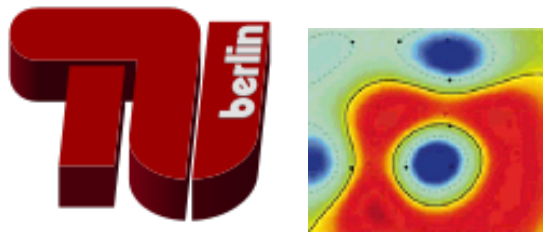
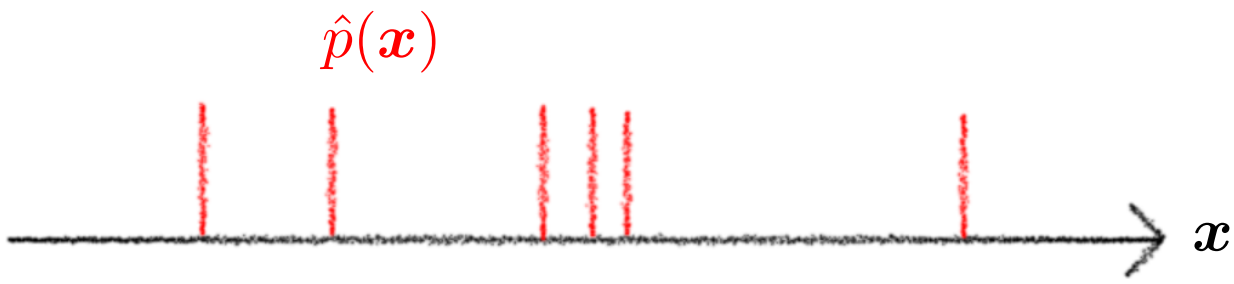
Lecture 6: Restricted Boltzmann Machines

Machine Learning 2



TU Berlin – SoSe 2016

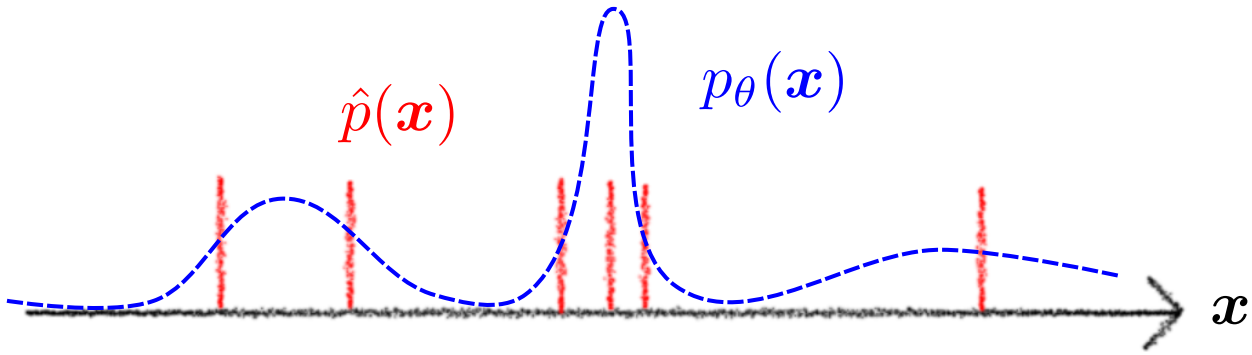
Density Estimation



Goal:

- Infer the probability distribution $p(\boldsymbol{x})$ that has been generating these few data points.

Density Estimation



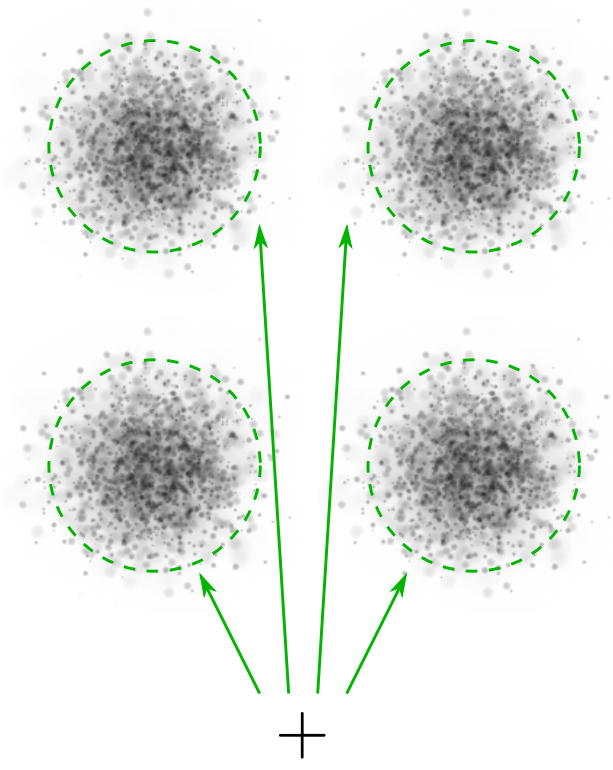
- Try to learn from $\hat{p}(\boldsymbol{x})$ a distribution $p_\theta(\boldsymbol{x})$ that is as close as possible to the true distribution $p(\boldsymbol{x})$.
- Divergence between distribution can be measured using e.g. the KL divergence:

$$\mathrm{KL}(\hat{p}||p_\theta) = \int_{\boldsymbol{x}} \hat{p}(\boldsymbol{x})[\log \hat{p}(\boldsymbol{x}) - \log p_\theta(\boldsymbol{x})]$$

- Other divergence measures exist (e.g. α-divergence, Wasserstein distance, etc.).

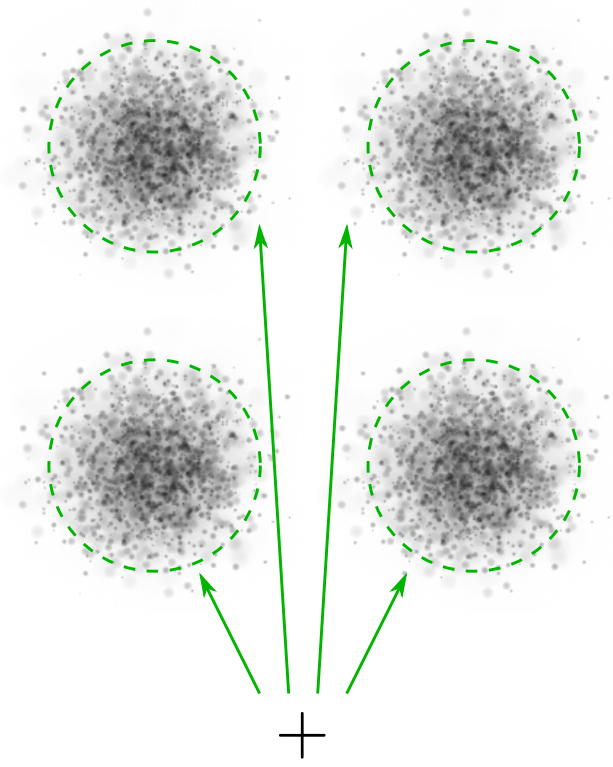
Various Approaches to Density Modeling

1. Local Mixture

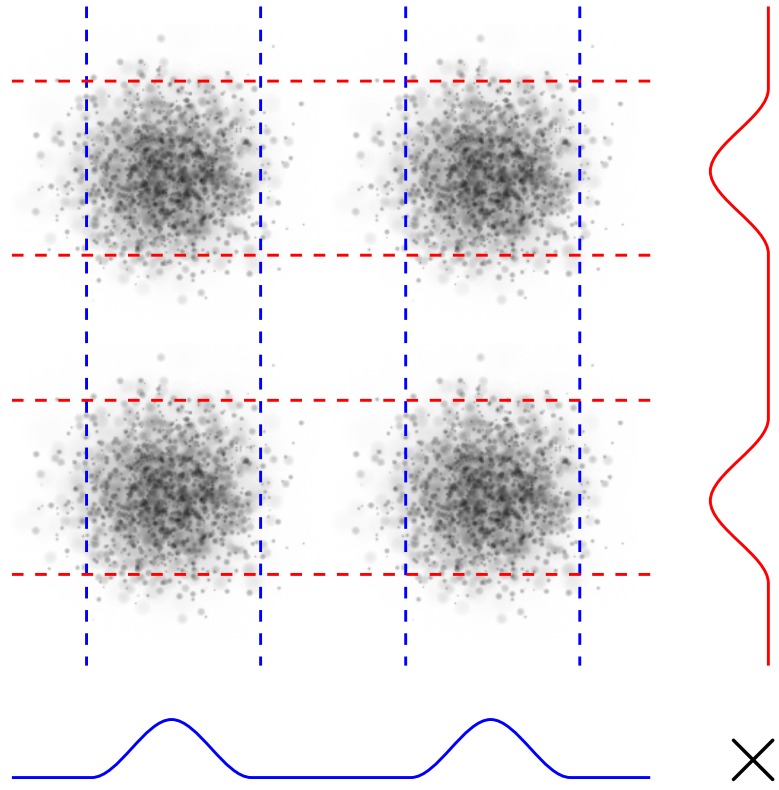


Various Approaches to Density Modeling

1. Local Mixture



2. Product of Experts



Various Approaches to Density Modeling

Gaussian Mixture Model *(studied in ML1)*

$$p_\theta(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \exp\Big(\underbrace{-(\boldsymbol{x}-\boldsymbol{\mu}_j)^\top \Sigma_j^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_j)}_{\text{local containment}} \Big)$$

Restricted Boltzmann Machine *(today's lecture)*

$$p_\theta(\boldsymbol{x}) = \frac{1}{Z} \exp(\boldsymbol{a}^\top \boldsymbol{x}) \prod_{j=1}^{N} \Big(1 + \exp\big(\underbrace{\boldsymbol{w}_j^\top \boldsymbol{x} + b_j}_{\text{global projection}} \big)\Big)$$

Restricted Boltzmann Machines

