7 Neuronale Netze am Beispiel des Hopfield Modells

In diesem Abschnitt sollen einige Anfangsgründe diskutiert werden von Systemen, deren Aufbau und auch Fähigkeiten in primitiver Weise dem menschlichen Gehirn ähneln. Man nennt sie daher neuronale Netze. Der Aspekt, um den es hier geht, ist die assoziative Speicherung, wobei die Adressierung über Teile des Speicherinhalts erfolgt. Ein Beispiel wäre das Erkennen eines bekannten Gesichtes aus einem verschwommenen Teilbild, oder die Erinnerung an ein Lied aus nur einem nicht ganz richtig gesummten Refrain. Das hier studierte Hopfield-Modell liefert eine Karikatur solcher Fähigkeiten. Zur Vorbereitung dieses Kapitels wurde [12] herangezogen.

7.1 Neuronen und Synapsen

Das Hopfield Netz besteht aus einer großen Zahl N von Neuronen S_i, die zweier Zustände fähig sind. In biologischen Netzen spricht man von feuernden und nicht feuernden bzw. aktivierten Neuronen, was wir in unserem Model durch $S_i = +1$ (feuernd) und $S_i = -1$ (nicht feuernd) darstellen. Dies ist selbstverständlich eine Konvention, $0, 1$ wäre genauso gut. Das Neuron S_i kann Neuron S_j mit variabler Stärke beeinflussen über Verbindungen oder Synapsen. Sie werden im Modell durch reelle Zahlen w_{ij} gegeben, die eine $N \times N$ Matrix bilden. Das Netz evolviert nun im einfachsten Fall in diskreten Zeitschritten von t nach $t+1$ nach dem Gesetz (McCulloch-Pitts Gleichung)

$$S_i(t+1) = \mathrm{sgn}(\sum_{\mathrm{j}} \mathrm{w_{ij}S_j(t)} - \Theta_{\mathrm{i}}). \tag{7.1}$$

Hier sind die Synapsen w_{ij} als gegeben betrachtet, und sgn ist die Vorzeichen (signum) Funktion. S_i feuert also, wenn es genügend von anderen Neuronen über Synapsen angeregt wird relativ zu Schwellenwerten Θ_i. Die w_{ij} können positiv (exzitatorisch) oder negativ (inhibitorisch) sein. In diesen Größen residiert der Speicherinhalt, der bestimmt, wie das Netz auf vorgelegte Anfangsmuster $S_i(t_0)$ "reagiert". Eine wichtige Frage ist natürlich, wie die w_{ij} durch "Trainieren" auf das gewüschte Verhalten, z. B. anhand von Beispielen, gesetzt werden. Man spricht hier von Lernalgorithmen. Verallgemeinerungen von (7.1) bestehen aus anderen Aktivierungsfunktionen (stochastisch, "fuzzy") und der Wahl zwischen synchronem und asynchronem Update (vgl.

Jacobi und Gauss-Seidel Relaxation in CP I). Letzteres ist für das Gehirn wohl plausibler und algorithmisch einfacher.

Beim Hopfield Modell wird (7.1) als ein Schritt zur iterativen Minimierung der Energiefunktion

$$H = -\frac{1}{2}\sum_{ij} w_{ij} S_i S_j + \sum_i \Theta_i S_i \tag{7.2}$$

angesehen. Hier sind offensichtlich nur symmetrische w_{ij} sinnvoll, was eine nichttriviale Spezialisierung von (7.1) darstellt. Setzt man nun jeweils ein einzelnes Neuron S_i — bei Festhalten der Übrigen — auf denjenigen seiner beiden Werte, für den H kleiner ist, so ergibt sich (7.1) im *asynchronen* Modus mit *Einschränkung der Summe auf der rechten Seite* auf $j \neq i$. Da H dabei laufend fällt oder gleichbleibt, läuft man in einen Fixpunkt[21] am (i. a. nur lokalen) Minimum hinein. Dieser Konvergenzbeweis gälte nicht im synchronen Modus!

Im Folgenden wollen wir uns auf das Hopfield Modell mit $\Theta_i = 0$ beschränken.

