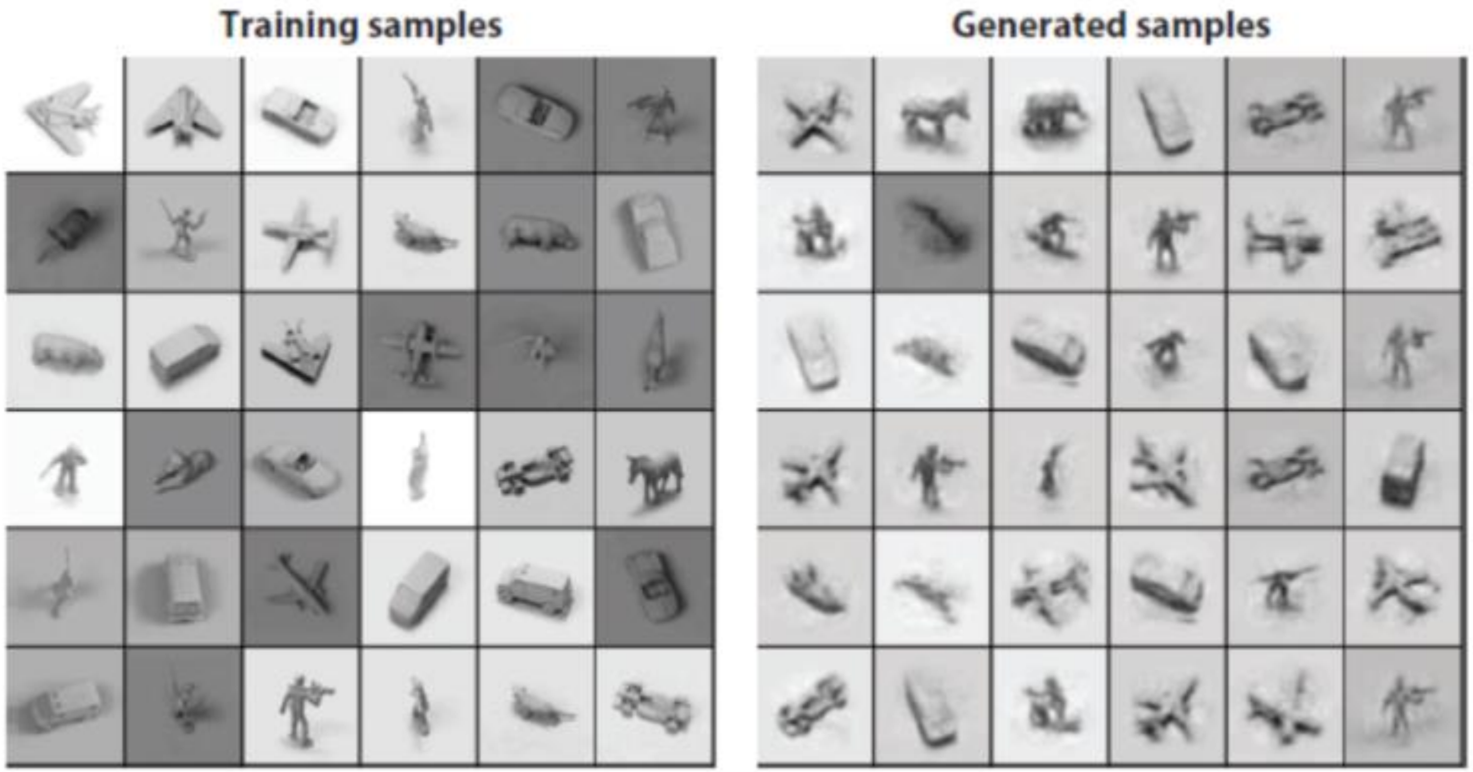
Boltzmann machines



From Hopfield to Boltzmann

- Hopfield networks minimize the quadratic energy function

$$E = -f_\theta(\boldsymbol{x}) = -\left(\sum_{i,j} w_{ij} x_i x_j + \sum_i b_i x_i\right)$$

- Boltzmann machines are stochastic Hopfield networks
- In Boltzmann machines the neuron response on activation a_i is

$$x_i = \begin{cases} +1 \text{ with probability } 1/1 + \exp(-2a_i) \\ -1 \qquad \text{otherwise} \end{cases}$$

