Alexander Hinneburg,
hinneburg@informatik.uni-halle.de

26.11.2005

Übung zur Vorlesung „Data Mining in Datenbanken“

Übungsblatt 6[1]

Auswahl der Attribute
Abgabe am 1.12.2005

Beim Finden von Assoziationsregeln spielt das Finden von häufig auftretenden Item-Mengen eine wichtige Rolle. Bei einigen Anwendungen sind diese häufigen Item-Mengen selbst von Interesse. Die Eingabe für dieses Problem besteht aus einer Menge von Item-Mengen, die oft Transaktionsmenge genannt wird. Der Name kommt daher weil eine sehr bekannte Anwendung das Finden von Produkten ist, die im Supermarkt häufig zusammen gekauft. Die Menge der Produkte, die zusammen auf einem Kassenzettel stehen, heißt Transaktion. Ein Beispiel sei

Transaktion 1 Brot, Milch, Saft, Bier, Tomaten, Salat, Chips

Transaktion 2 Bier, Chips, Pizza

Transaktion 3 Brot, Milch, Salat, Gurke, Joghurt

Transaktion 4 Salat, Tomaten, Gurke, Bier, Chips

Eine Item-Menge wird als häufig bezeichnet, wenn sie Teilmenge von mehr als *minsupport* Transaktionen ist.
Alternativ kann man sich die Eingabe als Binärmatrix vorstellen, bei die Vereinigungsmenge I der Items aller Transaktionen die Spalten und die Transaktionen selbst, die Zeilen bilden. Ein Eintrag (i, j) in der Matrix ist Eins, wenn das j-te Item in der i-ten Transaktion vorkommt. Diese Matrix bei den meisten Anwendungen sehr groß und ist sehr dünn mit Einsen besetzt, daher wird sie oft nicht direkt als Eingabe genutzt.
Der A-priory Algorithmus ist eine Möglichkeit zum Finden aller häufigen Item-Mengen. Der Algorithmus nutzt die Monotonie-Eigenschaft von häufigen Item-Mengen: Jede Teilmenge einer häufigen Item-Menge muß auch eine häufige Item-Menge sein. Sei C_k die Menge der Item-Mengen der Größe k, die Kandidaten für häufige Item-Mengen sind, und L_k die Menge der häufigen Item-Mengen der Größe k. Der abstrakte Pseudo-Kode des Algorithmus ist

1: L_1 = Menge der häufigen Item-Mengen der Größe 1
2: **for** ($k = 1$; $L_k \neq \emptyset$; k++) **do**
3: C_{k+1} = erzeuge Menge der Kandidaten aus L_k
4: **for all** Transaktionen t in der Eingabe **do**
5: Erhöhe die Zähler aller Kandidaten in C_{k+1}, die in t enthalten sind
6: **end for**
7: L_{k+1} = alle Kandidaten aus C_{k+1} deren Zähler $\geq$ *minsupport* ist
8: **end for**

[1] Achten Sie auch auf die Form Ihrer Lösungen, z.B. daß jedes Bild eine Bildunterschrift hat. Bitte geben Sie keine losen Blätter ab. Um die Übungen in angemessener Zeit zu diskutieren, bereiten Sie neben Ihrer Lösung ein kleine Präsentation vor, mit der Sie dann Ihre Ergebnisse zeigen können.

9: Gegebe $\bigcup_{i=1}^{k} L_i$ zurück

Der Schritt 3 besteht aus Join- und Prune-Schritten. Bei Join-Schritt wird aus zwei jeweils häufigen Item-Mengen der Größe k, die sich nur in einem Item unterscheiden, ein Kandidat für eine häufige Item-Menge der Größe $k+1$ erzeugt wird. Beim Prune-Schritt wird für jede k-elementige Teilmenge eines Kandidaten aus C_{k+1} geprüft, ob sie in L_k ist. Falls der Test für eine Teilmenge fehlschlägt, kann der Kandidat gelöscht werden.

Aufgabe 1 Entwickeln Sie einen ausführlichen Pseudo-Kode für den A-priory ALgorithmus unter der Annahme, daß Sie eine Suchdatenstruktur zur effizienten Verwaltung von Schlüssel-Wert Paaren (z.B. Hash, Dictionary, Map, Suchbaum, Heap, ...) und Sortieren nutzen können. Die Eingabe soll als Transaktionsliste erfolgen; in einer Zeile stehen nur die Items, die in der Transaktion enthalten sind. Der Algorithmus soll den Join und Prune Schritt implementieren. Der Wert *minsupport* ist ein Parameter des Algorithmus. Die häufigen Item-Mengen brauchen in keiner speziellen Reihenfolge ausgegeben werden.

Aufgabe 2 In welchen Iterationen der äußeren Schleife (Schritte 2-8) wirkt der Prune-Schritt nicht? Begründen Sie Ihre Antwort.

Aufgabe 3 Wenn eine Item-Menge X der Größe 6 häufig ist, mindestens wieviele weitere häufige Item-Mengen findet der A-priory Algorithmus bevor er X findet?

Aufgabe 4 Der A-priory Algorithmus durchsucht den Teilmengenverband von I (siehe Einleitung) angefangen von den ein-elementigen Mengen aufsteigend bis zu I. Zeichnen Sie für $|I| = 50$ ein Diagramm mit k auf der x-Achse und der Anzahl der theoretisch möglichen Teilmengen auf der y-Achse. Zeichnen Sie die y-Achse mit einer logarithmischen Skale.

Aufgabe 5 Die Anzahl t_k der theoretisch möglichen Teilmengen der Größe k ist eine relativ schlechte obere Schranke für die Anzahl der häufigen Item-Mengen der Größe k, weil nur $|I|$ einfließt. Entwickeln Sie eine einfache, genauere obere Schranke s_k, bei der die Eingabe und *minsupport* stärker berücksichtigt werden, die sich aber mit nur einem Lauf über die Daten berechnen läßt. Wie hoch ist der Laufzeit und Platzaufwand um diese neue Schranke für alle k zu berechnen? Gegeben seien Supermarktdaten und $minsupport = 2\%$, in welcher Iteration der äußeren Schleife (Schritte 2-8) des A-priory Algorithmus erwarten Sie, daß t_k, s_k und die tatsächliche Anzahl der häufigen Item-Mengen nahe zusammen liegen? Begründen Sie kurz Ihre Antwort.

Aufgabe 6 Implementieren Sie den A-priory Algorithmus mit möglichst kleiner Laufzeit und Haupspeicherverbrauch. Vermeiden Sie große Datenstrukturen doppelt im Hauptspeicher zu halten. Testen Sie Ihre Implementierung an den nominalen Wetterdaten mit $minsupport = 2$, die mit dem WEKA Paket mitgeliefert werden. Bestimmen Sie die Anzahl der häufigen Item-Mengen der Größe 1, 2, 3, 4, 5 für $minsupport = 2$ und 3.

Aufgabe 7 Stellen Sie sich die Eingabe statt einer Transaktionsmenge als Binärmatrix vor. Beschreiben Sie kurz wie eine häufige Item-Menge übertragen auf die Binärmatrixdarstellung charakterisiert werden kann. (Zusatz: wie könnte man ausgehend von dieser Darstellung häufige Item-Mengen verallgemeinern, um Daten mit Fehlern robuster untersuchen zu können?).