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

Sommersemester 2009

Übung 3

Ausgabe am 20. März 2009
Abgabe am 3. April 2009

Problemstellung

Eine Bank will ihre Entscheide über Hypothekarkreditanträge mit Hilfe eines neuronalen Netzwerks automatisieren. Für das Training der Netzwerke wird ein Datensatz von fünfzig Anträgen zur Verfügung gestellt, über die in den letzten Monaten entschieden wurde [**siehe Tabelle**].

Zur Auswahl stehen ein Winner-Take-All Netzwerk mit 10 Output-Neuronen und ein einfaches Perceptron. Ihre Aufgabe besteht nun darin, durch entsprechende Tests herauszufinden, welche der beiden Netzwerk-Typen sich für dieses Problem besser eignet.

Hinweise zum Vorgehen

1) Vorverarbeitung der Inputdaten

Wählen und skalieren Sie die 4 Inputs (siehe Tabelle) für die neuronalen Netzwerke wie folgt:

$$I_1 = \begin{cases} 0 & \text{falls alleinstehend} \\ 1 & \text{falls verheiratet} \end{cases}$$

I_2 = Anzahl Kinder / 4

I_3 = Jahreseinkommen / 100'000

I_4 = Gewünschte Hypothek / 1'000'000

2) Winner-Take-All Netzwerk

Wählen Sie ein Winner-Take-All Netzwerk mit 4 Inputs (I_1 bis I_4) und mit 10 Output-Neuronen. Fünf dieser Neuronen werden einem postiven Entscheid zugeordnet,

$C(1) = C(2) = C(3) = C(4) = C(5) = +1$,

und die restlichen fünf einem negativen Entscheid,

$C(6) = C(7) = C(8) = C(9) = C(10) = -1$.

Trainieren Sie dieses Netzwerk mit Hilfe des “Supervised LVQ” – Lernverfahrens:

(i) Zufällige Wahl der Anfangsgewichte $W_{ij}(0)$ im Bereich der Inputdaten I_j
($i = 1, \ldots, 10$; $j = 1, \ldots, 4$).

(ii) Iterationsschritt t:

- Zufällige Auswahl eines der 50 Lernbeispiele: $I_1(t)$, …, $I_4(t)$.
- Bestimmung des Winner-Neurons i*.
- Änderung der Gewichte:

$$W_{i^*j}(t+1) = \begin{cases} W_{i^*j}(t) + \eta(t)\left(I_j(t) - W_{i^*j}(t)\right) & \text{falls } C(i^*) = E(t) \\ W_{i^*j}(t) - \eta(t)\left(I_j(t) - W_{i^*j}(t)\right) & \text{falls } C(i^*) \neq E(t) \end{cases}$$

$$W_{ij}(t+1) = W_{ij}(t) \quad \text{falls } i \neq i^*$$

wobei

E(t) = Entscheid beim Lernbeispiel $\underline{I}(t)$

$\eta(t) = \eta_0\left(1 - \frac{t}{n}\right)$ [z.B. η_0 = 0.1, …., 0.5]

n = Anzahl Iterationen [z.B. n = 10'000, …, 20'000].

Trainieren Sie mehrere solcher Netzwerke (mit verschiedenen Werten für n und η_0) und bestimmen Sie jeweils die Anzahl der korrekt klassifizierten Trainingsbeispiele.

Welche Klassifizierungsrate erreichen Sie mit Ihrem besten Winner-Take-All Netzwerk?

3) <u>Perceptron</u>

Wählen Sie ein Perceptron mit 5 Inputs,

$I_0 \equiv 1,\ I_1,\ I_2,\ I_3,\ I_4$,

und mit einem Output:

$$O = \text{sign}\left(\sum_{j=0}^{4} W_j I_j\right).$$

Trainieren Sie dieses Netzwerk mit dem einfachen Perceptron-Lernverfahren:

(i) Zufällige Wahl der Anfangsgewichte $W_j(0)$, j = 0, 1, …, 4, im Bereich [–1, +1].

