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 4

Ausgabe: 12.11.2013
Abgabe: 26.11.2013

Adaptive Systeme 1

Aufgabe 4.1 Perzeptron Offline-Learning (8+2 Punkte)

Auf dem letzten Aufgabenblatt hatten wir mit unserem Perzeptron ein Online-Lernverfahren angewandt. Dass bedeutet: Wir haben für jedes Muster, das wir dem Netz vorgelegt haben, die Gewichte angepasst. Nun wollen wir einmal dass Ganze im folgenden Verfahren ausprobieren und mit unseren bisherigen Ergebnissen vergleichen.

a) Offline-Learning: Anstatt wie bisher zufällige Muster auszuwählen und die Gewichte sofort anzupassen (Online-Learning), geben wir nun jedes Muster in unserem Datensatz einmal ins Netzwerk und sammeln die Änderungen, ohne die Gewichte zu ändern. Am Ende summieren wir alle Änderungen auf, bilden den Mittelwert und aktualisieren dann erst, in einem Schritt, die Gewichte. Dies wird als Offline-Learning bezeichnet. Dass Ganze wiederholen wir wieder solange, bis das Netz den Datensatz korrekt klassifizieren kann.

Zum Training und Geschwindigkeitsvergleich nutzen Sie die Irisdaten (siehe Blatt 3, Aufgabe 3.3. Als Klassifizierungsmerkmale nehmen sie auch hier SepalWidth und SepalLenght.). Finden Sie diesmal selbst eine geeignete Lernrate!

b) Wie lange benötigt dieser Ansatz um so weit zu konvergieren, dass alle Muster richtig klassifiziert werden, im Vergleich zum bisherigen Vorgehen?

Aufgabe 4.2 XOR-Problem / Aktivität in Mehrschichtennetzwerken (10 Punkte)

Wir wollen nun ein neuronales Netz mit der Perzeptron-Lernregel lernen lassen, welches ein logisches XOR Gatter modelliert. Wie Sie in der Vorlesung gehört haben, lässt sich dies mit einem einzelnen Perzeptron nicht umsetzen, da die Muster nicht linear separierbar sind.
Trainieren Sie hierzu zwei Schichten mit Perzeptron-Lernregel getrennt. Trainieren Sie die erste Schicht auf die Funktionen:

$$y_1 = x_1 \wedge \bar{x}_2 \,,\; y_2 = \bar{x}_1 \wedge x_2$$

Und die zweite Schicht auf:

$$z = y_1 \vee y_2$$

Nach dem Training der ersten Schicht werden die Gewichte der ersten Schicht nicht mehr verändert. Dann trainieren sie die zweite Schicht bis auch diese ihre Funktion richtig abbildet. Beide Schichten zusammen sollen dann eine XOR-Funktion abbilden können.

Adaptive Systeme 2

Aufgabe 4.3 Newton Iteration (6 + 4 Punkte)

a) Implementieren sie ein AdaLinE welches als Gradientenabstieg auf der Zielfunktion die Newton Iteration verwendet. Als Zielfunktion verwenden sie:

$$R(w) = \langle |L - z| \rangle$$

Trainieren Sie es mit den bekannten Irisdaten.

b) Verleiht die Newton-Iteration einer Zielfunktion die Eigenschaften einer Ljapunow Funktion? Wenn ja, warum, wenn nein, warum nicht?

Aufgabe 4.4 Probleme bei Stochastischem Lernen (4 Punkte)

Stochastische Iteration: $w(t)=w(t-1)-\gamma(t)(w(t-1)-x(t))$

In der Vorlesung wurde gezeigt, dass die Gewichte durch stochastisches Lernen zum Erwartungswert der Zielfunktion für alle Muster konvergieren. (Folien, Abschnitt: Stochastische Iteration: Konvergenz):

Bei $\gamma(t):=\frac{1}{t}$ *ist immer* $w(t)=\langle x\rangle_x$

Aber: Für die Iteration von Klassenprototypen w_i (Klassentrennung) gilt dies nicht mehr!

Bei $w_i(t)=w_i(t-1)-\gamma(t)(w_i(t-1)-x(t))$ ist dann: $w_i(t)\neq\langle x\rangle_x$.

Warum ist dass so? Erklären sie die Problematik!

Aufgabe 4.5 Stochastische Approximation (2 + 3 + 3 Punkte)

Zeigen Sie dass die Lernrate $\gamma=\frac{a}{(b+t)}$ für eine stochastische Approximation geeignet ist!