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

Übungsblatt 6

Aufgabe 1 – 25%

Nehmen Sie an, Sie müssen einen Naive Bayes Klassifikator konstruieren, um zwischen positiven und negativen Filmreviews zu unterscheiden. Die folgenden Reviews sind als Trainingsdaten gegeben:

Positiv	**Negativ**
a, great, time	one, out, of, five
a, great, film	a, terrible, movie
a, great, movie	boring, film
amazing, movie	not, great
not, bad	not, great, time

Nehmen Sie Laplacian Smoothing mit dem Wert 1 an. Führen Sie nun die Klassifizierung für die folgenden Reviews durch:

1. not, great, movie
2. a, boring, film

Was fällt Ihnen auf? Nehmen wir an, wir wollen die Klassifizierung mittels Bigrammen (zwei benachbarte Wörter werden als Einheit angesehen) verbessern.
Die Trainingsdaten werden nun als Bigramme dargestellt:

Positiv	**Negativ**
# a, a great, great time, time #	# one, one out, out of, of five, five #
# a, a great, great film, film #	# a, a terrible, terrible movie, movie #
# a, a great, great movie, movie #	# boring, boring film, film #
# amazing, amazing movie, movie #	# not, not great, great #
# not, not bad, bad #	# not, not great, great time, time #

Transfomieren sie die Testdaten auch zu Bigrammen und führen Sie die Klassifizierung erneut durch. Was fällt Ihnen auf? Haben wir die Probleme gelöst? Welche neue Probleme treten auf? Was wäre der Fall, wenn wir statt Bigrammen beliebig-große n-gramme verwenden?

Aufgabe 2 – 25%

In der Vorlesung haben Sie die Evaluationsmaße Recall, Precision und F1-Measure für Klassifikatoren kennengelernt. Nehmen Sie an, sie sind nicht mit einem binären Klassifikationsproblem konfrontiert, sondern mit einem Klassifikationsproblem mit k Klassen. In der Vorlesung wurde hier zwischen der *Micro* und *Macro* Ebene unterschieden.
Berechnen Sie die Werte für *Micro*/*Macro* Recall, Precision und F1-Measure für die folgenden Klassifikationen:

True \ Predicted	Dog	Cat	Horse	Wolf
Dog	25	2	2	1
Cat	1	45	0	4
Horse	1	1	10	3
Wolf	4	0	0	1

Was fällt Ihnen auf? Wann könnte es sinnvoll sein, die *Micro*-Variante bzw. die *Macro*-Variante zu verwenden?

Aufgabe 3 – 50%

In dieser Programmier-Übung sollen sie sich mit Klassifikation von Texten mit Hilfe des Naive Bayes Classifier vertraut machen. Für diese Übung sind die Datensätze `20newsgroups.csv` und `spam_or_not_spam.csv` vorgesehen. Außerdem steht Ihnen ein Jupyter Notebook `naivebayes_classification` 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. Es wird empfohlen zur Lösung dieser Aufgabe `scikit-learn` zu verwenden.

Part 1

Das Datenset `20newsgroups.csv` enthält ca. 18000 Texte, die in 20 verschiedene Kategorien eingeteilt sind. Ziel dieser Aufgabe ist es, einen Naive Bayes Classifier (NBC) zu trainieren, der in der Lage ist ein Test-Set von Texten einer der 20 Kategorien zuzuordnen. Der NBC soll dabei denselben Algorithmus verwenden, der in der Vorlesung vorgestellt wurde. Das Test-Set soll 20% des Datensatz betragen. Zum Randomisieren der Reihenfolge der Daten verwenden Sie bitte `random_state=20`, ein Parameter der `train_test_split` Methode. Um den NBC zu trainieren, muss der Text zunächst in eine Vektordarstellung überführt (vektorisiert) werden. Hierzu sollen Sie eine in der Praxis verbreitete Vektorisierungsmethode verwenden: term frequency - inverse document frequency (TF-IDF). Hierbei wird relative Häufigkeit eines Wortes in einem Text über das Auftreten des Wortes in allen Texten des Trainings-Datensatzes normalisiert, wie auf der letzten Seite des Übungsblatts noch einmal zusammengefasst. Nachdem Sie ihren Classifier trainiert haben, erstellen Sie zunächst eine Konfusionsmatrix C ihrer Vorhersagen, wobei der Eintrag C_{ij} darstellt, wieviele der Texte aus Kategorie i stammen und als Kategorie j vorhergesagt wurden. Geben sie außerdem die Accuracy des NBC Modells an.

Part 2

Das Datenset `spam_or_not_spam.csv` enthält ca. 2900 Beispiele für E-Mails, die entweder als Spam oder als Nicht-Spam kategorisiert sind. Die Klasse der E-Mails, die Spam zugeordnet sind, ist hier jedoch deutlich kleiner als die Nicht-Spam Klasse. Führen sie dieselben Schritte wie in Part 1 durch. Berechnen Sie nicht nur die Accuracy des Modells, sondern zusätzlich Precision, Recall und F1-Score für beide Klassen. Wie ist die Performance des Modells zu bewerten? Eignet sich das Modell als Spam-Filter?

Test-Datensatz (optional!)

Um ihre Lösung zu testen steht Ihnen der Datensatz `test_dataset.csv` zur Verfügung. Wenn Sie die Anweisung zur Randomisierung und Größe des Test-Datensatz wie in Part 1 + 2 verfolgen,

und das Modell korrekt implementiert haben, sollten Sie den Test-Datensatz mit einer Accuracy von 0.75 vorhersagen können.

Bonus (optional!)

Wenn sie gerne mehr über die Funktionsweise von (TF-IDF) erfahren wollen, versuchen sie folgende Fragen mit Hilfe des scikit-learn Vektorisierers bzw. der Kombination aus `CountVectorizer` und `TfidfTransformer` für das zweite Datenset zu beantworten:

- Was sind die 10 häufigsten Wörter, die im Text-Corpus (Trainingsdatensatz) vorkommen?
- Welche 10 Wörter kommen selten in Dokumenten vor? Wie ist ihre IDF?
- Welche 10 Wörter haben den höchsten TF-IDF Wert in allen Spam-Mails des Trainingsdatensatz?

Zu dieser Bonus-Aufgabe wird es keine Musterlösung geben!

Bitte lösen Sie die Fragen und die Programmieraufgaben bis zum **4. Dezember 2023**. Ihren Python-Code, sowie Visualisierungen können Sie in Form eines Jupyter Notebooks oder PDFs hochladen.

Mathematische Definition: TF-IDF

- raw count: $w_{i,j}$ - how often does word j appear in document i
- m - total number of documents in the corpus
- term frequency (TF):

$$\text{TF} = \frac{w_{i,j}}{\text{length of document } i}$$

- inverse document frequency (IDF):

$$\text{IDF} = \log\left(\frac{m}{\text{number of documents containing word } j}\right)$$

  or its smoothed version

$$\text{IDF} = \log\left(\frac{1+m}{1+\text{number of documents containing word } j}\right)$$

- TF-IDF score: TF-IDF = TF $\cdot$ IDF