Deep Boltzmann Machines

Pierre-Aurelien Gilliot

03/12/19

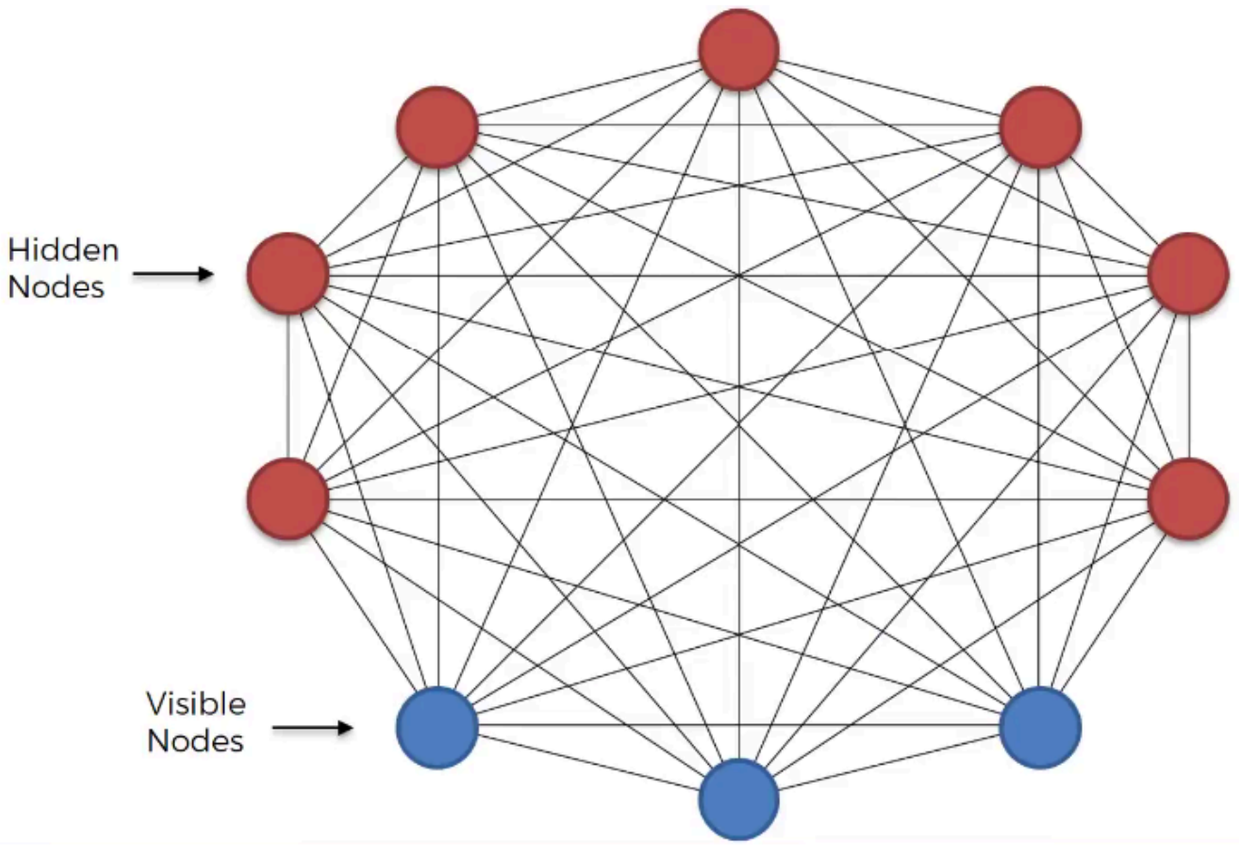
Overview

- Background
- Restricted Boltzmann Machines
- Deep (Restricted) Boltzmann Machines

Why should we care?

- Generative model
- Feature extracting technique
- State-of-the-art collaborative filtering 🤑🤑

