

Stacks of Restricted Boltzmann Machines

Honglak Lee

CSE division, EECS department
University of Michigan, Ann Arbor

8/5/2015

Deep Learning Summer School @ Montreal

Restricted Boltzmann Machines

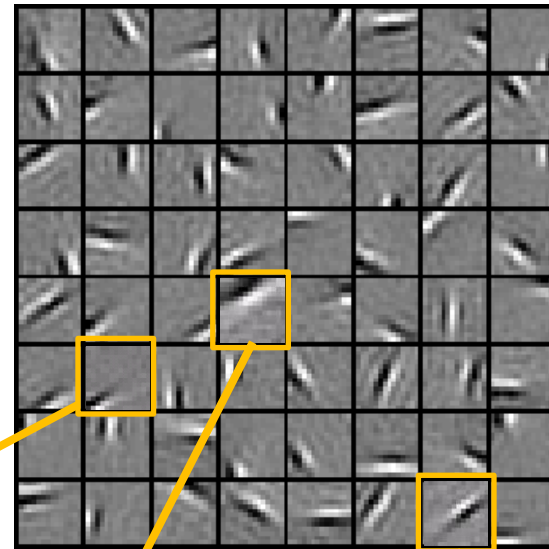
Learning Feature Hierarchy

[Lee et al., NIPS 2007; Ranzato et al., 2007]

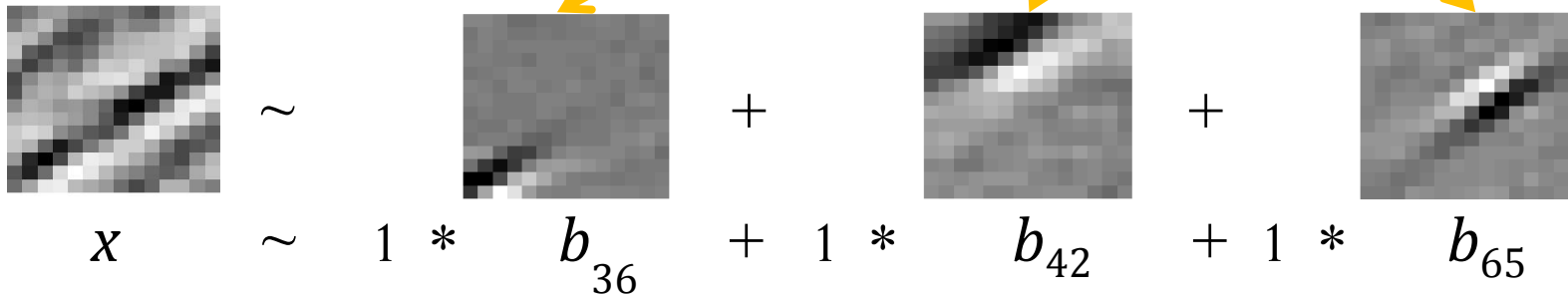
Natural Images



Learned bases: "Edges"



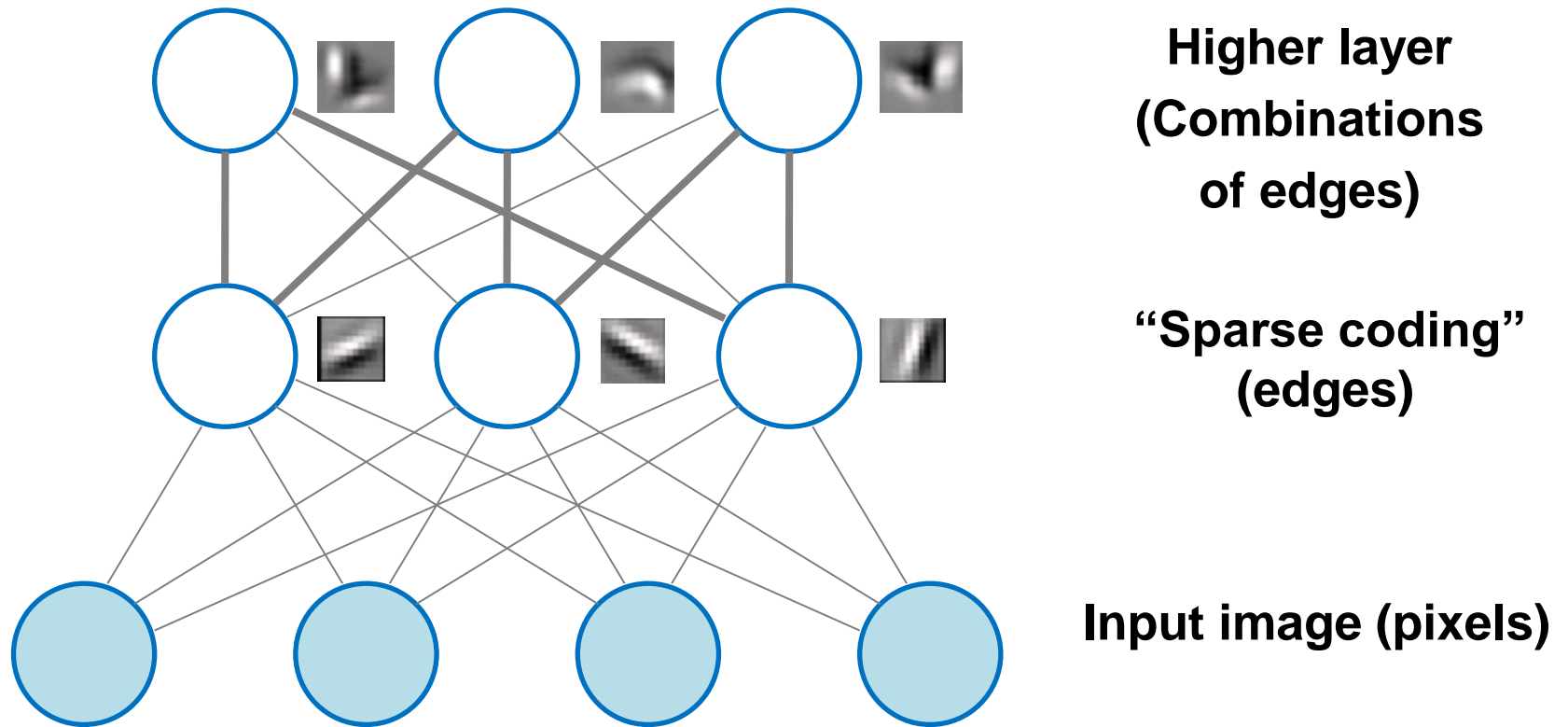
Test example



$[0, 0, \dots, 0, \mathbf{1}, 0, \dots, 0, \mathbf{1}, 0, \dots, 0, \mathbf{1}, \dots]$
= coefficients (feature representation)

Compact & easily interpretable

Learning Feature Hierarchy



Lee et al., NIPS 2007: DBN (Hinton et al., 2006) with additional sparseness constraint.

[Related work: Bengio et al., 2006; Ranzato et al., 2007, and others.]

Restricted Boltzmann Machines (RBMs)

- Representation

- Undirected bipartite graphical model

- $\mathbf{v} \in \{0,1\}^D$: observed (visible) binary variables

- $\mathbf{h} \in \{0,1\}^N$: hidden binary variables.

