Gedächtnisprotokoll Diplomprüfung Praktische Informatik

Prüfer: Prof. Seidl

Themen

- Algorithmen des Data Mining (nach Prof. Seidl)
- Artificial Intelligence (nach Prof. Lakemeyer; vereinbarter Prüfungsumfang: entspr. Kapitel im Russel/Norvig)
- Logikprogrammierung (nach Prof. Indermark; vereinbarter Prüfungsumfang gemäß Skript zur Vorlesung)

Ich habe mich bemüht, alle Fragen sinngemäß wiederzugeben; natürlich lautete die Formulierung in der Regel anders. Auch meine Antworten sind stark gekürzt. Speziell zum Teil „Logikprogrammierung“ hatte ich nachher die Fragen nur noch ungenau im Gedächtnis. Da es aber noch keine anderen Protokolle zu Prüfungen von Prof. Seidl in diesem Bereich gibt, kann ich Euch nur bitten, selbst welche zu erstellen ...

Data Mining

Wo steht Datamining, oder genauer gefragt was ist der KDD Prozess?

Daten sammeln, integrieren, filtern, zusammenfassen, dann das eigentliche Mining (z.B. Clustern, Klassifikation, Assoziationsregeln), dann Visualisieren (damit auch Manager das verstehen), dann entscheiden!
. . . ja, aber auch Data Mining-Experten benötigen Visualisierungen

Klassifikation: Was ist das?

Überwachtes Lernen, d.h. meine Daten teilen sich in Klassen ein, die kann ich aus den „Messdaten“ der Objekte allerdings nicht direkt bestimmen. Zum Lernen habe ich also Trainingsdaten

mit Klasse (ggf. manuell bestimmt) gegeben, mit deren Hilfe ein Klassifikator bestimmt wird; der nimmt dann später jeweils einen Datensatz (ohne Klasse) entgegen und gibt eine Klasse zurück.

Entscheidungsbäume: Wie funktionieren die?

Baum (aufgem)., habe Datensatz, gehe von Wurzel nach unten durch, bekomme jew in inneren Knoten vom Baum eine Frage zu einem Attribut gestellt (Boolsch: *ja/nein?*, kategorisch: *Ist gleich?/Ist in?*, numerisch: *Ist grösser als?*), gehe dann je nach Antwort im Baum weiter. Blätter sind mit Klasse beschriftet → Antwort.

Wie trainiert man einen Baum?

Habe Menge von Trainingsdaten mit Klasse vorliegen (Wurzel: Alle Daten), schaue mir dann jedes „noch mögliche" Attribut an und mögliche Fragen (s.o.), bestimme dann nach bestimmten Kriterium (Gini-Index, Transinformation $\hat{=}$ Information Gain) „bestmögliche" Frage zur Segmentierung der Trainingsdaten nach Klasse, unterteile dann die Trainingsdaten entsprechend, mache weiter. Abbruch bei:

1. Alle Trainingsdaten gleiche Klasse: Wunderbar – gebe Klasse zurück
2. Trainingsdaten unterschiedlich, aber keine Frage mehr überig (d.h. unterscheiden sich nicht in Attributen) → nichts weiteres möglich, also gib häufigste Klasse zurück
3. (optional): Fast alle Trainingsdaten in einer Klasse → Pruning möglich

Pruning: Was ist das? Wie?

Entscheide entweder schon beim Erstellen des Baumes, oder hinterher („Postpruning"), einen Teilbaum wegzulassen und durch ein Blatt zu ersetzen. (Habe ich auf χ^2-Test bzw. Verhältnis Klassifikationsfehler-Fehler zu Unterbaumgrösse hingewiesen? Weiß ich nicht mehr ...)

Und warum macht man das?

1. Baum nicht zu groß, sondern übersichtlich, verständlich
2. Overfitting: Baum trainiert sich auf Artefakte in Trainingsmenge, z.B. „gerade Schuhgrösse ⇒ guter Kunde"

Nearest-Neighbour-Klassifikator: Was ist das?

Erklärt, dabei Fehler im Ester/Sander erwähnt (hätte ich besser nicht gemacht, hat nur Zeit gekostet). Varianten erwähnt: 1-NN, k-NN, k-NN mit Wichtung nach Abstand.