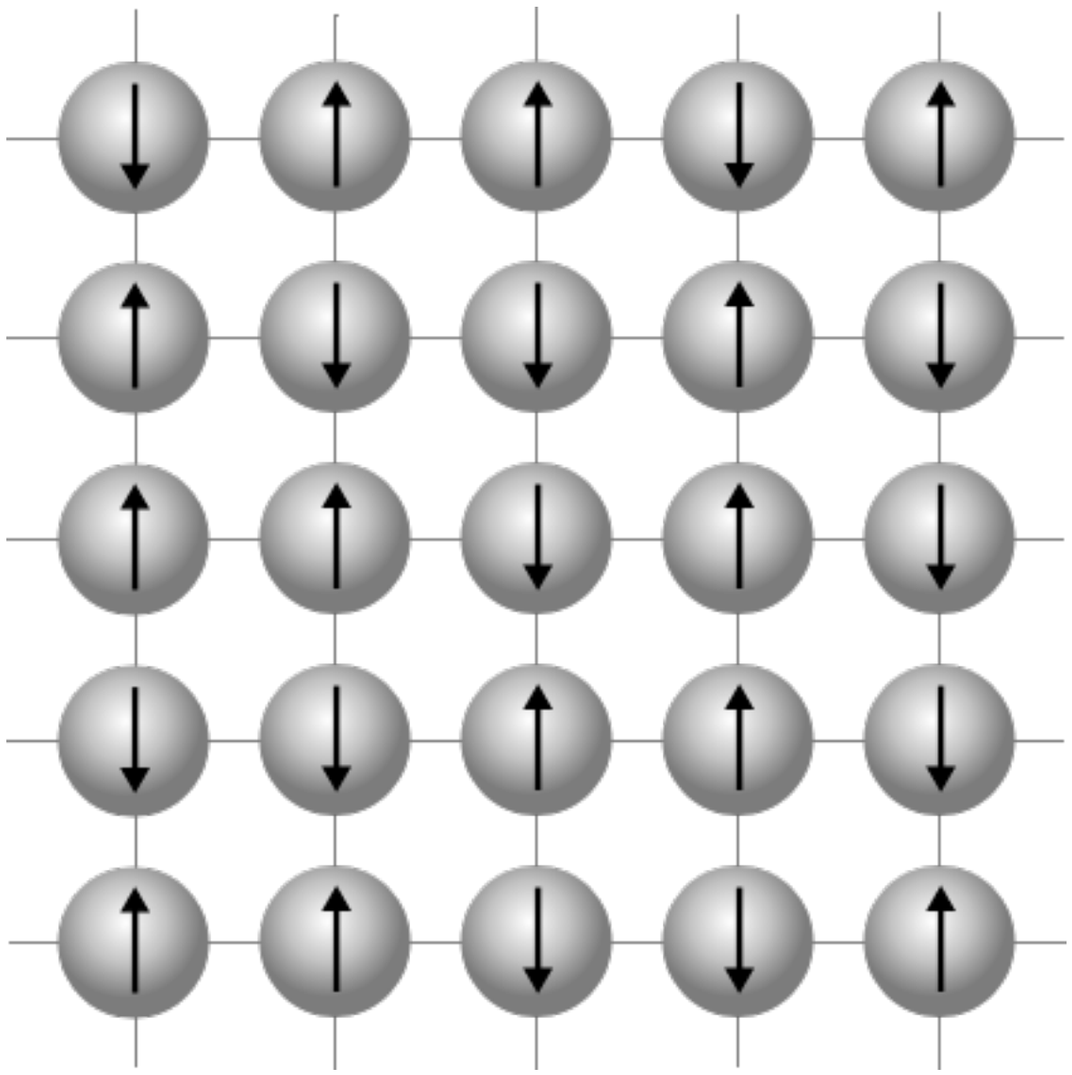
Boltzmann machines



- Graphical model
- $p(v)$, unsupervised
- Stochastic neural network
- Energy-based

Special case of graphical model, where the goal is to learn a probability distribution form a dataset, this being unsupervised.

Background- Ising model

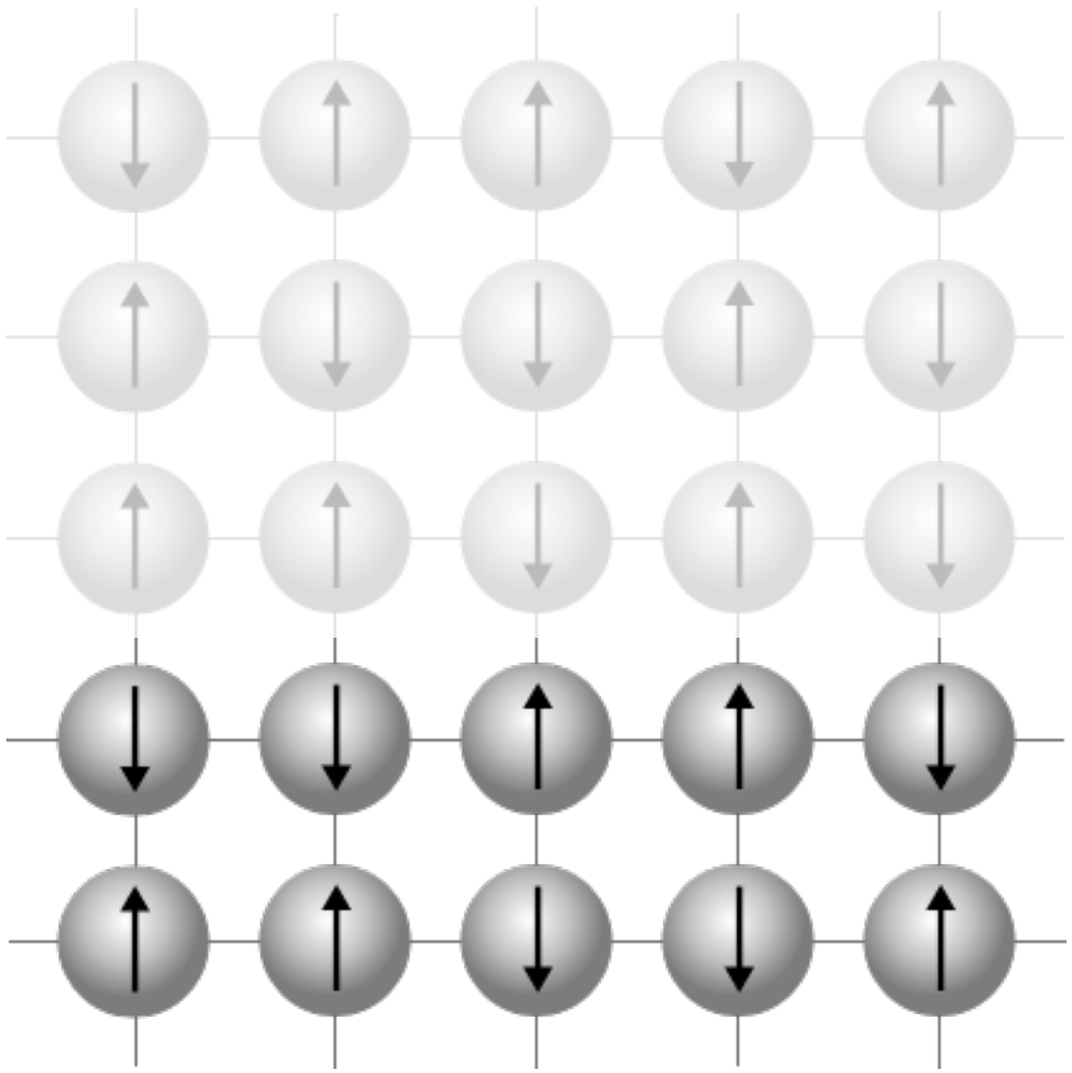


Spin configuration $\sigma = (\sigma_k)_\Lambda$

$$H(\sigma) = -\sum_{\langle i\ j\rangle} J_{ij}\sigma_i\sigma_j - \mu \sum_j h_j\sigma_j$$

$$P(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z} \qquad Z = \sum_\sigma e^{-\beta H(\sigma)}$$

