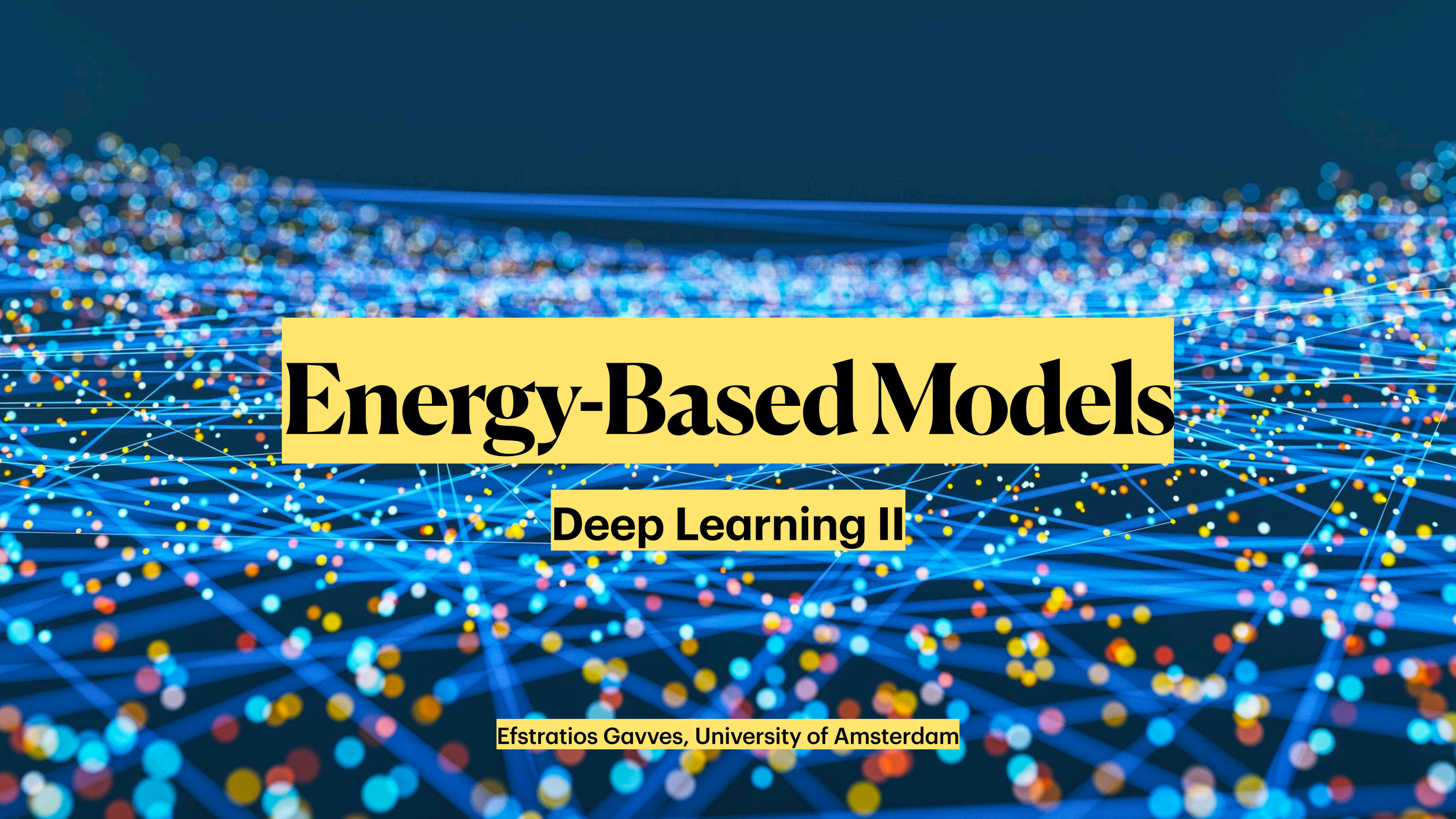


Energy-Based Models

Deep Learning II

Efstratios Gavves, University of Amsterdam

Lecture overview

- Brief intro to energy-based models
- Boltzmann Machines
- Restricted Boltzmann Machines
- Deep Boltzmann Machines
- Deep Belief Networks

Energy-based models for distributions

- Having a prescribed probability distribution, like a Gaussian, might be restrictive
- In the first lecture we saw we can define a distribution implicitly

$$p_\theta(\mathbf{x}) = \frac{f_\theta(\mathbf{x})}{Z_\theta}, \text{ where } Z_\theta = \int_{\mathbf{x}} f_\theta(\mathbf{x}) d\mathbf{x}$$

- Even more comfortable if we squeeze in an exponential

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(\mathbf{x}))}{Z_\theta} = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

where low energy means high probability

Energy-based models for distributions

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(\mathbf{x}))}{Z_\theta} = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

- Great flexibility in defining our energy function ⇒ any distribution can be learned 👍
- The normalisation constant creates lots of problems though
  - Sampling can be hard (the CDF approach requires another integral) 👎
  - Evaluating can be hard during training ⇒ Learning is hard 👎
  - No latent variables (in vanilla version) ⇒ no representation learning 👎

Key trick: ratio of likelihoods

- At a single point $\mathbf{x}_a$ it might be hard to evaluate an EBM
- But it is much simpler to compare which of two points has higher probability

$$\frac{p_\theta(\mathbf{x}_a)}{p_\theta(\mathbf{x}_b)} = \frac{\frac{\exp(-E_\theta(\mathbf{x}_a))}{Z_\theta}}{\frac{\exp(-E_\theta(\mathbf{x}_b))}{Z_\theta}} = \exp\Big(f_\theta(\mathbf{x}_a) - f_\theta(\mathbf{x}_b)\Big)$$

