Capacity of the Hopfield network (HN)

I neurons, N memories

Consider a binary HN, with the Hebbian learning rule.

Let's say the network is set to pattern (memory) $\bar{x}^{(n)}$ which was generated randomly:

$$a_i = \sum_{j \neq i} w_{ij} x_j^{(n)},$$

$$w_{ij} = \underbrace{x_i^{(n)} x_j^{(n)}}_{\substack{\text{signal} \\ \text{(reinforces } \bar{x}^{(n)})}} + \underbrace{\sum_{m \neq n} x_i^{(m)} x_j^{(m)}}_{\text{"noise"}}$$

$\eta = 1$ here

Then $a_i = \sum_{j \neq i} x_i^{(n)} \underbrace{x_j^{(n)} x_j^{(n)}}_{=1} +$

$$+ \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)} = \underbrace{(I-1)\, x_i^{(n)}}_{\text{desired state} \times} +$$

$\underbrace{I-1}_{>0} \Rightarrow$ keeps node i clamped to the correct value $x_i^{(n)}$

$$+ \underbrace{\sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)}}_{\text{sum of } (I-1)(N-1) \text{ independent random variables (by assumption)}}$$

Now, we have, by construction,

$$x_k^{(p)} = \begin{cases} +1, & p=\frac{1}{2} \\ -1, & p=\frac{1}{2} \end{cases}$$

$\forall k, \forall p$

Then $\underbrace{x_i^{(m)} x_j^{(m)} x_j^{(n)}}_{\text{all 3 terms distinct}}$

$(i \neq j, m \neq n)$

8 configurations:

$$\left\{ \begin{array}{rrr} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \\ -1 & -1 & -1 \end{array} \right. \quad p=\frac{1}{8} \text{ each} \quad \Rightarrow \quad \underbrace{\left\{ \begin{array}{r} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{array} \right.}_{\text{product}} \quad \Rightarrow \quad \begin{cases} +1, & p=\frac{1}{2} \\ -1, & p=\frac{1}{2} \end{cases}$$

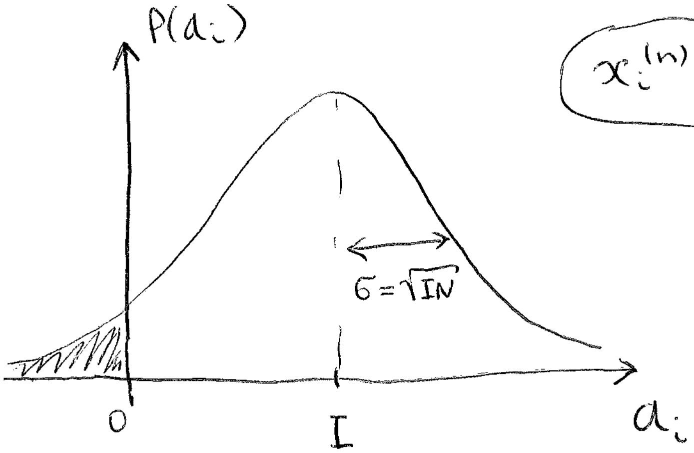
So, $\underbrace{\langle x_i^{(m)} x_j^{(m)} x_j^{(n)} \rangle}_{\text{mean}} = 0$

$$\mathrm{Var}\{x_i^{(m)} x_j^{(m)} x_j^{(n)}\} = \sum_{i=1}^{8} \frac{1}{8} 1 = 1 .$$

Central limit theorem:

$$P(d_i) = \mathcal{N}(d_i \mid \underbrace{(I-1)x_i^{(n)}}_{\mu}, \underbrace{(I-1)(N-1)}_{\sigma^2}) \simeq$$

$$\simeq \mathcal{N}(d_i \mid I x_i^{(n)}, IN).$$



$x_i^{(n)} = 1$

$$\underbrace{P(i \text{ is unstable})}_{\substack{\text{spin will flip}\\ \text{in a single iteration}}} = P(a_i < 0) =$$

$$\frac{1}{\sqrt{IN}}\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{0} da_i\, e^{-\frac{(a_i - I)^2}{2(IN)}} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{-\frac{I}{\sqrt{IN}}} dz\, \sqrt{IN}\, e^{-\frac{z^2}{2}}$$

$$z = \frac{a_i - I}{\sqrt{IN}}$$

$$= \Phi\left(-\frac{I}{\sqrt{IN}}\right) = \Phi\left(-\frac{1}{\sqrt{N/I}}\right)$$

Usually, $\frac{N}{I} \ll 1 \Rightarrow \frac{1}{\sqrt{N/I}} \gg 1$.

