Machine Learning 1
Refugeeks / intoCODE
2022

Torben Laue, M.Sc.
Florian Kluger, M.Sc.

K-Nearest-Neighbors

1 Aufgaben

Die erste Aufgabe können Sie im Kopf bzw. mit Stift und Papier bearbeiten. Für die darauf folgenden verwenden Sie bitte die von uns bereitgestellten Python-Templates.

1.1 Einfaches 2D-Beispiel

In Abb. 1 ist ein Datensatz dargestellt, welcher aus drei unterschiedlichen Klassen (rot, grün und blau) besteht, welche jeweils acht Punkte enthalten. Weiterhin sind fünf Punkte – dargestellt durch schwarze Dreiecke – vorgegeben, deren Klassenzugehörigkeiten unbekannt sind und mittels K-Nearest-Neighbour bestimmt werden sollen. Klassifizieren Sie die Punkte mit $K \in \{1, 2, 3, 5\}$ und bestimmen Sie jeweils die Genauigkeit der Klassifikation.

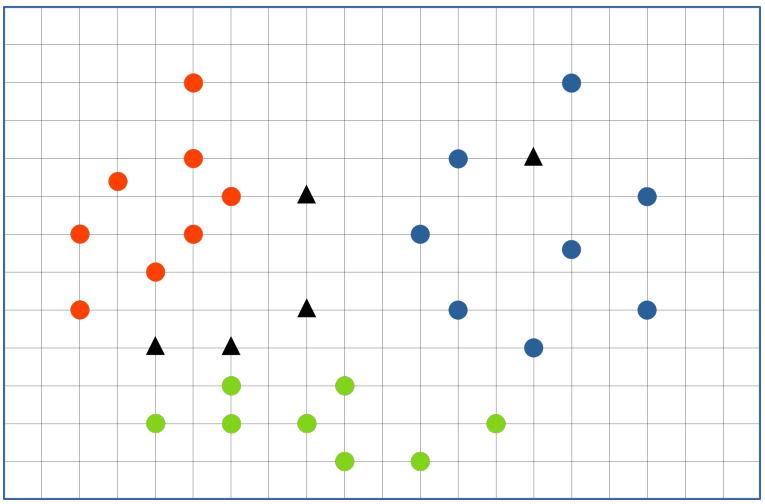
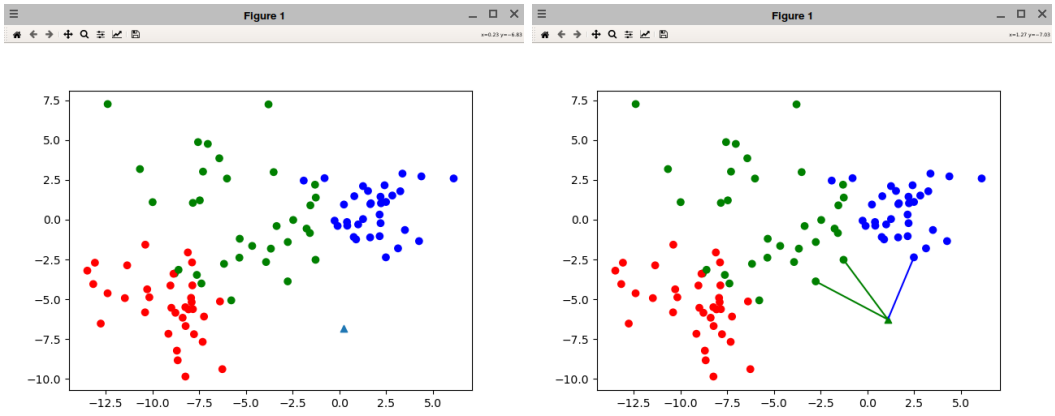


Abbildung 1: Punkte mit bekannten Klassen (rot, grün, blau) und Dreiecke (schwarz), welche klassifiziert werden sollen.

1.2 2D-Blobs

Verwenden Sie für diese Aufgabe die Datei `blobs_knn.py`. Beim ersten Ausführen (ohne Änderungen) sollte ein Fenster ähnlich Abb. 2a erscheinen. Wenn Sie mit der rechten Maustaste auf den Plot klicken, sollte an der entsprechenden Stelle ein schwarzes Dreieck erscheinen.



(a) (b)

Abbildung 2

Die dargestellten Punkte und ihre zugehörigen Ground-Truth-Klassen (Labels) liegen in den Arrays `data_train` und `labels_train` vor. Die vorgegebene Funktion `onclick` wird aufgerufen, sobald Sie innerhalb des Plots mit den Punkten einen Mausklick tätigen.

(a) Definieren Sie mittels der scikit-learn-Klasse `KNeighborsClassifier` einen KNN-Klassifikator mit $K = 3$ und lernen Sie ihn auf die gegebenen Daten an.

(b) Passen Sie die Funktion `onclick` so an, dass der mit dem Mauszeiger markierte Punkt mit den Koordinaten `sample` durch den KNN-Klassifikator mittels der `predict`-Funktion klassifiziert wird. Hinterlegen Sie die prädizierte Klasse in der Variable `pred`, damit der Punkt in der klassifizierten Farbe dargestellt wird.

(c) Schauen Sie nun, wie Punkte innerhalb der drei Cluster, in den Grenzbereichen und außerhalb der Cluster klassifiziert werden. Was passiert, wenn Sie K vergrößern oder verkleinern? Welches Problem kann auftreten, wenn Sie für K eine gerade Zahl verwenden?

1.3 MNIST: Handschriftliche Ziffern

Als nächstes arbeiten wir mit dem MNIST-Datensatz, welcher 70000 handschriftliche Ziffern $(0-9)$ in Form von 28×28-Pixeln großen Grauwert-Bildern enthält. Je ein Beispiel für jede der zehn Klassen ist in Abb. 3 dargestellt.

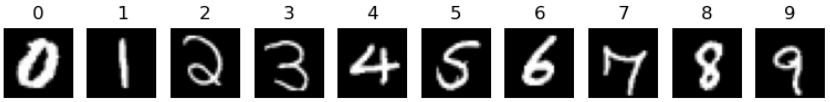


Abbildung 3

In der Datei `mnist_knn.py` finden Sie die Trainingsdaten in den Variablen `images_train` und `labels_train`, sowie die Testdaten in `images_test` und `labels_test`.

(a) Definieren Sie einen KNN-Klassifikator mit $K = 3$ und lernen Sie ihn auf die MNIST-Trainingsdaten an.

(b) Klassifizieren Sie nun die Testdaten und bestimmen Sie die Genauigkeit (Accuracy) des Klassifikators. Hinterlegen Sie die prädizierten Klassen in der Variable `pred_test`, damit einige Beispiele angezeigt werden können. Wie verändert sich die Genauigkeit, wenn Sie K verändern?

(c) Verwenden Sie die scikit-learn-Funktion `confusion_matrix`, um eine Konfusionsmatrix bezüglich der prädizierten und der Ground-Truth-Klassen zu bestimmen. Hinterlegen Sie diese Matrix in der Variable `conf`, damit diese anschließend visuell dargestellt wird. Welche Klassen werden besonders oft miteinander verwechselt? Woran liegt das?