Friedrich-Schiller-Universität Jena

Fakultät für Mathematik und Informatik
Institut für Informatik
Lehrstuhl für Digitale Bildverarbeitung

M.Sc. Clemens-Alexander Brust
M.Sc. Dimitri Korsch

Prof. Dr.-Ing. Joachim Denzler
http://www.inf-cv.uni-jena.de

Übung zur Vorlesung

Rechnersehen 1

WS 2018/2019

Übungsblatt 6: Hauptachsentransformation & Klassifikation

Ausgabe: 23.1.2019 Abgabe: 6.2.2019

Aufgabe 1 Laden des Datensatzes (1 Punkte)

Im Moodle finden Sie den Datensatz `toy-dataset`, welcher vier Ordner mit 16×16 Pixel großen Beispielbildern der Klassen *apple*, *car*, *cow* und *dog* enthält. Im Folgenden soll ein Klassifikator angelernt werden, welcher diese vier Klassen erfolgreich unterscheiden kann. Laden Sie dazu zunächst den Datensatz. Der nachfolgend zu realisierende Klassifikator soll jeweils 9 von 10 Bildern je Klasse während der Lernphase erhalten. Das jeweils verbleibende Bild soll als *unbekanntes* Testbild verwendet werden. Erstellen Sie eine Routine, welches für einen übergebenen Datensatz eine zufällige Einteilung in Trainings- und Testdaten realisiert. Achten Sie insbesondere darauf, das Verhältnis zwischen Trainings- und Testdaten variabel zu gestalten.

Aufgabe 2 Merkmalsextraktion (4 Punkte)

Bevor ein Klassifikator trainiert werden kann, müssen Bilder geeignet durch Merkmale repräsentiert werden.

a) Eine sehr einfache Möglichkeit, Merkmale aus den Bildern zu gewinnen besteht darin, die Pixelwerte der Bilder direkt als Merkmal zu wenden. In Fall einfacher Grauwertbilder wird dazu jedes Bild vektorisiert, indem die Pixelwerte zeilen- oder spaltenweise in einem Spalten-Vektor abgelegt (z.B. mit der `reshape`-Funktion aus dem Paket `numpy`.). Für die Bilder aus obigem Datensatz ergibt dies eine Merkmalsdimension $d = 256 \times 1$.

b) Häufig muss die Dimension der Merkmale reduziert werden, um eine schnellere Klassifikation mit geringerem Speicherbedarf zu ermöglichen. Dazu kann zum Beispiel eine Hauptachsentransformation durchgeführt werden (*principal component analysis*, PCA). In `scikit-learn` steht dafür die Funktion `sklearn.decomposition.PCA` zur Verfügung, die hier jedoch **nicht** verwendet werden soll. Stattdessen kann eine entsprechende Transformation auch wie folgt umgesetzt werden:

1. Erstellen Sie eine Datenmatrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ in der spaltenweise die n Merkmalsvektoren gelagert sind.
2. Machen sie jede Merkmalsdimension **mittelwertfrei**, d.h., subtrahieren Sie je Dimension den entsprechenden Mittelwert aller Trainingsdaten.
3. Berechnen Sie anschließend die Kovarianzmatrix $\mathbf{\Sigma} = \mathbf{X}\mathbf{X}^T$, mit $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$
4. Berechnen Sie schließlich die Eigenwerte und -vektoren von $\mathbf{\Sigma}$ mittels geeigneter `numpy`-Funktionen (z.B. `np.linalg.eig`, `np.linalg.svd`, ...)

Für eine Dimensionsreduktion auf die $k,\ 1 \leq k \leq d,$ *wichtigsten* Dimensionen werden die Eigenvektoren zu den k größten Eigenwerten verwendet. Wir fassen diese Eigenvektoren in einer Matrix $\mathbf{P} = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ zusammen. Auf diese k Eigenvektoren können nun alle ursprünglichen Merkmalsvektoren $\mathbf{x_i}$ mittels $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{P}^T\mathbf{x_i}$ projiziert werden.

Implementieren Sie beide Verfahren zur Beschreibung von Bildern. Achten Sie insbesondere darauf, die Pixelwerte für die Merkmalsvektoren auf den Wertebereich $[0, 1]$ zu **skalieren**! Vergleichen Sie im Zweifelsfall Ihre Ergebnisse mit denen der entsprechenden Python-Routinen.

Aufgabe 3 Nächster-Nachbar-Klassifikator (3 Punkte)

Der vermutlich einfachste Klassifikator ist der Nächste-Nachbar-Klassifikator. Dieses Verfahren vergleicht den zu klassifizierenden Merkmalsvektor $\mathbf{x}_*$ mit allen Merkmalsvektoren $\mathbf{x}_i$ der Trainingsdaten. Die Klasse desjenigen Merkmalsvektors der Trainingsdaten, welches den geringsten Abstand zu $\mathbf{x}_*$ aufweist, wird als Ergebnis der Klassifikation zurückgeliefert.

Implementieren Sie diesen sogenannten Nächsten-Nachbar-Klassifikator. Achten Sie dabei darauf, dass er mit beliebigen Stichproben trainiert werden kann.

Gestalten Sie die Distanzberechnung variabel. Welche Abstandsmaße sind Ihrer Meinung nach möglich bzw. sinnvoll?

Aufgabe 4 Leave-one-out Estimation (2 Punkte)

Stellen Sie die vorherigen Teilaufgaben zu einem Gesamtsystem zusammen. Dabei soll aus den Eingabedaten aus jeder Klasse ein Bild ausgewählt werden, welches *nicht* für das „Lernen“ der Hauptachsentransformation und das Trainieren des Nächsten-Nachbar-Klassifikators verwendet wird (s. erste Aufgabe). Anschließend werden diese Bilder als Testdaten verwendet, transformiert und schließlich klassifiziert. Ermitteln Sie die Erkennungsleistung des Klassifikators. Variieren Sie nun das Verhältnis zwischen Anzahl der Trainings- und Testbeispiele. Was ist zu beobachten?

Zusatz: Wiederholen Sie Ihr Experiment für mehrere zufällige Aufteilungen des Datensatzes in Trainings- und Testmenge. Was ist zu beobachten?

Viel Spaß und Erfolg!