- Gibbs sampling for pdf $p(\boldsymbol{x}) = \frac{1}{Z} \exp(\frac{1}{2} \boldsymbol{x}^T \boldsymbol{W} \boldsymbol{x})$

Restricted Boltzmann machines

- Boltzmann machines are too parameter heavy
  - For $\boldsymbol{x}$ with $256 \times 256 = 65536$ the $\boldsymbol{W}$ has 4.2 billion parameters
- Boltzmann machines learn no features
- Instead, add bottleneck latents $\boldsymbol{v}$

$$E = -f_\theta(\boldsymbol{x}) = -\left(\sum_{i,j} w_{ij} x_i v_j + \sum_i b_i x_i + \sum_j c_j v_j\right)$$

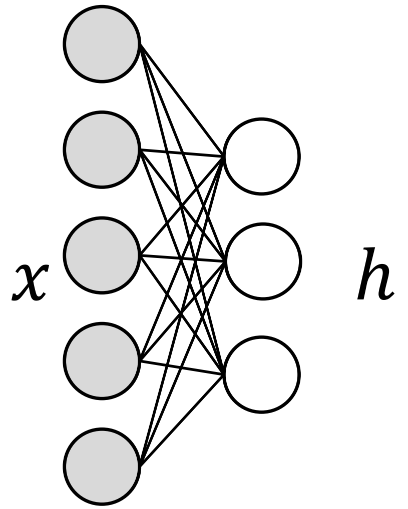
  - x_i and v_j are still binary variables in the original model
  - The quadratic term captures correlations
  - The unary terms capture priors: how likely is a (latent) pixel to be +1 or -1

Restricted Boltzmann Machines

- Energy function: $E(\boldsymbol{x}) = -\boldsymbol{x}^T \boldsymbol{W} \boldsymbol{v} - \boldsymbol{b}^T \boldsymbol{x} - \boldsymbol{c}^T \boldsymbol{v}$

$$p(\boldsymbol{x}) = \frac{1}{Z} \sum_{\boldsymbol{v}} \exp(-E(\boldsymbol{x}, \boldsymbol{v}))$$

  - Not in the form $\propto \exp(\mathbf{x})/Z$ because of the $\sum$



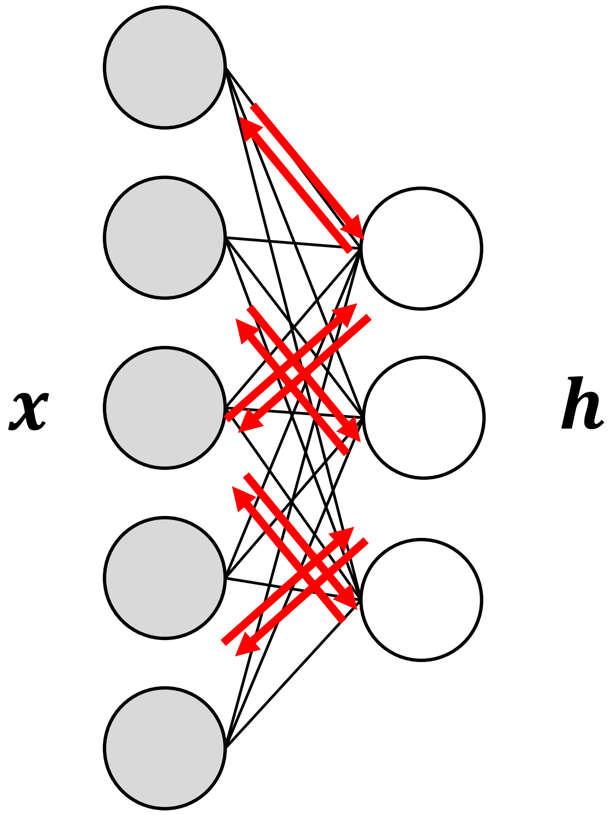
- Free energy function: $F(\boldsymbol{x}) = -\boldsymbol{b}^T \boldsymbol{x} - \sum_i \log \sum_{v_i} \exp(v_i(c_i + \boldsymbol{W}_i \boldsymbol{x}))$

