

Poetic interpretation of BM learning: Lecture 26

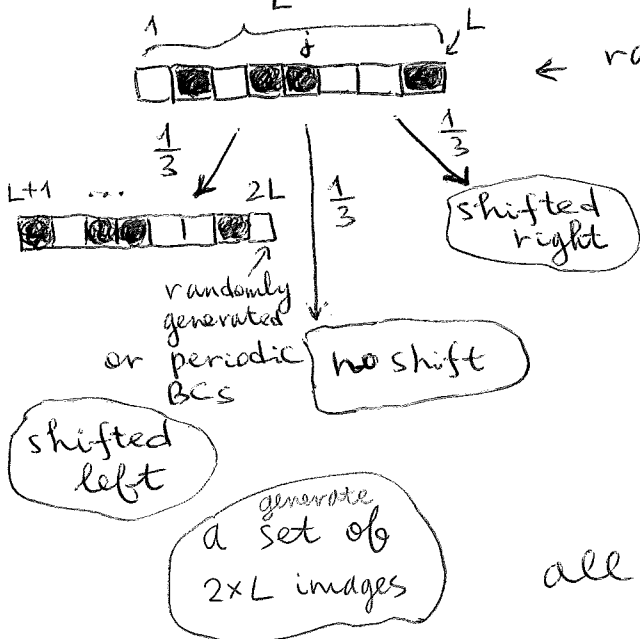
When the BM is "awake", it measures

i.e. ↑
gets input
from the world

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:



Then, away from the boundaries:

$$\begin{cases} \langle x_j x_{j+L} \rangle = \frac{1}{3} & \text{unshifted} \\ \langle x_j x_{j+L-1} \rangle = \frac{1}{3} & \text{left} \\ \langle x_j x_{j+L+1} \rangle = \frac{1}{3} & \text{right} \\ \text{all others are } = 0 \end{cases}$$

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

[But there are too many parameters]
in general.

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

BM with hidden units [restricted BM]

$\vec{y} = \begin{cases} \vec{x} \\ \vec{h} \end{cases}$ visible nodes state (M_1) vector
hidden nodes state (M_2) vector
 \uparrow node state \vec{y} , either visible or hidden
 $(M_1 + M_2)$ vector

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

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

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

\equiv \leftarrow partide partition function

As before, consider

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

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

$$\ominus \sum_n \left\{ \frac{1}{Z_{\vec{x}^{(n)}}} \sum_{\vec{h}} y_i^{(n)} y_j^{(n)} e^{\frac{1}{2} \vec{y}^{(n)T} W \vec{y}^{(n)}} - \underbrace{\langle x_i x_j \rangle_{P(\vec{x}, \vec{h})}}_{\text{as before}} \right\} \quad \equiv$$

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

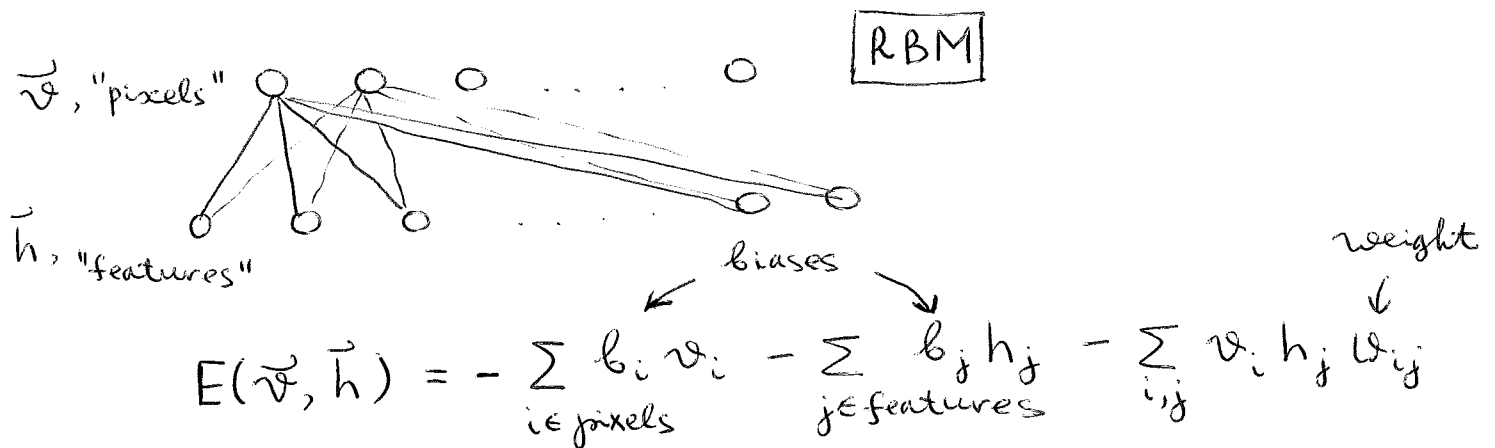
$$\equiv \sum_n \left\{ \underbrace{\langle y_i y_j \rangle_{P(\vec{h} | \vec{x}^{(n)})}}_{\text{estimate by gibbs sampling with } \vec{x}^{(n)} \text{ fixed (only hidden spins flipped)}} - \underbrace{\langle y_i y_j \rangle_{P(\vec{x}, \vec{h})}}_{\text{estimate by unrestricted gibbs sampling (both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

