Aufgabenstellung und Randbedingungen

Der Perzeptron-Lernalgorithmus und die Delta-Regel stellen zwei der einfachsten Lernverfahren für künstliche Neuronen dar. Moderne künstliche neuronale Netze verwenden dagegen Weiterentwicklungen des sog. Backpropagation-Lernalgorithmusses.

Ziel dieser Praktikumsaufgabe ist, diese Algorithmen in ihrer Funktionsweise genauer zu verstehen und sie anwenden und modifizieren zu können.

Teilversuch 1:

- Implementieren Sie den Perzeptron-Lernalgorithmus wie er in der Vorlesung vorgegeben wurde. Testen Sie Ihre Implementation auf den Datensatz, der in dem Jupyter-Notebook `Perceptron_dataset.ipynb` definiert ist.
- Erweitern Sie dazu den Datensatz um eine weitere Dimension um auch affin lineare Trennfunktionen lernen zu können.
- Wie viele Schleifendurchläufe benötigt Ihr Algorithmus, um alle Daten korrekt zu klassifizieren? Testen Sie dies für 10 zufällige Initialisierungen des Gewichtsvektors (standardnormalverteilt).
- Plotten Sie die Accuracy (den Prozentsatz der korrekten Klassifikationen) über die Zeit (ein Wert je Schleifendurchlauf) für die 10 Durchläufe.
- Plotten Sie außerdem die Länge des Gewichtsvektors für die 10 Durchläufe.

Teilversuch 2:

- Implementieren Sie die Delta-Regel als Variante des Perzeptron-Lernalgorithmus wie sie in der Vorlesung vorgegeben wurde. Testen Sie Ihre Implementation auf den Datensatz, der in dem Jupyter-Notebook `Perceptron_dataset.ipynb` definiert ist.
- Erweitern Sie dazu den Datensatz um eine weitere Dimension um auch affin lineare Trennfunktionen lernen zu können.
- Wie viele Schleifendurchläufe benötigt Ihr Algorithmus, um alle Daten korrekt zu klassifizieren? Wählen Sie hierzu einmalig einen zufälligen Gewichtsvektor und testen Sie 10 unterschiedliche Schrittweiten ($\eta = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$).
- Plotten Sie die Accuracy (den Prozentsatz der korrekten Klassifikationen) über die Zeit (ein Wert je Schleifendurchlauf) für die 10 Durchläufe.
- Plotten Sie außerdem die Länge des Gewichtsvektors für die 10 Durchläufe.

Teilversuch 3:

In dem Jupyter-Notebook `SimpleNeuralNetwork.ipynb` ist ein einfaches künstliches neuronales Netz inklusive einer Implementation des Backpropagation-Lernalgorithmus` gegeben.

Da dieser Algorithmus zu Ihrem Praktikumstermin möglicherweise noch nicht präsentiert wurde, können sie sich hier bereits über das Verfahren informieren:

- **Grober Überblick https://de.wikipedia.org/wiki/Backpropagation**
- **Gutes Buch http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/1996/NeuralNetworks/neuron.pdf (Kapitel 7)**
- **Gutes Video: https://www.youtube.com/watch?v=Ilg3gGewQ5U**

Auch den Begriff Accuracy sowie die unterschiedlichen Aktivierungsarten schlagen Sie bitte zunächst nach. Ich werde versuchen die Folien hierzu zeitnah online zu stellen.

Versuchen Sie anschließend den Code nachzuvollziehen und bearbeiten Sie dann die folgenden Teilaufgaben:

- Erweitern Sie das neuronale Netz um die Möglichkeit, die Trainings-Accuracy und auch die Test-Accuracy während des Trainings mitzuloggen, sowie anschließend den Verlauf als jeweils Liste abzufragen. Trainieren Sie anschließend das Netz auf dem gegebenen Datensatz und plotten Sie den Verlauf der Accuracy-Werte. Beachten Sie, dass dem Netz hierzu auch die Testdaten bekannt sein müssen.
- Erweitern Sie das neuronale Netz um die Möglichkeit, den Trainings-Loss und auch den Test-Loss während des Trainings mitzuloggen, sowie anschließend den Verlauf als jeweils Liste abzufragen. Trainieren Sie anschließend das Netz auf dem gegebenen Datensatz und plotten Sie den Verlauf der Loss-Werte.
- Erweitern Sie das Netzwerk um die Möglichkeit, per Parameter bei der Erstellung eines Netzes zwischen drei verschiedenen Arten von Aktivierungsfunktionen zu wählen: (1) logistische Aktivierung (bereits implementiert), (2) TanH-Aktivierung und (3) ReLU-Aktivierung. Testen Sie das Netz jeweils auf dem gegebenen Datensatz und plotten Sie jeweils den Verlauf von Accuracy und Loss auf den Trainings- und Testdaten.
- Erweitern Sie das Netzwerk um die Möglichkeit, Minibatches statt dem vollen Batch je Lernschritt zu verwenden. Passen Sie die Log-Funktionen entsprechend an. Die Trainingsperformance (Accuracy und Loss) soll je Mini-Batch geloggt werden und die Testperformance je ganzem Batch. Beachten Sie, dass die Trainingsdaten nach je einer Epoche neu durchmischt werden müssen.

Abnahme:

Abgenommen werden die (gut kommentierten) Jupyter-Notebooks der entwickelten Algorithmen und -erweiterungen, sowie deren Funktionstests. Je Teilversuch soll ein separates Jupyter-Notebook erstellt werden, Teilversuch 3 kann auch in vier separate Jupyter-Notebooks unterteilt werden.