed in recent years, fuelled by advances in data warehousing.

The process of finding patterns and extracting useful knowledge from data is known as the *knowledge discovery cycle***.** Data mining is a part of this larger cycle.[1] The cycle comprises the steps of data access and selection; data-cleaning and transformation; data mining (the analysis step); and visualization and interpretation (fig. 1).

Comprehensive data mining systems aim to support all stages of this cycle. Knowledge discovery has been applied to many kinds of problems, including stock market prediction, credit scoring, astronomical classification, and molecular biology (for a comprehensive overview, see Klösgen and Zytkow, 2000).

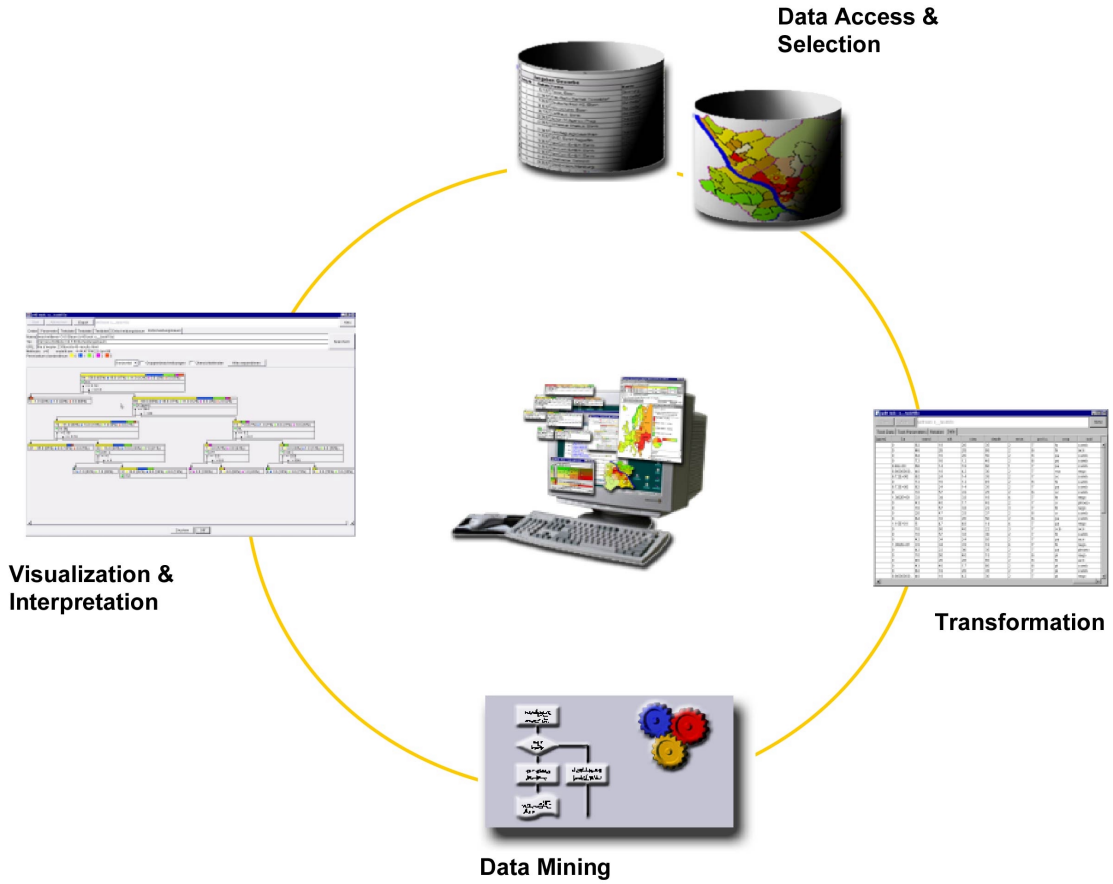


Figure 1: *Knowledge discovery cycle, comprising the steps of data access and selection, transformation, data mining (analysis), interpretation and visualization.*

Combining data mining and GIS

What benefits does data mining offer for the GIS user? Data mining and geographical information systems are best seen as complementary tools for describing and analyzing data. Whereas in GIS the user guides the search and generates hypotheses, data mining partially delegates this task to the computer, preselecting and presenting to the analyst only those patterns deemed most interesting (according to some measure of quality). Whereas GIS relies on visualization in geographical space, data-mining is hunting for patterns in multi-dimensional abstract space. Whereas a GIS query lets the user view what is inside the database, data mining typically performs *inductive generalizations*, generating patterns whose

[1] Note that terminology is not consistent here; the terms *data mining* and *knowledge discovery* are sometimes used synonymously.

logical content *exceeds* the content of the database, producing new and sometimes surprising knowledge.

Both techniques are essentially exploratory, leaving the final decision of whether a hypothesis is an important new finding (a "nugget" in data mining language) or just an artifact to the analyst.

How are spatial data handled within the knowledge discovery cycle? Although many data-mining applications deal at least implicitly with spatial data they essentially ignore the spatial dimension of the data, treating them as non-spatial (exceptions in the data mining community are Koperski & Han 1997, Ester et al. 1999, Klösgen 1998).

This has ramifications both for the analysis of data and for their visualization. First, one of the basic tasks of exploratory data analysis is to present the salient features of a data set in a format understandable to humans, and visualization in geographical space is known to be much easier to understand than visualization in abstract space. Secondly, results of a data mining analysis may be sub-optimal or even be distorted if unique features of spatial data, such as spatial autocorrelation (cf. Haining 1991), are ignored.

In sum, convergence of GIS and data mining in an internet enabled spatial data mining system is a logical progression for spatial data analysis technology. Related work in this direction has been done by Koperski and Han (1997) and Ester et al. (1999); an integration has also been proposed by MacEachren and Wachowitz (1999). The range of application areas is huge and there are many different types of applications in statistical analysis, urban planning, environmental decision making, and geomarketing.

The SPIN!-project

This paper describes the most comprehensive and ambitious attempt to bring together some of the best results in Data Mining and interactive thematic mapping to date, carried out under the European Commission Fifth Framework IST programme (IST-1999-10536 SPIN!). Partners come from five different European countries: Germany, Italy, Netherlands, UK, Russia. The project started in January 2000, its duration is 3 years. It is co-ordinated by GMD.

Building a spatial mining system is a demanding interdisciplinary task, requiring expertise in many fields including Geographic Information Systems, cartography, statistics, machine learning, and databases, as well as software engineering skills. The consortium has been chosen to reflect these skills. It includes

- a university and a national research center for computer science active in the areas of Data Mining, Statistics, Machine Learning, and GIS: **University of Bari**, Italy; **German National Research Center for Information Technology** (GMD), Bonn,
- an institute for geography active in exploratory spatial data analysis: **School of Geography, University of Leeds**, UK,
- two industrial partners active in data mining and Geographic Information Systems: **Dialogis Software & Services GmbH**, Bonn, Germany; **Professional GeoSystems (PGS)**, Amsterdam, Netherlands; and, on the application side,
- two universities having a leading role in the dissemination of statistical data in the UK: **Metropolitan and Victoria University, Manchester**, MIMAS project; and
- two institutes active in seismic data research: **IITP**, **Russian Academy of Sciences**, Moscow; **GeoForschungszentrum Potsdam**, Germany.

The overall objective of SPIN! is to develop a state of the art, extendable and internet-enabled GIS-Data-Mining platform. Data mining and GIS are quite complex tools with wide ranging functionality, so the *SPIN! Consortium* does not propose to start from scratch, but to build on existing tools. In recent years, a number of project partners have developed the technological components and scientific tools that are needed to develop the kernel of this type of spatial data mining system. During the project these individual efforts and the associated expertise and experience will be united in a joint effort to develop and integrate the missing pieces.

Specific application areas chosen for the project are volcano and earthquake research and decision support for urban development plans. Applications outside the SPIN!-project will include environmental issues, especially biodiversity (May 1999).

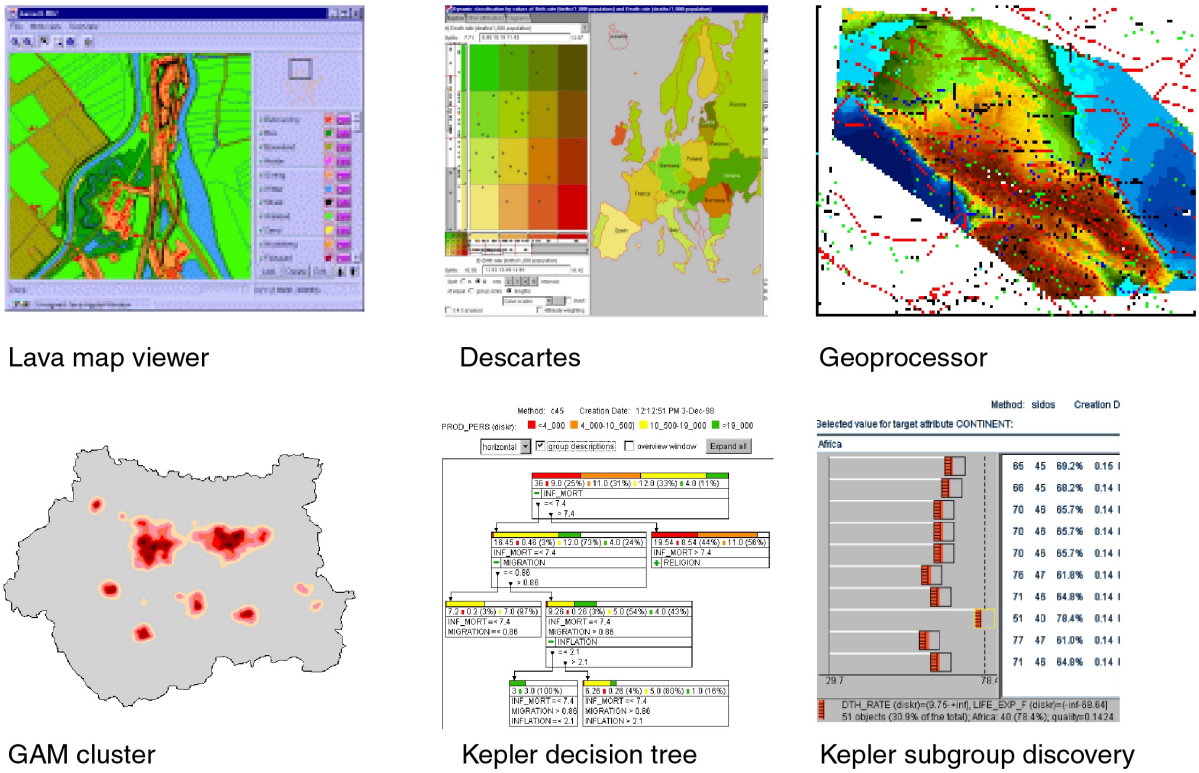


Figure 2: Elements of SPIN!. *From left to right: Lava map viewer, interactive classification produced by Descartes, topography relief in Geoprocessor spatial clusters found by GAM, visualization of decision trees and subgroup mining in Kepler.*

SPIN!: The Elements

This section describes the existing systems that will be extended and integrated during the project (fig. 2). To describe the functionality of the SPIN!-system, it is useful to distinguish five levels of functionality.

Level 1: Data access and management

The basis functionality will be provided by the data mining platform Kepler, jointly developed by GMD and Dialogis (Wrobel et al. 1996). Besides its data mining methods, discussed below, it already provides the following features:

- *data access* to heterogeneous data sources (JDBC-compliant databases, flat files, spatial data interfaces etc.), also over the internet,

- *data transformation* capabilities for discretization, restriction, projection, union, join, and calculated rows,
- *exploratory non-spatial visualization* using histogram, scatter-plots, or pie charts,
- facilities for *organizing* and *documenting* analysis tasks.

Special features of Kepler are: its plug-in architecture that allows to integrate third party analysis tools; the capability to analyze multi-relational (multi-table) data without performing explicit joins; and its support for inductive logic programming tools (Wrobel et al. 1996).

The current architecture will be redesigned to allow for a seamless integration with the other components, including accessing and organizing geometry data, map objects etc. and handling basic server-side tasks such as multi-user access and security.

Level 2: Internet-enabled Geographic Information System for displaying maps

As a basic map viewer we use the Lava/Magma system developed by PGS. It can display a map consisting of several layers and supports basic operations such as zooming, panning, querying features and changing visual properties such as colour, fill styles, drawing of labels. The client is implemented in Java and is operable over the internet. An important reason for using it is its advanced caching mechanism, leading to good scalability (van den Berg et al. 1999).

Level 3: Interactive thematic mapping for visualizing statistical data

For visual exploratory spatial analysis the Descartes module for interactive manipulation of statistical maps is used (Andrienko & Andrienko 1999b). To provide automated visualization of thematic maps, Descartes incorporates the knowledge of thematic cartography in the form of generic, domain-independent rules. To choose the adequate presentation techniques for given data, it takes into account data characteristics and relations among data components or attributes. The automation of map generation releases the user from the necessity of thinking about how to present the data and from the routine work of map building; it allows the user to concentrate on the analysis of his data. Among its features are linked displays, interactive cross-classification, box-plots and a module for temporal visualisation. The latter will be greatly extended and improved during the project (cf. Andrienko et al., this volume).

Level 4: Spatial cluster detection

Descartes can be used for interactive, visual identification of spatial clusters. Yet the SPIN!-system also contains modules for performing this search automatically. The objective of the *Geographical Analysis Machine* GAM (Openshaw 1998, Openshaw et al. 1999) is to look for local spatial clusters without knowing in advance where to look. GAM works by examining a large number of overlapping circles of varying sizes that completely cover a region of interest. E.g., to identify cancer clusters, it uses a population at risk count and an incidence rate, comparing the relative frequency for instances within a circle with an expected value; statistically significant circles are then retained. Then a kernel smoothing procedure is applied to the circles to produce a smoothed density surface, giving results as displayed in fig. 2.

Whereas GAM looks for a single attribute, a second tool, the *Geographic Exploration Machine* GEM (Openshaw 1998) can take also account attribute interaction. Since search in large attribute spaces leads to a combinatorial explosion, a main task of SPIN! will be to find more efficient search strategies; some initial tests using genetic algorithms have already been performed.

Level 5: Explaining clusters and spatial phenomena

Assume we have found a spatial cluster or interesting classification, using either the interactive approach of Descartes or the automated search of GAM. *What attributes are associated with a cluster that could potentially explain it*? To answer this question, spatial and aspatial analysis methods can be applied.

It is generally accepted that there currently exists no single best data mining method. Available methods differ in terms of complexity, representational power, accuracy, comprehensibility, and assumptions made about the data. It is therefore important that users have access to a variety of spatial data mining methods. Kepler already contains a variety of non-spatial data mining methods for *k*-nearest neighbor, decision and regression trees, association rules, subgroup discovery, and rule-based methods from inductive logic programming. It also provides visualizations for these methods in abstract space.

In developing SPIN! the state of the art in spatial data mining will be advanced along several routes. Since all these methods can be launched within a single, coherent platform, the project can also contribute to a comparison of the relative strengths and weaknesses of the methods and develop guidelines for their use in spatial mining.

In a GIS spatial query module, a user can ask questions such as "show me all houses with more than 5 rooms within 1 km distance to a kindergarten, and close to a shopping center". The database returns all instances satisfying this rule. In this case, the user already knows the rule. But what if we do not know a rule but have some positive and negative examples? E.g. we know several places where a rare plant species occurs and would like to know what environmental site conditions are favorable for their occurrence. This may include topological features such as being close to a river. While traditional attribute-value based learning methods have difficulties in expressing topological features such as `close_to`, `adjacent_to` etc. in a natural and general way, they can be easily expressed in first-order-logic.

This makes *inductive logic programming* (ILP), which uses a first-order representation, a natural and promising approach to many forms of spatial data mining. During the project we will specifically investigate spatial association rules and subgroup discovery (Malerba et al. 1998, Klösgen 1998, Wrobel 1998). A further topic is the automated interpretation of topographic maps (Esposito et al. 1998). In this case, symbolic first-order descriptions of cells of a map are automatically extracted from a vector representation of maps stored in an object-oriented database.

In a second line of research we approach *Bayesian statistics*. In the last years computationally intensive Bayesian methods have been developed that compare favorably with classical approaches. Instead of selecting an "optimal" model they generate a whole distribution of models which characterize their uncertainty in the light of the available data. Paaß and Kindermann (1998) have developed Bayesian classification methods based on Markov Chain Monte Carlo which use a Bayesian ensemble of decision trees or neural networks. These methods have already been successfully applied to other areas and will now be adapted to spatial data.

A third line of research is the work already mentioned that combines the GAM approach with search along the temporal dimension and in larger attribute spaces (Openshaw et al. 2000).

The way data mining results are presented to the user is crucial for their appropriate interpretation. The approach to be taken is a combination of cartographic and non-cartographic displays linked together through simultaneous dynamic highlighting of the corresponding parts (Andrienko et al. 1999).

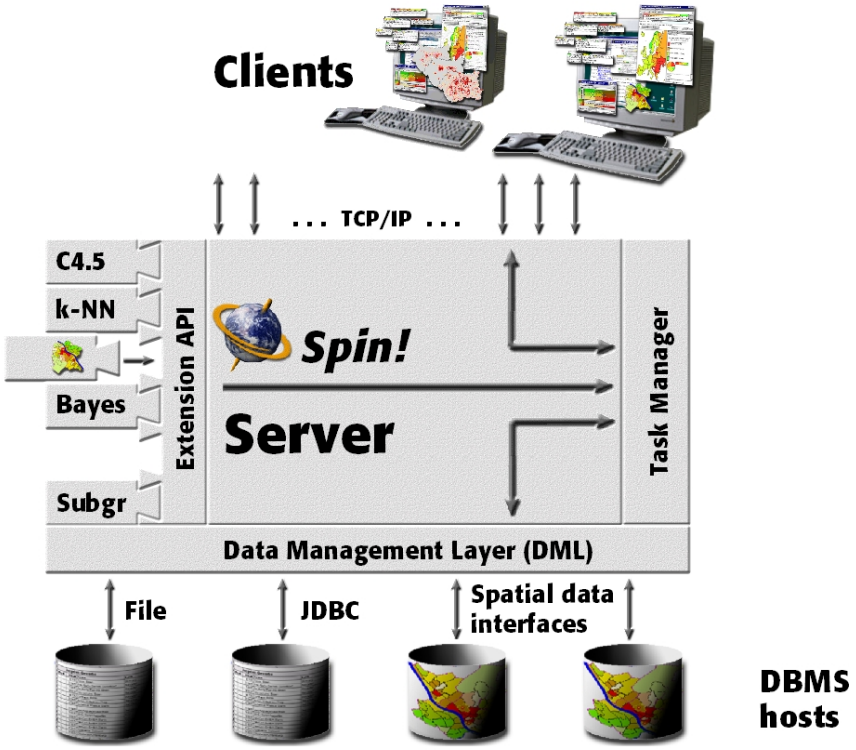


Figure 3: SPIN! architecture. *Data are stored in spatial and non-spatial databases or flat files. Analysis and visualization modules can be plugged into the system by*

State of integration

In the ESPRIT project *CommonGIS*, Descartes and Lava/Magma have already been integrated in a demonstrator (see contribution to this conference). There also exists a version of Kepler that is integrated with Descartes (Andrienko et al. 1999a) and first experiments with GAM running inside Kepler are underway. The challenge is now to combine this vast amount of functionality in a coherent system (fig 3). This will lead to a redesign of significant parts of the currently existing systems. Data are stored in spatial and non-spatial databases or flat files. Analysis and visualization modules can be plugged into the system by implementing an extension API. Clients can access the server over the internet. The technology used for implementing the architecture will be Enterprise Java Beans.

Applications

Seismic Data

The system will be used in several applications. One application area is volcano research and hazard management. The Merapi volcano in Central Java, Indonesia, is one of the most active volcanoes in the world. It has been classified as a high-risk volcano and included in the list of 15 Decade Volcanoes. In co-operation with the Volcanological Survey of Indonesia and other

institutions in Indonesia and Germany, the GFZ initiated an interdisciplinary monitoring program in 1994.

Continuous monitoring of the volcanic activities of Merapi (such as gas emanation, seismicity, deformation) as well as repeated measurements (geomagnetism, geoelectricity and gravity) are conducted within the MERAPI-project. Geological investigations are carried out *in situ* and in laboratories. Additional information about the population, landuse, infrastructure etc. will be collected and combined with the existing data about volcanic activity. The analysis of all the different data will give a better understanding of the volcanic activities and help the local authorities react in case of an eruption. Early warning and good knowledge about the infrastructure and population in this region may save human life. It is planned to utilize the SPIN!-system for the following tasks:

- Estimation of possible future eruption;
- Building interactive hazard maps and combining them with information about land use/land cover, infrastructure and population in order to make a damage assessment;
- Dissemination of information for volcano risk mitigation over the internet.

Another objective is to adapt the generic SPIN!-system to the specialized application area of earthquake and volcano research and hazard assessment by integrating methods for natural hazard assessment that have been developed by IITP and implemented in the Geoprocessor system (Gitis et al. 1995, Gitis 1998).

Census data

A second application area is the analysis and web-based dissemination of census data from statistical offices. Test data will be taken from the UK 1991 census. Work will be done collaboratively with the UK Office for National Statistics, which currently together with MIMAS is planning the tools and services for public access to the forthcoming UK national census in 2001.

To focus the application, we chose as our topic the public debate over Unitary Development Plans (UDP) in the United Kingdom (Petch and Gibson, 2000). The district chosen for investigation was Stockport, one of the ten Metropolitan Districts of Greater Manchester, UK. The UDP has eight main areas of policy, Environment, Countryside and Open Space, Housing and Population, Economy, Shopping, Transportation, Leisure and Community Facilities and Minerals and Waste Disposal. We will focus on Housing. The main requirements for Housing relate to the three main tasks in the development process, i.e.

- Forecasting numbers of houses needed
- Allocation of land
- Development control.

Under these headings we currently work on prioritising several of data processing issues based on their importance in the whole process of housing development and on the nature of the SPIN! objectives.

In a related application, the project aims to develop a detailed concept for a web-based information brokering service with georeferenced data as a foundation for a cost-effective dissemination of data. Web-based, interactive Spatial Mining can add a tremendous value to the mere distribution of data. This added value can be the key for commercializing the distribution of data for statistical offices, public agencies, and scientific institutions.

Conclusion and future work

The architecture of the SPIN! spatial data mining system was outlined. A first prototype of this system is expected by the end of 2000. It will combine a suite of existing data mining, spatial analysis and spatial visualization approaches in a coherent design. In the next stage, novel spatial analysis methods from machine learning to Bayesian statistics will be integrated and suitable visualization tools provided.

Acknowledgement

Work on this paper was funded by the European Commission under IST-1999-10536 SPIN!. Contributions by all project partners relating to their special area of expertise in the project are gratefully acknowledged. Any remaining errors in putting together the material are the responsibility of the author.

References

Andrienko, G., Andrienko N, "Knowledge-Based Visualization to Support Spatial Data Mining", In: Hand, D.J., Kok, J.N., and Berthold, M.R., *Advances in Intelligent Data Analysis*. IDA-99, Amsterdam, Berlin, Springer, 149-160, 1999

Andrienko, G.; Andrienko, N.. "Interactive Maps for Visual Data Exploration", *International Journal of Geographical Information Science* 13(5), 355-374, 1999a

Diaconis, P., "Theories of Data Analysis: From Magical Thinking Through Classical Statistics", in: Hoaglin, D. C., Mosteller, F., Tukey, J.W., *Exploring Data Tables, Trends, and Shapes*, New York, Wiley, 1985

Dykes, J., "Exploring spatial data exploration with dynamic graphics", *Computers and Geosciences*, 23, 345-370, 1997

Esposito, F., Lanza, A., Malerba, D. and Semeraro, G., "Information capture from topographic maps using machine learning", *Proc. of the Joint Workshop of the Italian Association for Artificial Intelligence (AI*IA) and the International Association for Pattern Recognition*, 122-127, 1998

Ester, M., Frommelt, A., Kriegel, H.P, Sander, J., "Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support", in *Data Minining and Knowledge Discovery, an International Journal*, 1999

Fayyad, U. Piatetsky-Shapiro, G., Uthurusamy, R. (eds.), *Advances in Knowledge Discovery and Data Mining*. Menlo Park, AAAI Press, 1996

Gitis V., "GIS technology for the design of computer-based models in seismic hazard assessment.- Geographical Information Systems", in: A.Carrara and F.Guzzetti (eds), *Assessing Natural Hazards*, Kluver Academic Publishers, 219-233, 1995

Gitis V., Dovgyallo A., Osher B., Gergely T., "GeoNet: an information technology for WWW on-line intelligent Geodata analysis", *Abstracts of 4th EC-GIS Workshop*, Hungary, 1998

Haining, R. *Spatial data analysis in the social and environmental sciences*, Cambridge University Press, 1991

Klösgen, W., "Deviation and association patterns for subgroup mining in temporal, spatial, and textual data bases", In: Polkowski, L., Skowron, A. (eds): *Rough sets and current trends in computing*, 1-18, New York, Springer, 1998

Klösgen, W., Zytkow, J. (eds.), *Handbook of Data Mining and Knowledge Discovery. Oxford University Press*, to appear 2000

Koperski, K., Han, J. "GeoMiner: A System Prototype for Spatial Mining", Proceedings ACM-SIGMOD, Arizona, 1997

MacEachren, A.M.; Wachowicz, M.; Edsall, R.; Haug, D.; Masters, R., "Constructing knowledge from multivariate spatiotemporal data: integrating geographical visualization with knowledge discovery in database methods", *International Journal of Geographical Information Science* 13(5): 311-334, 1999

Malerba D., Esposito F., and Lisi, F.A., "Learning recursive theories with ATRE", in H. Prade (Ed.), *Proceedings of the 13th European Conference on Artificial Intelligence*, 435-439, John Wiley & Sons, Chichester, England, 1998

May, M., "Applying Data Mining and advanced GIS to the Analysis of Vegetation Data", Windhorst, W., Enckell, P. (ed.), *Conf. proc. Sustainable Landuse Management. The Challenge of Ecosystem Protection*, European Ecological Federation & Ecology Center, University Kiel, Salzau 1999

Mitchell, T., *Machine Learning*. New York, McGraw-Hill, 1997

Openshaw, S., "Building automated Geographical Analysis and Exploration Machines", in Longley, P. A., Brooks, S. M. and Mcdonnell, B. (eds.) *Geocomputation: A primer* Macmillan Wiley Chichester, 95-115, 1998

Openshaw, S., Turner, A., Turton, I., Macgill, J. and Brunsdon, C., "Testing space-time and more complex hyperspace geographical analysis tools" in Martin, D. (ed.) *Innovations in GIS and GeoComputation 7,* Taylor and Francis, London, 2000

Openshaw, S., Turton, I., Macgill, J. and Davy, J., "Putting the Geographical Analysis Machine on the Internet", in Gittings, B. (ed.) *Innovations in GIS 6,* Taylor and Francis, London, 1999

Petch, J., Gibson, C., *User Requirements Report on Unitary Development Plan Application,* SPIN!-project, technical report, Manchester Metropolitan University, 2000

Paaß, G., Kindermann, J.: "Bayesian Classification Trees with Overlapping Leaves Applied to Credit-Scoring", In: X. Wu , R. Kotagiri, K.B. Korb (eds.): *Research and Development in Knowledge Discovery and Data Mining*, Berlin, Springer, 234 – 245,1998

van den Berg, C., F.Tuijnman, T.Vijbrief, C.Meijer, P. van Oosterom, and H. Uitermark "Multi-server Internet GIS: Standardization and Practical Experiences", in Goodchild, M., Egenhofer, M., Fegeas, R., and Kottman, C. (eds.) *Interoperating Geographic Information Systems*. Boston: Kluwer Academic Publishers, 365-377, 1999

Wrobel, S. "Scalability Issues in Inductive Logic Programming Data", In *Proc. 9th Int. Workshop on Algorithmic Learning Theory* (ALT-98), Berlin, Springer, 1998

Wrobel, S.; Wettschereck, D.; Sommer, E.; Emde, W., "Extensibility in Data Mining Systems", *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, AAAI Press, 1996

MediBook: Realisierung eines generischen Ansatzes für ein internetbasiertes Multimedia-Lernsystem am Beispiel Medizin

Cornelia Seeberg[1,2], Ivica Rimac[1], Stefan Hörmann[1], Andreas Faatz[1], Achim Steinacker[1], Abdulmotaleb El Saddik[1] und Ralf Steinmetz[1,2]

1

Industrielle Prozess- und Systemkommunikation (KOM)
Fachbereich Elektrotechnik und Informationstechnik
Technische Universität Darmstadt
Merckstr. 25 • 64283 Darmstadt

2

GMD IPSI
Forschungstentrum Informationstechnik
Institut für Integrierte Publikations- und Informationssysteme
Dolivostr. 15 • 64293 Darmstadt

{seeberg, rimac, hoermann, faatz, steinacker, el-saddik, steinmetz}@kom.tu-darmstadt.de

1. Einleitung

MediBook ist ein Kooperationsprojekt der medizinischen Fakultät der Justus-Liebig-Universität Gießen (Erstellung der Inhalte) und des Lehrstuhls KOM der Technischen Universität Darmstadt (Entwicklung der technischen Plattform). Es wird vom Hessischen Ministerium für Wissenschaft und Kunst gefördert, um die traditionelle Lehre in den ersten Semestern des Medizin-Studiums um ein flexibles, zeit- und ortsunabhängiges System zum Selbstlernen zu erweitern.

Es ist ein Werkzeug zum Speichern, Verwalten und vor allen Dingen Auffinden und Kombinieren von Lernressourcen. Es bietet Hilfsmittel, um einerseits bestehende Ressourcen zu beschreiben und in einen Zusammenhang zu stellen und andererseits einzelne Ressourcen zu einer Einheit, einem Kurs zu verbinden.

MediBook ist somit eine medizinische Wissensbasis mit einem effizienten Zugriff und Werkzeugen, um aus einzelnen, unzusammenhängenden Informationseinheiten einen kohärenten Kurs zu erzeugen.

Wissensbasis

Der Wissensbasis liegt eine formale Darstellung des Gebietes der Medizin zu Grunde (siehe Abschnitt 4.). Diese formale Darstellung enthält die "Grundwahrheiten" der Medizin: Die wichtigen Begriffe (*Concepts*) - z.B. Niere, Aspirin, Bakterie - sind durch semantische Relationen miteinander verbunden - z.B. Dickdarm ist-Teil Verdauungssystem. In MediBook heißt diese formale Wissensrepräsentation *ConceptSpace*.

Den Modulen - in MediBook Medienbausteine genannt - sind Begriffen zugeordnet. Jeder einzelne ist durch Metadaten beschrieben. Unter anderem werden hier die physikalische Größe der Ressource, die Rechte, das Erstellungsdatum, aber auch pädagogische Eigenschaften gespeichert. Wir verwenden als Metadaten-Schema LOM (*Learning Object Metadata*). Dieser IEEE-Vorschlag ist ein weitverbreiteter Entwurf zu einem internationaler Standard, um Lernressourcen zu beschreiben (siehe Abschnitt 3.). Auf diese Weise können auch andere Systeme auf die MediBook-Ressourcen zugreifen. Damit wird eine Wiederverwendung möglich.

Zusätzlich werden die Medienbausteine miteinander durch rhetorisch-didaktische Relationen verbunden, so dass ein Zusammenhang zwischen ihnen (z.B. Medienbaustein A erklärt Medienbaustein B) hergestellt werden kann.

MediBook ist offen für Ressourcen unterschiedlichsten Formats. Die einzelnen Informationseinheiten können Text, Bilder, Video-Filme, Audio-Dateien oder Animationen sein. Sie können Informationen, Fallbeispiele, Thesen, Motivationen oder Aufgaben enthalten. Schon bestehende Ressourcen sollen eingebunden werden, damit eine effiziente Wiederverwendung der oft sehr aufwändig erstellten Multimedia-Elemente möglich wird.

Der Ansatz, die Medienbausteine auf den unterschiedlichen Ebenen (LOM, rhetorisch-didaktische Relationen und die Zuordnung zu einer formalen Wissensrepräsentation) zu beschreiben, ist nicht auf die Medizin beschränkt. Für ein anderes Wissensgebiet müssen die Begriffe des *ConceptSpace* definiert und gegebenenfalls die Menge der semantischen Relationen erweitert werden.

Szenario

Es gibt drei Rollen, die von MediBook unterstützt werden:

- **Autor:** Der Autor ist ein Mediziner, der in der zu lehrenden Domäne ein erfahrener Experte ist. Die Aufgabe des Autor ist es, die Wissensbasis zu erstellen. Diese Aufgabe besteht aus drei Teilen: Generierung des *ConceptSpace*, Einbinden der eigentlichen Inhalte (Medienbausteine im *MediaBrickSpace*) und Verbinden der Medienbausteine mit den entsprechenden Begriffen.
- **Lehrende:** Der Lehrende, auch ein Mediziner, trifft eine Auswahl aus allen Medienbausteinen für Studierende und bestimmt deren Reihenfolge und die Gliederungsebenen. Damit er sich in der Wissensbasis zurechtfindet, muss das System ihn in geeigneter Weise unterstützen. Durch die im Benutzerprofil gespeicherten Daten über den Lehrenden kann das System adaptiv den relevanten Ausschnitt der Wissensbasis anzeigen.
  Er hat zusätzlich die Möglichkeit, Übergangsseiten zwischen den einzelnen Medienbausteinen zu generieren.
- **Lernender:** Der Lernende hat entweder die Möglichkeit, einem Vorschlag eines der Lehrenden zu folgen oder kann selbst auf der Wissensbasis navigieren, wobei er die gleichen Hilfsmittel wie der Lehrende benutzen kann. Bei individuellen Benutzerprofilen ist eine adaptive Anpassung der Wissensbasis möglich. Der Lernende hat aber nicht die Möglichkeit, einen Kurs für andere zu erstellen.