Use

$$\left[\ \Phi(-z) \simeq \frac{1}{\sqrt{2\pi}}\,\frac{e^{-z^2/2}}{z}, \quad z \gg 1\ \right] \quad (*)$$

Then, if we require the total error to be ε (all memories, all spins):

$$\Phi\left(-\frac{I}{\sqrt{NI}}\right) \leq \frac{\varepsilon}{NI}$$

error per spin flip

Using (*), we obtain:

$$\left[z = \sqrt{\frac{I}{N}} \right]$$

$$\frac{1}{\sqrt{2\pi}} \sqrt{\frac{N}{I}}\, e^{-\frac{1}{2}\frac{I}{N}} \lesssim \frac{\varepsilon}{NI}, \quad \text{or}$$

$$\underbrace{\log \frac{1}{\sqrt{2\pi}}}_{\downarrow} + \frac{1}{2}(\underbrace{\log N}_{\ll \log I} - \log I) - \frac{1}{2}\frac{I}{N} \lesssim \log \varepsilon - \underbrace{\log N}_{\ll \log I} - \log I,$$

$$+\frac{1}{2}\log I - \log \varepsilon \lesssim \frac{1}{2}\frac{I}{N},$$

$$\frac{I}{N} \gtrsim \log I - 2\log\varepsilon, \quad \text{or}$$

$$\left[N \lesssim \underbrace{\frac{I}{\log I - 2\log \varepsilon}}_{N_{max}} \right].$$

This analysis focuses on single-bit flips => no info about subsequent iterations (avalanches?)

Amit et al. PRL, 1985

spin-glass analysis of HN in the large I limit

$\frac{N}{I} > 0.138$ stable states exist but are uncorrelated with desired memories

$\frac{N}{I} \in (0, 0.138)$ there are stable states "close" to the desired memories

Also, in addition, $\frac{N}{I} \in (0, 0.05)$ desired states are lower in free energy than spurious states.

Correspondingly, if

$\frac{N}{I} \in (0.05, 0.138)$ spurious states dominate (some of them have lower free energies than desired states)

$\frac{N}{I} \in (0, 0.03)$ mixture states appear (combinations of several desired states) but they are higher in free energy than the desired states

---

Can we do better than the Hebbian learning rule?

Make an objective function that, for every pattern $\vec{x}^{(n)}$, if $x_j = x_j^{(n)}$ for $\forall j \neq i$ (all i.e., neurons other than i are set correctly) $\Rightarrow$ we prefer (i.e. assign higher score to) $x_i = x_i^{(n)}$.

So, try

$$E(W) = -\sum_{i=1}^{I} \sum_{n=1}^{N} \left[t_i^{(n)} \log y_i^{(n)} + (1 - t_i^{(n)}) \log (1 - y_i^{(n)}) \right],$$ where

$$t_i^{(n)} = \begin{cases} 1 & \text{if } x_i^{(n)} = 1, \\ 0 & \text{if } x_i^{(n)} = -1 \end{cases}$$

Furthermore,

$$y_i^{(n)} = \frac{1}{1+e^{-a_i^{(n)}}}, \qquad a_i^{(n)} = \sum_{j \neq i} w_{ij} x_j^{(n)}$$

Just like classification of each bit in each memory into +1/-1 classes (K=2)(!) We can use logistic regression to fit the weights & use those weights to build the HN.

[Stores more memories (empirically) than the Hebbian rule.]

Boltzmann machines (BM)

Consider
$$\begin{cases} E(\vec{x}) = -\frac{1}{2}\sum_{i,j} x_i w_{ij} x_j = -\frac{1}{2}\vec{x}^T W \vec{x}, \\ P(\vec{x}) = \frac{1}{Z} e^{-E(\vec{x})} \end{cases}$$

Stochastic Hopfield network (aka Boltzmann machine, BM) ← actually implements Boltzmann distr'n

Activity rule:

Gibbs sampling

$$a_i = \sum_j w_{ij} x_j ,$$

$$x_i = \begin{cases} +1, & \text{prob. } q_i = \dfrac{1}{1+e^{-2a_i}} = \dfrac{e^{a_i}}{e^{a_i}+e^{-a_i}} \\ -1, & \text{prob. } 1-q_i = \dfrac{e^{-a_i}}{e^{a_i}+e^{-a_i}} \end{cases}$$

↗ stochastic update

cf. p. 402 31.1

Consider

$$E(\vec{x}) = -\frac{1}{2}J\sum_{\substack{m,n \\ m\neq n}} x_m x_n - H\sum_n x_n$$

↗ spin glass

Then $b_n = J\sum_{\substack{m \\ m\neq n}} x_m + H$ is the local field for spin n.