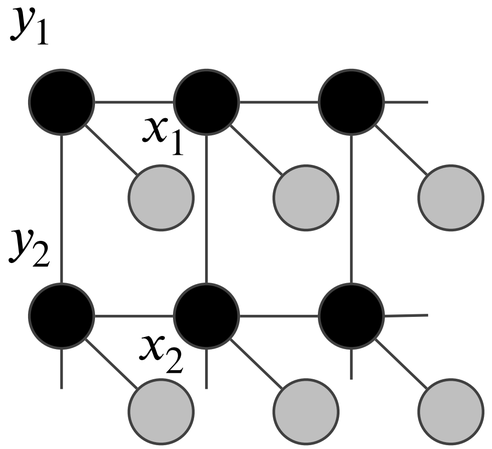
$$= \exp\Big(-E_\theta(\mathbf{x}_a) + E_\theta(\mathbf{x}_b)\Big)$$

- No normalisation constant anymore

Examples of energy models

- The Ising model is an energy-based model

$$p_\theta(\mathbf{x}, \mathbf{y}) = \frac{1}{Z_\theta} \exp\Big(\sum_i \psi_\theta(x_i) + \sum_{i,j} \psi_\theta(y_i, y_j) \Big)$$



- Product of experts, similar to an 'AND' function

$$p_{\theta,\phi,\omega}(\mathbf{x}) = \frac{1}{Z_{\theta,\phi,\omega}(\mathbf{x})} q_\theta(\mathbf{x}) r_\phi(\mathbf{x}) s_\omega(\mathbf{x})$$

- Restricted/Deep Boltzmann Machines
- Deep Belief Networks

Training any energy model

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(\mathbf{x}))}{Z_\theta} = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

- Maximizing log-likelihood

$$\frac{1}{N}\sum_i \log p_\theta(\mathbf{x}^{(i)}) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\Big[\log p_\theta(\mathbf{x})\Big]$$

$$= -\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\Big[E_\theta(\mathbf{x})\Big] + \log Z_\theta$$

where $\log Z_\theta = \log \sum_{\mathbf{x}} \exp(-E_\theta(\mathbf{x}))$ and $p_{data}(\mathbf{x})$ is the empirical data distribution

- The Z_θ is independent of the empirical data distribution (because of marginalisation)

Taking gradients

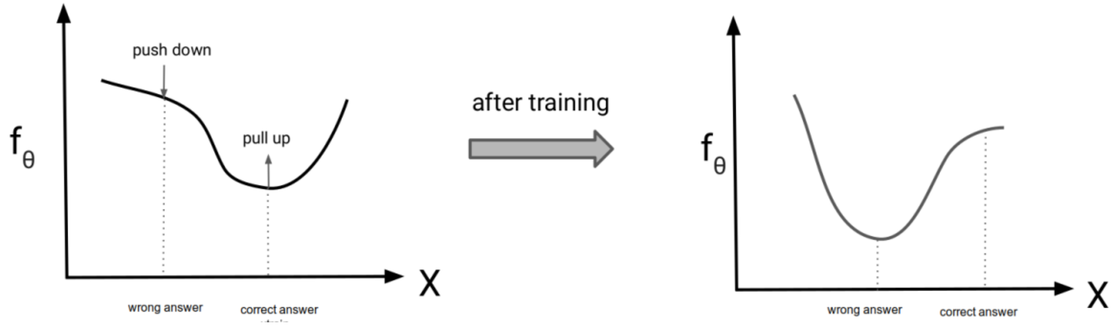
- Focusing on a single data point

$$\frac{d \log p_\theta(\mathbf{x})}{d\theta} = -\frac{dE_\theta(\mathbf{x})}{d\theta} - \frac{d \log Z_\theta}{d\theta}$$

$$= -\frac{dE_\theta(\mathbf{x})}{d\theta} - \frac{1}{Z_\theta}\frac{dZ_\theta}{d\theta}$$

$$= -\frac{dE_\theta(\mathbf{x})}{d\theta} + \sum_{\mathbf{x}'} \underbrace{\frac{1}{Z_\theta} \exp\big(-E_\theta(\mathbf{x}')\big)}_{p_\theta(\mathbf{x})} \frac{dE_\theta(\mathbf{x}')}{d\theta}$$

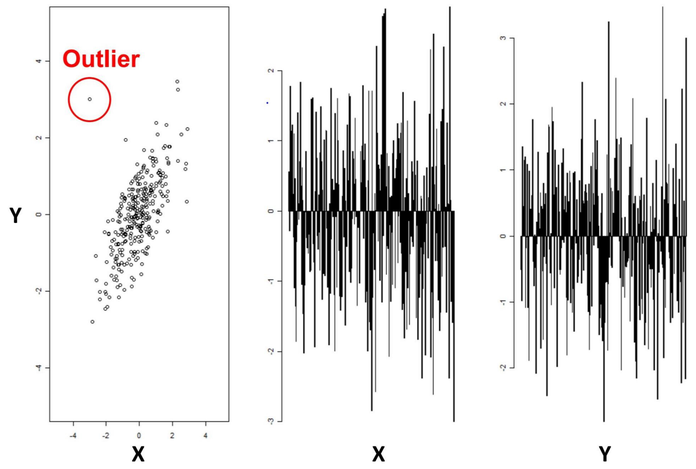
$$= -\frac{dE_\theta(\mathbf{x})}{d\theta} + \mathbb{E}_{\mathbf{x}' \sim p_\theta(\mathbf{x})}\left[\frac{dE_\theta(\mathbf{x}')}{d\theta}\right]$$



