Boltzmann machines (BM) **Lecture 22**

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

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

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

↗ stochastic update

cf. p. 402 31.1

o

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,
$$E = -x_n b_n + \text{const}(x_n)$$

Gibbs sampling: select spin n at random

$$\begin{cases} P(S_n = +1 \mid b_n) = \dfrac{e^{+\beta b_n}}{e^{\beta b_n} + e^{-\beta b_n}} = \dfrac{1}{1+e^{-2\beta b_n}}, \\ \\ P(S_n = -1 \mid b_n) = 1 - P(S_n = +1 \mid b_n) \end{cases}$$

all other spins fixed

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

This converges to Boltzmann equilibrium.

Metropolis sampling:

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

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:

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

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

We need

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

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

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

Then

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

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

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

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,

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

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,

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

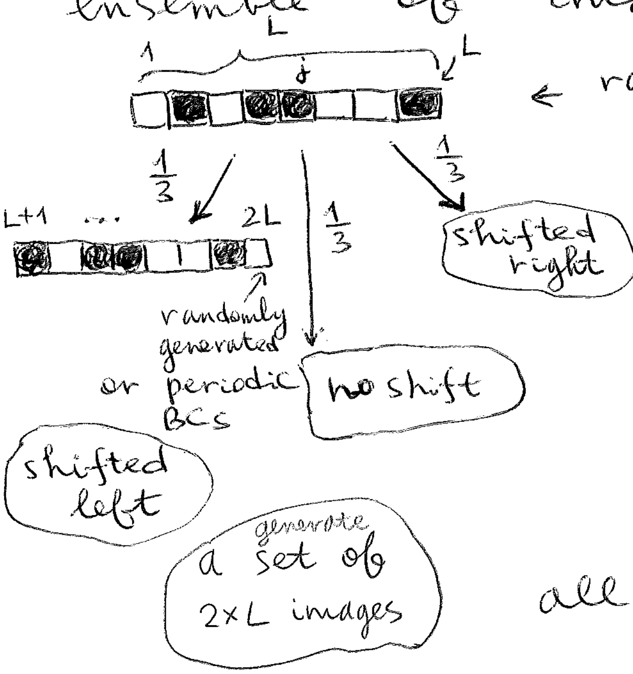
Hebbian learning rule is recovered in 1 iteration

Poetic interpretation of BM learning:

When the BM is "awake" (i.e. gets input from the world), it measures real-world correlations $\langle x_i x_j \rangle_D$ & uses them to adjust the weights. When it is "asleep", it does not adjust the weights - it "dreams" about the world & computes $\langle x_i x_j \rangle_P$ (i.e., its "idea" of the world). When $\langle x_i x_j \rangle_D = \langle x_i x_j \rangle_P$, the two views are balanced.

However, the "world" is represented by just two-point correlations $\langle x_i x_j \rangle_D$, seems to be too poor to really capture the richness of the world.

For example, consider a "shifter ensemble" of images:



← randomly generated pixels

Then, away from the boundaries:

$$\begin{cases} \langle x_j x_{j+L} \rangle = \frac{1}{3} & \text{unshifted} \\ \langle x_j x_{j+L-1} \rangle = \frac{1}{3} & \text{left} \\ \langle x_j x_{j+L+1} \rangle = \frac{1}{3} & \text{right} \end{cases}$$

all others are $= 0$

This seems too poor to describe the images $\Rightarrow$ need higher-order statistics:

$$P(\vec{x}) = \frac{1}{Z} e^{\frac{1}{2}\sum_{ij} w_{ij} x_i x_j + \frac{1}{6}\sum_{ijk} v_{ijk} x_i x_j x_k + \dots}$$

$\uparrow$
higher-order BM

Can get $\frac{\partial}{\partial w_{ij}} \log \mathcal{Z}$, $\frac{\partial}{\partial v_{ijk}} \log \mathcal{Z}$, etc.
do gibbs sampling

[But there are too many parameters]
in general.

<u>Idea:</u> (due to Hinton & Sejnowski, 1986)
introduce hidden variables to model higher-order correlations.

<u>BM with hidden units</u> [restricted BM]

$$\vec{y} = \begin{Bmatrix} \vec{x} \\ \vec{h} \end{Bmatrix} \begin{matrix} \text{visible nodes state } (M_1 \text{ vector}) \\ \text{hidden nodes state } (M_2 \text{ vector}) \end{matrix}$$

$\uparrow$ node states, either visible or hidden ($M_1 + M_2$ vector)

In particular, when visible nodes are "clamped" at $\vec{x}^{(n)} \Rightarrow \vec{y}^{(n)} \equiv (\vec{x}^{(n)}, \vec{h})$.

