Deep Boltzmann Machines

Sargur N. Srihari
srihari@cedar.buffalo.edu

Topics

1. Boltzmann machines
   2. Restricted Boltzmann machines
   3. Deep Belief Networks
   4. Deep Boltzmann machines
   5. Boltzmann machines for continuous data
   6. Convolutional Boltzmann machines
   7. Boltzmann machines for structured and sequential outputs
   8. Other Boltzmann machines
9. Backpropagation through random operations
10. Directed generative nets
11. Drawing samples from autoencoders
12. Generative stochastic networks
13. Other generative schemes
14. Evaluating generative models
15. Conclusion

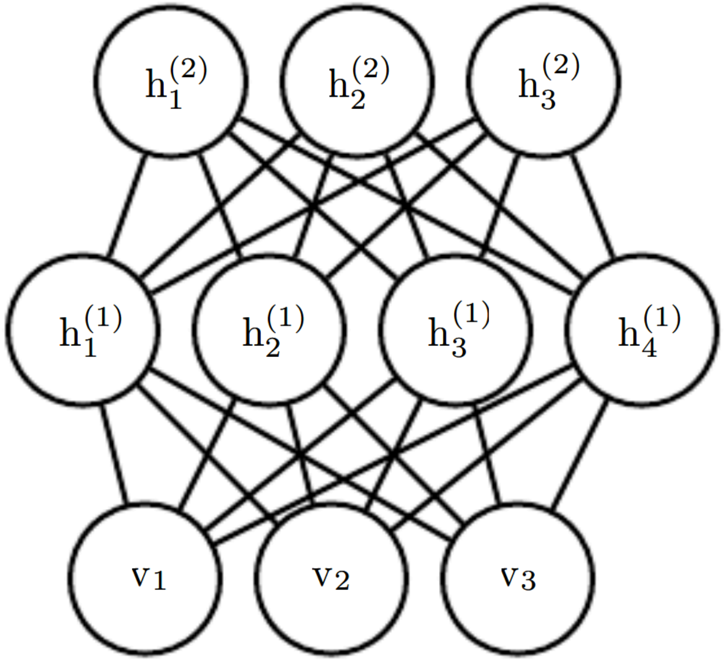
Topics in Deep Boltzmann Machines

- What is a Deep Boltzmann machine (DBM)?
  - Example of a Deep Boltzmann machine
- DBM Representation
- DBM Properties
- DBM Mean Field Inference
- DBM Parameter Learning
- Layerwise Pre-training
- Jointly training DBMs

What is a Deep Boltzmann Machine?

- It is deep generative model
- Unlike a Deep Belief network (DBN) it is an entirely undirected model
- An RBM has only one hidden layer
- A Deep Boltzmann machine (DBM) has several hidden layers

PGM for a DBM



Unlike a DBN, a DBM is an entirely undirected model

This one has one visible layer and two hidden layers

Connections are only between units in neighboring layers

Like RBMs and DBNs,
DBMs contain only binary units

Parameterization of DBM

- Joint parameterization as energy function E
- Case of DBM with one visible layer $\boldsymbol{v}$ and three hidden layers $\boldsymbol{h}^{(1)}$, $\boldsymbol{h}^{(2)}$, $\boldsymbol{h}^{(3)}$, joint probability is

$$\boxed{P\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}\right) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta})\right)}$$

  - Bias parameters are omitted for simplicity

  – DBM energy function is defined as follows:

$$\boxed{E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta}) = -\boldsymbol{v}^{\top}\boldsymbol{W}^{(1)}\boldsymbol{h}^{(1)} - \boldsymbol{h}^{(1)\top}\boldsymbol{W}^{(2)}\boldsymbol{h}^{(2)} - \boldsymbol{h}^{(2)\top}\boldsymbol{W}^{(3)}\boldsymbol{h}^{(3)}}$$