Background- Ising model

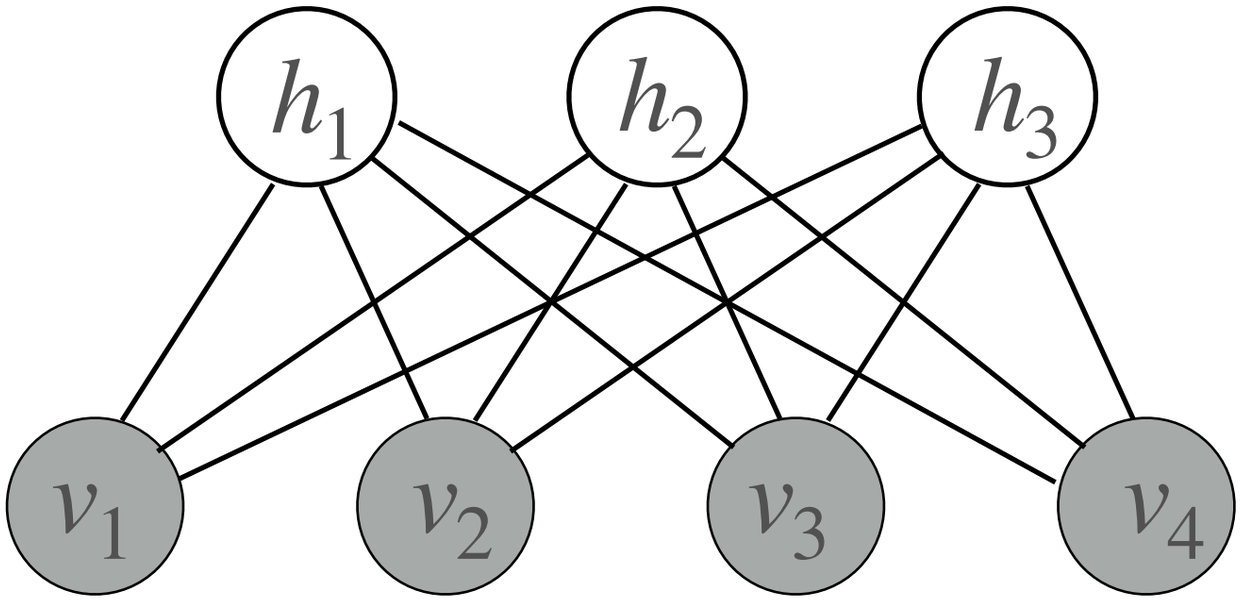


Spin configuration $\sigma = (\sigma_k)_\Lambda$

$$H(\sigma) = -\sum_{\langle i\ j\rangle} J_{ij}\sigma_i\sigma_j - \mu \sum_j h_j\sigma_j$$

$$P(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z} \qquad Z = \sum_\sigma e^{-\beta H(\sigma)}$$

