Werkzeuge der Mustererkennung und des Maschinellen Lernens Übung 7

June 11, 2017

Bemerkung

Da der knn-Algorithmus die Punkte von denen die Nachbarsberechnung ausgeht, zufällig auswählt, variieren die Ausgaben in der Tabellenmatrix m etwas. Würde die *cast()*-Funktion einige Nachkommastellen mehr stehen lassen, würde auch eine größere Änderung bei mehrmaliger Ausführung der Ergebnisse vorliegen.

d)

Bis zu $k = 7$ nimmt die Fehlerrate ab. Danach steigt sie in beiden Beobachtungen wieder an.
Die Fehlerrate ist in der zweiten Zeile deutlich größer. Das liegt haupstsächlich daran, dass bei Vertauschung von Lern- und Testdaten die Größenzuordnungnen nicht mehr vorteilhaft sind.
Die Fehlerquote der zweiten Spalte ergeben sich dadurch, dass mit 100 Merkmalsvektoren gelernt wurde, während mit 668 Vektoren getestet wurde. D.h., dass deutlich weniger Merkmalsvektoren zur Verfügung standen um eine bessere Separierung zu ermöglichen.
Üblich ist es mit ca. 80% aller vorhandenen Datensätze zu lernen und die restlichen 20% als Testdatensatz zu verwenden (vgl. Pareto-Verteilung). In manchen Fällen lohnt sich auch die Aufteilung in 90% Trainingsdaten und 10% Testdaten. Hier jedoch lag der umgedrehte Fall vor, dass mit ca. 13% gelernt und mit 87% getestet wurde.

e)

Die Raten liegen zwischen denen, die in Zeile 1 und 2 berechnet wurden.
Grundsätzlich ist zu erwarten, dass das Ergebnis der *Leave-One-Out*-Methode etwas größere Fehler erzeugt als mit der Berechungsmethode der ersten Spalte. Dies liegt in der ungünstigen Aufteilung von Test- und Trainingsmenge begründet unter welcher die *knn()*-Funktion hier aufgerufen wird.
Auch hier verändert sich die Fehlerrate bei steigenden k ähnlich wie in den anderen beiden Fällen.

f)

Gemäß der obigen Bemerkung verändern sich die Werte nur gering.
Auffällig bei unseren Betrachtungen ist, dass sich die Fehler bei erneuter Berechnung tendentiell verschlechtern, als sich zu verbessern. Gemäß der Bemerkung ist das ein Zufall. Die größten Abweichungen liegen auch nach mehrmaligen probieren um den Bereich $k = 3$.
Höherwertige ks haben tendentiell eine geringere Abweichung als die niedrigwertigereren.

i)

knn.heldout und *knn.leave1out* liefern sehr ähnliche Resultate. Daher wird im Folgenden sich hauptsächlich auf die Auswertung nach *knn.leave1out* bezogen.
Von allen Merkmalen i scheinen die ersten beiden $i = 1$ und $i = 2$ besonders hilfreich zu sein. Werden diese weggelassen, so ist der Fehler ungefähr 4% über denen der anderen Fälle.
Hingegen die Merkmale $i = 4$ und $i = 5$: Diese Merkmale scheinen weniger gut geeignet zu sein,

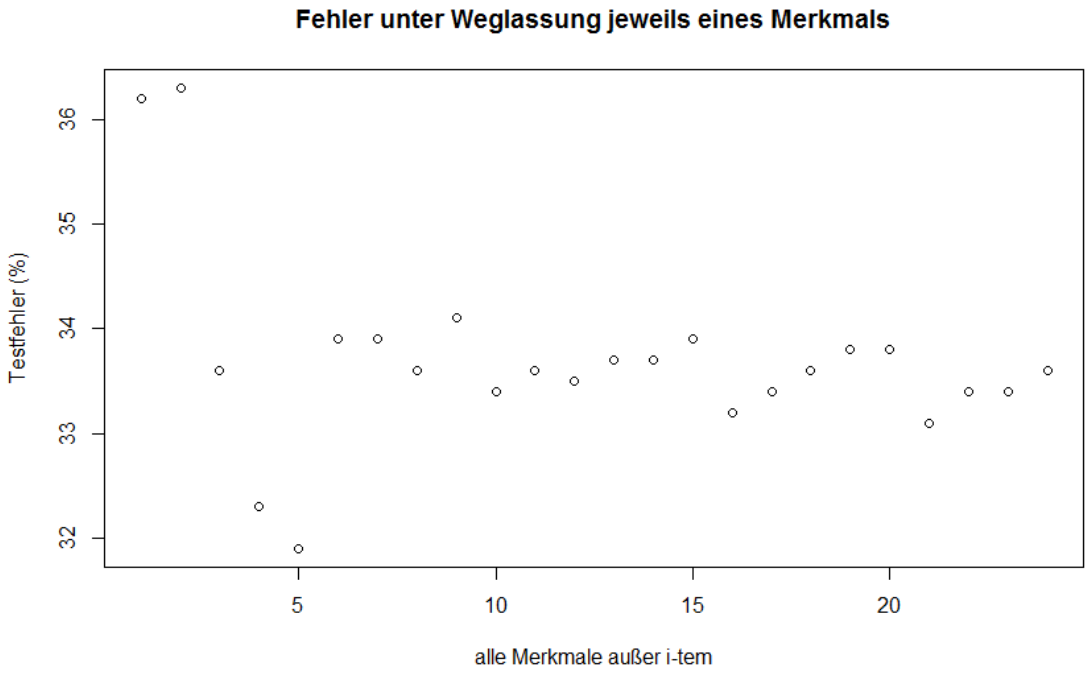

Figure 1: Testfehler für alle möglichen 23 dimensionalen Merkmalsräume unter Verwendung der Funktion *knn.leave1out*.

die Klassen von einander zu separieren. Sind sie entfernt, wird der Fehler geringer.
Der Fehlermedian liegt hier bei 33.6%. Der größte Fehler $k = 2$ weicht davon um 2.7% ab, der kleinste $k = 5$ um 1.7%.
Unter *knn.heldout* sind die Fehler alle um ca. 9% schlechter.
Werden die besten 22 Merkmale zu einem Raum zusammengefasst, so kann der knn-Algorithmus für $k = 1$ ein um ca. 1.5% geringeren Fehler gegenüber dem größtmöglichen Merkmalsraum liefern. Das ist kaum signifikant.
(siehe Code)