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

$$Z = \sum_{\boldsymbol{x}} \exp(-F(\boldsymbol{x}))$$

Restricted Boltzmann Machines

- The $F(\boldsymbol{x})$ defines a bipartite graph with undirected connections
  - Information flows forward and backward



Restricted Boltzmann Machines

- The hidden variables v_j are independent conditioned on the visible variables

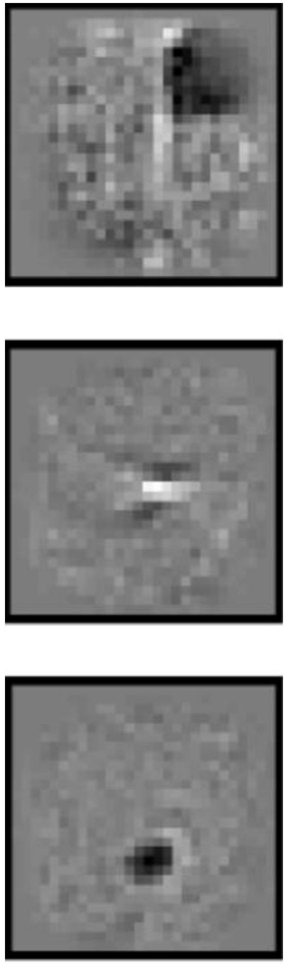
$$p(\boldsymbol{v}|\boldsymbol{x}) = \prod_j p(v_j|\boldsymbol{x}, \boldsymbol{\theta})$$

- The visible variables x_i are independent conditioned on the hidden variables

$$p(\boldsymbol{x}|\boldsymbol{v}) = \prod_i p(x_i|\boldsymbol{v}, \boldsymbol{\theta})$$

Training RBM conditional probabilities

Latent activations



- The conditional probabilities are defined as sigmoids

$$p(v_j|\boldsymbol{x},\boldsymbol{\theta}) = \sigma(\boldsymbol{W}_{.j}\boldsymbol{x} + b_j)$$
$$p(x_i|\boldsymbol{v},\boldsymbol{\theta}) = \sigma(\boldsymbol{v}^T\boldsymbol{W}_{i.} + c_i)$$

- Since RBMs are bidirectional ⇒ "Loop" between visible and latent

$$\boldsymbol{v}^{(1)} \sim \sigma(\boldsymbol{W}_{.j}\boldsymbol{x}^{(0)} + b_j) \Rightarrow$$
$$\boldsymbol{x}^{(1)} \sim \sigma(\boldsymbol{W}_{.j}\boldsymbol{v}^{(2)} + b_j) \Rightarrow$$
$$\boldsymbol{v}^{(2)} \sim \sigma(\boldsymbol{W}_{.j}\boldsymbol{x}^{(1)} + b_j) \Rightarrow \;\ldots$$

Training any energy model

- Maximizing log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{\mathrm{N}} \sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n) = \mathbb{E}_{p_0}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})]$$

- The expectation w.r.t. a pdf is equivalent to
  - sampling from the pdf and
  - then taking the average

$$\mathbb{E}_{\boldsymbol{x}\sim p_0}[\log p(\boldsymbol{x}|\boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{x}\sim p_0}[-E_{\boldsymbol{\theta}}(\boldsymbol{x})] - \log Z(\boldsymbol{\theta})$$

  - where $\log Z(\boldsymbol{\theta}) = \log \sum_{x'} \exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}'))$
  - and $p_0(\boldsymbol{x})$ is the data distribution

Taking gradients of any energy model

$$\frac{d}{d\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\frac{d}{\partial\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{d}{d\boldsymbol{\theta}}\log Z(\boldsymbol{\theta}) =$$

$$= -\frac{d}{\partial\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{1}{Z(\boldsymbol{\theta})}\frac{d}{d\boldsymbol{\theta}}Z(\boldsymbol{\theta})$$

$$= -\frac{d}{\partial\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x}) - \sum_{\boldsymbol{x}'}\frac{1}{Z(\boldsymbol{\theta})}\exp(-E_{\boldsymbol{\theta}}[\boldsymbol{x}'])\left(-\frac{d}{d\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$

$$= -\frac{d}{\partial\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x}) + \sum_{\boldsymbol{x}'}p_{\theta}(\boldsymbol{x}')\frac{d}{d\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x}')$$