Ausblick

Zur Zeit sind als Lernende Medizin-Studierende vorgesehen. Durch inhaltliche Erweiterungen der Wissensbasis (eine Systemerweiterung ist nicht notwendig) kann MediBook auch als Weiterbildungssystem für Ärzte oder als Informationssystem beispielsweise für Patienten und ihre Angehörigen verwendet werden.

Die Möglichkeiten für die Lernenden, Annotationen vorzunehmen, *Bookmarks* zu definieren oder interaktive Tests zu bearbeiten, können in weiteren Schritten geboten werden.

Es ist erstrebenswert, insbesondere, wenn das MediBook von einem heterogenen Lernerkreis benutzt wird, dass das System selbst die Generierung der Lektionen und Kurse vornimmt. Dazu ist ein erweitertes Benutzerprofil und ein regelbasiertes Systemmodul notwendig, das die Einträge des Benutzerprofils mit den Beschreibungen der Medienbausteine vergleicht, um die relevanten Bausteine zu finden und als einen Kurs mit Inhaltsverzeichnis zu präsentieren.

2. Architektur

Die Gesamtarchitektur von MediBook integriert den Zugriff auf Informationssysteme unterschiedlicher Ausprägung, die zudem noch an verschiedenen Standorten verwaltet werden. Dies erfordert eine *Broker* orientierte *Middleware*, die eine Zusammenführung und Integration der Metadatenbeschreibungen für die Kursmodule zusammen mit der formalen Wissensrepräsentation des Fachgebiets ermöglicht. Zusätzlich wird dadurch die verteilte Datenhaltung für die Benutzer des Systems transparent, sodass ein virtuelles Gesamtsystem entsteht. Wie man an Abbildung 1 sieht, erfolgen alle Zugriffe der *Client-Tools* auf die in MediBook verwalteten Informationen über diese Komponente. Das schließt sowohl die Zugangskontrolle und damit das Benutzerprofil, als auch alle *Retrieval*-, Erfassungs- und Kompositions-Werkzeuge ein.

Technisch wird der transparente Zugriff auf eine verteilte Wissensbasis durch eine *query*-basierte Vernetzung der einzelnen Datenbanken realisiert. Im Gegensatz zu indexbasierten Techniken, bei der die Datenbanken komplette Indizes ihrer Inhalte austauschen und die dadurch sehr performante Antwortzeiten liefern können, ermöglicht die *query*-basierte Vernetzung einen einfachen Aufbau und Erweiterung der kompletten Wissensbasis. Als mögliche Protokolle zur Kommunikation zwischen den Datenbanken kann ein auf XQL basierendes Anfrageprotokoll verwendet werden. Als Alternative hat die *Open Archives Group* ein einfaches offenes Anfrageprotokoll spezifiziert, mit dem beliebige Datenbanken abgefragt werden können. Dieses Protokoll verwendet allerdings zur Suche nur einen sehr eingeschränkten Satz von Metadaten. In MediBook wird zur Beschreibung und Verwaltung der einzelnen Kursmodule und der vollständigen Kurse der leicht erweiterte *Learning Object Metadata* (LOM) 4.1-Entwurf der IEEE verwendet. Entsprechend wurde für MediBook ein eigenes, auf LOM basierendes Anfrageprotokoll entwickelt. Datenbanken, die LOM-Daten speichern und verwalten und in die MediBook-Wissensbasis integriert werden sollen, senden eine Beschreibung der von ihnen verwalteten Daten an einen zentralen *Content Location Service*. Dieser *Service* wird von allen *Retrieval*-Diensten vor der Anfrage an die Wissensbasis kontaktiert. Der *Content Location Service* wählt anhand der Beschreibungen geeignete Datenbanken für die Anfragen aus und führt sie durch. Die einzelnen Anfragen werden von diesem Dienst gesammelt und

aufbereitet und dann an die aufrufende Komponente übermittelt. Als Datenbanken werden bei MediBook sowohl relationale Datenbanken wie Oracle 8 und DB2 von IBM, als auch die XML-basierte Datenbank Tamino der Software AG verwendet. Entsprechend muss das Anfrageprotokoll auf alle Datenbanksprachen umsetzbar sein. Zur Kodierung der Anfragen bzw. den Ergebnissen verwendet MediBook eine LOM 4.1-konforme DTD. Den *Client-Tools*, die in den folgenden Abschnitten beschrieben sind, liegt hingegen ein objektbasiertes Modell der LOM-Daten zugrunde. Das erfordert eine beliebige Umsetzung der Metadaten von bzw. nach einer XML-Repräsentation, einem Objektmodell und einem relationalen Datenschema, die von der *Middleware* realisiert wird und in Abbildung 2 skizziert ist.

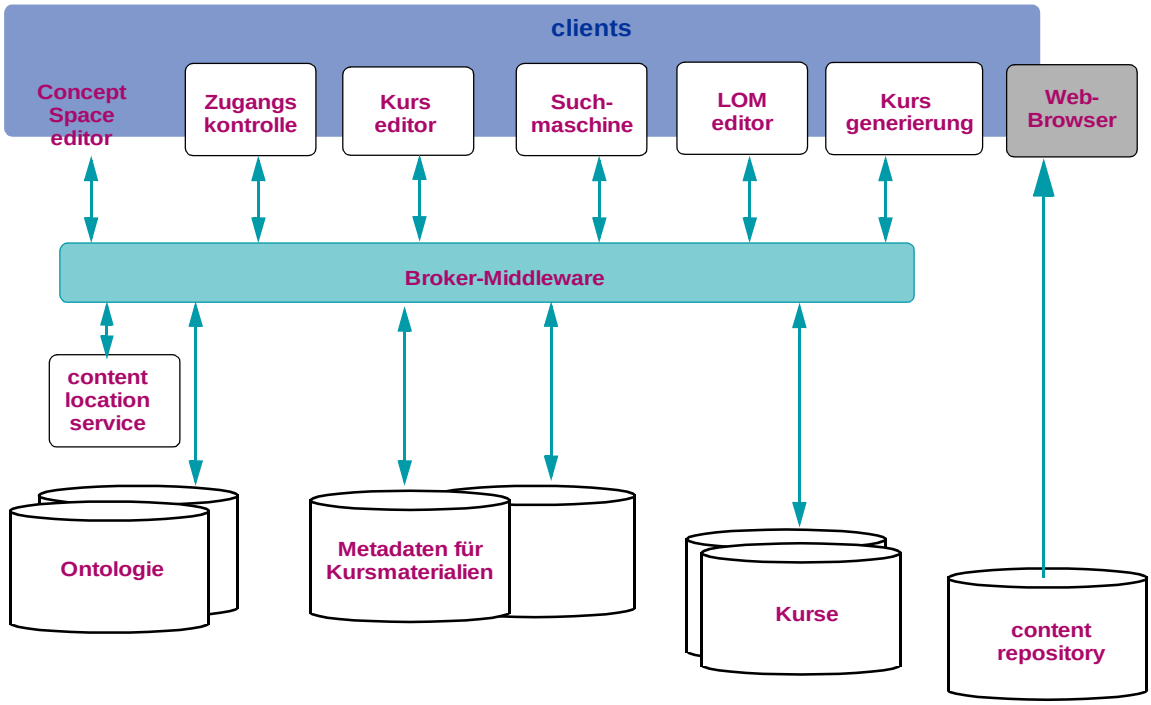


Abbildung 1: MediBook-Architektur

Zusätzlich zur Integration der verschiedenen Datenbanken mus die MediBook-*Middleware* auch die Verbindung zwischen den Metadaten der Medienbausteine und der formalen Repräsentation der Wissensbasis realisieren. Auf dem Gebiet des netzbasierten Wissensaustausch wird derzeit der *Ontology Inference Layer* (OIL) entwikkelt, der einen standardisierten Zugriff auf Ontologien ermöglichen soll. Da hier allerdings derzeit nur erste Entwürfe existieren und noch keine Implementierung verfügbar ist, verwenden wir bei MediBook ein proprietäres Verfahren zur Verbindung der einzelnen *Concepts* der Ontologie mit den Beschreibungen der Medienbausteine. Eine Integration mit dem ISO-Standard *Topic Maps* ist derzeit nicht vorgesehen.

Die Notwendigkeit, einzelne Ressourcen mit Metdaten zu beschreiben, um in der immer unüberschaubareren Informationsdichte effektiv und schnell gewünschte Informationen zu lokalisieren und zu nutzen, hat zu der Entwicklung von Metdaten-Standards für allgemeine Ressourcen wie *Dublin Core*, für Lernressourcen wie der von uns verwendete LOM-Entwurf und zu Metadatenbeschreibungen für multimediale Elemente wie MPEG7, geführt. Auch die Idee der Integration von Metadaten-*Repositories* mit formalen Beschreibungen einer Wissensdomäne findet zunehmend Verbreitung.

Ein kritischer Punkt, der die Akzeptanz der auf diesen Standards aufbauenden Systemen maßgeblich beeinflusst, ist, wie der Benutzer mit der Fülle an Datenmaterial konfrontiert wird. Obwohl eine große Zahl an Metadaten automatisch berechnet und generiert werden kann, ist die Erfassung der benötigten Beschreibungen ein signifikanter Aufwand für den Benutzer. Während bei LOM das Problem in der nötigen Sorgfalt und im Zeitaufwand des Benutzers liegt, stehen für Ontologien bisher praktisch keine Oberflächen zur Verfügung, die es einem Wissensexperten ohne technisches *Know How* er-

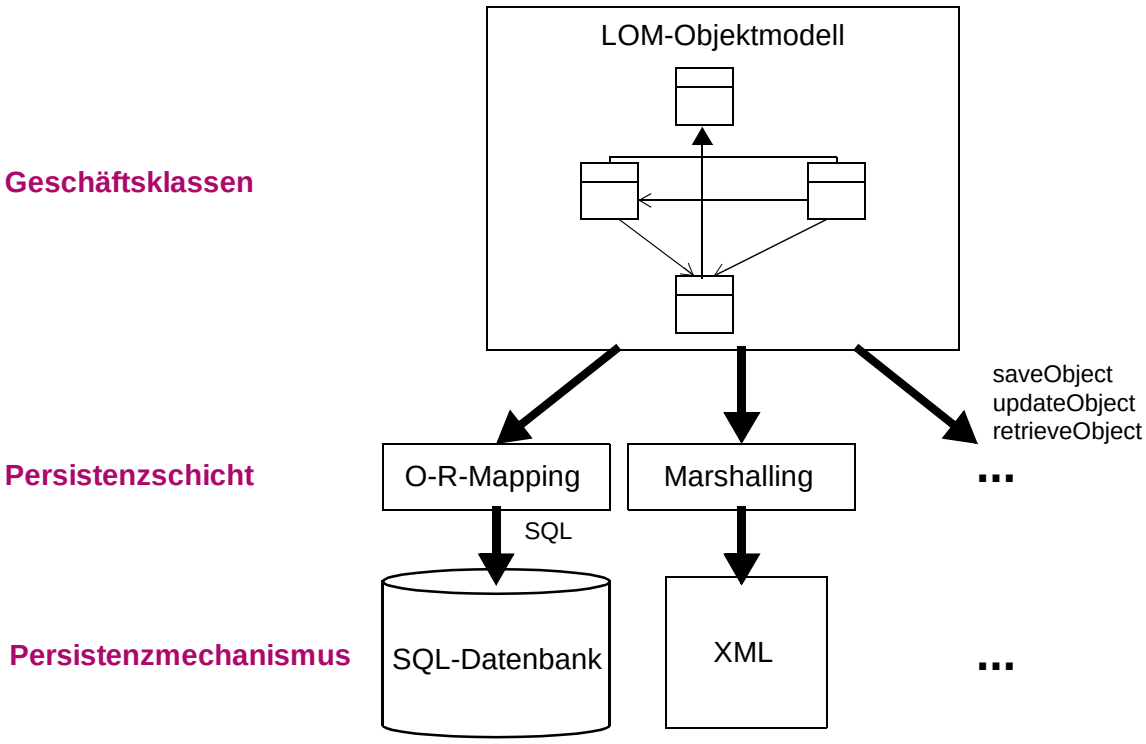


Abbildung 2: Offene Architektur: Mapping von den Geschäftsklassen auf die diversen Persistenzmechanismen

möglichen, eine Ontologie seines Wissensbereiches aufzubauen.

Anhand von drei ausgewählten Werkzeugen - *ConceptSpace-Editor*, *LOM-Editor* und Kursgenerator -, die in MediBook realisiert sind, wird im Folgenden gezeigt wie Benutzern angepasste Sichten auf das Material bereitgestellt werden, die es ermöglichen Informationen über Ressourcen einfach und schnell zu erfassen, auf ihnen zu suchen, sie untereinander und mit *Concepts* aus der Ontologie zu verbinden und sie zu adaptiven Präsentationen zusammenzustellen, die dann *online* verfügbar sind.

3. LOM-Editor

Um das gezielte Suchen und Finden von Informationen (Lernressourcen) in einer einfachen Art und Weise zu ermöglichen, wurde der *LOM-Editor* entwickelt. Dieser Metadaten-Editor basiert auf dem IEEE-LOM-Schema 4.1. LOM schlägt neun Kategorien für die Beschreibung einer Lernressource vor, die im Folgenden beschrieben werden:

- *General*:
  Die Kategorie *General* stellt die grundlegenden Informationen zur Verfügung, die dazu beitragen, eine Ressource in ihrer Gesamtheit zu beschreiben. Zu dieser Informationen gehören unter anderem Titel, Sprache, Struktur.
- *Life Cycle*
  LifeCycle beschreibt die Historie und den momentanen Stand einer Ressource und nimmt Bezug auf Personen und Gruppen, die an der Entwicklung teilgenommen haben. Beispiele sind Status, Version.
- *Meta MetaData*
  In der Kategorie *MetaMetaData* werden spezielle Informationen über den Metadatensatz an sich aufgenommen. Die Datenelemente beziehen sich auf die Vorgehensweise bei der Erstellung und auf die dabei beteiligten Personen, beinhalten damit also keine Informationen über die Ressource an sich.
- *Technical*
  Die Kategorie *Technical* fasst die technischen Anforderungen und Eigenschaften der Ressource zusammen. Beispiele sind Format und Anforderungen an das Betriebsystem oder an den *Browser*.

- *Educational*
  Diese Kategorie beschreibt die didaktischen und pädagogischen Eigenschaften einer Ressource. Die pädagogischen Eigenschaften einer Ressource dienen Autoren, Lehrern und Lernenden bei der zielgerichteten Zusammenstellung von Kursen. Beispiele sind u.a. Interaktivitätsgrad, Schwierigkeitsgrad.

mehr als eine Zielressource existiert, wird jedes Ziel durch eine eigene Instanz der Kategorie *Relation* beschrieben. Beispiele sind *partOf*, *basedOn*.

- *Annotation*
  Die Kategorie *Annotation* stellt Informationen über die Anwendung der Ressource sowie den Verfasser des Kommentars und den Erstellungszeitpunkt zur

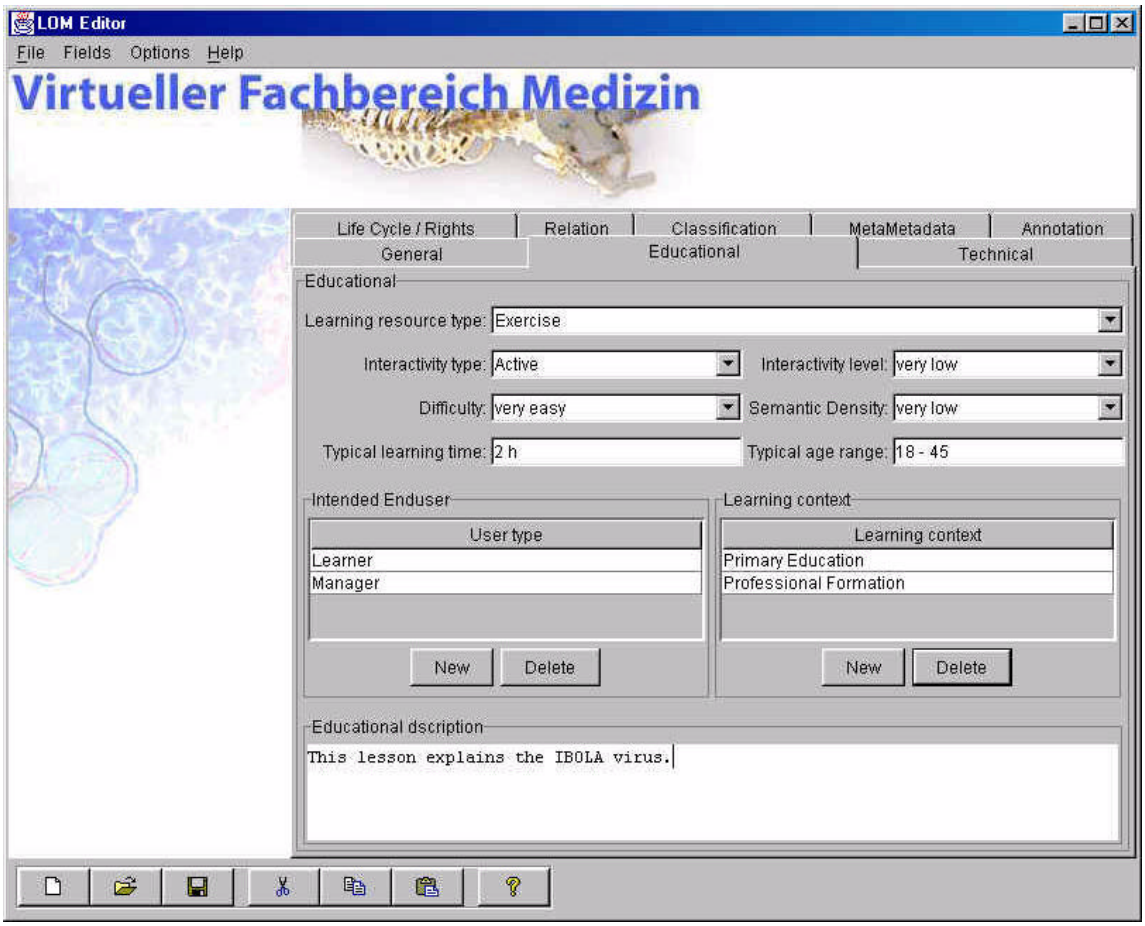


Abbildung 3: Screenshot des LOM-Editor

- *Rights*
  Die Kategorie *Rights* beschreibt die Rechte bezüglich des geistigen Eigentums einer Ressource und deren Nutzungsbedingungen. Die momentan bereitgestellten Datenfelder stellen allerdings nur eine Grundfunktionalität zur Beschreibung der Eigentumsrechte zur Verfügung.
- *Relation*
  Durch *Relation* werden die Beziehungen zwischen einzelnen, miteinander in Verbindung stehenden Ressourcen definiert. Um mehrere Beziehungen zwischen Ressourcen zu definieren, können mehrere Instanzen dieser Kategorie genutzt werden. Falls

Verfügung. Falls mehrere Anmerkungen zu einer Ressource benötigt werden, besteht die Möglichkeit, verschiedene Instanzen dieser Kategorie zu verwenden.

- *Classification*
  In der Kategorie *Classification* findet eine Einordnung der Ressource in bestimmte Klassifizierungssysteme statt. Das Wissen, das zur Klassifizierung einer Ressource notwendig ist, übersteigt den Informationsgehalt des Datenmaterials um ein Vielfaches. Die Eingliederung in eine hierarchische Struktur setzt nicht nur Kenntnisse über das zu charakterisierende Objekt voraus, sondern über alle Ressourcen eines

Klassifizierungsraums. Nur aus diesem Wissen lässt sich eine Relation zwischen Objekten definieren und eine Einordnung in ein bestimmtes System vornehmen:

Der entwickelte *LOM-Editor* (siehe Abbildung 3) kann verwendet werden, um einen Metadaten-Satz einer Lernressource zu generieren und an die *Middleware* zu übergeben, die für eine dauerhafte Speicherung zuständig ist. Er kann auch verwendet werden, um die Beschreibung vorhandenen Ressourcen zu ändern.

Die Vorteile, die sich durch die Nutzung von Metadaten bei der Suche und Wiederverwendung von Lernressourcen ergeben, hängen stark von der Qualität und dem Umfang der Metainformationen ab. Ein vollständig bestimmter Metadatensatz liefert im Allgemeinen eine bessere Grundlage für die konsistente und eindeutige Beschreibung einer Lernressource. Geht man vom Einsatz des LOM *Base Scheme* mit insgesamt 58 Datenelementen aus, so ist eine manuelle Beschreibung der Ressource wenig effizient. Der Realisierungsaufwand für einen programmgestützten Generierungsmechanismus wird sich nach kurzer Zeit amortisiert haben.

Während des Einsatzes des *LOM-Editors* bei der Beschreibung verschiedenen Lernressourcen wurde festgestellt, dass im Allgemeinen einige Basis-Metadaten-Elemente, wie z.B. *author, rights of the lesson*, oder *targeted user group* unverändert bleiben. Aus dieser Beobachtung wurde festgestellt, dass der Einsatz von Schablonen (*Templates*) das Generieren und Abspeichern von Metadaten vereinfachen wird. Jedem Autor wird hierbei die Möglichkeit gegeben, sich eigenständig Vorlagen zu definieren, die eine bestimmte Menge von Datenelementen beschreiben. Eine Möglichkeit wäre die Vorgabe von fest definierten Schablonen, bei denen der Autor die Möglichkeit hat, den einzelnen Datenelementen Werte zuzuweisen und die entsprechenden Eingaben zur weiteren Verwendung zu speichern. Denkbar ist beispielsweise die Zusammenfassung der gesamten Merkmalsgruppe "*Contribute*" zu einer Schablone, in der die einzelnen Personen eines Projektes mit Namen und Tätigkeit gemeinsam erfasst werden. Alle innerhalb dieses Projektes erstellten Ressourcen lassen sich anschließend über diese Schablone mit relativ geringem Zeitaufwand eindeutig beschreiben.

4. Der ConceptSpace-Editor

Der folgende Abschnitt stellt dar, wie das Werkzeug zur Wissensrepräsentation im MediBook, der *ConceptSpace-Editor*, aufgebaut ist. Dabei gehen wir hauptsächlich auf die logischen Schichten des *ConceptSpace-Editors* ein, die innerhalb des *Smalltalk Frame Kit* (SFK) implementiert wurden.

Das MediBook verwendet als formale Wissensrepräsentation eine Ontologie. Eine Ontologie ist eine formale Konzeptualisierung eines Wissensbereiches.

Die Modellierung spezieller Wissensbereiche erfolgt nach unserem Verständnis zielgerichtet, was sich auf den Aufbau und die zulässigen Strukturen der Ontologie auswirkt. Im MediBook erfüllt die Ontologie die Funktion einer netzförmigen Navigationsstruktur für medizinische Fachbegriffe und wird zur Anordnung von Medienbausteinen herangezogen.

Ein logischer und konsistenter Aufbau der Ontologie erfordert neben den *Concept*typen, die das Wissensgebiet charakterisieren, und den Relationstypen, die die *Concepts* miteinander verbinden, die Definition von Axiomen. Axiome dienen in unserem Fall zur ständigen logischen Überwachung der Wissensmodellierung. Zwei Beispiele für den Einsatz der Axiome sind das automatische Erstellen von Umkehrrelationen zwischen *Concepts* und die Einhaltung hierarchischer Beziehungen. Das bedeutet, dass man Vorgaben formalisieren muss, die beispielsweise für eine Aussage wie "Kolibakterien verursachen Durchfall" gleichzeitig eine Aussage "Durchfall wird durch Kolibakterien erzeugt" generieren. Im Falle der hierarchischen Anordnung von *Concepts*, wie etwa bei der Aussage "Muskeln sind ein funktionaler Bestandteil des menschlichen Bewegungsapparates" ist hingegen zu verhindern, dass eine nachträgliche fehlerhafte Aussage wie "der menschliche Bewegungsapparat ist ein funktionaler Bestandteil der Muskeln" angelegt werden kann.

Eine Programmiersprache, die die Vorgabe eines zweckdienlichen Schemas für die Erstellung einer Ontologie ermöglicht, ist das *Smalltalk Frame Kit*, abgekürzt SFK. Ein solches Schema besteht aus *Concept*typen, Relationstypen und Axiomen und wird erst bei der eigentlichen Erstellung des MediBook-*ConceptSpace* durch den Mediziner mit Fachbegriffen gefüllt. Unser Ansatz sieht vor, die Arbeit des Mediziners wesentlich zu erleichtern und seinen Fokus auf die eigentliche Anordnung der Fachbegriffe zu lenken. Er soll lediglich die Inhalte der Klassifikation und Verknüpfung medizinischer Fachbegriffe durch Relationen liefern, ohne sich um formale Logik, Axiomatik oder gar Implementierungstechniken kümmern zu müssen.

Als Resultat dieser Anforderung nehmen wir eine Arbeitsteilung bei der Ontologie-Erstellung vor: einerseits erfolgt eine Schemaerstellung im SFK durch einen Software-Entwickler, andererseits wird dem Mediziner mit dem *ConceptSpace-Editor* eine graphische und intuitive Möglichkeit zur Schemafüllung geboten. Das SFK unterstützt und überwacht dabei die Schemafüllung gemäß der Axiome, ohne dass der Mediziner sich mit der Implementierung des SFK-Schemas auseinandersetzen muss. Zwischen dem SFK-Software-Entwickler und dem Mediziner muss naturgemäß eine Absprache darüber herrschen, was in der Ontologie abgebildet werden soll. Im MediBook wurden folgende *Concept*typen im SFK-Schema modelliert: KrankheitOderSymptom, Zelle, Stoff, MediBook-*Concept*. Die Anzahl der *Concept*typen wurde für die erste Version bewusst gering gehalten, da aufgrund des stark objektorientierten Ansatzes des SFK eine Erweiterung um weitere *Concept*typen und die diesbezügliche Aktualisierung der Schema-Instanziierung durch den Mediziner leichter von statten gehen kann.

Die gesamte Arbeitsweise des SFK basiert auf dem Prinzip der *Frame*-Klassen. Das sind semantische Einheiten, die mit festgelegten Attributen (*Slots*) ausgestattet sind und instanziiert werden. Die *Slots* können mit Restriktionen an ihren Wertebereich und die Anzahl ihrer Werte ausgestattet werden und fungieren gleichzeitig als Träger der Axiome des Schemas. Für unsere beiden Beispiele der inversen Relationen und der hierachischen Anordnung stehen im SFK folgende Methoden zur Verfügung: *#inverseSlot*, die den Namen eines inversen *Slots* übergibt, und *#relationalProperties*, die für die obige Relation “ist funktionaler Bestandteil von” die Eigenschaften azyklisch und transitiv übergibt. Gerade für die automatische Berechnung von Frame-Klassen und *Slots* existieren im SFK reichhaltige Inferenzmechanismen, die Berechnungen transitiver Hüllen beispielsweise sind bereits vollständig vorgegeben. Somit ist nutzt jeder Aufbau eines SFK-Schemas bereits vorhandenen Algorithmen, die das zur Einhaltung der für das Schema relevanten Axiome notwendige Neuanlegen, Löschen und Verändern von *Frame*-Klassen vornehmen.

Ein ontologisches Schema im SFK ist somit eine Hierarchie von *Frame*-Klassen, deren Bezug zueinander festgelegt werden kann, indem man gleichzeitig das SFK als Programmiersprache benutzt. Die modellierte Hierarchie wird vom SFK selbst erst dann fest installiert, wenn das komplette Schema angelegt wurde. Inkonsistenzen wie beispielsweise mehrfaches Anlegen von gleichnamigen *Slots* in einem *Frame* werden an diesem Punkt bereits abgefangen. Jede Erweiterung des Schemas, die *Frames* und *Slots* hinzufügt (vorausgesetzt, die Vererbungsstruktur, die das Schema ausmacht, wird nicht verändert) ist jederzeit durchführbar.

Das SFK exportiert fortlaufend den aktuellen Stand einer Schema-Instanziierung per XML an ein Java-Werkzeug zur Visualisierung von Graphen. Umgekehrt wird die Schema-Instanziierung durch den Mediziner - das Anlegen von benannten Knoten (*Concepts*) und aus der Menge der Kantentypen selektierten Kanten (Relationen) im Graphen - per XML an das SFK zurückgegeben. Ist eine Erweiterung des *ConceptSpace* durch ein neues *Concept* und/oder das Ziehen einer neuen Relation unzulässig, wie etwa das Anlegen kreisförmiger Beziehungen in Hierarchien, so wird diese Erweiterung nicht zugelassen.

Zulässige Operationen des Mediziners werden visualisiert, wobei der Graph durch einen auf dem Prinzip der Federkräfte basierenden Algorithmus so übersichtlich wie möglich dargestellt wird. Das SFK, die XML-Schicht und der Java-*Client* ergeben zusammengenommen den *ConceptSpace-Editor*.

Adaptive Navigationssteuerung

Navigationssteuerungen für allgemeine netzförmige Graphen, mit denen der *ConceptSpace* dargestellt werden kann, sind derzeit kaum verbreitet. Bisher sind nur spezielle Anwendungen oder Systeme für spezielle Netztopologien in kommerziellen Anwendungen zu finden. Insbesondere ist hier die Baumstruktur zu nennen. In dieser hat jedes Element eine Verknüpfung zu seinem übergeordneten Element, ausgenommen das Wurzel-Element. Beliebig viele Verknüpfungen können zu möglicherweise untergeordneten Elementen vorliegen. Benachbarte Elemente können jedoch keine Beziehung untereinander haben. Die Baumstruktur vereinfacht in vielen Fällen die tatsächlich vorliegende relationale Struktur der Objektbeziehungen, da eine Ontologie in der Regel eine vernetzte Struktur aufweist und die Baumstruktur diese als grobe Vereinfachung darstellt. Mit dem Zwang einer Einordnung für jedes Element wird der Informationsraum in disjunkte Klassen zerlegt. Mit dieser Klassifikationsstruktur werden bestimmte Objektverknüpfungen verdeckt oder redundante Zusatzinformationen in die Baumstruktur eingefügt. Dazu kommt, dass auch bei großen Baumstrukturen der Überblick verloren gehen kann

Betrachtet man die existierenden Ansätze zur Visualisierung von allgemeinen Netzen, so ist festzustellen, dass eine interaktive Navigation i.d.R. nicht vorgesehen. ist. Zudem ändert sich bei geringen Veränderungen in der Netzstruktur das komplette Layout des Graphen. Weiter-

hin ist bei der Visualisierung einer Ontologie zu beachten, dass Überschneidungen von Knoten und Kanten und schematische Ausrichtungen nicht zu einer fehlerhaften semantischen Interpretation führen. Für den bei MediBook entwickelten Ontologie-Editor sind daher die folgenden Randbedingungen von Bedeutung:

- Navigation in allgemeinen Netzen,
- Stabilität bei geringen Veränderungen der Netzstruktur,
- Visualisierung von inhaltlicher Nähe durch räumliche Nähe.

Aufgrund der beschränkten Visualisierungsfläche auf einem Monitor und der begrenzten Aufnahmefähigkeit des menschlichen Betrachters können große, komplexe Graphen nicht in ihrer Gesamtheit dargestellt bzw. erfasst werden. Folgende drei unterschiedliche Basisvarianten zur Navigation durch einen Graphen können unterschieden werden:

- Verschiebung des sichtbaren Bereichs,
- Ausschnittvergrößerung bzw. Verkleinerung (*zoomen*),
- Öffnen und Schließen von einzelnen Knoten.

