Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Dr. Jakob Bernasconi

Informationsverarbeitung in neuronalen Netzwerken

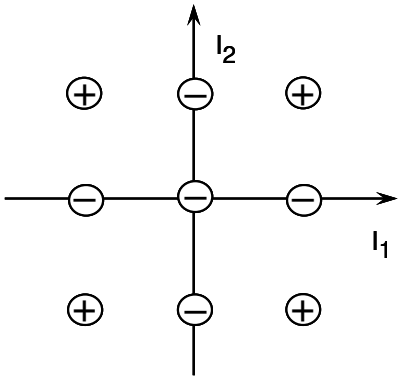
Frühjahrssemester 2009

Übung 4

Ausgabe am 3. April 2009
Abgabe am 24. April 2009

Versuchen Sie, mit einem Multilayer-Perceptron eine Funktion $O(I_1, I_2)$ zu modellieren, welche die folgenden 9 Lerndaten möglichst genau approximiert:

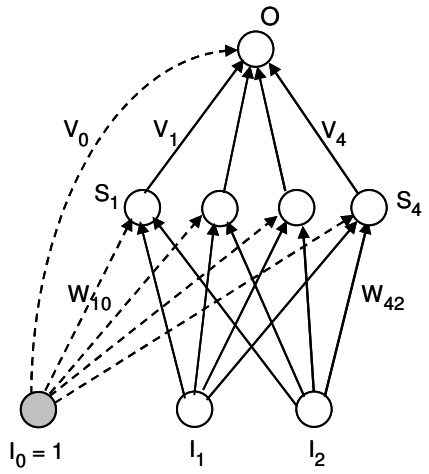
ν	I_1	I_2	D
1	–1	–1	+1
2	0	–1	–1
3	+1	–1	+1
4	–1	0	–1
5	0	0	–1
6	+1	0	–1
7	–1	+1	+1
8	0	+1	–1
9	+1	+1	+1



(D = gewünschter Output)

Aufgabe 1

Zeigen Sie analytisch, dass ein Multilayer-Perceptron mit 4 Hidden Units und mit Signum-Aktivierungsfunktionen den obigen Satz von Lernbeispielen korrekt klassifizieren kann:



wobei: $O = \mathrm{sign}\left(V_0 I_0 + \sum_{i=1}^{4} V_i S_i \right)$,

$S_i = \mathrm{sign}\left(\sum_{j=0}^{2} W_{ij} I_j \right)$, $\quad i = 1, 2, 3, 4.$

Hinweis:

Wählen Sie die Gewichte W_{ij} wie folgt,

$W_{10} = 0.5$	$W_{11} = 1$	$W_{12} = 0$
$W_{20} = 0.5$	$W_{21} = -1$	$W_{22} = 0$
$W_{30} = 0.5$	$W_{31} = 0$	$W_{32} = 1$
$W_{40} = 0.5$	$W_{41} = 0$	$W_{42} = -1$

und bestimmen Sie jeweils die durch die 4 Hidden Units definierten Trenngeraden und die entsprechende Unterteilung des (I_1, I_2) – Raums.

Danach untersuchen Sie, ob die Gewichte V_0 bis V_4 so gewählt werden können, dass der Output O des Multilayer-Perceptrons alle 9 Lernbeispiele richtig klassifiziert.

Aufgabe 2

Versuchen Sie, das Problem mit Hilfe des Error-Backpropagation-Lernalgorithmus zu lösen.

Für Ihre Untersuchungen steht Ihnen auf der Vorlesungs-Homepage ein Backpropagation-Simulator zur Verfügung. (Sie können natürlich auch ein eigenes Backpropagation-Programm schreiben!)

Der Simulator wurde im Sommersemester 2000 von Thomas Singer, Rolf Sigg und Rita Mayer-Sommer (D-ELEK) geschrieben. Mit Hilfe der folgenden Anleitung sollten Sie ihn problemlos verwenden können [unter Windows NT, Windows 95, Windows 98, Windows 2000 oder Windows XP].

Die Multilayer-Perceptrons, die Sie für Ihre Untersuchungen verwenden, müssen zwei Inputs und einen Output haben. Spielen können Sie mit den folgenden Netzwerk- und Lernparametern:

- Anzahl Hidden Layers (1 oder 2)
- Anzahl Neuronen in den Hidden Layers
- Form der Aktivierungsfunktionen
- Breite der Verteilung für die Anfangsgewichte
- Lernrate η
- Überrelaxationsrate α (Momentum Term)
- Lernmodus (“Example-by-Example” oder “Batch”)
- Anzahl Iterationen
- “Seed” für den Zufallszahlen-Generator

Führen Sie eine Reihe von Versuchen durch, und skizzieren Sie dann für zwei oder drei “interessante” Lösungen (erfolgreiche oder nicht erfolgreiche) den Verlauf der Netzwerkfunktion $O(I_1, I_2)$ im Bereich $-2 < I_1, I_2 < +2$ mit Hilfe von Höhenlinien.

Verwenden Sie insbesondere Multilayer-Perceptrons mit nur einem Hidden Layer und mit 4 Hidden Units. (Versuchen Sie es auch mit nur 3 Hidden Units!)