7.2 Speichern in Synapsen

Eine übliche Anwendung des Hopfield Modells besteht in der assoziativen Speicherung von p Mustern $\xi_i^{(\mu)} = \pm 1, \mu = 1 \ldots p$. Hier bildet man die Neuronen auf bestimmte Bildelementen (Pixel), z. B. in einer Ebene, ab. Wenn nun ein Aktivierungszustand $S_i(t)$ "in der Nähe" von $\xi_i^{(\mu)}$ liegt — etwa einer verrauschten Version des Bildes entspricht —, dann soll er per McCulloch-Pitts Gleichung sich dorthin als dem "nächstliegenden" Minimum entwickeln[22] und im Idealfall bei $S_i = \xi_i^{(\mu)}$ als Fixpunkt verharren. Dann besitzt der Konfigurationsraum zu jedem Fixpunkt-Muster einen assoziierten Bereich im Zustandsraum, sein "basin of attraction".

Für nur ein einziges Bild ist eine geeignete Wahl der w_{ij} trivial,

$$w_{ij} = \frac{1}{N}\xi_i \xi_j. \tag{7.3}$$

[21] Der Spezialfall, daß beide Zustände exakt entartet sind, ist für reelle w_{ij} vernachlässigbar.

[22] Man kann den Abstandsbegriff präzisieren durch den Hamming Abstand = Zahl der differierenden Bits oder Pixel.

Dann gilt

$$H = -\frac{1}{2N}\left(\sum_i \xi_i S_i\right)^2 \tag{7.4}$$

mit absoluten Minimum $H = -N/2$ bei $S_i = \xi_i$. Der gleiche Wert wird angenommen bei $S_i = -\xi_i$, daher müssen mehr als die Hälfte der Bits am Anfang stimmen, und der Attraktionsbereich ist der halbe Konfigurationsraum. Geometrisch betrachtet ist w_{ij} als $N \times N$ Matrix ein Projektor auf die Richtung ξ und die damit gebildete quadratische Form H für S mit fester Länge $S^2 = N$ minimal, wenn S in Richtung ξ zeigt (parallel oder antiparallel).

Eine offensichtliche Verallgemeinerung auf mehrere Muster ist die Hebb'sche Regel

$$w_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^{(\mu)} \xi_j^{(\mu)}. \tag{7.5}$$

Hier werden entsprechende Projektoren addiert. Für wenige Bilder werden, wenn S_i nahezu parallel zu einem $\xi_i^{(\mu)}$ ist, die anderen Skalarprodukte typischerweise klein sein, und den Konvergenzprozeß zum betreffenden lokalen Minimum kaum stören. Dies gilt nicht mehr, wenn es voll wird im Speicher. Für $p = \mathrm{O}(N)$ kann sich w_{ij} einem Vielfachen der Einheitsmatrix nähern (exakt der Fall für N orthogonale Bilder), was dem Gedächtnisverlust durch zuviel Speichern entspricht, da dann im Extremfall jeder Zustand Fixpunkt ist. Die Kapazität ist also begrenzt, wie wir unten genauer diskutieren.

Die Diagonalelemente $w_{ii} = p/N$ tragen wegen $S_i^2 = 1$ eine für die Minimierung irrelevante Konstante zu H bei. Es ist praktisch, dort (7.5) zu $w_{ii} = 0$ zu modifizieren, so daß in (7.1) die volle Summe (Matrixmultiplikation) stehen darf.

7.3 Speicherkapazität

Zur Speicherkapazität des Hopfield Netzes wollen wir hier einige heuristische Überlegungen anstellen und dann einige Resultate angeben, deren Ableitung hier zu umfänglich wäre.

