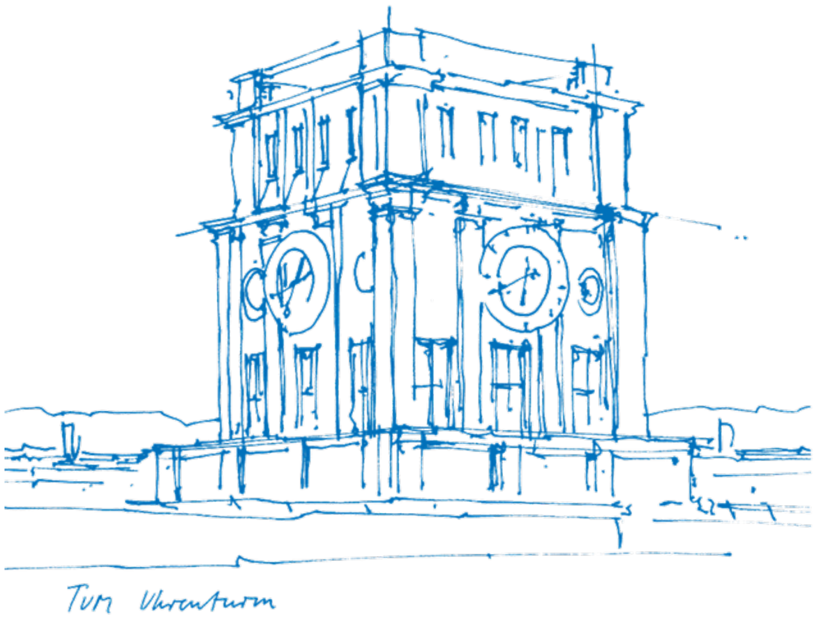
Computer Vision & Artificial Intelligence
Department of Informatics
Technical University of Munich

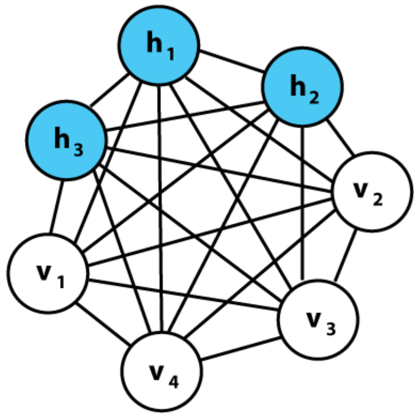
V : Boltzmann machine and contrastive divergence

Tao Wu, Yuesong Shen, Zhenzhang Ye

Computer Vision & Artificial Intelligence
Technical University of Munich



Boltzmann machine



[1]

Boltzmann machine $\simeq$ Binary-label pairwise Markov network

Parametrization ($x_i \in \{0, 1\}$; $w_{i,j}, u_i \in \mathbb{R}$):

$$p(x; w, u) = \frac{1}{Z(w, u)} \exp\Big(\sum_{(i,j)\in\mathcal{E}} w_{i,j} x_i x_j + \sum_{i\in\mathcal{V}} u_i x_i \Big). \tag{1}$$

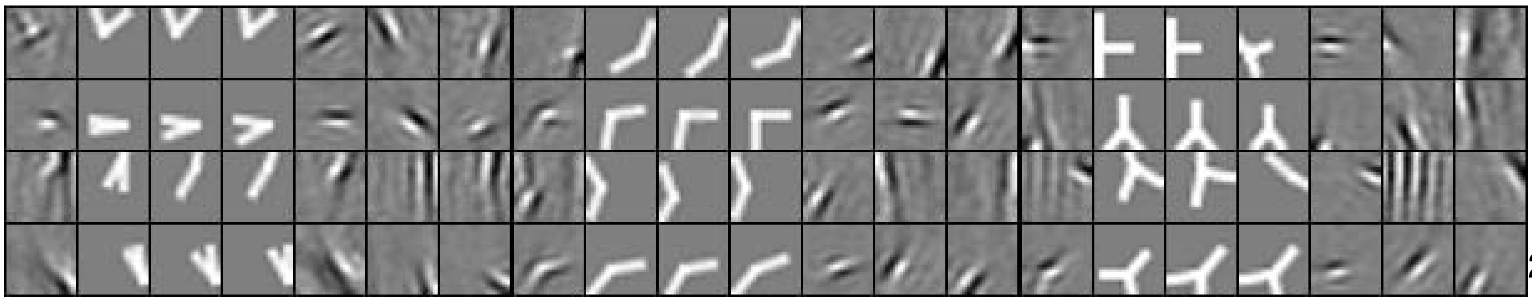
$\rightarrow$ Always possible: recall energy reparametrization for graphcut!

