Data Mining und Maschinelles Lernen

Prof. Kristian Kersting
Steven Lang
Felix Friedrich

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 2022
Übungsblatt 2

Benötigte Dateien
Alle benötigten Datensätze und Skriptvorlagen finden Sie in unserem Moodle-Kurs:
https://moodle.informatik.tu-darmstadt.de/course/view.php?id=1058

2.1 Kreuzvalidierung

Bei der Evaluierung eines Modells im maschinellen Lernen sind wir daran interessiert, wie gut das Modell auf ungesehenen Daten funktioniert, wenn wir es zur Vorhersage neuer Stichproben verwenden wollen. Daher ist es notwendig, die uns bereits vorliegenden Daten in einen Trainingssatz und einen Testsatz aufzuteilen. Das Modell wird auf dem Trainingssatz trainiert und auf dem Testsatz ausgewertet. Dies kann auf verschiedene Weise erfolgen, wie im Folgenden erläutert wird.

Das Problem bei der einfachen Train-Test-Aufteilung besteht darin, dass sie nur einen Bruchteil von $1 - p$ des vollständigen Datensatzes zur Bewertung der Modellleistung verwendet. Dies könnte problematisch werden, wenn der Datensatz selbst bereits recht klein ist. Ein kleinerer Testsatz erhöht die Bewertungsvarianz und kann leichter durch Ausreißer beeinflusst werden. Die Lösung für dieses Problem ist die so genannte k-fache Kreuzvalidierung.

a) **Train-Test-Aufteilung**

Implementieren Sie die Methode `train_test_split` und teilen Sie den Datensatz zu 70 % in eine Trainingsmenge und zu 30 % in eine Testmenge auf.

Sie können dabei die Methoden `numpy.random.seed`, `numpy.random.shuffle`, oder `numpy.random.choice` verwenden.

Sofern Sie diese Aufgabe nicht lösen können, verwenden Sie `train_test_split` von *sklearn*.

b) **Beschreibung der k-fachen Kreuzvalidierung**

Beschreiben Sie die Methodik der k-fachen Kreuzvalidierung (engl. k-fold Cross-Validation) und warum diese eine verlässlichere Validierung als ein einfacher Train-Test-Split ermöglicht. Ein Skizze ist hierbei hilfreich.

c) **Weitere Methoden zur Kreuzvalidierung**

Neben der üblichen k-fachen Kreuzvalidierung gibt es auch die folgenden Verfahren: Wiederholte k-fache Kreuzvalidierung (engl. Repeated k-Fold), Eins auslassen (engl. Leave One Out), P% auslassen (engl. Leave P Out) und Gemischte Trennung (engl. Shuffle Split). Beschreiben Sie kurz jeder dieser Techniken.

d) **Durchführung der Kreuzvalidierung**

Implementieren Sie die Funktion `acc_kfold_cross_val`, um die gegebene Kreuzvalidierungsmethode durchzuführen und die mittlere Trainings- und Testgenauigkeit zu errechnen. Verwenden Sie dabei die Methode `sklearn.model_selection.cross_validate`.

e) **Kreuzvalidierung bei großen Datenmengen**

Beurteilen Sie die Verwendung von Kreuzvalidierung bei Datenmengen mit 100, 1000 und einer Millionen Datenpunkten.

2.2 K nächste Nachbarn (kNN)

Die nachbarschaftsbasierte Klassifikation ist eine Art des instanzbasierten oder nicht-generalisierenden Lernens: Es wird nicht versucht, ein allgemeines internes Modell zu konstruieren, sondern es werden lediglich Instanzen der Trainingsdaten gespeichert. Die Klassifikation wird aus einer einfachen Mehrheitsentscheidung der nächsten Nachbarn jedes Punktes berechnet: Einem Abfragepunkt wird die Datenklasse zugewiesen, die innerhalb der nächsten Nachbarn des Punktes die meisten Vertreter hat.