Wir wollen nun die Stabilität eines Bildes diskutieren. Wir nehmen einen Anfangszustand der Neuronen, der exakt dem ersten Bild entspricht, $S_i(t) =$

$\xi_i^{(1)}$. Dann ist[23]

$$S_i' \equiv S_i(t+1) = \mathrm{sgn}\left(\frac{1}{\mathrm{N}}\sum_{\nu,\mathrm{j}\neq\mathrm{i}} \xi_{\mathrm{i}}^{(\nu)}\xi_{\mathrm{j}}^{(\nu)}\xi_{\mathrm{j}}^{(1)}\right) \simeq \mathrm{sgn}\left(\xi_{\mathrm{i}}^{(1)}(1-\mathrm{C_i})\right), \qquad (7.6)$$

wobei

$$C_i = -\frac{1}{N}\sum_{\nu>1,j\neq i} \xi_i^{(1)}\xi_i^{(\nu)}\xi_j^{(\nu)}\xi_j^{(1)} \qquad (7.7)$$

die "Störung" durch die übrigen $p-1$ gespeicherten Bilder beinhaltet. Offensichtlich werden Bits von $\xi_i^{(1)}$ instabil, wenn $C_i > 1$ wird.

Für zufällige unkorrelierte Muster wollen wir die Wahrscheinlichkeit dafür abschätzen. Wir betrachten C_i als eine Summe von pN zufälligen Größen ϵ_a mit jeweils gleich wahrscheinlichen Werten ± 1 geteilt durch N. Die Summe solcher Zufallsvariablen ist binomial verteilt, und diese Verteilung geht für viele Beiträge in die Normalverteilung über. Letztere ist festgelegt durch Mittelwert und Varianz,

$$\langle C_i\rangle = 0, \qquad (7.8)$$

$$\langle C_i^2\rangle = \frac{1}{N^2}\left\langle \sum_{a,b=1}^{pN}\epsilon_a\epsilon_b\right\rangle = \frac{1}{N^2}\sum_{a,b=1}^{pN}\delta_{ab} = \frac{p}{N} = \sigma^2. \qquad (7.9)$$

Damit ist die Wahrscheinlichkeit, daß $C_i > 1$ gilt, gegeben durch

$$P_{\mathrm{Fehler}} = \frac{1}{\sqrt{2\pi}\sigma}\int_1^\infty \exp(-x^2/2\sigma^2)\mathrm{d}x = \frac{1}{2}\,\mathrm{erfc}(\sqrt{N/2p}). \qquad (7.10)$$

Hier ist erfc die (komplementäre) Fehlerfunktion. Sie ist in MATLAB vorhanden, und Werte sind

```
>> a=[0.105 0.138 0.185 0.37 0.61]'; % Werte p/N
>> P=0.5*erfc(sqrt(0.5./a));         % P_Fehler
>> [a P]

ans =

    0.1050    0.0010
    0.1380    0.0036
```

[23] Wir vernachlässigen Terme der relativen Ordnung $1/N$.

```
0.1850    0.0100
0.3700    0.0501
0.6100    0.1002
```

Die linke Spalte sind p/N Verhältnisse, die rechte der Anteil fehlerhafter Bits.

Diese Betrachtung betraf instabile Bits nach einem Schritt beginnend beim perfekten Muster. Sie sagt nichts darüber aus, wie es weitergeht, ob z. B. eine Lawine von geflippten Bits einsetzt. Wesentlich kompliziertere Rechnungen [12] zeigen, daß die Grenze dafür bei $p/N \simeq 0.138$ liegt. Genauer gesagt, handelt es sich hier für $N \to \infty$ um einen Phasenübergang erster Ordnung. Bis dahin funktioniert das Gedächtnis gut, danach nicht mehr. Gerade bei diesem Grenzwert sind nach einem Schritt im Mittel 0.36 % Bits falsch. Dies steigt dann auf 1.6 %, wo dann ein Fixpunkt nahe beim Muster erreicht wird. Mit kleineren Füllfaktoren (?, load parameter) funktioniert alles fehlerärmer.