[1] From `https://en.wikipedia.org/wiki/Boltzmann_machine`

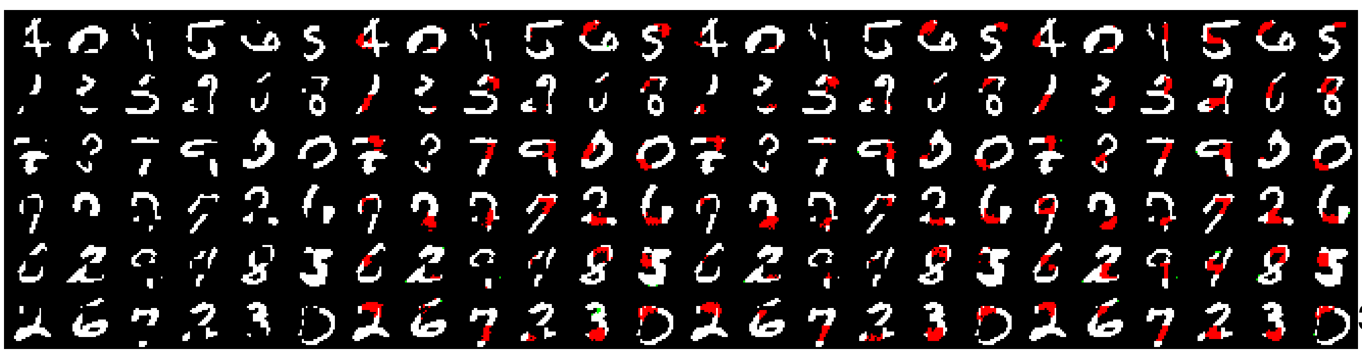
Why Boltzmann machine?

Boltzmann machines are useful for

- Unsupervised feature learning


[2]

- Generative learning


[3]

[2] Lee et al., "Sparse Deep Belief Net Model for Visual Area V2".

[3] Mnih et al., "Conditional Restricted Boltzmann Machines for Structured Output Prediction".

Inference and learning of Boltzmann machine

As we know from the lecture:

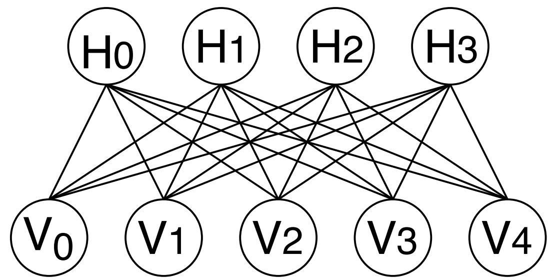
- Exact inference is intractable in general
  - Approximative methods: variational inference, MCMC
  - Sub-modular: graphcut for MAP inference

- probabilistic learning requires inference
  - Exact probabilistic learning also hard ...
  - Often as latent variable model (LVM): even more complicated ...

Nevertheless, we can use **Boltzmann machines with special structures** to simplify the computation while achieving satisfactory results.

Restricted Boltzmann machine (RBM)

RBM: simple LVM, bipartite graph with visible + hidden layer.



Parametrization ($v_i, h_j \in \{0, 1\}$; $w_{i,j}, b_i, c_j \in \mathbb{R}$):

$$p(v, h; w, b, c) = \frac{1}{Z(w, b, c)} \exp\Big(\sum_{(i,j)\in\mathcal{E}} w_{i,j} v_i h_j + \sum_{i\in\mathcal{V}_{\text{visible}}} b_i v_i + \sum_{j\in\mathcal{V}_{\text{hidden}}} c_j h_j \Big). \quad (2)$$

- v: visible layer $\rightsquigarrow$ observation
- h: hidden layer $\rightsquigarrow$ latent representation

RBM: inference

- Layerwise conditional distribution: close form!