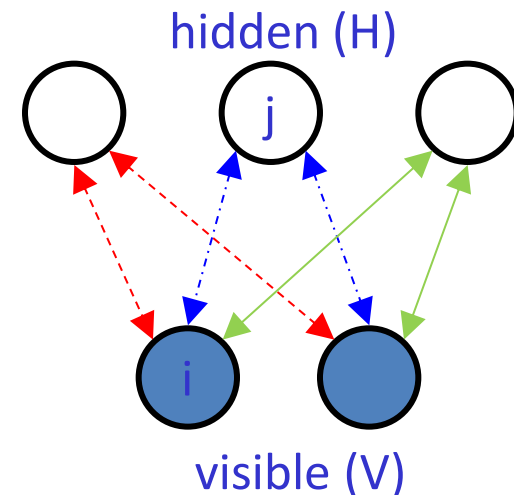
$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h}))$$

$$E(\mathbf{v}, \mathbf{h}) = - \sum_{ij} v_i W_{ij} h_j - \sum_j b_j h_j - \sum_i c_i v_i$$

$$= -\mathbf{v}^T \mathbf{W} \mathbf{h} - \mathbf{b}^T \mathbf{h} - \mathbf{c}^T \mathbf{v}$$

$$= -h_1(\mathbf{w}_1^T \mathbf{v}) + h_2(\mathbf{w}_2^T \mathbf{v}) + h_3(\mathbf{w}_3^T \mathbf{v}) - \mathbf{b}^T \mathbf{h} - \mathbf{c}^T \mathbf{v}$$

$$Z = \sum_{\mathbf{v} \in \{0,1\}^D} \sum_{\mathbf{h} \in \{0,1\}^N} \exp(-E(\mathbf{v}, \mathbf{h}))$$



Restricted Boltzmann Machines (RBMs)

- Representation

- Undirected bipartite graphical model

- $\mathbf{v} \in \{0,1\}^D$: observed (visible) binary variables

- $\mathbf{h} \in \{0,1\}^N$: hidden binary variables.

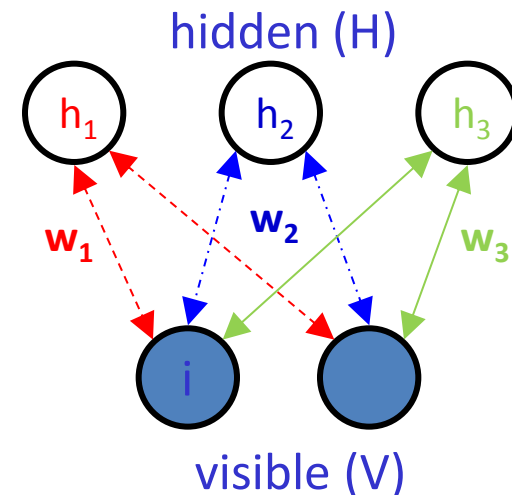
$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h}))$$

$$E(\mathbf{v}, \mathbf{h}) = - \sum_{ij} v_i W_{ij} h_j - \sum_j b_j h_j - \sum_i c_i v_i$$

$$= -\mathbf{v}^T \mathbf{W} \mathbf{h} - \mathbf{b}^T \mathbf{h} - \mathbf{c}^T \mathbf{v}$$

$$= -h_1(\mathbf{w}_1^T \mathbf{v}) + h_2(\mathbf{w}_2^T \mathbf{v}) + h_3(\mathbf{w}_3^T \mathbf{v}) - \mathbf{b}^T \mathbf{h} - \mathbf{c}^T \mathbf{v}$$

$$Z = \sum_{\mathbf{v} \in \{0,1\}^D} \sum_{\mathbf{h} \in \{0,1\}^N} \exp(-E(\mathbf{v}, \mathbf{h}))$$



Conditional Probabilities

(RBM with binary-valued input data)

- Given \mathbf{v} , all the h_j are conditionally independent

$$P(h_j = 1 | \mathbf{v}) = \frac{\exp(\sum_i W_{ij} v_i + b_j)}{\exp(\sum_i W_{ij} v_i + b_j) + 1}$$

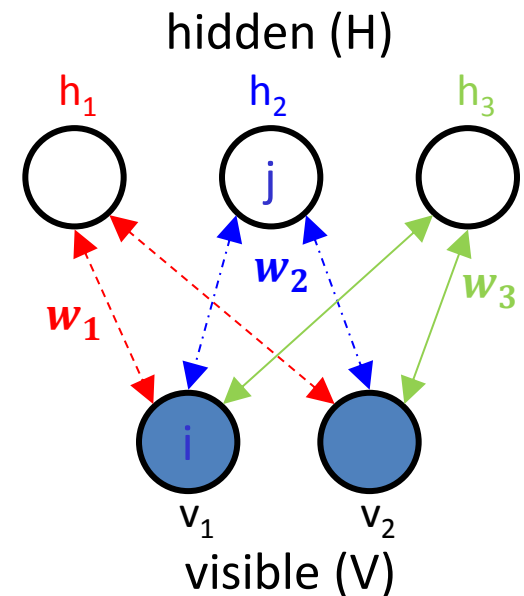
$$= \text{sigmoid}(\sum_i W_{ij} v_i + b_j)$$

$$= \text{sigmoid}(\mathbf{w}_j^T \mathbf{v} + b_j)$$

– $P(\mathbf{h} | \mathbf{v})$ can be used as “features”

- Given \mathbf{h} , all the v_i are conditionally independent

$$P(v_i | \mathbf{h}) = \text{sigmoid}(\sum_j W_{ij} h_j + c_i)$$

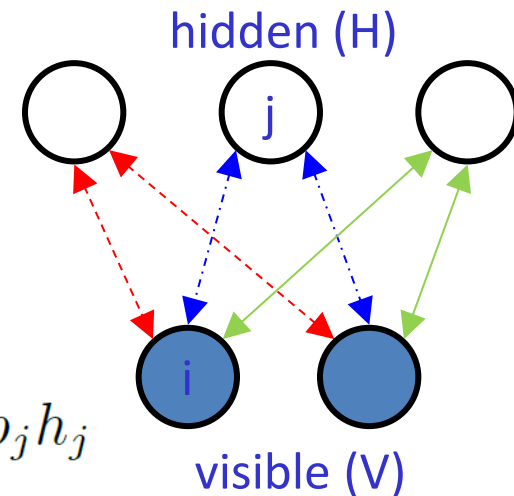


RBM with real-valued input data

- Representation
 - Undirected bipartite graphical model
 - V: observed (visible) real variables
 - H: hidden binary variables.

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h}))$$

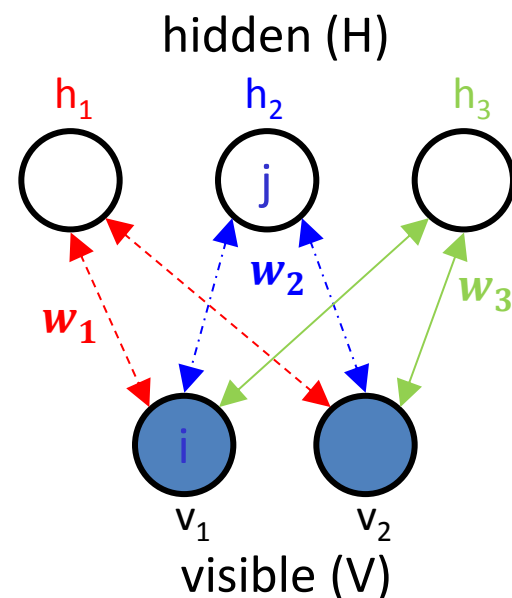
$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2\sigma^2} \sum_i (v_i - c_i)^2 - \frac{1}{\sigma} \sum_{i,j} v_i W_{ij} h_j - \sum_j b_j h_j$$