7.4 Simulation

In diesem Abschnitt sollen zur Erleichterung der Übungsaufgaben einige charakteristische Statements in MATLAB zur Realisierung des Hopfield Modells angegeben werden. Diese Umsetzung ist natürlich nicht die einzig mögliche. Zunächst kann man wie folgt Bilder erzeugen und die Hebb Matrix konstruieren:

```
n=100;          % # Neuronen
p=20;           % # Bilder

xi = sign(rand(n,p)-0.5); % Bilder

% Synapsen setzen nach Hebb:
w=zeros(n,n);
for i=1:p
  w=w+xi(:,i)*xi(:,i)';
end
for i=1:n, w(i,i)=0; end  % Diagonale zu Null
w=w/n;
```

Die folgenden Zeilen zeigen Update Schritte asynchron und (auskommentiert) synchron, sowie die Berechnung des Hamming Abstandes:

```
    s=xi(:,1);
%   sp=sign(w*s);                                  % synchron
    sp=s; for j=1:n, sp(j)=sign(w(j,:)*sp);end % asynchron
    ham=sum(s ~= sp);                              % Hamming Abstand
```

Wer möchte, kann sich auch ein Bild ansehen, obwohl dies bei Zufallsmustern nicht sehr erleuchtend ist:

```
b=10;    % Kantenlaenge Bild
n=b^2;   % # Neuronen
xi = sign(rand(n,1)-0.5); % 1 Bild
pic=zeros(b,b);
pic(:)= (xi > 0);  % xi -> Spalten des Bildes, -/+ in 0/1 verwandelt
spy(pic);          % plotted eine Matrix
```

7.5 Nebenminima

Außer den erwünschten Mustern, die bei der Hebb Regel in die w_{ij} eingehen, kann es andere unerwünschte Fixpunkte geben (spurious states). Ein Beipiel sind Überlagerungszustände (mixed states) aus einer ungeraden Zahl von Mustern. Wir betrachten als Beispiel drei Zustände,

$$\bar{\xi}_i = \mathrm{sgn}(\xi_\mathrm{i}^{(1)} + \xi_\mathrm{i}^{(2)} + \xi_\mathrm{i}^{(3)}). \tag{7.11}$$

Der Zustand $S_i = \bar{\xi}_i$ entwickelt sich zu $S_i' = \mathrm{sgn}(\bar{\mathrm{h}}_\mathrm{i})$ mit

$$\bar{h}_i = \frac{1}{N} \sum_{\nu, j \neq i} \xi_i^{(\nu)} \xi_j^{(\nu)} \bar{\xi}_j. \tag{7.12}$$

Für nicht zu viele Muster kann die Summe näherungsweise auf $\nu = 1, 2, 3$ eingeschränkt werden, da die übrigen Zustände dann typischerweise kleine Skalarprodukte mit den betrachteten haben. Nun sind die Produkte (z. B. $\nu = 1$)

$$\xi_j^{(1)} \bar{\xi}_j = \xi_j^{(1)} \, \mathrm{sgn}(\xi_\mathrm{j}^{(1)} + \xi_\mathrm{j}^{(2)} + \xi_\mathrm{j}^{(3)})$$

in 3 von 4 möglichen Konfigurationen von $\xi_j^{(2)}, \xi_j^{(3)}$ (außer $--$) positiv, und bei Summation über j erhalten wir im Mittel

$$\left\langle \sum_j \xi_j^{(1)} \bar{\xi}_j \right\rangle = \frac{N}{2} \tag{7.13}$$

und somit

$$\langle \bar{h}_i \rangle = \frac{1}{2}(\xi_i^{(1)} + \xi_i^{(2)} + \xi_i^{(3)}), \tag{7.14}$$

was die Stabilität von $\bar{\xi}_i$ zeigt. Das gleiche Resultat folgt für andere Wahlen der drei Vorzeichen auf der rechten Seite von (7.11). Darüber hinaus gibt es weitere unerwünschte Minima, die sogenannten Spinglas Zustände, die mit den gespeicherten Mustern nicht direkt korreliert sind. Trotz dieser Zustände bleibt das Hopfield Netz nützlich als Assoziativspeicher (content addressed), da die Attraktionsbereiche der gespeicherten Muster typischerweise größer sind.

7.6 Endliche Temperatur

Oft ist es nützlich, statt der deterministischen McCulloch-Pitts Gleichung ein stochastisches ("fuzzy") Element (Rauschen) einzuführen, so daß $S_i(t+1)$ sich manchmal auch gegen das Vorzeichen der rechten Seite von (7.1) stellt. Genau das ergibt sich, wenn wir mit der Energie H des Hopfield Modells statistische Mechanik treiben.