  $\rightsquigarrow$ Visible nodes are conditionally independent given hidden nodes;
  $\rightsquigarrow$ Hidden nodes are conditionally independent given visible nodes.

$$p(v_i|h; w, b) = \sigma\big((2v_i - 1)(\sum_j w_{i,j}h_j + b_i)\big), \tag{3}$$

$$p(h_j|v; w, c) = \sigma\big((2h_j - 1)(\sum_i w_{i,j}v_i + c_j)\big), \tag{4}$$

  where $\sigma : x \to 1/(1 + \exp(-x))$ is the sigmoid function.

- Joint distribution: intractable ...

  $\rightsquigarrow$ MCMC: unbiased, high quality, but slow;
  $\rightsquigarrow$ Variational inference: faster, but often biased, lower quality.

RBM: maximum likelihood estimation

Given a dataset $\mathcal{D} = (v_n)_{1 \le n \le N}$, the averaged log-likelihood is then:

$$l(\mathcal{D}; w, b, c) = \frac{1}{N} \log p(\mathcal{D}|w, b, c) = \frac{1}{N} \sum_n \log p(v_n|w, b, c) \tag{5}$$

$$= \frac{1}{N} \sum_n (\log \sum_h \exp(\sum_{(i,j)\in\mathcal{E}} w_{i,j} v_{i,n} h_j + \sum_{i\in\mathcal{V}_{\text{visible}}} b_i v_{i,n} + \sum_{j\in\mathcal{V}_{\text{hidden}}} c_j h_j)) \tag{6}$$

$$- \log Z(w, b, c). \tag{7}$$

Its derivatives w.r.t parameters w, b, c are:

$$\frac{\partial l(\mathcal{D}; w, b, c)}{\partial w_{i,j}} = \mathbb{E}_{V\sim r}[\mathbb{E}_{H\sim p(\cdot|V)}[V_i H_j]] - \mathbb{E}_{V,H\sim p}[V_i H_j]; \tag{8}$$

$$\frac{\partial l(\mathcal{D}; w, b, c)}{\partial b_i} = \mathbb{E}_{V\sim r}[V_i] - \mathbb{E}_{V\sim p}[V_i]; \tag{9}$$

$$\frac{\partial l(\mathcal{D}; w, b, c)}{\partial c_j} = \mathbb{E}_{V\sim r}[\mathbb{E}_{H\sim p(\cdot|V)}[H_j]] - \mathbb{E}_{H\sim p}[H_j]. \tag{10}$$

$\rightsquigarrow$ r/p: data / model distribution. Always diff of $\mathbb{E}_{\text{data}}$ and $\mathbb{E}_{\text{model}}$.

Contrastive divergence (CD)

For RBMs, data terms $\mathbb{E}_{\text{data}}$ have close form, but model terms $\mathbb{E}_{\text{model}}$ are intractable!

Main idea: accelerate MCMC via truncation.

- **layer-wise blocking Gibbs**: fast with close-form conditional distribution
- **k-step truncation**: no need to wait for / check convergence

$$\text{initialization: } v_n^{(0)} = v_n, h_n^{(0)} \sim p(h|v_n), \tag{11}$$

$$\text{for } i \text{ from 1 to } k\text{: } v_n^{(i)} \sim p(v|h^{(i-1)}), h_n^{(i)} \sim p(h|v^{(i)}). \tag{12}$$

- Even $k = 1$ works in practice;
- $v_n^{(k)}, h_n^{(k)}$ are biased samples;
- $k \to +\infty$: unbiased, blocking Gibbs sampling till convergence.

CD-k: procedure in practice

Directly estimating gradients with samples $v_n^{(k)}, h_n^{(k)}$ will result in high variance.