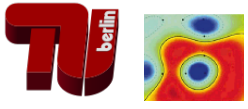
A Restricted Boltzmann machine (RBM) is a *parameterized* probability distribution over $\boldsymbol{x} \in \{0,1\}^d$ defined as:

$$p_\theta(\boldsymbol{x}) = \sum_{\boldsymbol{h} \in \{0,1\}^N} p_\theta(\boldsymbol{x}, \boldsymbol{h})$$

where

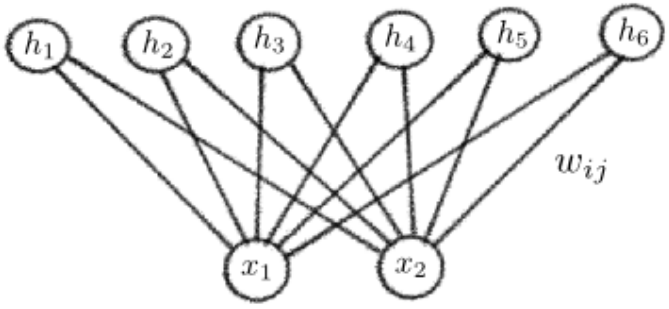
- $$p_\theta(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{Z} \exp\Big(\sum_{i,j} x_i w_{ij} h_j + \sum_i a_i x_i + \sum_j b_j h_j \Big)$$
$$= \frac{1}{Z} \exp \underbrace{\Big(\boldsymbol{x}^\top W \boldsymbol{h} + \boldsymbol{a}^\top \boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{h} \Big)}_{-E_\theta(\boldsymbol{x}, \boldsymbol{h})},$$

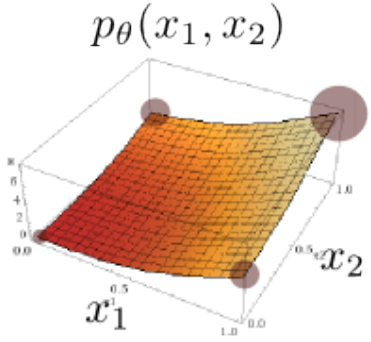
- Z is the *partition function* that normalizes the probability distribution
- $\theta = \{W, \boldsymbol{a}, \boldsymbol{b}\}$ are the parameters of the model.

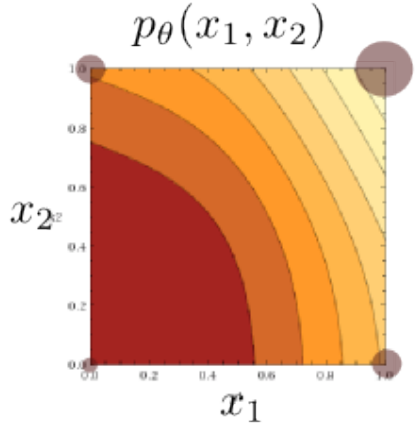


Restricted Boltzmann Machines

Example: RBM with $d = 2$ visible units and $N = 6$ hidden units.







