Maschinelles Lernen: Symbolische Ansätze

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2013/2014
Musterlösung für das 2. Übungsblatt

Aufgabe 1 Konzepte, Regeln und Hypothesenräume

Gegeben sei ein Beispielraum, der durch n binäre Attribute aufgespannt wird. Hypothesenraum sind alle möglichen Regeln, die sich durch Konjunktionen von Tests der Form *Attribut* = *Wert* ergeben.

a) Wie viele mögliche Konzepte gibt es?

Lösung: Ein Konzept ist definiert als eine Untermenge aller möglichen Objekte (Folie 2). Bei bool'schen Attributen sind die Objekte boolesche Vektoren. Hat man n binäre Attribute, so ergeben sich 2^n verschiedene mögliche Objekte. Da jedes Objekt in der Untermenge enthalten oder nicht enthalten sein kann erhält man 2^{2^n} mögliche Konzepte. Als Beispiel nochmal die Fälle $n = 1$ und $n = 2$ (1 kodiert enthalten / 0 kodiert nicht enthalten):

1. $n = 1 \rightarrow 4$ mögliche Konzepte

Attributwert des Attributs A_1	Teil des Konzeptes
true	0
false	0

Attributwert des Attributs A_1	Teil des Konzeptes
true	0
false	1

Attributwert des Attributs A_1	Teil des Konzeptes
true	1
false	0

Attributwert des Attributs A_1	Teil des Konzeptes
true	1
false	1

2. $n = 2 \rightarrow 16$ mögliche Konzepte C_i

A_1	A_2	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}
false	false	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
false	true	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
true	false	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
true	true	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

b) Wie viele mögliche Regeln gibt es?

Lösung: Ein Attribut kann in einer Regel den Wert „false“ und den Wert „true“ annehmen oder in der Regel überhaupt nicht vorkommen.

$\Rightarrow 3^n + 1$ mögliche Regeln.

Dazu kommt noch die Regel mit dem Body ”false” (deswegen $+1$).

In der Tupel-Notation aus der Vorlesung gibt es für jede Stelle die Möglichkeiten *true*, *false* und *?*, sowie die Theorie, die an allen Stellen $\emptyset$ ist.

c) Unter der Voraussetzung, daß das zu lernende Konzept im Hypothesenraum darstellbar ist, wie viele Fehler kann der Algorithmus Find-S beim Lernen dieses Konzepts maximal machen?

[**Anm:** *Ein Fehler ist hier ein (Trainings-)Beispiel, das von der momentanen Theorie falsch klassifiziert wird.*]

Lösung: Wenn das zu lernende Konzept im Hypothesenraum enthalten ist, ist es konsistent (deckt also keine negativen Beispiele ab) und komplett (deckt also alle positiven Beispiele ab). Der Algorithmus FindS beginnt mit der speziellsten Hypothese (der leeren Menge {}). Auf dem Weg zum Zielkonzept können keine negativen Beispiele abgedeckt werden. Da der Algorithmus seine Hypothese nur bei einem positiven Beispiel verändert, kann er während des Lernprozesses maximal so viele Fehler machen, wie es positive Beispiele gibt. Unterscheiden sich nun die Attributwerte der positiven Beispiele immer genau an einer Stelle und gibt es mehr positive Beispiele als Attribute in der Trainingsmenge, macht der Algorithmus nur so viele Fehler, wie es Attribute gibt, da er nach der letzten Änderung bereits die generellste Hypothese gefunden hat. Da er beim ersten Beispiel immer falsch liegt, kommt noch ein Fehler hinzu.

$\Rightarrow min\{|Attribute| + 1, |positiveBeispiele|\}$

Hierzu noch ein Beispiel:

Fall 1: *Der maximale Fehler entspricht der Anzahl der* $Attribute + 1$

A_1	A_2	A_3	Klasse	Hypothese	Fehler?
-	-	-	-	{}	-
false	false	false	+	(false,false,false)	ja
true	false	false	+	(?,false,false)	ja
true	true	false	+	(?,?,false)	ja
false	false	true	+	(?,?,?)	ja
true	true	true	+	(?,?,?)	nein
false	true	true	+	(?,?,?)	nein

$min\{4, 6\} = 4$

Fall 2: Der maximale Fehler entspricht der Anzahl der positiven Beispiele

A_1	A_2	A_3	A_4	Klasse	Hypothese	Fehler?
-	-	-	-	-	{}	-
false	true	true	false	+	(false,true,true,false)	ja
true	false	false	true	+	(?,?,?,?)	ja
false	false	false	true	+	(?,?,?,?)	nein

$min\{5, 3\} = 3$, *wobei der Algorithmus hier nur 2 Fehler macht*

d) Gegeben sei ein Hypothesenraum, der nur *Disjunktionen* von binären Attribut-Wert-Paaren erlaubt. Eine gültige Regel wäre also z.B.

if $(att_i = t)$ **or** $(att_j = f)$ **or** $(att_k = f)$ **then** $+$

Überlegen Sie sich Verallgemeinerungs- und Spezialisierungsvorschriften für diesen Hypothesenraum und geben Sie einen geeigneten Lernalgorithmus an.

