Einführung in die Bioinformatik

Prof. Dr. Enno Ohlebusch, Dr. Karlheinz Holzmann,
Tobias Badura

WS 15/16

Übungsblatt 6

Abgabe bis 10.02.2016, 12:00. Lösungen bitte elektronisch (an tobias.badura@uni-ulm.de) mit Namen in Text und Quelltextdateien abgeben.

1. Aufgabe (12): Preprocessing

Für diese Aufgaben wird der `Golub`-Leukämie-Datensatz (ALL versus AML) von der Vorlesungswebseite benötigt. Lesen Sie diesen mit `read.table("rgolub.train",check.names=FALSE)` in die Variable X ein. Die Zeilen enthalten Gene und die Spalten die verschiedenen Fälle (ALL, AML). Konvertieren Sie den eingelesenen data.frame in eine matrix mit `as.matrix()`.

(a) Begrenzen Sie die Expressionswerte elementweise auf das Intervall $[100, 16000]$ d.h. Werte < 100 werden auf 100 gesetzt und Werte > 16000 werden auf 16000 gesetzt.

(b) Filtern Sie alle Gene heraus, so daß für jedes übriggebliebene Gen gilt $max/min > 5$ und $max - min > 500$ (über alle Samples). Geben Sie die Anzahl der übriggebliebenen Gene aus.
Hinweis:
Schreiben Sie eine Funktion (oder eine Lambda-Funktion), die für eine gegebene Zeile einen booleschen Wert zurückgibt und wenden sie `apply` an.

(c) Normalisieren Sie den Datensatz mit der Lowess Regression.
Hinweis: Verwenden Sie hierzu `lowess` aus dem `stats` Paket.

(d) Erstellen Sie einen MA-Plot für die Ausgangsdaten und die normalisierten Daten.

(e) Logarithmieren Sie alle normierten Expressionswerte (Logarithmus zur Basis 2).

(f) Plotten Sie eine Häufigkeitsverteilung (Histogramm) der Genexpressionswerte aus Aufgabe (e).

2. Aufgabe (2): Multiples Testen

Laden Sie den Genexpressionsdatensatz `multTest.txt` von der Vorlesungswebseite herunter (Zeilen = Konditionen, Spalten = Gene, letzte Spalte = Klassenlabel).

(i) Geben Sie die statistisch signifikanten Gene zwischen kranker und gesunder Gruppe auf einem Signifikanzniveau von 5% an. Begründen Sie die Wahl der Teststatistik.

(ii) Korrigieren Sie für multiples Testen. Geben Sie die berechneten Gene unter Verwendung der Bonferroni-Korrektur sowie der Holm-Prozedur an.

3. Aufgabe (6): Partitionierende Clusteranalyse

Implementieren Sie den *k-means* Clusteralgorithmus in R. Verwenden Sie die Euklid'sche Distanz als Abstandsmaß. Als anfängliche Clusterzentren sollen zufällig Punkte aus der gegebene Datenmenge gezogen werden. Testen Sie den Algorithmus mit dem Golub-Datensatz von der Vorlesungswebseite (`golub50.test`). Führen unterschiedliche Initialisierungen der Clusterzentren zu unterschiedlichen Ergebnissen?

4. Aufgabe (0): Nächster-Nachbar-Klassifikator

Ein Klassifikator weist einem neuen unbekannten Muster eine Klasse aus vorher gelernten Daten (z.B. Gen-Expressions-Profile von Zellen bekannter Klasse) zu. Um einen Klassifikator auf seine **Generalisierungsleistung** (d.h. wie gut kommt der Klassifikator mit neuen, ungelernten Daten zurecht) zu überprüfen, werden die Daten oft in einen **Trainingsdatensatz** (TR) und einen davon unabhängigen **Testdatensatz** (TE) aufgeteilt. Mit TR wird der Klassifikator trainiert und dann die Klassifikationsleistung (d.h. die Anzahl der Fehler - falsch zugewiesene Labels) auf TE gemessen.
Einer der einfachsten und oft sehr gut funktionierenden Klassifikatoren ist der *one nearest neighbor* Klassifikator, der einem unbekannten Muster die Klasse des nächsten Datenpunktes in TR zuweist.
Implementieren Sie den *one nearest neighbor* (1-NN) Klassifikator in R. Verwenden Sie die Euklid'sche Norm als Abstandsmaß. Testen Sie das R-Skript mit dem Golub-Datensatz von der Vorlesungswebseite (Training mit `golub50.train` und Test mit `golub50.test`). Die letzte Spalte enthält die Klassenbezeichnung $(0, 1, 2)$. Geben Sie die Anzahl der falsch klassifizierten Datenpunkte auf der Trainings- und der Testmenge aus.
Hinweise:
Der 1-NN weist einem Abfrage-Datenpunkt $\mathbf{x} \in \mathbb{R}^n$ die Klasse des (bzgl. des spezifizierten Abstandsmaßes d) nächsten Nachbarn aus der Trainingsmenge $(\mathbf{x}^\mu, y^\mu) \in \mathbb{R}^n \times \{1, 2, \ldots, c\}$ zu ($\mu = 1 \ldots m$, c die Anzahl der Klassen), d.h. die Klasse y^{μ^*} mit

$$\mu^*(\mathbf{x}) = \arg \min_{\mu=1\ldots m} d(\mathbf{x}, \mathbf{x}^\mu) \quad .$$

Für den Fall des Euklid'schen Abstandes gilt $d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} (x_i - y_i)^2$ mit $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Teilen Sie für diese Aufgabe den Klassifikator am besten auf in

1. eine **Trainingsfunktion** `classifier <- train.1nn(X,y)`, die eine Zeilenmatrix `X`$\in \mathbb{R}^{m \times n}$ (je ein Trainings-Datenpunkt pro Zeile) und einen Labelvektor `y`$\in \mathbb{R}^m$ akzeptiert und einen Klassifikator zurückgibt (beim 1-NN besteht das Training nur darin sich die Datenpunkte zu merken - z.B. in einer benannten Liste `list(train=X,labels=y)`), und in

2. eine **Testfunktion** `y <- predict.1nn(classifier,x)`, die einem n-dimensionalen Datenpunkt (Vektor) `x` mittels des trainierten Klassifikators `classifier` eine Klasse y zuweist.