RBM with real-valued input data

- Given \mathbf{v} , all the h_j are conditionally independent

$$\begin{aligned} P(h_j = 1 | \mathbf{v}) &= \frac{\exp(\frac{1}{\sigma} \sum_i W_{ij} v_i + b_j)}{\exp(\frac{1}{\sigma} \sum_i W_{ij} v_i + b_j) + 1} \\ &= \text{sigmoid}(\frac{1}{\sigma} \sum_i W_{ij} v_i + b_j) \\ &= \text{sigmoid}(\frac{1}{\sigma} \mathbf{w}_j^T \mathbf{v} + b_j) \\ &\text{– } P(\mathbf{h} | \mathbf{v}) \text{ can be used as “features”} \end{aligned}$$



- Given \mathbf{h} , all the v_i are conditionally independent

$$P(v_i | \mathbf{h}) = \mathcal{N}(\sigma \sum_j W_{ij} h_j + c_i, \sigma^2)$$

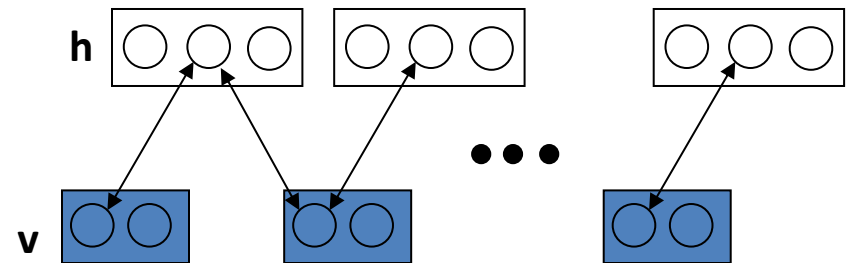
$$P(\mathbf{v} | \mathbf{h}) = \mathcal{N}(\sigma \mathbf{W} \mathbf{h} + \mathbf{c}, \sigma^2 \mathbf{I})$$

Inference

- Conditional Distribution: $P(\mathbf{v}|\mathbf{h})$ or $P(\mathbf{h}|\mathbf{v})$
 - Easy to compute (see previous slides).
 - Due to conditional independence, we can sample hidden units in parallel given visible units (& vice versa)
- Joint Distribution: $P(\mathbf{v},\mathbf{h})$
 - Requires Gibbs Sampling (approximate)

```
Initialize with  $\mathbf{v}^0$ 
Sample  $\mathbf{h}^0$  from  $P(\mathbf{h}|\mathbf{v}^0)$ 

Repeat until convergence ( $t=1,\dots$ ) {
    Sample  $\mathbf{v}^t$  from  $P(\mathbf{v}^t|\mathbf{h}^{t-1})$ 
    Sample  $\mathbf{h}^t$  from  $P(\mathbf{h}|\mathbf{v}^t)$ 
}
```



Training RBMs

- Maximum likelihood training
 - Objective: Log-likelihood of the training data

$$L = \sum_{m=1}^M \log P(\mathbf{v}^{(m)}) = \sum_{m=1}^M \log \sum_{\mathbf{h}} P(\mathbf{v}^{(m)}, \mathbf{h})$$

$$\text{where } P_{\theta}(\mathbf{v}^{(m)}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}^{(m)}, \mathbf{h}; \theta))$$


- Computing exact gradient intractable.
- Typically sampling-based approximation is used (e.g., contrastive divergence).
- Usually optimized via stochastic gradient descent


$$\theta^{new} := \theta^{old} + \eta \frac{\partial \log P(\mathbf{v})}{\partial \theta}$$

Training RBMs

- Model: $P_{\theta}(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h}; \theta))$
- How can we find parameters θ that maximize $P_{\theta}(\mathbf{v})$?

$$\frac{\partial}{\partial \theta} \log P(\mathbf{v}) = \mathbb{E}_{\mathbf{h} \sim P_{\theta}(\mathbf{h}|\mathbf{v})} \left[-\frac{\partial}{\partial \theta} E(\mathbf{h}, \mathbf{v}) \right] - \mathbb{E}_{\mathbf{v}', \mathbf{h} \sim P_{\theta}(\mathbf{v}, \mathbf{h})} \left[-\frac{\partial}{\partial \theta} E(\mathbf{h}', \mathbf{v}) \right]$$

 Data Distribution
(posterior of h given v)

 Model Distribution

- We need to compute $P(\mathbf{h}|\mathbf{v})$ and $P(\mathbf{v}, \mathbf{h})$, and derivative of E w.r.t. parameters $\{W, b, c\}$
 - $P(\mathbf{h}|\mathbf{v})$: tractable (see previous slides)
 - $P(\mathbf{v}, \mathbf{h})$: intractable
 - Can approximate with Gibbs sampling, but requires lots of iterations

Contrastive Divergence

- Approximation of the log-likelihood gradient
 1. Replace the average over all possible inputs by samples

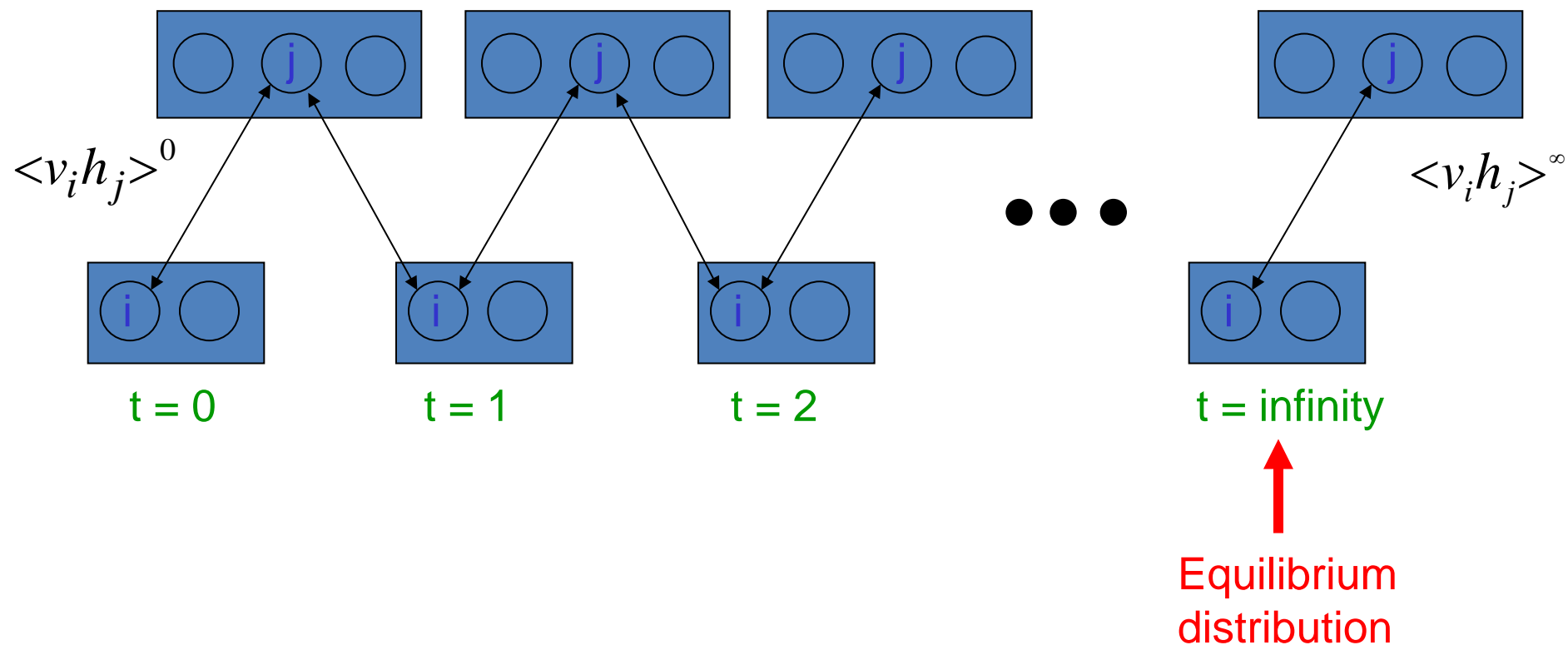
$$\frac{\partial}{\partial \theta} \log P(\mathbf{v}) = \mathbb{E}_{\mathbf{h} \sim P_{\theta}(\mathbf{h}|\mathbf{v})} \left[-\frac{\partial}{\partial \theta} E(\mathbf{h}, \mathbf{v}) \right] - \mathbb{E}_{\mathbf{v}', \mathbf{h} \sim P_{\theta}(\mathbf{v}, \mathbf{h})} \left[-\frac{\partial}{\partial \theta} E(\mathbf{h}', \mathbf{v}) \right]$$

