Praktikum zu Wissensbasierte Methoden

Vorgesehen für die Praktikumssitzungen am 29. März und 1. April 2016

Versuch Sie zunächst, Aufgabe 1 vollständig zu bearbeiten. Bei ausreichender Zeit empfiehlt es sich, auch Aufgabe 2 zu bearbeiten.

1. Aufgabe: In dieser Aufgabe sollen Sie den *Apriori*-Algorithmus zum Finden von Assoziationsregeln implementieren. Verwenden Sie bei der Implementation *Matlab*, und richten Sie sich nach der Vorlesung bzw. nach dem Skript zur Vorlesung. Zu klären ist zunächst, wie man *Item*-Mengen und insbesondere Transaktionen in *Matlab* darstellt. Verwenden Sie dazu Bitvektoren und gehen wie folgt vor:

   - Man legt eine feste Numerierung der vorhandenen *Items* fest (mit $N =$ Itemzahl):

     $$I_1,\ I_2,\ I_3, \ldots,\ I_N$$

   - Man stellt eine *Item*-Menge (und damit auch eine Transaktion) durch einen Zeilenvektor der Länge N dar, dessen i-te Komponente genau dann 1 ist, wenn das i-te *Item* enthalten ist, und sonst 0 ist. Beispiel: Bei insgesamt $N = 6$ vorhandenen *Items* enthält die folgende *Item*-Menge genau das erste, zweite und fünfte *Item*:

     $$Z \ = \ [1,\ 1,\ 0,\ 0,\ 1,\ 0]$$

   Enthält die Datenbasis n Transaktionen, so wird sie durch eine $n \times N$ Matrix dargestellt, deren Zeilen den erfaßten Transaktionen entsprechen.

   (a) Implementieren Sie den ersten Teil des *Apriori*-Algorithmus (einschließlich des *AprioriGen*-Algorithmus), der zum Finden häufiger *Item*-Mengen dient. Hinweise:

   - Zum Zählen, in wievielen Transaktionen der Datenbasis T eine *Item*-Menge vorkommt, können Sie die folgende Funktion verwenden, müssen aber dabei deren Funktionsweise erklären können:

     ```
     function a=Anzahl(T,z)
     a=sum((T*z')==sum(z));
     ```

   - Beim *AprioriGen*-Algorithmus muß geprüft werden, ob sich zwei *Item*-Mengen gleicher Größe nur durch die beiden jeweils an der letzten Stelle stehenden *Items* unterscheiden. Hierzu können Sie wiederum die folgende Funktion verwenden, müssen aber dabei auch deren Funktionsweise erklären können:

     ```
     function a=AnfangGleich(z1,z2)
     k=sum(z1);
     if k==0; a=1;return; end;
     s=min(find(z1,1,'last'),find(z2,1,'last'))-1;
     if sum(and(z1(1:s),z2(1:s)))==k-1
        a=1;
     else
        a=0;
     end
     ```

(b) Vervollständigen Sie den *Apriori*-Algorithmus, indem Sie den *AprioriReg*-Algorithmus zum Finden der Regeln mit hohem Konfidenzwert implementieren. Beachten Sie dabei, daß Sie den zuvor implementierten ersten Teil des *Apriori*-Algorithmus mit geringen Anpassungen und den *AprioriGen*-Algorithmus direkt wiederverwenden können.

Verwenden Sie zum Testen Ihrer Implementation das Beispiel aus der Vorlesung bzw. aus dem Skript. Zusätzlich finden Sie am bekannten Ort noch zwei Testdatenbestände (in den Dateien "wbsa0206_20.dat" sowie "wbsa0206_100.dat"); wenn Sie diese mit den Werten $minsupp = 0.4$ und $minconf = 0.7$ testen, sollten sich einmal 16 und einmal 89 Regeln ergeben.

2. Aufgabe: Von einhundert Schwertlilien wurden folgende Werte[1] erhoben (alles in cm):

Kelchblatt-länge	Kelchblatt-breite	Blütenblatt-länge	Blütenblatt-breite
5.1	3.3	1.7	0.5
6.1	3.0	4.6	1.4
4.8	3.1	1.6	0.2
5.0	3.4	1.5	0.2
4.5	2.3	1.3	0.3
5.4	3.4	1.7	0.2
5.1	2.5	3.0	1.1
5.5	2.6	4.4	1.2
4.8	3.4	1.9	0.2
...	...	...	...

Deuten diese Werte darauf hin, daß sich unter diesen Pflanzen mehrere Unterarten der Schwertlilie befinden? Nehmen Sie eine Clusterbildung mit Hilfe von *Matlab* vor. Beginnen Sie dabei mit der hierarchischen Clusterung. Treffen Sie anhand des *Dendrogramms* eine Entscheidung über die Anzahl der Cluster.

Führen Sie mit derselben Clusterzahl zum Vergleich die K-Mittelwertclusterung durch. Versuchen Sie zu klären, ob die beiden Clusterungsmethoden ungefähr auf dieselben Ergebnisse führten. Vergleichen Sie dazu zunächst die Größen der bei beiden Methoden entstandenen Cluster und anschließend auch deren Inhalte.

Führen Sie schließlich mit selber Clusterzahl die fuzzifizierte Mittelwertmethode durch, um festzustellen, ob irgendwelche "Mischformen" vorhanden sind. Suchen Sie dazu alle Datensätze, die zu mindestens je 1/3 mindestens zwei Clustern zuzuordnen sind oder die zu keinem Cluster mehr als 1/3 zuzuordnen sind.

Es empfiehlt sich, die Daten zu normalisieren. (Warum?)

[1] Zu finden in der Datei „wbsa0350.dat" am üblichen Ort.