Programmierabgabe 2

Abgabe

Die Abgabe erfolgt durch Upload eines Zip-Files in Moodle. Ein genauer Abgabeort wird rechtzeitig bekannt gegeben. In diesem Zip-File müssen sich die folgenden Dateien befinden:

- Alle Python-Skripte, die zur Durchführung der Aufgabe nötig sind.
- Ein kurzes Readme-File, das beschreibt, wie Ihr ML-System zu bedienen ist.
- Ein kurzer Report (ca. 1 Seite, kann auch etwas kürzer oder länger ausfallen), in dem Sie beschreiben, welche Parameter/Methoden Sie gewählt haben und aus welchem Grund. Präsentieren Sie außerdem Ihre Untersuchungsergebnisse in geeigneter Form. Wenn der Vergleich verschiedener Parameter/Methoden Bestandteil Ihrer Aufgabe war, gehen Sie an dieser Stelle auch darauf ein.
- Eine Erklärung, wie Sie die Arbeit innerhalb Ihres Teams aufgeteilt haben.

Anforderung an das Gesamtsystem ist, dass es sich nach Lesen des Readme-Files und Anpassen des Pfades zum Datenset und gegebenenfalls weiterer nötiger Parameter direkt ausführen lässt. Sollten Sie aus irgendeinem Grund weitere Bibliotheken als tensorflow, numpy, scipy, sklearn, skimage und matplotlib verwenden, geben Sie diese zusätzlich benötigten Pakete in Ihrem Readme-File an.

Hinweis: Da sich Ihr Kurs für eine gemeinsame Abgabefrist für beide Teilaufgaben entschieden hat, besteht auch die Möglichkeit, statt separaten Kurzreports, Readme-Files etc. beide Teilaufgaben in gemeinsamen Files zu behandeln.

Datensets

Wie auch für die erste Teilaufgabe gilt: Da das Training mit den vollständigen Datensets - je nach gewähltem Klassifikator und je nach Datenset - eine ganze Weile Dauern kann, empfiehlt es sich, erste Versuche (z.B. für die Wahl geeigneter Hyperparameter oder zur Auswahl geeigneter Methoden für die weiteren Untersuchungen) mit einem kleineren Subset durchzuführen. Die Ergebnisse solcher Anfangsversuche werden zwar in den meisten Fällen nicht sehr gut sein, geben aber in der Regel trotzdem einen Anhaltspunkt dafür, ob Parameter X oder Y bzw. Methode A oder B vielversprechender ist. Verwenden Sie das gesamte Datenset erst, wenn Sie sich für Methoden bzw. Parameter entschieden haben und ihr endgültiges Modell evaluieren möchten.

Es ist sinnvoll, zur Evaluation Ihrer Ergebnisse eine Kreuzvalidierung durchzuführen, d.h. Ihr System mehrmals mit unterschiedlichen Trainings- und Testsplits zu trainieren. Um reproduzierbare Ergebnisse zu erhalten und somit die Auswirkung von Änderungen am ML-System sinnvoll bewerten zu können, empfiehlt sich für zufälliges Splitting die explizite Initialisierung per random seed für jeden Trainings-/Testsplit. In den Aufgaben ist 3-fold-Kreuzvalidierung angegeben, eine höhere Anzahl an Durchläufen ist jedoch gerne gesehen (sofern von den Ressourcen Ihrer PCs möglich, höhere Anzahl Durchläufe beeinflusst die Note nicht).

Wenn Sie mit einem unbalanced Datenset arbeiten, d.h. manche Klassen deutlich häufiger vorkommen als andere, beachten Sie diesen Umstand bei Ihrer Evaluation und Ergebnisdarstellung, indem Sie geeignete Evaluationsmetriken verwenden

Deep Learning in Python mit Keras und Tensorflow

Für die generelle Implementierung von Deep-Learning-Modellen mit Keras/Tensorflow finden Sie hilfreiche Informationen und Beispiele in der Dokumentation unter:
`https://keras.io/getting_started/`
`https://www.tensorflow.org/overview`

Kurzreport

Da Sie nach Abschluss der zweiten Teilaufgabe sowohl mit klassischen, merkmalsbasierten ML-Methoden als auch mit Deep Learning das gleiche Problem bearbeitet haben, führen Sie bitte in Ihrem Kurzreport zusätzlich zu den reinen Inhalten der zweiten Teilaufgabe eine separate Diskussion, in der Sie darauf eingehen, welche Unterschiede Sie mit beiden Teilaufgaben in der Implementierung erlebt haben, welche Methoden Sie bevorzugt auswählen würden und warum.

Allgemeine Hinweise

Es ist im Vorraus ohne genaue Kenntnis Ihrer PC-Setups leider schwierig, die Rechenzeiten der einzelnen Aufgabenschritte abzuschätzen. Sollten Sie während Ihrer Aufgabe Schwierigkeiten bezüglich der Ihnen zur Verfügung stehenden Rechenkapazitäten feststellen (d.h. das Klassifikator-Training dauert zu lange), melden Sie sich daher bitte direkt bei mir. Dann können wir gemeinsam nach einer Lösung suchen und gegebenenfalls die Aufgabe entsprechend abändern.

Wie auch bei der ersten Teilaufgabe ist es möglich, dass Sie bei Ihren Aufgaben keine sehr hohe Klassifikator-Performance erzielen. Wie gut ein ML-System funktioniert, hängt schließlich von sehr vielen unterschiedlichen Faktoren ab und der Weg zu einer sehr guten Performance ist teilweise lang. **Bitte behalten Sie deshalb im Hinterkopf, dass die finale Performance (z.B. höhe der Accuracy) NICHT für Ihre Note entscheidend ist. Diese Programmierabgabe soll hauptsächlich zeigen, dass Sie grundsätzlich in der Lage sind, ein komplettes ML-System in Python aufzubauen, d.h. dass Sie die einzelnen Bestandteile und deren Notwendigkeit verstanden haben.** Wichtig ist deshalb, dass Sie eine saubere, gut kommentierte Implementierung abliefern, die (in Kombination mit dem Kurzreport) deutlich macht, dass Sie strukturiert vorgegangen sind und sinnvolle Überlegungen zur Wahl von Parametern oder Methoden angestellt haben.