2. Run the MCMC chain (Gibbs sampling) for only k steps starting from the observed example

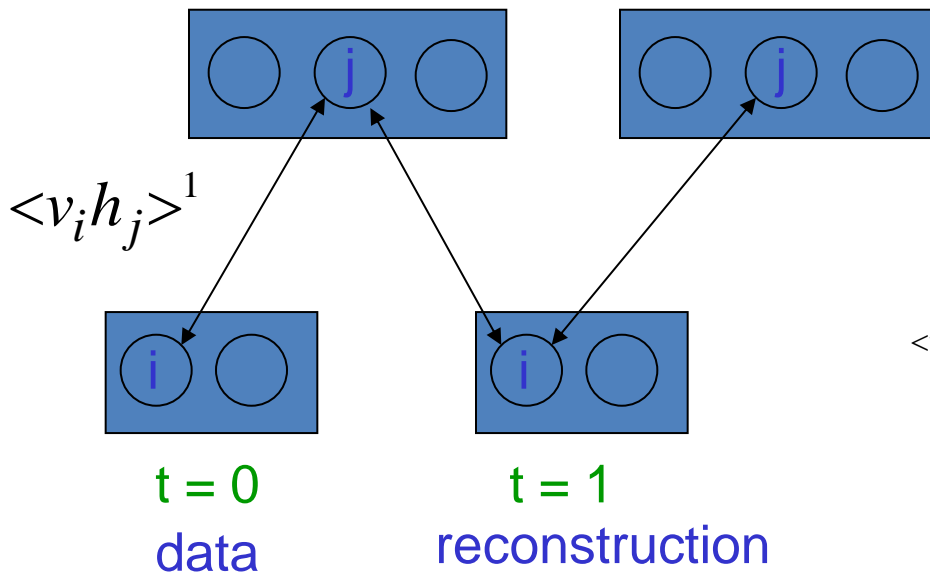
```
Initialize with  $\mathbf{v}^0 = \mathbf{v}$ 
Sample  $\mathbf{h}^0$  from  $P(\mathbf{h}|\mathbf{v}^0)$ 

For  $t = 1, \dots, k$  {
    Sample  $\mathbf{v}^t$  from  $P(\mathbf{v}^t|\mathbf{h}^{t-1})$ 
    Sample  $\mathbf{h}^t$  from  $P(\mathbf{h}|\mathbf{v}^t)$ 
}
```

Maximum likelihood learning for RBM



Contrastive divergence to learn RBM



Start with a training vector on the visible units.

Update all the hidden units in parallel

Update the all the visible units in parallel to get a “reconstruction”.

Update the hidden units again.

Update rule: Putting together

- Training via stochastic gradient.

- Note, $\frac{\partial E}{\partial W_{ij}} = h_i v_j$. Therefore,

$$\begin{aligned}\frac{\partial}{\partial W_{ij}} \log P(\mathbf{v}) &= \mathbb{E}_{\mathbf{h} \sim P_{\theta}(\mathbf{h}|\mathbf{v})} [v_i h_j] - \mathbb{E}_{\mathbf{v}', \mathbf{h} \sim P_{\theta}(\mathbf{v}, \mathbf{h})} [v_i h_j] \\ &\approx v_i P(h_j|\mathbf{v}) - v_i^k P(h_j|\mathbf{v}^k)\end{aligned}$$

- where \mathbf{v}^k is a sample from k-step CD
- Can derive similar update rule for biases b and c
- Mini-batch (~ 100 samples) are used to reduce the variance of the gradient estimate
- Implemented in ~ 10 lines of matlab code

Other ways of training RBMs

- **Persistent CD** (Tieleman, ICML 2008; Tieleman & Hinton, ICML 2009)
 - Keep a background MCMC chain to obtain the negative phase samples.
 - Related to Stochastic Approximation
 - Robbins and Monro, Ann. Math. Stats, 1957
 - L. Younes, Probability Theory 1989
- **Score Matching** (Swersky et al., ICML 2011; Hyvarinen, JMLR 2005)
 - Use score function to eliminate Z
 - Match model's & empirical score function

$$p(x) = q(x)/Z \quad \psi = \frac{\partial \log p(x)}{\partial x} = \frac{\partial \log q(x)}{\partial x}$$

Variants of RBMs

Sparse RBM / DBN

- Main idea [Lee et al., NIPS 2008]
 - Constrain the hidden layer nodes to have “sparse” average values (activation). [cf. sparse coding]
- Optimization
 - Tradeoff between “likelihood” and “sparsity penalty”

$$\text{minimize}_{\{W,b,c\}} \underbrace{- \sum_{k=1}^m \log P(\mathbf{v}^{(k)})}_{\text{Log-likelihood}} + \underbrace{\phi(W, b, c)}_{\text{Sparsity penalty}}$$

$$\text{where } \phi \triangleq \lambda \sum_{j=1}^n d(\mathbb{E}_{\text{sample}}[h_j | \mathbf{v}], p)$$

$d(\cdot, \cdot)$: penalty function on deviation from the target sparsity (e.g., KL-divergence)

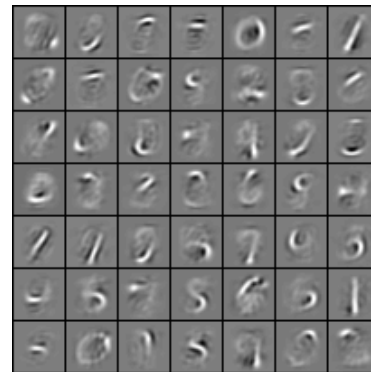
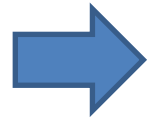
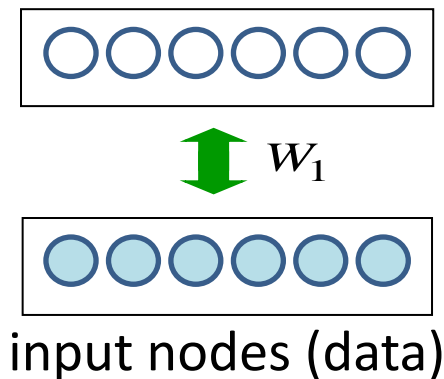
Average activation (over training data)

Target sparsity

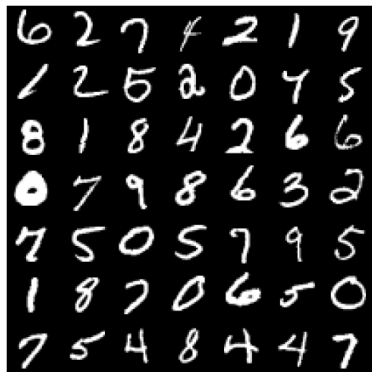
$$\mathbb{E}_{\text{sample}}[h_j | \mathbf{v}] = \frac{1}{m} \sum_{k=1}^m P(h_j^{(k)} = 1 | \mathbf{v}^{(k)})$$

Modeling handwritten digits

- Sparse dictionary learning via sparse RBMs



First layer bases
("pen-strokes")



Training examples

Learned sparse representations can be used as features. In practice, sparsity regularization is very often used.