Solution: use “soft version” (i.e. sample distribution) directly instead!

$$\widetilde{h}_n^{(0)} = p(h|v_n), \widetilde{v}_n^{(k)} = p(v|h^{(k-1)}), \widetilde{h}_n^{(k)} = p(h|v^{(k)}). \tag{13}$$

CD-k procedure in practice:

- Iterate over observations v_n:
  - Initialization: $v_n^{(0)} = v_n$;
  - For i from 0 to $k-1$: $h_n^{(i)} \sim p(h|v^{(i)}), v_n^{(i+1)} \sim p(v|h^{(i)})$;
  - Compute $\widetilde{h}_n^{(0)} = p(h|v_n), \widetilde{v}_n^{(k)} = p(v|h^{(k-1)}), \widetilde{h}_n^{(k)} = p(h|v^{(k)})$;
  - Stochastic gradients
    $\Delta w = \widetilde{h}_n^{(0)} v_n^\top - \widetilde{h}_n^{(k)} \widetilde{v}_n^{(k)\top}, \Delta b = v_n - \widetilde{v}_n^{(k)}, \Delta c = \widetilde{h}_n^{(0)} - \widetilde{h}_n^{(k)}$

We can then update parameters with gradient-based optimization.

Persistent CD (PCD)

In CD-k, the MCMC is reinitialized for every sample, resulting in short chains.

If the distribution changes slowly (e.g. small learning rate), then the last sample of the previous chain is a good initialization for the next chain.

$\rightsquigarrow$ We can keep the MCMC running, resulting in **"persistent" CD**.

PCD procedure in practice:

- Iterate over observations v_n:
  - Initialization: $v_n^{(0)} = v_{n-1}^{(k)}$;
  - For i from 0 to $k-1$: $h_n^{(i)} \sim p(h|v^{(i)}),\ v_n^{(i+1)} \sim p(v|h^{(i)})$;
  - Compute $\widetilde{h}_n^{(0)} = p(h|v_n),\ \widetilde{v}_n^{(k)} = p(v|h^{(k-1)}),\ \widetilde{h}_n^{(k)} = p(h|v^{(k)})$;
  - Stochastic gradients
    $\Delta w = \widetilde{h}_n^{(0)} v_n^\top - \widetilde{h}_n^{(k)} \widetilde{v}_n^{(k)\top},\ \Delta b = v_n - \widetilde{v}_n^{(k)},\ \Delta c = \widetilde{h}_n^{(0)} - \widetilde{h}_n^{(k)}$

We can then update parameters with gradient-based optimization.

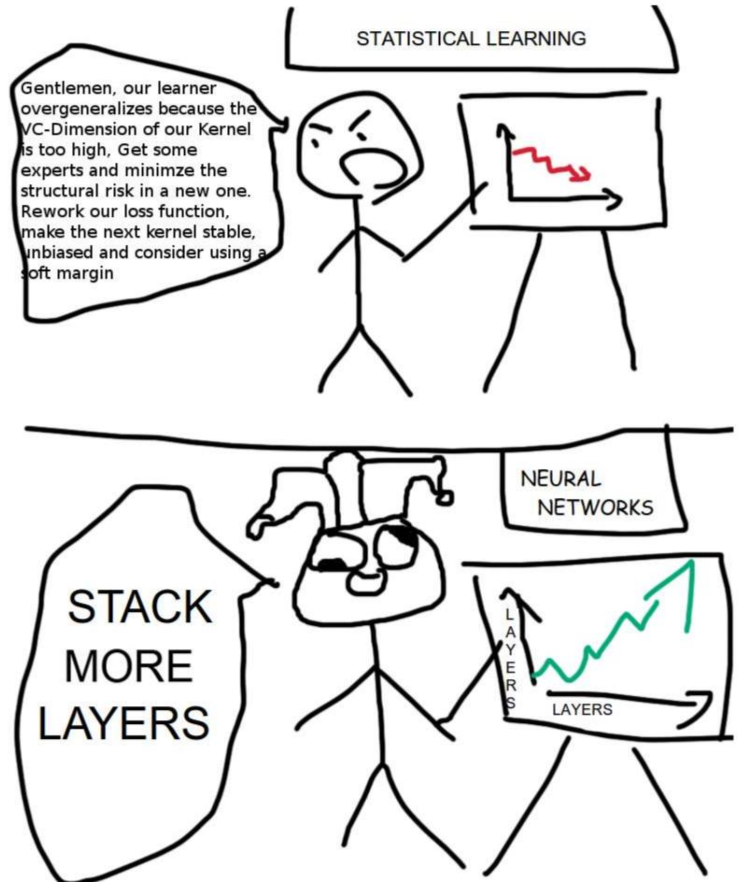
Going deeper ...

