Programmierabgabe 1

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 numpy, scipy, sklearn, skimage und matplotlib verwenden, geben Sie diese zusätzlich benötigten Pakete in Ihrem Readme-File an.

Falls Ihr Team sich für die Aufgabe für ein Subset eines Datensets (d.h. nur einen Teil der verfügbaren Klassen) entscheiden soll: Laden Sie die von Ihnen verwendeten Daten ebenfalls hoch.

Datensets

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

Merkmalsextraktion

Zur Extraktion von Merkmalen aus Bildern ist das Paket Scikit Image hilfreich. Die Dokumentation finden Sie unter `https://scikit-image.org/`. Allgemein lässt sich sagen, dass sich Merkmale gut eignen, die von den Graustufen- oder Farb-Histogrammen oder der Entropie von Bildern abgeleitet werden. Weitere Stichworte für mögliche Merkmale (Farbbilder müssen hierfür ggfls in Graustufen konvertiert werden): Gray-Level-Cooccurance-Matrix (GLCM), Gray-Level Run Length, Gray-Level Zone Size Matrix. Auch von Transformationen abgeleitete Merkmale können sehr hilfreich sein, beispielsweise Merkmale abgeleitet von Kantendetektionen (z.B. über Gradientenfilter, Faltung o.ä.).

Machine Learning in Python

Für die generelle Implementierung von ML-Systemen mit Scikit Learn finden Sie hilfreiche Informationen und Beispiele im User Guide von Scikit-Learn unter "Getting Started":
`https://scikit-learn.org/stable/getting_started.html`

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.

Falls Ihre Gruppe sich für einen geeigneten Klassifikator entscheiden muss: Da sich die zweite Programmierabgabe mit dem Thema Deep Learning beschäftigen wird, nutzen Sie in dieser Abgabe bitte konventionelle ML-Methoden und *keine* neuronalen Netze!

Es ist 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.