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

Remember: $\sum p(x)\, f(x) = \mathbb{E}_{p(x)}[f(x)]$

$\int_x p(x)f(x)dx = \mathbb{E}_{p(x)}[f(x)]$

Taking gradients in an RBM

- For an RBM we must integrate out the latent variables

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{\mathrm{N}} \sum_n \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n) = \frac{1}{\mathrm{N}} \sum_n \log \sum_{\boldsymbol{v}} p_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{v})$$

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log \sum_{\boldsymbol{v}} p_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{v}) = -\mathbb{E}_{\boldsymbol{v} \sim p_{\boldsymbol{\theta}}(\boldsymbol{v} | \boldsymbol{x}_n)} \left[\frac{d}{d\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{v}) \right] + \mathbb{E}_{\boldsymbol{x}', \boldsymbol{v} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{v})} \left[\frac{d}{d\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}', \boldsymbol{v}) \right]$$

Taking gradients in an RBM

- And since for RBM $E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{v}) = -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x} - \boldsymbol{c}^T \boldsymbol{v}$

$$\frac{d}{d\boldsymbol{W}_{\boldsymbol{ij}}} E_{\boldsymbol{\theta}}(x_i, v_j) = -x_i v_j \Rightarrow$$

$$\frac{d\mathcal{L}}{d\boldsymbol{W}_{\boldsymbol{ij}}} = \mathbb{E}_{v \sim p_{\boldsymbol{\theta}}(\boldsymbol{v}|\boldsymbol{x}_n)}[x_i v_j] - \mathbb{E}_{\boldsymbol{x}', \boldsymbol{v} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{v})}[x_i v_j]$$

- Easy: substitute $\boldsymbol{x}_n$ and sum over $\boldsymbol{v}$
- Hard (normalization): sum over all 2^{m+d} combinations of images & latents
  - Intractable due to exponential complexity w.r.t. $m + d$
  - Evaluating and optimizing $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{v})$ takes a long time
  - If we had only the unnormalized part we would have no problem

Tackling intractability by sampling

- $\mathbb{E}_{\boldsymbol{x}',\boldsymbol{v}\sim p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{v})}\left[\frac{d}{d\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x}',\boldsymbol{v})\right]$ stands for an expectation
  - One can sample very many $\boldsymbol{x}',\boldsymbol{v}$ from $p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{v})$
  - Take average instead of computing analytically (Monte Carlo sampling)
- Question: how to even sample from a hard pdf?
  - Markov Chain Monte Carlo with Gibbs sampling
  - Convergence after many rounds

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

Sampling the normalizing constant

- We can rewrite the gradient as

$$\frac{d}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = -\mathbb{E}_0 \left[\frac{d}{\partial \boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}) \right] + \mathbb{E}_\infty \left[\frac{d}{\partial \boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}') \right]$$

  - $\mathbb{E}_0 \equiv E_{\boldsymbol{x} \sim p_0}$ means sampling from training data and average gradients
  - $\mathbb{E}_\infty \equiv E_{\boldsymbol{x}, \boldsymbol{v} \sim p_{\boldsymbol{\theta}}}$ means sampling from the model and average gradients

- Unfortunately, MCMC can be very slow → 2^{nd} source of intractability

Ergo, contrastive diverge learning

- To motivate contrastive divergence, we revisit maximum likelihood learning

$$\mathrm{KL}(p_0 \parallel p_\infty) = \int p_0 \log p_0 - \int p_0 \log p_\infty \propto -\int p_0 \log \mathrm{p}_\infty$$

- Contrastive divergence minimizes

$$\mathrm{CD}_n = \mathrm{KL}(p_0 \parallel p_\infty) - \mathrm{KL}(p_n \parallel p_\infty)$$