Application of BM in neural networks (NN)

Hinton & Salakhutdinov,
Science 2006

Idea: 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)



given pixel states,

$$(1) \quad \text{driven by data} \quad h_j = \begin{cases} 1, & \sigma(b_j + \sum_i v_i w_{ij}) \\ 0, & \text{otherwise} \end{cases} \quad (*)$$

record $v_i h_j$

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

$$(2) \quad \text{"confabulation"} \quad v_i = \begin{cases} 1, & \sigma(b_i + \sum_j h_j w_{ij}) \\ 0, & \text{otherwise} \end{cases} \quad (**)$$

$w_{ij} = w_{ji}$

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

driven by confabulation record $v_i h_j$

Repeat many times, compute
 $\langle v_i h_j \rangle_{\text{data}}$ & $\langle v_i h_j \rangle_{\text{recon}}$

Finally, adjust weights:

$$\Delta w_{ij} = \eta (\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{recon}})$$

↑
learning rate

— 0 —

Iterate to convergence.

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

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

Then

$$P(v_i = +1) = \frac{e^{b_i + \sum_j h_j w_{ij}}}{e^{b_i + \sum_j h_j w_{ij}} + 1}$$

↑ $v_i = 1$ state ↑ $v_i = 0$ state

$$= \sigma(b_i + \sum_j h_j w_{ij})$$

$$P(v_i = 0) = 1 - P(v_i = +1)$$

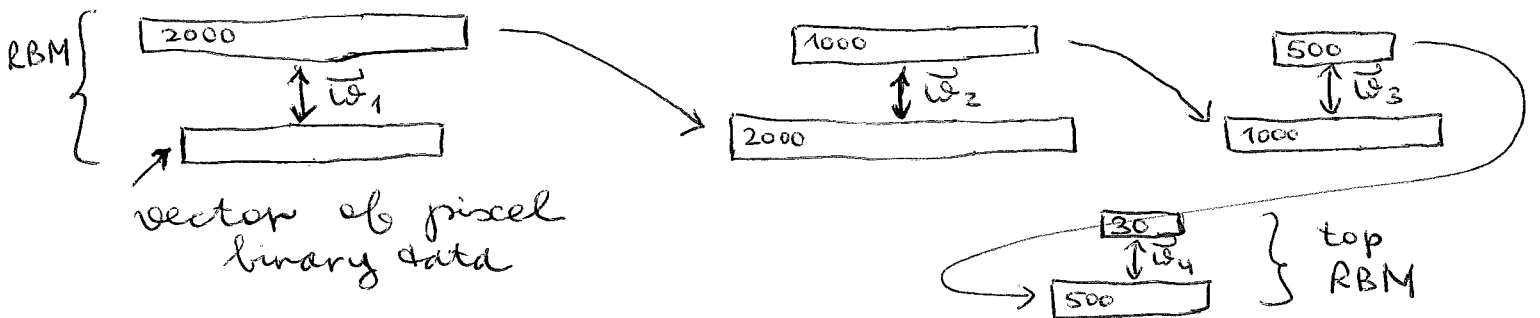
Same as (**)

Likewise,

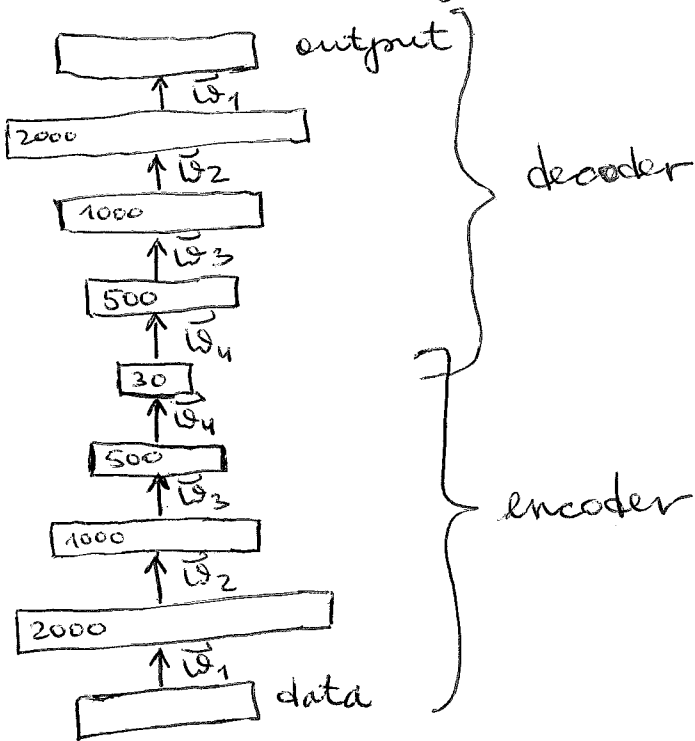
$$E(\vec{v}, \vec{h}) = - \sum_j h_j \left[b_j + \underbrace{\sum_i v_i w_{ij}}_{\text{local field for } h_j} \right] + \text{const}(\vec{h})$$

leading to (*)

Finally, the whole architecture:



Unrolling:



For backpropagation, replace stochastic units with δ -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.