- The RBM assigns probability mass on vertices $\{0, 1\}^2$ of the input hypercube.
- The value of the probability function at other locations is irrelevant.

The Feature View of an RBM

The probability distribution

$$p_\theta(\boldsymbol{x}) = \sum_{\boldsymbol{h} \in \{0,1\}^N} p_\theta(\boldsymbol{x}, \boldsymbol{h})$$

with

$$p_\theta(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{Z} \exp\left(\sum_{i,j} x_i w_{ij} h_j + \sum_i a_i x_i + \sum_j b_j h_j \right)$$

can be rewritten as:

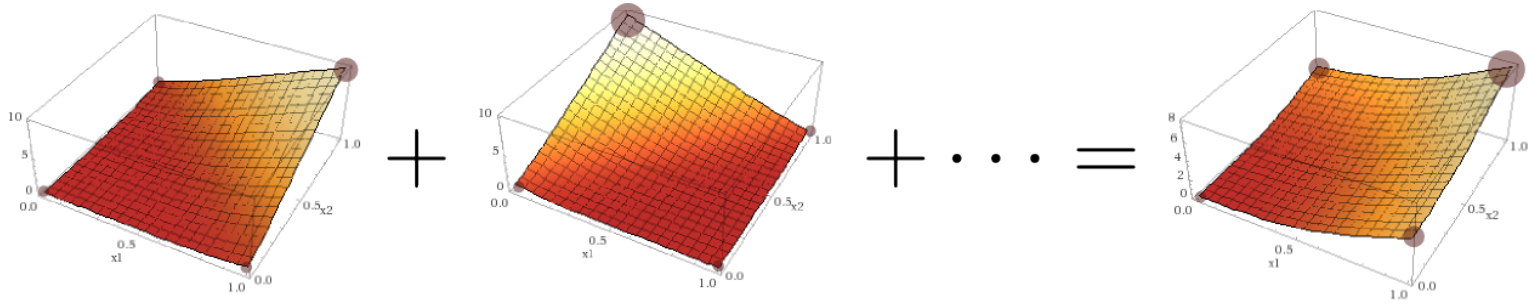
$$p_\theta(\boldsymbol{x}) = \frac{1}{Z} \exp\Bigg(\underbrace{\boldsymbol{a}^\top \boldsymbol{x} + \sum_{j=1}^{N} \operatorname{softplus}(W_j^\top \boldsymbol{x} + b_j)}_{-F_\theta(\boldsymbol{x})} \Bigg)$$

where $\operatorname{softplus}(t) = \log(1 + e^t)$.

[Derivations on the blackboard]

RBMs can Model any Binary Distributions

Any *binary* (log-)probability distribution $\log p(\boldsymbol{x})$ over $\{0,1\}^d$ can be written as a superposition of many softplus functions:



Remark: Universality does not hold for the *continuous* distributions in $[0,1]^d$. Reason: a sum of convex functions can only produce a convex function. Any non-convex log-probability function cannot be represented.

Learning RBMs

- **Goal:** Learn a set of RBM parameters $\theta = \{W, \boldsymbol{a}, \boldsymbol{b}\}$ from an ensemble of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$.

- **Optimization problem:**

$$\min_{\theta} \quad D_{\mathrm{KL}}(p \,\|\, p_\theta)$$

where

$$p(\boldsymbol{x}) = \frac{1}{n} \sum_{k=1}^{n} \delta(\boldsymbol{x} - \boldsymbol{x}_k)$$

and

$$p_\theta(\boldsymbol{x}) = \frac{1}{Z} \exp(-F_\theta(\boldsymbol{x}))$$

- **Idea:** Minimize the objective using gradient descent.

Learning RBMs

- **Result 1:** The gradient of the KL divergence is given by:

$$\nabla_\theta D_{\mathrm{KL}}(p \,\|\, p_\theta) = \langle \nabla_\theta F_\theta(\boldsymbol{x}) \rangle_p - \langle \nabla_\theta F_\theta(\boldsymbol{x}) \rangle_{p_\theta}$$

  and depends on the RBM free energy $F_\theta(\boldsymbol{x})$.

