

# Machine learning for vision

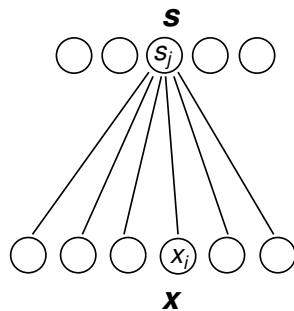
Fall 2013

Roland Memisevic

Lecture 8, October 16, 2013



## RBM graphical model



- ▶ The density over  $\mathbf{x}$  is ICA-like:



## Binary Restricted Boltzmann Machines

- ▶ RBMs define *joint* distribution over data  $\mathbf{x}$  and hidden variables  $\mathbf{s}$  using the energy function:

$$E(\mathbf{x}, \mathbf{s}) = - \sum_{i=1}^D \sum_{j=1}^K W_{ij} x_i s_j - \sum_j b_j s_j - \sum_i c_i x_i$$

- ▶ Exponentiate and normalize to get the distribution:

$$\begin{aligned} p(\mathbf{x}, \mathbf{s}) &= \frac{1}{Z} \exp(-E(\mathbf{x}, \mathbf{s})) \\ &= \frac{1}{Z} \exp\left(\sum_{ij} W_{ij} x_i s_j + \sum_j b_j s_j + \sum_i c_i x_i\right) \end{aligned}$$

with  $Z = \sum_{\mathbf{s}, \mathbf{x}} \exp(-E(\mathbf{x}, \mathbf{s}))$



## Binary RBM marginals

- ▶ The log-probability of an observation,  $\mathbf{x}$ , is:

$$\begin{aligned} \log p(\mathbf{x}) &= \log \sum_{\mathbf{s}} p(\mathbf{x}, \mathbf{s}) = -\log Z + \log \sum_{\mathbf{s}} \exp\left(\sum_{ij} W_{ij} x_i s_j + \sum_j b_j s_j + \sum_i c_i x_i\right) \\ &= -\log Z + \log\left(\exp\left(\sum_i c_i x_i\right) \sum_{s_1} \dots \sum_{s_K} \exp\left(\sum_j s_j [b_j + \sum_i W_{ij} x_i]\right)\right) \\ &= -\log Z + \sum_i c_i x_i + \log \sum_{s_1} \dots \sum_{s_K} \prod_j \exp\left(s_j [b_j + \sum_i W_{ij} x_i]\right) \\ &= -\log Z + \sum_i c_i x_i + \log \sum_{s_1} \exp\left(s_1 [b_1 + \sum_i W_{i1} x_i]\right) \sum_{s_2} \exp(\dots) \sum_{s_3} \dots \\ &\quad \dots \sum_{s_K} \exp\left(s_K [b_K + \sum_i W_{iK} x_i]\right) \\ &= -\log Z + \sum_i c_i x_i + \sum_j \log\left(1 + \exp\left(b_j + \sum_i W_{ij} x_i\right)\right) \end{aligned}$$

- ▶ So the probability is proportional to a product over  $K$  terms, each of which is based on a filter-response  $\mathbf{w}_j^T \mathbf{x}$ :

$$p(\mathbf{x}) \propto \prod_j \left(1 + \exp\left(b_j + \sum_i W_{ij} x_i\right)\right)$$



## Tractable computations

- ▶ Computing the probability is not tractable in general, because we cannot compute  $(\log)Z$ .
- ▶ But inferring the feature responses is easy.
- ▶ We can also *compare* the  $(\log)$ -probabilities of two points  $\mathbf{x}_1, \mathbf{x}_2$  because  $(\log) Z$  will cancel in the ratio (difference). So we can always tell which point is more likely under the model.

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍

Roland Memisevic

Machine learning for vision

## Binary RBM conditional over data

- ▶ Using an analogous calculation for  $\mathbf{x}$  one gets:

$$p(x_i = 1 | \mathbf{s}) = \frac{1}{1 + \exp(-[c_i + \sum_j W_{ij}s_j])}$$

- ▶ It turns out that if we change the RBM energy function to

$$E(\mathbf{x}, \mathbf{s}) = - \sum_{i=1}^D \sum_{j=1}^K W_{ij} x_i s_j - \sum_j b_j s_j + \frac{1}{2\sigma^2} \sum_j (x_j - c_j)^2$$

the conditional over  $\mathbf{x}$  turns into a Gaussian with mean  $c_i + \sum_j W_{ij} x_j$  and variance  $\sigma^2$  in each dimension.

- ▶ Similarly, one can turn the conditionals over data or hiddens into any *exponential family* distribution.