Restricted Boltzmann Machine



Hidden Layer

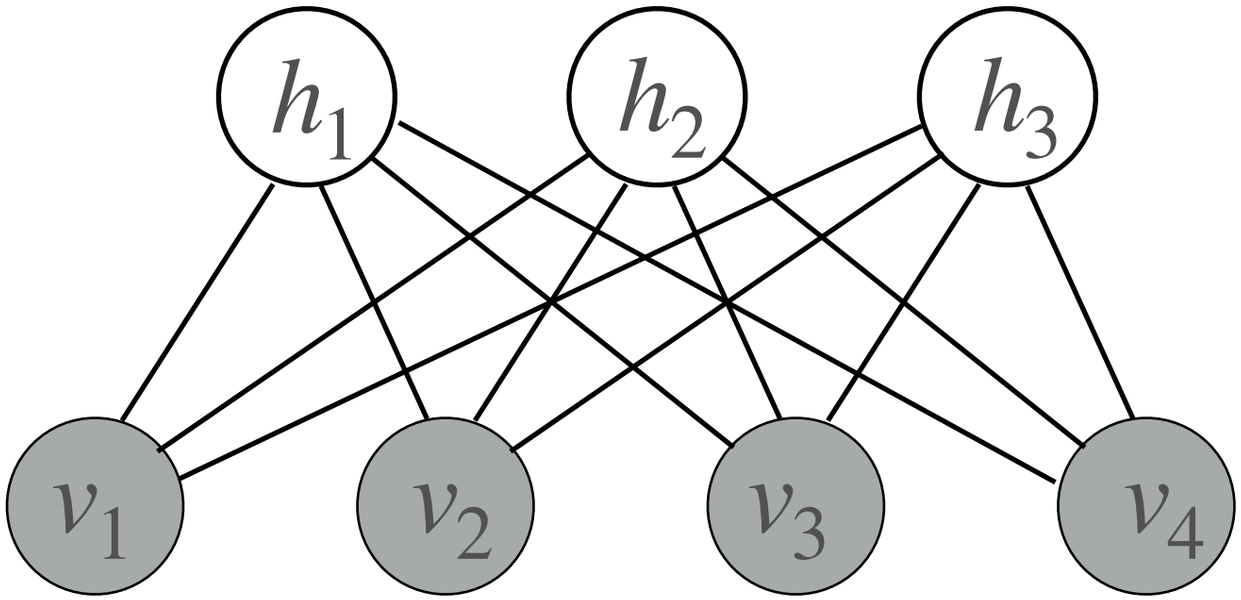
Input Layer

Energy of a configuration

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$$

$$= -a^{\mathrm{T}} v - b^{\mathrm{T}} h - v^{\mathrm{T}} W h$$

Restricted Boltzmann Machine



Hidden Layer

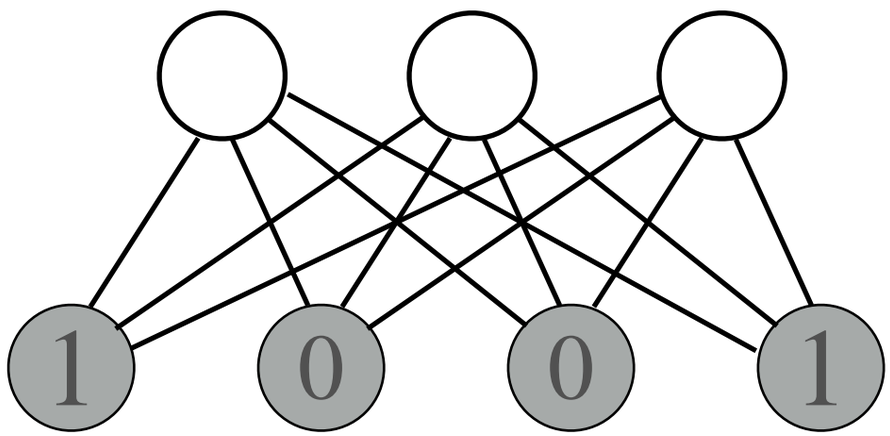
Input Layer

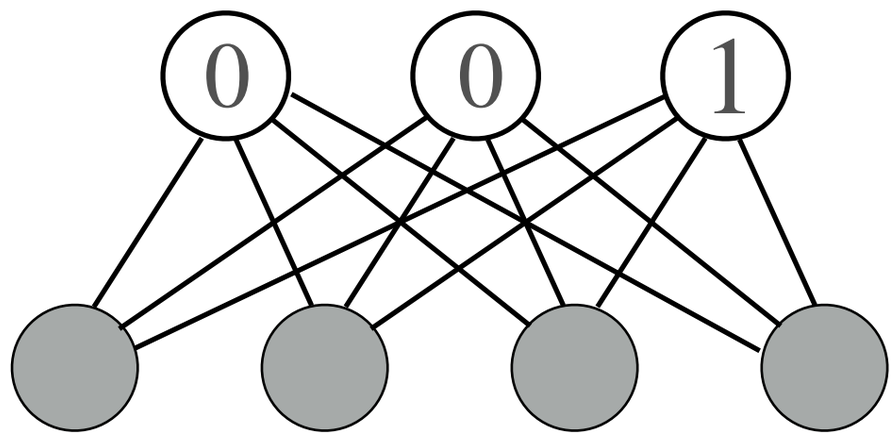
Probability of a configuration

$$p(\mathbf{v};\theta) = \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h};\theta))}{Z(\theta)}$$

$$= \frac{\exp(-\mathscr{F}(\mathbf{v}))}{Z(\theta)}$$

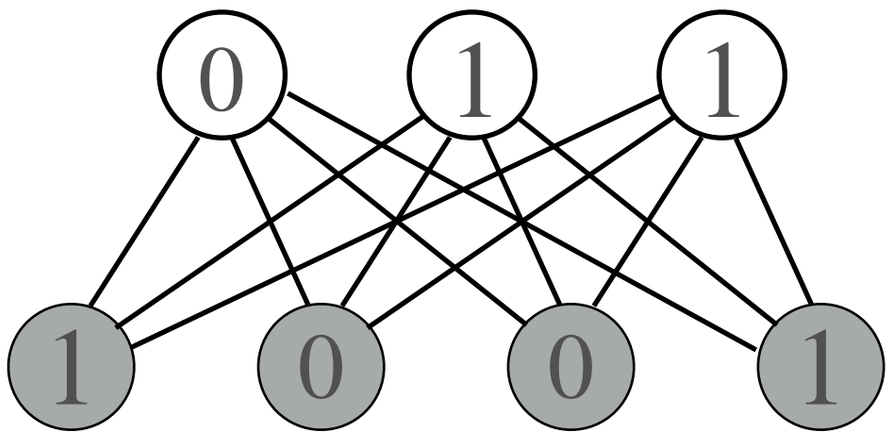
Conditional distributions

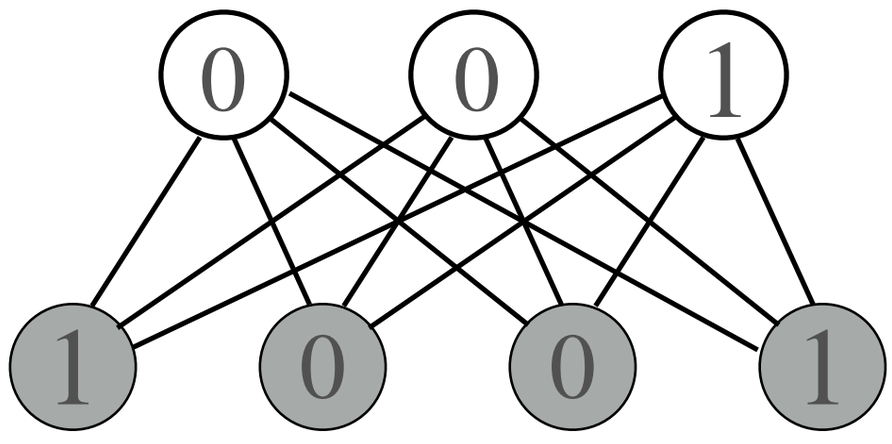




$$p(h_j = 1 \mid \mathbf{v}) = \sigma(b_j + \sum_{i=1}^{m} W_{ij} v_i)$$

$$p(v_j = 1 \mid \mathbf{h}) = \sigma(a_j + \sum_{i=1}^{m} W_{ij} h_i)$$





(Approximate) Learning

Maximise likelihood:

$$\frac{\partial \log p(\mathbf{v_n};\theta)}{\partial \theta} = \mathbb{E}[-\frac{\partial E(\mathbf{v},\mathbf{h};\theta)}{\partial \theta} \mid \mathbf{v}=\mathbf{v_n}] - \mathbb{E}[-\frac{\partial E(\mathbf{v},\mathbf{h};\theta)}{\partial \theta}]$$