Kurzanleitung für den Backpropagation-Simulator "neuro"

Auf der Vorlesungs-Homepage finden Sie die folgenden Files:

go.bat	**config_xor.txt**
neuro.exe	**examples_xor.txt**

Diese Files können Sie in einen Folder auf Ihrem Computer herunterladen (Windows-Betriebssystem!) und dann wie folgt benutzen. (Die Files **weights_xor.txt** und **output_xor.txt** werden beim Lern- bzw. Auswerteprozess automatisch erzeugt.)

Initialisierung der Applikation:

Ein Doppelclick auf **go.bat** öffnet Ihnen ein Command-Window, in dem Sie die Befehle für das Starten eines Lernprozesses oder einer Output-Auswertung eingeben können.
*[Falls **go.bat** nicht funktioniert, können Sie auch einfach einen Command Prompt (MS-DOS Prompt) öffnen und in das Directory mit den heruntergeladenen Files wechseln].*

Starten eines Lernprozesses:

Mit dem Befehl.

neuro -L config_name.txt examples_name.txt weights_name.txt

wird ein Netzwerk nach den Spezifikationen im Konfigurations-File "config_name.txt" erstellt und mit den im File "examples_name.txt" aufgelisteten Lernbeispielen trainiert. Die resultierenden Gewichte werden ins Gewichts-File "weights_name.txt" geschrieben.

(Die Gewichte werden bei jedem neuen Versuch überschrieben!)

Starten der Output-Auswertung:

Mit dem Befehl.

neuro -X config_name.txt examples_name.txt weights_name.txt >> output_name.txt

wird ein Netzwerk nach den Spezifikationen im Konfigurations-File "config_name.txt" erstellt, und die (trainierten) Gewichte werden gemäss den in "weights_name.txt" gespeicherten Werten gesetzt. Dann werden die im File "examples_name.txt" angegebenen Trainings-Inputs verarbeitet, und die entsprechenden Output-Werte werden in das File "output_name.txt" geschrieben.

(Die alten Output-Werte werden bei einem neuen Versuch nicht überschrieben!)

Die in den verschiedenen Files benutzten Begriffe werden im Folgenden anhand des XOR-Problems erklärt. Die entsprechenden txt-Files sind auf der Vorlesungs-Homepage abgelegt.

Um die Aufgabe dieser Übung zu lösen, können Sie dann einfach die Files "config_xor.txt" und "examples_xor.txt" entsprechend abändern.

Lernbeispiel-File "examples_xor.txt":

```
# xor examples
# I1  I2   D

 1   1   1
 1  -1  -1
-1   1  -1
-1  -1   1
```

Konfigurations-File "config_xor.txt":

```
#xor config-file

inputs=2                    (Anzahl Inputs)
h1=2                        (Anzahl Neuronen im Hidden Layer 1)
h2=0                        (Anzahl Neuronen im Hidden Layer 2)
outputs=1                   (Anzahl Outputs)

output_function=1           (Aktivierungsfunktion der Output-Neuronen)
hidden_function=1           (Aktivierungsfunktion der Hidden Neuronen
A=1                             (siehe unten)
B=1                             (siehe unten)


eta=0.2                     (Lernrate)
alpha=0                     (Überrelaxations-Rate)

w0=1                        (Anfangsgewichte zufällig in[-w0, +w0]))

seed=34571                  (Seed für den Zufallszahlen-Generator)

mode=0;                     (Lernmodus:  0: "Example-by-Example"
                                         1: "Batch")
iterations=10000;           (Anzahl Iterationen)
```

Für die Aktivierungsfunktionen stehen drei Versionen zur Verfügung:

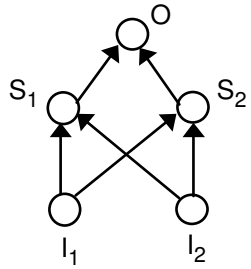
0: $f_0(x) = \frac{1}{1+\exp(-x)}$

1: $f_1(x) = A\ \frac{1-\exp(-B * x)}{1+\exp(-B * x)}$

2: $f_2(x) = x$

(Für die Versionen 0 und 2 haben die im Konfigurations-File angegebenen Werte für A und B keine Bedeutung.)

Das in "config_xor.txt" spezifizierte Netzwerk sieht also folgendermassen aus,



$O = f_{out}(V_0 + V_1 S_1 + V_2 S_2)$

$S_k = f_{hid}(W_{k0} + W_{k1} I_1 + W_{k2} I_2)$

und die trainierten Gewichte werden in "weights_xor.txt" wie folgt dargestellt:

Beispiel für Gewichts-File:“weights_xor.txt”:

IH1 = [
[-3.271516e+000 2.920122e+000] → $[W_{11}\ W_{21}]$
[3.284298e+000 -2.908411e+000] → $[W_{12}\ W_{22}]$
];

H1O = [
[4.826905e+000] → $[V_1]$
[4.882618e+000] → $[V_2]$
];

H1_levels = [
3.000943e+000 → $[W_{10}]$
2.544734e+000 → $[W_{20}]$
];

O_levels = [
-4.260368e+000 → $[V_0]$
];

Beispiel für Output-File: “output_xor.txt”:

O = [
[0.973073] *(Netzwerk-Output für Lernbeispiel 1)*
[-0.961794] *(Netzwerk-Output für Lernbeispiel 2)*
[-0.962731] *(Netzwerk-Output für Lernbeispiel 3)*
[0.972358] *(Netzwerk-Output für Lernbeispiel 4)*
];