[Lee et al., NIPS 2008; Ranzato et al, NIPS 2007]

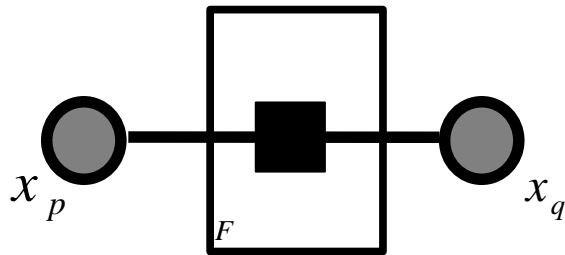
3-way factorized RBM

[Ranzato et al., AISTATS 2010; Ranzato and Hinton, CVPR 2010]

- Models the covariance structure of images using hidden variables
 - 3-way factorized RBM / mean-covariance RBM (mcRBM)

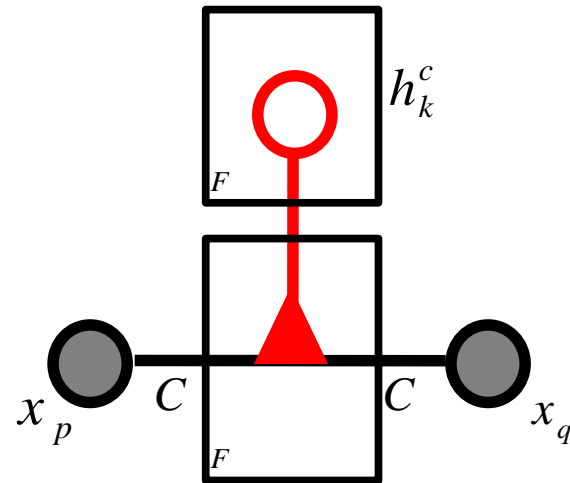
Gaussian MRF

$$E(\mathbf{x}, \mathbf{h}) = \frac{1}{2} \mathbf{x}^T \Sigma^{-1} \mathbf{x}$$



3-way (gated) RBM

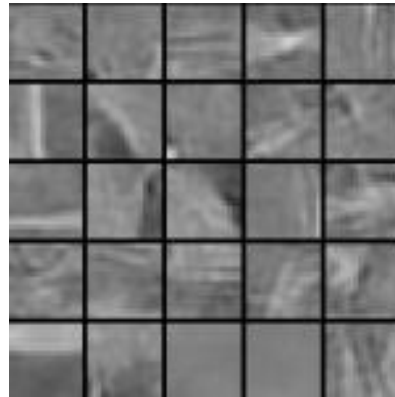
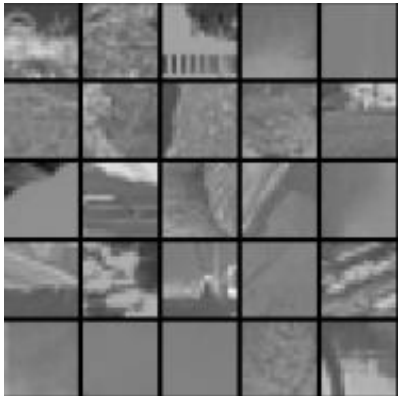
$$E(\mathbf{x}, \mathbf{h}) = \frac{1}{2} \mathbf{x}^T C [\text{diag}(P\mathbf{h})] C^T \mathbf{x}$$



[Slide Credit: Marc'Aurelio Ranzato]

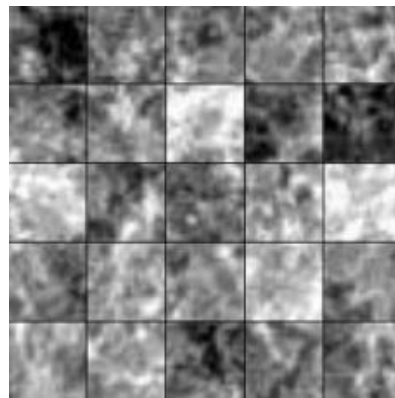
Generating natural image patches

Natural images



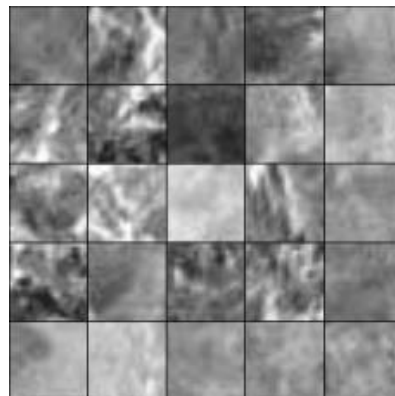
mcRBM

Ranzato and Hinton CVPR 2010



GRBM

from Osindero and Hinton NIPS 2008



S-RBM + DBN

from Osindero and Hinton NIPS 2008

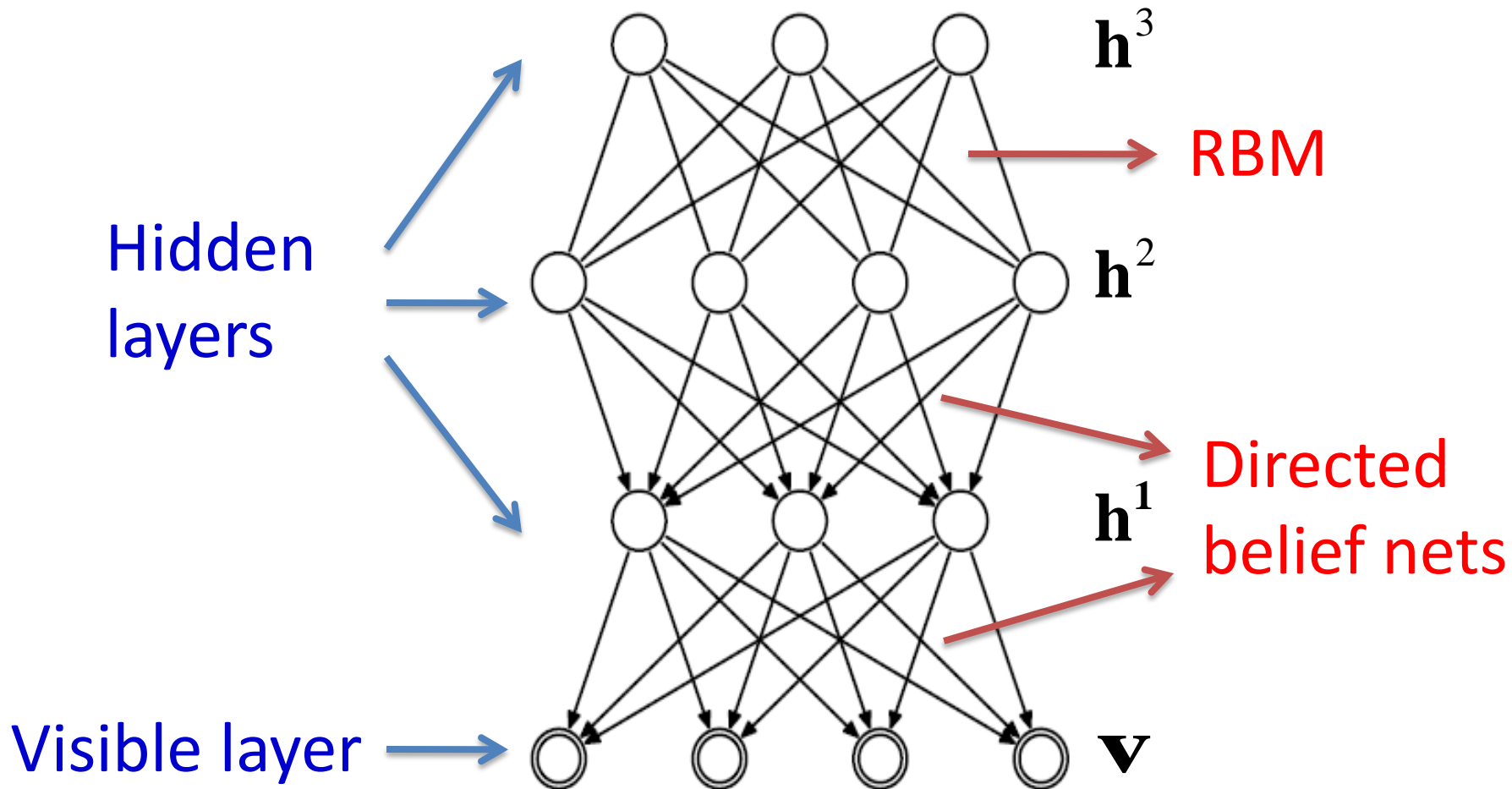
Stacking of RBMs as Deep Belief Networks