  - In comparison with RBM energy function

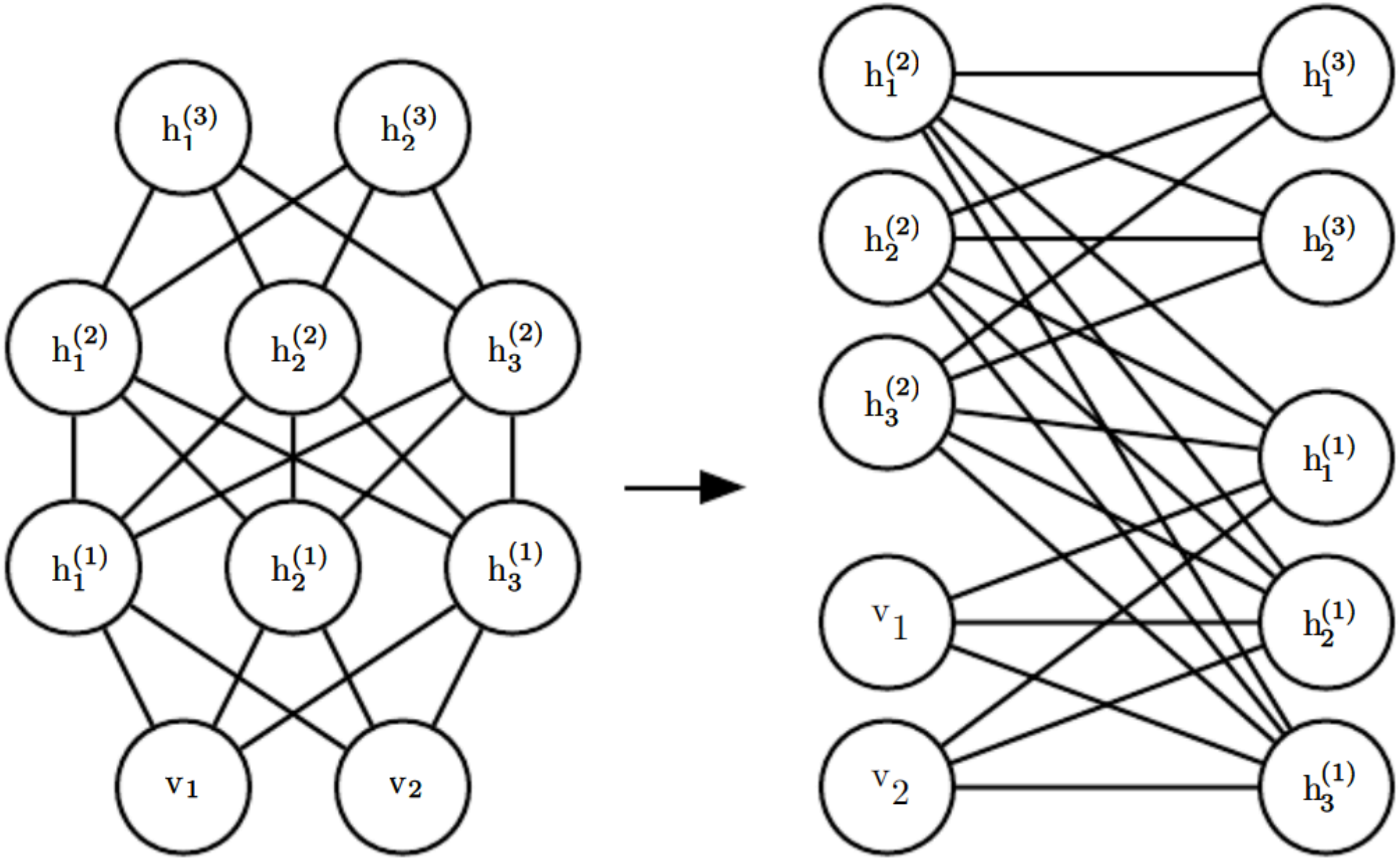
    $E(\boldsymbol{v},\boldsymbol{h}) = -\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{h} - \boldsymbol{v}^{\mathrm{T}} W\boldsymbol{h}$

    DBM includes connections between hidden units in the form of weight matrices $W^{(1)}$ and $W^{(2)}$

