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 8

Ausgabe: 17.12.2013
Abgabe: 14.01.2014

Adaptive Systeme 1

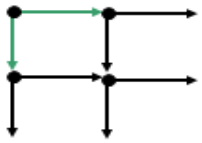
Aufgabe 8.1 Kohonennetze (14 + 6 Punkte)

a) Implementieren Sie eine Kohonenkarte mit den folgenden Eigenschaften:
Eingabe ist ein 2D-Vektor, die Ausgabe ist ebenfalls 2D. Verwenden Sie 144 Neuronen (12*12) und 100.000 Iterationen.

Als Eingabe erzeugen Sie zufällige, uniform verteilte Punkte im Bereich -1 bis +1 (für jede Dimension). Trainieren Sie mit diesen Eingabe ihr Kohonennetz.

Erzeugen Sie bei 50, 100, 500, 1.000, 10.000 und 100.000 Iterationen eine grafische Ausgabe ihres Netzes. Stellen Sie die Gewichte als Punkte dar und zeichnen Sie von jedem Gewicht zu seinem nächsten und darunter liegenden Nachbarn eine Verbindungslinie ein.

Zeichnung :



Hinweise:

- Sie können auch in eine Bilddatei plotten anstatt ein Fenster mit grafischer Ausgabe zu erzeugen. Dies geschieht mit dem `savefig` Befehl, z.B. `plt.savefig('fig1.png', dpi=72)`.
- 100.000 Iterationen dauern sehr lange: Verwenden Sie deshalb beim Testen nur wenige 1.000 Iterationen. Das sollte schon genügen um zu sehen, ob ihr Netz sich in die richtige Richtung entwickelt oder Unsinn produziert.

Verwenden Sie *eine* der folgenden Varianten für die restlichen Parameter:

Lernrate = 0.2, h() = 1 wenn Neuron in Nachbarschaft, sonst 0.
Starten Sie mit einem Nachbarschaftsmaß von 4, reduzieren Sie es ab 100 Iterationen auf 3, ab 1.000 Iterationen auf 2 und ab 10.000 Iterationen auf 1.

Alternativ können Sie auch eine Nachbarschaftsfunktion für alle Neuronen verwenden:

$$
\begin{aligned}
h(\underline{p}_k, \underline{p}_*, t) &= \exp\left(-\frac{||\underline{p}_k - \underline{p}_*||_2^2}{\sigma(t)^2}\right) \\
\sigma(t) &= \sigma_0 \cdot \exp(-t/\tau_1) \\
\eta(t) &= \eta_0 \cdot \exp(-t/\tau_2) \\
t &= 0, 1, 2 \ldots
\end{aligned}
$$

Legende:
p_k : *Neuron k* , p_* : *Ausgewähltes Neuron*
t : *Iterationsschritt* , T : *Gesamtanzahl Iterationen*
σ_0 : *Anfangsradius Nachbarschaft* $= 8$
$\eta(t)$: *Lernrate* , η_0 : *Startlernrate* $= 0.4$
$\tau_1 : -\left(\frac{T}{\ln(0.01/\sigma_0)}\right) \tau_2 : -\left(\frac{T}{\ln(0.1/\eta_0)}\right)$

Die zweite Variante ist komplizierter zu implementieren, aber konvergiert schneller und führt zu einem besseren Ergebnis.

b) Trainieren Sie eine Kohonenkarte nun statt mit uniform verteilten Punkten mit einer Gauss'schen Zufallszahlenquelle. Verwende dazu folgenden Befehl von ***numpy***:

numpy.random.normal(loc=0.0, scale=0.5, size=None) (jeweils für beide Achsen)

Verwenden Sie diesmal 576 Neuronen (24*24), 10.000 Iterationen und, wenn Sie die Variante mit der Exponentialfunktion implementiert haben, als Anfangsradius 16 für die Nachbarschaft. Ansonsten verwenden Sie dieselben Parameter wie in Aufgabenteil a).

- Erzeugen Sie an denselben Iterationsschritten wie in Aufgabe a) wieder grafische Repräsentationen ihres Netzes.
- Vergleichen Sie die Ergebnisse mit denen aus Aufgabe a).

Adaptive Systeme 2

Aufgabe 8.2 Fixpunkte (10 Punkte)

Bestimmen Sie die Fixpunkte und ihre Art (stabil oder labil) von der Gleichung:

$$w(t+1)=g(w)=w^3-6w^2+12w-6$$

Aufgabe 8.3 TLMSE (10 Punkte)

Approximieren Sie eine Gerade anhand des Datensatzes der Aufgabe 6.1.
Kombinieren Sie dazu Test, Trainings und Validierungsmenge zu einem Datensatz aus Vektoren **x**. Beachten Sie, dass die neuen **x** als (n+1)-te Komponente auch $f(x_1,...,x_n)$ enthalten.
Bestimmen Sie zur Minimierung des Total Least Mean Square Error den relevanten Eigenvektor (Hinweise: PCA, minimaler Eigenwert).

Geben Sie die gefundene Geradengleichung sowohl in Hessescher Normalform **xw**-c als auch in der Geradengleichungsform (y = mx+b) an. Wie lauten w, c, m und b?