Am Beispiel: `(B  x A  B)`

- 1-NN: A

- 3-NN ohne Wichtung: B
- 3-NN mit entsprechender Wichtung: A

Wonach wichtet man?

- Nach Abstand (invers)
- nach A-Priori-Wahrscheinlichkeit der Klassen (wenn in Trainingsdaten nicht gegeben)
- wenn Fehlerfunktion nicht symmetrisch (Beispiel Pre-Safe in Autos:
  $P(Unfall) \ll P(\neg Unfall)$,
  aber $Kosten(Unfall|\neg PreSafe) \gg Kosten(\neg Unfall|PreSafe)$)

Association Rules: Was ist das?

Einfachster Fall: Eindimensionale Transaktionsdaten, z.B. Einkäufe.

Beispiel angegeben: kauft(Bier) $\Rightarrow$ kauft(Windeln) $[support = \ldots, confidence = \ldots]$

Erklärung Support $= p(\text{Bier}, \text{Windeln})$, Konfidenz = Zuverlässigkeit $= p(\text{Windeln}|\text{Bier})$

Da gibt es Algorithmen zur Bestimmung. Was ist APRIORI?

Zunächst: APRIORI findet häufige Itemsets, daraus kann man dann später die Regeln gewinnen.

Vorgehensweise:

1. alle häufigen (Erklärung: MinSupport) einelementigen Itemsets finden

$k+1$ Zur Bestimmung aller Itemsets der Größe $k+1$: Selbst-Join auf k-Itemsets, nehme dann alle $k+1$-IS, für die jew alle k-Unter-IS (die also durch Weglassen eines Items entstehen) in der Ebene k bereits häufig waren.
Beispiel $k=3$: {A,B,C ; A,B,E ; A,C,E ; A,C,E ; B,C,E; B,D,E } $\rightsquigarrow$ $k=4$: {A,B,C,E}.
Dann (wichtig) prüfen, ob die $k+1$ IS auch häufig sind (Datenbank)

Muss man wirklich prüfen?

Ja: Beispiel: 100 Kunden kaufen nur A,B,C, 100 kaufen nur A,B,E usw., aber **keiner** kauft A,B,C,E.

Wieviele Durchgänge durch die Datenbank braucht man?

Bei guter Programmierung, ganze Ebene im Speicher: Einen pro Ebene

Und wie macht man daraus Assoc Rules (am Bsp)?

Erste Antwort: A,B,C⇒E ; A,B,E⇒C ; A,C,E⇒B ; B,C,E⇒A

Da fehlen aber noch welche!

„Echt?“

Es können rechts auch mehrere stehen. Wieviele gibt es dann?

(überlegt) 2^n Kombinationen, aber links und rechts immer mindestens eins $\Rightarrow 2^n - 2$.

Und wie bestimmt man Support und Konfidenz?

Support haben wir schon bei APRIORI für jedes IS bestimmt, Konfidenz: $P(XY) = P(X,Y)/P(X)$, also Bsp. $P(E|A,B,C) = \frac{N(A,B,C,E)}{N(A,B,C)}$ wobei die $N(\cdot)$ als der entspr. Support bei $k = 4$ bzw. $k = 3$ berechnet wurde.

Müssen wir dafür noch einmal in die Datenbank schauen?

Nein, wir haben noch alle Informationen im Speicher.

Artificial Intelligence

In der AI haben wir Suchprobleme. Wie sehen die aus?

- Zustandsmenge
- Menge von Startzuständen (Idealfall: genau einen)
- Operatoren (= Übergänge von Zustand zu Zustand)
- Pfadkostenfunktion (erklärt)
- Goal Check, d.h. Funktion, die uns sagt, ob wir in einem Endzustand sind (i.A. Black Box! Nur in Sonderfällen wissen wir mehr)
- Wenn wir Glück haben: Heuristik, d.h. Abschätzung der Kosten bis zum Ziel $\rightsquigarrow$ informierte Suche statt uninformierte Suche möglich.

Was gibt es an uninformierten Suchverfahren?