Daneben existieren Kombinationen der Basisvarianten, die weitere Möglichkeiten erschließen, zum Beispiel eine Navigation mit zwei Fenstern. In dem einen Fenster wird ein Überblick des gesamten Graphen dargestellt, mit der Möglichkeit, auf einen einzelnen Bereich zu fokussieren. Dessen Detailansicht wird in einem zweiten Fenster dargestellt. Dieses Vorgehen entspricht einer Landkarte mit einer Überblicksdarstellung und einer zusätzlichen Detailansicht in einem Stadtplan. Eine weitere Möglichkeit ist die Fischaugen-Darstellung. Dabei wird die Ausschnittsvergrößerung auf einen Bereich beschränkt, ähnlich wie bei einem Blick durch eine Lupe. Bei Lösungen mit *Scrollbars* bzw. beim *Zoomen* in einen Bereich hinein, kann schnell der Kontext verlorengeht. Wie bereits erwähnt, hat sich die Navigation in hierarchischen Baumstrukturen, wie sie im Explorer praktiziert wird, bewährt. Bei dieser Navigation werden interaktiv einzelne Knoten geöffnet bzw. geschlossen. Damit ist gleichzeitig eine Übersichts- und Detaildarstellungen des Graphen möglich. Diese ist quasi das Optimum zur Darstellung und Navigation auf einem begrenzten Raum, und daher wurde beim *ConceptSpace-Editor* dieses Verfahren auf

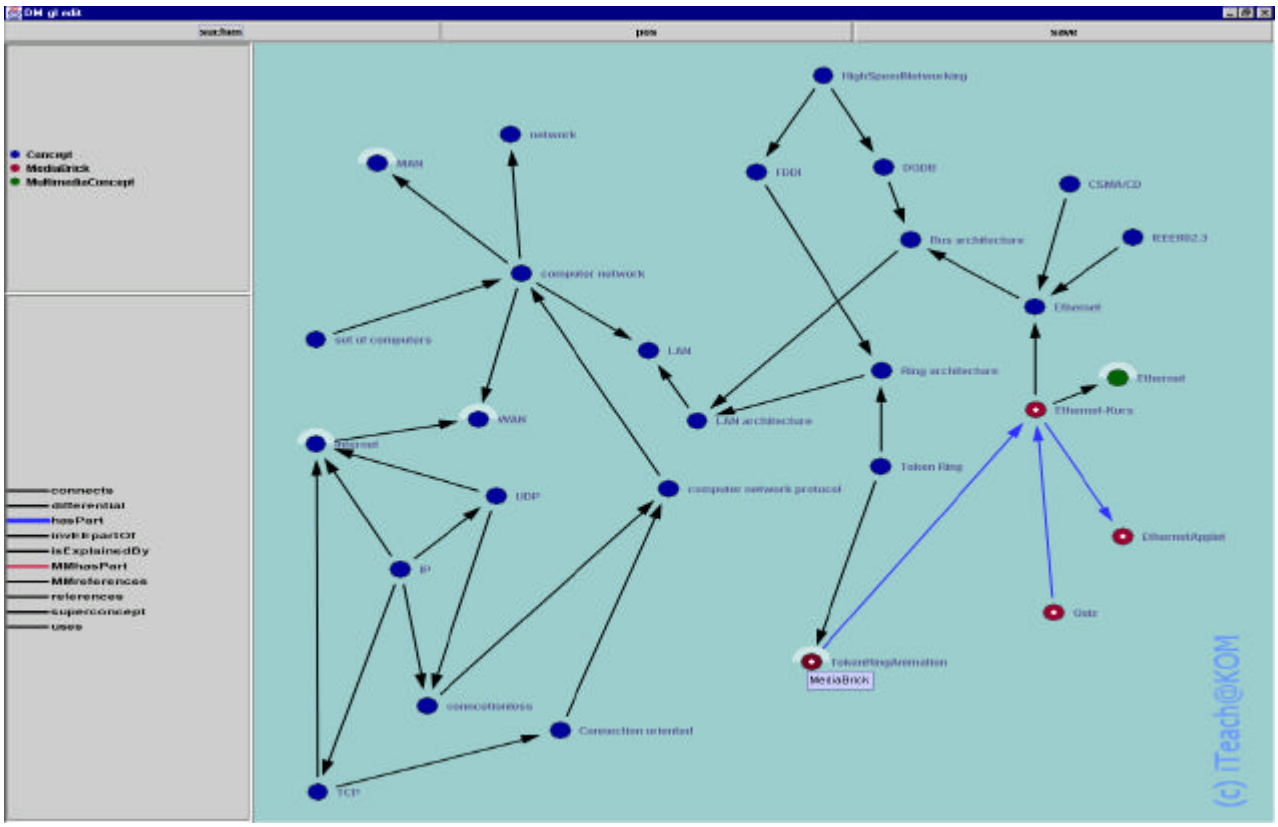


Abbildung 4: Screenshot der graphischen Oberfläche des ConceptSpace-Editors

allgemeine Graphen übertragen. Einen Screenshot, des Editors zeigt Abbildung 4.

Aus dieser Abbildung wird auch ersichtlich, wie die Verbindung zwischen einzelnen Konzepten der Ontologie und den Medienbausteinen erfolgt. Medienbausteine, die mit dem LOM-Editor erfasst und in der Wissensbasis gespeichert wurden, können auf der graphischen Oberfläche eingefügt und mit den Konzepten aus der Ontologie verbunden werden.

Das Vorgehen bei dieser Navigation entspricht einer Informationsreduktion des Graphen auf das Wesentliche. Es werden nur die Informationen angezeigt, die zur Zeit für den Benutzer zur Navigation wichtig sind. Dabei können durch das Zusammenfassen von Teil-Graphen, weitentfernte Knoten gleichzeitig dargestellt werden. Der visuelle Eindruck über die Struktur der verbliebenen Knoten und Kanten bleibt dabei im Wesentlichen erhalten.

Zukünftige Arbeiten bestehen in einer Evaluierung des derzeitigen Schemas und einer entsprechenden Erweiterung. Ein weiteres Ziel, das über das Projekt MediBook hinausgeht, ist die Vereinfachung der Schemaerstellung im SFK durch einen ebenfalls graphisch unterstützten Editor.

5. Kursgenerator

Mit dem Modul, das den Namen Kursgenerator trägt, können Präsentationen der Kurse erzeugt werden, die mit dem Kurseditor erstellt wurden (siehe Abbildung 1). Die zu präsentierenden Kurse setzen sich aus Lektionen zusammen, die eine Aufzählung mit festgelegter Reihenfolge von Medienbausteinen sind. Die Referenzierung der Medienbausteine einer Lektion wird über Relationen des Typs *HasPart* und *IsPartOf* der *Learning Object Metadata* der Kategorie *Relation* realisiert. Die baumförmige Struktur der Kurse, deren Blätter stets Medienbausteine enthalten, resultiert aus der Möglichkeit der Referenzierung von Lektionen als Teil einer Lektion, wodurch Unterlektionen definiert werden. Die auf diese Weise in Zusammenhang gebrachten Medienbausteine können komplexe Lehrangebote wie beispielsweise Vorlesungen bilden. Abbildung 5 zeigt den Teil einer leicht abstrahierten Struktur eines Kurses, aus der die logischen Zusammenhänge von Lektionen und Medienbausteinen zu entnehmen sind.

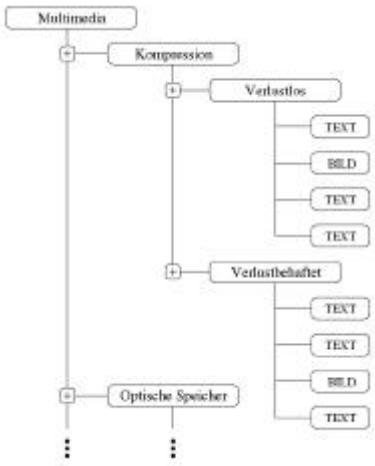


Abbildung 5: Kursstruktur

Für die automatische Erzeugung von Präsentationen dieser Kurse eignen sich in erster Linie die Dateiformate HTML und PDF. Die beiden Ausgabeformate der Kurse sind für das Durcharbeiten der Kurse am Computer beziehungsweise zum Ausdrucken der Kurse geeignet. Hieraus ergeben sich zwei grundsätzlich verschiedene Anforderungen an die Präsentationen, die bei der Erzeugung von Präsentationen zu differierenden Lösungsansätzen führen.

Der Vorteil der Präsentation in Form von HTML-Seiten liegt darin, dass sie direkt von den Lernenden am Computer betrachtet werden können, ohne dass dafür weitere Software außer einen Internet-Browser installiertwerden muss. Der jedoch wichtigste Vorteil bei der Erzeugung von HTML-Seiten liegt in der Verwendungsmöglichkeit von kontinuierlichen Medien, wie Klänge und Bildsequenzen, um die Präsentation der Lernressourcen multimedial zu gestalten. Es wird jedoch zusätzlich ein Mechanismus benötigt, mit dessen Hilfe die Medienbausteine zu kleinen Einheiten zusammengefasst werden können. Dadurch soll erstens das Durcharbeiten des Kurses am Computer vereinfacht werden und zweitens das schnelle Auffinden der relevanten Lernressourcen ermöglicht werden. Abbildung 6 zeigt ein Beispiel für die Präsentation des Kurses nach Abbildung 5 in HTML-Seiten.

Die Präsentation der Medienbausteine eines Kurses in Form einer PDF-Datei lässt sich zwar ebenfalls direkt am Computer betrachten, ist jedoch hauptsächlich für den Ausdruck auf Papier der Kurse gedacht. Aus diesem Grund wird die Präsentation der Kurse auf Medienbau-

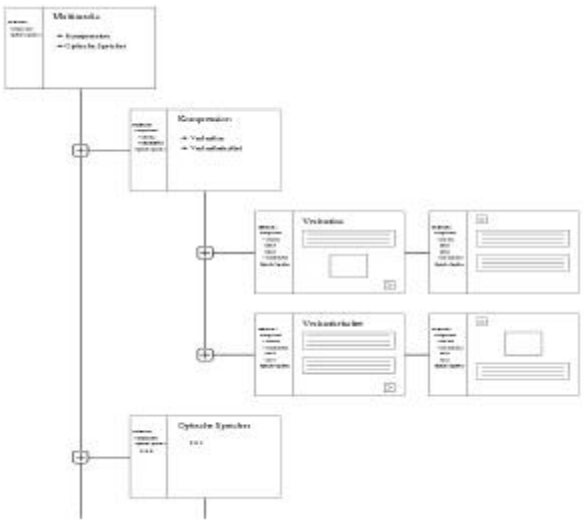


Abbildung 6: HTML-Seiten

steine mit statischem Inhalt, wie beispielsweise Texte und Bilder, beschränkt. Anders als bei der Erzeugung von HTML-Seiten werden die Medienbausteine des Kurses nicht zu kleinen Gruppen zusammengefasst, sondern in *Pre-Order* Tiefensuche des baumförmigen Kurses in der PDF-Datei hintereinander gehängt, wodurch die von Büchern gewohnte lineare Struktur der Lernressourcen entsteht (siehe Abbildung 7).

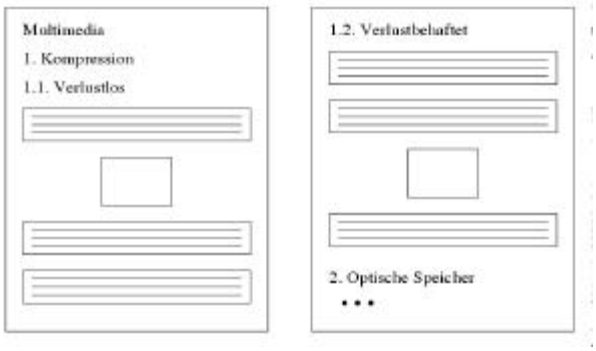


Abbildung 7: PDF-Datei

Bevor jedoch die eigentlichen Präsentationen erzeugt werden können, wird eine Beschreibung des Kurses erzeugt, die als XML-Datei exportiert werden kann. Zusammen mit dem Pfad innerhalb des Kurses zu der Lektion, von der eine Präsentation erzeugt werden soll, und dem XSL-Stylesheet, das für die Erzeugung der Präsentation verwendet werden soll, kann die Präsentation erzeugt werden. Die Generierung von Präsentationen ist unabhängig davon, ob erzeugte Dateien auf Festplatte oder CD-ROM gespeichert werden oder direkt über Internet zu den Lernenden übertragen werden. Die generierten Präsentationen können von den Lernenden mit Standardsoftware betrachtet werden. Für die PDF-Dateien kann der kostenfrei verfügbare *Viewer* der Firma Adobe verwendet werden. Die Betrachtung der HTML-Dateien erfolgt mit einem handelsüblichen Java-fähigen Internet-*Browser*. Bei der Betrachtung der HTML-Dateien werden die Lernenden durch ein Java-Applet, das im Internet-*Browser* eingebettet wird, bei der Navigation durch den Kurs unterstützt. Damit hierbei die Orientierung der Lernenden zu keinem Zeitpunkt verloren geht, wird eine Übersicht über die Struktur des Kurses angezeigt, in der die aktuell angezeigte HTML-Seite markiert ist.

Abstrakt

Werkzeuge zur interaktiven Clusteranalyse

Alexander Hinneburg
Martin-Luther-Universität Halle/Wittenberg
hinneburg@informatik.uni-halle.de

Einführung und Problemstellung

In vielen Data Mining Anwendungen wird Clusteranalyse als eine wichtige Technik zur Wissensentdeckung eingesetzt. Jedoch die Zielstellung und Anforderungen an eine solche Analyse sind sehr vielfältig und variieren in Abhängigkeit des Anwendungskontext. Demgegenüber stehen eine Vielzahl von automatischen Clusterungstechniken, die jedoch oft eine spezielle Cluster-Definition benutzen. Die Auswahl der Technik hängt somit jeweils auch von der Anwendung ab. Um den Anwender stärker in den Wissensentdeckungsprozeß mit einzubeziehen und dadurch die Ergebnis-Qualität und das Verständnis des Ergebnisses zu verbessern, erforschen wir in unserer Arbeitsgruppe Möglichkeiten automatische Verfahren mit interaktiven, visuellen Techniken zu kombinieren.

Viele Daten, die in KDD Anwendungen vorkommen, enthalten hochdimensionale Eigenschaftsvektoren, die numerische und kategorische Komponenten enthalten. Hier möchte ich mich auf Datenmengen mit Vektoren fester Länge mit numerischen Attributen beschränken ($D = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$). Bei fast allen realen Datenmengen kann im allgemeinen nicht ausgeschlossen werden, daß sie einen Anteil von Ausreißern (Rauschen) enthalten. Ausreißer sind Objekte, die sich nicht in einen Cluster einordnen lassen.

Viele Verfahren zeigen auf hochdimensionalen Daten ($d > 16$) ein quadratisches Laufzeitverhalten, was für große Datenmengen unakzeptabel ist, oder sie können keine sinnvolle Clusterstruktur in den Daten finden. Der Grund für die erste Beobachtung ist, daß viele Algorithmen [4, 10, 2, 14] nächste Nachbar- oder Bereichsanfragen nutzen und eine Laufzeit von $O(N \cdot T_{query})$ haben. Weber et. al. [13] zeigten, daß alle bekannten Index-Methoden diese Anfragen nicht in sublinearer Laufzeit für beliebig dimensionale Räume und unbekannte Daten Verteilung beantworten können. Zur zweiten Beobachtung, dem Fehlen einer sinnvollen Clusterstruktur in hochdimensionalen Daten, wurden in jüngster Zeit Arbeiten veröffentlicht, die untersuchen, ob der herkömmliche Ähnlichkeitsbegriff (z.B. euklidischer Abstand), auf dem alle Clusterdefinitionen aufbauen, bei zunehmender Dimensionalität sinnvoll bleibt [3, 6, 1]. Erste Resultate zeigten, daß dies nicht der Fall ist, weil die Datenpunkte in einem hochdimensionalen Raum sehr dünn verteilt sind. Beyer et. al zeigten, daß bei zunehmender Dimensionalität der Abstand der Datenpunkte zueinander schneller steigt als die Differenz der Distanzen zum nächsten und weitesten Nachbarn. Daraus folgt, daß ein auf Abstandsmessungen basierendes Ähnlichkeitsmaß in hochdimensionalen Räumen an Selektivität und Unterscheidungsvermögen einbüßt.

In [6] zeigten wir: hochdimensionale Feature-Vektoren beschreiben Objekte sehr genau und können somit auch Informationen über nicht relevante Eigenschaften enthalten. Wir modifizierten das Ähnlichkeitsmaß derart, daß nur die relevanten Attribute zum Messen des Abstands genutzt wurden. Mathematisch entspricht dieses Vorgehen, dem Messen des Abstands in einem projizierten Unterraum des hochdimensionalen Datenraumes. Übertragen auf Clusteralgorithmen muß ein Cluster nur einer bestimmten Projektion des Datenraumes, der Clusterdefinition genügen.

Die Frage ist nun, wie man eine solche Projektion bestimmen kann. Das Problem hat im allgemeinen einen exponentiellen Suchraum (bei der Beschränkung auf achsenparallele Projektionen). In unserer Arbeitsgruppe verfolgen wir den Ansatz eine große Anzahl von Projektionen automatisch von einen Optimierungsalgorithmus (genetischer oder Greedy Algorithmus) generieren zu lassen, diese automatisch auf Relevanz zu testen und zu filtern und dann manuell mit Hilfe von Visualisierungen vom Anwender bewerten zu lassen. Im folgenden Teil möchte ich auf den automatischen Relevanz-Test eingehen.

Ein Relevanz-Test für Projektionen

Die Projektionen die hier betrachtet werden sollen, haben die zusätzliche Eigenschaft, daß die Abstände der Punkte zueinander nach der Projektion nicht größer werden. Da bei einer Projektion Information weggelassen wird, muß eine Clusterstruktur nicht vollständig erkennbar sein. Die Projektion ist aber schon nützlich, wenn sich Teile der Custerstruktur erkennen lassen. Um eine Clusterstruktur zu finden, ist unser neues Paradigma, die Objektmenge in Untermengen zu aufzuteilen, wobei diese bezüglich eines Abstandsmaßes voneinander separiert sind. Basierend auf dem Begriff der Punktdichte (siehe [12, 11]) definieren wir einen Separators, der eine Punktmenge geometrisch (bezüglich einer Projektion) in zwei Untermengen teilt. Die Separations-Qualität ist die maximale Punktdichte auf der Grenze der beiden Mengen und die Teilungs-Qualität mißt wie balanciert der Schnitt ist. Um einen mehrdimensionalen Schnitt effizient und effektiv bestimmen zu können, haben wir eine neue Methode entwickelt, bei der mehrdimensionale Histogramme zur Ausreißerbehandlung [7], die kMeans Variante LBG-U [5] und eine verbesserte Form des konkurrierenden Hebbian-Lernens [9] verarbeitet wurden.

Falls ein Schnitt mit hoher Separations- und Teilungsqualität gefunden wurde und er vom Anwender als sinnvoll angesehen wird, kann man in den Untermengen rekursiv weiter nach geeigneten Schnitten suchen. So entsteht als Ergebnis eine einem Entscheidungsbaum ähnliche Struktur, die sich aber nicht an einer Trainingsmenge sondern an der Datenverteilung und der Aufgabe des Anwenders orientiert. Erste Ergebnisse wurden in [8] vorgestellt.

Literatur

[1] Charu Aggarwal and Phillip Yu. The igrid index: Reversing the dimensionality curse for similarity indexing in high dimensional space. In *SIGKDD 2000, Proceedings 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 119–129, 2000.

[2] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 49–60. ACM Press, 1999.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful? In *Proc. of the Int. Conf. Database Theorie*, pages 217–235, 1999.

[4] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD 1996, Proceedings 3rd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[5] B. Fritzke. The LBG-U method for vector quantization – an improvement over LBG inspired from neural networks. *Neural Processing Letters*, 5(1):35–45, 1997.

[6] A. Hinneburg, C. Aggarwal, and D.A. Keim. What is the nearest neighbor in highdimensional spaces. In *Proc. 26th Int. Conf. on Very Large Data Bases*, 2000.

[7] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD 1998, Proceedings 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 58–65. AAAI Press, 1998.

[8] A. Hinneburg, D.A. Keim, and M. Wawryniuk. Hd-eye: Visual mining high-dimensional data. *IEEE Computer Graphics and Applications*, 19(5):22–31, 1999.

[9] T.-M. Martinetz. Competitive hebbian learing rule forms perfectly topology preserving maps. In *Proceeding of the Int. Conf. Artificial Neural Networks*, pages 427–434. Springer, Amsterdam, 1993.

[10] J. Sander, M. Ester, H.-P.Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery, an International Journal, Kluwer Academic Publishers*, 2(2):169–194, 1998.

[11] D.W. Scott. *Multivariate Density Estimation.* Wiley and Sons, 1992.

[12] B.W. Silverman. *Density Estimation.* Chapman & Hall, 1986.

[13] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. of 24rd Int. Conf. on Very Large Data Bases (VLDB'98)*, pages 194–205, 1998.

[14] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Jörg Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the Fourteenth International Conference on Data Engineering, February 23-27, 1998, Orlando, Florida, USA*, pages 324–331. IEEE Computer Society, 1998.

Extending K-Means Clustering to First-Order Representations

Mathias Kirsten[1] and Stefan Wrobel[2]

[1] German National Research Center for Information Technology, GMD - AiS.KD, Schloß Birlinghoven, D-53754 Sankt Augustin, mathias.kirsten@gmd.de
[2] Otto-von-Guericke-Universität Magdeburg, IWS, P.O.Box 4120, D-39106 Magdeburg, wrobel@iws.cs.uni-magdeburg.de

Abstract. In this paper, we present an in-depth evaluation of two approaches of extending k-means clustering to work on first-order representations. The first-approach, k-medoids, selects its cluster center from the given set of instances, and is thus limited in its choice of centers. The second approach, k-prototypes, uses a heuristic prototype construction algorithm that is capable of generating new centers. The two approaches are empirically evaluated on a standard benchmark problem with respect to clustering quality and convergence. Results show that in this case indeed the k-medoids approach is a viable and fast alternative to existing agglomerative or top-down clustering approaches even for a small-scale dataset, while k-prototypes exhibited a number of deficiencies.

1 Introduction

K-means clustering [16] has been a very popular technique for partitioning sets of objects and still is an interesting subject for research as the number of publications indicate [7, 11, 8]. In recent years the original restriction to numerical data has been relaxed by introduction of an extended k-means algorithm by [11] which is able to handle symbolic data as well. Nevertheless the family of k-means methods so far has been limited to propositional data only. This may turn out as a great disadvantage in domains where a larger expressive power is needed for adequate representation of the data.

In this paper, we therefore study two approaches that extend k-means clustering to work on first-order representations. The first approach, k-medoids, selects its cluster center from the given set of instances, and is therefore limited in its choice of possible centers. The second approach, k-prototypes, uses a heuristic prototype construction algorithm that is capable of generating new centers. The two approaches are empirically evaluated on a standard benchmark problem with respect to clustering quality and convergence.

The paper is organized as follows. In section 2, we outline the k-means optimization task for numerical and symbolic propositional data. In section 3, we present two approaches to extend k-means to work on first-order representations, namely k-medoids and k-prototypes, including a heuristic prototype construction algorithm. Results from our empirical evaluation are reported in section 4.

In the related work section, we discuss the relation to other first-order clustering systems. We conclude with a summary and some pointers to future work. The appendix gives a detailed description of the prototype construction algorithm.

2 The *k*-means algorithm

The general algorithm was introduced by [4] ([16] and [1] first named it k-means) and has become widely popular since. It is defined as follows: given a set of n instances $\mathcal{I}$ from an instance space X, a natural number $1 \leq k \leq n$, and a distance measure $d(I_1, I_2) \rightarrow \mathbb{R}^{\geq 0}, I_1, I_2 \in X$, find a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ into k non-empty disjunct clusters $C_l, l \in \{1, \ldots, k\}$, with $C_l \cap C_j = \emptyset$ and $\bigcup_l C_l = \mathcal{I}$ such that the overall sum of squared distances between instances and their clusters' center Q_l is minimized. Following the lines of [11], we can use indicator variables $w_{i,l}$ which take the value 1 iff instance I_i is in C_l, and 0 otherwise. Then the optimization task can be written as:

$$\text{Minimize} \qquad P(W, Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{i,l} d(X_i, Q_l) \tag{1}$$

$$\text{with } (W)_{n,k} = \begin{pmatrix} w_{1,1} & \cdots & w_{1,k} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \cdots & w_{n,k} \end{pmatrix}, \quad \sum_{l=1}^{k} w_{i,l} = 1, 1 \leq i \leq n, \text{ and } w_{i,l} \in \{0, 1\}.$$

To achieve the minimization, k-means iterates through a three-step process until the $P(W, Q)$ converges to some (local) minimum:

1. Select or construct a set of k instances $Q^{(0)} = \{Q_i^{(0)} | Q_i^{(0)} \in D, 1 \leq i \leq k, \forall i \neq j \; Q_i^{(0)} \neq Q_j^{(0)}\}$ (the cluster centers), and set $t = 0$.
2. Keep $Q^{(t)}$ fixed and compute the $W^{(t)}$ that minimizes $P(W, Q^{(t)})$ – i.e. regarding $Q^{(t)}$ as the cluster centers, assign each instance to the cluster of its nearest cluster center.
3. Keep $W^{(t)}$ fixed and generate $Q^{(t+1)}$ such that $P(W^{(t)}, Q^{(t+1)})$ is minimized – i.e. construct new cluster centers according to the current distribution of instances $\mathcal{I}$.
4. In case of convergence or if a given stopping criterion is fulfilled, print the result and terminate, otherwise set $t = t + 1$ and go to step 2.

As can be seen from equation (1) and step 3 in the above process, the two crucial operations are the distance computation between elements of the instance space X, and the computation of the new cluster centers $Q^{(t+1)}$ from a given distribution of the instances $\mathcal{I}$. Both operations depend strongly on the underlying representation formalism as well as on the elementary value types allowed within.

For numerical domains represented in attribute-value logic the (weighted) Euclidean distance is commonly chosen as the natural distance measure. With

this distance measure, computation of the mean of a cluster's instances returns the cluster's center, fulfilling the minimization condition of step 3.

[11] (and also [8]) showed that on propositional domains the k-means method can be extended to symbolic data by employing a simple matching distance measure together with a majority-vote strategy to compute the cluster "center" (i.e. the Q that minimizes $P(W,Q)$). In more detail, the distance between two instances $I_1, I_2 \in X$, with $I_1 = (a_{1,1}, \ldots, a_{1,m})$ and $I_2 = (a_{2,1}, \ldots, a_{2,m})$ consisting of symbolic arguments only, can be defined as:

$$d(I_1, I_2) = \sum_{i=1}^{m} d(a_{1,i}, a_{2,i}), \text{ with } d(a,b) = \begin{cases} 0, & \text{if } a = b \\ 1, & \text{otherwise} \end{cases}$$

With this distance measure, a majority vote strategy that selects an argument's most frequent value can be used to compute the cluster "center". So, given a cluster (respectively a set of instances) $\{I_1, \ldots, I_o\}$, with $I_l = (a_{l,1}, \ldots, a_{l,m}), 1 \leq l \leq o$, its center $Q = (q_1, \ldots, q_m)$ is computed by assigning $q_j, 1 \leq j \leq m$ the value most frequently encountered in $\{a_{1,j}, \ldots, a_{o,j}\}$.

This meets the minimization condition, as has been shown in [11]. Furthermore, by combining both the methods for numerical and categorical data, k-means can be extended to handle domains with mixed value types. To distinguish the methods for different argument types, [11] uses "k-means" for strictly numerical data, "k-modes" for categorical values, and "k-prototypes" for the mixture of both.

Nevertheless, k-means methods have been limited to propositional domains. In the following we will hence elaborate the components necessary to extend the existing approaches to first-order domains.

3 *K*-Means for first-order representations

The task of k-means clustering on a first-order representation as used in this paper can be defined as follows:

Given:

- a set of *instances* $\mathcal{I} \subseteq X$
- a *distance function* $d : X \times X \to \mathbb{R}$
- a *center function* $\mathsf{center} : 2^X \to X$
- background knowledge $\mathcal{B}$
- the number of clusters k
- an initial set of k *seeds* $Q^0 \subseteq X$

Find:

- a set of clusters $\mathcal{C} = \{C_1, \ldots, C_k\} \subseteq 2^X$
- that maximizes given *quality criteria*

While the set of seeds Q^0 can be initialized by randomly drawing k instances from $\mathcal{I}$, which is the defacto standard according to [7], the crucial issues for getting k-means to work with first-order representations are the choice of an appropriate distance measure and the definition of the function finding a clusters' center.

Distance measures For propositional numerical data, the (weighted) Euclidean distance is a natural choice, as are simple matching similarities for categorical data. As a matter of fact, similarly natural choices for multirelational data are not at hand. Nevertheless, several distance measures for first-order representations have been proposed [18, 12, 6, 22, 21, 9]. Because the distance measure employed in RIBL [9] has exhibited excellent results on several domains [6, 10], we decided to use it as the basis for this work.

Computing cluster centers As reported in section 2, for propositional data solutions exist that compute the exact cluster center (i.e. the point that minimizes the sum of squared distances $\sum_{I_l \in C_i} d(I_l, Q_i)$). However, until now, no such approach is available for multirelational representations. Furthermore, the phrase "cluster center" allows no geometric or intuitive notion in case of multirelational data. Nevertheless, one can think of several options what the center of a set of multirelational instances could be:

One option is the LGG of the set of instances [19]. Unfortunately, for our purposes, the LGG exhibits three shortcomings. Firstly, computing the LGG of a set of instances and their background knowledge is computationally very expensive, and the LGG can grow exponentially with the number of instances. Secondly, the distance measure of RIBL is only defined between instances comprising sets of ground facts, i.e. no variables are allowed. And thirdly, it is very likely that LGG's would be very general (and often just the most general clause), so two different sets of instances could yield the same LGG, thus spoiling the further k-means process (e.g. assigning cluster-memberships).

Alternatively, the union of all instances in the set (respectively their ground facts) could do as the center. But in fact, such a "monster-instance" would hardly be suitable for further calculations, and the idea to achieve better computational complexity by using a cluster center instead of a cluster's set of instances would be spoiled totally.

Third, the most central instance of the set, its medoid, could be used as an approximation for the center. The most central instance is the instance whose sum of squared distances to all other instances in the set is minimal. This approach is already known as the k-medoids method [13, 14]. Depending on the domain, the results may suffer from the restrictions on possible cluster centers. If the (locally) optimal solution for a cluster center does not coincide with existing instances, the optimal solution cannot be found. Furthermore, the convergence process may fail when an oscillating state between either sides of an optimum is reached.

Fourth, an averaged instance, constructed by copying the structure from the instances of the set (or by overlapping several structures, if more than one structure exists). The facts in the structure can be computed recursively, by building averaged objects or values for each argument of a fact. On the downside of this approach, it is not yet clear, if such a constructive approach can satisfy the criterion for the cluster center, or if a good approximation can be constructed anyhow.

Having detailed the different options, it becomes clear that only the latter two are computationally reasonable alternatives.

3.1 *K*-medoids

Similar to the k-means methods, the k-medoids algorithm [14] follows the process detailed in section 2. It differs in the computation of the cluster centers Q^{t+1} as the medoid approach restricts Q to be a subset of the existing instances $\mathcal{I}$ whereas k-means allows Q to be a subset of the whole instance space X. So, k-medoid approximates the actual cluster center by taking an existing instance that minimizes the given center criterion (e.g. the sum of squared distances) within its cluster.

Obviously, the whole process relies on distance information only, and in contrast to k-means, is hence applicable to any representation for which an appropriate distance measure is available. On the downside, when making no prior assumptions about the representation used, the computational complexity suffers from the search for the best approximations which is proportional to the squared number of instances in a cluster.

3.2 *K*-prototypes

The center finding function for the k-prototypes method should be able to find better approximations to the actual cluster center, as it is not limited in the choice of possible centers. In fact, an ideal center function constructs prototypes that minimize the center criterion (e.g. sum of squared distances). For non-propositional representations with structured and complex objects this is hard to achieve, and depends strongly on the underlying distance measure.

To construct a prototype for a set of multirelationally represented instances $\mathcal{I} = \{I_1, \ldots, I_n\}$ our approach uses a recursive descent strategy in order to take into account not only the arguments of the instances but also the available background knowledge. Each instance I_j in our representation consists of one target fact F_j and the set of related facts from the background knowledge $\mathcal{B}$.

For a start we consider the set of target facts $F = \{F_1, \ldots, F_n\}$ with a given predicate symbol and arity q. The prototype is then computed by generating a new q-ary atom and iteratively calculating the values of its arguments. In case an argument is of an atomic type like number or constant, this can be done by falling back to the known prototype functions for propositional representations. Otherwise, if an argument is a link to another object (respectively fact), all objects referred to by the argument are gathered and a new (sub-)prototype is built from these objects recursively.

Following this recursively descending process, the prototype's structure should eventually resemble the structure of the original instances. As the structure may contain cycles, a depth bound is employed to avoid infinite recursion.

Figure 1 shows an examplary prototype for a set of three instances I_1, I_2, I_3 (see appendix A for a detailed description).

Instances:

I_1 := { package(*set*1,125,*personal*),
cheese(*set*1,*camembert*,150,*france*),
wine(*set*1,*mouton*,1988,0.75),
wine(*set*1,*gallo*,1995,0.5),
vineyard(*gallo*,*famous*,*large*,*use*),
vineyard(*mouton*,*famous*,*small*,*france*)}

I_2 := { package(*set*25,195,*mail*),
cheese(*set*25,*roque fort*,200,*france*),
cheese(*set*25,*ricotta*,100, *italy*),
wine(*set*25,*mouton*,1995,0.75),
vineyard(*mouton*,*famous*,*small*,*france*)}

I_3 := { package(*set*10,150,*personal*),
cheese(*set*10,*camembert*,300,*france*)}

Resulting prototype:

Pr := { package(*new_id*1,156,*personal*),
cheese(*new_id*1,*camembert*,187,*france*),
wine(*new_id*1,*new_id*2,1992,0.6667),
vineyard(*new_id*2,*famous*,*small*,*use*)}

Fig. 1. Example of generated prototype from a set of three instances.

In order to apply the prototype algorithm to data sets containing lists, one needs to find the list with the minimal sum of distances between itself and all other lists. This leads to two alternatives. First, to use the medoid from the set of lists as an approximation. Second, to construct a new list that comes close to the desired minimization property.

Unfortunately, the medoid approach is computationally too expensive as soon as a more sophisticated distance measure like edit distance is employed, so this option is not further considered here.

The constructive approach we employ is less costly but does not take the neighborhoods of arguments into account as it is done by the edit distance. The length of the prototype list is set to the averaged length of the set of lists. Then, starting at the first position, each element of the prototype list is determined by a majority vote over the elements of the lists at the specific position. This can be done in $O(nm)$ time, where m is the average number of elements in a list and n is the number of lists.