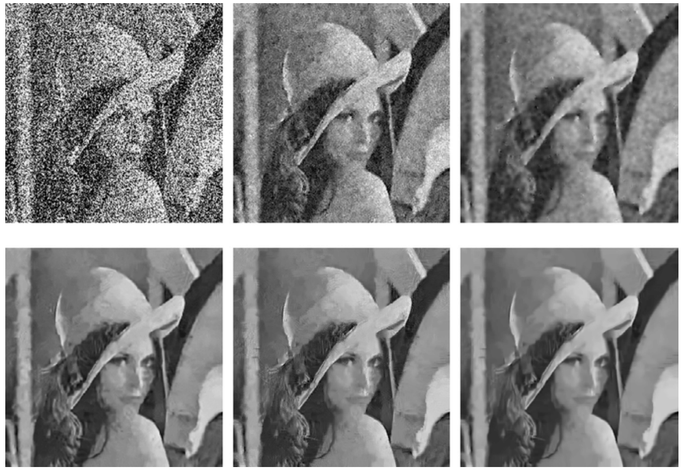
- Note: the expectation now is not wrt the empirical data distribution but the learned distribution by the model

Applications beyond generation

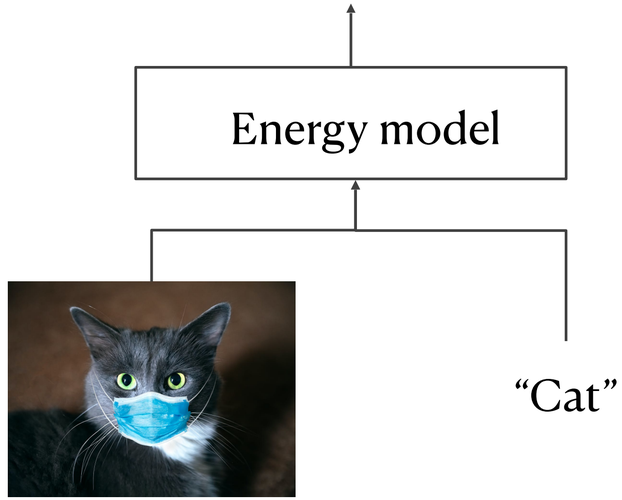
Anomaly detection



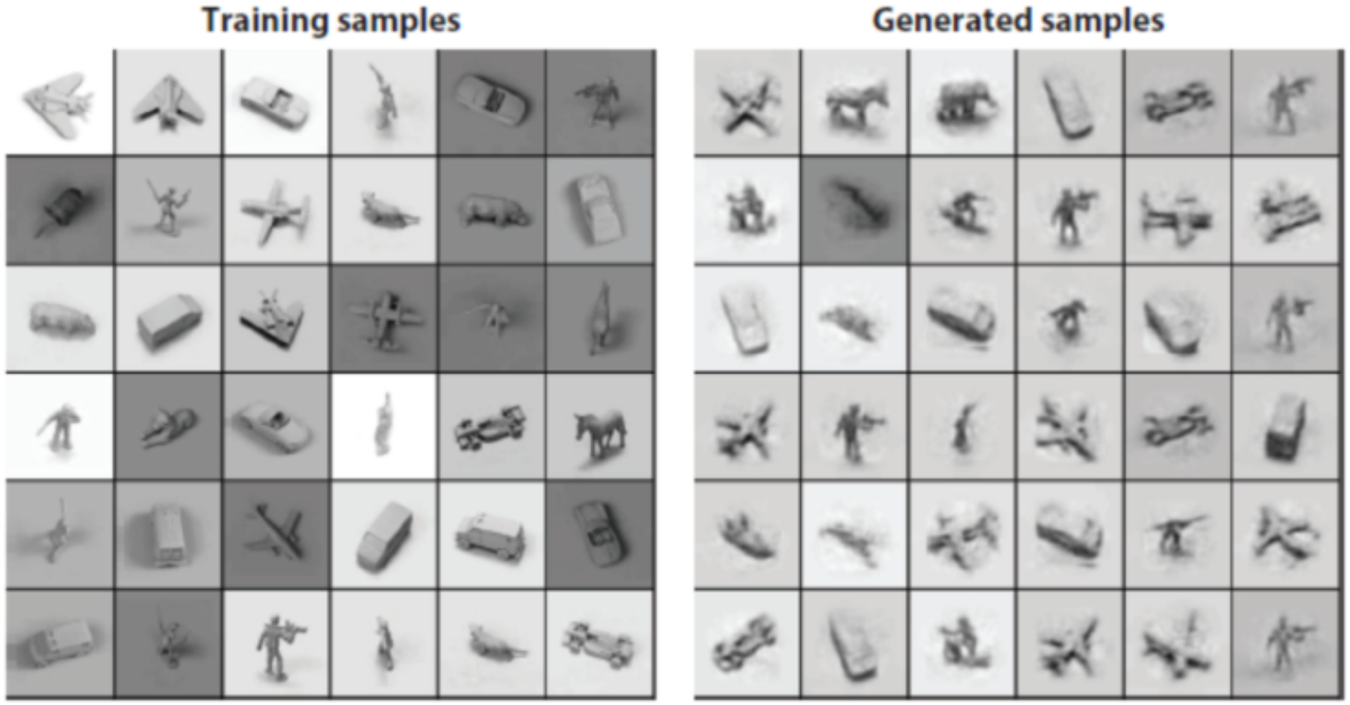
Denoising/restoration



Classification



Boltzmann machines



Boltzmann Machines

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(\mathbf{x}))}{Z_\theta} = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

- The energy function is defined as

$$E_\theta(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x} + \mathbf{b}^T\mathbf{x} \quad \text{with } \mathbf{x} \in \{-1,1\}^d$$

- For $\mathbf{x}$ being an image with $256 \times 256 = 65{,}536$ dimensions
- We would need a parameter matrix $\mathbf{W}$ with more than 4 billion parameters (for a single layer)
- Clearly, not tractable for large dimensions
- Assuming we already somehow had the $p_\theta(\mathbf{x})$, then we can sample using Gibbs sampling

Initialization: Initialize $\mathbf{x}^{(0)} \in \mathcal{R}^D$ and number of samples N