- Updates weights using CD_n gradients instead of ML gradients

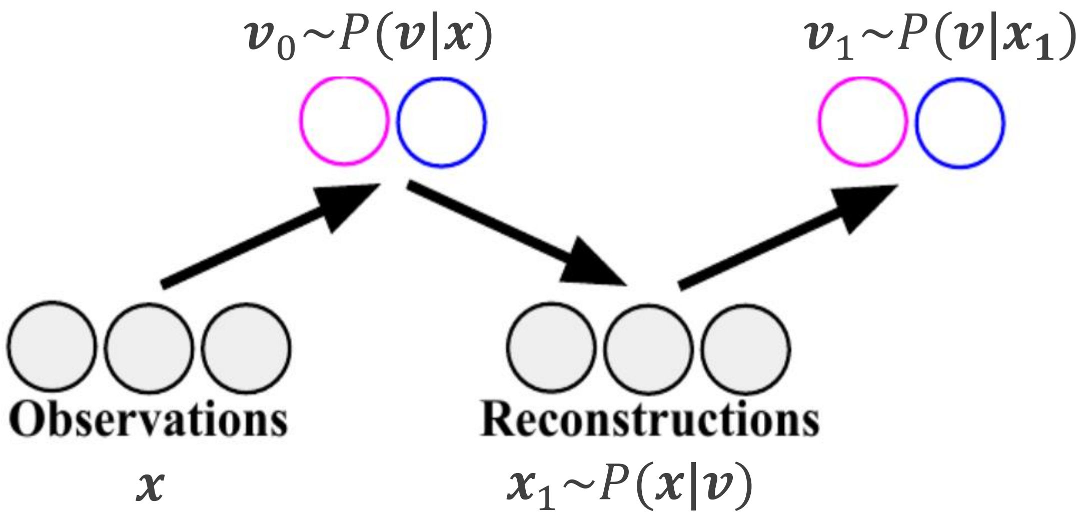
$$\frac{d}{\partial \boldsymbol{\theta}} \mathrm{CD}_n = -\mathbb{E}_0 \left[\frac{d}{\partial \boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}) \right] + \mathbb{E}_n \left[\frac{d}{\partial \boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}') \right] + \frac{d}{\partial \boldsymbol{\theta}} [\ldots]$$

  - where $\mathbb{E}_n$ is computed by sampling after n steps in the Markov Chain
  - The last term is small and can be ignored

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

Contrastive diverge learning: intuition

- Make sure after n sampling step not far from data distribution
  - Usually, one step only (n=1) is enough
  - Something similar to 'minimizing reconstruction error'
- Because of conditional independence of $\boldsymbol{x}|\boldsymbol{v}$ and $\boldsymbol{v}|\boldsymbol{x}$ → parallel computations
  - Sample a data point $\boldsymbol{x}$
  - Compute the posterior $\boldsymbol{p}(\boldsymbol{v}|\boldsymbol{x})$
  - Take sample of latents $\boldsymbol{v} \sim \boldsymbol{p}(\boldsymbol{v}|\boldsymbol{x})$
  - Compute the conditional $\boldsymbol{p}(\boldsymbol{x}|\boldsymbol{v})$
  - Sample from $\boldsymbol{x}' \sim \boldsymbol{p}(\boldsymbol{x}|\boldsymbol{v})$
  - Minimize difference using $\boldsymbol{x}, \boldsymbol{x}'$