4 Empirical Evaluation

In the preceding sections, we have presented two different adaptations of the k-means clustering method to first-order data, the k-medoid and the k-prototype algorithms. Similar to the original k-means, as pointed out above, both of these have to be regarded as optimization methods that search for a clustering with maximal quality with respect to the chosen distance measure. For k-means, it is known that excellent cluster quality results with very fast convergence in most domains. Consequently, the primary goal of the experiments reported subsequently was to find out whether these positive properties of propositional k-means as an optimization method carry over to first-order k-medoid and k-prototype.

In particular, we have examined the following questions.

– Is clustering quality on a par with the results of RDBC [15], an agglomerative clustering method based on the same distance measure[1]? Does prototype construction offer improved results by using better centers than those available in the data?
– Do k-medoid and/or k-prototype converge fast, and can they thus improve on agglomerative clustering where the entire hierarchy has to be built? Are the convergence properties of k-prototypes better than k-medoid which might oscillate due to lack of proper centers in the data?

4.1 Experimental setup

In order to allow detailed evaluation and easy comparison with other researchers' results, we have selected the standard "mutagenicity" benchmark dataset [23] for our experiments. In this application, the goal is to classify chemical compounds into "non-mutagenic" and "mutagenic". For the purposes of clustering, we have removed this class information and just worked with the compound descriptions, consisting of the usual set of twenty extensionally defined predicates encoding structural and non-structural information about the compounds. For most experiments reported here, the full background knowledge "BG4" was used; however, in order to be able to separately test the effect of list construction on the performance of k-prototypes, the list-free "BG3" background knowledge was used in one experiment also. Except where explicitly marked, we have worked with the larger ("regression-friendly") dataset consisting of 188 compounds.

All results reported below were obtained by averaging, for each value of k from 2 to 15, 10 different runs with independently chosen random starting configurations (k-medoid), respectively 5 different runs (k-prototypes). If error bars are shown in a figure, they are standard deviations across the different runs. To examine convergence, we let each method run continue for 50 iterations, in which period all runs had reached convergence or cycled around a local optimum. The reported quality measure values in summary evaluations are the arithmetic mean of the last 10 iterations (40 to 50).

In addition, the same data was applied to several versions of RDBC [15] of which we report the results for the average link and the single link runs. The RDBC algorithms normally construct cluster hierarchies and turn them into flat partitions via a threshold algorithm that automatically tries to find the optimal number of clusters. For sake of comparison, we forced the threshold algorithm to produce a predefined number of clusters from a clustering hierarchy and calculated the quality measures accordingly. Since RDBC is not iterative and does not depend on starting conditions all figures reported refer to this single run of the algorithm and its resulting single clustering.

[1] Note that in past work [15], we have already compared RDBC's agglomerative clustering to other non-distance-based clustering approaches, with favorable results.

4.2 Quality measures

Since k-means type algorithms optimize squared error with respect to their cluster center instead of average intra-cluster distance (which is used by the average-link variant of RDBC), both measures were considered and computed as follows[2] ($\mathsf{center}(C_i)$ denotes the selected medoid for k-medoid, respectively the computed prototype for k-prototypes):

$$\mathsf{intra\_sim}(C) = \frac{\sum_{i=1}^{k} \sum_{I_j, I_k \in C_i, j \neq k} \mathsf{similarity}(I_j, I_k)}{\sum_{i=1}^{k} |C_i|(|C_i|-1)}$$

$$\mathsf{avg\_squared\_dist}(C) = \frac{\sum_{i=1}^{k} \sum_{I_j \in C_i} d(I_j, \mathsf{center}(C_i))^2}{\sum_{i=1}^{k} |C_i|}$$

In addition, since the mutagenicity domain really is a classification problem, it is possible to "abuse" the clustering (built without class information) for classification by labeling (a posteriori) each cluster with its majority class, and classifying a test instance according to the label of the closest cluster. These figures thus can be compared to those found in literature for classificatory ILP systems [23, 24].

4.3 Results on cluster quality

As shown in Figure 2, experimental results[3] confirmed our expectations regarding the general quality of k-medoid clustering results to be on a par with the quality of agglomerative clustering . However, for k-prototypes, Figure 2 shows surprising results. While for small k, it performs comparably to k-medoid clustering, its performance drops for larger k. Given that a larger number of clusters each with a smaller number of elements should allow even more exact cluster prototype construction and smaller distances, this result is surprising.

Upon inspection of the runs, it could be seen that part of the reason for this outcome is the "loss" of clusters that happens during the process. This loss occurs in step 3 of the clustering process (see section 2) where each instance is assigned to the cluster of the prototype (or medoid) nearest to it. Some of the prototypes end up with no instances assigned to them, thus forming empty clusters which have to be removed to proceed with the process. As it turns out, only 2 to 4 effective clusters remained after 50 iterations. When comparing k-prototypes to k-medoid on effective number of clusters (Figure 3), the counter-intuitive drop in performance disappears, but overall performance is still lower, indicating that indeed the proposed method of prototype computation does not accurately construct a cluster center.

[2] Note that intra-cluster similarity as reported here normalizes the sum of similarities by the number of similarities, not by cluster size. The latter did not show any qualitative change in the results.

[3] Results shown here are for average intra-cluster distance, but similar results were obtained for square error.

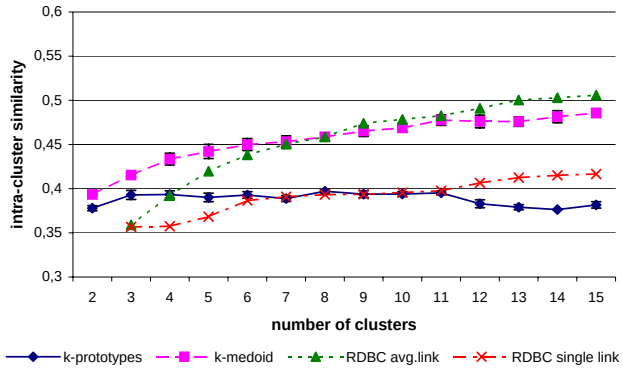


Fig. 2. Intra-cluster similarity for $k \in \{2, \ldots, 15\}$ clusters.

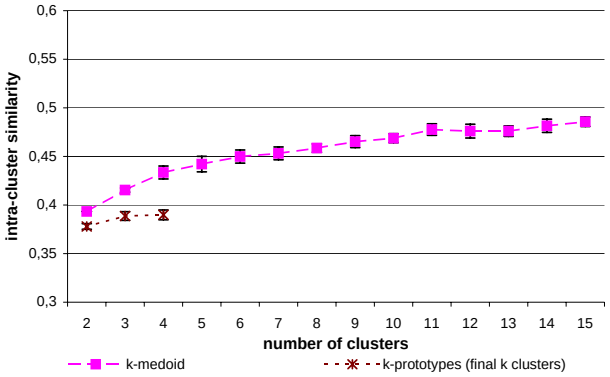


Fig. 3. Intra-cluster similarities of k-prototypes and k-medoids in regard to remaining number of clusters.

Since k-medoids performed well on this dataset, apparently even in 188 instances appropriate cluster centers were available. We therefore conducted additional experiments with the 42-instance dataset to see if lack of appropriate centers would force down performance there, and indeed (not shown graphically), for k larger than 9, a drop in k-medoid performance was observed. To complete the favorable results of k-medoid on this dataset, Figure 4 shows accuracy results compared to PROGOL [17], FOIL [20], and RIBL, where good accuracies are obtained even with a small number of clusters [4].

4.4 Results on convergence

Ideally, an iterative clustering method should reach a stable solution after a few iterations, as is often the case for propositional k-means. This should be reflected

[4] The tree-based first-order clustering system TIC has also been applied to this domain [3], however, no experiments using the complete background knowledge BG4 have been reported.

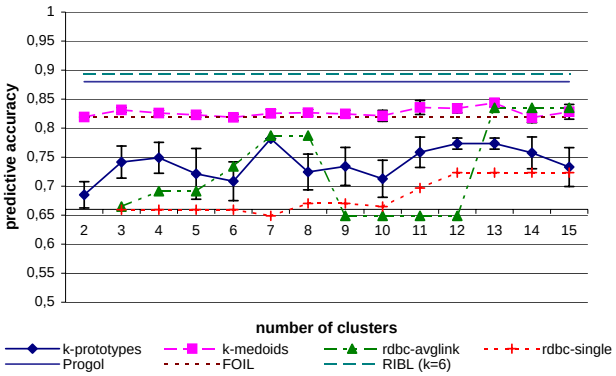


Fig. 4. Predictive accuracies on regression friendly data with BG4. Position of the X-axis indicating the accuracy of the default rule.

in stable quality measures for upcoming iterations. To evaluate convergence, we therefore computed (summed across all k) the 40 standard deviations of the intra-cluster similarities within a window of 10 consecutive iterations, e.g. data point 1 shows the standard deviation of quality within iterations 1 to 10 (Figure 5). As can be seen, the convergence of the k-medoids algorithm happens within the first 20 iterations. K-prototypes, however, starts the convergence process with slightly smaller changes than k-medoids and settles down at a significantly higher level of deviation, i.e., could not reach convergence.

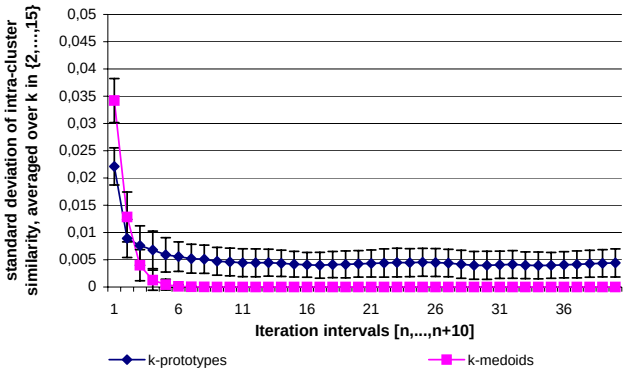


Fig. 5. Convergence properties of k-prototypes and k-medoids on regression friendly data with BG4, using the standard deviation of intra-cluster similarity on a window of 10 iterations, averaged over $k \in \{2, \ldots, 15\}$, as an indicator.

A closer look at the single runs reveals the reason, which lies in k-prototypes' strong tendency to go into cyclic states. Figure 6 shows an example of an 18-step

cycle reached after 9 iterations. As almost all runs end up in such cyclic states, strategies to resolve this dilemma are necessary.

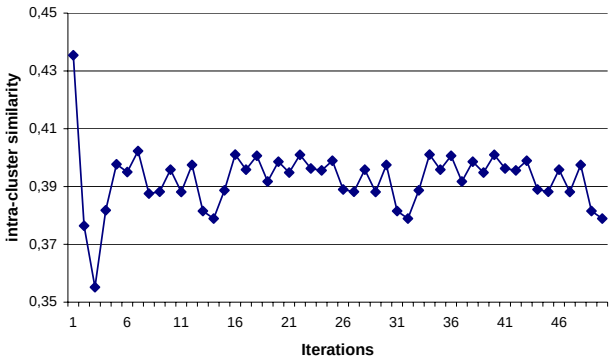


Fig. 6. Example of k-prototypes running into an 18-step cycle.

Evaluating the convergence of accuracy and squared-distance scores results in very similar figures, which are omitted here for sake of shortness.

4.5 Detailed study of k-prototype properties

Since the above results indicate that k-prototypes does not appropriately construct cluster centers, we decided to further investigate why this is the case. To investigate which parts of the construction are responsible for this deficiency, we singled out the list-construction part which uses an extremely simplistic approach that quite obviously does not mirror the edit-distance criterion used in the distance measure. However, as shown in Figure 7, experiments with the list-free background knowledge "BG3" did not show any marked difference.

To test whether this was due to non-center positioning of prototypes, we explicitly examined the average square distance to the cluster center. Figure 8 shows both intra- and inter-cluster distances of prototypes and medoids in comparison. Interestingly, k-prototypes *does* exhibit smaller intra-cluster distances than k-medoid. However, its inter-cluster distances are also lower than for the medoids approach, indicating that there exists a qualitative difference between the generated prototypes and actual instances — otherwise intra and inter-cluster distances would have to be more similar to those of the medoids.

5 Related Work

The presented k-prototypes and k-medoids approaches contrast with a number of previous first-order clustering systems, namely KBG [2], Cola-2 [5], TIC[3], and RDBC [15]. These systems cover different clustering strategies. RDBC and

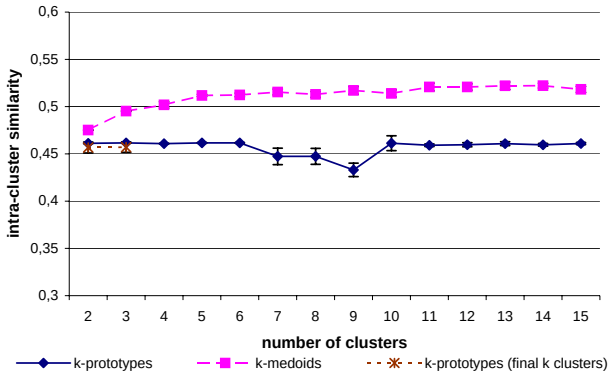


Fig. 7. Intra-cluster similarities of k-prototypes and k-medoids plotted in regard to the predefined number of clusters and in regard to the remaining number of clusters for k-prototypes.

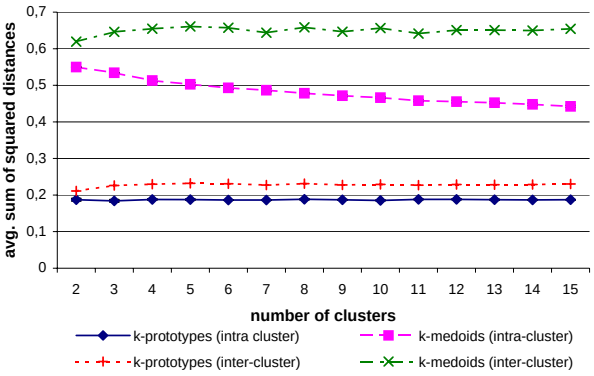


Fig. 8. Intra and inter-cluster distances of prototype and medoid algorithms for different numbers of clusters.

KBG follow a bottom-up clustering approach, and TIC builds its clustering trees in top-down manner, while the k-prototypes and k-medoid methods iteratively optimize an existing partitioning. Both the k-prototypes as well as the k-medoid method rely on a first-order distance measure and k-medoids compares favorably to the other methods in terms of cluster quality. Furthermore, in addition to the purely distance based approach taken in RDBC, k-prototypes and k-medoids return a prototype respectively the medoid of each cluster. Although less succinct than the cluster descriptions generated by conceptual clustering systems like KBG, COLA-2, or TIC, the set of prototypes/medoids can nevertheless be useful for subsequent (distance-based) processing.

In addition, this work is related to the extension of k-means to handle symbolic arguments as proposed in [11]. There, a majority vote strategy is used to build prototypes for propositional instances with symbolic arguments. The mul-

tirelational prototype construction within our k-prototypes algorithm adopts this approach to determine the symbolic arguments of facts in a prototype.

6 Conclusion and future work

In this paper, we have presented an in-depth evaluation of two approaches of extending k-means clustering to work on first-order representations. The first-approach, k-medoids, selects its cluster center from the given set of instances, and is thus limited in its choice of centers. The second approach, k-prototypes, uses a heuristic prototype construction algorithm that is capable of generating new centers.

The empirical evaluation of the two approaches on the mutagenicity domain presents a very clearcut conclusion. In this domain, both in terms of cluster quality and convergence, k-medoids is on a par with non-iterative state-of-the-art agglomerative clustering as previously examined in RDBC [15], while k-prototypes performed significantly worse and generated artefacts due to its way of constructing clusters. If these findings generalize to further domains, they show that indeed the k-medoids approach presented here is a viable alternative to existing agglomerative or top-down clustering approaches even in small-scale datasets. To this end, more experiments should be done in the future with other domains.

As for prototype generation, improvements to the method presented here appear possible that might improve its center computation. If prototype generation did not simply use the most frequent fact, but instead selected the x most frequent facts to be part of the prototype, this might lead to better approximations of the cluster center (with respect to the goal of decreasing intra cluster and increasing inter cluster distance).

References

1. G. H. Ball and D. J. Hall. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12:153–157, 1967.
2. G. Bisson. Conceptual Clustering in a First-Order Logic Representation. In B. Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 458–462. John Wiley, 1992.
3. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In J. Shavlik, editor, *Proceedings of the Fiteenth International Conference on Machine Learning (ICML-98)*, pages 55–63. Morgan Kaufmann, 1998.
4. D. Cox. Note on grouping. *J. Am. Stat. Assoc.*, 52:543–547, 1957.
5. W. Emde. Inductive learning of characteristic concept descriptions from small sets to classified examples. In F. Bergadano and L. De Raedt, editors, *Proceedings of the 7th European Conference on Machine Learning*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 103–121. Springer-Verlag, 1994.
6. W. Emde and D. Wettschereck. Relational Instance-Based Learning. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann, 1996.

7. U. M. Fayyad, C. Rain, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In R. Agrawal, P. E. Stolorz, and G. Piatetsk-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 194–198. AAAI Press, 1998.
8. S. K. Gupta, K. Sambasiva Rao, and V. Bhatnagar. K-means Clustering Algorithm for Categorical Attributes. In M. K. Mohania and A. Min Tjoa, editors, *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery (DaWaK-99)*, volume 1676 of *Lecture Notes in Computer Science*, pages 203–208. Springer-Verlag, 1999.
9. T. Horváth, S. Wrobel, and U. Bohnebeck. Relational instance-based learning with lists and terms. Machine Learning (to appear).
10. T. Horváth, S. Wrobel, and U. Bohnebeck. Term comparisons in first-order similarity measures. In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *LNAI*, pages 65–79. Springer-Verlag, 1998.
11. Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.
12. A. Hutchinson. Metrics on Terms and Clauses. In M. Someren and G. Widmer, editors, *Proceedings of the 9th European Conference on Machine Learning*, volume 1224 of *LNAI*, pages 138–145. Springer-Verlag, 1997.
13. L. Kaufmann and P. J. Rousseeuw. Clustering by means of medoids. In Y. Dodge, editor, *Statistical Data Analysis based on the L_1 Norm*, pages 405–416. Elsevier Science Publishers, 1987.
14. L. Kaufmann and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley, 1990.
15. M. Kirsten and S. Wrobel. Relational distance-based clustering. In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *LNAI*, pages 261–270. Springer-Verlag, 1998.
16. J. McQueen. Some methods of classification and analysis of multivariate observations. In L. K. Le Cam and J. Neyman, editors, *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–293, 1967.
17. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
18. S.-H. Nienhuys-Cheng. Distance Between Herbrand Interpretations: A Measure for Approximations to a Target Concept. In N. Lavrač and S. Džeroski, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297 of *LNAI*, pages 213–226. Springer-Verlag, 1997.
19. G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.
20. J. Quinlan and R. Cameron-Jones. FOIL: A midterm report. In P. Brazdil, editor, *Proceedings of the 6th European Conference on Machine Learning*, volume 667 of *Lecture Notes in Artificial Intelligence*, pages 3–20. Springer-Verlag, 1993.
21. J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 271–280. Springer-Verlag, 1998.
22. M. Sebag. Distance induction in first order logic. In N. Lavrač and S. Džeroski, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, LNAI, pages 264 – 272. Springer-Verlag, 1997.

23. A. Srinivasan, S. Muggleton, and R. King. Comparing the use of background knowledge by inductive logic programming systems. In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming*, pages 199–230. Department of Computer Science, Katholieke Universiteit Leuven, 1995.
24. A. Srinivasan, S. Muggleton, M. Sternberg, and R. King. Theories for mutagenicity: a study in first-order and feature-based induction. *Artificial Intelligence*, 85:277 – 299, 1996.

A Heuristic prototype generation

To construct a prototype for a set of first-order instances $\mathcal{I} = \{I_1, \ldots, I_n\}$ our approach uses a recursive descent strategy in order to take into account not only the arguments of the instances' target fact but also the available background knowledge. Each instance I_j in our representation consists of one target fact F_j and the set of related facts from the background knowledge $\mathcal{B}$.

To illustrate the construction process, we give an example using a slightly extended version of the "present package" example introduced in [9]. The data comprises three instances of culinary packages together with the corresponding background knowledge. It is assumed that about each package, its price and its delivery mode is known (personal, mail, or pick-up). So, each instance is represented by a (target) fact containing three arguments: identifier, price, and delivery mode plus the related background knowledge. As the contents of each package is known – it consists of different red wines and cheeses – and information about the different vineyards is given, this information makes up the background knowledge. The two packages are represented as follows:

$$F_1 := \text{package}(set1,125,personal)$$
$$F_2 := \text{package}(set25,195,mail)$$
$$F_3 := \text{package}(set10,150,personal)$$

In these target facts, the first argument is of type object and refers to further information in the background knowledge $\mathcal{B}$:

wine(*set*1, *mouton*, 1988, 0.75)
wine(*set*1, *gallo*, 1995, 0.5)
wine(*set*25, *mouton*, 1995, 0.75)

cheese(*set*1, *camembert*, 150, *france*)
cheese(*set*10, *camembert*, 300, *france*)
cheese(*set*25, *roque fort*, 200, *france*)
cheese(*set*25, *ricotta*, 100, *italy*)

vineyard(*gallo*, *famous*, *large*, *usa*)
vineyard(*mouton*, *famous*, *small*, *france*)

To compute the prototype of a set of instances, in our example $\{I_1, I_2, I_3\}$, a new but yet empty fact Pr is generated first.

$$Pr := \text{package}(arg_1,arg_2,arg_3)$$

The values for the arguments (arg_1, arg_2, and arg_3) are computed sequentially. In case an argument is of an atomic type like number or constant, its value can

be computed by falling back to the known prototype functions for propositional representations.

In our example the second argument is of type integer and the third of symbolic type. Hence, arg_2 can be computed by calculating the mean of the argument's values and arg_3 is determined by majority vote on the third argument:

$$\begin{aligned} arg_2 &= \left\lfloor \tfrac{125+195+150}{3} \right\rfloor && = 156 \\ arg_3 &= \mathsf{majority\_vote}(\{personal, mail, personal\}) && = personal \end{aligned}$$

If an argument is of type object, like the first argument of F_j in our example, two cases have to be distinguished.

In the first case the argument points *down* to objects one level deeper in the instances' structure and one has to decide (e.g. via a depth bound) wether (sub-)prototypes of the related objects should be generated or not. If (sub-)prototypes should be generated, a new id for linking to them must be provided. Otherwise, the argument's value can be determined by majority vote analogous to symbolic argument types.

In our example the first argument of F_j is of type object and allows the wine and cheese facts to refer to it. As we are still at the top-level, prototypes for the underlying objects should be constructed. Hence, an ID for the new (sub-)protoypes is generated.

$$arg_1 = \mathsf{generate\_unique\_id}() = id_1$$

Having decided to build prototypes for the underlying objects, all facts the argument values refer to are gathered and sorted into subsets $(G_1, \ldots, G_r)$ according to their arity and predicate symbol. For each subset G_j a new prototype is constructed recursively.

Coming back to the example and gathering the background knowledge referring to I_1, I_2, and I_3, the related facts are grouped into subsets according to their predicate symbol and arity:

$$\begin{aligned} G_1 = \{&\text{cheese}(set1,\ camembert,\ 150,\ france),\\ &\text{cheese}(set10,\ camembert,\ 300,\ france),\\ &\text{cheese}(set25,\ roque\ fort,\ 200,\ france),\\ &\text{cheese}(set25,\ ricotta,\ 100,\ italy)\}\\ G_2 = \{&\text{wine}(set1,\ mouton,\ 1988,\ 0.75),\\ &\text{wine}(set1,\ gallo,\ 1995,\ 0.5),\\ &\text{wine}(set25,\ mouton,\ 1995,\ 0.75)\} \end{aligned}$$

For each subset a new (sub-)prototype is generated, in this case, one for wine and another one for cheese. All numeric and symbolic arguments are generated as described above, which results in the following prototypes:

$$\begin{aligned} Pr_{sub1} &= \text{cheese}(id_1, camembert,\ 187,\ france)\\ Pr_{sub2} &= \text{wine}(id_1,\ arg_{wine,2},\ 1992,\ 0.666667) \end{aligned}$$

In the second case, the argument points *up* to a fact one level above in the instances' structure. id_1 and id_2 from the (sub-)prototypes Pr_{sub1} and Pr_{sub2}

are good examples. The value for the argument has been generated in the level above and allows the new prototypes to refer to their parent fact (Pr).

Following our example, we find that the second argument of Pr_{sub2} is of type object and links down to vineyard. Assuming that a depth bound is not yet reached and a new subprototype should be constructed, we generate another id and proceed as above by constructing a new prototype for the referred background knowledge (i.e. the vineyard facts):

$$\begin{aligned} arg_{wine,2} &= \mathsf{generate\_unique\_id}() && = id_2 \\ Pr_{sub2_{sub1}} &= \text{vineyard}(id_2, \textit{famous}, \textit{small}, \textit{usa}) \end{aligned}$$

As the reader has possibly noticed, trying to compute the values for the last two arguments of $Pr_{sub2_{sub1}}$ using a simple majority vote results in a draw. Such a draw can be resolved by a random selection from the most frequent argument values or by using overall frequency in the dataset.

prototype (F,DepthBound,CurrentDepth,Link):
var
F : Set of q-tuples $\{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$
DepthBound : Depth bound for recursive prototype generation
CurrentDepth : Current depth level of the process
Link : Link to above level fact (possibly empty)

A : Vector of argument values $\{a_1, \ldots, a_n\}$

1. **for** $l := 1$ **to** q **do**
2. $a_i := \mathsf{f}_i[l], \forall i \in \{1, \ldots, n\}$ %% $\mathsf{f}_i[l]$: l-th argument of fact f_i
3. **case of** $type(a_i)$:
4. number: $\mathsf{arg}_l := \left(\sum_{i=1}^{n} a_i\right)/n$
5. constant: $\mathsf{arg}_l :=$ majority_vote(A)
6. list: $\mathsf{arg}_l :=$ list_prototype(A)
7. link: **if** (Link $\neq \{\}$) **then**
8. $\mathsf{arg}_l :=$ Link
9. **else if** (CurrentDepth $\geq$ DepthBound) **then**
10. $\mathsf{arg}_l :=$ majority_vote(A)
11. **else**
12. $\mathsf{arg}_l :=$ generate_unique_id()
13. NewFacts := gather_related_facts(A, F)
14. $(\mathsf{G}_1, \ldots, \mathsf{G}_r) :=$ split_into_q_atoms(NewFacts)
15. **for** $j := 1$ **to** r **do**
16. prototype$(\mathsf{G}_j, \mathsf{DepthBound}, \mathsf{CurrentDepth} + 1, \mathsf{arg}_l)$
17. **end for**
18. **end if**
19. **end case**
20. **end for**
21. **return** prototype_instance$(\mathsf{arg}_1, \ldots, \mathsf{arg}_q)$

gather_related_facts (A,F):
var A : Vector of values $\{a_1, \ldots, a_n\}$
F : Set of q-atoms $\{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$
1. **return** the set of all facts from the reduced background knowledge $B \setminus \mathsf{F}$ where a value from A appears in the position of an object-typed argument, i.e. return all facts referring to or referred by any of the values in A.

split_into_q_atoms (NewFacts):
var NewFacts : List of facts $\{\mathsf{nf}_1, \ldots, \mathsf{nf}_n\}$
1. **return** the list of subsets $(\mathsf{NewFacts}^1, \ldots, \mathsf{NewFacts}^r)$ that results from sorting and splitting the list of facts NewFacts according to their arity and predicate symbol. I.e. $\mathsf{NewFacts}^j$ contains only facts of the same arity and predicate symbol.

Fig. 9. Prototype algorithm for set of facts.

Fuzzy-Modelle zur expliziten Repräsentation zeitlicher Beziehungen

Andreas Fick, Hubert B. Keller
Institut für Angewandte Informatik
Forschungszentrum Karlsruhe
Hermann-von-Helmholtz-Platz 1
D-76344 Leopoldshafen
E-mail: fick@iai.fzk.de

1. September 2000

Abstract

Der Artikel beschäftigt sich mit der symbolischen, regelbasierten Modellierung des Verhaltens dynamischer Systeme. Neben der Erzeugung eines solchen Modells aus Prozeßdaten (Zeitreihen) besteht zunächst das Problem, überhaupt eine geeignete Modellstruktur zu definieren. Eine solches Modell sollte sowohl unscharfe Daten verarbeiten können als auch die explizite Repräsentation zeitlicher Beziehungen ermöglichen. Als Lösung wird eine auf dem Fuzzy Control-Ansatz von Mamdani basierende Modellstruktur vorgeschlagen und beschrieben.

1 Einleitung

Steht für ein dynamisches System kein analytisches Modell zur Verfügung, weil eine mathematische Modellierung beispielsweise aufgrund der hohen Anzahl von Prozeßgrößen, weil nicht alle Größen meßtechnisch erfaßbar sind oder weil viele Systemzusammenhänge nicht bekannt sind, gescheitert ist, so kann der Einsatz datengetriebener Verfahren eine Möglichkeit darstellen, dennoch ein brauchbares Verhaltensmodell zu erhalten [KF98].

Zur Erzeugung von Verhaltensmodellen aus Prozeßdaten (Zeitreihen) werden beispielsweise künstliche Neuronale Netze erfolgreich eingestzt [MK96]. Ein solches Modell besitzt jedoch den Nachteil, eine „Black Box“ darzustellen, die vom Menschen nur sehr schwer verstanden werden kann. Insbesondere ist auch die Beurteilung des Modells schwierig, was verständlicherweise auch zu Akzeptanzproblemen beim Anwender führt, der einem Modell vertrauen soll, das keinerlei Begründung für sein Verhalten liefert. Deshalb besteht das Ziel, verständliche, regelbasierte Modelle zu verwenden.

Zur Erzeugung derartiger symbolischer Modelle steht prinzipiell eine große Zahl maschineller Lernverfahren zur Verfügung [Kel00], die jedoch nicht direkt verwendet werden können [Wei95].

Die Probleme beginnen bereits damit, daß nicht klar ist, welches die zu betrachtenden Basismerkmale sind, d.h. auf welcher Grundlage überhaupt nach Zusammenhängen gesucht werden soll. Geht man von einzelnen Meßwerten, von Meßintervallen, von ganzen Meßwertreihen oder geeigneten, aggregierten Größen als Attributen aus? Damit besteht die erste Aufgabe eines automatischen Modellierungssystems in der Erzeugung einer geeigneten symbolischen Repräsentationsbasis meßbarer Größen (Definition der Merkmale), die dann die Grundlage für die gewünschte Analyse und Regelgenerierung darstellt. Dieser Übergangsschritt von der numerischen zur symbolischen Form stellt einen wesentlichen Unterschied z.B. gegenüber der Suche nach implizitem Wissen in Datenbanken (data mining, knowledge discovery in databases) dar.

Des weiteren ist bei der Modellierung eines dynamischen Systems die Zeit als wichtiger Parameter zu berücksichtigen. Dies bedeutet, daß Beziehungen zwischen verschiedenen Systemgrößen und damit entsprechend auch zwischen den daraus abgeleiteten Merkmalen nicht statisch sind, sondern verschiedenartige, zustandsabhängige Beeinflussungen mit gewissen Zeitverzögerungen anzunehmen sind. Deshalb müssen Lernbeispiele neben der aktuellen Situation auch alle relevanten Situationen der Vergangenheit enthalten, was aufgrund der Größe des zu durchsuchenden Hypothesenraums (kombinatorische Explosion) zwangsläufig zum Scheitern direkt angewandter etablierter maschinellen Lernverfahren wie z.B. ID3/CN2/C4.5 führt.

Als weiterer Punkt soll die Forderung nach dem angemessenen Umgang mit unpräziser, unscharfer In-

formation genannt werden. Dies betrifft insbesondere auch die Ausgabe des Modells, bei der es nicht genügt, einen bestimmten Zustand aus einer Menge vorgegebener Prozeßzustände im Sinne einer Klassifikation vorherzusagen, sondern bei der die Werte möglichst aller Größen zu prognostizieren sind.

2 Bezeichnungen

Gegeben seien die Systemgrößen X_i. Wird eine eindeutige oder beliebige Systemgröße betrachtet, so wird im folgenden der Index weggelassen. Zur einfacheren Unterscheidung von den Größen der Prämisse werden in den Konklusionen von Fuzzyregeln auftretende Größen mit Y beziehungsweise Y_i bezeichnet.

$x_i(t)$ bezeichnet den Wert der Systemgröße X_i zum Zeitpunkt t. Schätzwerte werden durch ein „ˆ“ gekennzeichnet, z.B. $\hat{x}_i(t)$.

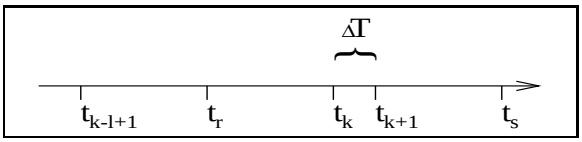


Abbildung 1: Bezeichnungen auf der Zeitachse.

Die Folge $\{t_n\}$ äquidistanter, diskreter Zeitpunkte mit dem Abstand ΔT, d.h. $t_{n+1} = t_n + \Delta T$ bezeichnet die Folge der Zeitpunkte, zu denen die Werte des gegebenen Systems betrachtet werden, bei Meßgrößen also die Abtastzeitpunkte.

A und B sind Fuzzymengen mit Zugehörigkeitsfunktion $\mu_A(x)$ bzw. $\mu_B(x)$. T bezeichnet eine Fuzzymenge für eine (relative) Zeitangabe und wird analog definiert.