- **for** $i = 0$ to $N-1$ **do**
- $x_1^{(i+1)} \sim p(x_1|x_2^{(i)}, x_3^{(i)}, ..., x_D^{(i)})$
- $x_2^{(i+1)} \sim p(x_2|x_1^{(i+1)}, x_3^{(i)}, ..., x_D^{(i)})$
- $\vdots$
- $x_j^{(i+1)} \sim p(x_j|x_1^{(i+1)}, x_2^{(i+1)}, ..., x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, ..., x_D^{(i)})$
- $\vdots$
- $x_D^{(i+1)} \sim p(x_D|x_1^{(i+1)}, x_2^{(i+1)}, ..., x_{D-1}^{(i+1)})$

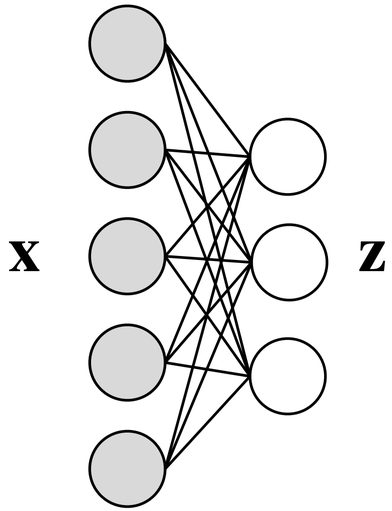
return $(\{\mathbf{x}^{(i)}\}_{i=0}^{N-1})$

Restricted Boltzmann Machines

- We can reduce dimensionality by having observed variables not interact directly with one another, but via latent (binary) variables instead

$$E_\theta(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{W}\mathbf{z} + \mathbf{b}^T\mathbf{x} + +\mathbf{c}^T\mathbf{z}$$

- The quadratic term captures correlations
- The unary terms capture priors, how likely the latent or the pixel is to be -1 or +1



Free energy

- Unfortunately, since we do not know the latents we must integrate them out to compute the likelihood

$$p_\theta(\mathbf{x}) = \frac{\sum_{\mathbf{z}} \exp(-E_\theta(\mathbf{x}, \mathbf{z}))}{Z_\theta}$$

- Which means we got no energy-based definition anymore (note the sum)
- We can rewrite this as the exponential of a free-energy

Free energy

- We can rewrite this as the exponential of a free-energy $F_\theta(\mathbf{x})$

$$p_\theta(\mathbf{x}) = \frac{\sum_{\mathbf{z}} \exp\big(-E_\theta(\mathbf{x}, \mathbf{z})\big)}{Z_\theta} = \frac{\exp\big(-F_\theta(\mathbf{x})\big)}{Z_\theta}$$

For which we have

$$\exp(-F_\theta(\mathbf{x})) = \sum_{\mathbf{z}} \exp(-E_\theta(\mathbf{x}, \mathbf{z})) \Rightarrow$$

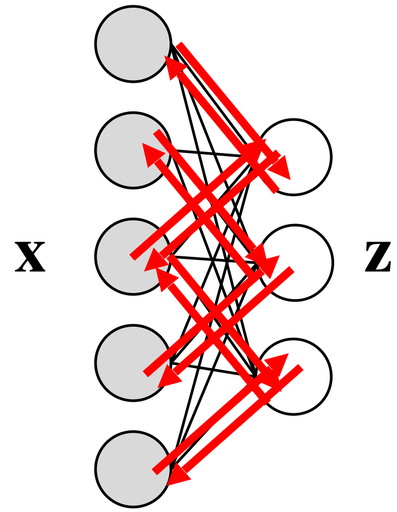
$$F_\theta(\mathbf{x}) = -\log\left(\sum_{\mathbf{z}} \exp(-E_\theta(\mathbf{x}, \mathbf{z}))\right)$$

RBMs are not feed-forward!

- There is no 'arrow of compute' in the energy (and the free-energy)

$$E_\theta(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{W}\mathbf{z} + \mathbf{b}^T\mathbf{x} + +\mathbf{c}^T\mathbf{z}$$

- The latents $\mathbf{z}$ depend on the observed variables $\mathbf{x}$
- And the observed variables $\mathbf{x}$ depend on the latents $\mathbf{z}$
- Information flows in both directions, forward and backward



Probabilistic model for the RBM

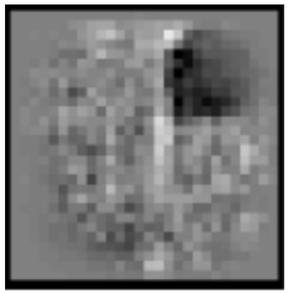
- Considering conditional independence given latent or observed variables

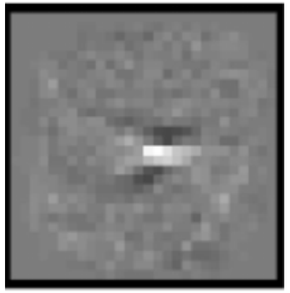
$$p_\theta(\mathbf{x}\,|\,\mathbf{z}) = \prod_j p_\theta(x_j\,|\,\mathbf{z})$$

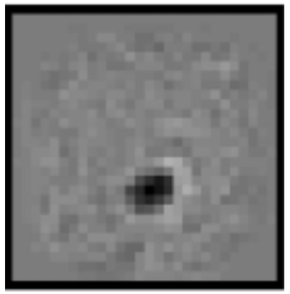
where $p_\theta(z_j\,|\,\mathbf{x}) = \sigma(\mathbf{W}_{j,.}\mathbf{x} + b_j)$, $p_\theta(x_j\,|\,\mathbf{z}) = \sigma(\mathbf{z}^T\mathbf{W}_{.,j} + c_j)$ and $\theta = \left\{W, b, c\right\}$