  [Derivations on the blackboard]

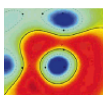
- **Result 2:** The gradient of the free energy is given by

$$\nabla_{w_{ij}} F_\theta(\boldsymbol{x}) = \underbrace{\mathrm{sigm}(W_j^\top \boldsymbol{x} + b_j)}_{p(h_j=1|\boldsymbol{x})} \cdot x_i$$

  [Derivations on the blackboard]

Learning RBMs

- **Problem:** Evaluating the expectation $\langle \cdot \rangle_{p_\theta}$ in the gradient requires to evaluate a function for an exponentially large number of states (2^d).
- **Idea:** Replace the expectation operator by an empirical estimate, and sample only n realizations from the $p_\theta(x)$.
- **Remark:** Ideally, the sample must be unbiased so that for n large, its statistics converge to those of the original probability distribution p_θ.
- **Question**: How do we sample from p_θ ?



Sampling in an RBM

- **Observation:**

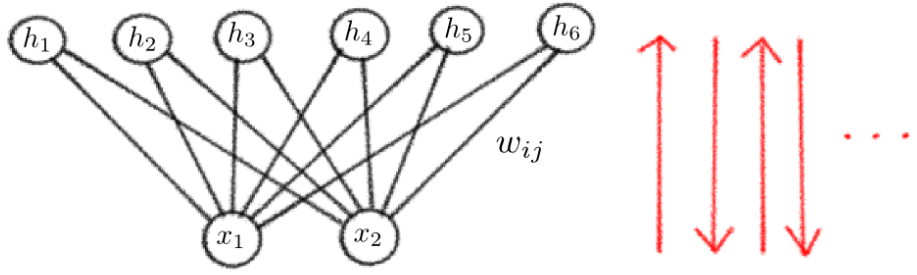
$$p(\boldsymbol{h}|\boldsymbol{x}) = \prod_j \underbrace{p(h_j|\boldsymbol{x})}_{\mathrm{Ber}(\mathrm{sigm}(W_j^\top \boldsymbol{x}))} \quad \text{and} \quad p(\boldsymbol{x}|\boldsymbol{h}) = \prod_i \underbrace{p(x_i|\boldsymbol{h})}_{\mathrm{Ber}(\mathrm{sigm}(W_i^\top \boldsymbol{h}))}$$

[Derivations on the blackboard]

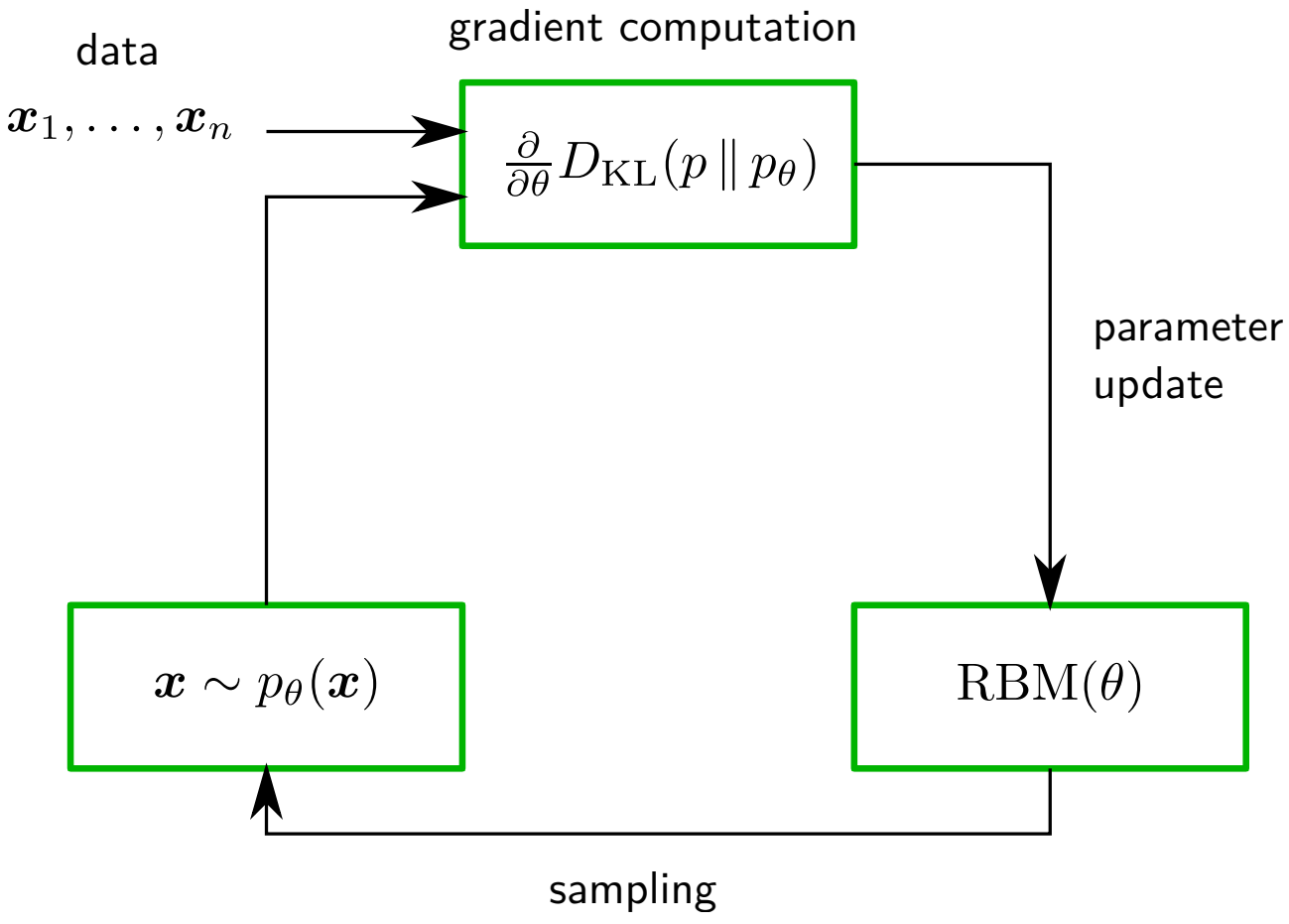
- **Consequence:** Knowing $\boldsymbol{x}$, we can sample $\{h_j\}$'s all at once and knowing $\boldsymbol{h}$, we can sample $\{x_i\}$'s all at once.