Deep Belief Networks (DBNs)

[Hinton et al., 2006]

- Probabilistic generative model
 - With deep architecture (multiple layers)
- Unsupervised pre-training with RBMs provides a good initialization of the model
 - **Theoretically justified** as maximizing the lower-bound of the log-likelihood of the data
- Supervised fine-tuning
 - Generative: Up-down algorithm
 - Discriminative: backpropagation (convert to NN)

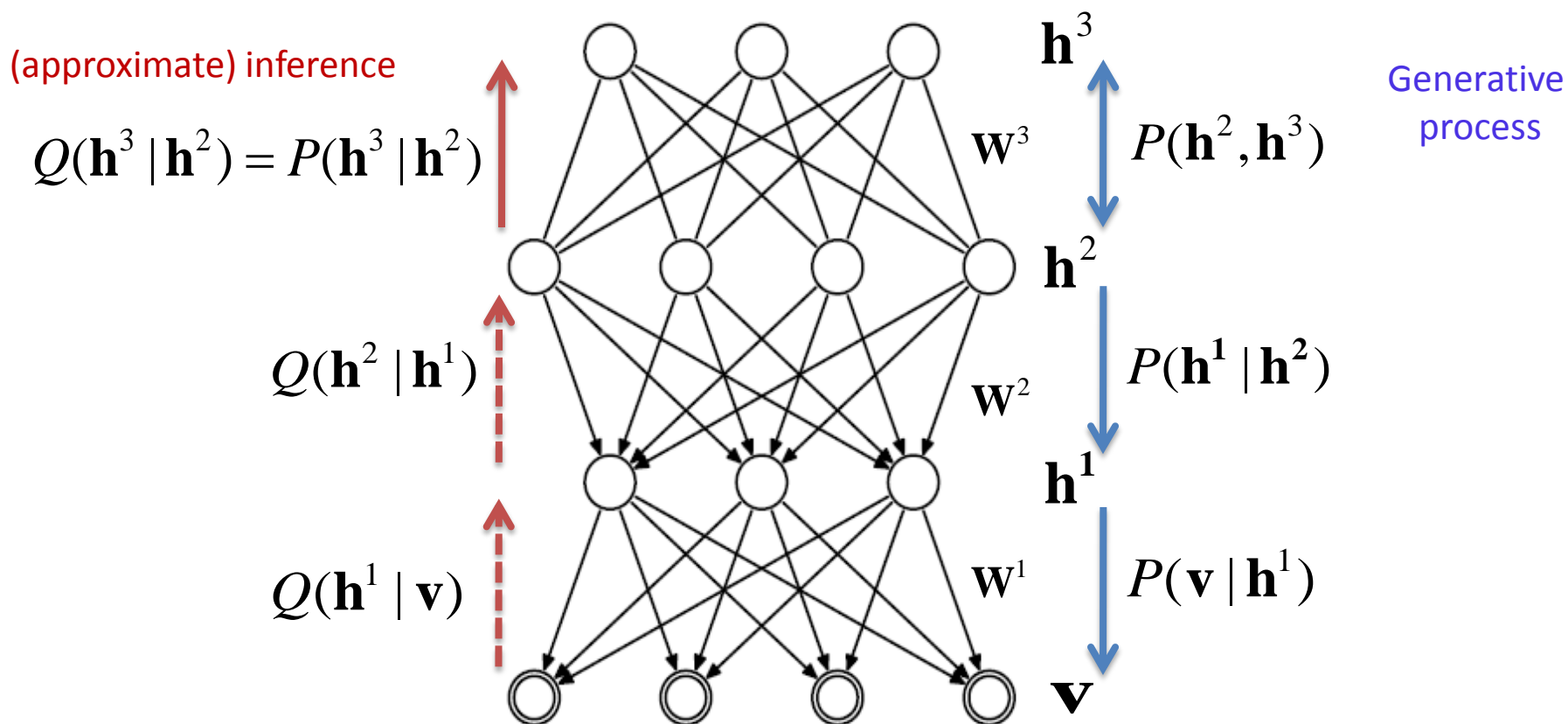
Deep Belief Networks (DBN)



$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \dots, \mathbf{h}^l) = P(\mathbf{v} | \mathbf{h}^1) P(\mathbf{h}^1 | \mathbf{h}^2) \dots P(\mathbf{h}^{l-2} | \mathbf{h}^{l-1}) P(\mathbf{h}^{l-1}, \mathbf{h}^l)$$

DBN structure

Hinton et al., 2006



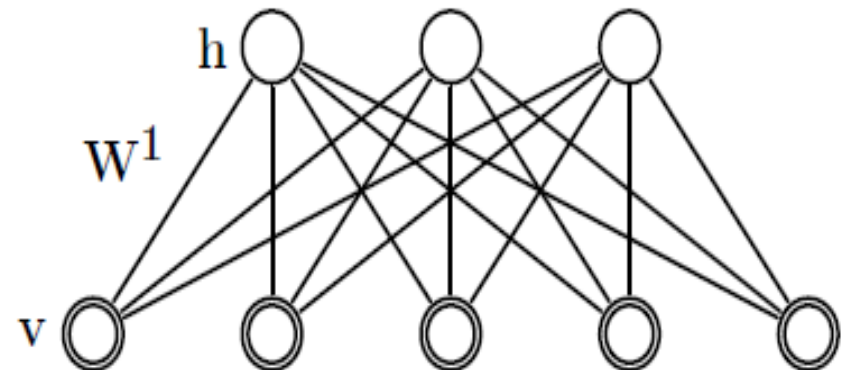
$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \dots, \mathbf{h}^l) = P(\mathbf{v} | \mathbf{h}^1) P(\mathbf{h}^1 | \mathbf{h}^2) \dots P(\mathbf{h}^{l-2} | \mathbf{h}^{l-1}) P(\mathbf{h}^{l-1}, \mathbf{h}^l)$$

$$Q(\mathbf{h}^i | \mathbf{h}^{i-1}) = \prod_j \text{sigm}(\mathbf{b}_j^{i-1} + \mathbf{W}_j^i \mathbf{h}^{i-1}) \quad P(\mathbf{h}^{i-1} | \mathbf{h}^i) = \prod_j \text{sigm}(\mathbf{b}_j^i + \mathbf{W}_j^i \mathbf{h}^i)$$