Structure of DBM

- In comparison to fully connected Boltzmann machines (with every unit connected to every other unit), DBM offers advantages similar to those offered by RBM
- DBM layers can be organized as a bipartite graph

DBM Rearranged as Bipartite



Rearranged to show its bipartite structure

Odd layers on one side, even layers on other

Conditional distributions in DBM

- Bipartite structure implication
  - Can use same equations as for conditionals of RBM
  - Units within each layer are conditionally independent of each other given the values of neighboring layers
    - So distributions over binary variables can be fully described by the Bernoulli parameters
  - For two hidden layers, activation probabilities are:

$$\boxed{P(v_i = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(\boldsymbol{W}_{i,:}^{(1)} \boldsymbol{h}^{(1)}\right)}$$

$$\boxed{P(h_i^{(1)} = 1 \mid \boldsymbol{v}, \boldsymbol{h}^{(2)}) = \sigma\left(\boldsymbol{v}^\top \boldsymbol{W}_{:,i}^{(1)} + \boldsymbol{W}_{i,:}^{(2)} \boldsymbol{h}^{(2)}\right)}$$

$$\boxed{P(h_k^{(2)} = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(\boldsymbol{h}^{(1)\top} \boldsymbol{W}_{:,k}^{(2)}\right)}$$

Gibbs sampling in a DBM

- Bipartite structure makes Gibbs sampling efficient
  - Naiive approach to Gibbs is to update only one variable at a time
  - RBMs allow all visible units to be updated in one block and all hidden units in a second block
  - Instead of $l+1$ updates, can update all units in two iterations
- Efficient sampling is important for training with stochastic maximum likelihood

Interesting properties of DBMs

- Computing $P(\boldsymbol{h}|\boldsymbol{v})$ is simpler than for DBNs
  - Simplicity allows richer approximations to posterior
- Classification is performed using a heuristic approximate inference
  - Mean field expectation of hidden units provided by an upward pass in an MLP with sigmoid and same weights as original DBN
    - *Any* $Q(\boldsymbol{h})$ obtains variational lower bound on log-likelihood
- Lack of intra-layer interactions makes possible use of fixed point equations to optimize variational lower bound

DBM and Neuroscience

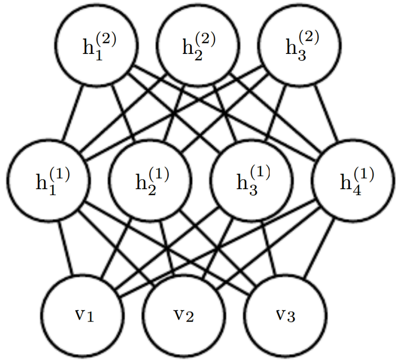
- Use of proper mean-field allows approximate inference to capture influence of top-down feedback interactions
- This makes DBMs interesting from neuroscience viewpoint
  - Human brain is known to use many top-down feedback connections
  - Therefore DBMs have been used as computational models for real neuroscientific phenomena

Sampling from a DBM is difficult

- One unfortunate property is that sampling from a DBM is relatively difficult
- This is in contrast with DBNs
  - Need only need to use MCMC sampling in their top pair of layers, other layers are only used at the end in an efficient ancestral sampling pass
- To generate a sample from a DBM, it is necessary to use MCMC across all layers
  - With every layer of the model participating in every Markov chain interaction

DBM Mean Field Inference

- The conditional distribution over one DBM layer given the neighboring layers is factorial
  - In the case of two hidden layers they are $P(\boldsymbol{v}|\boldsymbol{h}^{(1)})$, $P(\boldsymbol{h}^{(1)}|\boldsymbol{v},\boldsymbol{h}^{(2)})$ and $P(\boldsymbol{h}^{(2)}|\boldsymbol{h}^{(1)})$



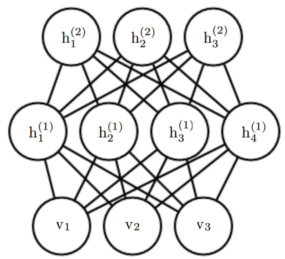
  - The distribution over all hidden layers generally does not factorize because of the interaction of the interaction weights $W^{(2)}$ between $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(2)}$, which render these variables dependent

