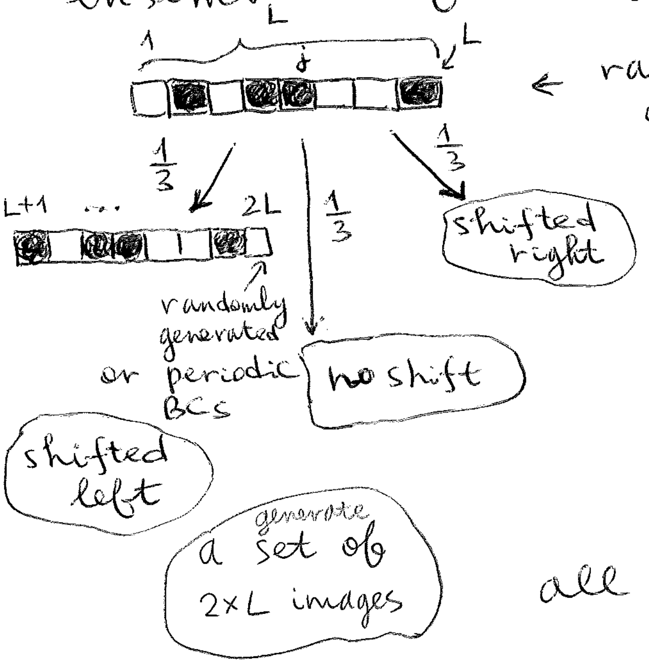
Poetic interpretation of BM learning:

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:


1
L
j
L
← randomly generated pixels
1/3
1/3
1/3
L+1 ...
2L
randomly generated or periodic BCs
shifted right
no shift
shifted left
a generate set of 2×L images

Then, away from the boundaries:

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

all others are $=0$

This seems too poor to describe the images ⇒ need higher-order statistics:

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

↑ 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]

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

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

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

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

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

as before, consider

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

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

$$\text{⊜} \sum_n \Big\{ \frac{1}{Z_{\vec{x}^{(n)}}} \sum_{\vec{h}} y_i^{(n)} y_j^{(n)} e^{\frac{1}{2}\vec{y}^{(n)T} W \vec{y}^{(n)}} - \underbrace{\langle x_i x_j \rangle_{P(\vec{x},\vec{h})}}_{\text{as before}} \Big\} \boxed{=}$$

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

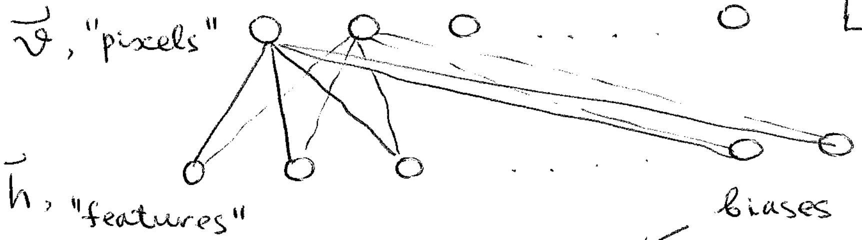
$$\boxed{=} \sum_n \Big\{ \underbrace{\langle y_i y_j \rangle_{P(\vec{h}|\vec{x}^{(n)})}}_{\substack{\text{estimate by} \\ \text{gibbs sampling} \\ \text{with } \vec{x}^{(n)} \text{ fixed} \\ \text{(only hidden spins} \\ \text{flipped)}}} - \underbrace{\langle y_i y_j \rangle_{P(\vec{x},\vec{h})}}_{\substack{\text{estimate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

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)



$\boxed{\text{RBM}}$

biases

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

given pixel states,

(1) driven by data

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

record $v_i h_j$

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

(2)

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

"confabulation"

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

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:

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

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

$$\begin{cases} \overset{\text{all other spins fixed}}{P(v_i = +1)} = \dfrac{e^{(b_i + \sum_j h_j w_{ij})}}{\underset{\nearrow \atop v_i = 1 \text{ state}}{e^{(b_i + \sum_j h_j w_{ij})}} + \underset{\nearrow \atop v_i = 0 \text{ state}}{1}} = \\ \qquad = \sigma(b_i + \sum_j h_j w_{ij}) \\ P(v_i = 0) = 1 - P(v_i = +1) \end{cases}$$