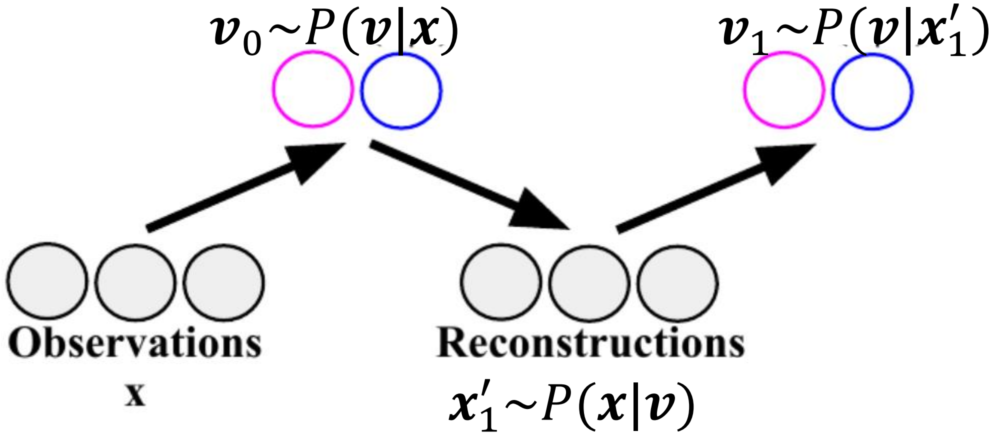


Contrastive divergence for RBMs

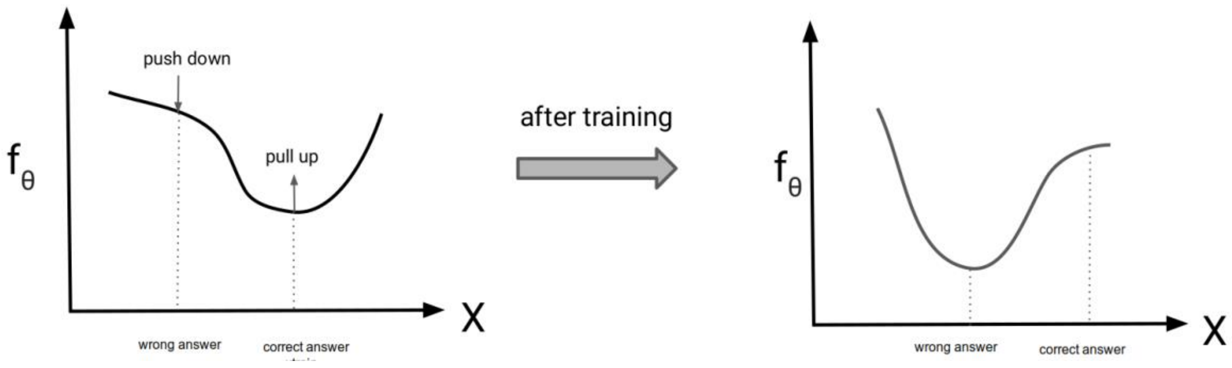
- Contrastive divergence approximates gradient by k-steps Gibbs sampler

$$\frac{d}{d\boldsymbol{\theta}} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) = -\frac{d}{d\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{v}_0) - \frac{d}{d\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}'_k, \boldsymbol{v}_k)$$

- Pushing the nominator up while pushing the denominator down



Hinton, 2002 *Carreira-Perpinan and Hinton, 2005*



[*Ermon and Grover, Deep Generative Models*]

How to sample? Markov Chain Monte Carlo

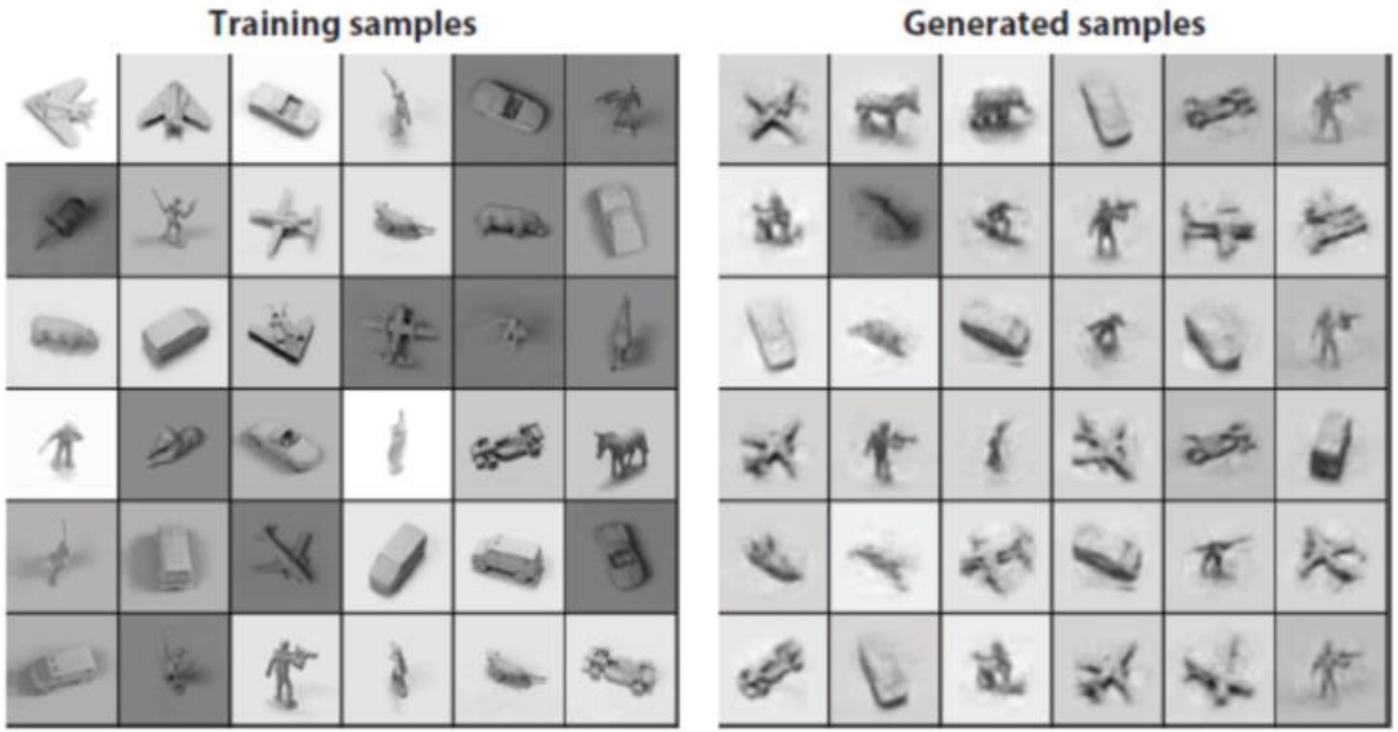
We want to sample an $\boldsymbol{x}$ from a pdf $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ with MCMC with Gibbs sampler