$R = R_1, \ldots, R_N$ bezeichnet die Menge der Regeln eines Fuzzyreglers, die sich im folgenden zur Vereinfachung der Schreibweise implizit alle auf dieselbe linguistische Variable bzw. Größe Y beziehen sollen.

Im Zusammenhang mit Fuzzyregeln wird im folgenden der Zeitpunkt des Zutreffens der Prämisse, bzw. des betrachteten Fuzzy-Terms der Prämisse, mit t_r und der Zeitpunkt des Zutreffens der Konklusion mit t_s bezeichnet.

Die zeitliche Differenz zwischen beiden Zeitpunkten ist $\Delta t := t_s - t_r = t_d$ bzw. $\Delta t = t_d$ mit $d = s - r$ bzw. $\Delta t = d * \Delta T$.

Zur Erläuterung konkreter Berechnungen im Zusammenhang mit Prognosen ist wichtig, welche Daten zum Zeitpunkt der Prognoseerstellung bekannt sind. Hierzu werden im folgenden die bekannten Zeitpunkte, d.h. das (endliche) „Gedächtnis“ der Länge t_l, mit $t_{k-l+1} \ldots t_k$ bezeichnet, die betrachtete „Zukunft“ wird durch die Zeitpunkte $t_{k+1} \ldots t_{k+m}$ repräsentiert.

3 Ausgangspunkt

Die in der Einleitung angeführten Forderungen an die Modellstruktur lassen sich mit den Stichworten „regelbasiertes Modell“, „Verarbeitung unscharfer Daten“ und „Behandlung von Zeitabhängigkeiten“ zusammenfassen. Ausgangspunkt der folgenden Überlegungen ist daher eine Fuzzy-Regelbasis nach Mamdani [MA75], die zwar eigentlich zum Zweck der Regelung (Fuzzy Control) konzipiert worden ist, sich aber aufgrund der universellen Struktur auch zur Modellierung des Verhaltens dynamischer Systeme anbietet. Sie basiert auf Regeln der Art

Wenn [x ist A] Dann [y ist B].

Zu einer Menge solcher Regeln ist zum einen die Prämissenauswertung (vgl Abbildung 2) nach den Regeln der Fuzzylogik [Zad65] mit den bekannten Wahlmöglichkeiten hinsichtlich der logischen Verknüpfungen (t-Normen und t-Conormen) und zum anderen das Inferenzprinzip (siehe Abbildung 3) definiert.

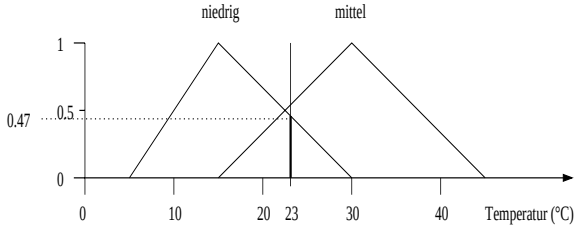


Abbildung 2: Prämissenauswertung von „Wenn t ist niedrig Und t ist mittel Dann v ist offen“ für $t_0 = 23$ und Singleton-Fuzzifizierung.

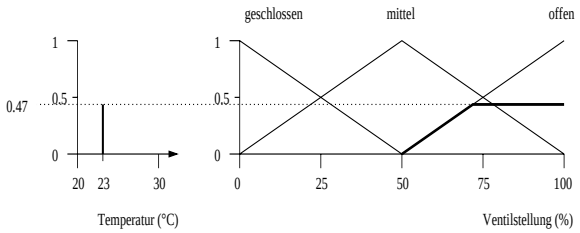


Abbildung 3: Konklusionsberechnung von „Wenn t ist niedrig Und t ist mittel Dann v ist offen“ für $t_0 = 23$ mit Minimum-Inferenz.

Bei diesem wird im wesentlichen die Prämisse jeder einzelnen Regel auf einen Wert reduziert und anschließend in verschiedenen Ausprägungen, z.B.

durch Minimum- oder Produktbildung, mit der jeweiligen unscharfen Konklusionsmenge verknüpft.

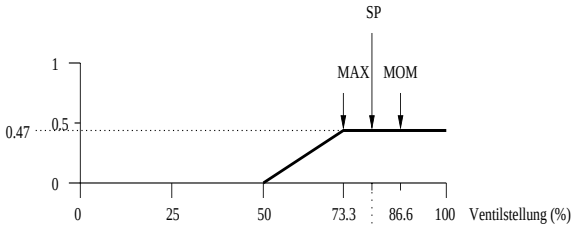


Abbildung 4: Defuzzifizierung der Ergebnismenge mit Maximum-, Mean-of-Max- und Schwerpunktmethode.

Anschließend folgt ein „Komposition“ genannter Schritt, der die zu einer Größe gehörenden Konklusionsmengen zu einer Menge zusammenfaßt und zum Schluß die Defuzzifizierung (siehe Abbildung 4) zur Reduktion der Ergebnismenge auf einen scharfen Wert.

Der im folgenden beschriebene Ansatz verfolgt das Ziel, einen möglichst großen Teil dieser etablierten Methode beizubehalten. Dadurch können ihre Vorteile wie Bekanntheit, Transparenz oder Effizienz genutzt werden, und es kann allgemein auf den damit gemachten weitgehend positiven Erfahrungen und den erzielten Forschungsergebnissen aufgebaut werden. Außerdem ist so eine einfache Realisierung (Implementierung) auf Basis existierender Software-Tools, wie z.B. Matlab, möglich.

4 Beobachtung

Im Gegensatz zu Differentialgleichungsmodellen, in die zeitliche Bezüge direkt eingehen, beschreibt eine Regel einer Fuzzyregelbasis im wesentlichen einen statischen Zusammenhang zwischen den Fuzzymengen A und B. Dynamische Aspekte werden in Regelbasen nach Mamdani lediglich implizit über den Auswertezeitpunkt der Prämissen und den „Takt“ des Reglers, der zyklisch in bestimmten Abständen Werte erfaßt, auswertet und anschließend Stellgrößen ausgibt, berücksichtigt. Desweiteren können sie bei Regelbasen nach Sugeno-Takagi [Sug85], die als Abwandlung des Mamdani-Ansatzes angesehen werden können, prinzipiell in die Funktionsausdrücke der Konklusionen eingehen.

Trotz der Repräsentation ohne Zeitbezug muß bei der Verwendung einer Regel im Rahmen von Fuzzy Control natürlich eine zeitliche Zuordnung zwischen Prämisse bzw. Konklusion der Regel und Werten der Systemgrößen X bzw. Y vorgenommen werden, und der durch obige Regel festgelegte Zusammenhang läßt sich genauer beschreiben als Beziehung zwischen $\mu_A(x(t_r))$ und $\mu_B(y(t_s))$ mit dem „Meßzeitpunkt“ t_r und dem „Stellzeitpunkt“ t_s.

5 Regeln mit expliziter Zeitdarstellung

Eine explizite Angabe der Zeitpunkte innerhalb der Regel würde den zeitlichen Zusammenhang zwischen Prämisse und Konklusion offensichtlich machen. Dies entspräche einer am menschlichen Denken in Ursache-Wirkungsbeziehungen orientierten Beschreibung, die die Verständlichkeit solcher Regelbasen für den Menschen erleichtern würde [Dö76]. Außerdem würde sie eine Behandlung unterschiedlicher Verzögerungszeiten auf der Ebene des Fuzzy Control ermöglichen. Unterschiedliche Totzeiten im betrachteten dynamischen System könnten in der Regelbasis explizit angegeben werden und müßten nicht in unterlagerte Schichten, z.B. in die Definition der Meßgrößen, „eingebaut“ werden.

Natürlich wäre eine absolute Angabe der Zeitpunkte, im obigen Beispiel also von t_r und t_s, nicht sinnvoll. Vielmehr ist nur die Verzögerungszeit (Totzeit) Δt zwischen Ursache und Wirkung als zeitliche Differenz von t_s und t_r wesentlich.

Desweiteren sollte die Angabe ungefährer Zeitverzögerungen durch die Verwendung unscharfer Mengen für relative Zeitangaben möglich sein. Der Grund hierfür liegt zum einen im Wunsch nach Verständlichkeit des resultierenden Modells, die durch Verwendung linguistischer Werte für Zeitangaben erreicht werden soll. Zum anderen erlaubt die Zusammenfassung (Clusterung) zeitlich ähnlicher, nicht aber unbedingt identischer Beziehungen im Rahmen der maschinellen Modellerstellung aus Zeitreihen lediglich ungefähre Zeitangaben (vgl. [Wei95]). Auch das Einbringen unscharfen (Experten-) Wissens wird so ermöglicht.

„Scharfe“ Zeitangaben ergeben sich als Sonderfälle durch die Wahl geeigneter Zugehörigkeitsfunktionen (Singletons).

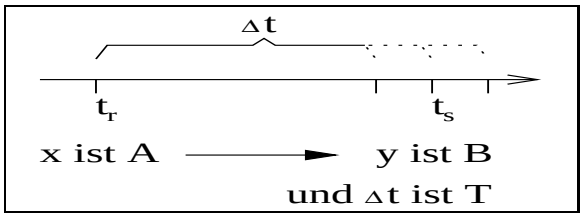


Abbildung 5: Regel Typ 1.

Unter diesen Voraussetzungen und Beibehaltung der Struktur des Fuzzy Control-Ansatzes nach

Mamdani bieten sich zwei äquivalente Darstellungsformen an. Zum einen

$$\text{Wenn } [x \text{ ist } A] \text{ Dann } [y \text{ ist } B \text{ und } \Delta t \text{ ist } T]. \tag{1}$$

mit Δt als der Zeitverzögerung, nach der „y ist B“ gelten soll, angegeben relativ zum implizit angenommenen Bezugszeitpunkt t_r der Auswertung von „x ist A“.

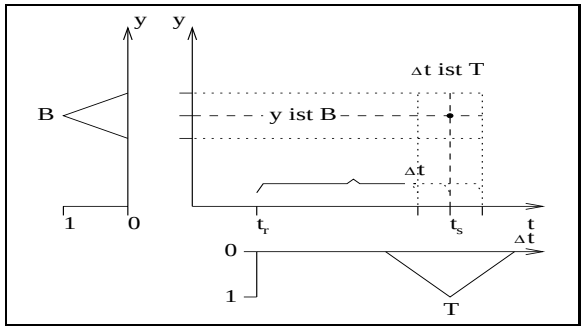


Abbildung 6: [y ist B und ΔT ist T].

Hier wurde eine „Zeitverzögerung von T“ in Fuzzy-Schreibweise als „Δt ist T“ übersetzt und zur Verdeutlichung der zeitlichen Zuordnung (vgl. Abbildung 6) „und“ geschrieben. Dieses „und“, das so etwas wie „nach“ oder „zum (relativen) Zeitpunkt“ ausdrückt, *kann* als übliche Fuzzy-Und-Verknüpfung (t-Norm) umgesetzt werden (siehe Abschnitt 6), und die Entscheidung für dieselbe t-Norm wie bei der Konjunktion kann gegebenenfalls zu einem besonderes effizienten Verarbeitungsschema führen. Im Prinzip ist die Wahl der entsprechenden zweidimensionalen Fuzzyrelation jedoch unabhängig von den sonst gewählten Verknüpfungsoperatoren wie z.B. Minimumbildung oder Multiplikation. Selbstverständlich sollten dabei sinnvolle Nebenbedingungen, beispielsweise Monotonie in den Zugehörigkeitsgeraden, berücksichtigt werden.

Im folgenden wird ein mit einer unscharfen (relativen) Zeitangabe T versehener Fuzzy-Term, d.h. z.B. die zweidimensionale Fuzzymenge (Fuzzyrelation) $B\tilde{\wedge}T$ als „zeitbehafteter Fuzzy-Term“, kurz „t-Fuzzy-Term“, bezeichnet.

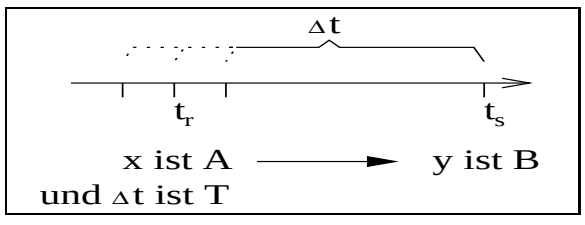


Abbildung 7: Regel Typ 2.

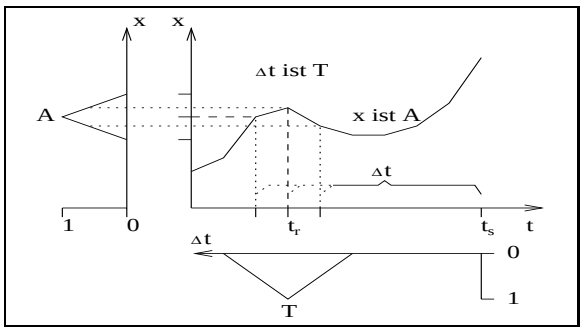


Abbildung 8: [x ist A und ΔT ist T].

Die zweite Möglichkeit besteht in der Angabe der Zeitverzögerung von „x ist A“ relativ zum Zeitpunkt t_s von „y ist B“ (Abbildung 7):

$$\text{Wenn } [x \text{ ist } A \text{ und } \Delta t \text{ ist } T] \text{ Dann } [y \text{ ist } B]. \tag{2}$$

Da im vorliegenden Fall keine allgemeine logische, sondern eine kausale Beziehung, bei der „x ist A“ immer vor „y ist B“ liegt, angegeben wird, genügt es, sich auf positive Zeitverzögerungen (Beträge) zu beschränken, und für beide Darstellungsformen können dieselben Fuzzymengen verwendet werden.

Zu beiden Darstellungsmöglichkeiten ist zu bemerken, daß bei diesen einfachen Regeln eindeutig erkennbar ist, auf welche Zeitpunkte sich die Zeitverzögerung bezieht. Bei Regeln mit mehreren Fuzzy-Termen in der Prämisse und einem Term in der Konklusion ist die jeweilige Zeitverzögerung für jeden einzelnen Fuzzy-Term der prämisse anzugeben, falls nicht alle Fuzzy-Terme zum selben (relativen) Zeitpunkt ausgewertet werden sollen, etwa

$$\begin{aligned} &\text{Wenn } [x_1 \text{ ist } A_1 \text{ und } \Delta t_1 \text{ ist } T_1] \\ &\quad \text{Und } [x_2 \text{ ist } A_2 \text{ und } \Delta t_2 \text{ ist } T_2] \\ &\text{Dann } [y \text{ ist } B]. \end{aligned} \tag{3}$$

Im folgenden wird ausschließlich der Ansatz nach Regel 2 weiterverfolgt, da dieser, wie gerade angedeutet, eine naheliegende Erweiterung auf Prämissen mit mehreren Bedingungen und damit auch mehreren Zeitverzögerungen, die sich auf dieselbe Konklusionsgröße auswirken, ermöglicht, während z.B. mehrere verknüpfte Fuzzy-Terme in der Konklusion nicht üblich sind und eine Vermischung beider Ansätze unübersichtlich würde.

An dieser Stelle soll darauf hingewiesen werden, daß beide Arten von Regeln entgegen der intuitiven Vorstellung nur bei geeigneter Wahl des Inferenzschemas logisch äquivalent sind (siehe Abschnitt 7).

6 Semantik und Inferenzschema

Im folgenden wird die Angabe der Gleichzeitigkeit als Konjunktion „$\tilde{\wedge}$“ interpretiert (siehe Erläuterungen oben) und als Verknüpfungsoperation das Minimum gewählt. Weiter wird von Singleton-Fuzzifizierung ausgegangen. Als Aggregationsoperator für die „normale“ Fuzzy-Konjunktion wird die Multiplikation verwendet, für die Inferenz ebenfalls. Auch hier ist jeweils die Auswahl unter den üblichen Methoden möglich.

Damit führt die Analyse der linken Seite einer Regel nach Art von 2 im Rahmen der Fuzzylogik zu

$$\mu_{A\tilde{\wedge}T}(x, \Delta t) = \min\{\mu_A(x),\ \mu_T(\Delta t)\}.$$

Die Konjunktionsrelation „$A\tilde{\wedge}T$“ beschreibt eine zweidimensionale Fuzzymenge, was bedeutet, daß sie nicht nur eine Aussage für einen bestimmten Zeitpunkt, sondern für eine Zeitspanne macht. Zur Beibehaltung des prinzipiellen Vorgehens nach Mamdani ist diese Menge für die Inferenz auf einen Wert zu reduzieren.

Am einfachsten ist es, hierzu als Ergebnis der Auswertung des zeitbehafteten Fuzzy-Terms den Wert mit der maximalen Zugehörigkeit zu verwenden, d.h. nach dem besten Kompromiß aus x und Δt zu suchen. Diese Definition des Zugehörigkeitsgrades basiert auf einer Interpretation eines t-Fuzzy-Terms dahingegend, daß er dann als gut erfüllt gilt, wenn er dieses zu irgendeinem Zeitpunkt ist. In Abbildung 8 würde also z.B. zwischen der Auswertung des t-Fuzzy-terms $A\tilde{\wedge}T$ zum Zeitpunkt t_{r-1} mit $\mu_A(x(t_{r-1})) = 1$, aber nicht passender Zeit, sowie zum perfekt passenden Zeitpunkt t_r ($\mu_T t_r = 1$), aber zu großen Wert von $x(t_r)$, abgewogen werden. Als Ergebnis zum Bezugszeitpunkt t_s ergibt sich

$$\mu_P(t_s) = \max_{\Delta t}\{\mu_{A\tilde{\wedge}T}(x(t_s - \Delta t), \Delta t)\}. \tag{4}$$

Man erhält als Ergebnis die Zugehörigkeitsfunktion der Konklusion zum Zeitpunkt t_s

$$\mu_C(t_s, y) := \mu_{A\tilde{\wedge}T\to B}(y) = \mu_P(t_s) * \mu_B(y). \tag{5}$$

Bei mehreren t-Fuzzy-Termen in der Prämisse ergibt sich beispielsweise (vergleiche Regel 3):

$$\mu_{(A_1\tilde{\wedge}T_1, A_2\tilde{\wedge}T_2)\to B}(y) = (\mu_{P_1}(t_s) * \mu_{P_2}(t_s)) * \mu_B(y). \tag{6}$$

7 Alternativen

Zur expliziten Repräsentation zeitlicher Zusammenhänge ist eine bisher nicht überschaubare Menge verschiedener Ansätze denkbar. Als Ergänzung zum oben erläuterten sollen kurz zwei weitere Möglichkeiten erwähnt werden.

7.1 Zeitangaben in der Konklusion

Geht man von Regeln der Art (1) aus, so erhält man zunächst als zweidimensionale Zugehörigkeitsfunktion der Konklusion

$$\mu_{B\tilde{\wedge}T}(y, \Delta t) = \min\{\mu_B(y), \mu_T(\Delta t)\}.$$

Mit Produktinferenz ergibt sich

$$\mu_{A\to(B\tilde{\wedge}T)}(y, \Delta t) = \mu_A(x(t_r)) * \mu_{B\tilde{\wedge}T}(y, \Delta t).$$

Hierdurch werden ausgehend vom Wert von x zum Bezugszeitpunkt t_r Aussagen über Zugehörigkeitsfunktionen der Konklusionsmengen für verschiedene Zeitpunkte $t_s = t_r + \Delta t$ gemacht. Möchte man alle Aussagen, die einen bestimmten Zeitpunkt t_s betreffen, zu einer Fuzzymenge zusammenfassen, um später sukzessive für jeden Zeitpunkt t_s die Konklusionsmengen verschiedener Regeln mittels Komposition auf übliche Weise weiterverarbeiten zu können, so erhält man, falls man analog zur Herleitung von Gleichung 4 vorgeht,

$$\mu_{C'}(t_s, y) := \max_{\Delta t}\{\mu_A(x(t_s - \Delta t)) * \mu_{B\tilde{\wedge}T}(y, \Delta t)\}. \tag{7}$$

Man erkennt in der ausgeschriebenen Version, daß sich dieses Ergebnis im allgemeinen von Gleichung 5 unterscheidet:

$$\begin{aligned}
&\mu_{C'}(t_s, y)\\
&= \max_{\Delta t}\{\mu_A(x(t_s - \Delta t)) * \mu_{B\tilde{\wedge}T}(y, \Delta t)\}\\
&= \max_{\Delta t}\{\mu_A(x(t_s - \Delta t)) * \min\{\mu_B(y), \mu_T(\Delta t)\}\}\\
&= \max_{\Delta t}\{\min\{\mu_A(x(t_s - \Delta t)) * \mu_B(y),\\
&\qquad \mu_A(x(t_s - \Delta t)) * \mu_T(\Delta t)\}\}\\
&= \max_{\Delta t}\{\min\{\mu_A(x(t_s - \Delta t)), \mu_A(x(t_s - \Delta t))\\
&\qquad * \frac{\mu_T(\Delta t)}{\mu_B(y)}\} * \mu_B(y)\}\\
&\overset{i.a.}{\neq} \max_{\Delta t}\{\min\{\mu_A(x(t_s - \Delta t)), \mu_T(\Delta t)\} * \mu_B(y)\}\\
&= \max_{\Delta t}\{\min\{\mu_A(x(t_s - \Delta t)), \mu_T(\Delta t)\}\} * \mu_B(y)\\
&= \max_{\Delta t}\{\mu_{A\tilde{\wedge}T}(x(t_s - \Delta t), \Delta t)\} * \mu_B(y)\\
&= \mu_P(t_s) * \mu_B(y)\\
&= \mu_C(t_s, y)
\end{aligned}$$

Die Gleichungen 5 und 7 wären beispielsweise dann äquivalent, wenn man statt des Minimums für „$\tilde{\wedge}$“ das Produkt wählen würde, oder wenn in den Gleichungen statt des Produktes Minimumbildung stehen würde, d.h. für die Inferenz statt der Dot-Methode Minimumbildung gewählt worden wäre.

7.2 Regeln als zusammenfassende Regeln

Man kann eine Regel nach (2) auch dahingehend interpretieren, daß es sich um die Kurzschreibweise für eine Menge von Regeln mit jeweils präzisen Zeitangaben handelt, wobei jede Regel gemäß dem Zutreffen der Zeitangabe gewichtet würde. Eine Regel der Art von (2) entspräche z.B. den „normalen“ gewichteten Fuzzyregeln

Wenn $[x(t_r - \Delta T)$ ist $A]$ Dann $[y$ ist $B]$
  Mit Gewicht $\mu_T(\Delta T)$.
Wenn $[x(t_r - 2 * \Delta T)$ ist $A]$ Dann $[y$ ist $B]$
  Mit Gewicht $\mu_T(2 * \Delta T)$.
Wenn $[x(t_r - 3 * \Delta T)$ ist $A]$ Dann $[y$ ist $B]$
  Mit Gewicht $\mu_T(3 * \Delta T)$.
...
  ...
Wenn $[x(t_r - k * \Delta T)$ ist $A]$ Dann $[y$ ist $B]$
  Mit Gewicht $\mu_T(k * \Delta T)$.

Für jede dieser Regeln ergäbe sich eine normale Konklusionsberechnung, und die Konklusionen aller dieser Regeln könnte man beispielsweise wie bei „normalen“ Einzelregeln mittels Komposition zusammenfügen.

7.3 Ergänzung

Für jeden der beiden skizzierten Alternativansätze ergeben sich durch die Wahlmöglichkeiten bei den Fuzzy-Operatoren (Mengenoperatoren bzw. t-Normen, Inferenzprinzip, Defuzzifizierungsstrategie, Gewichtung usw.) wiederum verschiedene Möglichkeiten. Je weniger man sich an vorhandenen Ansätzen (z.B. Mamdani) orientiert, desto mehr Freiheitsgrade kommen hinzu.

8 Komposition, Defuzzifizierung und Ergänzungen

Legt man Regeln der Struktur (2) zugrunde und berechnet das Ergebnis jeder Regel nach Gleichung 5, so ändert sich nach der zeitbezogenen Auswertung der Prämissenterme an Komposition und Defuzzifizierung durch die explizite Repräsentation der Zeitbezüge gegenüber dem Konzept von Mamdani nichts.

Als Komposition der Ergebnisse der Regeln $R_1 \ \ldots \ R_N$ ergibt sich für jeden Zeitpunkt t_s

$$\mu_K(t_s, y) = \max\{\mu_{C_{R_1}}(t_s, y) \ \ldots \ \mu_{C_{R_N}}(t_s, y)\}. \tag{8}$$

Die Defuzzifizierung geschieht anhand von $\mu_C(t_s, y)$ für den gewählten Zeitpunkt t_s. Im folgenden steht $\hat{y}(t)$ für den Schätzwert der Größe Y zur Zeit t, der aus $\mu_K(t_s, y)$ durch Defuzzifizierung berechnet wird. Entsprechend wird im folgenden die Schreibweise $\hat{x}(t)$ für einen Schätzwert von $x(t)$ verwendet.

8.1 Alternativansätze

Interessant im Zusammenhang mit dem ersten der vorgestellten Alternativansätze ist die Möglichkeit, auf Basis von Gleichung 7 nicht die Fuzzymengen zu einem bestimmten Zeitpunkt t_s für Komposition und anschließende Defuzzifizierung nach y zu verwenden, sondern direkt mit der durch die Gleichung definierten zweidimensionalen Fuzzymenge zu arbeiten.

Nach einer zweidimensionalen Überlagerung (Komposition) könnte man z.B. einen Wert vorgeben und nach der Zeit defuzzifizieren, was man als Schätzung des Auftrittszeitpunktes eines erwarteten Wertes deuten könnte. Beispielsweise könnte so der Augenblick des Überschreitens eines Schwellwertes bestimmt werden.

Bei Definition eines geeigneten zweidimensionalen Defuzzifizierungsverfahrens, das sowohl einen Zeitpunkt als auch einen Wert liefern würde, wäre es auch denkbar, einen allgemein möglichst verläßlichen Schätzwert zu erhalten. D.h. zu dem gefundenen Zeitpunkt wäre die dafür abgegebene Prognose wahrscheinlich ziemlich gut.

Einen interessanten Ansatz zur Rückkopplung der Ausgänge einer Fuzzyregelbasis *ohne* Defuzzifizierung findet man in [SKS97]. Durch den Verzicht auf eine Defuzzifizierung kann man gegenüber unserem Ansatz prinzipiell den bei der Reduktion der Fuzzymengen auf Schätzwerte auftretenden Informationsverlust vermeiden, jedoch ergeben sich daraus sehr harte Anforderungen sowohl an das Inferenzverfahren als auch an die verwendeten Fuzzymengen.

Der Ansatz sieht keine explizite Repräsentation zeitlicher Bezüge, insbesondere von Verzögerungszeiten, vor, sondern arbeitet streng taktweise, d.h. in jedem Schritt werden die aktuellen Eingangs-Fuzzymengen auf Fuzzymengen am Ausgang abgebildet, die wiederum einen Teil der Eingangs-Fuzzymengen für den nächsten Zeitschritt darstellen. Ein Rückgriff auf ältere Eingangs-Fuzzymengen findet nicht statt.

9 Wohldefiniertheit (Schätzung unbekannter Werte)

Die Verwendung einer Regelbasis mit Regeln der oben angegebenen Struktur zur Berechnung der Größe Y zu einem Zeitpunkt t_s ist, sofern die Werte aller Größen bis t_s bekannt sind, abgesehen von Effizienzgesichtspunkten (siehe unten) unproblematisch, da der Definitionsbereich von T auf den positiven Bereich beschränkt und damit die Prämisse $\mu_P(t_s)$ in Gleichung 5 jeweils für jede Regel wohldefiniert ist, so daß sich $\mu_K(t_s, y)$ berechnen läßt.

Anders sieht es aus, wenn die Regelbasis z.B. als Prozeßmodell (Verhaltensmodell) verwendet werden soll, um zu einem gegebenen Zeitpunkt t_r Voraussagen (Prognosen) über den Wert bestimmter Größen zu einem bestimmten Zeitpunkt t_s in der Zukunft zu treffen.

Problematisch ist in der Definition der Prämissenzugehörigkeit in Gleichung 4 der Term

$$\mu_{A\tilde{\wedge}T}(x(t_s - \Delta t), \Delta t) = \min\{\mu_A(x(t_s - \Delta t)),\ \mu_T(\Delta t)\},$$

falls $x(t_s - \Delta t)$ nicht bekannt ist. Der Term kann aber geschätzt werden, beispielsweise mit folgender Definition:

$$\hat{\mu}_{A\tilde{\wedge}T}(x(t_s - \Delta t), \Delta t)) := \begin{cases} \min\{\mu_A(x(t_s - \Delta t)),\ \mu_T(\Delta t)\}, & (a) \\ \quad \text{falls } x(t_s - \Delta t_i) \text{ bekannt}, & \\ 0, \text{ falls } \mu_T(\Delta t) = 0, & (b) \\ \min\{\mu_A(\hat{x}(t_s - \Delta t)), \mu_T(\Delta t)\}, & (c) \\ \quad \text{falls } \hat{x}(t_s - \Delta t) \text{ bekannt}, & \\ \text{undefiniert, sonst.} & (d) \end{cases} \quad (9)$$

(b) ist sinnvoll, da unter der Voraussetzung $\mu_T(\Delta t) = 0$ gilt $min\{\mu_A(x), \mu_T(\Delta t)\} = 0$ unabhängig von x.
(c) berücksichtigt die beste verfügbare Schätzung von x und ist deshalb sinnvoll, obwohl so nicht unbedingt die beste Schätzung für $\mu_P(t_s)$ bzw. $\mu_C(t_s, y)$ sichergestellt werden kann. Dies liegt daran, daß nur ein Schätzwert berücksichtigt wird und außerdem keinerlei Information über die Güte dieser Schätzung eingeht.
(d) bedeutet, daß keine Schätzung möglich ist. Auf diesen Fall wird im nachfolgenden Abschnitt 10 näher eingegangen.

Gegebenenfalls ist auch folgender einfacher Ansatz sinnvoll, der nicht auflösbare Beziehungen in Gleichung 4 einfach ausblendet, indem er nur bekannte Werte verwendet, so daß Regeln mit nicht definierten Prämissen gar nicht berücksichtigt werden:

$$\hat{\mu}_{A\tilde{\wedge}T}(x(t_s - \Delta t), \Delta t)) := \begin{cases} \min\{\mu_A(x(t_s - \Delta t)),\ \mu_T(\Delta t)\}, \\ \quad \text{falls } x(t_s - \Delta t_i) \text{ bekannt} \\ 0, \text{ sonst.} \end{cases}$$

Auch der Rückgriff auf Defaultwerte, z.B. den beobachteten Mittelwert einer Größe wäre denkbar.

Entsprechende Definitionen sind bei der Verwendung anderer Operatoren als der Konjunktion für $\tilde{\wedge}$ ebenfalls möglich.

10 Umsetzung unter Effizienzgesichtspunkten

Die folgenden Betrachtungen gehen davon aus, daß ausgehend von einem Zeitpunkt t_k innerhalb eines vorgegebenen Zeithorizontes $t_k \ldots t_{k+m}$ eine Prognose für eine gegebene Größe Y zu erstellen ist. Als Information stehen die zwischengespeicherten Werte der Größen aus dem Bereich von t_{k-l+1} bis t_k, dem „Gedächtnis“ des Systems, zur Verfügung. Die Prämissenberechnung soll auf Basis von Definition 9 erfolgen.

Zunächst ist festzustellen, daß effektive (nicht nur effiziente!) Berechnungen überhaupt nur dann möglich sind, wenn der bei jeder Prämisse bzw. Konklusion betrachtete Zeitbereich endlich ist, d.h. wenn $\mu_T(t)$ nur zu endlich vielen Zeitpunkten von Null abweicht. Eine Berechnung von Konklusion bzw. Prämisse nach Gleichung 5 bzw 4 kann sich dann auf diesen endlichen Zeitbereich beschränken.

Außerdem müssen alle direkt oder indirekt zur Berechnung oder Schätzung von Werten notwendigen Informationen innerhalb des „Gedächtnisses“ des Systems vorhanden sein. Das bedeutet, daß für jeden in einer Regel auftretenden zeitbehafteten Fuzzy-Term gelten muß, daß für jedes Δt entweder $\mu_T(\Delta t) = 0$ oder $t_{k-l+1} \leq t_k - \Delta t \leq t_k$.

Desweiteren ist offensichtlich, daß sich eine „direkte“ Umsetzung mittels eines rekursiven Ansatzes, der gemäß Gleichung 9 gegebenenfalls auf Schätzwerte $\hat{x}(t_s - \Delta t)$ zurückgreift und diese bei Bedarf berechnet, so daß Fall (d) in Definition 9 nicht auftritt, aus Gründen des Berechnungsaufwandes verbietet. Ein Lösungsansatz besteht in der Vermeidung von Mehrfachberechnungen von Schätzwerten, beispielsweise indem diese Werte in der richtigen zeitlichen Reihenfolge berechnet und im Sinne des dynamischen Programmierens (zwischen-) gespeichert werden, so daß bei Bedarf auf sie zurückgegriffen werden kann.

Ein weitergehender Weg zur Steigerung der Effizienz könnte in einer prinzipiell inkrementel-

len Berechnung von auf Schätzungen basierenden Werten bestehen. Im Vordergrund stände die Vermeidung einer kompletten Neuberechnung von $\mu_P(t_s)$ bzw. $\mu_C(t_s, y)$ nach Gleichung 4 bzw. 5 bei Eintreffen neuer Informationen, d.h. nach jedem Takt, durch Ausnutzung der Eigenschaften des Maximum-Operators.

