Übung 6

Alexander Hinneburg

Apriory Algorithmus

Hauptprogramm

1. (Hash rows,Hash items,Int rowCount) = Lese Daten;
2. frequentSets(rows, items,Int rowCount, Int minSupp);

Daten Lesen {

Lege leeren Hash für rows und items an;

rowCount=0;

while(<STDIN>) {

Liste row= zerlege neue Zeile in Elemente

Füge row in rows als String ein und intialisiere Zähler mit 1
oder falls row schon vorhanden erhöhe Zähler um 1

Füge alle Elemente von row in items ein und intialisiere Zähler mit 1
oder falls Element schon vorhanden erhöhe Zähler um 1

erhöhe rowcount um 1

Gebe rows, items und rowcount zurück

}

Apriory Algorithmus

```
frequentSets ( hash rows,  hash items,  int rowCount,  int minSupp) {
Gehe alle Elemente in Items durch
     falls (Item.Zaehler >= minsup) => gebe es aus
     sonst lösche es aus items
candidates=generateCandidates(items)
loesche items
while(candidates != leer) {
   forall row in rows {
     forall candidate in candidates {
            falls (candidate Teilmenge von row) => erhöhe Zähler um 1
      }
   }
   forall candidate in candidates {
      falls (Zähler von candidate in candidates >= minSupp) => print candidate
      sonst lösche candidate aus candidates
   }
   candidates=generateCandidates(candidates);
}
```

Apriory Algorithmus

```
generateCandidates (frequentsets) {
    Hash new_candidates = leer;
    k=Laenge der Itemsets in frequentsets
    forall itemset_a  in (sortierte frequentsets) {
       Liste a= Elemente von itemset_a;
       forall itemset_b>itemset_a in (sortierte frequentsets) {
           Liste b= Elemente von itemset_b;
           AundB=joinRows(a, b);
           Falls AundB != leer {
             prune_test=true;
             for (i=1;i<=k-1) {
               if (AundB \ { a[i] } in frequentsets nicht vorhanden ) => prune_test=false; break;
             }
             if (prune_test==true) => Füges AundB zu new_candidates
           }
           Sonst break
       }
    }
return candidates;
```

Aufgabe 2

- Der Prune Test kann in der Iteration k=1 nicht wirken, da bei der Erzeugung der zwei-elementigen Kandidaten die Häufigkeit aller möglichen Teilmengen schon durch Join-Schritt gesichert wird, d.h. es werden alle durch Join generierten Kandidaten übernommen.

