Übung 5

Alexander Hinneburg

Aufgabe 5.1.1

```
# die Dateien ALL_AML_gcol.test.join.csv  ALL_AML_gcol.train.join.csv aus der
# letzten Uebung sind im akt. Verzeichnis, ebenso transpose.awk

rm all_t.txt aml_t.txt  gene_names.txt all_s2n.txt aml_s2n.txt

cat ALL_AML_gcol.train.join.csv | gawk -f s2n_t.awk
# all_t.txt all_t.txt all_s2n.txt aml_s2n.txt und gene_names.txt werden erzeugt

paste all_s2n.txt gene_names.txt |sort -r -n | head -n 50 > all_s2n_top50.txt
paste all_t.txt gene_names.txt |sort -r -n | head -n 50 > all_t_top50.txt
paste aml_s2n.txt gene_names.txt |sort -r -n | head -n 50 > aml_s2n_top50.txt
paste aml_t.txt gene_names.txt |sort -r -n | head -n 50 > aml_t_top50.txt
```

Aufgabe 5.1.1 (s2n_t.awk)

```
BEGIN {
FS=","
}
NR==1 {
    n = NF
    for (i = 2; i <= n; i++) {
        all_x[i] = 0
        all_x2[i] = 0
        all_n[i] = 0
        aml_x[i] = 0
        aml_x2[i] = 0
        aml_n[i] = 0
        head[i] = $i
    }
}
```

Aufgabe 5.1.1 (s2n_t.awk)

```
NR>1 {
    for (i = 2; i <= n-1; i++) {
        if ($n == "AML") {
            aml_x[i] = aml_x[i] + $i
            aml_x2[i] = aml_x2[i] + ($i)^2
            aml_n[i] = aml_n[i] + 1
        } else {
            all_x[i] = all_x[i] + $i
            all_x2[i] = all_x2[i] + ($i)^2
            all_n[i] = all_n[i] + 1
        }
    }
}
```

Aufgabe 5.1.1 (s2n_t.awk)

```
END {
   file1="gene_names.txt“;file2="all_s2n.txt“;file3="all_t.txt“;file4="aml_s2n.txt“;file5="aml_t.txt"
for (i = 2; i <= n-1; i++) {
      all_avg[i] = (all_x[i]/all_n[i])
      aml_avg[i] = (aml_x[i]/aml_n[i])
      all_sigma[i] = sqrt(1/all_n[i] * (all_x2[i] - (all_x[i]^2/all_n[i])) )
      aml_sigma[i] = sqrt(1/aml_n[i] * (aml_x2[i] - (aml_x[i]^2/aml_n[i])) )
      if (all_sigma[i] + aml_sigma[i] == 0) {
         all_s2n[i] = 0; aml_s2n[i] = 0; all_t[i] = 0; aml_t[i] = 0
      } else {
         all_s2n[i] = (all_avg[i] - aml_avg[i])/(all_sigma[i] + aml_sigma[i])
         aml_s2n[i] = (aml_avg[i] - all_avg[i])/(all_sigma[i] + aml_sigma[i])
         all_t[i] = (all_avg[i] - aml_avg[i])/sqrt((all_sigma[i]^2/all_n[i]) + (aml_sigma[i]^2/aml_n[i]))
         aml_t[i] = (aml_avg[i] - all_avg[i])/sqrt((all_sigma[i]^2/all_n[i]) + (aml_sigma[i]^2/aml_n[i]))
      }
      printf ("%s\n",head[i]) >> file1; printf ("%2.5f\n",all_s2n[i]) >> file2
      printf ("%2.5f\n",all_t[i]) >> file3; printf ("%2.5f\n",aml_s2n[i]) >> file4
      printf ("%2.5f\n",aml_t[i]) >> file5
   }
}
```