Positive statistic

Negative statistic

$$\frac{\partial \log p(\mathbf{v_n})}{\partial wij} = \mathbb{E}[h_i v_j \mid \mathbf{v}=\mathbf{v_n}] - \mathbb{E}[h_i v_j]$$

problem

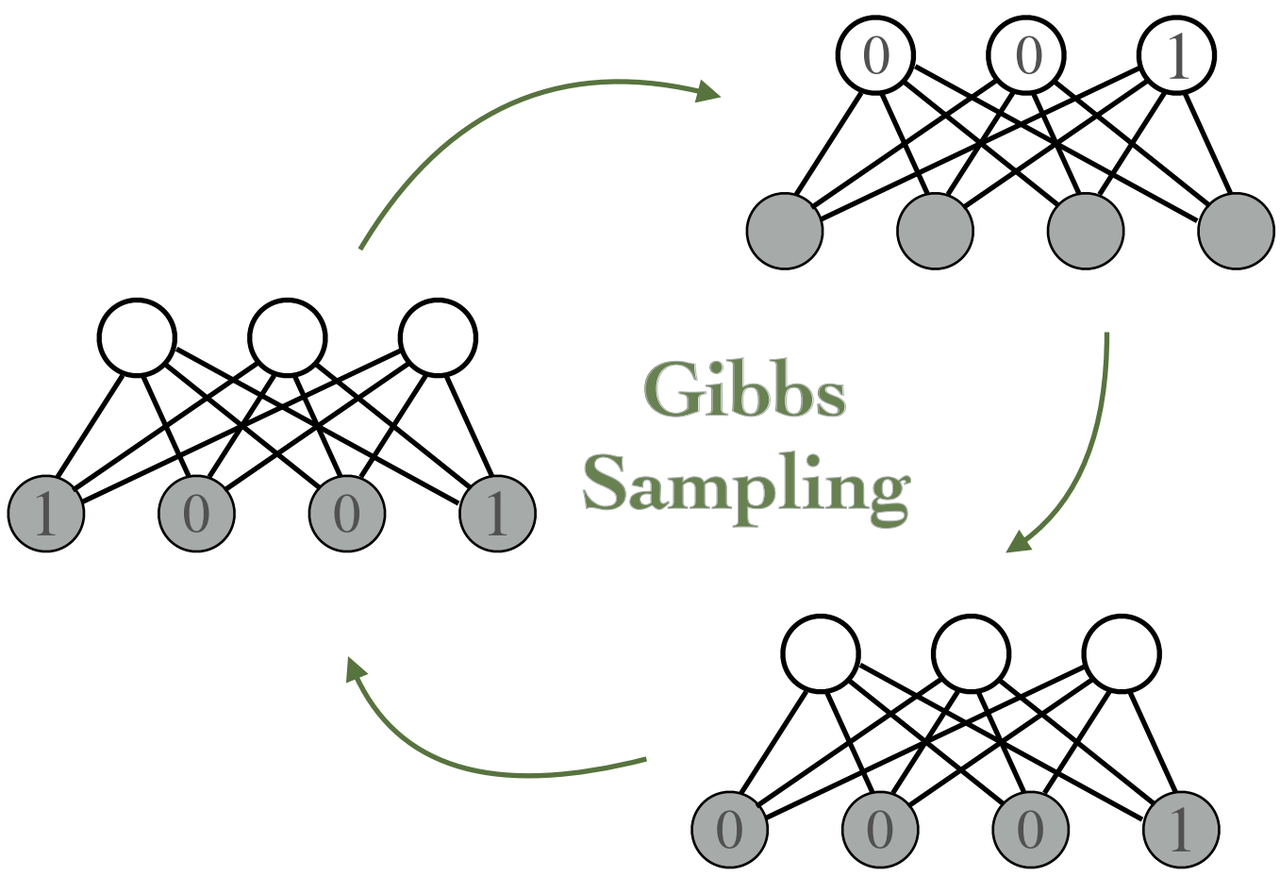
$$\sigma(W\mathbf{v_n} + b_i)$$

Although exact maximum likelihood learning in RBM's is still intractable, learning can be carried out efficiently using Contrastive Divergence

Persistent Contrastive Divergence

$$\mathbb{E}[h_i v_j] \approx \sum_m h_{mi} v_{mj}$$

$$(\mathbf{v}_m^p, \mathbf{h}_m^p) \longrightarrow (\mathbf{v}_m^\infty, \mathbf{h}_m^\infty)$$