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍

Roland Memisevic

Machine learning for vision

## Binary RBM conditional over hiddens

- ▶ The conditional over the hiddens is

$$\begin{aligned} p(\mathbf{s} | \mathbf{x}) \propto p(\mathbf{x}, \mathbf{s}) &= \Omega_1 \exp\left(\sum_{ij} W_{ij} x_i s_j + \sum_j b_j s_j + \sum_i c_i x_i\right) \\ &= \Omega_1 \exp\left(\sum_j s_j [b_j + \sum_i W_{ij} x_i]\right) \exp\left(\sum_i c_i x_i\right) \\ &= \Omega_2 \exp\left(\sum_j s_j [b_j + \sum_i W_{ij} x_i]\right) \\ &= \Omega_2 \prod_j \exp\left(s_j [b_j + \sum_i W_{ij} x_i]\right) \end{aligned}$$

- ▶ This is a product over all  $s_j$ , so the hiddens are independent, given the data.
- ▶ For each  $s_j$  we have  $p(s_j | \mathbf{x}) \propto \exp\left(s_j [b_j + \sum_i W_{ij} x_i]\right)$
- ▶ But  $s_j$  is binary, so

$$p(s_j = 1 | \mathbf{x}) = \frac{\exp(b_j + \sum_i W_{ij} x_i)}{1 + \exp(b_j + \sum_i W_{ij} x_i)} = \frac{1}{1 + \exp(-[b_j + \sum_i W_{ij} x_i])}$$

- ▶ This is just a logistic sigmoid applied to a linear projection of the data.

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍

Roland Memisevic

Machine learning for vision

## RBM training

- ▶ Neither the log-probability nor its derivative are tractable.
- ▶ So one has to use approximations for learning.
- ▶ One such approximation is to use Gibbs sampling.
- ▶ This leads to a learning method known as *contrastive divergence*.
- ▶ Like  $K$ -means, it comes down to the combination of a Hebbian term and “active forgetting”.

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍

Roland Memisevic

Machine learning for vision

## Binary RBM marginals and free energy

- ▶ Recall that we can compute  $p(\mathbf{x})$  up to normalizing constant:

$$p(\mathbf{x}) = \sum_{\mathbf{s}} p(\mathbf{x}, \mathbf{s}) = \frac{1}{Z} \sum_{\mathbf{s}} \exp(-E(\mathbf{x}, \mathbf{s})) \\ \propto \prod_j \left( 1 + \exp(b_j + \sum_i W_{ij} x_i) \right)$$

- ▶ To derive learning rules it can be convenient to write this as

$$p(\mathbf{x}) = \frac{1}{Z} \exp(-\mathcal{F}(\mathbf{x}))$$

with

$$\mathcal{F}(\mathbf{x}) = -\log \sum_{\mathbf{s}} \exp(-E(\mathbf{x}, \mathbf{s}))$$

- ▶  $\mathcal{F}(\mathbf{x})$  is known as *Free Energy* (we can compute it)

## Binary RBM derivatives

- ▶ To train the RBM we need the derivative of the log-probability of data.
- ▶ Let  $\theta$  denote some parameter in the model.
- ▶ The derivative of  $\log p(\mathbf{x}^*)$  for some training point  $\mathbf{x}^*$  is

$$\frac{\partial \log p(\mathbf{x}^*)}{\partial \theta} = -\frac{\partial \mathcal{F}(\mathbf{x}^*)}{\partial \theta} + \frac{1}{Z} \sum_{\mathbf{x}} \exp(-\mathcal{F}(\mathbf{x})) \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \theta} \\ = -\frac{\partial \mathcal{F}(\mathbf{x}^*)}{\partial \theta} + \sum_{\mathbf{x}} p(\mathbf{x}) \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \theta}$$

## Binary RBM marginals and free energy

- ▶ Recall that the Free Energy can be written:

$$\mathcal{F}(\mathbf{x}) = -\sum_j \log \left( 1 + \exp(b_j + \sum_i W_{ij} x_i) \right) - \sum_i c_i x_i$$

- ▶ The Free Energy is convenient to work with because it depends only on  $\mathbf{x}$  and not on  $\mathbf{s}$

## Binary RBM derivatives

- ▶ Furthermore, we have

$$\frac{\partial \mathcal{F}(\mathbf{x}^*)}{\partial \theta} = \sum_{\mathbf{s}} \frac{\exp(-E(\mathbf{x}^*, \mathbf{s}))}{\sum_{\mathbf{s}'} \exp(-E(\mathbf{x}^*, \mathbf{s}'))} \frac{\partial E(\mathbf{x}^*, \mathbf{s})}{\partial \theta} \\ = \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}^*) \frac{\partial E(\mathbf{x}^*, \mathbf{s})}{\partial \theta}$$

- ▶ Finally, we have

$$\frac{\partial E(\mathbf{x}, \mathbf{s})}{\partial W_{ij}} = -x_i s_j, \quad \frac{\partial E(\mathbf{x}, \mathbf{s})}{\partial b_j} = -s_j, \quad \frac{\partial E(\mathbf{x}, \mathbf{s})}{\partial c_i} = -x_i$$