- Step 1. Initialize $\boldsymbol{x}^0$ randomly
- Step 2. Let $\widehat{\boldsymbol{x}} = \boldsymbol{x}^t + \text{noise}$
  - If $f_{\boldsymbol{\theta}}(\widehat{\boldsymbol{x}}) > f_{\boldsymbol{\theta}}(\boldsymbol{x}^{\mathrm{t}})$, set $\boldsymbol{x}^{t+1} = \widehat{\boldsymbol{x}}$
  - Otherwise $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t$ with probability $\frac{p(\widehat{\boldsymbol{x}})}{p(\boldsymbol{x}^t)} = \exp(f_{\boldsymbol{\theta}}(\widehat{\boldsymbol{x}}) - f_{\boldsymbol{\theta}}(\boldsymbol{x}^{\mathrm{t}}))$
- Go to step 2

- Because of the ratio of likelihoods → no $Z(\boldsymbol{\theta})$

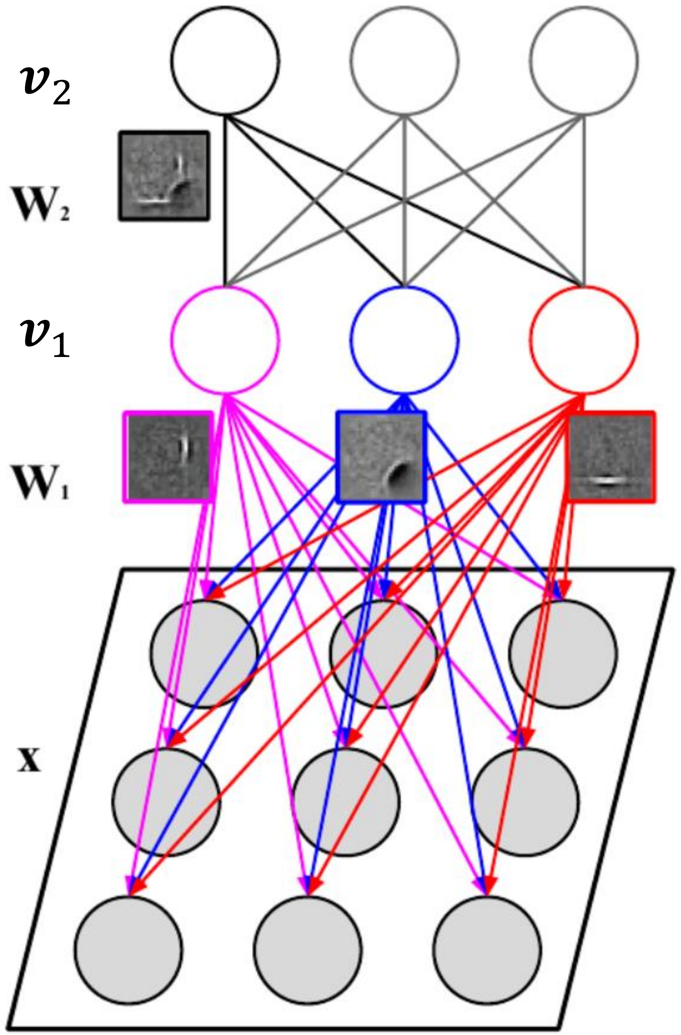
Using RBMs

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



Deep Belief Network

- Stack RBM layers assuming conditional independence
  $$p(\boldsymbol{x}, \boldsymbol{v}_1, \boldsymbol{v}_2) = p(\boldsymbol{x}|\boldsymbol{v}_1) \cdot p(\boldsymbol{v}_1|\boldsymbol{v}_2)$$