- Since $\mathbf{x}$ depends on $\mathbf{z}$ and $\mathbf{z}$ on $\mathbf{x}$, we must iterate between them

$$\mathbf{z}^{(1)} \sim p_\theta(\mathbf{z}\,|\,\mathbf{x}) \rightarrow \mathbf{x}^{(1)} \sim p_\theta(\mathbf{x}\,|\,\mathbf{z}^{(1)}) \rightarrow \mathbf{z}^{(2)} \sim p_\theta(\mathbf{z}\,|\,\mathbf{x}^{(1)}) \rightarrow \ldots$$







RBM gradients

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(\mathbf{x}))}{Z_\theta} = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta} \Rightarrow$$

$$\frac{d \log p_\theta(\mathbf{x})}{d\theta} = \frac{d \log \sum_{\mathbf{z}} \exp\big(-E_\theta(\mathbf{x}, \mathbf{z})\big)}{d\theta} - \frac{d \log \sum_{\mathbf{x}', \mathbf{z}} \exp\big(-E_\theta(\mathbf{x}', \mathbf{z})\big)}{d\theta}$$

$$= \frac{1}{\sum_{\mathbf{z}} \exp\big(-E_\theta(\mathbf{x}, \mathbf{z})\big)} \frac{d}{d\theta} \sum_{\mathbf{z}} \exp\big(-E_\theta(\mathbf{x}, \mathbf{z})\big) - \frac{1}{\sum_{\mathbf{x}', \mathbf{z}} \exp\big(-E_\theta(\mathbf{x}', \mathbf{z})\big)} \frac{d}{d\theta} \sum_{\mathbf{x}', \mathbf{z}} \exp\big(-E_\theta(\mathbf{x}', \mathbf{z})\big)$$

$$= -\sum_{\mathbf{z}} \underbrace{\frac{1}{\sum_{\mathbf{z}} \exp\big(-E_\theta(\mathbf{x}, \mathbf{z})\big)} \exp\big(-E_\theta(\mathbf{x}, \mathbf{z})\big)}_{p_\theta(\mathbf{z}|\mathbf{x})} \frac{dE_\theta(\mathbf{x}, \mathbf{z})}{d\theta} + \sum_{\mathbf{x}', \mathbf{z}} \underbrace{\frac{1}{\sum_{\mathbf{x}', \mathbf{z}} \exp\big(-E_\theta(\mathbf{x}', \mathbf{z})\big)} \exp\big(-E_\theta(\mathbf{x}', \mathbf{z})\big)}_{p_\theta(\mathbf{x}', \mathbf{z})} \frac{dE_\theta(\mathbf{x}', \mathbf{z})}{d\theta}$$

$$= -\mathbb{E}_{\mathbf{z} \sim p_\theta(\mathbf{z}|\mathbf{x})}\left[\frac{dE_\theta(\mathbf{x}, \mathbf{z})}{d\theta}\right] + \mathbb{E}_{\mathbf{x}, \mathbf{z} \sim p_\theta(\mathbf{x}', \mathbf{z})}\left[\frac{dE_\theta(\mathbf{x}', \mathbf{z})}{d\theta}\right]$$

RBM gradients

$$\frac{d \log p_\theta(\mathbf{x})}{d\theta} = -\mathbb{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x})}\left[\frac{dE_\theta(\mathbf{x},\mathbf{z})}{d\theta}\right] + \mathbb{E}_{\mathbf{x},\mathbf{z}\sim p_\theta(\mathbf{x}',\mathbf{z})}\left[\frac{dE_\theta(\mathbf{x}',\mathbf{z})}{d\theta}\right]$$

- We can easily evaluate the first term either integrating over $\mathbf{z}$
- Evaluating the second term wrt $\mathbf{x}, \mathbf{z} \sim p_\theta(\mathbf{x}', \mathbf{z})$ is much bigger problem
- Also because sampling in energy-based models is not straightforward

Gibbs sampling of the partition function

- We can approximate $\mathbb{E}_{\mathbf{x},\mathbf{z}\sim p_\theta(\mathbf{x}',\mathbf{z})}\left[\frac{dE_\theta(\mathbf{x}',\mathbf{z})}{d\theta}\right]$ by MCMC Gibbs sampling
- Convergence after many rounds
- Unfortunately, it may take lots of rounds ("infinite") and that just for a single evaluation
- Not only theoretical (due to the integral) but also computational intractability

Initialization: Initialize $\mathbf{x}^{(0)} \in \mathcal{R}^D$ and number of samples N

- **for** $i = 0$ to $N-1$ **do**
- $x_1^{(i+1)} \sim p(x_1|x_2^{(i)}, x_3^{(i)}, ..., x_D^{(i)})$
- $x_2^{(i+1)} \sim p(x_2|x_1^{(i+1)}, x_3^{(i)}, ..., x_D^{(i)})$
- $\vdots$
- $x_j^{(i+1)} \sim p(x_j|x_1^{(i+1)}, x_2^{(i+1)}, ..., x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, ..., x_D^{(i)})$
- $\vdots$
- $x_D^{(i+1)} \sim p(x_D|x_1^{(i+1)}, x_2^{(i+1)}, ..., x_{D-1}^{(i+1)})$

return $(\{\mathbf{x}^{(i)}\}_{i=0}^{N-1})$

Contrastive Divergence

- Maximizing likelihood is akin to minimising the KL between the true data distribution and the learned one

$$\mathrm{KL}(p_{data}\|p_\theta) = \int p_{data} \log\left(\frac{p_{data}}{p_\theta}\right) dp_{data} = \cancel{\int p_{data} \log p_{data} dp_{data}} - \int p_{data} \log p_\theta dp_{data}$$

- Ideally, after n MCMC steps we want our intermediate model $p_\theta^{(n)}$ to be as close to the true data distribution p_{data} as our 'optimal' learned one in the limit $p_\theta^{\infty} = p_\theta$

$$\mathrm{CD}^{(n)} = \mathrm{KL}(p_{data}\|p_\theta^{(\infty)}) - \mathrm{KL}(p_\theta^{(n)}\|p_\theta^{(\infty)})$$

