ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
INSTITUT FÜR INFORMATIK
Lehrstuhl für Mustererkennung und Bildverarbeitung
Prof. Dr.-Ing. Hans Burkhardt

Georges-Köhler-Allee
Geb. 052, Zi 01-029
D-79110 Freiburg
Tel. 0761 - 203 - 8260

Übungen zur Vorlesung
Grundlagen der Bilderzeugung und Bildanalyse (Mustererkennung)
WS 03/04

Aufgabenblatt 11 (12 Punkte)

Vorlesungsstoff: bis ME-II, Kap. 8b, S.56

Abgabe am Donnerstag, 15.01.2004, vor der Übung

Bitte Name und Matrikelnummer auf den Lösungen angeben.

Aufgabe 11.1: Backpropagation (4 Punkte)

Um sich von den Nachteilen des Gradientenalgorithmus zu befreien, sind etliche Vorschläge gemacht worden, darunter auch Ausweichen auf Variationen des Newton-Algorithmus. Letzterer erfordert jedoch die zweiten Ableitungen der Kostenfunktion J nach den Gewichten, die in der Hesse-Matrix

$$\frac{\partial^2 J}{\partial w_{jk}^q \partial w_{nm}^r}$$

zusammengefasst werden. Leiten Sie die Hesse-Matrix für ein einfaches Zwei-Schicht-Neuronalnetz her, das zwei verdeckte Neuronen und ein Ausgangsneuron hat.

Aufgabe 11.2: Backpropagation (4 Punkte)

Wie modifiziert sich der in der Vorlesung angegebene Backpropagation-Algorithmus, wenn als Aktivierungsfunktion die Funktion

$$\psi(x) = \frac{1}{1 + e^{-a(x-b)}}$$

verwendet wird?

Aufgabe 11.3: Programmieraufgabe: Perceptron ohne Offset (4 Punkte)

Auf der Kursseite finden Sie in der Datei `data10_3.dat` 3 Datensätze $\mathbf{X}_1, \mathbf{l}_1$, $\mathbf{X}_2, \mathbf{l}_2$ und $\mathbf{X}_3, \mathbf{l}_3$, die für folgende Aufgabe benötigt werden.

1. Implementieren Sie den Perceptron-Algorithmus aus der Vorlesung `[w, J] = train_perceptron(X, l, rho, niter)` Hierbei ist $\mathbf{X}$ eine $n \times N$ Matrix, deren Spalten die N beliebigdimensionalen Trainingsvektoren darstellen. $\mathbf{l} \in \{1, 2\}^N$ stellt die Klassenzugehörigkeit der Trainingsvektoren dar. n_{iter} ist die Anzahl der Lerniterationen, ρ ist ein Faktor, der das exponentielle Abfallen/Anwachsen der Lernrate in Iteration $i = 1, \ldots, n_{iter}$ definiert via $\rho_i = \rho^{i-1}$. Ergebnis der Funktion soll der trainierte Normalenvektor $\mathbf{w}$ sein, und ein Vektor $\mathbf{J}$, der die Werte $J(\mathbf{w})$ der Optimierungsfunktion in allen Iterationen auflistet. Die Initialisierung für $\mathbf{w}$ soll einfach über den Nullvektor geschehen.

   Hinweis: die Vorzeichen in der Definition der δ_x auf Folie ME-I, Kap. 8a 32 müssen vertauscht werden!

2. Trainieren Sie ein lineares Perceptron auf den drei Datensätzen $\mathbf{X}_i, \mathbf{l}_i$ für $i = 1, 2, 3$ mit $\rho = 1$ und $n_{iter} = 1000$. Geben Sie die resultierenden Normalenvektoren $\mathbf{w}$ an, und plotten Sie jeweils $\mathbf{J}$. Interpretieren Sie diese Diagramme.

3. Stellen Sie den zweiten Datensatz grafisch dar, und geben Sie zwei Möglichkeiten an, wie durch Modifikation der Daten erfolgreiches Perceptron-Training mit obiger Routine erfolgen kann.

4. Stellen Sie ebenfalls den dritten Datensatz grafisch dar. Ist hier eine erfolgreiche Klassifikation aller Daten mit obiger Routine möglich? Wenn ja, wie?

Der Code ist durch Kommentarzeilen ausführlich zu dokumentieren.