Approximating the DBM Posterior

- As with DBN, we are left to seek out methods to approximate the DBM posterior distribution
  - Unlike DBN, posterior (given visible units) over hidden units is easy to variationally approximate using mean-field
    - In mean field we restrict the approximating distribution to fully factorial distributions
- In DBMs mean field equations capture bidirectional interactions between layers
  - Approximating family is the set of distributions where the hidden units are conditionally independent

Equations for Mean-Field Inference

- We develop mean field approach for the example with two hidden layers



  – Let $Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}|\boldsymbol{v})$ be the approximation to $P(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}|\boldsymbol{v})$

- The mean-field assumption means that

$$Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}|\boldsymbol{v}) = \prod_j Q(h_j^{(1)}|\boldsymbol{v}) \prod_k Q(h_k^{(2)}|\boldsymbol{v})$$

  - The mean field approximation attempts to find a member of this family that best fits the true posterior $P(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}|\boldsymbol{v})$

- Importantly the inference process must be run again to find a distribution Q every time we use a new value of $\boldsymbol{v}$

Measuring how well $Q(\boldsymbol{h}|\boldsymbol{v})$ fits $P(\boldsymbol{h}|\boldsymbol{v})$

- The variational approach is to minimize

$$\mathrm{KL}(Q\|P) = \sum_{\boldsymbol{h}} Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v}) \log \left(\frac{Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v})}{P(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v})} \right)$$

  - In general we do not have to provide a parametric form for the approximating distribution beyond enforcing the independence assumptions
    - The variational approximation is able to recover a functional form of the approximate distribution
- However in mean field on binary hidden units
- No loss of generality in fixing parameterization in advance

Parameterizing Q as Bernoulli product

- Parameterize Q as a product of Bernoulli distributions
  - i.e., we associate the probability of each element of $\boldsymbol{h}^{(1)}$ with a parameter, Specifically

$$\text{For each } j,\ \hat{h}_j^{(1)} = Q(h_j^{(1)} = 1 \mid \boldsymbol{v}),\ \text{where } \hat{h}_j^{(1)} \in [0,1]$$

$$\text{For each } k,\ \hat{h}_k^{(2)} = Q(h_k^{(2)} = 1 \mid \boldsymbol{v}),\ \text{where } \hat{h}_k^{(2)} \in [0,1]$$

  - Thus we have the following approximation to the posterior

$$Q(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)} \mid \boldsymbol{v}) = \prod_j Q(h_j^{(1)} \mid \boldsymbol{v}) \prod_k Q(h_k^{(2)} \mid \boldsymbol{v})$$
$$= \prod_j (\hat{h}_j^{(1)})^{h_j^{(1)}} (1 - \hat{h}_j^{(1)})^{(1-h_j^{(1)})} \times \prod_k (\hat{h}_k^{(2)})^{h_k^{(2)}} (1 - \hat{h}_k^{(2)})^{(1-h_k^{(2)})}.$$

Choosing member of family

- Mean field approximation: $q(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_i q(h_i \mid \boldsymbol{v})$.
  - Optimal $q(h_i|\boldsymbol{v})$ may be obtained by normalizing the unnormalized distribution $\boxed{\tilde{q}(h_i \mid \boldsymbol{v}) = \exp\left(\mathbb{E}_{\mathbf{h}_{-i} \sim q(\mathbf{h}_{-i}|\boldsymbol{v})} \log \tilde{p}(\boldsymbol{v}, \boldsymbol{h})\right)}$
    - These equations were derived by solving for where the derivatives for the variational bound are zero
    - Applying these general equations, we obtain update rules

$$\hat{h}_j^{(1)} = \sigma\left(\sum_i v_i W_{i,j}^{(1)} + \sum_{k'} W_{j,k'}^{(2)} \hat{h}_{k'}^{(2)}\right), \quad \forall j$$

$$\hat{h}_k^{(2)} = \sigma\left(\sum_{j'} W_{j',k}^{(2)} \hat{h}_{j'}^{(1)}\right), \quad \forall k.$$