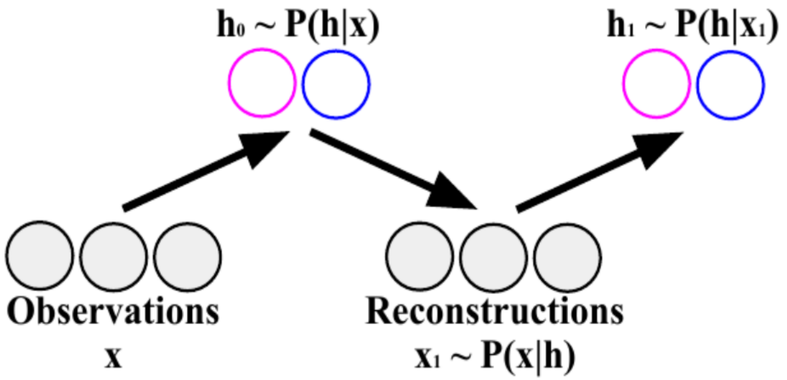
- This is the contrastive learning objective

Hinton, Training Products of Experts by Minimizing Contrastive Divergence, Neural Computation, 2002

Contrastive Diverge intuition

- Contrastive divergence removes the requirement to sample from the converged model at infinity
- Make sure after n (often 1 is enough) sampling steps the model is not far from data distribution
- This looks like 'minimising reconstruction error'
- Due to conditional independences we can do parallel computations and loop over dimensions

1. Sample data point $\mathbf{x}$
2. Compute the posterior $p(\boldsymbol{v} \mid \boldsymbol{x})$
3. Sample latents $\boldsymbol{v} \sim p(\boldsymbol{v} \mid \boldsymbol{x})$
4. Compute the conditional $p(\boldsymbol{x} \mid \boldsymbol{v})$
5. Sample from $\boldsymbol{x}' \sim p(\boldsymbol{x} \mid \boldsymbol{v})$
6. Minimise difference using $\boldsymbol{x}$, $\boldsymbol{x}'$



Carreira-Perpinan and Hinton, 2005

Contrastive Divergence gradients

- The CD (not ML) gradients are

$$\frac{d\mathrm{CD}^{(n)}(\mathbf{x})}{d\theta} = -\mathbb{E}_{\mathbf{x}\sim p_{data}}\left[\frac{dE_\theta(\mathbf{x})}{d\theta}\right] + \mathbb{E}_{\mathbf{x}'\sim p_\theta^{(n)}}\left[\frac{dE_\theta(\mathbf{x}')}{d\theta}\right] + \underbrace{\frac{dp_\theta^{(n)}}{d\theta}\frac{dKL(p_\theta^{(n)}\|p^{(\infty)})}{dp_\theta^{(n)}}}_{\approx 0}$$

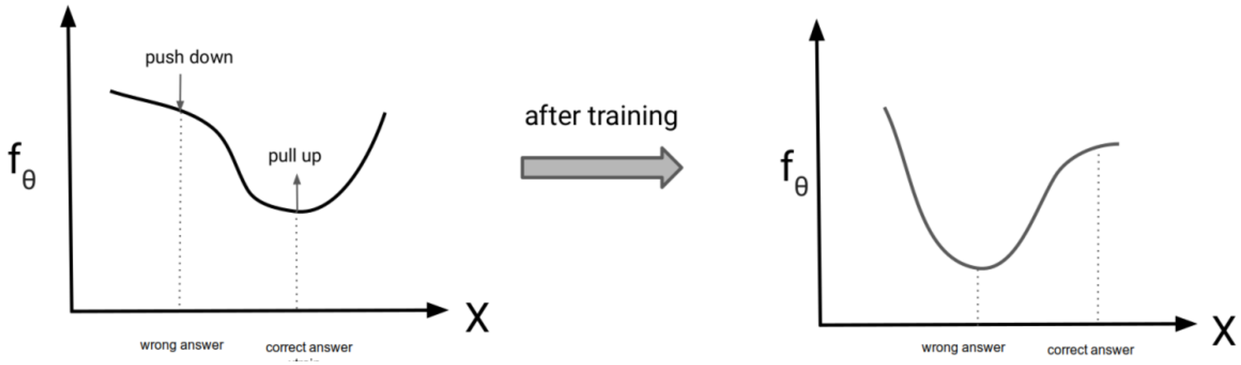
where $\mathbf{x}' \sim p_\theta^{(n)}$ is sampling from the learned model in the n-th Gibbs step

Hinton, *Training Products of Experts by Minimizing Contrastive Divergence*, Neural Computation, 2002

Interpreting the CD/RBM gradients

- Gradients push the nominator up in the observed points, while pulling down everywhere else by sampling from the learned model
- CD gradients approximates the gradients by k-steps Gibbs sampler

$$\frac{d \log p_\theta(\mathbf{x})}{d\theta} = -\frac{dE_\theta(\mathbf{x}, \mathbf{z}_0)}{d\theta} + \frac{dE_\theta(\mathbf{x}', \mathbf{z}_k)}{d\theta}$$



Ermon and Grover, Deep Generative Models

Carreira-Perpinan and Hinton, 2005

RBM sampling with MCMC

- We can sample with MCMC

1. Initialise $\mathbf{x}^{(0)}$ randomly
2. Add noise $\mathbf{x}' = \mathbf{x} + \text{noise}$
    1. If the energy of the new point is lower, $-E_\theta(\mathbf{x}') > -E_\theta(\mathbf{x})$, then accept and set: $\mathbf{x}^{t+1} = \mathbf{x}'$
    2. Otherwise set $\mathbf{x}^{t+1} = \mathbf{x}'$ with the probability of the likelihood ratio: $\exp(-E_\theta(\mathbf{x}') + E_\theta(\mathbf{x}^t))$
3. Go to step 2 and repeat

