Einführung in das Maschinelle Lernen **Universität Hamburg**
WiSe 2023/24 Sören Laue

Übungsblatt 5

Aufgabe 1 – 10%

Nehmen Sie an, Sie haben einen Trainingsdatensatz mit $n_{\text{train}} = 1000$ Punkte. Von diesen haben 750 Punkte das Label 0 und alle restlichen das Label 1.
Sie haben einen Testdatensatz mit $n_{\text{test}} = 500$ Punkten. Von diesen haben 250 Punkte Label 0 und alle restlichen Punkte Label 1. Bei dem Trainingsdatensatz ist es nicht der Fall, dass die Koordinaten zweier Punkte identisch sind.
Sie führen den k-Nearest-Neighbor-Classifier auf diesen Daten folgendermaßen aus:

1. Sie setzen $k = 1$. Wie hoch ist die Genaugikeit auf dem **Trainingsdatensatz**? Begründen Sie.

2. Sie setzen $k = 501$. Wie hoch ist die Genaugikeit auf dem **Trainingsdatensatz**? Wie hoch ist die Genaugikeit auf dem **Testdatensatz**? Begründen Sie.

3. Könnten die Ergebnisse anders ausfallen, falls doch zwei Punkte im Trainingsdatensatz die gleichen Koordinaten haben? Begründen Sie.

Aufgabe 2 – 10%

Nehmen Sie an, dass Sie einen Datensatz mit n Datenpunkten haben, die jeweils d Features haben. Wie hoch ist die Laufzeit des kNN-Algorithmus für die Klassifizierung eines Punktes?
Welche Schwierigkeiten sehen Sie hinsichtlich der Skalierbarkeit der Methode? Beschreiben Sie einen möglichen Lösungsansatz.

Aufgabe 3 – 30%

Folgende Trainingsdatenpunkte $x_1 = (-10, 1)$, $x_2 = (10, 5)$, $x_3 = (-20, 6)$, $x_4 = (20, 0)$ mit den dazugehörigen Labels $y_1 = 0$, $y_2 = 1$, $y_3 = 1$, $y_4 = 0$ sind gegeben. Außerdem ist der Testdatenpunkt $p = (21, 4)$ mit Label $y_p = 1$ gegeben.
Führen Sie 1-Nearest-Neighbor-Classification für den Testpunkt durch. Nutzen Sie im Folgenden die euklidische Distanz.

In der Vorlesung haben Sie bereits ein Feature Scaling Verfahren kennengelernt.
Eine andere Möglichkeit ist die Min-Max-Skalierung. Dabei werden die Daten auf das Intervall $[0, 1]$ abgebildet. Die Formel für die Min-Max-Skalierung lautet:

$$x_{\text{skaliert}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Wenden Sie die **Min-Max-Skalierung** sowie **die aus der Vorlesung bekannte Normalisierung** an und führen Sie dann erneut 1-Nearest-Neighbor-Classification für den Testpunkt durch. Die Distanzen nach der Anwendung der Normalisierung aus der Vorlesung sind gegeben: $d(x_1, p) = 1.99$, $d(x_2, p) = 0.15$, $d(x_3, p) = 1.99$, $d(x_4, p) = 0.196$.

Zuletzt fügen Sie noch den Punkt $x_5 = (0, 30)$ mit Label $y_5 = 0$ den Trainingsdaten hinzu. Wie

verändern sich die Vorhersagen für den Testpunkt? Die Distanzen nach der Anwendung der Normalisierung aus der Vorlesung sind gegeben: $d(x_1, p) = 1.82$, $d(x_2, p) = 0.12$, $d(x_3, p) = 1.97$, $d(x_4, p) = 0.19$, $d(x_5, p) = 1.55$.
Was fällt Ihnen auf? Was ist die Ursache dafür?

Aufgabe 4 – 50%

In der Vorlesung wurde das Problem der Klassifikation thematisiert und den k-Nearest-Neighbor-Classifier (kNN) als einen grundlegenden Algorithmus kennengelernt, der auf dieses Problem angewendet werden kann. In dieser Programmier-Übung sollen sie sich mit Klassifikation und dem kNN-Classifier vertraut machen. Für diese Übung sind die Datensätze `iris_2D.csv`, `iris_original.csv`, `iris_randfeatures.csv` und `iris_scaledfeatures.csv` vorgesehen. Außerdem steht Ihnen ein Jupyter Notebook `kNN_classification.ipynb` zur Verfügung, welches ein grobes Gerüst für die Aufgabe bereitstellt. Sie sind nicht verpflichtet das Notebook zu nutzen, es dient nur zur Orientierung und Hilfestellung.

Arbeiten Sie zunächst mit dem Datensatz `iris_2D.csv`. Finden Sie über 10-fache Kreuzvalidierung die beste Anzahl von Nachbarn n, mit dem der kNN-Classifier die beste Performance erreicht. Wählen sie n im Intervall $[1, 15]$. Verwenden Sie als Evaluationsmetrik *Accuracy*. Stellen Sie in einem 2D-Plot dar, welche Bereiche der durch die zwei Features aufgespannten Fläche vom kNN-Classifier den jeweiligen Klassen zugeordnet ist. Visualisieren Sie außerdem die Vorhersagen für den von Ihnen erstellten Test-Datensatz. Sowohl eine Implementation eines kNN-Classifier als auch Möglichkeiten zur Visualisierung (`DecisionBoundaryDisplay.from_estimator`) finden sich in `scikit-learn`. Als Herausforderung können Sie ebenfalls eine eigene Implementation von Modell und Visualisierung mit `numpy` und `matplotlib` anfertigen.
Überlegen Sie, wie die verschiedenen Bereiche, die der kNN-Classifier den jeweiligen Klassen zuordnet, zu erklären sind. Machen Sie sich dabei bewusst, welche Informationen der kNN-Classifier für seine Vorhersage verwendet.

Im nächsten Schritt arbeiten Sie mit den Datensätzen `iris_original.csv`, `iris_randfeatures.csv` und `iris_scaledfeatures.csv`. Gehen Sie wie in der vorhergien Aufgabenstellung vor und finden Sie für jeden der Datensätze den kNN-Classifier, der auf dem jeweiligen Test-Set die beste Performance (*Accuracy*) erreicht. Eine Visualisierung Ihrer Ergebnisse ist nicht notwendig. Welche Datensätze sind für die kNN-Methode am schwersten zu modellieren?
Führen sie jetzt dieselbe Analyse noch einmal durch, aber verwenden Sie vor der Analyse und dem Training ihrer Klassifikations-Algorithmen eine Skalierungs-Methode mit der Sie ihre Daten präprozessieren können. Diskutieren Sie kurz ihr Ergebnis. Hat die Skalierung für alle Datensätze denselben Effekt auf die Modell-Performance?

Bitte lösen Sie die Fragen und die Programmieraufgaben bis zum **27. November 2023**. Ihren Python-Code, sowie Visualisierungen können Sie in Form eines Jupyter Notebooks oder PDFs hochladen.