Kognitive Systeme II: Lernende Systeme

Schmid, Kitzelmann, Röglinger

SS 05

4. Übungsblatt

Besprechung und spätester Abgabetermin: 14.6.05 in der Übung

Organisatorisches

Auf dieses Übungsblatt gibt es 3 Punkte. Insgesamt können 10 Bonuspunkte für die Klausur erreicht werden. Für die Programmieraufgabe ist der Quelltext per e-mail an mich zu senden.

1 Sequential-Covering-Algorithmus (2 Punkte)

Es soll ein Sequential-Covering-Algorithmus zum Lernen einer Menge von propositionalen Regeln (Horn-Klauseln) der Form

$$\text{target\_attr} \leftarrow \text{attr}_1 = \text{val}_1, \ldots, \text{attr}_n = \text{val}_n$$

(zum Beispiel enjoy_sport $\leftarrow$ sky $=$ sunny, wind $=$ strong) in Prolog implementiert werden.

Im Gegensatz zum Algorithmus von Lecture 6 Folien 4 und 8 gilt die *Closed World Assumption*, d.h. der Körper jeder Regel besteht aus einer Konjunktion von Attribut/Wert-Paaren, die beschreibt, wann das Ziel-Attribut (z.B. enjoy_sport) *wahr* ist. Falls für eine gegebene Instanz (Attributvektor, der jedem Attribut ein Wert zuweist) *keine* Konjunktion wahr ist, ist das Ziel-Attribut *falsch*.

Repräsentiert werden die propositionalen Regeln als (Prolog-)Fakten der Form

$$\text{target\_attr}(\text{arg}_1, \ldots, \text{arg}_n)$$

wobei jeder Index ein Attribut bezeichnet und die arg_i Werte oder Variable sein können. Offensichtlich ist eine Anfrage

$$\text{target\_attr}(\text{Instanz})$$

genau dann wahr (Prolog antwortet *yes*), falls die Instanz für die Attribute, die in dem Fakt durch einen Wert spezifiziert sind, diesen Wert aufweisen und für alle anderen Attribute einen beliebigen Wert aufweisen.

Die Trainingsbeispiele sind vorklassifizierte Instanzen und haben die Form:

$$\text{expl}(\text{target\_attr}(\text{Instanz}))$$

bzw.

$$\text{nexpl}(\text{target\_attr}(\text{Instanz}))$$

expl repräsentiert ein positives Beispiel, d.h. target_attr soll bei gegebener Instanz wahr sein, nexpl repräsentiert ein negatives Beispiel, d.h. target_attr soll bei gegebener Instanz falsch sein.

Der Algorithmus:

1. Es soll eine Regel/ein Fakt nach der anderen gelernt werden (sequential covering / äußere Schleife).
2. Jede neu gelernte Regel soll möglichst viele der verbleibenden positiven Beispiele positiv klassifizieren (covern) und *kein* negatives Beispiel covern.
3. Alle Regeln zusammen sollen *alle* positiven Beispiele covern, d.h. die äußere Schleife terminiert, wenn nur noch negative Beispiele als noch nicht gecovert übrig sind.
4. Jede Regel/jeder Fakt soll so gelernt werden, dass beginnend mit der allgemeinsten Regel (alle Argumente sind Variable) ein Attribut nach dem anderen spezialisiert wird (innere Schleife), d.h. die jeweilige Variable durch einen Wert ersetzt wird. Eine Regel ist fertig gelernt, wenn *kein* negatives Beispiel mehr von ihr gecovert wird.
5. Es soll beim Lernen einer Regel immer dasjenige Attribut spezialisiert werden, für welches die so spezialisierte Regel möglichst viele positive (*mindestens aber eines!*) und möglichst wenige negative Beispiele covert. Das heißt, der *Score* zum Bewerten der Spezialisierungen ist: Anzahl positiver Beispiele, die gecovert werden, minus Anzahl negativer Beispiele, die gecovert werden.

Zur Implementierung:

1. Die Vorgabe ist `lpr.pl`, die Trainingsbeispiele befinden sich in `training.dat`.
2. Prozeduren zum Einlesen der Beispiele sowie einige weitere Hilfsprozeduren sind bereits implementiert.
3. Jede gelernte Regel soll mit `assert/1` zur Datenbasis hinzugefügt werden.
4. Alle Variable einer Regel werden während des Lernens durch die Konstante `undef` repräsentiert (es muss beim Lernen einer Regel also ein `undef` nach dem anderen durch einen konkreten Wert ersetzt werden). Bevor die gelernte Regel zur Datenbasis hinzugefügt wird, müssen die verbleibenden `undef` durch Variable ersetzt werden. Dazu dient die bereits implementierte Prozedur `introduce_vars/2`.

1.1 Handsimulation (0 Punkte)

Führt zur Vorbereitung eine Handsimulation des Algorithmus' für die Beispiele in `training.dat` durch. Notiert die jeweils ausgewählten Spezialisierungen. Das Lernen jeder Regel startet mit der allgemeinsten Regel: enjoy_sport(undef, undef, undef, undef, undef, undef).

1.2 Implementierung (2 Punkte)

Implementiert den Algorithmus.

1.3 Theoretische Fragen (0 Punkte)

1. Sowohl Entscheidungsbäume als auch Sequential-Covering-Algorithmen zum Lernen propositionaler Regeln lernen Konzepte $c : X \to \{0,1\}$, wobei X die Menge aller möglichen Attributvektoren für eine fixierte Menge von Attributen und deren Wertebereichen ist. Entscheidungsbäume sind in ihrer Hypothesensprache vollständig, d.h. es kann prinzipiell jedes Konzept c repräsentiert werden. Gilt dies auch für Mengen propositionaler Regeln?
2. Hat der beschriebene Algorithmus einen Language-Bias? Hat er einen Search-Bias? Beschreiben Sie den jeweiligen Bias.
3. Im Algorithmus aus Lecture 6 Folien 4/8 wird beim Lernen einer Regel jeweils die Spezialisierung ausgewählt, die die höchste Performance aufweist, wobei die Performance jeweils die negative Entropie der von einer Spezialisierung gecoverten Beispiele ist. Warum ist dieses Performance-Maß für den Algorithmus auf den Folien vernünftig, für den hier beschriebenen und zu implementierenden Algorithmus jedoch nicht?

2 Backpropagation Handsimulation (1 Punkt)

Betrachtet ein KNN mit einem Input-, einem Hidden- und einem Output-Layer. Es gibt zwei Input-Units a und b, eine Hidden-Unit c und eine Output-Unit d. Dieses Netz hat fünf Gewichte, $(w_{ca}, w_{cb}, w_{c0}, w_{dc}, w_{d0})$, wobei w_{x0} das Gewicht der Schwelle für Unit x repräsentiert. Die Gewichte seien mit

$$(0.1, 0.1, 0.1, 0.1, 0.1)$$

initialisiert. Betrachtet nun folgende beiden Trainingsbeispiele:

Beispiel	a	b	d
1.	1	0	1
2.	0	1	0

Gebt an, welche Werte die Gewichte nach der ersten und nach der zweiten Trainingsiteration des Backpropagation-Algorithmus haben. Nehmt dazu eine Lernrate $\eta = 0.3$ und ein Momentum $\alpha = 0.9$ an.