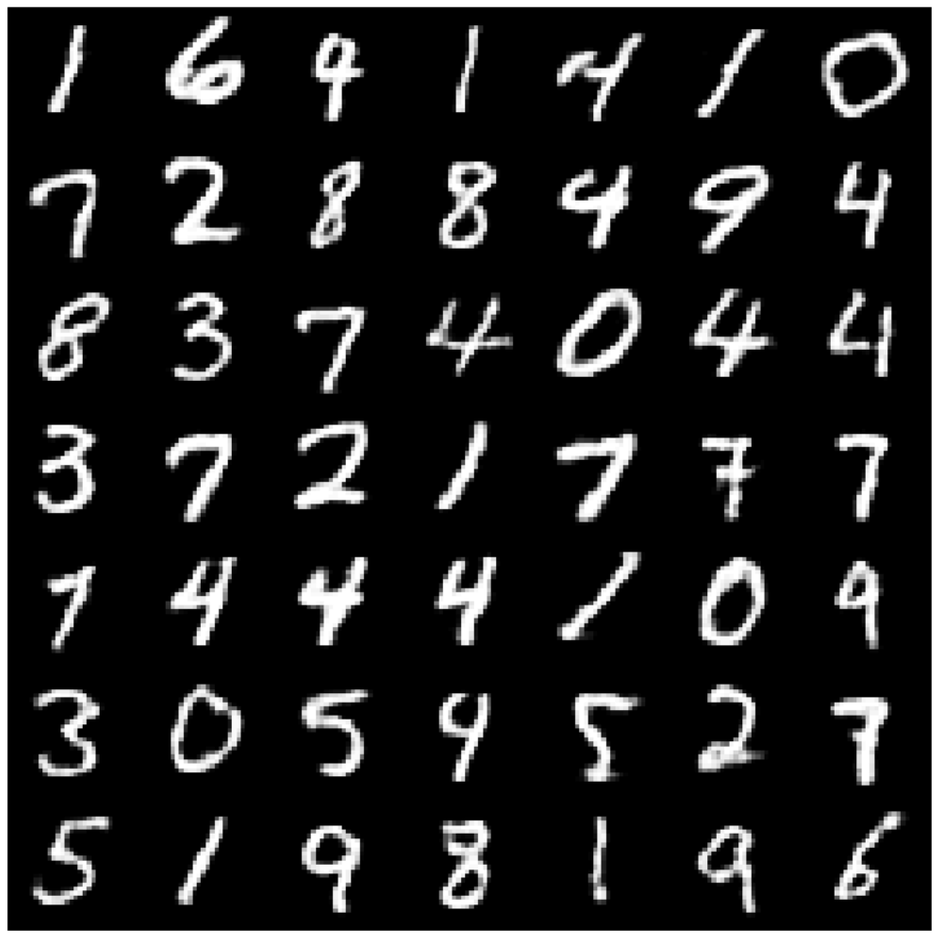


- Start with training data
- Run for a few steps (1-5)
- Update θ, use end state as initialisation

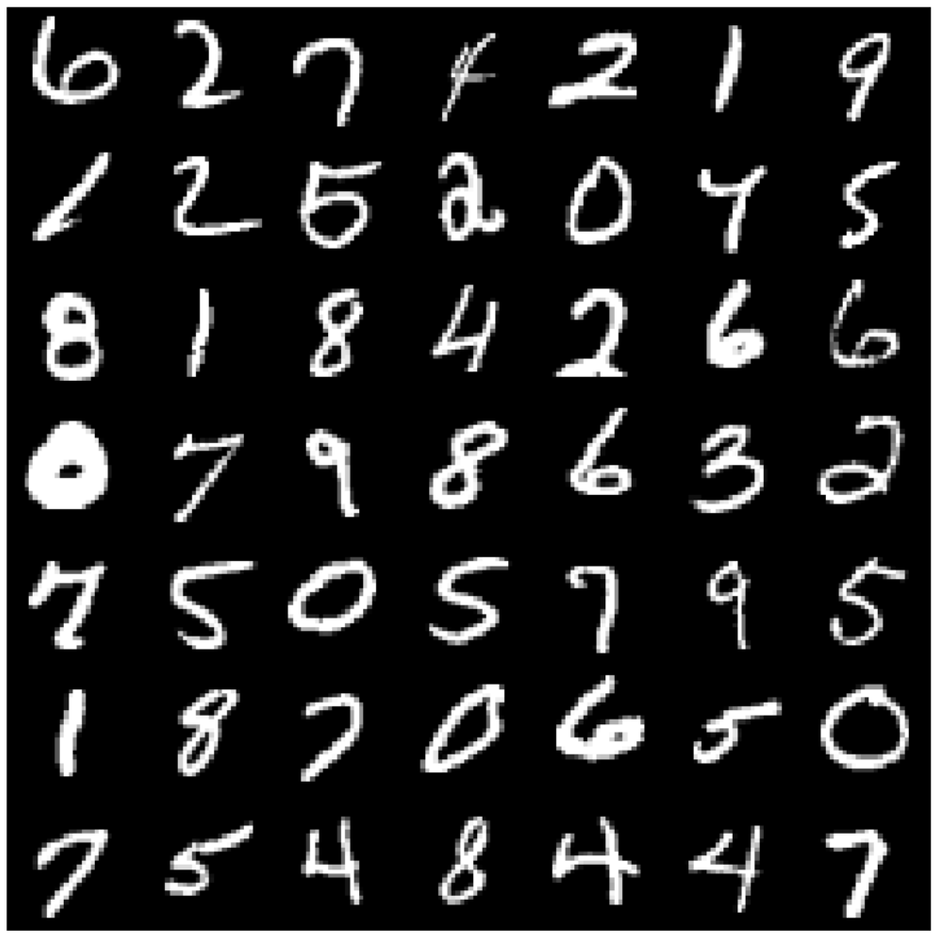
"Fantasy" particles

Generative model

3-layer BM



Training Samples



Random samples from the training set, and samples generated from one deep Boltzmann machines by running the Gibbs sampler for 100,000 steps

Fig. 4 shows samples generated from the two DBM's by randomly initializing all binary states and running the Gibbs sampler for 100,000 steps. Certainly, all samples look like the real handwritten digits.

Model Evaluation

Metrics?