$$\boldsymbol{h} \leftarrow \mathbf{Ber}(\mathbf{sigm}(W^\top \boldsymbol{x} + \boldsymbol{b}))$$
$$\boldsymbol{x} \leftarrow \mathbf{Ber}(\mathbf{sigm}(W\boldsymbol{h} + \boldsymbol{a}))$$

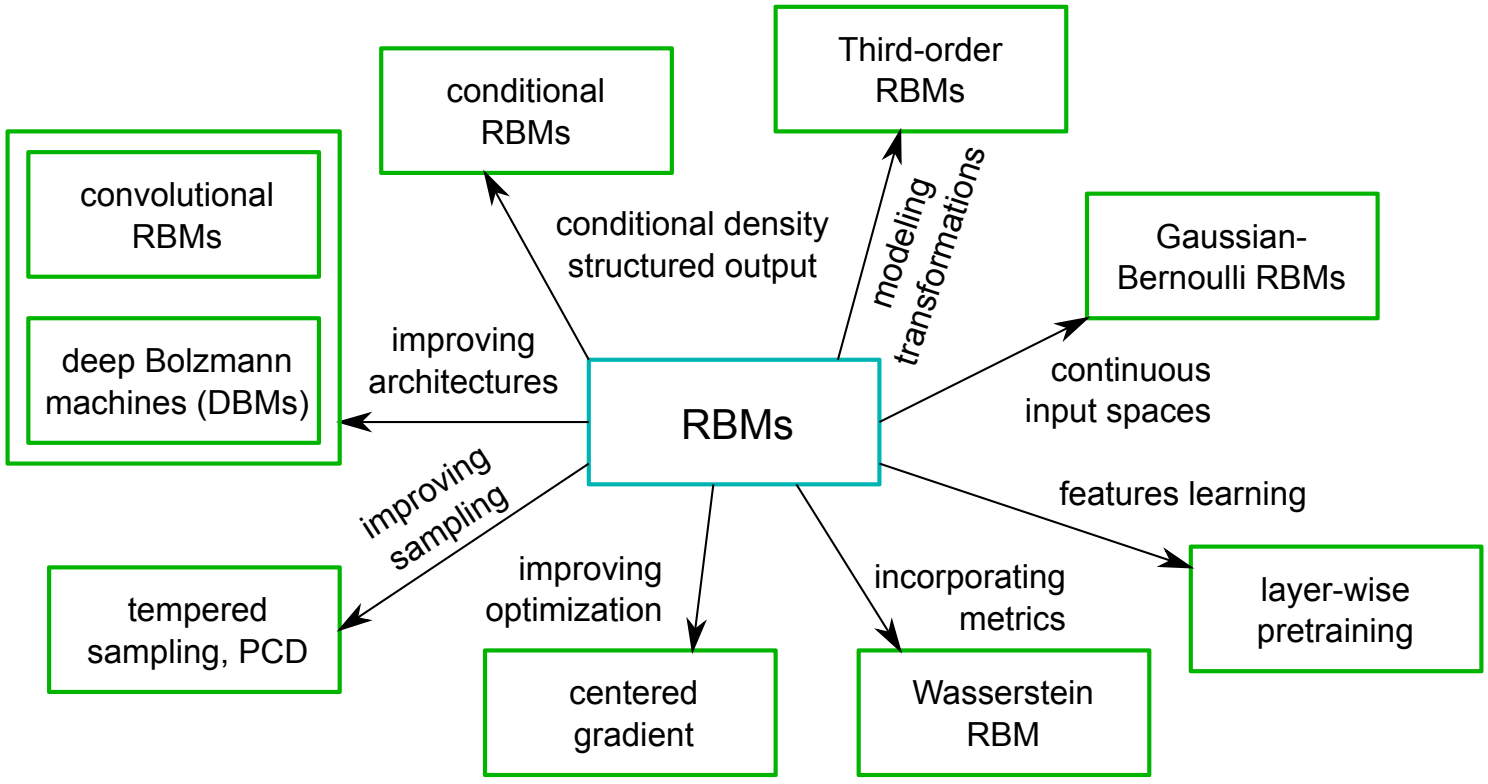
where $\mathbf{Ber}()$ and $\mathbf{sigm}()$ apply element-wise.