    - At a fixed point of this system of equations we have a local maximum of the variational lower bound L(Q)
    - They define an iterative algorithm where we alternate updates of the two equations

DBM Parameter Learning

- Learning in a DBM must confront two intractabilities:
  - Intractable partition function
    - $P(\boldsymbol{h}|\boldsymbol{v})$ is intractable (since DBM is undirected)
    - Solution: construct mean-field distribution $Q(\boldsymbol{h}|\boldsymbol{v})$
  - Intractable posterior distribution
    - Log-likelihood $\log P(\boldsymbol{v};\boldsymbol{\theta})$ is intractable
    - Solution: maximize $\mathcal{L}(\boldsymbol{v}, Q, \boldsymbol{\theta})$, the variational lower bound on $\log P(\boldsymbol{v};\boldsymbol{\theta})$

Lower bound on log-likelihood

- For a DBM with two layers

$$\mathcal{L}(Q,\boldsymbol{\theta}) = \sum_i \sum_{j'} v_i W^{(1)}_{i,j'} \hat{h}^{(1)}_{j'} + \sum_{j'} \sum_{k'} \hat{h}^{(1)}_{j'} W^{(2)}_{j',k'} \hat{h}^{(2)}_{k'} - \log Z(\boldsymbol{\theta}) + \mathcal{H}(Q)$$

- Expression still contains log partition $\log Z(\theta)$
  - So approximate methods are required
  - Training the model requires gradient of log-partition
  - DBMs are typically trained using Stochastic Maximum Likelihood
- Contrastive Divergence is slow for DBMs
  - Because they do not allow efficient sampling of the hidden units given the visible units
    - CD would require burning in a Markov chain every time a negative sample is required

Variational Stochastic MLE Algorithm:

For training a 2-layer DBM

Set ϵ, the step size, to a small positive number
Set k, the number of Gibbs steps, high enough to allow a Markov chain of $p(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}; \boldsymbol{\theta} + \epsilon\Delta_{\boldsymbol{\theta}})$ to burn in, starting from samples from $p(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}; \boldsymbol{\theta})$.
Initialize three matrices, $\tilde{\boldsymbol{V}}$, $\tilde{\boldsymbol{H}}^{(1)}$ and $\tilde{\boldsymbol{H}}^{(2)}$ each with m rows set to random values (e.g., from Bernoulli distributions, possibly with marginals matched to the model's marginals).
while not converged (learning loop) **do**
Sample a minibatch of m examples from the training data and arrange them as the rows of a design matrix $\boldsymbol{V}$.
Initialize matrices $\hat{\boldsymbol{H}}^{(1)}$ and $\hat{\boldsymbol{H}}^{(2)}$, possibly to the model's marginals.
while not converged (mean field inference loop) **do**

$$\hat{\boldsymbol{H}}^{(1)} \leftarrow \sigma\left(\boldsymbol{V}\boldsymbol{W}^{(1)} + \hat{\boldsymbol{H}}^{(2)}\boldsymbol{W}^{(2)\top}\right).$$

$$\hat{\boldsymbol{H}}^{(2)} \leftarrow \sigma\left(\hat{\boldsymbol{H}}^{(1)}\boldsymbol{W}^{(2)}\right).$$

end while

$\Delta_{\boldsymbol{W}^{(1)}} \leftarrow \frac{1}{m}\boldsymbol{V}^\top \hat{\boldsymbol{H}}^{(1)}$
$\Delta_{\boldsymbol{W}^{(2)}} \leftarrow \frac{1}{m}\hat{\boldsymbol{H}}^{(1)\top} \hat{\boldsymbol{H}}^{(2)}$
for $l = 1$ to k (Gibbs sampling) **do**
Gibbs block 1:

$$\forall i, j, \tilde{V}_{i,j} \text{ sampled from } P(\tilde{V}_{i,j} = 1) = \sigma\left(\boldsymbol{W}^{(1)}_{j,:}\left(\tilde{\boldsymbol{H}}^{(1)}_{i,:}\right)^\top\right).$$

$$\forall i, j, \tilde{H}^{(2)}_{i,j} \text{ sampled from } P(\tilde{H}^{(2)}_{i,j} = 1) = \sigma\left(\tilde{\boldsymbol{H}}^{(1)}_{i,:}\boldsymbol{W}^{(2)}_{:,j}\right).$$