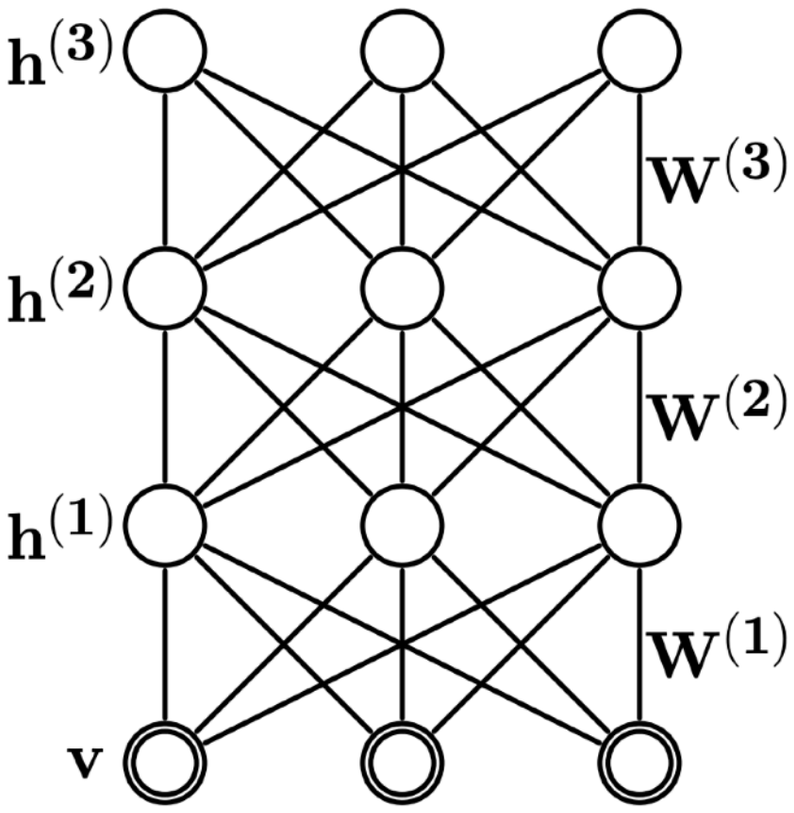
- Visual inspection
- Reconstruction error
- Likelihood

$$p(\mathbf{v};\theta) = \frac{p^*(\mathbf{v};\theta)}{Z(\theta)} \qquad \frac{p(\mathbf{v};\theta_A)}{p(\mathbf{v};\theta_B)} = \frac{p^*(\mathbf{v};\theta_A)}{p^*(\mathbf{v};\theta_B)} \frac{Z(\theta_B)}{Z(\theta_A)}$$

AIS Runs		True $\ln Z$	Estimates $\ln \hat{Z}$	Estimates $\ln(\hat{Z} \pm \hat{\sigma})$	Estimates $\ln(\hat{Z} \pm 3\hat{\sigma})$	Time (mins)
100	CD1(25)	255.41	256.52	255.00, 257.10	0.0000, 257.73	3.3
	CD3(25)	307.47	307.63	307.44, 307.79	306.91, 308.05	3.3
	CD1(20)	279.59	279.57	279.43, 279.68	279.12, 279.87	3.1

On the Quantitative Analysis of Deep Belief Networks, Salakhutdinov et al, 2008

Going DEEP



- Stacked RBMs
- Complex representations

$$P(\mathbf{v};\theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left(\sum_{ij} W_{ij}^{(1)} v_i h_j^{(1)} + \sum_{jl} W_{jl}^{(2)} h_j^{(1)} h_l^{(2)} + \sum_{lm} W_{lm}^{(3)} h_l^{(2)} h_m^{(3)} \right)$$

What changes?

- Learning is harder (Positive statistic)

  Greedy layerwise pretraining+Variational Learning

$$\frac{\partial \log p(\mathbf{v_n};\theta)}{\partial \theta} = \mathbb{E}[-\frac{\partial E(\mathbf{v},\mathbf{h};\theta)}{\partial \theta} \,|\, \mathbf{v}=\mathbf{v_n}] - \mathbb{E}[-\frac{\partial E(\mathbf{v},\mathbf{h};\theta)}{\partial \theta}]$$

Results

2-layer BM

3-layer BM

Training Samples

Table 1: Results of estimating partition functions of BM models, along with the estimates of lower bound on the average training and test log-probabilities. For all BM's we used 20,000 intermediate distributions. Results were averaged over 100 AIS runs.

	Estimates		Avg. log-prob.	
	$\ln \hat{Z}$	$\ln\left(\hat{Z} \pm \hat{\sigma}\right)$	Test	Train
2-layer BM	356.18	356.06, 356.29	-84.62	-83.61
3-layer BM	456.57	456.34, 456.75	-85.10	-84.49

