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 6

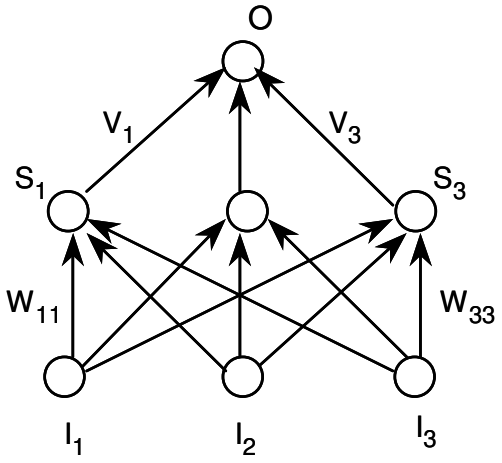
Ausgabe am 8. Mai 2009
Abgabe am 15. Mai 2009

Das 3-Input Parity Problem

Die Parität P von 3 binären Grössen I_1, I_2, I_3 ($I_i = \pm 1$) ist wie folgt definiert:

$$P = I_1 * I_2 * I_3 .$$

Versuchen Sie, diesen Zusammenhang mit einem Multilayer-Perceptron mit drei Hidden Units zu realisieren:



$$O = \text{sign}\ (V_0 + \sum_{i=1}^{3} V_i S_i)$$

$$S_i = \text{sign}\ (W_{i0} + \sum_{j=1}^{3} W_{ij} I_j)$$

Diese Netzwerk-Architektur hat 16 Gewichte (inklusive Schwellen V_0, W_{10}, W_{20}, W_{30}).

Ihre Aufgabe besteht also darin, die 16 Gewichte so zu bestimmen, dass der Output O des Netzwerks die Parität P für alle 8 Input-Kombinationen (I_1, I_2, I_3) richtig anzeigt.

Vorgehen

Verwenden Sie ein stochastisches "**Iterative Improvement**"-Verfahren:

1) Die zu minimierende Zielfunktion ist E(W):
   E(W) = Anzahl der falschen Outputwerte für die 8 möglichen Input-Kombinationen.
2) Wählen Sie die Startwerte für die 16 Gewichte W_{ij} zufällig im Bereich (–1,+1).

3) Bestimmen Sie eine zufällige Gewichtsänderung $\underline{W} \rightarrow \underline{W}'$ wie folgt:
   $W'_{ij} = W_{ij} + \Delta W \cdot r_{ij}$,
   wobei die 16 r_{ij}-Werte zufällig zwischen –1 und +1 gewählt werden.

4) Die Änderung der Gewichte wird akzeptiert, falls $E(\underline{W}') \leq E(\underline{W})$, andernfalls bleiben die W_{ij} unverändert.

5) Beendigen Sie den Versuch, wenn $E(\underline{W}) = 0$ ist, oder wenn die vorgegebene maximale Anzahl von Iterationen (z.B. $n_{max} = 500$) erreicht ist. Sonst zurück zu 3).

Aufgaben

- Wählen Sie $\Delta W = 1$ und führen Sie mehrere Versuche (z.B. 100 oder 500) mit verschiedenen, zufällig gewählten Anfangsgewichten durch, und bestimmen Sie dann die Statistik der gefundenen $E(\underline{W})$ – Werte nach n_{max} Iterationen:

  Anzahl Netzwerke mit $E(\underline{W}) =$ 0 Fehlern
  " " " " 1 Fehler
  " " " " 2 Fehlern
  etc.

- Bestimmen Sie den Anteil perfekter Lösungen (Lösungen mit Null Fehlern) als Funktion von n_{max} (z.B. $n_{max} = 50, 100, 150, \ldots, 1000$).

- Für welches n_{max} benötigt man im Mittel am wenigsten Iterationen, um mit einer Wahrscheinlichkeit von 99% eine perfekte Lösung zu finden?

  Hinweis: Sei p der Anteil perfekter Lösungen bei festem n_{max}. Die Wahrscheinlichkeit, nach n Versuchen keine perfekte Lösung gefunden zu haben, ist dann $q = (1-p)^n$. Um in 99% der Fälle eine perfekte Lösung zu finden, muss gelten: $q \leq 0.01$. Daraus lässt sich die notwendige Anzahl Versuche n berechnen, und die totale Anzahl Iterationen ist dann gleich $n_{max} * n$.

Programm-Beispiele (Matlab)

Hauptprogramm "parity_3":

```
dw = 1.0;
nmax = 500;
nst = 100;

inp = [1,1,1,1,-1,-1,-1,-1; ...          [Parity-Lernbeispiele]
       1,1,-1,-1,1,1,-1,-1; ...
       1,-1,1,-1,1,-1,1,-1];
od = [1,-1,-1,1,-1,1,1,-1];

nen = zeros(9,1);                        [Initialisierung der Statistik]

for ns=1:nst                             [Loop über verschiedene Versuche]
   w1 = 2*rand(1,4)-ones(1,4);
   w2 = 2*rand(1,4)-ones(1,4);
   w3 = 2*rand(1,4)-ones(1,4);
   v = 2*rand(1,4)-ones(1,4);
```

```
   for nit=1:nmax                          [Loop für "Iterative Improvement"]
      w1s = w1;
      w2s = w2;
      w3s = w3;
      vs = v;
      parity3_eval
      nf = nfs;
      w1s = w1 + dw*(2*rand(1,4)-ones(1,4));
      w2s = w2 + dw*(2*rand(1,4)-ones(1,4));
      w3s = w3 + dw*(2*rand(1,4)-ones(1,4));
      vs = v + dw*(2*rand(1,4)-ones(1,4));
      parity3_eval
      nfm = nf;
      if nfs <= nf
         w1 = w1s;
         w2 = w2s;
         w3 = w3s;
         v = vs;
         nfm = nfs;
      end
   end
   nen(nfm+1) = nen(nfm+1)+1;          [Aufdatieren der Statistik]
end
......                                 [Statistik ausdrucken, etc.]
```

Statistik-Auswertung "parity3_eval":

```
nfs = 0;                               [Initialisierung Fehleranzahl]
for k=1:8
   it = [1;inp(:,k)];                  [Berechnung des Netzwerk-Outputs]
   s1 = sign(w1s*it);
   s2 = sign(w2s*it);
   s3 = sign(w3s*it);
   on = sign(vs(1)+vs(2)*s1+vs(3)*s2+vs(4)*s3);
   if on ~= od(k)                      [Anzahl Fehler aufdatieren]
      nfs = nfs+1;
   end
end
```