Indeed, for 2 spins

$$E = -\frac{J}{2}(x_1x_2 + x_2x_1) - H(x_1+x_2) =$$

$$= -x_2(\underbrace{Jx_1 + H}_{b_2}) - Hx_1 =$$

$$= -x_1(\underbrace{Jx_2 + H}_{b_1}) - Hx_2$$

In general,
$E = -x_n b_n + \mathrm{const}(x_n)$

Gibbs sampling: select spin n at random

$$\begin{cases} P(S_n=+1 \mid b_n) = \dfrac{e^{+\beta b_n}}{e^{\beta b_n}+e^{-\beta b_n}} = \dfrac{1}{1+e^{-2\beta b_n}}, \\ \quad \text{all other spins fixed} \\ P(S_n=-1 \mid b_n) = 1 - P(S_n=+1 \mid b_n) \end{cases}$$

Use these probabilities to set the spin state: ± 1.

This converges to Boltzmann equilibrium.

Metropolis sampling:

Compute $\Delta E = \begin{cases} x_n = 1 \Rightarrow x_n=-1 : E=-b_n + \text{const} \Rightarrow b_n + \text{const} : \Delta E = 2b_n \\ x_n=-1 \Rightarrow x_n = 1 : E = b_n + \text{const} \Rightarrow -b_n + \text{const} : \Delta E = -2b_n \end{cases}$

So, $\underline{\underline{\Delta E = 2 b_n x_n}}$.

$$P(\text{accept spin flip}) = \begin{cases} 1 & \Delta E \le 0 \\ e^{-\beta \Delta E} & \Delta E > 0 \end{cases}$$

This converges to Boltzmann eq'm as well.

Now, given a set of N examples $\{\vec{x}(n)\}_1^N$, we might adjust weights W s.t. the likelihood of generating those examples from the Boltzmann distribution $P(\vec{x})$ is maximized:

$$\mathcal{L} = \prod_{n=1}^{N} P(\vec{x}^{(n)}) \quad , \text{ or}$$

$$\log \mathcal{L} = \sum_{n=1}^{N} \log P(\vec{x}^{(n)}) = \sum_n \left[\frac{1}{2} \vec{x}^{(n)T} W \vec{x}^{(n)} - \right.$$

$$\left. - \log Z \right] .$$

We need

$$\frac{\partial}{\partial w_{ij}} \log Z = \frac{1}{Z} \frac{\partial}{\partial w_{ij}} \left\{ \sum_{\vec{x}} e^{-E(\vec{x})} \right\} =$$

$$= -\sum_{\vec{x}} P(\vec{x}) \frac{\partial}{\partial w_{ij}} E(\vec{x}) = \sum_{\vec{x}} x_i x_j P(\vec{x}) =$$

$$= \langle x_i x_j \rangle_P .$$

Then

$$\frac{\partial}{\partial w_{ij}} \log \mathcal{L} = \underbrace{\sum_n x_i^{(n)} x_j^{(n)}}_{\equiv \, N \langle x_i x_j \rangle_D} - N \langle x_i x_j \rangle_P =$$

$$= N \Big[\underbrace{\langle x_i x_j \rangle_D}_{\substack{\text{empirical} \\ \text{2-point} \\ \text{correl'n}}} - \underbrace{\langle x_i x_j \rangle_P}_{\substack{\text{model} \\ \text{2-point} \\ \text{correl'n}}} \Big] .$$

If $\frac{\partial}{\partial w_{ij}} \log \mathcal{L} = 0 \implies \langle x_i x_j \rangle_D = \langle x_i x_j \rangle_P$

compute directly

estimate by gibbs sampling

Otherwise,

$\langle x_i x_j \rangle_D - \langle x_i x_j \rangle_P$ provides the gradient for optimization algorithms.

Note that if $W = 0 \Rightarrow E(\vec{x}) = 0, \ \forall \vec{x}$.

$\uparrow$ like the $\beta = 0$ limit (infinite T)

Then

$$\langle x_i x_j \rangle_P = \langle x_i \rangle_P \langle x_j \rangle_P = 0,$$

since all spins are equally likely to be up or down.

If the weights are adjusted by the gradient descent,

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \underbrace{\eta}_{\text{learning rate, } >0} \frac{\partial}{\partial w_{ij}} \log \mathcal{L} \Big|_{w_{ij}^{(\tau)}}$$

$\nearrow$ guaranteed to increase $\log \mathcal{L}$ if the step is small:

$$\log \mathcal{L}(w_{ij}^{(\tau+1)}) \simeq \log \mathcal{L}(w_{ij}^{(\tau)}) +$$

$$+ \underbrace{\eta \underbrace{\frac{\partial}{\partial w_{ij}} \log \mathcal{L} \Big|_{w_{ij}^{(\tau)}} \times \frac{\partial}{\partial w_{ij}} \log \mathcal{L} \Big|_{w_{ij}^{(\tau)}}}_{\geq 0}}_{\geq 0 \text{ if } \eta > 0}.$$

Thus in the $W = 0$ case,

$$w_{ij}^{(1)} = \underbrace{w_{ij}^{(0)}}_{=0, \text{ say}} + \eta \sum_n x_i^{(n)} x_j^{(n)} \leftarrow$$

Hebbian learning rule is recovered in 1 iteration