Then $$P(\vec{x}^{(n)}) = \sum_{\vec{h}} P(\vec{x}^{(n)}, \vec{h}) = \frac{1}{Z} \underbrace{\sum_{\vec{h}} e^{\frac{1}{2}\vec{y}^{(n)T} W \vec{y}^{(n)}}}_{\parallel \; Z_{\vec{x}^{(n)}}},$$

$\leftarrow$ partide partition function

$$Z = \sum_{\vec{x},\vec{h}} e^{\frac{1}{2}\vec{y}^T W \vec{y}}.$$

as before, consider

$$\frac{\partial}{\partial w_{ij}} \log \mathcal{L} = \sum_n \frac{\partial}{\partial w_{ij}} \left\{ \log Z_{\vec{x}^{(n)}} - \log Z \right\} =$$

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

$$= \sum_n \left\{ \frac{1}{Z_{\vec{x}^{(n)}}} \sum_{\vec{h}} y_i^{(n)} y_j^{(n)} e^{\frac{1}{2} \vec{y}^{(n)T} W \vec{y}^{(n)}} - \right.$$

$$\left. - \underbrace{\langle x_i x_j \rangle_{P(\vec{x}, \vec{h})}}_{\text{as before}} \right\} \boxed{=}$$

$$\frac{\sum_{\vec{h}} y_i^{(n)} y_j^{(n)} e^{\frac{1}{2} \vec{y}^{(n)T} W \vec{y}^{(n)}}}{\underbrace{\sum_{\vec{h}} e^{\frac{1}{2} \vec{y}^{(n)T} W \vec{y}^{(n)}}}_{Z_{\vec{x}^{(n)}}}} \equiv \sum_{\vec{h}} y_i^{(n)} y_j^{(n)} P(\vec{h} | \vec{x}^{(n)}) = \langle y_i y_j \rangle_{P(\vec{h}|\vec{x}^{(n)})}$$

$$\boxed{=} \sum_n \left\{ \underbrace{\langle y_i y_j \rangle_{P(\vec{h}|\vec{x}^{(n)})}}_{\substack{\text{estimate by} \\ \text{gibbs sampling} \\ \text{with } \vec{x}^{(n)} \text{ fixed} \\ \text{(only hidden spins} \\ \text{flipped)}}} - \underbrace{\langle y_i y_j \rangle_{P(\vec{x}, \vec{h})}}_{\substack{\text{estimate by unrestricted} \\ \text{gibbs sampling} \\ \text{(both visible \&} \\ \text{hidden spins} \\ \text{flipped)}}} \right\}$$

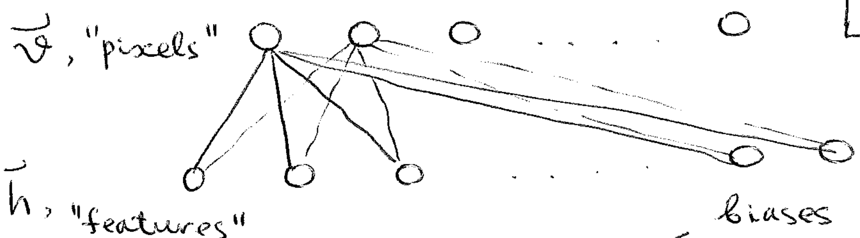
Application of BM in neural networks (NN)

Hinton & Salakhutdinov, Science 2006

<u>Idea</u>: build a multi-layer NN, pre-train intermediate layers using BMs, then refine the weights by backpropagation.

Consider data that can be represented as binary vectors, e.g. images
(0,1) (or vector of spins)

$\boxed{\text{RBM}}$



biases ... weight

$$E(\vec{v},\vec{h}) = -\sum_{i\in \text{pixels}} b_i v_i - \sum_{j\in \text{features}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

given pixel states,

(1) driven by data

$$h_j \underset{\forall j}{=} \begin{cases} 1, & \sigma(b_j + \sum_i v_i w_{ij}) \\ 0, & \text{otherwise} \end{cases} \qquad (*)$$

record $v_i h_j$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

(2) "confabulation"

$$v_i \underset{\forall i}{=} \begin{cases} 1, & \sigma(b_i + \sum_j h_j w_{ij}) \\ 0, & \text{otherwise} \end{cases} \qquad (**)$$

$w_{ij} = w_{ji}$

(3) $\forall j$ $h_j = \begin{cases} 1 & , \sigma(b_j + \sum_i v_i w_{ij}) \\ 0 & , \text{otherwise} \end{cases}$

driven by confabulation

record $v_i h_j$

Repeat many times, compute

$\langle v_i h_j \rangle_{data}$ & $\langle v_i h_j \rangle_{recon}$

Finally, adjust weights:

$$\Delta w_{ij} = \underset{\substack{\uparrow \\ \text{learning rate}}}{\eta} \left(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon} \right)$$

Iterate to convergence.