Wrapup


data
$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$
gradient computation
$\frac{\partial}{\partial \theta} D_{\mathrm{KL}}(p \,\|\, p_\theta)$
parameter
update
$\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
$\mathrm{RBM}(\theta)$
sampling

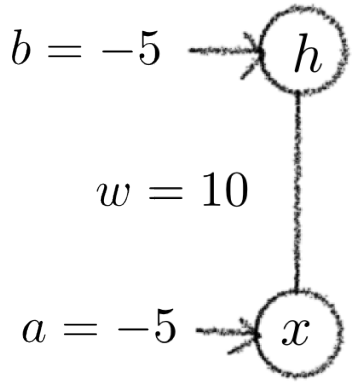
The world of Boltzmann Machines



Improving Sampling

- **Question:** How long do we need to run the Gibbs sampling procedure for it to produce an unbiased sample of the data distribution?
- **Example: Two-Unit RBM**

$$p(x) = \frac{1}{Z} \exp(xwh + bh + ax)$$



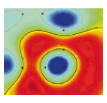
	$h = 0$	$h = 1$
$x = 0$	0.497	0.003
$x = 1$	0.003	0.497

Will move from one mode to the other with probability 0.006 (on average every 167 iterations).

Improving Sampling

Strategies to overcome the sampling problem:

- **Annealing:** Consider a simpler model θ_{easy} and run the alternate Gibbs sampler while gradually restoring θ to its original value.
- **Improved architectures:** Design Boltzmann machine architectures in which sampling is faster.
- **Approximate sampling**:
  - *Contrastive divergence:* Sample by running a predefined number k of Gibbs updates starting from an existing data point (CD-k).
  - *Sampling by learning:* Use the impulse of learning to force the Gibbs sampler to escape from its current mode (persistent CD).