11 Zusammenfassung und Ausblick

Die vorgeschlagene Modellstruktur ermöglicht den Umgang mit unscharfen Daten und die explizite Repräsentation zeitlicher Beziehungen im Rahmen eines verständlichen, regelbasierten Modells. Damit eignet sie sich zur Repräsentation von auf Basis maschineller Lernverfahren anhand von Zeitreihen erkannten Systemabhängigkeiten (vgl. etwa [Wei95]).

Die Art der Struktur ist dadurch motiviert, daß zwar eine Reihe von erfolgreichen oder zumindest erfolgversprechenden Arbeiten zur Erzeugung von Fuzzyregelbasen aus Daten und zu ihrer Anwendung zur Vorhersage von Größen dynamischer Systeme (z.B. [Bet96] oder [Kra94]) existiert, jedoch keiner der bekannten Ansätze (vgl. z.B. [Zim99]) eine wünschenswerte explizite Repräsentation zeitlicher Zusammenhänge ermöglicht. Dies ist eigentlich um so erstaunlicher, als sich Fuzzymengen für die Repräsentation unpräziser Zeitangaben geradezu aufdrängen.

Bei der Analyse prinzipieller Eigenschaften des Modells liegt der Schwerpunkt der Arbeit insbesondere auch auf einem Vergleich mit intuitiven Erwartungen an das Verhalten des Modells, die aufgrund der verbalen Formulierung der Regeln geweckt werden. Gleichzeitig werden Alternativansätze betrachtet, die in Abschnitt 7 angedeutet werden.

Wichtigstes langfristiges Forschungsthema ist die Entwicklung eines Systems zur maschinellen Generierung von Regelbasen aus Zeitreihen, das sich an der menschlichen Informationsverarbeitung orientiert [FK98] und auf dem in [KW92] beschriebenen Ansatz zur Heuristischen Modellierung beruht.

Literatur

[Bet96] Kurt Dirk Bettenhausen. *Automatische Struktursuche für Regler und Strecke.* TH Darmstadt, 1996.

[Dö76] Dietrich Dörner. *Problemlösen als Informationsverarbeitung.* Kohlhammer, Stuttgart, 1976.

[FK98] Andreas Fick and Hubert B. Keller. Konzeptbildung zur automatischen Modellierung dynamischer Systeme. In Ute Schmid, editor, *Workshop Maschinelles Lernen und Konzepterwerb, Bremen, 15.-17.9.1998*, pages 28–42. TU Berlin, 1998.

[Kel00] Hubert B. Keller. *Maschinelle Intelligenz.* Vieweg, 2000.

[KF98] Hubert B. Keller and Andreas Fick. Maschinelle Lernverfahren am Beispiel der thermischen Abfallbehandlung. *KI*, 1998.

[Kra94] M. Krabs. *Das ROSA-Verfahren zur Modellierung dynamischer Systeme mit statistischer Relevanzbewertung.* VDI Verlag, 1994.

[KW92] Hubert B. Keller and Thomas Weinberger. Heuristische Modellierung - ein Arbeiten mit Hypothesen. In *Workshop Modellierung und Simulation im Umweltbereich*, Rostock, 1992. Universität Rostock, Fb. Informatik.

[MA75] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy controller. *International Journal of Man Machine Studies*, 7:1–13, 1975.

[MK96] Bernd Müller and Hubert B. Keller. Neural networks for combustion process modelling. In *International Conference on Engeneering Applications of Neural Networks (EANN 96)*, 1996.

[SKS97] Elmar Schäfers, Volker Krebs, and Klaus Schmid. Inferenzverfahren für dynamische Fuzzy-Systeme. In *7. Workshop Fuzzy Control des GMA-UA 1.4.2*, pages 165–178, 1997.

[Sug85] M. Sugeno. An introductory survey of fuzzy control. *Information Sciences*, 36:59–83, 1985.

[Wei95] Thomas Weinberger. *Ein Ansatz zur Modellierung dynamischer Systeme.* VDI Verlag, 1995.

[Zad65] Lofti A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[Zim99] Hans-Jürgen Zimmermann, editor. *Proceedings / EUFIT '99 : 7th European Congress on Intelligent Techniques & Soft Computing ; Aachen, Germany, September 13-16, 1999*, Aachen , Mainz, 1999.

A Sequential Sampling Algorithm for a General Class of Utility Criteria

Tobias Scheffer and Stefan Wrobel
University of Magdeburg, FIN/IWS
P.O. Box 4120
39016 Magdeburg, Germany
scheffer, wrobel@iws.cs.uni-magdeburg.de

ABSTRACT

Many discovery problems, *e.g.,* subgroup or association rule discovery, can naturally be cast as n-best hypothesis problems where the goal is to find the n hypotheses from a given hypothesis space that score best according to a given utility function. We present a sampling algorithm that solves this problem by issuing a small number of database queries while guaranteeing precise bounds on confidence and quality of solutions. Known sampling algorithms assume that the utility be the average (over the examples) of some function, which is not the case for many frequently used utility functions. We show that our algorithm works for all utilities that can be estimated with bounded error. We provide such error bounds and resulting worst-case sample bounds for some of the most frequently used utilities, and prove that there is no sampling algorithm for another popular class of utility functions. The algorithm is sequential in the sense that it starts to return (or discard) hypotheses that already seem to be particularly good (or bad) after a few examples. Thus, the algorithm is often even faster than its worst-case bounds.

1. INTRODUCTION

Even with discovery algorithms optimized for very large data sets, for many application problems it is infeasible to process all of the given data. In this case, an obvious strategy is to use only a randomly drawn *sample* of the data. Clearly, if parts of the data are not looked at, it is impossible, in general, to guarantee that the results produced by the discovery algorithm will be identical to the results returned on the complete dataset. If the use of sampling is to be more than a practitioner's "hack", sampling must be combined with discovery algorithms in a fashion that allows us to give the user *guarantees* about how far the obtained results differ from the optimal (non-sampling based) results. The goal of a sampling discovery algorithm then is to guarantee this quality using the minimum amount of examples.

Existing research has concentrated primarily on discovery problems where the goal is to select from a space of possible hypotheses H one of the elements with maximal value of an *instance-averaging* utility function f, or all elements with an f-value above a user-given threshold (*e.g.,* all association rules with sufficient support) [5, 8]. With instance-averaging utility functions, the quality of a hypothesis h is the average across all instances in a dataset D of an instance utility function f_{inst}.

Many discovery problems, however, cannot easily be cast in this framework. Firstly, it is often more natural for a user to ask for the n best solutions instead of the single best or all hypotheses above a threshold (see *e.g.,* [20]). Secondly, many popular utility measures cannot be expressed as an averaging utility function. This is the case, *e.g.,* for all functions that combine coverage and distributional properties of a hypothesis, as popular in subgroup discovery. The task of subgroup discovery [12] is to find maximally general subsets of database transactions within which the distribution of a focused feature differs maximally from the default probability of that feature in the whole database. As an example, consider the problem of finding groups of customers who are particularly likely (or unlikely) to buy a certain product.

In this paper, we present a general sampling algorithm for the n-best hypotheses problem that works for any utility functions that can be estimated with bounded error. To this end, in Section 2, we first define the n-best hypotheses problem more precisely and identify appropriate quality guarantees. Section 3 then presents the generic algorithm. Our algorithm is a *sequential* sampling algorithm [19], in the sense that it does not wait for a fixed number of examples that can be guaranteed to suffice even in the worst case before starting the analysis. It starts to return (or discard) hypotheses that already seem to be particularly good (or bad) after a few examples. Thus, the algorithm is often faster than its worst-case bounds. In Section 4, we prove that many of the popular utility functions that have been used in KDD indeed can be estimated with bounded error, giving detailed bounds. For one popular class of functions that cannot be used by our algorithm, we prove that there cannot be a sampling algorithm at all. Our results thus also give an indication as to which of the large numbers of popular utility functions are preferable with respect to sampling. In Section 5, we evaluate our results and discuss their relation to previous work.

2. APPROXIMATING n-BEST HYPOTHESES PROBLEMS

In many cases, it is more natural for a user to ask for the n best solutions instead of the single best or all hypotheses above a threshold. Such n-best hypotheses problems can be stated more precisely as follows (adapted from [20], where this formulation is used for subgroup discovery): Let D be a database of instances, H a set of possible hypotheses, f a quality or utility function on H mapping a hypothesis and a database to a nonnegative number, and n, $1 \leq n \leq |H|$ an integer, the number of desired solutions. The *n-best hypotheses problem* is to find a set $G \subseteq H$ of size n such that there is no $h' \in H$: $h' \notin G$ and $f(h', D) > f_{min}$, where $f_{min} := min_{h \in G} f(h, D)$.

Whenever we use sampling, the above optimality property cannot be guaranteed, so we must find appropriate alternative guarantees. Since for n-best problems, the exact quality and rank of hypotheses is often not central to the user, it is sufficient to guarantee that G indeed "approximately" contains the n best hypotheses. We can operationalize this by guaranteeing that there will never be a non-returned hypothesis that is "significantly" better than the worst hypothesis in our solution. More precisely, we will use the following problem formulated along the lines of PAC (probably approximately correct) learning:

DEFINITION 1 (APPROXIMATE n-BEST HYPOTHESES). *Let D, H, f and n as in the preceding definition. Then let δ, $0 < \delta \leq 1$, be a user-specified confidence, and $\varepsilon \in \mathbb{R}^+$ a user-specified maximal error. The* approximate n-best hypotheses problem *is to find a set $G \subseteq H$ of size n such that, with confidence $1 - \delta$, there is no $h' \in H$: $h' \notin G$ and $f(h', D) > f_{min} + \varepsilon$, where $f_{min} := min_{h \in G} f(h, D)$.*

In other words, we want to find a set of n hypotheses such that, with high confidence, no other hypothesis outperforms any one of them by more than ε, where f is an arbitrary performance measure. In order to design an algorithm for this problem, we need to make certain assumptions about the utility function f. Ideally, an algorithm should be capable of working (at least) with the kinds of utility functions that have already proven themselves useful in practical applications. If the problem is to *classify* database items (*i.e.*, to find a total function mapping database items to class labels), *accuracy* is often used as utility criterion. For the discovery of *association rules*, by contrast, one usually relies on *generality* as primary utility criterion [1]. Finally, for subgroup discovery, it is commonplace to combine both generality and *distributional unusualness*, resulting in relatively complex evaluation functions (see, *e.g.*, [13] for an overview).

In light of the large range of existing and possible future utility functions, in order to avoid unduly restricting our algorithm, we will not make syntactic assumptions about f. In particular, unlike [5], we will not assume that f is a single probability nor that it is based on averages of instance properties. Instead, we only assume that it is possible to determine a function Δ for a particular f that bounds the probability of errors when computing f based on a sample, and vanishes with increasing sample sizes. As we will show in Section 4 below, finding such Δ is relatively straightforward for classification accuracy, and is also possible for all but one of the popular utility functions from association rule and subgroup discovery. More precisely, we define an error probability bound function Δ for f as follows.

DEFINITION 2 (ERROR PROBABILITY BOUND). *Let f be a utility function, let $h_1 \in H$ and $h_2 \in H$ be two hypotheses. Let $f_1 := f(h_1, D)$ denote the true utility of h_1 on the entire dataset, $\hat{f}_1 := f(h_1, S)$ its estimated utility computed based on a sample $S \subseteq D$ of size m ($f_2, \hat{f}_2$ defined analogously). Then $\Delta : \mathbb{N} \times \mathbb{R} \times \mathbb{R} \to [0, 1]$ is an* error probability bound *for f iff for any $\varepsilon, \hat{\varepsilon}$*

$$Pr_S[\hat{f}_1 - \hat{f}_2 > \hat{\varepsilon} | f_1 - f_2 \leq \varepsilon] \leq \Delta(m, \varepsilon, \hat{\varepsilon}). \quad (1)$$

Equation 1 says that Δ bounds the probability of drawing a sample S (when drawing m transactions independently and identically distibuted from D), such that the empirical difference between two utility values appears overly large. We will refer to $\hat{f}(h, S)$ as the *measured*, or *empirical* utility. If, in addition, for any $\delta, 0 < \delta \leq 1$ and any ε there is a number m such that $\Delta(m, \varepsilon, 0) \leq \delta$ we say that Δ *vanishes*. Note that it can be meaningful for ε to be negative; in this case, Δ bounds the chance that h_1 appears better on the sample ($\hat{f}_1 - \hat{f}_2$ positive) although h_2 really is better ($f_1 - f_2$ negative).

3. SAMPLING ALGORITHM

The general approach to designing a sampling algorithm is to use an appropriate error probability bound to determine the required number of examples for a desired level of confidence and accuracy. When estimating a single probability, Chernoff bounds that are used in PAC theory [10, 18] and many other areas of statistics and computer science can be used to determine appropriate sample bounds [17]. When such algorithms are implemented, the Chernoff bounds can be replaced by tighter normal or Student's t-distribution tables.

Unfortunately, the straightforward extension of such approaches to selection or comparison problems like the n-best hypotheses problem leads to unreasonably large bounds: to avoid errors in the worst case, we have to take very large samples to recognize small differences in utility, even if the actual differences between hypotheses to be compared are very large. This problem is addressed by *sequential* sampling methods [4, 19] (that have also been referred to as *adaptive* sampling methods [5]). The idea of sequential sampling is that when a difference between two frequencies is very large after only a few examples, then we can conclude that one of the probabilities is greater than the other with high confidence; we need not wait for the sample size specified by the Chernoff bound, which we have to when the frequencies are similar. Sequential sampling methods have been reported to reduce the required sample size by several orders of magnitude (*e.g.*, [7]).

In our algorithm (Table 1), we combine sequential sampling with the popular "loop reversal" technique found in many KDD algorithms. Instead of processing hypotheses one after another, and obtaining enough examples for each hypothesis

to evaluate it sufficiently precisely, we keep obtaining examples (step 2b) and apply these to all remaining hypotheses simultaneously (step 2c). This strategy allows the algorithm to be easily implemented on top of database systems (assuming they are capable of drawing samples), *and* enables us to reach tighter bounds. After the statistics of each remaining hypothesis have been updated, the algorithm checks whether it has seen enough examples to distinguish all the remaining good hypotheses from the bad ones with sufficient confidence, in which case it can exit (step 2f). Otherwise, in step 2g it checks all remaining hypotheses and (i) outputs those where it can be sufficiently certain that the number of better hypotheses is no larger than the number of hypotheses still to be found (so they can all become solutions), or (ii) discards those hypotheses where it can be sufficiently certain that the number of better other hypotheses is at least the number of hypotheses still to be found (so it can be sure the current hypothesis does not need to be in the solutions). Indeed it can be shown that this strategy leads to a total error probability less than δ as required.

THEOREM 1. *The algorithm will output a group G of exactly n hypotheses such that, with confidence $1-\delta$, no other hypothesis in H has a utility which is more than ε higher than the utility of any hypothesis that has been returned:*

$$Pr[\exists h \in H \setminus G : f(h) > f_{min} + \varepsilon] \leq \delta \qquad (2)$$

where $f_{min} = \min_{h' \in G}\{f(h')\}$; assuming that $|H| \geq n$.

The idea of the proof is that we have to sum up the probabilities that either one of the n best hypotheses is discarded or any significantly worse hypothesis is returned over all steps i. This sum must be no more than δ. Due to lack of space, we leave the proof for the full paper.

4. INSTANTIATIONS

In order to implement the algorithm for a given interestingness function we have to find a function $\Delta(m, \varepsilon, \hat{\varepsilon})$ that satisfies Equation 1 for that specific f. We will in the following present a list of Δ functions for the most commonly used interestingness functions. Table 2 summarizes our results and presents, for each studied utility function f, the error bound Δ and a corresponding worst-case bound on the required sample size. (Since the database is constant, we abbreviate $f(h, D)$ as $f(h)$.)

4.1 Instance-Averaging Functions

This simplest form of a utility function is the average, over all example queries, of some instance utility function $f_{inst}(h, q_i)$. The utility is then defined as $f(h) = \frac{1}{|D|}\sum_{i=1}^{D} f_{inst}(h, q_i)$ (the average over the whole database) and the estimated utility is $\hat{f}(h, Q_m) = \frac{1}{m}\sum_{i=1}^{m} f_{inst}(h, q_i)$ (average over the example queries). An easy example of an instance-averaging utility is the classifiation accuracy. Besides being potentially useful, this class of utility functions serves as an introducing example of how Δ functions can be derived. We assume that there is a lower bound $lb = \min_{q \in D, h \in H} f_{inst}(h, q_i)$ and an upper bound $ub = \max_{q \in D, h \in H} f_{inst}(h, q)$ for this function (*e.g.*, classification accuracy is bounded between 0 and 1) and we define $\Lambda = \max(f_{inst}(h_1, q) - f_{inst}(h_2, q)) - \min(f_{inst}(h'_1, q') - f_{inst}(h'_2, q'))$ as the range of possible values of measured performance differences.

Table 1: Sequential sampling algorithm for the n-best hypotheses problem
Input: *num* (number of desired hypotheses), ε and δ (approximation and confidence parameters). **Output:** *num* approximately best hypotheses (with confidence $1-\delta$).

1. **Let** $n = num$ (n counts the number of hypotheses that we still need to find) and **Let** $H_1 = H$ (the set of hypotheses that have, so far, neither been discarded nor accepted). **Let** $Q_1 = \emptyset$ (no sample drawn yet).
2. **For** $i = 1 \ldots \infty$
   (a) **Let** $H_{i+1} = H_i$.
   (b) Query a random item of the database q_i. **Let** $Q_i = Q_{i-1} \cup \{q_i\}$.
   (c) Update the empirical utility $\hat{f}$ of the hypotheses in H_i.
   (d) **Let** h_n be the hypothesis in H_i that achieves the nth highest empirical utility $\hat{f}$.
   (e) **Let** h_{n+1} be the hypothesis in H_i that achieves the $n+1$st highest empirical utility $\hat{f}$.
   (f) **If** $\Delta(i, -\varepsilon, 0) \leq \frac{2\delta}{3|H_i|^2}$ **Then Exit** (the for loop).
   (g) **For** $j = 1 \ldots |H_i|$
      i. **If** $\hat{f}(h_j, Q_i) > \hat{f}(h_{n+1}, Q_i)$ (h_j appears good) **and** $n > 0$ **and** $\Delta\left(i, -\varepsilon, \hat{f}(h_j, Q_i) - \hat{f}(h_{n+1}, Q_i)\right) \leq \frac{4\delta}{|H_i|^2 i^2 \pi^2}$ **Then Output** hypothesis h_j and then **Delete** h_j from H_{i+1} and decrement n.
      ii. **If** $\hat{f}(h_j, Q_i) < \hat{f}(h_n, Q_i)$ (h_j appears poor) **and** $|H_i| > n$ **and** $\Delta\left(i, 0, \hat{f}(h_n, Q_i) - \hat{f}(h_j, Q_i)\right) \leq \frac{4\delta}{|H_i|^2 i^2 \pi^2}$ **Then Delete** h_j from H_{i+1}.
   (h) **If** $n = 0$ **Or** $|H_{i+1}| = n$ **Then Exit** (the For loop).
3. **Output** the n hypotheses from H_i which have the highest empirical utility.

$\hat{f}(h_1, Q_i) - \hat{f}(h_2, Q_i)$ is a random variable with mean value $f(h_1) - f(h_2)$ and bounded range Λ. We can use the Hoeffding inequality [9] to bound the chance that an arbitrary (bounded) random variable takes a value which is far away from its mean value. When X is a random variable with expectation $E(X)$ and range at most Λ and the sample size is m, then the Hoeffding inequality guarantees that $Pr[X - E(X) > \epsilon] \leq \exp\{-2m\frac{\epsilon^2}{\Lambda^2}\}$. In our situation, this implies Equation 3

$$Pr[\hat{f}(h_1) - \hat{f}(h_2) > \hat{\varepsilon} | \bar{f}(h_1) - \bar{f}(h_2) \leq \varepsilon] \leq \exp\left\{-2m\frac{(\hat{\varepsilon} - \varepsilon)^2}{\Lambda^2}\right\}. \qquad (3)$$

Table 2: Summary of Instantiations

$f(h)$	$\Delta(m, \varepsilon, \hat{\varepsilon})$	w/c bound on m
instance-averaging	$\exp\left\{-2m\frac{(\hat{\varepsilon}-\varepsilon)^2}{\Lambda^2}\right\}$	$\frac{\Lambda^2}{\varepsilon^2}\log\frac{\sqrt{3}\lvert H_i\rvert}{\sqrt{2\delta}}$
$g(p-p_0)$, $g\lvert p-p_0\rvert$, $g\frac{1}{c}\sum_{i=1}^{c}\lvert p_i-p_{0_i}\rvert$	$4\exp\left\{-m\frac{(\hat{\varepsilon}-\varepsilon)^2}{8}\right\}$	$\frac{16}{\varepsilon^2}\log\frac{\sqrt{6}\lvert H_i\rvert}{\sqrt{\delta}}$
$g^2(p-p_0)$, $g^2\lvert p-p_0\rvert$, $g^2\frac{1}{c}\sum_{i=1}^{c}\lvert p_i-p_{0_i}\rvert$, $g^2(p-p_0)^2$, $g^2\frac{1}{c}\sum_{i=1}^{c}(p_i-p_{0_i})^2$	$4\exp\left\{-2m\left(\sqrt{1+\frac{\lvert\hat{\varepsilon}-\varepsilon\rvert}{4}}-1\right)^2\right\}$	$\frac{4}{\left(\sqrt{4+\varepsilon}-2\right)^2}\log\frac{\sqrt{6}\lvert H_i\rvert}{\sqrt{\delta}}$
$\sqrt{g}(p-p_0)$, $\sqrt{g}\lvert p-p_0\rvert$, $\sqrt{g}\frac{1}{c}\sum_{i=1}^{c}\lvert p_i-p_{0_i}\rvert$	$4\exp\left\{-m\frac{(\hat{\varepsilon}-\varepsilon)^4}{128}\right\}$	$\frac{256}{\varepsilon^4}\log\frac{\sqrt{6}\lvert H_i\rvert}{\sqrt{\delta}}$
$\frac{g}{1-g}(p-p_0)^2$, $\frac{g}{1-g}\frac{(p-p_0)^2}{p_0}$, $\frac{g}{1-g}\cdots$	1	∞

We can therefore define Δ as in Equation 4.

$$\Delta(m, \varepsilon, \hat{\varepsilon}) \quad = \quad \exp\left\{-2m\frac{(\hat{\varepsilon}-\varepsilon)^2}{\Lambda^2}\right\}. \qquad (4)$$

The algorithm exits the for loop (at latest) when $\Delta(m, -\varepsilon, 0) \leq \frac{2\delta}{3|H_i|^2}$. This is the case with certainty when $m \geq \frac{\Lambda^2}{\varepsilon^2}\log\frac{\sqrt{3}|H_i|}{\sqrt{2\delta}}$ (the proof is left for the full paper). But note that our algorithm will generally terminate much earlier; firstly, because we use the t-distribution (for large m) rather than the Hoeffding bound and, secondly, our sequential sampling approach will terminate much earlier when the n best hypotheses differ considerably from many of the "bad" hypotheses. The worst case occurs only when all hypotheses in the hypothesis space are equally good which makes it much more difficult to identify the n best ones.

4.2 Other Utility Functions

Often the task of data mining problems is to identify sets of transactions that are both frequent (*i.e.,* general) and statistically unusual. We define the generality g as the probability that a transaction lies within the support of a hypothesis (*i.e.,* the hypothesis applies to the transaction). A number of utility functions have been proposed that measure these two properties of hypotheses. We refer the reader to [11] for a discussion on the background of these utility functions.

One class of utility functions weights the generality g of a subgroup and the deviation of the probability p of a certain feature from the default probability p_0 equally [16]. Hence, these functions multiply generality and distributional unusualness of subgroups. Alternatively, we can use the absolute distance $|p-p_0|$ between probability p and default probability p_0. The multi-class version of this function is $g\frac{1}{c}\sum_c |p_i - p_{0_i}|$ where p_{0_i} is the default probability for class i. The appropriate definition of Δ as well as the resulting worst-case sample bounds can be found in Table 2.

Squared terms [20] are introduced to put more emphasis on either the generality or the difference between p and the default probability. The resulting utility functions are variations of $g^2(p-p_0)$.

The Binomial test heuristic [11] is based on elementary considerations. Suppose that the probability p is really equal to p_0 (*i.e.,* the corresponding subgroup is really uninteresting). How likely is it, that the subgroup with generality g displays a frequency of $\hat{p}$ on the sample Q with a greater difference $|\hat{p}-p_0|$? For large $|Q|\times g$, $(\hat{p}-p_0)$ is governed by the normal distribution with mean 0 and variance at most $\frac{1}{2\sqrt{m}}$. The probability density function of the normal distribution is monotonic, and so the criterion $\sqrt{m}(p-p_0)$ (which is $\sqrt{g}(p-p_0)$ times a constant factor) orders the hypotheses according to the probability that they are uninteresting. Several variants of this utility function have been used. See Table 2 for the results.

4.3 Negative Result

Several independent impurity criteria have led to utility functions which are equivalent (up to a constant factor) to $f(h) = \frac{g}{1-g}(p-p_0)^2$; *e.g.,* Gini diversity index, twoing criterion [3], and the chi-square test [16]. The order which this criterion imposes on hypotheses is also equal to the order imposed by the criterion of Inferrule [2]. Unfortunately, this utility function is not bounded and a few examples that have not been included in the sample can impose dramatic changes on the values of this function.

THEOREM 2. *There is no algorithm that satisfies Theorem 1 when* $f(h) = \frac{g}{1-g}(p-p_0)^2$.

The idea of the proof is that $(\hat{f}(h_1, Q) - \hat{f}(h_2, Q)) - (f(h_1) - f(h_2))$ is unbounded for any finite m. This implies that, even after an arbitrarily large sample has been observed (that is smaller than the whole database), the utility of a hypothesis with respect to the sample can be arbitrarily far from the true utility. But one may argue that demanding $\hat{f}(h, Q)$ to be within an additive constant ε is overly restricted. However, the picture does not change when we

require $\hat{f}(h, Q)$ only to be within a multiplicative constant, since $(\hat{f}(h_1, Q) - \hat{f}(h_2, Q))/(f(h_1) - f(h_2))$ is unbounded for any finite m as well.

5. DISCUSSION

Learning algorithms which require a number of examples that can be guaranteed to suffice for finding a nearly optimal hypothesis even in the worst case have early on been criticized as being impractical. Maron, Moore, & Lee [14, 15] have introduced sequential sampling techniques [4, 19] into the machine learning context by proposing the "Hoeffding Race" algorithm that combines loop-reversal with adaptive Hoeffding bounds. A general scheme for sequential local search has been proposed by Greiner [8]. Sequential sampling can often reduce the required sample sizes in cases by considerable factors [7].

Sampling techniques are particularly needed in the context of knowledge discovery in databases where often much more data are available than can be processed. A non-sequential sampling algorithm for KDD has been presented by Toivonen [17]; a sequential algorithm (that imposes further restrictions on f and possesses an additional parameter) by Domingo *et al.* [5, 6].

So far, all sampling algorithms have been restricted to instance-averaging utility functions (such as error probabilities), and to finding a single approximately best hypothesis. For the subgroup discovery problem, utility functions are used that combine generality and a distributional property of the hypothesis; this cannot be expressed as an instance-averaging function. Also, users might often be interested in the n best hypotheses. We presented an algorithm that works for a wide range of utility functions. For the only widely used function for which our algorithm does not work $(g/(1-g)\ldots)$ we proved that there exists no sampling algorithm at all.

By giving worst-case bounds on the sample size (and proving that there is no sampling algorithm for some utility functions) our results give an indication as to which of the many utility functions appear preferable from a sampling point of view.

Acknowledgement

We wish to thank Frank Schulz for carefully proof-reading the paper and giving us helpful comments.

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference on Management of Data*, pages 207–216, 1993.

[2] R. Uthurusamy an U. Fayyad and S. Spangler. Learning useful rules from inconclusive data. In *Knowledge Discovery in Databases*, pages 141–158, 1991.

[3] L. Breiman, J Friedman, R. Ohlsen, and C. Stone. *Classification and Regression Trees*. Pacific Grove, 1984.

[4] H. Dodge and H. Romig. A method of sampling inspection. *The Bell System Technical Journal*, 8:613–631, 1929.

[5] C. Domingo, R. Gavelda, and O. Watanabe. Practical algorithms for on-line selection. In *Proc. International Conference on Discovery Science*, pages 150–161, 1998.

[6] C. Domingo, R. Gavelda, and O. Watanabe. Adaptive sampling methods for scaling up knowledge discovery algorithms. Technical Report TR-C131, Dept. de LSI, Politecnica de Catalunya, 1999.

[7] R. Greiner and R. Isukapalli. Learning to select useful landmarks. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B:473–449, 1996.

[8] Russell Greiner. PALO: A probabilistic hill-climbing algorithm. *Artificial Intelligence*, 83(1–2), July 1996.

[9] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[10] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[11] W. Klösgen. Problems in knowledge discovery in databases and their treatment in the statistics interpreter explora. *Journal of Intelligent Systems*, 7:649–673, 1992.

[12] W. Klösgen. Assistant for knowledge discovery in data. In P. Hoschka, editor, *Assisting Computer: A New Generation of Support Systems*, 1995.

[13] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In Fayyad et al., editor, *Advances in Knowledge Discovery and Data Mining*, pages 249–271. AAAI, 1996.

[14] O. Maron and A. Moore. Hoeffding races: Accelerating model selection search for classification and function approximating. In *Advances in Neural Information processing Systems*, pages 59–66, 1994.

[15] A. Moore and M. Lee. Efficient algorithms for minimizing cross validation error. In *ICML-94*, pages 190–198, 1994.

[16] G. Piatetski-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248, 1991.

[17] H. Toivonen. Sampling large databases for association rules. In *Proc. VLDB Conference*, 1996.

[18] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1996.

[19] A. Wald. *Sequential Analysis*. Wiley, 1947.

[20] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proc. First European Symposion on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87, Berlin, 1997.

Generalizing Recursive Program Schemes with Anti-Unification (Abstract)

Ute Schmid, Uwe Sinha, and Fritz Wysotzki
Methods of Artificial Intelligence, Computer Science Department
Sekr. Fr 5–8, Technical University Berlin, Franklinstr. 28/29, D-10587 Berlin, Germany
Email: {schmid, uwesinha, wysotzki}@cs.tu-berlin.de

Program or algorithm schemes play an important role in deductive as well as inductive program synthesis. For example, in the KIDS system [Smi90], schemes for divide-and-conquer, local, and global search algorithms are used for deriving Lisp programs from specifications. Flener [Fle95] proposes an approach to inductive synthesis of Prolog programs restricted by algorithm schemes. Typically, such schemes are provided by a human expert. We are interested in automatic acquisition of schemes from example programs.

A scheme represents the abstract or conceptual structure of an algorithm, i. e., a class of concrete programs. It is defined over (higher order) variables which are place-holders for concrete objects (constants), functions, and predicates. During scheme-guided program synthesis, these variables are instantiated, that is, the abstract scheme is transformed into a concrete program. We present an approach to the acquisition of schemes based on anti-unification of pairs of functions.

Recursive Program Schemes

Our work is in the context of inductive synthesis of functional recursive programs [SW98]. A program is represented as recursive program scheme (RPS), defined over some arbitrary term algebra *M(V, $F \cup \Phi$)* with V as set of variables, F as set of function symbols, and Φ as set of function variables (names for user-defined functions). As usual in functional programming, predicates are considered as boolean functions.

Definition 1 (RPS). *An RPS is a pair $\langle \Sigma, t \rangle$ with t as initial function call ("main program") and Σ as set of user-defined functions ("sub-programs") with $\Sigma = \{G_i(x_1, \ldots, x_n) = t_i \mid 1 \leq i \leq m\}$, with $n, m \in \mathcal{N}$, $G_i \in \Phi$, $x_1, \ldots x_n \in V$, and $t_i \in M(V, F \cup \Phi)$.*

The left-hand side of an equation in Σ corresponds to a function head, the right-hand side to a function body. If a term t_i contains G_i at least once, the function is recursive.

Since program terms are defined over function *symbols*, in general, an RPS represents a class of programs. In the following we assume a fixed interpreter function, associating each function symbol in F with a primitive Lisp function – that is, an RPS defined over a set of function symbols corresponds to a concrete Lisp program. Abstract schemes are obtained by generalizing over such concrete RPSs, replacing primitive function symbols by function variables χ. Please note, that these function variables do *not* correspond to names user-defined functions Φ!

Currently, our generalization approach is restricted to RPSs where Σ contains only a single equation and where t consists only of the call of this equation.

Restricted 2nd Order Anti-Unification

To capture as much of the common structure of two programs as possible in an abstract scheme we need at least second order (function) mapping. In contrast to first order approaches, higher order anti-unification (as well as unification) in general has no unique solution and cannot be calculated efficiently. Therefore, we developed a very restricted second order algorithm. Based on the results obtained with this algorithm, we plan careful extensions, maintaining uniqueness and efficiency. A more powerful approach to second order anti-unification was proposed for example by [Has95].

Our simple algorithm is presented in table 1. Details, proofs of uniqueness, correctness, and termination, are given in [Sin00].

Input are two program terms, output is the generated anti-instance together with a set of unique term-substitutions. First order term-substitutions were introduced by [IA93] to allow re-construction of the original instances t_1, t_2. For our algorithm, term-substitution are constructed over first-order variables y as well as over second order (function) variables χ.