RBM has simple structure:

- + unsupervised LVM
- + easy to train
- - limited modeling power

Solution: **more hidden layers**

- + unsupervised LVM
- + stronger modeling power
- + efficient computation
  $\rightsquigarrow$ tensor ops, blocking Gibbs
- - more complex than RBM
  $\rightsquigarrow$ no closed form for data term!



[4]

[4]From https://www.reddit.com/r/ProgrammerHumor/comments/5si1f0/machine_learning_approaches/

Greedy training of multiple hidden layers

- Training 1 hidden layer – solved: CD-k, PCD ...
- Training more hidden layers – unknown

$\rightsquigarrow$ CD doesn't easily apply: no more close form for $p(h|v)$!

$\rightsquigarrow$ How do we solve it?

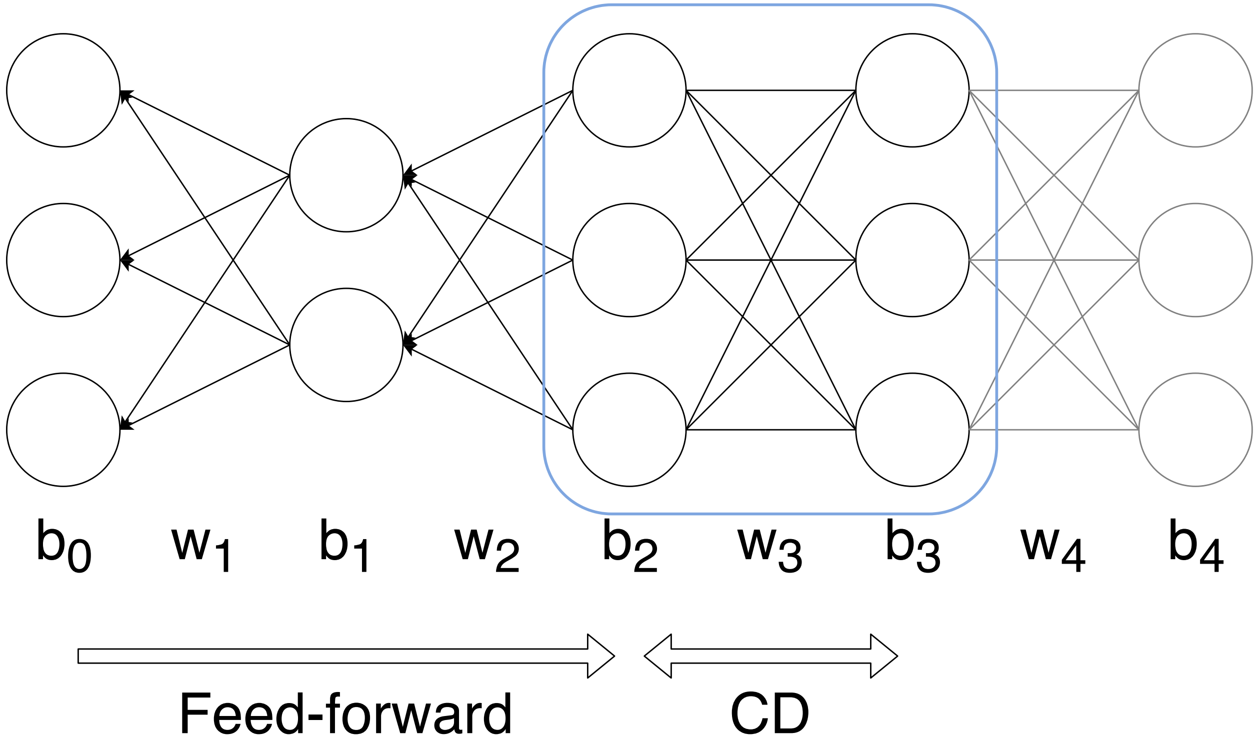
Intuitive idea: train hidden layers one by one!