Tiefensuche, Breitensuche, Uniform Cost, DLS, Iterative Deepening, Bidirectional Search

Was hat die Tiefensuche für Eigenschaften?

Nicht vollständig (bei unendlichen Pfaden), nicht optimal

Breitensuche: Kosten, Komplexität?

(erklärt), grundsätzlich vollständig, optimal wenn Pfadkosten monotone Funktion der Pfadlänge

Iterative Deepening: Wie funktioniert das?

DLS mit $d = 1$, DLS mit $d = 2$, ... bis Lösung gefunden. Vollständig bei (s.o.)

Ist das nicht zu teuer?

Nein, sofern Branching Factor $b \geq 2$ ($b = 1$ Sonderfall, kann vernünftig behandelt werden): Zeit BFS $= O(b^d)$ IDS $= O(\frac{b}{(b-1)} \cdot b^d)$, für $d = 2$ also das Doppelte, für große b praktisch das Gleiche.

Aber: Platz BFS $= O(b^d)$ (üblicherweise zu viel), IDS $= O(b \cdot d)$ (erklärt)

Informierte Suche, genauer gesagt A*. Was ist das?

Erklärt: In jedem Knoten x optimistische Heuristik (erklärt) $h(x)$, Pfadkosten bisher $g(x)$. Berechne $f(x) := g(x) + h(x)$. Das sind dann die minimalen Kosten eines Pfades zum Ziel, der mit dem entspr. Pfad anfängt. Expandiere dann jeweils den Knoten mit geringstem f, sortiere Kinder entspr. ein.

(Nochmal nähere Erläuterungen zur optimistischen Heuristik)

(Wir waren uns nicht völlig sicher, ob bei nichtoptimistischer Heuristik A* lediglich nicht-Optimal oder sogar unvollständig wird. Vermutlich letzteres. Wer Punkte schinden möchte: Vor der Prüfung Beweis nachsehen, erwähnen ...)

Kann man BFS als A* beschreiben?

Ja, wenn wir $h(x) = 0$ setzen.

Kann man DFS als A* beschreiben?

Eigentlich nicht, da DFS unvollständig und nichtoptimal. Vielleicht doch mit entspr. „falscher“ Heuristik. Mit Pfadkostenfunktion $g(x) = 0$ und $h(x) = 0$ geht es, wenn in die Queue „möglichst vorne“ einsortiert wird. Hängt also von der Implementierung ab ⇒ eigentlich vermutlich nicht.

Spiele: Was kann man da machen?

Erklärt: Zwei Spieler, Nullsummenspiel, komplett einsichtig, ohne Zufallselement. Daher Bewertung positiv wenn Spieler 1 gewinnt, Bewertung negativ wenn Spieler 2 gewinnt. Daher nennen

wir die Spieler MAX und MIN. Suchbaum MAX/MIN aufgezeichnet, Ebenen benannt, abwechselndes Spielen (nicht „gleichzeitig“). Minimax erklärt.

Kann man das optimieren?

Ja: alpha-beta-Suche erklärt. Formulierung: „MAX hat da schon 5 zurückbekommen. Er sagt also dem nächsten MIN-Knoten: Gib mir dein Minimum, aber was kleineres als 5 brauchst du gar nicht zu versuchen – dann nehm ich *den* Zug sowieso nicht. Wenn MIN jetzt 5 oder kleiner sieht, kann es die Suche abbrechen – denn dann würde es 5 oder kleiner zurückgeben, so dass MAX das eh nicht nimmt. Umgekehrt genauso $\Rightarrow \alpha, \beta$. “

Was machen wir, wenn wir den Suchbaum nicht bis zu den Lösungen konstruieren können?

Bewertungsfunktion der Spielsituation vorgeben (Utility). Beispiel Schach: Punkte für Bauern, ..., Extrabewertung für Stellungen.

Logikprogrammierung

Woraus besteht denn ein Logikproblem?