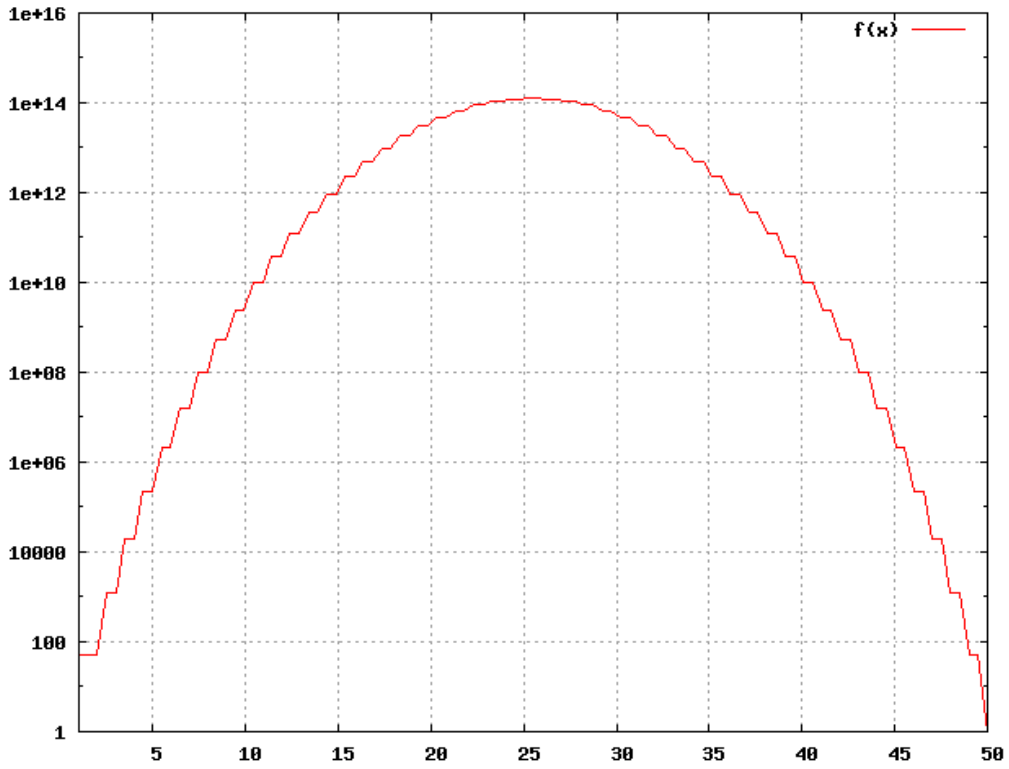
Aufgabe 3

- Falls eine häufige Item-Menge X der Größe 6 gefunden wird, werden 62 kleinere Item-Mengen vorher gefunden, die in X enthalten sind.

$$\sum_{i=1}^{6-1} \binom{6}{i} = 62$$

Aufgabe 4

- Gnuplot
  - f(x)=50!/((floor(x))!*(50-floor(x))!)
  - plot [1:50] f(x)



Aufgabe 5

- l_{max} maximale Länge einer Transaktion $l_{max} = \max_{t \in T}\{|t|\}$
- $n_{l_{max}}$ Anzahl der Transaktionen mit Länge l_{max}
- Falls $n_{l_{max}} \geq minsupport$ gilt, ist größte häufige Item-Menge l_{max}
- Ansonsten können $n_{l_{max}}$ Transaktionen der Länge l_{max} eine Item-Menge der Größe $l_{max} - 1$ um $n_{l_{max}}$ unterstützen
- Allgemein: größte häufige Item-Menge kann $l_{max} - i^*$ lang sein

$$i^* = \max_{0 \leq i \leq l_{max}} \{i : n_{l_{max}-i} + \sum_{j=0}^{i-1} n_{l_{max}-j} \geq minsupport\}$$

- Obere Schranke

$$s_k = \begin{cases} 0 & k > l_{max} - i^* \\ \binom{|L_1|}{k} & k \leq l_{max} - i^* \end{cases}$$

- Einen Durchlauf über die Transaktionen, Hash mit auftretenden Längen
- In den Schritten k=1, 2 werden die Schranken recht nah zusammenliegen, da die meisten Transaktionen bei Supermarktdaten klein sind.

Aufgabe 6

```
sed "1,1d" wetter_nominal.csv >wetter_nominal_ohneKopf.csv
perl.exe miner.pl 2 <wetter_nominal_ohneKopf.csv
gawk -F" " '{if (NF==2) print $0}'  <wetter_nominal_result.txt|wc
gawk -F" " '{if (NF==3) print $0}'  <wetter_nominal_result.txt|wc
gawk -F" " '{if (NF==4) print $0}'  <wetter_nominal_result.txt|wc
gawk -F" " '{if (NF==5) print $0}'  <wetter_nominal_result.txt|wc
```

- Minsupport=2
  - 1:12, 2:47, 3:39, 4:6
- Minsupport=3
  - 1:12, 2:26, 3:4, 4:0

Aufgabe 7

- Häufige Item-Mengen sind kombinatorische Teilmatrizen mit mindestens *minsupport* Zeilen, die nur Einsen enthalten
- Eine Verallgemeinerung ist, kombinatorische Teilmatrizen zu finden, mit mindestens *minsupport* Zeilen, die zu fast 100% Einsen enthalten.