The algorithm presupposes that all functions have a fixed number of arguments. For example, the addition of numbers is restricted to two arguments (`(+ x y)`), addition of more elements must be realized by nesting

Table 1 A Simple AU-Algorithm

Function Call: $\mathbf{au}(t_1, t_2)$ with terms $t_1, t_2 \in M(V, F \cup \Phi)$
Initialization: term-substitutions $\varphi = \emptyset$
Rules:

- **Same-Term**: $\mathbf{au}(t, t) = t$
- **Var-Term**: $\mathbf{au}(t_1, t_2) = y$ with $\varphi = \varphi \circ [t_1, t_2/y]$ (where either $t_1 \in V$ and $t_2 \in M(V, F \cup \Phi)$ or $t_1 \in M(V, F \cup \Phi)$ and $t_2 \in V$
- **Same-Function**: $\mathbf{au}(f(v_1, \ldots, v_n), f(v'_1, \ldots, v'_n)) = f(\mathbf{au}(v_1, v'_1), \ldots, \mathbf{au}(v_n, v'_n))$
- **Same-Arity**: $\mathbf{au}(f(v_1, \ldots, v_n), g(v'_1, \ldots, v'_n)) = \chi(\mathbf{au}(v_1, v'_1), \ldots, \mathbf{au}(v_n, v'_n))$ with $\varphi = \varphi \circ [f, g/\chi]$
- **Diff-Arity**: $\mathbf{au}(f(v_1, \ldots, v_n), g(v'_1, \ldots, v'_m)) = y$ with $\varphi = \varphi \circ [f(v_1, \ldots, v_n), g(v'_1, \ldots, v'_m)/y]$ (where $n \neq m$)

(e. g., `(+ x (+ y z))`). Functions with different numbers of arguments are currently not treated (rule "Diff-Arity" reduces such terms to a first order variable).

Examples

For RPSs consisting only of a function call involving a single (recursive) equation, it is sufficient to anti-unify the recursive equations. Because our algorithm is defined for prefix-notation, following the syntactic conventions of Lisp, recursive equations $G_i(x_1, \ldots, x_n = t_i)$ are presented as `(= (Gi x1 ... xn) ti)`.

Table 2 gives some abstract schemes obtained with our algorithm. The results are edited for better readability. In every example, we used a function for calculating the factorial of a number as first instance. If *factorial* is anti-unified with *sum*, the resulting scheme preserves the complete structure of both programs, as should be expected for syntactically isomorph terms. A similar result is obtained for incrementing the elements of a list. If *factorial* is anti-unified with a program for calculating the square of a number, the resulting scheme abstracts from the first argument of the "else"-case, which is a "constant" for factorial and a more complex term for the *sqr*. Anti-unification of *factorial* and *fibonacci* returns a very general scheme because the programs have very different structures: one versus two conditionals and a linear recursion versus a tree-recursion. The same is true for functions with different numbers of parameters, as *factorial* and *mult*.

Extensions

Anti-unification can be used not only for generalization learning, but additionally as a method for retrieval of a suitable source problem in the context of programming by analogy. The current problem is anti-unified with each potential source problem. The most specific element of the subsumption hierarchy of the anti-instances is returned as the most suitable source.

Table 2 Example Generalizations

t_1:	fac(x) =	if(eq0(x),	1,	*(x, fac(p(x))))
t_2:	sum(z) =	if(eq0(z),	0,	+(z, sum(p(z))))
	$\chi_1(y_1)$ =	if(eq0(y_1),	y_2,	$\chi_2(y_1, \chi_1(\text{p}(y_1)))$
t_1:	fac(x) =	if(eq0(x),	1,	*(x, fac(p(x))))
t_2:	incl(l) =	if(null(l),	nil,	cons(s(car(l)), incl(cdr(l))))
	$\chi_1(y_1)$ =	if($\chi_2(y_1)$,	y_2,	$\chi_3(y_3, \chi_1(\chi_4(y_1)))$
t_1:	fac(x) =	if(eq0(x),	1,	*(x, fac(p(x))))
t_2:	sqr(z) =	if(eq0(z),	0,	+(+(z, p(z)), sqr(p(z))))
	$\chi_1(y_1)$ =	if(eq0(y_1),	y_2,	$\chi_2(y_3, \chi_1(\text{p}(y_1)))$
t_1:	fac(x) =	if(eq0(x),	1,	*(x, fac(p(x))))
t_2:	fib(z) =	if(eq0(z),	0,	if(eq0(p(z)), 1, +(fib(p(z)), fib(p(p(z))))))
	$\chi_1(y_1)$ =	if(eq0(y_1),	y_2,	y_3)
t_1:	fac(x) =	if(eq0(x),	1,	*(x, fac(p(x))))
t_2:	mult(u v) =	if(eq0(u),	0,	+(v, mult(p(u), v)))
	y_1 =	if(eq0(y_2),	y_3,	$\chi_1(y_4, y_5)$)

That is, retrieval can be based on qualitative information instead of a similarity measure as typically used in case-based reasoning [Pla95].

Our algorithm can deal with concrete RPSs (defined over primitive functions) and abstract RPSs (containing object and function variables) in an uniform way. That is, a hierarchy of schemes on different levels of abstraction can be generated, resulting in a structured and efficient memory organization.

Finally, the algorithm can be extended to include semantic information as types [Sin00] and function evaluation [KW95].

References

[Fle95] P. Flener. *Logic Program Synthesis from Incomplete Information.* Kluwer, 1995.

[Has95] R. W. Hasker. *The Replay of Program Derivations*. PhD thesis, Univ. of Ill., Urbana-Cham., 1995.

[IA93] P. Idestam-Almquist. Generalization under implication by recursive anti-unification. In *Proc. of ICML93*, pp. 151–158. Morgan Kaufmann, 1993.

[KW95] T. Kolbe and C. Walther. Second-order matching modulo evaluation: A technique for reusing proofs. In *Proc. of the 14th IJCAI*, pp. 190–195. Morgan Kaufmann, 1995.

[Pla95] E. Plaza. Cases as terms: A feature term approach to the structured representation of cases. In *Proc. of ICCBR-95*, pp. 265–276. Springer, 1995.

[Sin00] U. Sinha. Gedächtnisorganisation und Abruf von rekursiven Programmschemata beim analogen Programmieren durch typindizierte Anti-Unifikation. Diplomarbeit, FB Informatik, TU Berlin, 2000.

[Smi90] D. R. Smith. KIDS: A semiautomatic program development system. *IEEE Transactions on Software Engineering*, 16(9):1024–1043, 1990.

[SW98] U. Schmid and F. Wysotzki. Induction of recursive program schemes. In *Proc. of ECML-98*, pp. 214–225. Springer, 1998.

Research Issues for Open Adaptive Hypermedia Systems

Nicola Henze

Universität Hannover
henze@kbs.uni-hannover.de

Adaptive hypermedia courseware benefits from being distributed over the Web: content can always be kept up-to-date, discussions and interactions between instructors and learners can be supported, new courses can easily be distributed to the students. Nevertheless, adaptive hypermedia systems are - even in the web content - still stand-alone systems as long as they lack the ability to integrate and adapt information from arbitrary places in the web. Within our KBS Hyperbook project [1], we are working on concepts and techniques for building adaptive hypermedia systems which are open, e.g. which are able to integrate distributed information resources [2]. In this talk, we discuss advantages and problems of creating an open, adaptive hypermedia system. We present our approach for building an open courseware system and show how we used it to implement an undergraduate course about Java programming. We discuss our experience with these courseware systems and show research issues which need to be addressed in the area of open adaptive systems which might benefit from applying machine learning methods.

[1] Nicola Henze, Wolfgang Nejdl and Martin Wolpers: Modeling Constructivist Teaching Functionality and Structure in the KBS Hyperbook System. Computer Supported Collaborative Learning Conference (CSCL'99), Stanford, CA, USA, December 12-15, 1999.

[2] N. Henze and W. Nejdl: "Extendible Adaptive Hypermedia Courseware: Integrating Different Courses and Web Material". International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000), 28-31 August 2000, Trient, Italy.

Active Learning – ein Ansatz zur Modellierung des Grammatiklernens in Fremdsprachen

Werner Dilger, Jens Zeidler,
Technische Universität Chemnitz,
Fakultät für Informatik,
Professur Künstliche Intelligenz,
09107 Chemnitz,
Email: {wdi,jzei}@informatik.tu-chemnitz.de

Zusammenfassung

Dieser als Arbeitspapier gedachte Artikel beschreibt die Ansätze zur Modellierung und Simulation des Lernverhaltens eines Nutzers der *Internet Grammar*. Relevante Begriffe, die sowohl bei sprachwissenschaftlichen Untersuchungen als auch auf dem Gebiet des Maschinellen Lernen (ML) vorkommen, werden verglichen und entsprechend angepasst. Speziell wird die Modellierung mit *Active Learning*-Methoden betrachtet. Nachfolgend soll dieses Verfahren für die Simulation aufbereitet werden. Mittels des von uns entwickelten Modells werden im späteren Verlauf der Arbeiten die Erkenntnisse aus den sprachwissenschftlichen Arbeiten überprüft. Die Simulation des Grammatiklernens in Fremdsprachen soll Bestandteil einer Art „Künstlicher Internetnutzer“ werden.

1 Einführung

An der Technischen Universität Chemnitz wurde 1998 eine von der Deutschen Forschungsgemeinschaft (DFG) geförderte Forschergruppe „Neue Medien im Alltag: Von individueller Nutzung zu soziokulturellem Wandel“ installiert. Insgesamt sind daran derzeit neun Projekte aus der Philosophischen Fakultät und der Fakultät für Informatik beteiligt. Die Philosophische Fakultät ist durch die Fachgebiete Allgemeine Sprachwissenschaft, Amerikanistik, Anglistik, Germanistik, Musikwissenschaften, Psychologie, Sozialforschung und Soziologie repräsentiert.

Die Fakultät für Informatik wird durch zwei Projekte der Arbeitsgruppe Künstliche Intelligenz vertreten. Im Projekt „Modellierung und Simulation der Rezeption textuell repräsentierter Inhalte im Internet“ (siehe [2] und [8]) geht es um die Schaffung eines Modells für einen Internetnutzer, welches sich zur Simulation auf dem Computer eignet. Die Idee ist, die Zusammenarbeit mit den Projekten aus der Psychologie und der Sprachwissenschaft zu nutzen, um ein Nutzermodell auf der Basis der dort vorhandenen Ansätze aufzubauen un dieses dann als „Künstlichen Internetnutzer“ zu simulieren. Unsere Simulation soll im Wesentlichen ein Nutzermodell verwalten und durch bekannte Lernmethoden das Modell ständig an Veränderungen anpassen können. In diesem Arbeitspapier geht es speziell um die Modellierung der sprachwissenschaftlichen Ansätze.

Kernstück der Arbeiten in der englischen Sprachwissenschaft ist das Programm *Internet Grammar*. Um die Grundlagen der Aneignung der englischen Grammatik, wie sie sich die Kollegen in der Anglistik vorstellen, soll es im Abschnitt 2 gehen. Abschnitt 3 beschreibt das *Active Learning*, welches nicht nur aus dem ML-Umfeld bekannt ist. Das Prinzip des *Active Learnings* bildet dann die Grundlage für ein neues Modells eines Grammatiklerners bei Fremdsprachen (siehe Abschnitt 4). Schließlich wird in Abschnitt 5 ein Blick auf Probleme und weitere Arbeiten zur Umsetzung und Erprobung des Modells geworfen.

2 Die Internetgrammatik

2.1 Allgemeines

Das Teilprojekt „Lernerverhalten in der Internetgrammatik“[1] wird durch die englischen Sprachwissenschaftler so motiviert:

> „Die neuen Medien, vor allem das Internet mit seinen vielfältigen Möglichkeiten zur Interaktion, eröffnen eine neue Dimension von Lerneruntersuchungen. In jüngster Vergangenheit sind die Angebote an WWW-basierten Lehrmaterialien aller Arten sprunghaft angestiegen; ihre Wirkungen blieben bisher im Großen und Ganzen unerforscht. Nachdem die durch die Kontrastive Analyse in den sechziger Jahren aufgekommenen Hoffnungen auf bessere Lehrerfolge vor allem bei grammatischen Strukturen des Englischen ausblieben, wird nun eine neue lernerzentrierte Betrachtungsweise von Sprachwissenschaftlern, Fachdidaktikern und Lehrern, auch in sog. Selbstlernzentren, angestrebt. Ausgangspunkt für Veränderungen im Lehrangebot und im Lehrprozess waren bisher vor allem auftretende Fehler der Lerner; die Frage nach dem Wie des Lernens wurde nur selten gestellt. Es ist nur sehr vereinzelt gelungen, von reinen Lernerfolgsmessungen abzugehen und das Verhalten der Lerner zu untersuchen. Die Entwicklung einer Internetgrammatik soll uns ermöglichen, sowohl dieses Wie zu erforschen als auch die Wirkung eines interaktiven Lehrwerks zu untersuchen.“

2.2 Die Untersuchungen

Die Sprachwissenschaftler stellen sich die Frage, wie sich verschiedene Gruppen von Internetnutzern verhalten, die Teile der englischen Grammatik mit Hilfe einer kontrastiven und interaktiven Internetgrammatik erlernen wollen. Dafür wurde ein WWW-Lehrwerk entwickelt[2], das mit einem „Tracking“-Mechanismus versehen ist. Der Lerner wird beobachtet, während er von einer Webseite zur anderen über Hypertextverbindungen wechselt, die einen beliebigen Pfad durch das Material erlauben. Das (streng anonym gehaltene) Aufzeichnen von solchen Daten wird mit den Ergebnissen von Befragungen verglichen, um verschiedene Lernstrategien und Lernertypen zu identifizieren.

2.3 Die Vorgehensweise

Dazu führen die Kollegen aus dem Bereich Englische Sprachwissenschaft aus:

> „Die Internetgrammatik wird so aufgebaut, daß wir alle ‚Bewegungen‘ innerhalb dieser Lernumgebung genau nachvollziehen können. Dies wird uns ermöglichen, einer Verbindung zwischen Lernstil und Lernerfolg nachzugehen. Das Lehrwerk selbst wird in drei Versionen angeboten – für Studierende, LehrerInnen und SprachwissenschaftlerInnen – wobei die Darstellungen sich sowohl im Stil als auch in den inhaltlichen Schwerpunkten unterscheiden. Jeder Teil besteht aus drei Komponenten, zwischen denen der Lerner sich frei bewegen kann: Die induktive Komponente bietet dem Lerner die Möglichkeit, anhand einer Datenbank von ausgewählten authentischen Sprachbeispielen zu einer möglichen Regel zu gelangen. Die Beispiele werden dem Chemnitzer Übersetzungscorpus entnommen und stellen englische Sätze und ihre deutschen Äquivalente (und umgekehrt) dar. Die deduktive Komponente besteht aus einer Anleitung mit formulierten Grammatikregeln und anschließenden (typischen, authentischen) Beispielen aus derselben Datenbank. Die dritte Komponente ist eine Sammlung von Übungen unterschiedlicher Art und Schwierigkeit. Alle Bewegungen zwischen den Komponenten werden aufgezeichnet und lernertypspezifisch analysiert.“

3 Active Learning

Wie sieht ein Modell des Nutzers der Internetgrammatik aus? Um die Ansätze zu formulieren, bietet das Maschinelle *Lernen* zunächst

[1] www.tu-chemnitz.de/phil/NeueMedien/Schmied.html
[2] www.tu-chemnitz.de/phil/InternetGrammar/

einen Anknüpfungspunkt allein vom Begriff her. Menschliches Sprachlernen und Maschinelles Lernen sind natürlich zwei beträchtlich vershiedene Dinge. Aber eine Verbindung zwischen beiden wird durch eine Begriff hergestellt, der in den unterschiedlichsten Forschungsgebieten auftaucht, aber gerade in letzter Zeit auch im Maschinellen Lernen eine große Rolle spielt – *Active Learning*.

Bei den Sprachwissenschaftlern wird dieser Begriff eher als Synonym für induktives Lernen gesehen. Diese Betrachtung könnte sinnvoll sein, muss aber noch mehr ausgearbeitet werden! Hier sollen hier zwei Ausprägungen des *Active Learning* näher behandelt werden, die die Verbindung zwischen Sprachwissenschaft und Maschinellem Lernen aufzeigen.

3.1 Active Learning in der Didaktik

In der Didaktik wird *Active Learning* als Gegenteil des passiven Lernens gesehen, was plausibel erscheint. Dabei handelt es sich um keine neue Idee. Bestimmte Ansätze gehen bis auf Sokrates zurück. In vielen Klassenzimmern und Hörsäälen wurde vergessen, dass es sich beim Lernen um einen aktiven Prozess handelt. Bonwell und Eison (siehe [1]) definieren *Active Learning* so:

> „Lernende müssen mehr tun als zuhören. Sie sollten lesen, schreiben, diskutieren oder sich mit Problemlösungen beschäftigen. Am wichtigsten ist es – um aktiv eingebunden zu sein –, dass sich Lernende mit höhergeordneten Denkaufgaben, wie Analyse, Synthese und Evaluation beschäftigen. Strategien, die aktives Lernen fördern, sind als anweisende Aktivitäten definiert, die Lernende darin einbinden, Dinge zu tun und darüber nachzudenken, was sie tun."

Dieser Prozess trifft natürlich auch auf das Grammatiklernen bei Fremdsprachen zu.

3.2 Active Learning beim Induktiven Lernen

Mooney und Kollegen [7] stellten ein Verfahren vor, das in das induktive Lernen einzuordnen ist. Als Anwendungen kamen das Lernen einer Nutzeroberfläche für eine geografische Datenbank, wo natürlichsprachliche Fragen in Prolog-Anfragen umgewandelt und somit Antworten gegeben werden können, sowie die Wissensgewinnung aus Postings in Newsgroups zur Sprache.

Hauptmotiv des Einsatzes von *Active Learning* ist die fehlende Verfügbarkeit von ausreichend kommentierten bzw. klassifizierten Beispielen. Darüber hinaus mangelt es an der im Maschinellen Lernen bekannten Strukturierung durch Merkmale. Beispiele in allgemeiner Form gibt es reichlich. Methoden des *Active Learnings* versuchen für die Kommentierung/Klassifikation nur die informativsten Beispiele auszuwählen. Darum sind sie für Anwendungen der natürlichen Sprache sehr nützlich. Wenn man dort nicht auf einen geduldigen Lehrer oder eine zufällige Sammlung von Beispielen vertrauen will, kann eine Methode genutzt werden, die aktiv an der Sammlung von Beispielen mitwirkt. Ziel ist die Reduzierung der überwachten Trainingsbeispiele, die notwendig sind, um einen gewissen Leistungsgrad zu erreichen. Gleichzeitig entsteht eine Datenbank mit lernwirksamen Beispielen.

Das informativste unter den unkommentierten Beispielen kann auf zwei verschiedene Arten ausgewählt werden. Mooney stellt das sicherheitsbasierte und das komitee-basierte Verfahren vor.

- sicherheits-basiert:
  Zunächst wird aus wenigen kommentierten Beispielen ein einfacher Klassifikator gelernt. Das System prüft einige unkommentierte Beispiele und beschreibt sie durch Sicherheiten der vorausgesagten Kommentierung. Die k Beispiele mit der niedrigsten Sicherheit werden dem Nutzer zum Kommentierung vorgelegt.

- komitee-basiert:
  Wieder wird aus wenigen Beispielen eine Gruppe verschiedener Klassifikatoren gelernt. Jedes Komiteemitglied kommentiert ein Beispiel. Beispiele mit der kleinsten Übereinstimmung über alle Komiteemitglieder werden dem Nutzer zur Kommentierung vorgelegt.

Im Folgenden soll nur der komitee-basierte Ansatz betrachtet werden. Er eignet sich besonders, weil er wie die parallele Anwendung mehrerer verschiedener Klassifikatoren (Menge von Regeln, Entscheidungsbäume, Neuronale Netze

usw.) beim induktiven Lernen funktioniert. Ähnliche Ansätze sind bei schiefen Entscheidungsbäumen (siehe [5] und [3]) und bei sogenannten Entscheidungswäldern (siehe [4]) anzutreffen.

4 Das Grammatiklernen in Fremdsprachen

4.1 Grundsätzliches zum Modell

Wenn man das Fremdsprachenlernen als aktiven Prozess betrachtet, können die Überlegungen zum *Active Learning* als Grundlage für ein Modell des Grammatiklernens in Fremdsprachen dienen. Beide Parts – Lehrer und Lerner – werden vom Computer übernommen, da der Lerner modellhaft nachgebildet und der Lehrer von der Internetgrammatik repräsentiert werden soll. Diese Form wird von uns als *Consultative Learning* bezeichnet. Die Tabellen 1 und 2 zeigt das vollständige wechselseitige Verhältnis von Lehrer und Lerner sowie Computer und Mensch im Prozess rund um das *Active Learning*.

	Lerner	Lehrer
Mooneys Active Learning	Computer	Mensch
Sprachlernen Schule	Mensch 1	Mensch 2
Internet-Grammatik Tutorsysteme	Mensch	Computer
Consultative Learning	Computer 1	Computer 2

Tabelle 1: Einteilung der Lernformen

Lehrer / Lerner	Mensch	Computer
Mensch	Schule Lehre	Tudor-systeme
Computer	Active Learning	Consul-tative Learning

Tabelle 2: Lehrer und Lerner

Active Learning kann dann wie in Abbildung 1 interpretiert werden – egal, ob Computer oder Mensch den Lehrer bzw. Lerner repräsentieren. Das Prinzip, welches immer wieder wiederholt wird, sieht so aus:

1. Erstelle/Modifiziere mit einer begrenzten Menge von Beispielen einen odere mehrere Klassifikatoren (Komitee).
2. Klassifiziere mit dem Komitee weitere Beispiele.
3. Entscheide, welches Beispiel am informativsten ist – also durch das Komitee am unterschiedlichsten und unsichersten klassifiziert wurde.
4. Lege es dem Lehrer zur Bewertung vor und füge es dann der Menge der kommentierten Beispiele hinzu.

Zum möglicherweise notwendigen Modifizieren des Modells wäre ein inkrementelles Lernverfahren von Vorteil, um nicht vollständig neu lernen zu müssen.

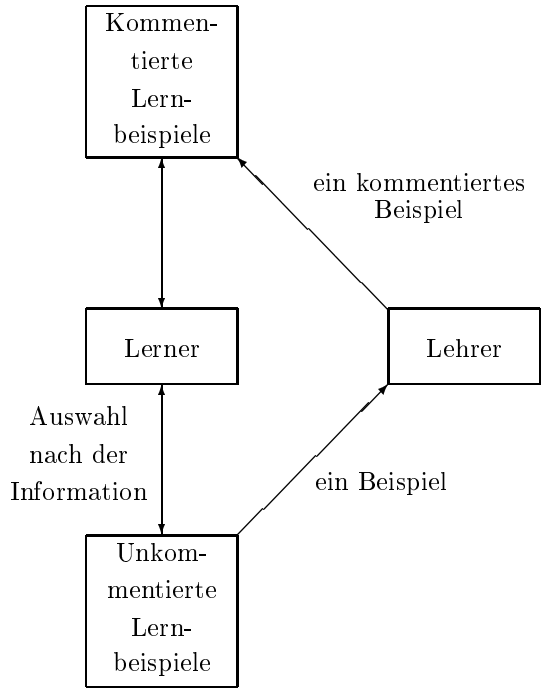


Abbildung 1: *Active Learning* beim Sprachlernen

4.2 Agenten und Active Learning

Russell und Norvig stellen in [6] ein Agentenmodell vor, dessen Funktionsweise in Abbildung 2

noch einmal aufgezeigt wird. Ein solcher Agent kann auch den Lerner oder Lehrer modellieren. Mittels Sensoren und Effektoren wird mit der Umwelt interagiert. Der Performanzstandard kann ebenfalls „von außen“ vorgegeben werden. Zentrale Komponenten sind das Lern- und das Performanzelement.

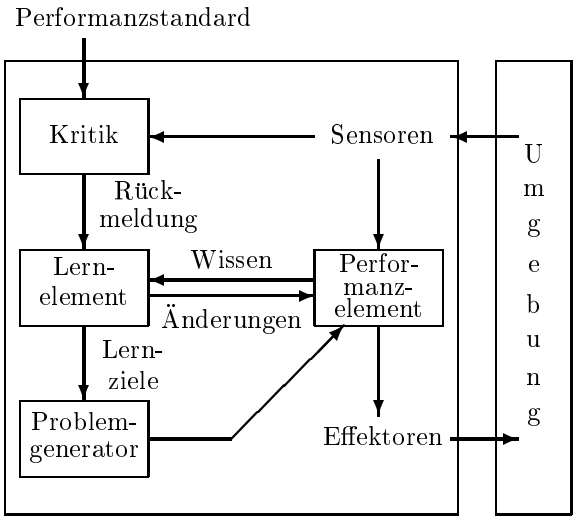


Abbildung 2: Agent nach Russell und Norvig

Mit dem Agentenmodell von Russell und Norvig läßt sich die Idee vom künstlichen Lerner und Lerner weiter ausbauen. Lehrer und Lerner kommunizieren über Sensoren und Effektoren. Darüber hinaus hat der Lehrer Einfluss auf den Performanzestandard des Lerners (siehe Abbildung 3). Wir gehen davon aus, dass der maschinelle Lehrer über kein vollkommenes Sprachwissen verfügt (vermutlich auch die meisten menschlichen Lehrer nicht), und deshalb nicht alle Bitten des Lerners um Kommentierung/Klassifikation von Beispielen auf Grund seines aktuellen Wissens erfüllen kann. Drei Möglichkeiten zur Lösung sind in dieser Situation denkbar:

1. Der Lehrer stößt selbst eine Lernprozess an und ermittelt auf der Basis seiner Beispiele und seines allgemeinen Sprachwissens eine Kommentierung des vorgelegten Beispieles.

2. Es gelingt dem Lehrer nicht, eine hinreichend sichere Kommentierung zu bestimmen – nur mehrere mögliche Alternativen. Lehrer und Lerner versuchen im Dialog sich auf eine Alternative zu einigen.

3. Es wird ein besonders kompetenter menschlicher Supervisor eingeschaltet. Dieser erhöht gewissermaßen von außen den Performanzstandard von Lehrer **und** Lerner. Dieser Fall ist in Abbildung 3 illustriert.

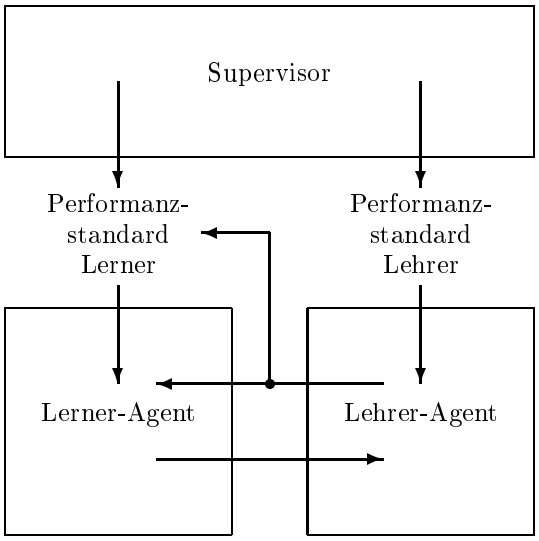


Abbildung 3: Zwei Agenten und ein Supervisor

4.3 Ziel und Stand der Arbeit

In unserem früheren Beitrag [2] wurde als Ziel der Arbeit die Entwicklung eines „künstlichen Internetnutzers“. Damals war die Absicht die Nutzermodelle aus anderen Disziplinen wie Psychologie oder Sprachwissenschaft zu übernehmen, zu formalisieren und zu implementieren. Durch die Simulation sollte die Tragfähigkeit der Modelle überprüft werden und die Simulationsergebnisse sollten mit den Ergebnissen der empirischen Untersuchungen verglichen werden.

Im Lauf der Arbeit stellte sich heraus, dass ein wesentlicher Aspekt bei der Modellierung eines Internetnutzers seine Lernfähigkeit ist. Besonders in dem sprachwissenschaftlichen Teilprojekt steht dieser Aspekt mit der Nutzung der *Internet Grammar* im Vordergrund, aber auch im psychologischen Teilprojekt spielt er insofern eine wichtige Rolle, als dort versucht wird den Nutzen verschiedener Formen der Präsentation durch das bei der Informationsaufnahme Gelernte zu beschreiben. Für das Sprachlernen bietet sich die Methode des *Active Learning* als Mittel an um die Tätig-

keit eines Nutzers der *Internet Grammar* zu modellieren. Deshalb ist unser aktuelles Ziel die Entwicklung eines künstlichen Nutzers der *Internet Grammar*, dessen wesentliche Methode des Lernens das in Abschnitt 4.1 erwähnte *Consultative Learning* ist.

Die Implementierung des komitee-basierten Lernverfahrens ist im Wesentlichen abgeschlossen. Die Klassifikatoren sind Entscheidungsbäume, die nach dem Prinzip des *Information Gain* erzeugt werden. In nächster Zeit sollen noch durch Neuronale Netze erzeugte Klassifikatoren dazu genommen werden. Der sprachliche Untersuchungsbereich wird zunächst auf die Verwendung von Relativpronomina beschränkt, weil hier bereits Vorarbeiten im sprachwissenschaftlichen Teilprojekt vorhanden sind. Zur Zeit laufen Arbeiten zur Aufbereitung von Beispielsätzen mit Relativpronomina in eine für das Lernverfahren verarbeitbare Form. Mit diesen Beispielen wird zunächst das Training durchgeführt. Als Testbeispiele werden die in der *Internet Grammar* verwendeten einschlägigen Übungsaufgaben eingesetzt.

5 Ausblick

Nach den Beispielsätzen mit Relativpronomina (siehe Abschnitt 4.3) wollen wir den Sprachumfang noch um andere grammatikalische Elemente ergänzen. Dazu sind aber weitere Untersuchungen und Vorarbeiten im sprachwissenschaftlichen Teilprojekt notwendig.

Ebenfalls sollen die Erkenntnisse und Ergebnisse aus den Untersuchungen der Kollegen aus der Allgemeine Psychologie und Arbeitspsychologie, die sich unter anderem auch mit der Anwendung der *Internet Grammar* beschäftigen, in den künstlichen Nutzer einfließen. In dem Projekt wird die nutzerorientierte Präsentation von Informationen im Internet untersucht. Die Bildschirminhalte der Sitzungen der Versuchspersonen werden dort auf Video aufgezeichnet. Außerdem werden Logfiles der Sitzungen erzeugt und die Daten der Blickbewegung aufgezeichnet. Aus der Gesamtheit dieser Informationen wollen wir weitere Erkenntnisse für die Gestaltung des künstlichen Internetnutzers herausarbeiten.

Literatur

[1] C. C. Bonwell and J. A. Eison. Active learning: Creating excitement in the classroom. ashe-eric higher education report no. 1. Technical report, School of Education and Human Development, George Washington University, Washington, D.C., 1991.

[2] W. Dilger and J. Zeidler. Das Projekt „Modellierung und Simulation der Rezeption textuell repräsentierter Inhalte im Internet". In F. Wysotzki, P. Geibel, and K. Schädler, editors, *Beiträge zum 11. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3*, pages 85–89, Berlin, Germany, 1998.

[3] D. Heath, S. Kasif, and S. Salzberg. Induction of oblique decision trees. In *Proc. of the 13 th IJCAI*, pages 1002–1007, Chambéry, France, 1993.

[4] P. M. Murphy and M. J. Pazzani. Exploring the decision forest: An empirical investigation of occam's razor in decision tree induction. In *Journal of Artifical Intelligence Research*, volume 1, pages 257–275. AI Access Foundation and Morgan Kaufmann Publishers, 1994.

[5] S. Murthy, S. Kasif, S. Salzberg, and R. Beigel. OC1: Randomized induction of oblique decision trees. In *Proc. of the 11 th Nat. Conf. on AI AAAI-93*, pages 322–327, Washington, D.C., 1993.

[6] S. J. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[7] C. A. Thompson, M. E. Califf, and R. J. Mooney. Active learning for natural language parsing and information extraction. In I. Bratko and S. Dzeroski, editors, *Proc. of* 16 *th International Machine Learning Conference*, pages 406–414, Bled, Slovenia, 1999.

[8] K. Boehnke u.a. *Neue Medien im Alltag: Von individueller Nutzung zu soziokulturellem Wandel, Band 1.* Pabst Science Publishers, Lengerich, 1999.

Ariadne: ein fokussierter Web-Crawler mit adaptiver Klassifikation der Hyperlinks

Martin Ester, Matthias Groß
Institut für Informatik, Ludwig-Maximilians-Universität München,
Oettingenstr. 67, D-80538 München
{ester|gross}@dbs.informatik.uni-muenchen.de

Extended Abstract

1. Einleitung

Das World-Wide Web wächst so rasant, daß die Suchmaschinen nur einen relativ kleinen Teil aller Webseiten indizieren können. Aus diesem Grund ist der *Recall* (der Anteil der für eine Anfrage gelieferten relevanten Webseiten an der Menge aller relevanten Webseiten) der Suchmaschinen relativ gering. Zusätzlich sind im allgemeinen viele Antworten veraltet oder URLs nicht mehr gültig. Oft ist außerdem die *Precision* (Anteil der relevanten Webseiten an der Menge aller gelieferten Antworten) der Suchmaschinen schlecht, d.h. neben den relevanten Antworten wird ein Vielfaches an irrelevanten Webseiten geliefert. Fokussierte Crawler sind eine Alternative zu den konventionellen Suchmaschinen, die diese Schwächen vermeiden.