Wir betrachten die Zustandssumme

$$Z = \sum_{\{S_i\}} \exp(-\beta H) \tag{7.15}$$

mit der inversen Temperatur β. Nun implementieren wir eine Monte Carlo Simulation mit diesem Boltzmann Gewicht durch lokale Wärmebad Schritte für einzelne S_i:

$$P(S_i' = \pm) = \frac{\exp(\pm\beta h_i)}{\exp(\beta h_i) + \exp(-\beta h_i)} \tag{7.16}$$

mit

$$h_i = \sum_{j \neq i} w_{ij} S_j. \tag{7.17}$$

Hier taucht die sogenannte Sigmoid Funktion auf,

$$P(S_i' = +) = 1 - P(S_i' = -) = f_\beta(h_i) = \frac{1}{1 + \exp(-2\beta h_i)}. \tag{7.18}$$

Offenbar entsprechen diese Schritte für $\beta \to \infty$ genau der McCulloch-Pitts Dynamik. Bei endlicher Temperatur können kleinere Nebenminima durch thermische Fluktuationen wieder verlassen werden um den Weg in größere

basins of attraction zu finden. Dort kann man natürlich keine exakte Konvergenz mehr erwarten, da ja weiterhin Fluktuationen möglich bleiben. Die Wahl der Temperatur stellt natürlich eine gewisse Kunst dar. Es gibt dann ein Phasendiagramm des Hopfield Modells in einer Ebene der Parameter Temperatur und Füllfaktor [12]. Der kritische Wert des Letzteren fällt allerdings mit steigender Temperatur.

In der Optimierungstheorie gibt es das eng verwandte Verfahren des "Simulated Annealing". Dort geht es darum, das absolute Minimum einer Funktion vieler Variabler (Kostenfunktion) zu finden. Ein typisches Problem besteht darin, daß man mit deterministischen Verfahren wie "steepest descent" in lokalen Minima hängen bleibt. Ähnlich wie hier geht man zur statistischen Mechanik mit der Kostenfunktion als Energie über, und erhöht nun β während der Iteration nach einem bestimmten Fahrplan. So kann man z. B. versuchen das klassische Optimierungsproblem anzugehen, die Gesamtwegstrecke des "travelling salesman", der eine vorgegebene Zahl von Städten in beliebiger Reihenfolge besuchen muß, zu minimieren.

Literatur

[1] S.E.Koonin und D.C.Meredith, Computational Physics, Addison–Wesley

[2] E. Ott, Chaos in Dynamical Systems, Cambridge University Press 1993

[3] H. G. Schuster, Deterministisches Chaos, VCH Verlagsgesellschaft, Weinheim

[4] H. Goldstein, Klassische Mechanik, AULA-Verlag, Wiesbaden

[5] D. E. Knuth, Seminumerical Algorithms, Vol. 2 of The Art of Computer Programming (Addison Wesley)

[6] Martin Lüscher, A PORTABLE HIGH QUALITY RANDOM NUMBER GENERATOR FOR LATTICE FIELD THEORY SIMULATIONS. Comput.Phys.Commun.79:100-110,1994, hep-lat/9309020

[7] H. Erlenkötter und V. Reher, Programmiersprache C, rororo Grundkurs Computerpraxis (16,80 DM)

[8] Die Programmiersprache C, herausgegeben vom Regionalen Rechenzentrum Niedersachsen (RRZN)

[9] D. Stauffer und A. Aharony, Perkolationstheorie, VCH Verlagsgesellschaft, 1995

[10] W. Nolting, Grundkurs Theoretische Physik Bd. 6: Statistische Physik
J. Schnakenberg, Algorithmen in der Quantentheorie und Statistischen Physik
beide: Verlag Zimmermann-Neufang, Ulmen

[11] U. Wolff, Numerical Simulation in Quantum Field Theory, in Computational Physics, K.H.Hoffmann und M.Schreiber Eds., Springer 1996

[12] J. Hertz, A. Krogh und R.G.Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling und B. P. Flannery, Numerical Recipes, Cambridge University Press
Von diesem nützlichen Buch gibt es verschiedene Ausgaben mit Programmen in Fortran oder C