DBN Greedy training

Hinton et al., 2006

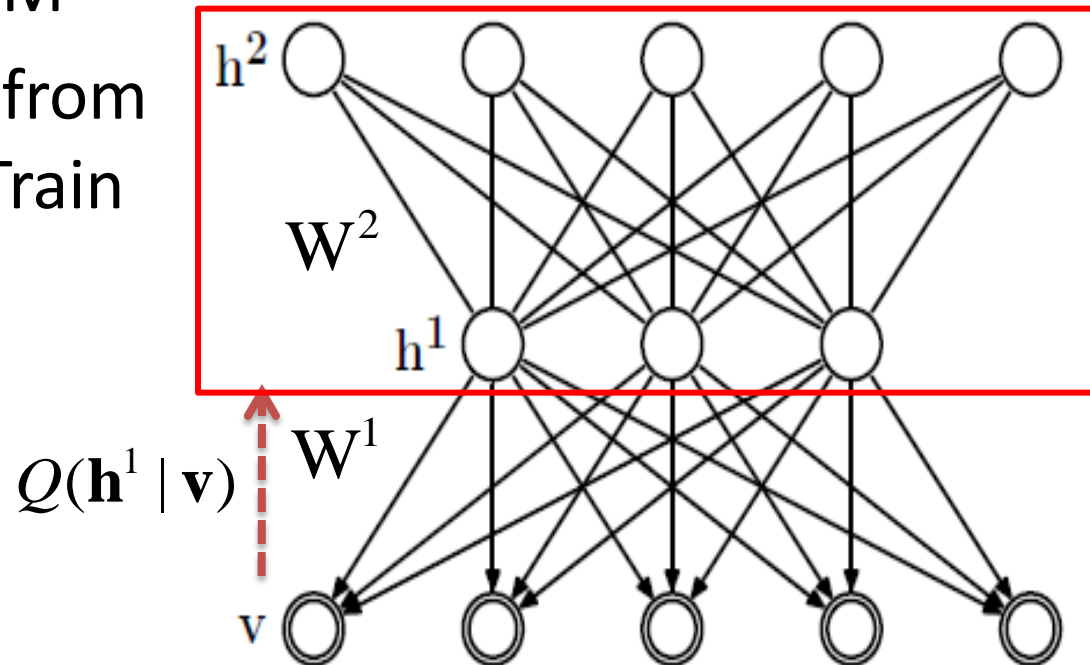
- **First step:**
 - Construct an RBM with an input layer \mathbf{v} and a hidden layer \mathbf{h}
 - Train the RBM



DBN Greedy training

Hinton et al., 2006

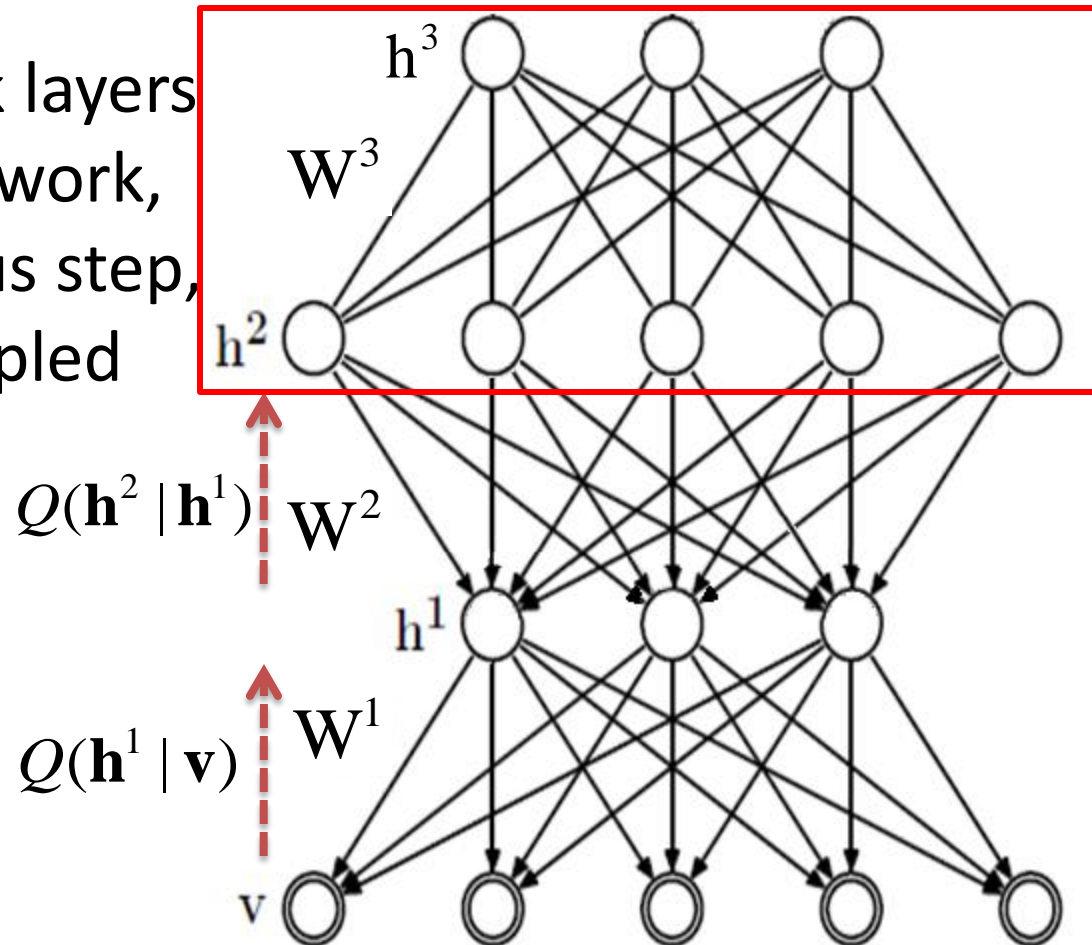
- **Second step:**
 - Stack another hidden layer on top of the RBM to form a new RBM
 - Fix W^1 , sample \mathbf{h}^1 from $Q(\mathbf{h}^1 | \mathbf{v})$ as input. Train W^2 as RBM.