Hier hatte ich Schwierigkeiten, die Frage zu verstehen. Im Endeffekt wollte er im Laufe der nächsten Fragen folgendes erklärt haben:

- Syntax von PL1:
  - Atome:
    - true, false, $P(t_1, \ldots, t_n)$
    - $\vee, \wedge, \neg$, ($\rightarrow, \leftarrow, \leftrightarrow$ als syntaktischer Zucker)
  - Terme:
    - const
    - $f(t_1, ..., t_n)$
- Semantik/Modell/Interpretiation:
  - Menge
  - Konstanten
  - Funktionen („interpretiert wie im richtigen Leben“)
  - Prädikate

Was sind Hornklausen?

$\{L_1 \vee \ldots \vee L_n\}$, max. ein Literal positiv (d.h. nicht negiert)

- nur ein Positives: Fakt
- ein Positives, Rest negativ: Definite Klausel, Regel
- nur Negative: Anfrageklausel (Warum: Auf Später verwiesen)

Warum Hornklauseln?

SLD-Resolution als effizientes Entscheidungsverfahren zur Erfüllbarkeit möglich

Wofür steht SLD?

- „Selection" → Auswahlverfahren (immer das Erste, links → rechts)
- „Linear": Lineare Resolution (grob (!) erklärt).
- „Definite Clauses": Wir resolvieren immer mit definiten Klauseln.

Prolog: Sind solche Logikprogramme. Aber z.B.: Cut Operator. Was ist das?

(das ging mir etwas schnell. Ich wollte eigentlich über Unterschiede Lp/Prolog darauf kommen. Aber ...) Schneidet Suchbaum an der Stelle ab. Zum Beispiel

- um unendliche Zweige abzubrechen (gut)
- um Berechnungen, die nicht zu Lösungen führen werden, abzubrechen (gut)
- kann aber auch Semantik ändern (schlecht!)

Beispiel (allerdings erwähnt, dass das eben schon ein problematisches ist).

```
max(A,B,B) :- B >= A, !.
max(B,A,B).
```

Beispiele: `max(2,5,X)` und `max(5,2,X)`

Prolog ist einer deklarative Programmiersprache: Ja oder Nein?

Jein. Prinzipiell zwar deklarativ (d.h. wir beschreiben Lösung und mögliche Schritte). generate-and-test auch durchaus üblich. Aber wir müssen schon irgendwie vorgeben, wie die Lösung zu finden ist – insbes läuft g&t bei großer Anzahl mögl. Kandidaten sonst ewig. Zudem durch Auswertereihenfolge auch sehr „imperativer" Programmierstil möglich.

Dann wollte er wohl noch auf etwas raus, das in meiner Vorlesung nicht dran kam (?). Ich hatte das so verstanden, dass er auf $\mathcal{D}[\![\mathcal{L}_\mathcal{P}, G]\!] = \mathcal{P}[\![\mathcal{L}_\mathcal{P}, G]\!] = \mathcal{F}_\mathcal{P}[\![\mathcal{L}_\mathcal{P}, G]\!]$ hinauswollte.

- $\mathcal{D}[\![\mathcal{L}_\mathcal{P}, G]\!]$ beschrieben. (Semantische Folgerung, $\mathcal{L}_\mathcal{P} \models H$)
- $\mathcal{P}[\![\mathcal{L}_\mathcal{P}, G]\!]$ beschrieben (Resolutionsfolgerung, $H = G\,\underline{\text{sub}}$ für alle $\langle G, [] \rangle \vdash^*_{\mathcal{L}_\mathcal{P}} \langle \square, \underline{\text{sub}} \rangle$

- $\mathcal{F}_{\mathcal{P}}[\![\mathcal{L}_{\mathcal{P}}, G]\!]$ mangels Zeit nur kurz beschrieben ($trans_{\mathcal{L}_{\mathcal{P}}}$, Fixpunkt, H Grundsubstitution von G mit Grundtermen aus dem Fixpunkt).

Bei dem Logikprogrammierung-Teil hatte ich etwas Schwierigkeiten, zu erkennen, worauf Prof. Seidls Fragen abzielten (beim Lernen hatte ich hier eine etwas andere Prüfungs-Struktur erwartet). Es hilft wohl, sich nochmal genauestens die Definition einer Logik mit allen zugehörigen Begriffen anzuschauen).

Alles in allem empfand ich die Atmosphäre als angenehm und entspannt (auch wenn mir der Vergleich fehlt – das war immerhin meine erste Diplomprüfung).