- Deep Belief Networks are directed models
- Dense layers with single forward flow
  - As RBM is directional: $p(x_i|\boldsymbol{v}, \boldsymbol{\theta}) = \sigma(\boldsymbol{W}_{.i}\boldsymbol{x} + c_i)$

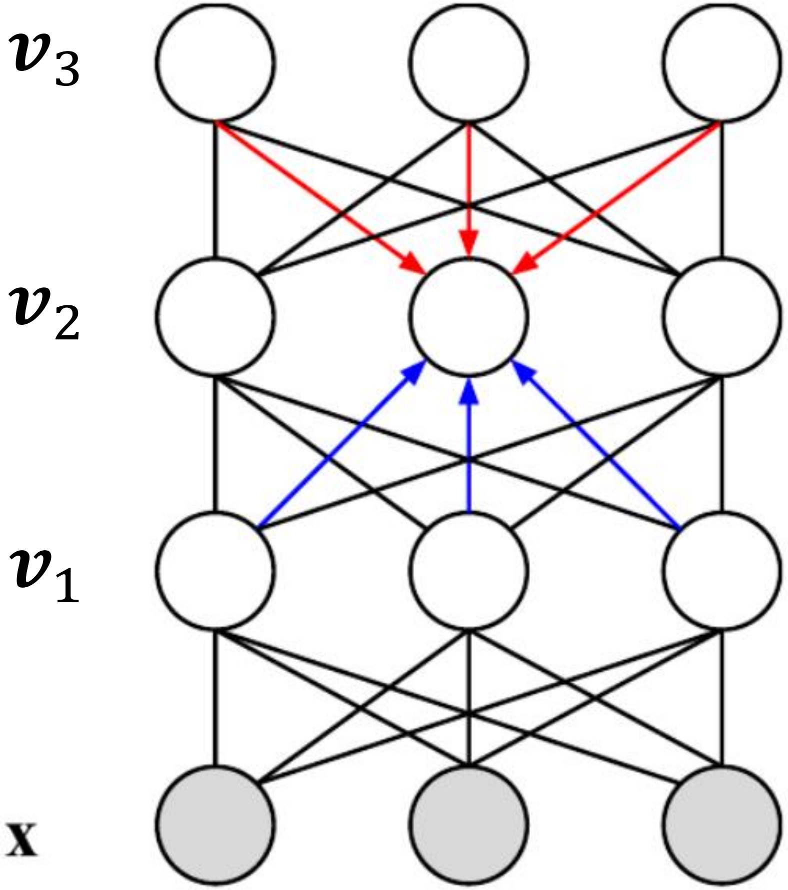


Deep Boltzmann machines

- Stacking RBM layers from above and below layers
  - Markov model
- Energy function

$$E(\boldsymbol{x}, \boldsymbol{v}_1, \boldsymbol{v}_2 | \boldsymbol{\theta}) = \boldsymbol{x}^T \boldsymbol{W}_1 \boldsymbol{v}_1 + \boldsymbol{v}_1^T \boldsymbol{W}_2 \boldsymbol{v}_2 + \boldsymbol{v}_2^T \boldsymbol{W}_3 \boldsymbol{v}_3$$

$$p(\boldsymbol{v}_2^k | \boldsymbol{v}_1, \boldsymbol{v}_3) = \sigma\left(\sum_j \boldsymbol{W}_1^{jk} \boldsymbol{v}_1^j + \sum_l \boldsymbol{W}_3^{kl} \boldsymbol{v}_3^k\right)$$



Training deep Boltzmann machines

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