- Due to likelihood ratio no need to compute the partition function Z_θ

RBM sampling with Langevin sampling

- We can also use Langevin sampling to sample from a distribution using the score function $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$

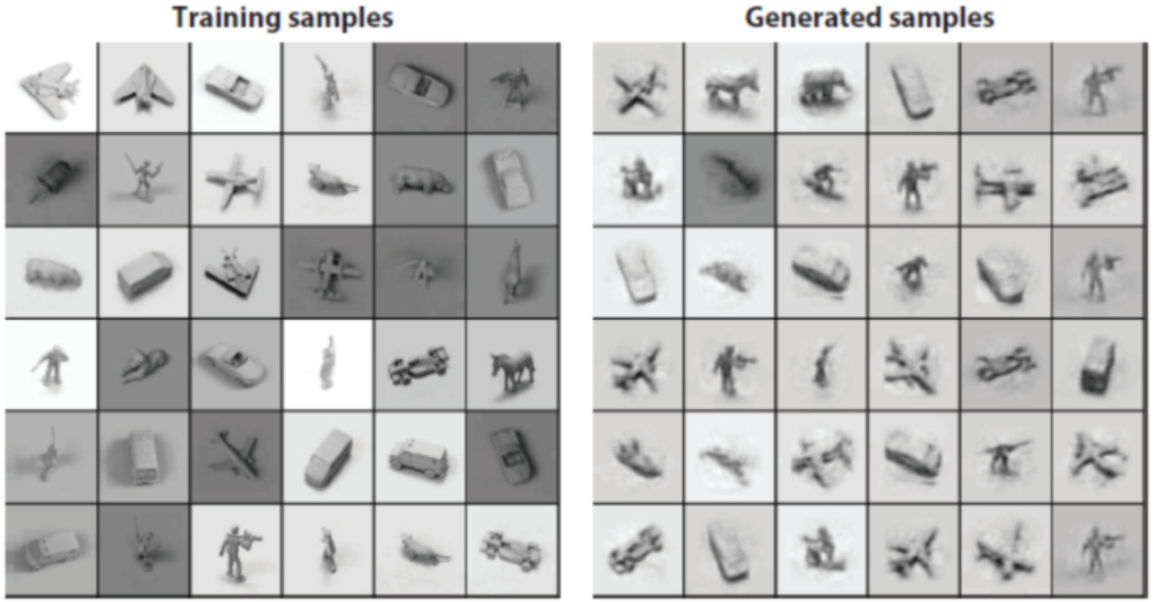
$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \lambda \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}^{(t)}) + \epsilon\sqrt{\lambda}, \text{ where } \epsilon \sim N(0,1) \text{ for } t = 1,...,T$$

- At $t = 0$ we start from a known and easy-to-sample prior distribution $\mathbf{x}^{(0)} \sim \pi(\mathbf{x})$
- For $\lambda \to 0$ and $T \to \infty$ we have that $\mathbf{x}^T \sim p_\theta(\mathbf{x})$
- Again no partition function involved because

$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E(\mathbf{x}) + \cancel{\nabla_{\mathbf{x}} Z_\theta} = -\nabla_{\mathbf{x}} E(\mathbf{x})$$

Using RBMs

- Some of the first models to show nice generations of images
- Use RBMs to pretrain networks for classification afterward



Deep Belief Network

- Deep Belief Networks are stacked RBM layers assuming layer-wise conditional independence

$$p(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2) = p(\mathbf{x} \mid \mathbf{z}_1) p(\mathbf{z}_1 \mid \mathbf{z}_2)$$

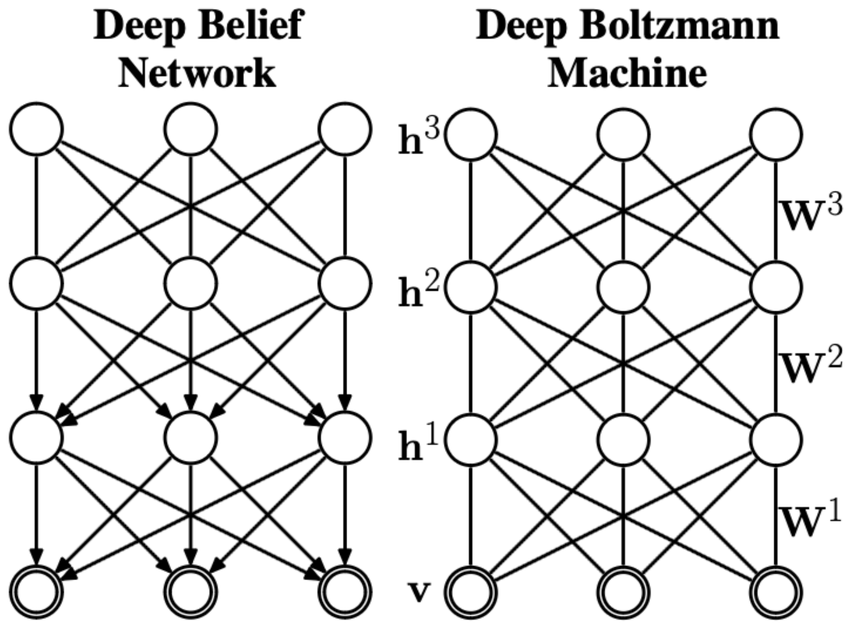
- Deep Belief Networks are directed models

$$\mathbf{x} \rightarrow \mathbf{z}_1 \rightarrow \mathbf{z}_2$$

- Dense layers with single forward flow

$$p_\theta(x_j \mid \mathbf{z}_1) = \sigma\left(\mathbf{W}_{j,.}\mathbf{z}_1 + c_j\right)$$