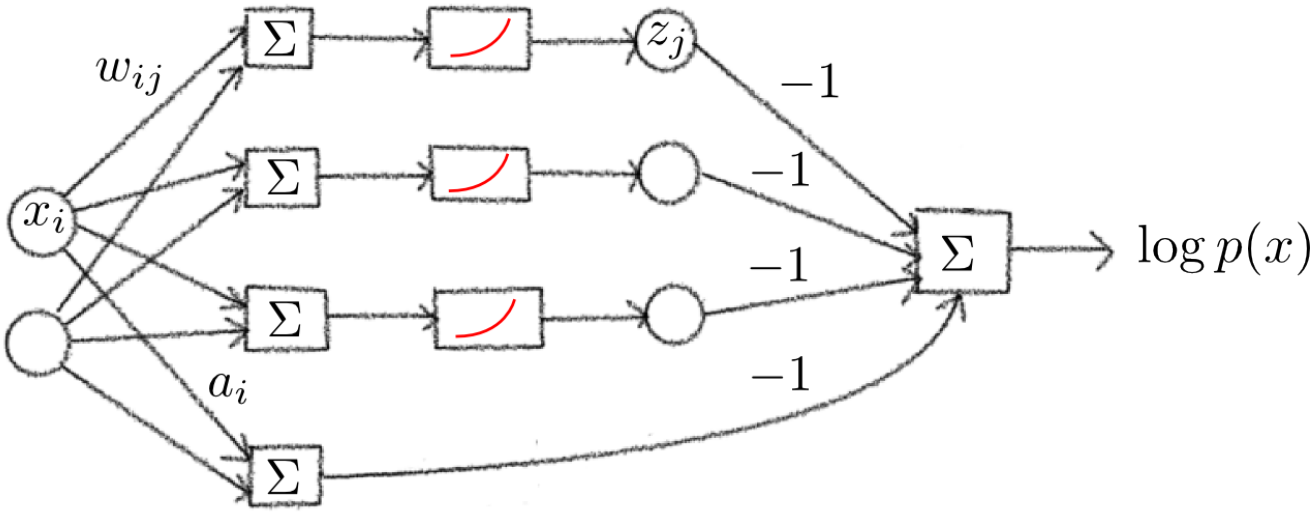


Initializing Neural Networks with RBMs

Reminder: The free energy of an RBM is given by:

$$F(\boldsymbol{x}) = -\boldsymbol{a}^\top \boldsymbol{x} - \sum_{j=1}^{N} \operatorname{softplus} \underbrace{(W_j^\top \boldsymbol{x} + b_j)}_{z_j}$$

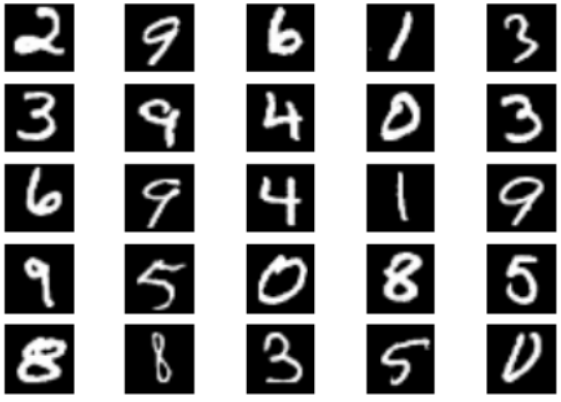
Neural Network Equivalent:



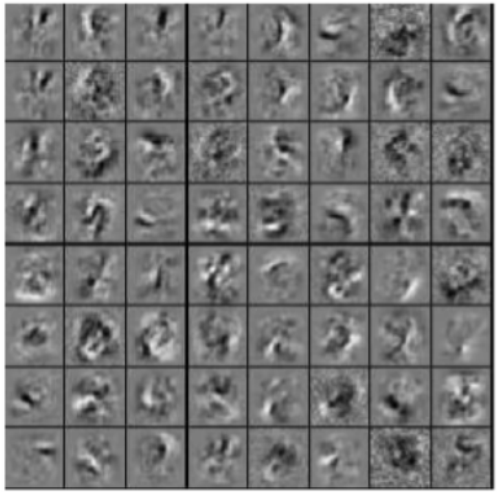
Idea: Reuse the first layer of the network to train on the discriminative problem.

Initializing Neural Networks with RBMs

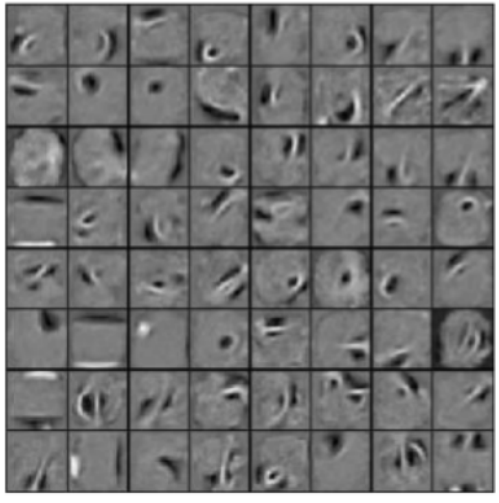
Example: MNIST



First layer weights
without pretraining



First layer weights
with pretraining



Source: Erhan, JMLR, 2010