Lösung: Als Verallgemeinerungsvorschrift fügt man nun Bedingungen hinzu und als Spezialisierungsvorschrift entfernt man Bedingungen.

Die Rollen der beiden Algorithmen FindS und FindG vertauschen sich damit.

Da FindG Bedingungen hinzufügt (spezialisiert), was nicht eindeutig ist (siehe Folie 21), sollte man zum Lernen den Algorithmus FindS verwenden. Da die Rollen aber in diesem Hypothesenraum wie oben erwähnt vertauscht sind, ist der geeignete Lernalgorithmus FindG.

Aufgabe 2 Find-S und Find-GSet

Sie wollen den Find-S Algorithmus auf numerische Daten anwenden, indem Sie Intervalle definieren.

a) Wie sieht das spezifischste und generellste Element der Sprache aus, wenn Sie offene Intervalle verwenden?

Lösung:
spezifischstes Element: $(x, x) = \{\}$ mit $x \in \Re$
generellstes Element: $(-\infty, \infty)$

b) Wie sieht das spezifischste und generellste Element der Sprache aus, wenn Sie geschlossene Intervalle verwenden?

Lösung: es gibt kein eindeutiges spezifischstes und auch kein generellstes Element; hier muss man sich virtuelle Elemente vorstellen, die einerseits der leere Menge entsprechen und andererseits alle möglichen Werte abdecken.

c) Finden Sie eine passende Generalisierungsvorschrift und simulieren den Algorithmus Find-S auf folgenden Beispielen:

A1	Klasse
0.5	−
1.0	+
2.1	−
0.8	−
1.5	+
1.8	+

Lösung: Als Intervall verwenden wir geschlossene Intervalle. Als Generalisierungsvorschrift verwenden wir die Aktualisierung einer der beiden Intervallgrenzen (abhängig vom Wert des Beispiels).

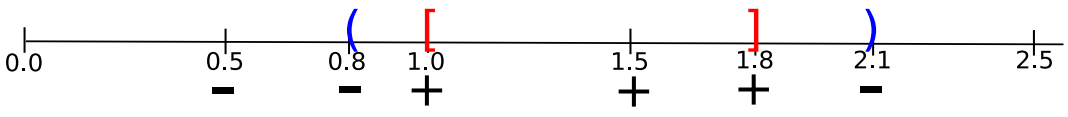
$S_0 = \{\}$, $S_1 = S_0$, $S_2 = [1.0, 1.0]$, $S_3 = S_2$, $S_4 = S_3$, $S_5 = [1.0, 1.5]$, $S_6 = [1.0, 1.8]$

d) Finden Sie eine passende Spezialisierungsvorschrift und simulieren Sie den Algorithmus Find-GSet auf denselben Beispielen.

Als Intervall verwenden wir offene Intervalle. Als Spezialisierungsvorschrift verwenden wir eine Aufspaltung des aktuellen Intervalls in beide mögliche Intervalle.

$G_0 = (-\infty, \infty)$ — *Initialisiere mit generellstem Intervall*
$G_1 = (-\infty, 0.5), (0.5, \infty)$ — *spalte das Intervall auf*
$G_2 = (0.5, \infty)$ — *entferne* $(-\infty, 0.5)$*, da das positive Beispiel nicht abgedeckt ist (*$1.0 \notin (-\infty, 0.5)$*)*
$G_3 = (0.5, 2.1)$ — *1. aufspalten:* $(0.5, 2.1)$*,* $(2.1, \infty)$ *und 2. nur die hinzufügen, die alle vorherigen positiven Beispiele abdecken (letzter Schritt des Algorithmus), da* $1.0 \notin (2.1, \infty)$ *nur* $(0.5, 2.1)$
$G_4 = (0.8, 2.1)$ — *1. aufspalten:* $(0.5, 0.8)$*,* $(0.8, 2.1)$ *und 2.* $(0.5, 0.8)$ *entfernen, da* $1.0 \notin (0.5, 0.8)$
$G_5 = G_4$
$G_6 = G_5$

e) Skizzieren Sie beide Lösungen und vergleichen Sie die Allgemeinheit.



Lösung: Das Ergebnis von FindS ist die speziellste vollständige und konsistente Theorie (die Theorie, die gerade noch kein positives Beispiel ausschließt). In der Grafik ist diese durch das rot markierte, geschlossene Intervall gekennzeichnet. Das Ergebnis von FindGSet ist die allgemeinste vollständige und konsistente Theorie (die Theorie, die gerade noch kein negatives Beispiel einschließt), welche in der Grafik durch das blau markierte, offene Intervall repräsentiert ist.