- Train per layer (not end-to-end) using contrastive divergence



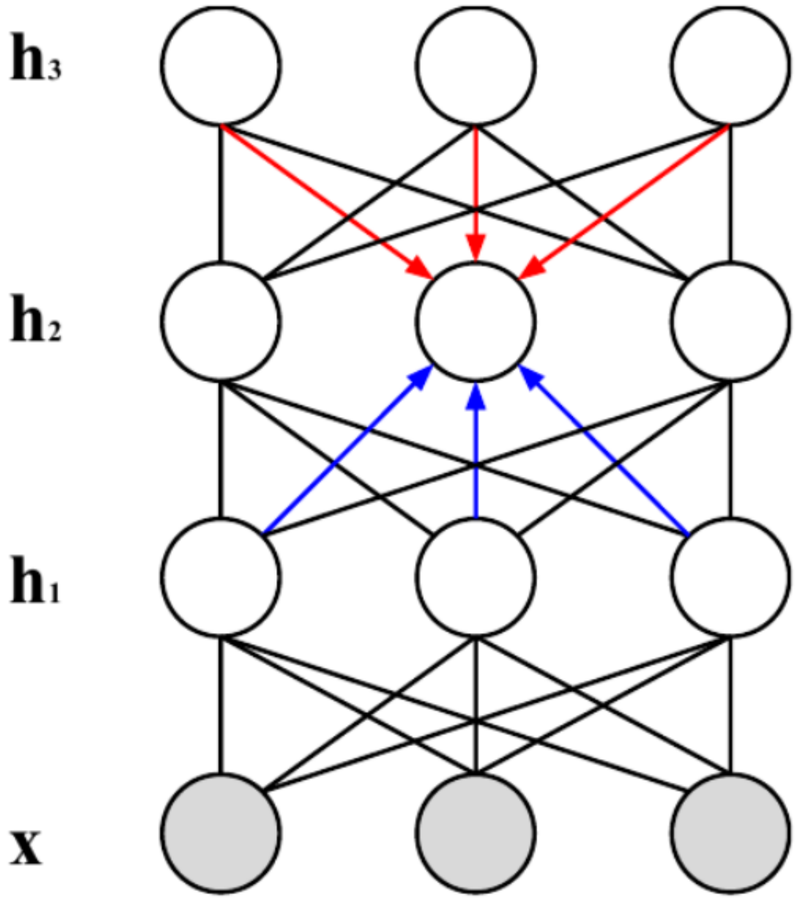
Deep Boltzmann machines

- Deep Boltmann Machines are stacked RBM layers from above and below layers →Markov model
- The energy function is

$$E_\theta(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) = \mathbf{x}^T \mathbf{W}_1 \mathbf{z}_1 + \mathbf{z}_1^T \mathbf{W}_2 \mathbf{z}_2 + \mathbf{z}_2^T \mathbf{W}_3 \mathbf{z}_3$$

- The per layer probabilistic model is

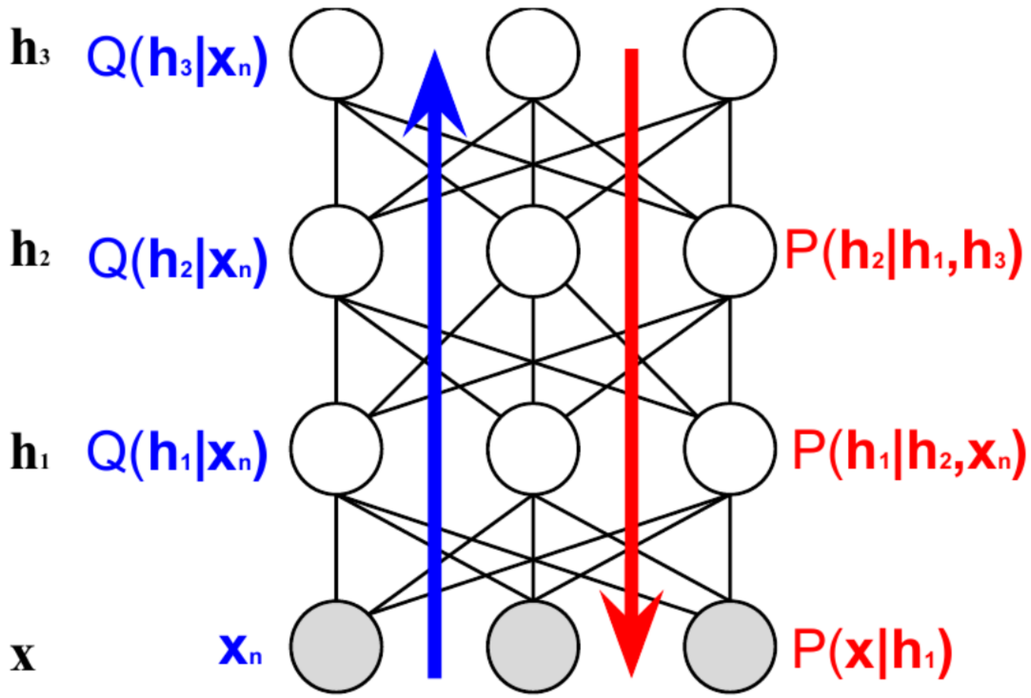
$$p(z_{2,k} \,|\, \mathbf{z}_1, \mathbf{z}_3) = \sigma\Big(\sum_j \mathbf{W}_{1,j,k} z_{1,j} + \sum_j \mathbf{W}_{3,j,k} z_{3,j} \Big)$$



Salakhutdinov and Hinton, 2009

Training deep Boltzmann machines

- Computing gradients is intractable
- Instead, complex training methodologies are used
  - like variational methods (mean-field)
  - or sampling methods



Summary

Brief intro to energy-based models
Boltzmann Machines
Restricted Boltzmann Machines
Deep Boltzmann Machines
Deep Belief Networks