a) **Vorverarbeitung des Datensatzes**

In dieser Aufgabe verwenden wir einen Datensatz über Schwertlilien, bekannt als Iris-Dataset, welcher von dem Biologen Ronald Fisher in dem Jahre 1936 veröffentlicht wurde [1]. Wählen Sie die ersten beiden Merkmale (engl. features): Kelchblattbreite (engl. sepal width) und Kelchblattlänge (engl. sepal length) über die Methode `load_reduced_iris_dataset` aus. Den vollständigen Iris-Datensatz können Sie über die Funktion `sklearn.datasets.load_iris` laden.

b) **Evaluierung unterschiedlicher k-Werte**

Der einzige und Hauptparameter von kNN ist die Anzahl, wie viele nächste Nachbarn eines Datenpunktes entscheiden, welchem Label der Datenpunkt zugeordnet wird. Evaluieren Sie die Trainings- und Testgenauigkeit für alle k-Werte $k \in [1, 3, 5, \ldots, 99]$ mittels der Funktion `evaluate_ks`. Verwenden Sie dabei die Methode `utils.evaluate` um eine Kreuzvalidierung mit 10 Splits für jeden K-Wert durchzuführen.

c) **Visualisierung der k-Werte**

Visualisieren Sie die k-Werte gegenüber der Traings- und Testgenauigkeit in der Funktion `plot_k_to_acc`. Der k-Wert wird auf die X-Achse und die Genauigkeit auf die Y-Achse abgebildet. Beschriften Sie ebenfalls beide Achsen.

d) **Bester k-Wert**

Bestimmen Sie anhand der zuvor erhaltenen Zwischenergebnisse den besten k-Wert über die Funktion `get_best_k`.

e) **Entscheidungsregionen**

Wir können auch die Entscheidungsregionen visualisieren, die ein kNN-Klassifikator für gegebene Werte von k erstellt. Erzeugen Sie einen kNN-Klassifikator und stellen Sie dessen Entscheidungsregionen visuell in der Methode `plot_decision_boundary_for_k` dar.

2.3 Regression (Fortsetzung)

Die lineare Regression ist die Aufgabe, eine Linearkombination der Eingabe X zu finden, um das Ziel y zu schätzen. Wir nehmen an, dass die Daten $\mathbf{x_i}$ linear mit dem Ziel y_i korreliert sind, unter Verwendung der Koeffizienten $\beta_0, \ldots, \beta_K \in \mathbb{R}$:

$$y_i = \beta_0 + \beta_1 x_i^1 + \beta_2 x_i^2 + \cdots + \beta_K x_i^K + \varepsilon_i \,, \tag{1}$$

wobei ε_i einen Störungsterm modelliert und der Superskript den Eingangsvariablen-Index entspricht.

Die lineare Regression lässt sich am besten an einem einfachen zweidimensionalen Beispiel demonstrieren. Hier reduziert sich diese auf den klassischen Fall $y_i = b + m x_i$.

a) **Robustheit**

Bewerten Sie, ob sich die lineare Regression robust gegenüber Ausreißer verhält und warum.

b) **Polynomiale Regression**

Wir können die lineare Regression leicht auf nicht-lineare Eingangsvariablen-Zielvariablen-Korrelationen ausdehnen. Daher ist es nur notwendig, unsere Eingabe in einen Merkmalsraum abzubilden, in dem die Merkmale unserer Meinung nach linear mit der Zielvariablen korreliert sind. Ein gutes Beispiel ist die Polynomiale Regression, die den Eingabedatenpunkt x nimmt und ihn auf den Satz polynomialer Merkmale $\left(x^0, x^1, x^2, \ldots, x^d\right)$ abbildet, wobei d der Grad der Polynomtransformation ist.

Implementieren Sie die Funktion `map_polynomial`, welche diese Operation durchführt.

Literatur

[1] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.