[Next, make the hidden units the visible units of the next RBM.]

Note: $E(\vec{v}, \vec{h}) = - \sum_i v_i \underbrace{[b_i + \sum_j h_j w_{ij}]}_{\text{local field for } v_i} + \text{const}(\vec{v})$

Then

$$\begin{cases} P(v_i = +1) \text{ (all other spins fixed)} = \dfrac{e^{(b_i + \sum_j h_j w_{ij})}}{\underbrace{e^{(b_i + \sum_j h_j w_{ij})}}_{v_i = 1 \text{ state}} + \underbrace{1}_{v_i = 0 \text{ state}}} = \\ \quad = \sigma(b_i + \sum_j h_j w_{ij}) \\ P(v_i = 0) = 1 - P(v_i = +1) \end{cases}$$

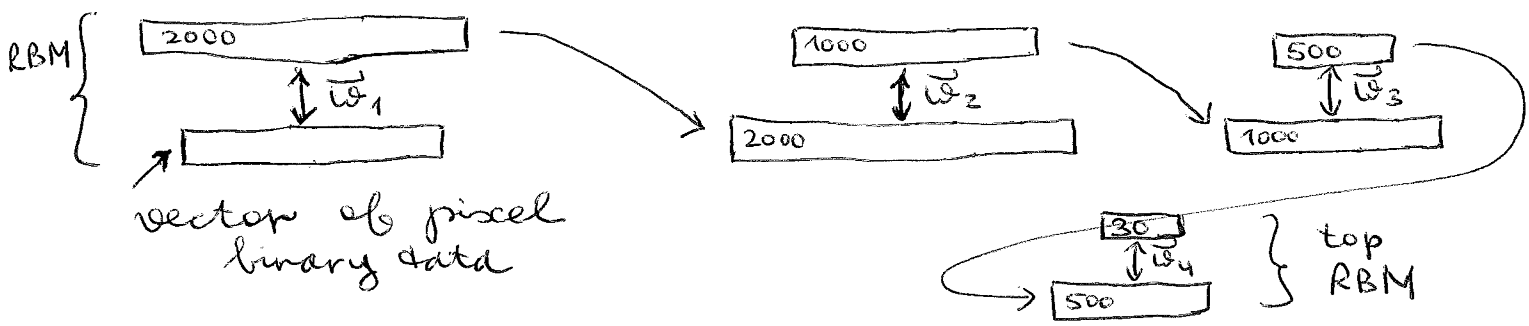
Same as (**)

Likewise,

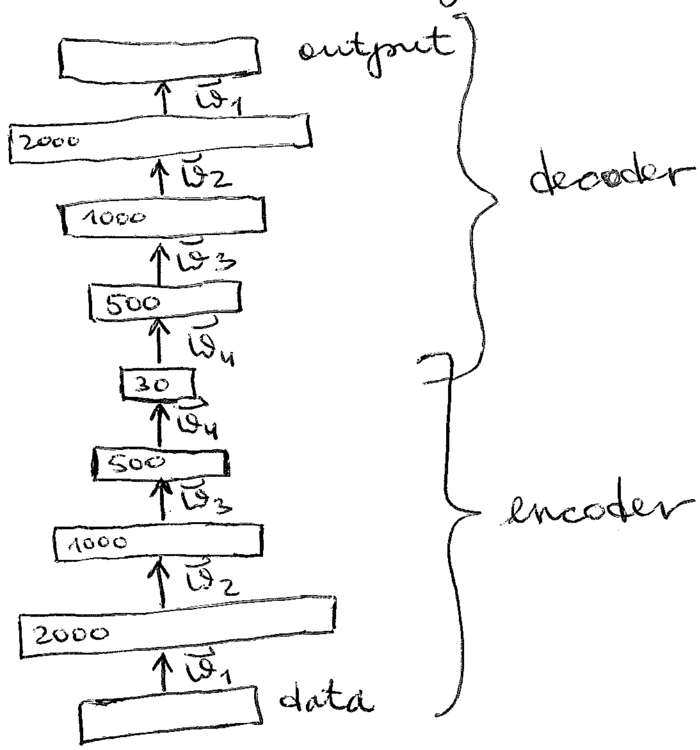
$$E(\vec{v},\vec{h}) = -\sum_j h_j \Big[\underbrace{b_j + \sum_i v_i w_{ij}}_{\text{local field for } h_j}\Big] + \text{const}(\vec{h})$$

leading to (*)

Finally, the whole architecture:



Unrolling:



For backpropagation, replace stochastic units with σ-units with local fields as activations

Minimize the error between output & data by backpropagation with conjugate gradients used on 10^3 data vectors at a time.