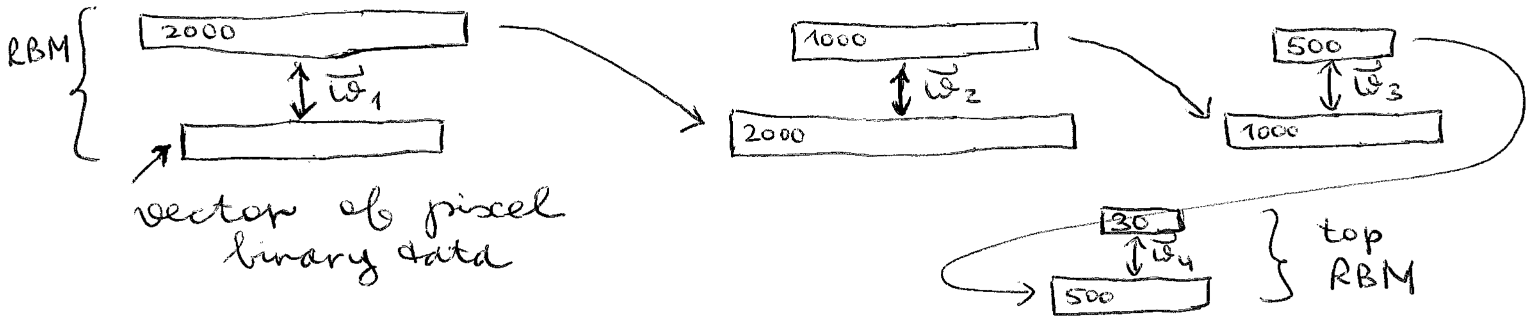
Same as (**)

Likewise,

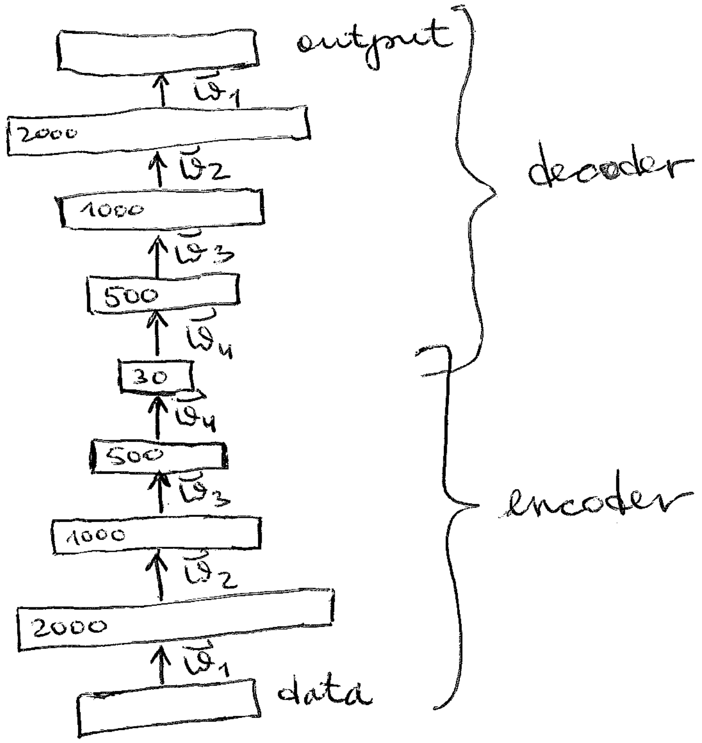
$$E(\vec{v},\vec{h}) = -\sum_j h_j \left[\underbrace{b_j + \sum_i v_i w_{ij}}_{\text{local field for } h_j}\right] + \text{const}(\vec{h})$$

leading to (*)

Finally, the whole architecture:



Unrolling:



For backpropagation, replace stochastic units with 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.