(ii) Iterative Änderung der Gewichte:

$W_j(t+1) = W_j(t) + \eta\left[E(t) - O(t)\right] I_j(t)$ [z.B. η = 0.1, …, 0.5] ,

wobei

$\{I_0 \equiv 1,\ I_1(t),\ I_2(t),\ I_3(t),\ I_4(t)\}$ = zufällig ausgewähltes Lernbeispiel

E(t) = entsprechender Entscheid.

Sie werden feststellen, dass das Lernverfahren nicht konvergiert (das Problem ist offenbar nicht linear separabel).

Testen Sie deshalb von Zeit zu Zeit die Klassifizierungsrate, und speichern Sie jeweils die Gewichte des bis anhin besten Netzwerks ["Pocket-Algorithmus"].

Welche Klassifizierungsrate erreichen Sie mit diesem Netzwerk-Typ?

Tabelle der Trainingsdaten:

Antrags-Nr.	Zivilstand	# Kinder	Einkommen/Jahr	Gewünschte Hypothek	Entscheid
1	verh.	4	83'000	400'000	–
2	alleinst.	1	77'000	317'000	+
3	verh.	0	84'000	498'000	–
4	verh.	4	128'000	274'000	+
5	verh.	0	80'000	508'000	+
6	verh.	1	116'000	456'000	+
7	alleinst.	1	92'000	489'000	+
8	verh.	0	72'000	341'000	+
9	verh.	1	75'000	259'000	+
10	alleinst.	1	122'000	417'000	+
11	alleinst.	0	77'000	484'000	–
12	alleinst.	1	128'000	462'000	+
13	alleinst.	2	63'000	371'000	–
14	alleinst.	0	53'000	503'000	–
15	alleinst.	0	90'000	427'000	+
16	alleinst.	1	79'000	519'000	–
17	alleinst.	1	50'000	461'000	–
18	verh.	3	84'000	388'000	+
19	verh.	2	95'000	306'000	+
20	verh.	0	114'000	559'000	+
21	verh.	0	86'000	545'000	–
22	verh.	1	56'000	427'000	–
23	verh.	3	87'000	454'000	+
24	verh.	2	74'000	441'000	–
25	verh.	4	100'000	383'000	+
26	alleinst.	0	73'000	353'000	–
27	verh.	1	50'000	275'000	–
28	verh.	2	59'000	395'000	–
29	verh.	2	93'000	302'000	+
30	alleinst.	0	89'000	513'000	–
31	verh.	3	53'000	504'000	–
32	alleinst.	4	98'000	392'000	+
33	verh.	1	107'000	324'000	+
34	verh.	4	105'000	516'000	+
35	verh.	4	121'000	326'000	+
36	verh.	2	72'000	271'000	+
37	alleinst.	2	120'000	529'000	+
38	verh.	0	60'000	449'000	–
39	verh.	1	80'000	417'000	–
40	verh.	1	76'000	277'000	+
41	alleinst.	0	113'000	536'000	+
42	alleinst.	3	83'000	563'000	–
43	alleinst.	0	75'000	421'000	+
44	verh.	2	80'000	451'000	–
45	verh.	1	74'000	330'000	+
46	verh.	1	79'000	435'000	–
47	verh.	1	101'000	610'000	+
48	verh.	2	60'000	279'000	–
49	alleinst.	2	67'000	306'000	–
50	alleinst.	2	93'000	329'000	+

Anhang: Programm-Beispiele (Matlab)