Aufgabe 5.1.1 (Ergebnis)

all_2sn
1.37076 U22376_cds2_s_at
1.15262 X59417_at
1.14397 X52142_at
1.14064 M28170_at
1.12754 U05259_rna1_at
1.06807 M92287_at
1.06554 L13278_at
1.06116 U09087_s_at
1.05152 X74262_at
…
0.84544 U62136_at
0.84234 X76061_at
0.83791 M11722_at

aml_s2n
1.51783 M55150_at
1.50752 X95735_at
1.47852 U50136_rna1_at
1.25940 M84526_at
1.25459 M16038_at
1.24816 M23197_at
1.24800 M81933_at
1.23530 U82759_at
1.23231 Y12670_at
…
0.84825 L38608_at
0.84727 M83667_rna1_s_at
0.84622 Y00339_s_at

Aufgabe 5.1.1 (Ergebnis)

all_t	aml_t
8.07155 U22376_cds2_s_at	8.42142 M55150_at
6.95879 X59417_at	6.73723 U50136_rna1_at
6.45704 M31211_s_at	6.51187 U82759_at
6.37643 M28170_at	6.43535 M81933_at
6.35239 M92287_at	6.38065 U12471_cds1_at
6.31705 U09087_s_at	6.00419 X95735_at
6.29592 U05259_rna1_at	5.84083 D49950_at
6.26510 D26156_s_at	5.78834 M11147_at
6.14368 L13278_at	5.63275 X17042_at
…	...
5.04665 M77142_at	4.04069 D43682_s_at
5.01579 U88666_at	4.02906 M22960_at
4.99469 AF005043_at	4.02499 D14874_at

Aufgabe 5.1.2

- Für zwei Klassen:

$$S2N(x_{all}, x_{aml}) = -S2N(x_{aml}, x_{all})$$

$$T(x_{all}, x_{aml}) = -T(x_{aml}, x_{all})$$

- Für mehre Klassen gilt dies nicht

Aufgabe 5.1.3

- Fehler in der Aufgabenstellung
  - Schnitt zwischen den Top50 von S2N und T einer Klasse bilden
    => eine Top-Liste für ALL und eine für AML
- Top ALL: 46 Gene, Top AML: 36 Gene
- Schnitte zwischen den Top 3:
  - ALL: U22376_cds2_s_at, X59417_at
  - AML: M55150_at, U50136_rna1_at

Aufgabe 5.1.3

```
cut -f 2 all_s2n_top50.txt |sort >all_s2n_genes_sorted.txt
cut -f 2 all_t_top50.txt |sort >all_t_genes_sorted.txt
join all_s2n_genes_sorted.txt all_t_genes_sorted.txt > all_top.txt

cut -f 2 aml_s2n_top50.txt |sort >aml_s2n_genes_sorted.txt
cut -f 2 aml_t_top50.txt |sort >aml_t_genes_sorted.txt
join aml_s2n_genes_sorted.txt aml_t_genes_sorted.txt > aml_top.txt
```

Aufgabe 5.1.4

- Konnte in Originalaufgabe nicht gelößt werden
- Alternativen?

Mehrschritt-Eigenschaft der Entropie

- siehe Tafel

Apriory Algorithmus

Hauptprogramm

1. (Hash rowsref,Hash itemsref,Int rowCount) = Lese Daten;
2. frequentSets(rowsref, itemsref,Int rowCount, Int minSupp);

Daten Lesen {

Lege leeren Hash an für rows und items;

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 vorhanden erhöhe Zähler um 1

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

erhöhe rowcount

Gebe rows items und rowcount zurück

}

Apriory Algorithmus

```
frequentSets ( rows,  items,  rowCount,  minSupp)
Gehe alle Elemente in Items durch
      falls >= minsup: gebe es aus
      sonst: lösche es aus items
Liste survivors=sortierte items
Lösche items
candidates=generateCandidates(survivors)
while(candidates != leer) {
survivors = leere Liste;
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
       }
   }
    survivors=sortierte candidates;
    Lösche candidates
   candidates=generateCandidates(survivors);
}
```

Apriory Algorithmus

```
generateCandidates (itemsets) {
    Hash candidates = leer;
    for(i=0; i<Anzahl Itemsets; i++) {
    Liste a= Elemente von itemsets[i]);
    for(j=i+1; j<=Anzahl itemsets; j++) {
        AundB=joinRows(a, b);
        Falls geklappt
            Füges es zu candidates
            sonst break
return candidates;
```