Gibbs block 2:

$$\forall i, j, \tilde{H}^{(1)}_{i,j} \text{ sampled from } P(\tilde{H}^{(1)}_{i,j} = 1) = \sigma\left(\tilde{\boldsymbol{V}}_{i,:}\boldsymbol{W}^{(1)}_{:,j} + \tilde{\boldsymbol{H}}^{(2)}_{i,:}\boldsymbol{W}^{(2)\top}_{j,:}\right).$$

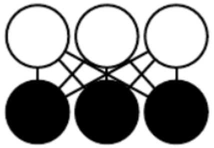
end for

$\Delta_{\boldsymbol{W}^{(1)}} \leftarrow \Delta_{\boldsymbol{W}^{(1)}} - \frac{1}{m}\boldsymbol{V}^\top \tilde{\boldsymbol{H}}^{(1)}$
$\Delta_{\boldsymbol{W}^{(2)}} \leftarrow \Delta_{\boldsymbol{W}^{(2)}} - \frac{1}{m}\tilde{\boldsymbol{H}}^{(1)\top} \tilde{\boldsymbol{H}}^{(2)}$
$\boldsymbol{W}^{(1)} \leftarrow \boldsymbol{W}^{(1)} + \epsilon\Delta_{\boldsymbol{W}^{(1)}}$ (this is a cartoon illustration, in practice use a more effective algorithm, such as momentum with a decaying learning rate)
$\boldsymbol{W}^{(2)} \leftarrow \boldsymbol{W}^{(2)} + \epsilon\Delta_{\boldsymbol{W}^{(2)}}$
end while

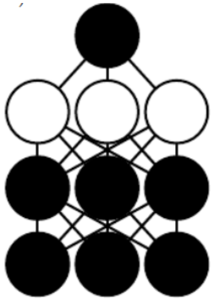
Layer-wise Pretraining

- Training a DBM using stochastic maximum likelihood usually results in failure
- Various techniques that permit joint training have been developed
  - Popular method: greedy layer-wise pretraining
    - Each layer is trained in isolation as an RBM
    - First layer is trained to model input data
    - Each subsequent RBM is trained to model samples from the previous RBM's posterior distribution
    - All the RBMs are combined to form a DBM
    - The DBM may then be trained with PCD
    - PCD makes small change in parameters/performance

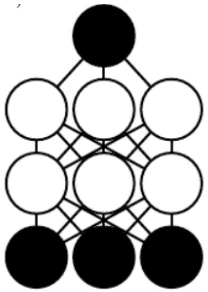
DBM Training to classify MNIST



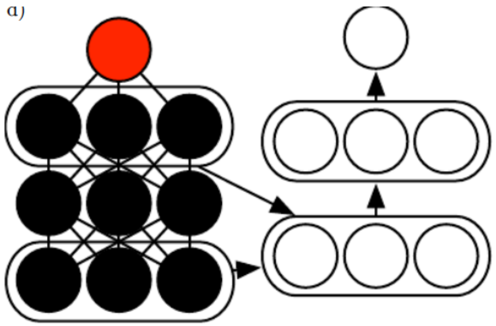
Train an RBM using CD to approximately maximize $\log\, p(\boldsymbol{v})$



Train an second RBM that models $\boldsymbol{h}^{(1)}$ and target class y using CD-k to appropriately maximize $\log\, p(\boldsymbol{h}^{(1)}, y)$ where $\boldsymbol{h}^{(1)}$ is drawn from the first RBMs posterior conditioned on the data. Increase k from 1 to 20 during learning.