Winner-Take-All Netzwerk:

```
load hypdat_2009.txt;                    [Lernbeispiele laden]
ziv = hypdat_2009(1:50,1);
nok = hypdat_2009(1:50,2)/4;
jek = hypdat_2009(1:50,3)/100000;
hyp = hypdat_2009(1:50,4)/1000000;
od = hypdat_2009(1:50,5);

nt = 20000;                              [Initialisierung]
e0 = 0.1;
x = zeros(4,1);
d = zeros(1,10);
class = ones(1,10);
class(1,6:10) = -ones(1,5);
ww = rand(4,10);
ww(1,:) = 0+1*ww(1,:);
ww(2,:) = 0.5+0.5*ww(2,:);
ww(3,:) = 0.50+0.78*ww(3,:);
ww(4,:) = 0.26+0.30*ww(4,:);
rand('state',sum(100*clock));

for ni=1:nt                              [LVQ-Lernverfahren]
   e1 = e0*(1-ni/nt);
   cr = rand(1,1);
   ir = floor(50*cr)+1;
   x(1) = ziv(ir);
   x(2) = nok(ir);
   x(3) = jek(ir);
   x(4) = hyp(ir);
   for j=1:10
      d(j) = 0;
      for i=1:4
         d(j) = d(j)+(x(i)-ww(i,j))^2;
      end
   end
   [dm,jx] = sort(d);
   jm = jx(1);
   for i=1:4
      eps = e1;
      if class(jm)*od(ir)<0
         eps = -e1;
      end
      ww(i,jm) = ww(i,jm)+eps*(x(i)-ww(i,jm));
   end
end

anzp = zeros(10,1);          [Klassifizierung der Lernbeispiele]
anzm = zeros(10,1);
for k=1:50
   x(1) = ziv(k);
   x(2) = nok(k);
   x(3) = jek(k);
   x(4) = hyp(k);
   for j=1:10
      d(j) = 0;
      for i=1:4
         d(j) = d(j)+(x(i)-ww(i,j))^2;
      end
   end
   [dm,jx] = sort(d);
   jm = jx(1);
   anzp(jm) = anzp(jm)+(1+od(k))/2;
   anzm(jm) = anzm(jm)+(1-od(k))/2;
end
```

```
wta_err = 0;                          [Berechnung der Anzahl Fehler]
for in=1:5
   wta_err = wta_err +anzm(in);
end
for in=6:10
   wta_err = wta_err +anzp(in);
end
wta_rate = 1- wta_err /50;            [Klassifizierungsrate]
```

[Anzahl Fehler (wta_err), Klassifizierungsrate (wta_rate) und ev. Gewichte, etc. ausdrucken!]

Perceptron

```
load hypdat_2009.txt;                 [Lernbeispiele laden]
i1 = hypdat_2009(1:50,1);
i2 = hypdat_2009(1:50,2)/4;
i3 = hypdat_2009(1:50,3)/100000;
i4 = hypdat_2009(1:50,4)/1000000;
od = hypdat_2009(1:50,5);

rand('state',sum(100*clock));         [Initialisierung]
eta = 0.3;
nr = 200;
kr = 200;
on = zeros(50,1);
wi = 2*rand(5,1)-ones(5,1);

for n=1:nr

   for k=1:kr                         [Perceptron-Lernverfahren]
      cr = rand(1,1);
      is = floor(50*cr)+1;
      on(is) =
      wi(1)*i1(is)+wi(2)*i2(is)+wi(3)*i3(is)+wi(4)*i4(is)+wi(5);
      on(is) = sign(on(is));
      if on(is)==0
         on(is) = 1;
      end
      dod = od(is)-on(is);
      wi(1) = wi(1)+eta*dod*i1(is);
      wi(2) = wi(2)+eta*dod*i2(is);
      wi(3) = wi(3)+eta*dod*i3(is);
      wi(4) = wi(4)+eta*dod*i4(is);
      wi(5) = wi(5)+eta*dod;
   end

   p_err = 0;                         [Berechnung der Anzahl Fehler]
   for j=1:50
      on(j) = wi(1)*i1(j)+wi(2)*i2(j)+wi(3)*i3(j)+wi(4)*i4(j)+wi(5);
      on(j) = sign(on(j));
      if on(j)==0
         on(j) = 1;
      end
      p_err = p_err+0.25*(od(j)-on(j))^2;
   end
p_rate = 1- p_err /50;                [Klassifizierungsrate]
```

[Anzahl Fehler (p_err), Klassifizierungsrate (p_rate) und ev. Gewichte, etc. ausdrucken!]

```
end
```