Given layers $V = H_0, H_1, H_2, \ldots, H_n$, proceed as follows:

For i from 0 to $n - 1$ do:

- Transform inputs via feed-forward through trained layers till layer i;
- Train the weights and biases associated with H_i and H_{i+1} as an RBM;
- Fix the parameters in this trained layer.

Greedy training: illustration



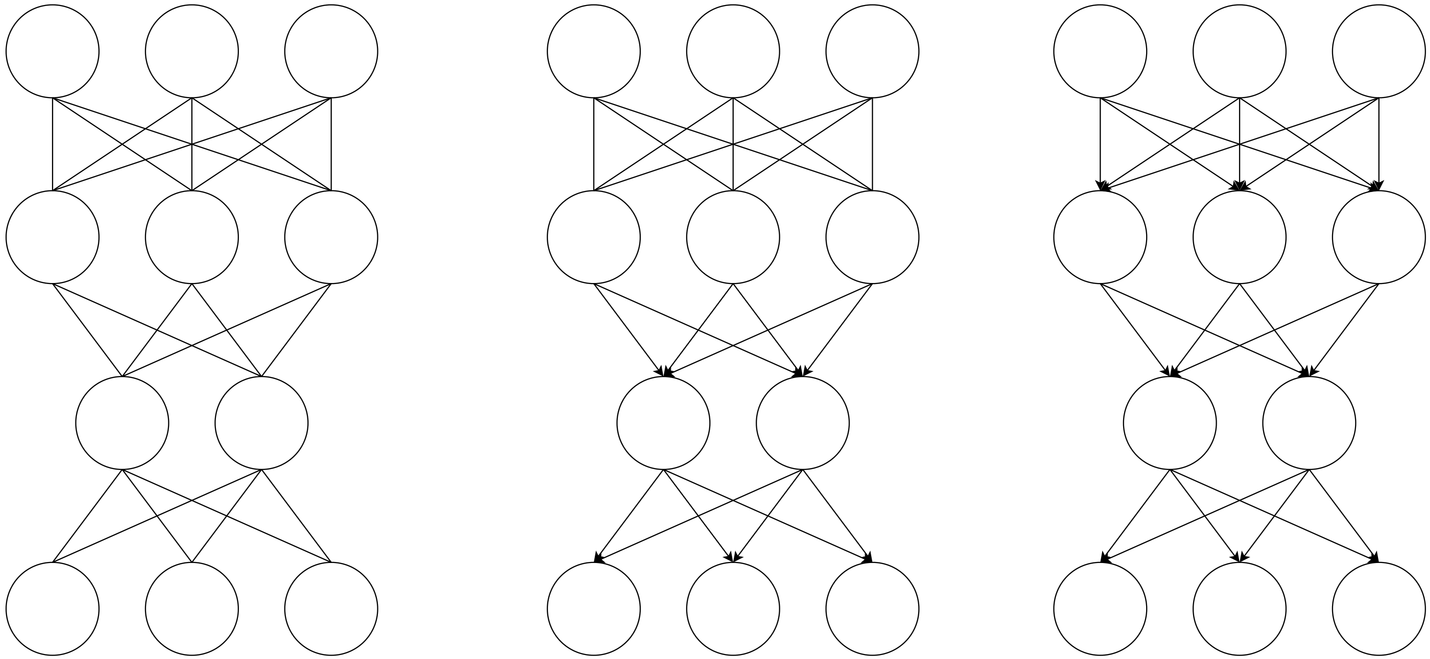
- Input transform (feed-forward): $v_2 = \sigma(w_2^\top \sigma(w_1^\top v + b_1) + b_2)$
- Layer training (CD): train with CD layer 2 and 3 as an RBM, this will update w_3, b_3 and a temporary bias $\widetilde{b_2}$ for layer 2 which will be discarded.[5]

[5] See implementation from e.g. `http://deeplearning.net/tutorial/DBN.html`.