Ein *fokussierter Crawler* erhält eine Menge von Webseiten (*Trainingsseiten*), die ein für den Benutzer interessantes Thema spezifizieren. Auf der Suche nach weiteren relevanten Seiten geht der Crawler von den Trainingsseiten aus und besucht eine geeignet eingeschränkte Teilmenge der durch Hyperlinks direkt oder indirekt verbundenen Webseiten. Dieser Ansatz basiert auf der Beobachtung, daß durch einen Hyperlink miteinander verbundene Seiten sehr viel wahrscheinlicher das gleiche Thema behandeln als zwei zufällig gewählte Seiten. Zentrale Aufgabe beim fokussierten Crawling ist eine Anordnung der Hyperlinks nach absteigender "Qualität", so daß die besten Links jeweils zuerst verfolgt werden können.

In [2] wird gezeigt, daß bereits mit einer einfachen Bewertung des Texts der Quellseite und des Ankertexts des Hyperlinks wesentlich bessere Ergebnisse erzielt werden als bei zufälliger Wahl der Priorität der Links. In [1] wird zur Ordung der Links sowohl die Relevanz der Quellseite als auch ihre "Zentralität" bestimmt. Die *Zentralität* einer Webseite wird rekursiv durch die Zentralität der benachbarten Seiten definiert. Die Berechnung erfolgt mit Hilfe der Adjazenzmatrix eines Teiles des Webgraphen, wobei die Links initial mit der Relevanz ihrer Zielseite gewichtet werden. Diese Methode hat sich experimentell bewährt, erfordert aber zur Bestimmung der Zentralität das Laden zahlreicher Webseiten, die für das benutzerdefinierte Thema irrelevant sind.

2. Der fokussierte Crawler Ariadne

Wir betrachten als maßgeblichen Kostenfaktor das Laden der Webseiten und verfolgen daher ein alternatives Konzept: die einzelnen Hyperlinks werden unter Nutzung von Ankertext und Ziel-URL themenspezifisch klassifiziert, ohne die jeweilige Zielseite schon zu laden. Der Link mit der höchsten Bewertung wird als nächster verfolgt. Dieser Link-Klassifikator ist adaptiv, d.h. er wird im Lauf des Crawls aufgrund der Erfahrung mit den tatsächlich geladenen Webseiten verbessert. Das neue Konzept wird momentan im System *Ariadne* (Algorithm Regarding Interesting Anchortext for Directed Neighborhood Expansion) prototypisch implementiert und evaluiert.

Ariadne läßt sich in folgende Phasen gliedern:

- *Vorverarbeitung*
  Aus den Webseiten im HTML-Format werden die reinen Texte extrahiert. Dann erfolgt ein Stemming der Terme sowie eine Elimination von Stoppwörtern.
- *Feature-Extraktion*
  Nach der Zählung der Häufigkeiten aller enthaltenen Terme werden seltene Terme eliminiert. Die Auswahl der für die Klassifikation der Texte und der Hyperlinks wichtigsten Terme (Features) geschieht in Kooperation mit dem Benutzer, der einen Parameter eingibt und die vorgeschlagenen Features modifizieren kann.
- *Crawl*
  Der Crawler startet mit den Trainingsseiten und

durchsucht einen Teil des Webgraphen. Die dabei gefundenen für den Benutzer "relevanten" Webseiten werden ausgegeben.

Die zentrale Phase des eigentlichen Crawls wiederholt für die jeweils aktuelle Webseite folgende Schritte:

- *Klassifikation des Textes der Webseite*
  Aus der geladenen Webseite wird wie in der Vorverarbeitung der reine Text extrahiert und die Häufigkeiten der Features gezählt. Mit Hilfe eines einfachen Textklassifikators wird nun die Wahrscheinlichkeit bestimmt, daß die aktuelle Seite relevant ist.
- *Klassifikation der enthaltenen Hyperlinks*
  Ein zweiter Klassifikator nutzt den Text der aktuellen Webseite sowie die Ankertexte und die URLs, um die in der aktuellen Seite enthaltenen Hyperlinks für die Zwecke des Crawl zu bewerten. Alle diese Hyperlinks werden mit ihrer Bewertung in eine sortierte Liste der *offenen Hyperlinks* eingefügt.
- *Verfolgen des Links mit der besten Bewertung*
  Nach Abarbeiten der aktuellen Webseite wird als nächstes der beste offene Hyperlink, d.h. der Hyperlink mit der global höchsten Bewertung verfolgt. Die Webseiten werden also nur nach Prioritäten geordnet, es werden keine Seiten explizit von der Suche ausgeschlossen.

3. Klassifikation der Hyperlinks

Der zentrale Schritt des fokussierten Crawlers ist die Klassifikation der Hyperlinks, die im folgenden genauer erläutert wird.

Jedem Link wird eine Zahl zwischen 0 und 1 zugeordnet, die als Wahrscheinlichkeit dafür interpretiert wird, daß die Zielseite für das Anfragethema relevant ist. Diese Bewertung ergibt sich als gewichtete Summe von Einzelkriterien wie Themenbezug der Quellseite sowie Bewertung von Ankertext und Ziel-URL.

Beim Start des Crawlers sind nur die Trainingsseiten bekannt. Es wurden noch keine Hyperlinks verfolgt, so daß der Link-Klassifikator noch nichts lernen konnte. In der initialen Phase des Crawls wird daher zur Bewertung der Ankertexte und URLs ein statischer Klassifikator verwendet.

Im weiteren Verlauf des Crawls soll die Klassifikation der Hyperlinks adaptiv aufgrund der Erfahrungen mit den geladenen Webseiten verbessert werden. Der Link-Klassifikator soll insbesondere von Fällen lernen, in denen seine Vorhersage stark von der späteren Klassifikation des Textes der Zielseite abwich. Während der initialen Phase sollen deshalb für die Zwecke des Lernens auch genügend viele als "schlecht" bewertete Links verfolgt werden.

Am Ende der initialen Phase werden aus der Menge aller bisher geladenen Webseiten zwei Typen von Trainingsseiten ausgewählt:

1. *einfache Seiten*, d.h. Seiten, bei denen die Bewertung durch den Link-Klassifikator und durch den Text-Klassifikator gut übereinstimmte.
2. *schwierige Seiten*, d.h. Seiten, bei denen die Klassifikation des Links stark von der späteren Klassifikation des Textes der Webseite abwich.

Beide Typen von Webseiten müssen angemessen berücksichtigt werden. Mit Hilfe dieser Trainingsseiten wird nun ein Naiver Bayes-Klassifikator für Ankertexte und URLs trainiert, der im Vergleich zum einfachen statischen Klassifikator im allgemeinen eine wesentlich größere Anzahl von Features berücksichtigt. Nach dem Trainieren des Link-Klassifikators müssen alle offenen Links mit Hilfe dieses Klassifikators neu bewertet werden, womit im allgemeinen ihre Ordung geändert wird.

Die Adaption des Link-Klassifikators wird während des Crawls zu geeigneten Zeitpunkten wiederholt. Der Text-Klassifikator ist dagegen statisch, d.h. das für den Benutzer relevante Thema wird während des Crawls als invariant angenommen.

4. Ausblick

Während der Implementierung wurden erste Experimente durchgeführt. Die Precision wird automatisch mit Hilfe des Text-Klassifikators berechnet und ist erfolgversprechend. Der Recall soll durch Vergleich mit den Ergebnissen einer Suchmaschine ermittelt werden, was sich erst nach Integration von Ariadne mit einem Datenbanksystem effizient durchführen lassen wird. Die Integration mit einem relationalen Datenbanksystem, das sowohl die offenen Links als auch die geladenen Webseiten effizient verwaltet, ist weitgehend abgeschlossen.

Ein experimenteller Vergleich von Ariadne mit Verfahren aus der Literatur ist geplant. Ein wichtiges Thema unserer weiteren Forschung soll die Integration des Benutzers in den Lernprozeß des Crawlers sein, damit sein Feedback möglichst früh zur Fokussierung der Suche genutzt werden kann.

Referenzen

[1] Chakrabarti S., van den Berg M., Dom B.: "*Focused Crawling: a new Approach to Topic-Specific Web Resource Discovery*", Proc. WWW 1999.

[2] Cho J., Garcia-Molina H., Page L.: "*Efficient Crawling Through URL Ordering*", Proc. WWW 1998.

Lernen von Ausspracheregeln mit dem IAK-Modell

Martin Heydemann

Institut für Psychologie, Technische Universität Darmstadt, Steubenplatz 12, D-64293 Darmstadt
Zur Zeit: Katholische Universität Eichstätt, Philosophische Fakultät, Ostenstr. 25, D-85071 Eichstätt.
Email: heydemann@psychologie.tu-darmstadt.de

Zusammenfassung. Bei dem IAK-Modell handelt es sich um ein neuronales Netz. Das Modell zeichnet sich dadurch aus, dass gleichzeitig sowohl eine sehr hohe Diskrimination (zwischen den Reizen der Trainingsphase) als auch eine sehr gute Generalisierung (für neue Reize) erzielt wird. Die Anwendung des Modells auf das Lernen der Aussprache von geschriebenen Wörtern wird dargestellt. Für neuronale Netze liegt die Schwierigkeit in diesem Anwendungsbereich darin, dass wenige recht allgemeine Regeln und gleichzeitig eine Vielzahl von Ausnahmen gelernt werden müssen. Das IAK-Modell eignet sich für diese Lernanforderung recht gut. Hierzu werden erste Ergebnisse aus einem noch laufenden Projekt berichtet.

Beim IAK-Modell handelt es sich um ein neu entwickeltes neuronales Netz (IAK steht für **I**nformations-**A**uswertung von **K**onfigurationen). Gegenüber dem wohl am weitesten verbreiteten Lernverfahren für neuronale Netze, dem „Backpropagation“ Verfahren, weist IAK eine Reihe von Unterschieden auf. Dazu gehören die folgenden Eigenschaften:

- Inputvariablen müssen nominalskaliert sein („Merkmal ist vorhanden“ vs. „Merkmal ist nicht vorhanden“).
- Ein große Anzahl von Inputvariablen kann verarbeitet werden. Dabei dürfen für den einzelnen Lernreiz nur eine geringe Anzahl von Variablen den Wert „Merkmal ist vorhanden“ aufweisen.
- Eine Verbesserung der Diskriminationsleistung geht nicht auf Kosten der Generalisierungsfähigkeit.
- Neue Daten können hinzu gelernt werden, ohne dass das bereits Gelernte wiederholt werden muss.

Diese Eigenschaften des IAK-Modells lassen das Modell für bestimmte Anwendungsbereiche als besonders geeignet erscheinen. Dazu gehören Anwendungen, bei denen Symbole (z.B. Worte oder Buchstaben) verarbeitet werden.

Im Vortrag wird die Anwendung des IAK-Modells in einem noch in der Anfangsphase befindlichen Projekt vorgestellt. Inhaltlich geht es dabei um die Aussprache von geschriebenen Wörtern, also die Umsetzung einer Folge von Buchstaben in eine Phonemsequenz. Menschen beherrschen hierbei sowohl die Besonderheiten der Aussprache für spezielle Wörter, als auch allgemeine Regeln, die beispielsweise angewendet werden müssen, um Nicht-Wörter (unbekannte Buchstabenfolgen) zu lesen und auszusprechen. Die Regeln, mit denen die korrekte Aussprache von Nicht-Wörtern möglich ist, lassen sich analytisch bestimmen (siehe Coltheart, 1978, 1987; Coltheart, Curtis, Atkins & Haller, 1993).

Für neuronale Netze besteht beim Lernen die Schwierigkeit darin, gleichzeitig diese Regeln und eine große Zahl an Ausnahmen zu lernen. Wurden beispielsweise die Ausnahmen korrekt gelernt, dann gab es Schwierigkeiten bei der Vorhersage der Aussprache von Nicht-Wörtern (z.B. Seidenberg & McClelland, 1989). Inzwischen ist es jedoch gelungen, ein spezielles auf einer modifizierten Form von Backpropagation beruhendes neuronales Netz zu konstruieren, das sowohl die Ausnahmewörter lernt, als auch bei Nicht-Wörtern vergleichbar gute Ergebnisse liefert wie die Anwendung der analytisch bestimmten Regeln (Seidenberg, Plaut, Petersen, McClelland & McRae, 1994). Allerdings lässt sich dieses Verfahren nur auf einsilbige Wörter anwenden, die zudem noch vorsegmentiert werden müssen.

Während die Konstruktion von neuronalen auf Backpropagation beruhenden Netzen in diesem Anwendungsbereich sehr schwierig ist, lässt sich erwarten, dass das IAK-Modell aufgrund seiner Eigenschaften hier besonders gut geeignet ist. Das IAK-Modell wurde ursprünglich als Modell für das Lernen von Kategorien beim Menschen konstruiert (Heydemann, 1995, 1998a). Es lässt sich jedoch auch erfolgreich in Bereichen des maschinellen Lernens anwenden, wenn es um die Vorhersage von Kategorien geht, z.B. bei der Prognose von Herzerkrankungen (siehe Heydemann, 1998b).

Beim Lernen werden im IAK-Modell über einen Zufallsmechanismus Teilmengen der Reizmerkmale ausgewählt und als Einheiten (sogenannte *Gedächtniseinheiten*) im neuronalen Netz gespeichert. Hierbei wird nur eine begrenzte Anzahl aller möglichen Merkmalskombinationen gespeichert. Für das Anwendungsbeispiel "Wortaussprache" umfassen die Gedächtniseinheiten einzelne Buchstaben oder Kombinationen aus Buchstaben. Die Gedächtniseinheiten werden durch das Lernen mit Einheiten verknüpft, die Phoneme repräsentieren. Sobald eine Gedächtniseinheit im Netz angelegt ist, registriert sie in allen nachfolgenden Lerndurchgängen die für den Lerndurchgang korrekten Antworten oder Kategorien, z.B. bei der Wortaussprache die richtigen Phoneme. Diese bei den Gedächtniseinheiten gespeicherte Information wird benötigt um für neue Reize (z.B. unbekannte Buchstabenfolgen) die korrekte Kategorie (z.B. das richtige Phonem) vorherzusagen. Für diese "Informationsauswertung" mit dem Ziel der Vorhersage der richtigen Kategorie verwendet das IAK-Modell einen speziellen Berechnungsalgorithmus, bei dem für jede Gedächtniseinheit die Zuverlässigkeit der bei ihr gespeicherten Information berücksichtigt wird.

Schwerpunkt des Beitrags bildet die Beschreibung des IAK-Modells, die Darstellung der Besonderheiten für die konkrete Modellanwendung, sowie erste Ergebnisse aus dem zur Zeit noch laufenden Projekt.

Literatur

Coltheart, M. (1978). Lexical access in simple reading tasks. In G. Underwood (Ed.), *Strategies of information processing* (pp. 151-215). San Diego, CA: Academic Press.

Coltheart, M. (1987). Functional architecture of the language-processing system. In M. Coltheart, G. Satori, & R. Job (Eds.), *Cognitive neuropsychology of language* (pp. 1-26). Hillsdale, NJ: Erlbaum.

Coltheart, M., Curtis, B., Atkins, P., & Haller, M. (1993). Models of reading aloud: dual-route and parallel distributed processing approaches. *Psychological Review, 100*, 589-608.

Heydemann, M. (1995). A connectionist model for classification learning - The IAK model. In J.D. Moore & J.F. Lehman (Eds.): *Proceedings of the seventeenth annual conference of the Cognitive Science Society* (pp. 293-297). Mahwah, New Jersey: Lawrence Erlbaum Associates.

Heydemann, M. (1998a). Ein neuronales Netz zum Lernen von Kategorien: Das IAK-Modell (pp. 56-63). In F. Wysotzki, P. Geibel & K. Schädler (Eds.), *Beiträge zum Treffen der GI-Fachgruppe 1.1.3: Maschinelles Lernen, FGML-98* (pp. 56-63). Technischer Bericht 98/11; Fachbereich 13, Technische Universität Berlin.

Heydemann, M. (1998b). *Lernen von Kategorien*. Wiesbaden: Deutscher Universitäts-Verlag.

Seidenberg, M.S., & McClelland, J.L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review, 96*, 523-568.

Seidenberg, M.S., Plaut, D.C., Peterson, A.S., McClelland, J.L. & McRae, K. (1994). Nonword pronunciation and models of word recognition. *Journal of Experimental Psychology: Human Perception and Performance, 20*, 1177-1196.

Ontologies that Help Document Databases Find Better Answers

- Extended Abstract -

Wolfgang Wohner

FORWISS (Bavarian Research Center for Knowledge-Based Systems)
Orleansstr. 34, D-81667 Munich, Germany
Email: wohner@forwiss.de

Ontologies and taxonomies provide a powerful means for assigning semantic structure to a given information space such as the World Wide Web. Current search strategies and their corresponding indexing techniques rely heavily on simple keywords while human users think and communicate on a rather different scale: word usage and meaning depend foremost on the particular context they are used in. Even additional search features like introducing boolean operators such as AND, OR and NOT, grouping keywords together in order to denote phrases, etc. cannot overcome this lack of context as searching algorithms are mainly based on simple pattern matching. Consequently, query result sets are often diluted by irrelevant hits (caused e.g. by homonyms) whereas a substantial number of documents satisfying the user's actual information need cannot be found because they do not contain the exact phrase or keywords (but synonymous terms or hypernyms/hyponyms, meronyms/holonyms, etc.). What is needed is a technique that introduces semantic structure to these mere syntactic, keyword-driven search capabilities.

A feasible approach is to replace keywords and terms (strings) by more complex, semantically enriched constructs that are capable of giving an account of what they are representing, i.e. *concepts*. Concepts hold a clear definition of the object they are referring to thus avoiding linguistic confusion about the respective term. Accordingly, the concept CAT extends the English word 'cat' by providing an intrinsic description (in its simplest form a list of equal or related terms). Moreover, concepts are interrelated (e.g. CAT — FUR, CAT — MAMMAL) presenting the structure of a given domain characterized by a set of concepts. The overall structure as outlined by the relations between these concepts then defines the domain's ontology.

The basic idea of our approach is to use the semantics inherent in such an ontology for optimizing the system response quality to user queries while maintaining efficiency (fast response times). We are currently developing such an enhanced knowledge management and retrieval system at the *Bavarian Research Center for Knowledge-based Systems (FORWISS)*. The system's architecture falls into three main components: (a) a *domain ontology*, (b) a *query interface* and (c) a *retrieval system*.

At the core of the overall system is the *domain ontology*. We assume that some community of interests (e.g. a company or a scientific community) defines its field of expertise using concepts and their correlations that ultimately form the ontology. Thus, a semantic network representing human understanding of a certain domain is being modeled (representational faculty). But the domain ontology also has to serve a very different purpose, namely providing an index structure to the associated corpus of information units, e.g. XML documents (retrieval faculty). In order to meet both of these requirements the internal organization of the domain ontology

has to follow certain guidelines: the internal structure must be kept simple (hierarchical) for indexing purposes yet semantically expressive with regard to modeling a given domain. A suitable solution is to split the domain into several subcategories (e.g. time, geographical region, etc.), each represented by a strictly hierarchical (leveled) taxonomy of respective concepts. This practice aims at providing simple access structures while maintaining adequate expressiveness. Since the same concept may be part of several taxonomies any interdependency between concepts can still be modeled.

A *query interface* is used to translate user requests into the conceptual space of the domain ontology. An adequate query language will be designed that takes into account the structure and capabilities of ontologies. According to its type the original query can then be refined or generalized in order to achieve a better result set (semantic query rewriting). For example a query *'find all sports clubs in Chicago'* might be restated with regard to the concept SPORTS CLUB (provided that the domain ontology contains a corresponding taxonomy) to a set of queries *'find all baseball clubs in Chicago'*, *'find all football clubs in Chicago'*, etc. Further refinement applied to CHICAGO will follow a similar pattern. Semantic query rewriting thus enlarges the set of documents relevant to the original query while the transposition of user requests to match the concepts of the domain ontology within a certain context (taxonomy) eliminates linguistic ambiguity.

The essential task of the *retrieval system* is to answer the semantically rewritten queries. Prerequisite to any query processing a set of documents has to be classified according to the domain ontology, i.e. documents, or parts thereof, are being linked to concepts of the ontology. Again, the information held within the documents must be translated into the conceptual space of the respective taxonomies. This can be done e.g. by linguistic analysis of their content or by taking advantage of their (known) internal structure. For reasons of simplicity we assume that the documents in fact are accordingly structured as is the case e.g. with XML documents based on DTDs that comply with the ontology's own structure. The hierarchical makeup of XML documents then typically contains substructures which can be mapped to subtrees of taxonomies. That way concepts can be easily linked to elements of XML documents.

The system layout just characterized then has to be designed on the physical (storage) layer in such a way that response times are minimized. Conceptually, a multidimensional (MD) view on the data models can be established for indexing a data set (e.g. XML documents) by an ontology. Each dimension within that multidimensional space is classified by a taxonomy of the domain ontology. Taxonomies can therefore be used to define *dimension hierarchies.* Semantic query rewriting typically yields operations which retrieve and rank all documents that match some hierarchy level in several taxonomies at the same time. Therefore, we are promoting a physical clustering of the documents so that queries exhibiting these characteristics are performed efficiently. In order to optimize access to these documents the index laid out by the taxonomies needs to be easy and fast to process. For that reason we are considering a *multidimensional hierarchical clustering (MHC)* technique that encodes hierarchies using *compound surrogates* (bit vectors prefixing each hierarchy level for direct access). Further research will examine the applicability of this technique in the context of ontology based knowledge management and retrieval.

Verzeichnis der Teilnehmenden der FGML-2000

Klaus-Dieter Althoff; Fraunhofer IESE; SLI; Sauerwiesen 6; 67661 Kaiserslautern; tel.: 06301/707-230; althoff@iese.fhg.de; http://www.iese.fhg.de/Staff/althoff

Niko Beerenwinkel; GMD SCAI; tel.: 02241/14 2786; niko.beerenwinkel@gmd.de

Thorsten Belker; Universität Bonn; Informatik III; Romerstrasse 164; 53117 Bonn; tel.: 0228/734516 belker@cs.uni-bonn.de; http://www.informatik.uni-bonn.de/~belker/

Thorsten Boseniuk; Anwendungszentrum Neuro-Informatik; Hagenower Str. 73; 19061 Schwerin; tel: 0385/3993436; thorsten@ani.de; http://www.ani.de

Christian Buck; AiS; KD; tel.: 02241/142261; CBuck@lantis.de

Thomas Burwick; Thinking Networks AG; Markt 45 - 47; tel.: 0241/47072 200; thomas.burwick@gmi-mbh.de

Werner Dilger; TU Chemnitz; Fakultät für Informatik; Straße der Nationen 62; 09107 Chemnitz; tel.: 0371/531-1529; Dilger@informatik.tu-chemnitz.de; http://www.tuchemnitz.de/informatik/HomePages/KI/ki.html

Boulanger Dmitri; DIALOGIS Software & Services GmbH; 53115 Bonn; tel.: 030/6392 1905; dmitri@first.gmd.de; http://wdp.first.gmd.de/dima

Christoph Engels; Thinking Networks AG; Markt 45-47; 52062 Aachen; tel.: 0241/47072 200; christoph.engels@thinking-networks.com

Martin Ester; Ludwig-Maximilians-Universität; Institut für Informatik; Oettingenstr. 67; 80538 München; tel.: 089/2178 2195; ester@dbs.informatik.uni-muenchen.de; http://www.dbs.informatik.uni-muenchen.de/~ester

Andreas Faatz; TU Darmstadt; KOM; Merckstr. 25; 64283 Darmstadt; tel.: 06151/162078; afaatz@kom.tu-darmstadt.de

Andreas Fick; Forschungszentrum Karlsruhe; Institut für Angewandte Informatik; Herrmann-von-Helmholtz-Platz 1; 76344 Leopoldshafen; tel.:07247/82-5782; fick@iai.fzk.de; http://wwwserv2.iai.fzk.de/Institut/UI/INPRO/index.html

Josef Fink; humanIT GmbH; Rathausallee 10; 53757 Sankt Augustin; tel.: 02241/92926 00; josef.fink@humanit.de; http://www.humanit.de

Joachim Gebauer; Dialogis Software & Services; Trierer Str. 70-72; 53115 Bonn; tel.: 0228/24982 40

Ralf Haselmann; Uni-Erlangen; LS für Compilerbau und algorithmische Sprachen; Martensstraße 3; 91058 Erlangen; tel.: 09131/85 27620; rfhaselm@cip.informatik.uni-erlangen.de; http://www2.informatik.uni-erlangen.de

Nicola Henze; Universität Hannover; ITI, Abtg. Rechnergestützte Wissensverarbeitung; Appelstr. 4; 30167 Hannover; tel.: 0511/762 19716; henze@kbs.uni-hannover.de; http://www.kbs.uni-hannover.de/~henze

Martin; Heydemann; Technische Universität Darmstadt; Institut für Psychologie; Steubenplatz 12; 64293 Darmstadt; tel.: 06151/597457 heydeman@psychologie.tu-darmstadt.de

Alexander Hinneburg; Universitat Halle; Informatik; Kurt Mothes Str. 1; 06120 Halle/Saale; tel.: 0345/55 24737; hinneburg@informatik.uni-halle.de; http://www.informatik.uni-halle.de/~hinnebur

Susanne Hoche; Otto-von-Guericke-Universität Magdeburg; IWS; Postfach 4120; 39016 Magdeburg; tel.: 0391/67 12347; hoche@iws.cs.uni-magdeburg.de

Tamás Horváth; GMD; AiS.KD; Schloss Birlinghoven; D-53754; Sankt Augustin; tel.: 02241;14 2686; tamas.horvath@gmd.de; http://ais.gmd.de/people/Tamas.Horvath.html

Brijnesh Johannes Jain; TU Berlin; Moderne Methoden der KI; Franklinstr. 28/29; 10587 Berlin; bjj@cs.tu-berlin.de

Thorsten Joachims; GMD Forschungszentrum Informtionstechnik; AIS.KD; Schloss Birlinghoven; 53754 Sankt Augustin; tel.:02241/14-2870; Thorsten.Joachims@gmd.de; http://www-ai.cs.uni-dortmund.de/thorsten

Kristian Kersting; Universität Freiburg, Institut fuer Informatik, Maschinnelles Lernen und Natuerlichsprachliche Systeme; Georges-Koehler-Allee 79; 79085; Freiburg i. Br.; tel.: 0761/203-8010; kersting@informatik.uni-freiburg.de; http://www.informatik.uni-freiburg.de/~kersting

Mathias Kirsten; GMD; AiS.KD; Schloss Birlinghoven; 53754 Sankt Augustin; tel.: 02241/14-2686; mathias.kirsten@gmd.de; http://ais.gmd.de/~kirsten/

Ralf Klinkenberg; Universität Dortmund; Fachbereich Informatik, Lehrstuhl für Künstliche Intelligenz; Baroper Str. 301; 44221 Dortmund; tel.: (0231) 755-5103; klinkenberg@ls8.cs.uni-dortmund.de; http://www-ai.cs.uni-dortmund

Gabriella Kókai; Friedrich-Alexander Universität Erlangen - Nürnberg ; Lehrstuhl für Programmier- u. Dialogsprachen sowie Compiler; Martenstr 3; 91056 Erlangen; tel.: 09131-8527830; kokai@informatik.uni-erlangen.de; http://www2.informatik.uni-erlangen.de/~kokai/

Wolfgang Konen; Thinking Networks SI GmbH; Leiter Data Mining; Universitätsstr. 160; 44801 Bochum; tel.: 0234/9787-55; wolfgang.konen@e-miners.de; http://www.e-miners.de

Mark-A. Krogel; Otto-von-Guericke-Universität Magdeburg, FIN; Institut für Wissens- und Sprachverarbeitung; Postfach 41 20; 39016 Magdeburg; tel.: 0391/67 11 399; krogel@iws.cs.uni-magdeburg.de; http://kd.cs.uni-magdeburg.de/~krogel/

Andrea Luethje; Dialogis Software und Services GmbH; Trierer Str. 70-72; 53115 Bonn; tel.: 0228/24982 61; andrea.luethje@dialogis.de; http://www.dialogis.de

Alexander Maedche; Institut AIFB; Universitaet Karlsruhe; 76128 Karlsruhe; tel.: 0721/6086558; ama@aifb.uni-karlsruhe.de; http://www.aifb.uni-karlsruhe.de/WBS/ama

Alexander Mehler; Universität Trier; FB II, Linguistische Datenverarbeitung; Universitätsring 15; 54286 Trier; tel.: 0651/201 2265; mehler@uni-trier.de; http://www.ldv.uni-trier.de.8080/mehler.html

Martin Eric Mueller; Institut fuer Semantische Informationsverarbeitung; Universität Osnabrueck; 49069 Osnabrück; tel.: 0541/969 6226; martmuel@uos.de; http://www.aye-aye.de

Lourdes Pena Castillo; OvG-Universität Magdeburg; Fakultät für Informatik, Institut für Wissens-und Sprachverarbeitung; Universitätsplatz 2; 39106 Magdeburg; tel.: 391/6711399; pena@iws.cs.uni-magdeburg.de

Marco Poloni; MIT GmbH; Marketing und Vertrieb; Promenade 9; 52076 Aachen; tel.: 02408/94580; marco.poloni@mitgmbh.de; http://www.mitgmbh.de

Marion Quink; Dialogis Software & Services; Trierer Str. 70 - 72; 53115 Bonn; tel.: 0228/24982 40; marion.quink@dialogis.de

Matthias Rychetsky; Technische Universität Darmstadt; Fachgebiet Mikroelektronische Systeme; Karlstrasse 15; 64283 Darmstadt; tel.: 06151/164439; rychetsky@mes.tu-darmstadt.de; http://www.microelectronic.e-technik.tu-darmstadt.de/staff/rych/rych_ef.html

Knut Sander; Uni-Erlangen; LS fuer Compilerbau und algorithmische Sprachen; Martensstraße 3; 91058 Erlangen; tel.: 09131/85 27620; knut.sander@informatik.stud.uni-erlangen.de; http://www2.informatik.uni-erlangen.de

Tobias Scheffer; Uni Magdeburg; Institut für Wissens- und Sprachverarbeitung; Universitätsplatz 2; 39106 Magdeburg; tel.: 0391/67 11309; scheffer@iws.cs.uni-magdeburg.de; http://kd.cs.uni-magdeburg.de/~scheffer

Ute Schmid; TU Berlin, FB Informatik; Methoden der KI; Franklinstr. 28; 10587 Berlin; tel.: 030/314-23938; schmid@cs.tu-berlin.de; http://ki.cs.tu-berlin.de/~schmid

Günter Schulz; Thinking Networks AG; Markt 45-47; 52062 Aachen; tel.: 0241/47072200; guenter.schulz@gmi-mbh.de

Cornelia Seeberg; TU Darmstadt; KOM; Merckstr. 25; 64283 Darmstadt; tel.: 06151/166103; seeberg@kom.tu-darmstadt.de

Ingo Schwab; GMD-FIT; Schloss Birlinghoven; 53754 St. Augustin; tel.: 02241/14 2856; ingo.schwab@gmd.de

Uwe Sinha; TU Berlin, FB Informatik; Methoden der KI; Franklinstr. 28; 10587 Berlin; tel.: 030/6237599; uwesinha@cs.tu-berlin.de; http://www.cs.tu-berlin.de/~uwesinha

Sascha Trautzsch; PRUDENTIAL SYSTEMS SOFTWARE GmbH; GF; Annaberger Straße 240; 09125 Chemnitz; tel.: 0371/5347-123; trautzsch@prudsys.com; http://prudsys.de

André Weiser; IBM; data mining development; Schoenaicher Str. 220; 71032; Böblingen; tel.: 07031/16 4775; aweiser@de.ibm.com

Dietrich Wettschereck; Dialogis Software & Services GmbH; Trierer Straße 70-72; 53115; Bonn; tel.: 0228/249820; dietrich.wettschereck@dialogis.de; http://www.dialogis.de

Arnd Winter; SPSS GmbH software; Rosenheimer Str. 30; 81669 München; tel.: 089489074128; awinter@spss.com

Kurt-Ulrich Witt; Fachhochschule Bonn-Rhein-Sieg; Fachbereich angewandte Informatik Grantham-Allee 20; 53757 Sankt Augustin; tel.: 02241/865-200; kurt-ulrich.witt@fh-rhein-sieg.de; http://www.fh-rhein-sieg.de

Wolfgang Wohner; FORWISS; Forschungsgruppe Wissensbasen; Orleanstraße 34; 81667 München; wohner@forwiss.de; http://forwiss.tu-muenchen.de/~wohner

Maria Wolters;Uni Bonn; Institut für Kommunikationsforschung und Phonetik; Poppelsdorfer Allee 47; 53115 Bonn; tel.: 0228/733081; wolters@ikp.uni-bonn.de; http://www.ikp.uni-bonn.de/~mwo

Jens Zeidler; TU Chemnitz, Fakultät für Informatik; Arbeitsgruppe Künstliche Intelligenz; 9107 Chemnitz; tel.: 0371/5311392; jzei@informatik.tu-chemnitz.de; http://www.tu-chemnitz.de/~jzei

Jörg Zimmermann; Universität Bonn; Institut für Informatik III; Römerstraße 164; 53117 Bonn; tel.: 0228/73 4530

Regina Zücker; Rentenanstalt Swiss Life; CC/ITRD; General Guisan-Quai 40; Ch-8022 Zürich; tel.: 0041/1/284 4847; regina.zuecker@swisslife.ch; http://research.swisslife.ch