Combine the two RBMs into a DBM. Train it to approximately maximize $\log\, P(\boldsymbol{v}, \mathrm{y})$ using stochastic maximum likelihood with k=5



Delete y from the model. Define a new set of features $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(2)}$ that are obtained by running mean field inference in the model lacking y. Use these features as input to an MLP whose structure is the same as an additional pass of a mean field, with and additional output layer for the estimate of y. Initialize the MLP's weights to be the same as the DBM.s weights. Train the MLP to approximately maximize $\log\, P(\mathrm{y}|\boldsymbol{v})$ using SGD and dropout

Jointly Training DBMs

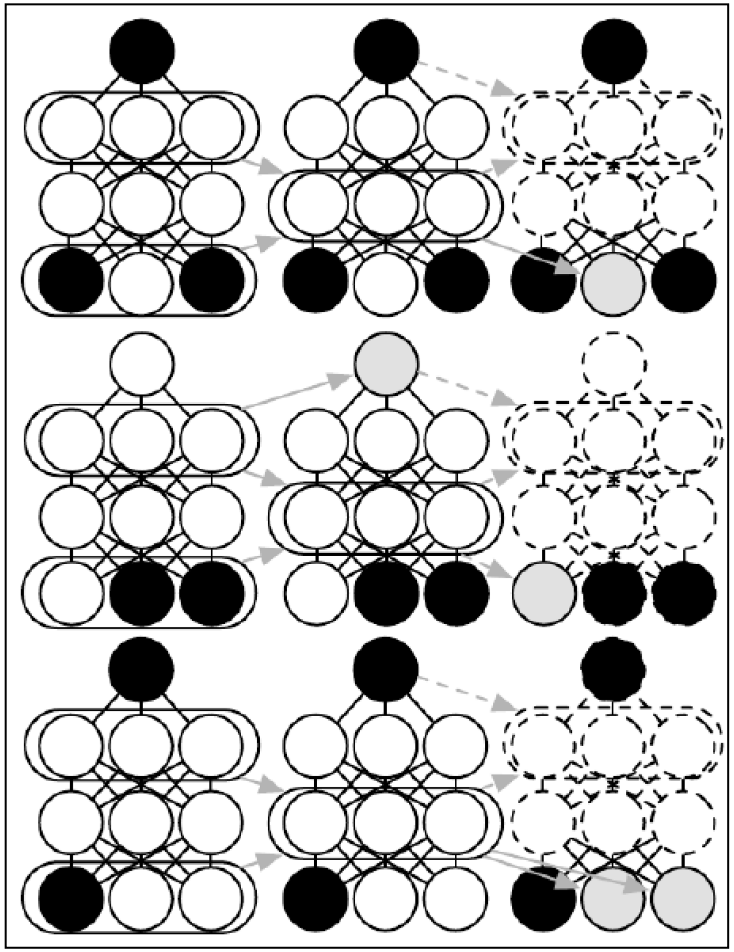
- DBMs require greedy unsupervised pretraining
- To perform classification well, require a separate MLP-based classifier on top of the hidden features they extract
- This has undesirable properties
  - Hard to track performance during training
    - We cannot evaluate performance of full DBM while training first RBM– cannot track performance of hyperparameters
  - Software implementations of DBMs
    - require many different CD training of RBMs, PCD training of full DBM

Resolving the joint training problem

- Two ways to resolve joint training of DBMs
  1. Centered DBM
     - Reparameterize the model in order to make the Hessian of the cost function better conditioned at the beginning of the learning process
  2. Multiprediction DBM (MP-DBM)
     - Works by viewing the mean field equations as defining a family of recurrent networks for approximately solving every possible inference problem

Multiprediction Training of DBM

- Each row indicates a different example within a minibatch for the same training step
- Each column represents a time step within the mean-field process
- For each example, we sample a subset of the data variables to serve as inputs to the inference process
  - They are shaded black to indicate conditioning



We then run the mean field inference process, with arrows indicating which variables influence which other variables in the process