## Binary RBM derivatives

- ▶ Putting everything together yields:

$$\frac{\partial \log p(\mathbf{x}^*)}{\partial \theta} = - \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}^*) \frac{\partial E(\mathbf{x}^*, \mathbf{s})}{\partial \theta} + \sum_{\mathbf{x}, \mathbf{s}} p(\mathbf{x}, \mathbf{s}) \frac{\partial E(\mathbf{x}, \mathbf{s})}{\partial \theta}$$

with

$$\frac{\partial E(\mathbf{x}, \mathbf{s})}{\partial W_{ij}} = -x_i s_j, \quad \frac{\partial E(\mathbf{x}, \mathbf{s})}{\partial b_j} = -s_j, \quad \frac{\partial E(\mathbf{x}, \mathbf{s})}{\partial c_i} = -x_i$$

- ▶ This is the sum of two terms: (i) an expectation under  $p(\mathbf{s}|\mathbf{x}^*)$  and (ii) an expectation under  $p(\mathbf{x}, \mathbf{s})$ .
- ▶ The second term is, in general, is not tractable, so we have to approximate it.



## Digression: Gibbs Sampling

- ▶ Goal: Get a sample (a set of points  $\mathbf{z}$ ) from some distribution

$$p(\mathbf{z}) = p(z_1, \dots, z_M)$$

- ▶ Solution: Define a Markov chain in  $\mathbf{z}$ -space.
- ▶ If  $p(\mathbf{z})$  is an invariant distribution of the chain, and under some additional technical assumptions (“detailed balance”, “ergodicity”), running the Markov chain long enough will yield samples from  $p(\mathbf{z})$ .



## Approximating the expectation

- ▶ But the second term is just an expectation under the model.
- ▶ Consider the expectation of some function  $f(\mathbf{z})$  under some distribution  $p(\mathbf{z})$ :

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

- ▶ If we have  $L$  IID samples  $\mathbf{z}^{(l)}$  from  $p(\mathbf{z})$ , we can approximate the expectation using the empirical expectation

$$\hat{f} = \frac{1}{L} \sum_l f(\mathbf{z}^{(l)})$$

- ▶ Note that

$$\mathbb{E}[\hat{f}] = \mathbb{E}[f]$$



## Digression: Gibbs Sampling

- ▶ Gibbs sampling is one way to define such a Markov chain:
- ▶ It works in the case where we *can* sample from each conditional

$$p(z_i | z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_K)$$

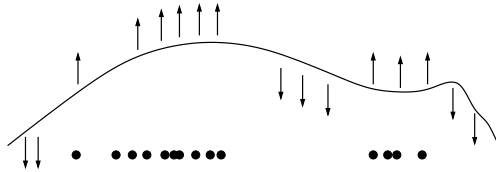
given all the other variables.

- ▶ (This is the case for the RBM.)
- ▶ It amounts to cycling through all variables (randomly or in some specific order) and updating  $\mathbf{z}$  by sampling the conditionals:





## RBM learning intuition



- ▶ As we saw before, this has the effect of increasing  $p(\mathbf{x})$  at the data, and decreasing  $p(\mathbf{x})$  everywhere else.



## Encouraging sparsity

- ▶ The RBM hiddenes are typically not all that sparse and the filters do not always resemble simple cell features.
- ▶ It is common to encourage hidden variables to be sparse(r) by adding a penalty function to the learning objective, like

$$\lambda \sum_{j=1}^K \left( B - \frac{1}{N} \sum_{n=1}^N p(s_j | \mathbf{x}_n) \right)^2$$

where  $B$  is a small positive number, for example,  $B = 0.1$ , and  $\lambda$  is a sparsity parameter.

- ▶ This will pull hidden variable activities towards  $B$  during training.



## Contrastive divergence training

- ▶ This intuition suggests a short-cut to speed up the sampling, consisting of two modifications to standard Gibbs sampling:
  1. Start sampling *at the data*.
  2. Rather than waiting to reach the equilibrium, sample for *just a few steps* (for example, for 1 step).
- ▶ This is known as *contrastive divergence* (CD) learning.
- ▶ For one-step CD, the positive phase consists in computing, or sampling from,  $p(\mathbf{s} | \mathbf{x})$  and the negative phase of sampling from  $p(\mathbf{x} | \mathbf{s})$ .
- ▶ The result is a model that is a good *near the data* and potentially bad elsewhere in the data space (but that may not matter!).



## Stacking

- ▶ After training, we can infer the  $\mathbf{s}$  for data, and train another model on the inferred  $\mathbf{s}$ .
- ▶ Since the  $\mathbf{s}$  depend non-linearly on the data, this won't be a vacuous operation. (For linear model it would be).
- ▶ This is what coined the term “deep learning” in 2006.
- ▶ It allows us to train a feature hierarchy greedily, layer-by-layer.
- ▶ It is now common to stack other non-linear models, such as autoencoders.