Deep belief network (DBN)

Does this greedy training make sense? Actually, yes!
Do we get a deep Boltzmann machine? Surprisingly, no!

We actually get a hybrid graphical model called **deep belief network**.



Deep Boltzmann machine Deep belief network Sigmoid belief network

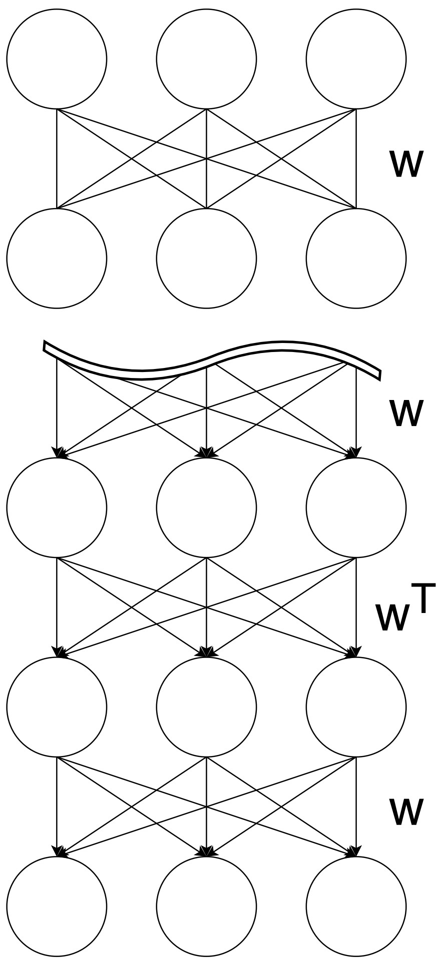
Justification: DBN as unrolled RBM[6]

An RBM can be unrolled as an infinite SBN

- Blocking Gibbs $\rightarrow$ ancestral sampling
- Weights are coupled
- Partial unrolling $\rightarrow$ deep belief net
- Layers above form a “complementary prior”
  $\rightsquigarrow$ Cancels out “explaining away” effect;
  $\rightsquigarrow$ Layer-wise RBM training makes sense!

General DBNs violate assumptions above

- Uncouple weights, arbitrary layer sizes
- No longer complementary prior
- Nevertheless a practical approximation



[6] Hinton et al., “A fast learning algorithm for deep belief nets”.

Fine tuning a DBN

The greedily (pre)trained DBN can be fine-tuned ...

- For generative tasks: wake-sleep algorithm + MCMC

  $\rightsquigarrow$ Wake-sleep for the directed part, MCMC for top layer;

  $\rightsquigarrow$ Sampling process: MCMC on top layer + top-down ancestral sampling.

- For discriminative tasks: feed-forward + backprop

  $\rightsquigarrow$ DBN is turned into a neural network (NN)

  $\rightsquigarrow$ Helped “old” NNs to beat SVM and achieve state-of-the-art

  $\rightsquigarrow$ No longer useful for “new” NNs with better activation functions (rectified linear unit (ReLU)), initialization (Xavier / Kaiming), regularization (dropout, batch-normalization) ...

Deep Boltzmann machine (DBM)