Benchmark

$$p(\mathbf{x_m}|\theta) = \sum_{k=1}^{m} \pi_k \prod_j B(x_{mj}|\mu_{jk})$$

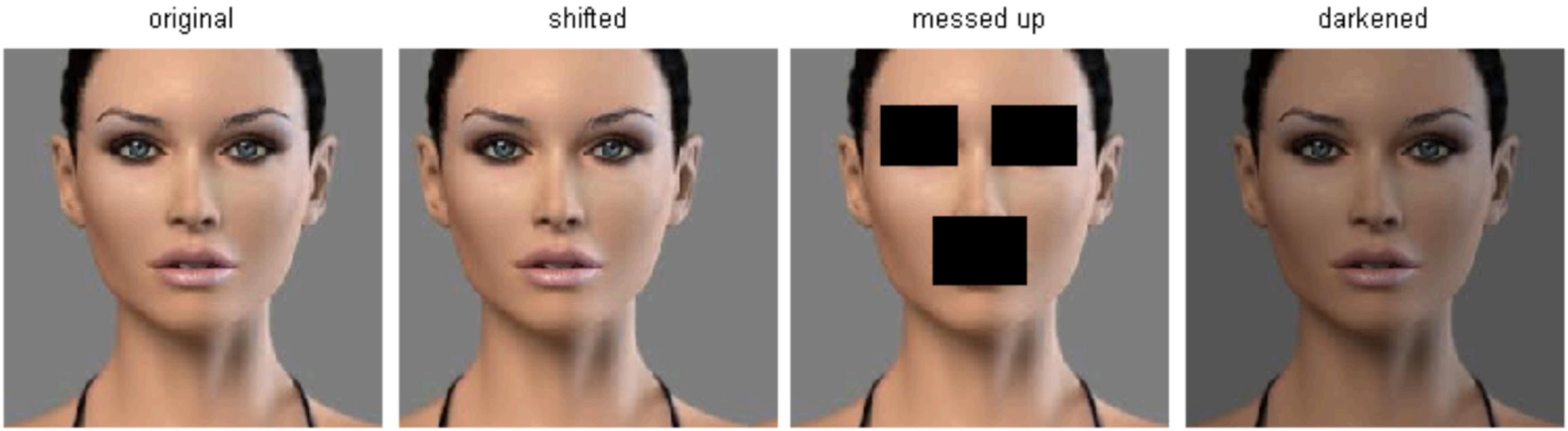
For a simple comparison we also trained several mixture of Bernoullis models with 10, 100, and 500 components. The corresponding average test log-probabilities were -168.95, -142.63, and -137.64. Compared to DBM's, a mixture of Bernoullis performs very badly. The difference of over 50 nats per test case is striking.

Finally, after discriminative fine-tuning, the 2-layer BM achieves an error rate of 0.95% on the full MNIST test set. This is, to our knowledge, the best published result on the permutation-invariant version of the MNIST task. The 3-layer BM gives a slightly worse error rate of 1.01%. This is compared to 1.4% achieved by SVM's (Decoste and Schölkopf, 2002), 1.6% achieved by randomly initialized backprop, and 1.2% achieved by the deep belief network, described in Hinton et al. (2006).

Conclusion

- Useful Representations
- Hard to train
- Probability only known up to Z

L2 in high dimensional data



Pixel-based distances on high-dimensional data (and images especially) can be very unintuitive. An original image (left) and three other images next to it that are all equally far away from it based on L2 pixel distance. Clearly, the pixel-wise distance does not correspond at all to perceptual or semantic similarity.

Learning algorithm

Boltzmann Machine Learning Procedure:

Given: a training set of N data vectors $\{\mathbf{v}\}_{n=1}^{N}$.

1. Randomly initialize parameters θ^0 and M fantasy particles. $\{\tilde{\mathbf{v}}^{0,1}, \tilde{\mathbf{h}}^{0,1}\}$, ..., $\{\tilde{\mathbf{v}}^{0,M}, \tilde{\mathbf{h}}^{0,M}\}$
2. For t=0 to T (# of iterations)
   (a) For each training example $\mathbf{v}^n$, n=1 to N
      - Randomly initialize μ and run mean-field updates Eq. 8 until convergence.
      - Set $\mu^n = \mu$.
   (b) For each fantasy particle m=1 to M
      - Obtain a new state $(\tilde{\mathbf{v}}^{t+1,m}, \tilde{\mathbf{h}}^{t+1,m})$ by running a k-step Gibbs sampler using Eqs. 4, 5, initialized at the previous sample $(\tilde{\mathbf{v}}^{t,m}, \tilde{\mathbf{h}}^{t,m})$.
   (c) Update

$$W^{t+1} = W^t + \alpha_t \left(\frac{1}{N} \sum_{n=1}^{N} \mathbf{v}^n (\mu^n)^\top - \frac{1}{M} \sum_{m=1}^{M} \tilde{\mathbf{v}}^{t+1,m} (\tilde{\mathbf{h}}^{t+1,m})^\top \right).$$

   Similarly update parameters L and J.

   (d) Decrease α_t.

Mean Field

Approximate Learning: Exact maximum likelihood learning in this model is intractable, but efficient approximate learning of DBMs can be carried out by using a mean-field inference to estimate data-dependent expectations, and an MCMC based stochastic approximation procedure to approximate the model's expected sufficient statistics [7]. In particular, consider approximating the true posterior $P(\mathbf{h}|\mathbf{v};\theta)$ with a fully factorized approximating distribution over the three sets of hidden units: $Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) = \prod_{j=1}^{F_1}\prod_{l=1}^{F_2}\prod_{k=1}^{F_3} q(h_j^{(1)}|\mathbf{v})q(h_l^{(2)}|\mathbf{v})q(h_k^{(3)}|\mathbf{v})$, where $\boldsymbol{\mu} = \{\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\mu}^{(3)}\}$ are the mean-field parameters with $q(h_i^{(l)} = 1) = \mu_i^{(l)}$ for $l = 1, 2, 3$. In this case, we can write down the variational lower bound on the log-probability of the data, which takes a particularly simple form:

$$\log P(\mathbf{v};\theta) \;\geq\; \mathbf{v}^\top \mathbf{W}^{(1)} \boldsymbol{\mu}^{(1)} + \boldsymbol{\mu}^{(1)\top} \mathbf{W}^{(2)} \boldsymbol{\mu}^{(2)} + \boldsymbol{\mu}^{(2)\top} \mathbf{W}^{(3)} \boldsymbol{\mu}^{(3)} - \log \mathcal{Z}(\theta) + \mathcal{H}(Q), \quad (2)$$

where $\mathcal{H}(\cdot)$ is the entropy functional. Learning proceeds by finding the value of $\boldsymbol{\mu}$ that maximizes this lower bound for the current value of model parameters θ, which results in a set of the mean-field fixed-point equations. Given the variational parameters $\boldsymbol{\mu}$, the model parameters θ are then updated to maximize the variational bound using stochastic approximation (for details see [7, 11, 14, 15]).