Prof. Dr. Rüdiger Brause
Robert-Mayer-Str. 11-15
Tel.: +049-(0)69-798-23977
R.Brause@cs.uni-frankfurt.de

Markus Hildebrand
markhild@stud.uni-frankfurt.de

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Vorlesung Adaptive Systeme WS 13/14
Übungsblatt 3

Ausgabe: 05.11.2013
Abgabe: 12.11.2013

Adaptive Systeme 1

Aufgabe 3.1 Perzeptron (10 Punkte)

Programmieren Sie ein neuronales Netz mit der Perzeptron-Lernregel in Python. Verwenden Sie dafür die Lernregel Perzeptron3 (siehe Vorlesungsfolien) mit einer Lernrate γ von 0.1.
Trainieren Sie das Netz nun mit ihren gesammelten Daten von Übungsblatt 2 auf die Klassifizierung des Geschlechts von Personen anhand von zwei Merkmalen.

Hinweis: Sie sollten ihre Daten zur Klassifizierung auf eine Größenordnung um 1 skalieren, sonst wird das Konvergieren des Netzes mit der Lernrate von 0.1 lange dauern.
Wenn ihr System mit den Daten nicht konvergiert, weil die Daten nicht linear separierbar sind, verwenden sie eine Zeitabhänge Lernrate von $\gamma = 1/t$

Aufgabe 3.2 AdaLinE (10 Punkte)

Programmieren Sie nun ein weiteres Neuronales Netz in Python, diesmal ein AdaLinE mit der Widrow-Hoff Lernregel.
Trainieren Sie es wieder mit den bekannten Daten aus Aufgabe 2.1.

Adaptive Systeme 2

Aufgabe 3.3 Perzeptron (10 Punkte)

Programmieren Sie ein neuronales Netz mit der Perzeptron-Lernregel in Python. Verwenden Sie dafür die Lernregel „Perzeptron1“ (siehe Vorlesungsfolien) mit einer Lernrate γ von 0.1.
Trainieren Sie das Netz mit den Irisdaten von der Webseite um Blütensorten zu unterscheiden.
Hinweis: Die Daten können Sie sehr einfach mit numpy einlesen, beispielsweise so:

```
dataset = numpy.loadtxt("irisdaten.txt", skiprows=(2))
```

Klassifizieren sie anhand von SepalLength und SepalWidth (Die ersten beiden Spalten der Irisdaten).

Aufgabe 3.4 AdaLinE (10 Punkte)

Programmieren Sie nun ein weiteres Neuronales Netz in Python, diesmal ein AdaLinE. Verwenden sie die folgende, in der Vorlesung hergeleitete Lernregel:

$$w(t) = w(t-1) - \gamma(t)(w^T x - L(x))x$$

Implementieren sie auch dass die Lernrate abhängig von der Zeit ist (zB mit $\gamma = c/t, c = 0.1$)
Trainieren sie es wieder mit den Irisdaten.

Vergleichen sie die Konvergenzgeschwindigkeit (d.h. Wie viele Iterationen benötigt es bis alle Eingaben richtig klassifiziert werden) dieses Netzes mit der des Perzeptrons.

Hinweise zu Programmieraufgaben in Python:

- *Wir haben mit Python 2.7 immer noch bessere Erfahrungen gemacht als mit 3.x, weil einige 3rd Party Pakete noch nicht auf 3.x umgestellt wurden. Sie können auf eigene Gefahr natürlich trotzdem verwenden was Sie möchten.*
- *Verwenden Sie am besten Numpy (www.numpy.org) als Vektor/Matrixbibliothek. Numpy bietet ihnen spezialisierte Datentypen für Vektoren und Matrizen, die auf hohe Geschwindigkeit optimiert sind und eignet sich gut für unsere Anforderungen.*
- *Wenn Sie lieber in einer richtigen IDE arbeiten als in Pythons IDLE, nutzen Sie dass pydev Plugin für Eclipse. (http://pydev.org/ , http://www.eclipse.org/)*
- *Mit der Bibliothek matplotlib (http://matplotlib.org/) können Sie einfach Grafische Ausgaben ihrer Daten erzeugen. Zum Überprüfen Ihrer Ergebnisse bietet es sich sehr oft an diese einfach mal zu visualisieren.*
- *Sie können gerne andere/weitere Erweiterungen und Bibliotheken verwenden, aber machen sie die verwendeten Komponenten bitte in der Dokumentation deutlich (mit Link zum Projekt)*
- *Zur Abgabe jeder Programmieraufgabe gehört eine **Dokumentation!***
  *Bei Aufgaben wie dieser eignet sich am besten eine **Entwicklungsdokumentation**:*
  -Was war die Aufgabe? Wie soll die Aufgabe gelöst werden? Wie gingen Sie dabei vor? Gab es bei der Entwicklung bemerkenswerte Zwischenstationen? (Ansätze die nicht funktionierten etc.)
- *Geben Sie wenn nötig auch eine **Benutzerdokumentation** an – wie ist das Programm zu benutzen und wie ist die Ausgabe zu interpretieren? (Dies ist nicht nötig wenn dass Programm keine Benutzereingaben erfordert und die Ausgaben klar selbsterklärend sind).*
- *Schließlich sollten sie auch eine **Testdokumenation** erstellen: Welche Tests zeigen, dass Sie keine Programmierfehler haben? Was sind geeignete Tests und Testmuster?*
- *Wenn bis zur Abgabe dass Programm immer noch nicht wie gewünscht funktioniert – keine Panik! Dokumentieren Sie was nicht funktioniert, wo Sie den Fehler vermuten und was sie versucht haben um das Problem zu lösen. Eine solche Abgabe gibt immer noch anteilig Punkte.*
- *Dass der Programmcode selbst kommentiert sein soll versteht sich von selbst.*