DBM: multi-hidden layer Boltzmann machine

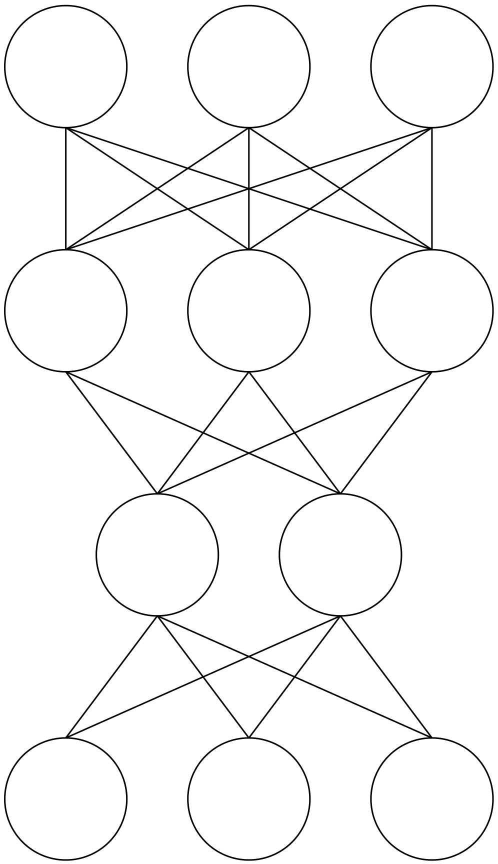
From lecture: partially observed MRF

- MLE gradient has data and model terms;
- Both need inference.

Training? According to Salakhutdinov and Hinton[7]:

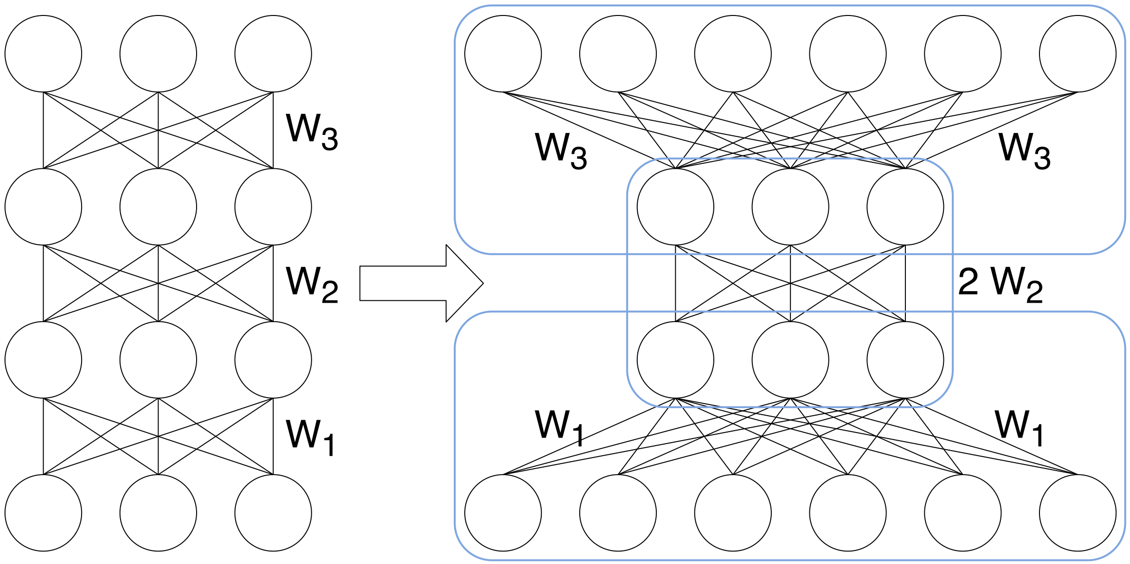
- Data term $\to$ mean field;
- Model term $\to$ blocking Gibbs sampling
  $\rightsquigarrow$ 2 blocks: odd / even layers

Pre-training? Yes!



[7] Salakhutdinov and Hinton, "Deep Boltzmann Machines".

Deep Boltzmann machine: pre-training



Similar to DBN, DBM can also benefit from layerwise CD pretraining!

However, modifications needed to approximate information from both sides:

- 1st and last layers are cloned with coupled weights;
- weights in all middle layers are doubled.

Other extensions

- Gaussian input nodes: Gaussian-Bernoulli RBM ...
- Sparse connections: convolutional layer connections
- Directed counter part: sigmoid belief network
- “A la mode” DL generative models:
  - Variational auto-encoder (VAE);
  - Generative adversarial network (GAN);
  - ...
- ...

Further Reading

- Section 27.7 and 28.2 of Murphy MLAPP.
- Chapter 20 of Deep Learning (Goodfellow, Bengio & Courville, 2016).