DBN Greedy training

Hinton et al., 2006

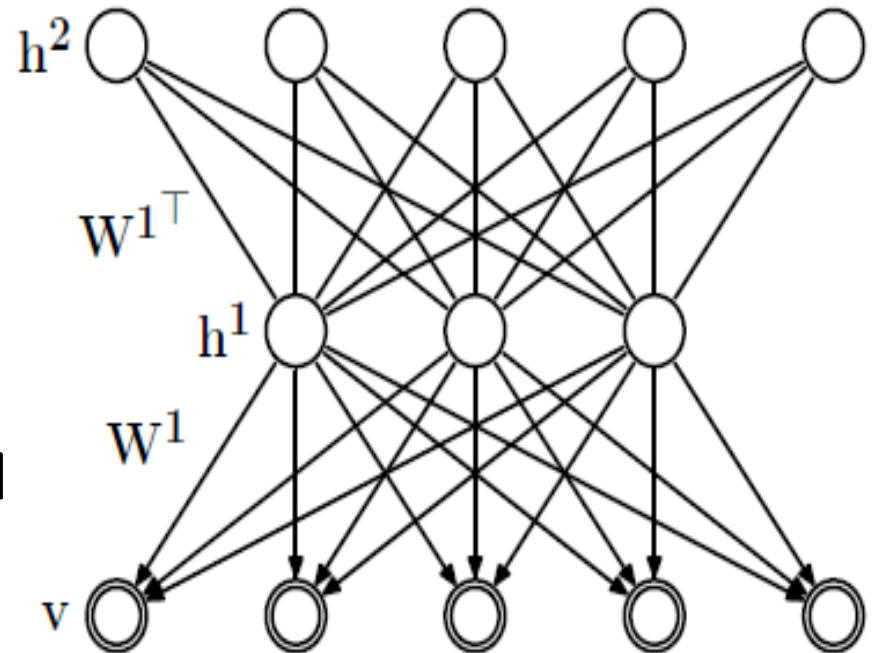
- Third step:
 - Continue to stack layers on top of the network, train it as previous step, with sample sampled from $Q(\mathbf{h}^2 | \mathbf{h}^1)$
- And so on...



Why greedy training works?

Hinton et al., 2006

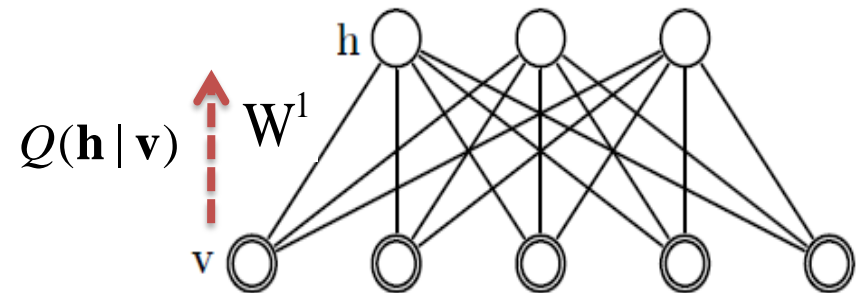
- RBM specifies $P(v, h)$ from $P(v|h)$ and $P(h|v)$
 - Implicitly defines $P(v)$ and $P(h)$
- Key idea of stacking
 - Keep $P(v|h)$ from 1st RBM
 - Replace $P(h)$ by the distribution generated by 2nd level RBM



Why greedy training works?

Hinton et al., 2006

- Theoretical Justification
 - Variational lower-bound of the log-likelihood improves with greedy layerwise training of RBMs



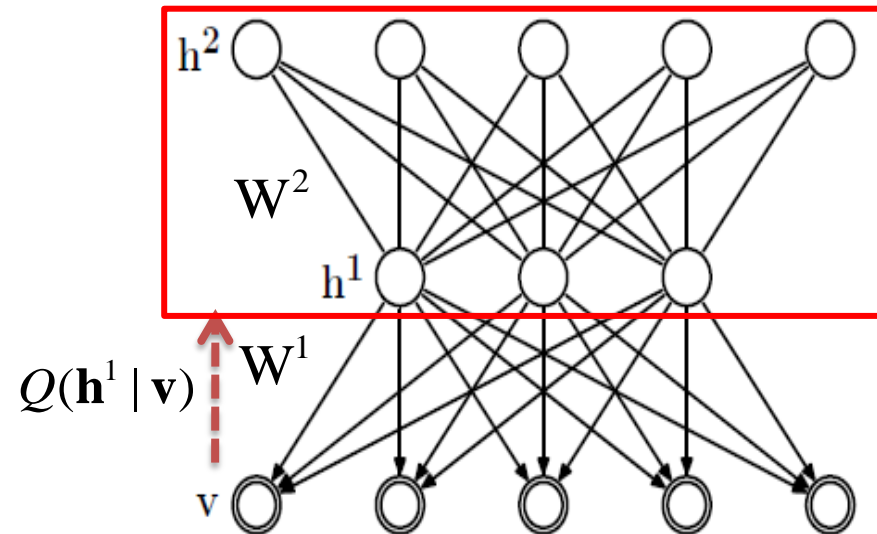
$$\log P(\mathbf{x}) \geq H_{Q(\mathbf{h}|\mathbf{x})} + \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{x}) (\log P(\mathbf{h}) + \log P(\mathbf{x}|\mathbf{h}))$$

Trained by the second layer RBM

Why greedy training works?

Hinton et al., 2006

- Theoretical Justification
 - Variational lower-bound of the log-likelihood improves with greedy layerwise training of RBMs
 - Note: RBM and 2-layer DBN are equivalent when $W^2 = (W^1)^T$. Therefore, the lower bound is tight and the log-likelihood improves by greedy training.



$$\log P(\mathbf{x}) \geq H_{Q(\mathbf{h}|\mathbf{x})} + \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{x}) (\log P(\mathbf{h}) + \log P(\mathbf{x}|\mathbf{h}))$$

Trained by the second layer RBM

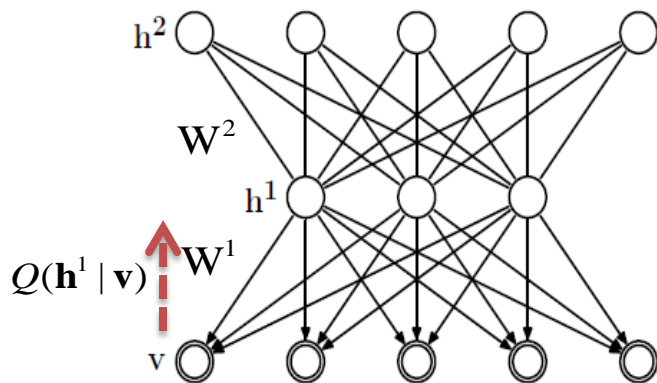
Derivation

- Variational Bound (true for any probabilistic model)

$$\begin{aligned} \log P(\mathbf{v}) &= \sum_{\mathbf{h}} Q(\mathbf{h}) \log P(\mathbf{h}, \mathbf{v}) - \sum_{\mathbf{h}} Q(\mathbf{h}) \log Q(\mathbf{h}) + KL(Q(\mathbf{h}) \| P(\mathbf{h} | \mathbf{v})) \\ &\geq \sum_{\mathbf{h}} Q(\mathbf{h}) \log P(\mathbf{h}, \mathbf{v}) - \sum_{\mathbf{h}} Q(\mathbf{h}) \log Q(\mathbf{h}) \end{aligned}$$

where $Q(h_j) \triangleq Q(h_j | \mathbf{v}) = \sigma(h_j = 1 | \mathbf{v}; W^1)$

- When $W^2 = (W^1)^T$, the RBM and 2-layer DBN are equivalent and the above lower bound is tight



$$\begin{aligned} \log P_{DBN}(\mathbf{v}, \mathbf{h}; W^1, W^2 = W^{1T}) \\ = \log P_{RBM}(\mathbf{v}, \mathbf{h}; W^1) \end{aligned}$$

Derivation

- Variational Bound

$$\log P(\mathbf{v}) \geq \sum_{\mathbf{h}} Q(\mathbf{h}) \log P(\mathbf{h}, \mathbf{v}) - \sum_{\mathbf{h}} Q(\mathbf{h}) \log Q(\mathbf{h})$$

$$\text{where } Q(h_j) \triangleq Q(h_j|\mathbf{v}) = \sigma(h_j = 1|\mathbf{v}; W^1)$$

- When $W^2 = (W^1)^T$, the RBM and 2-layer DBN are equivalent and the above lower bound is tight

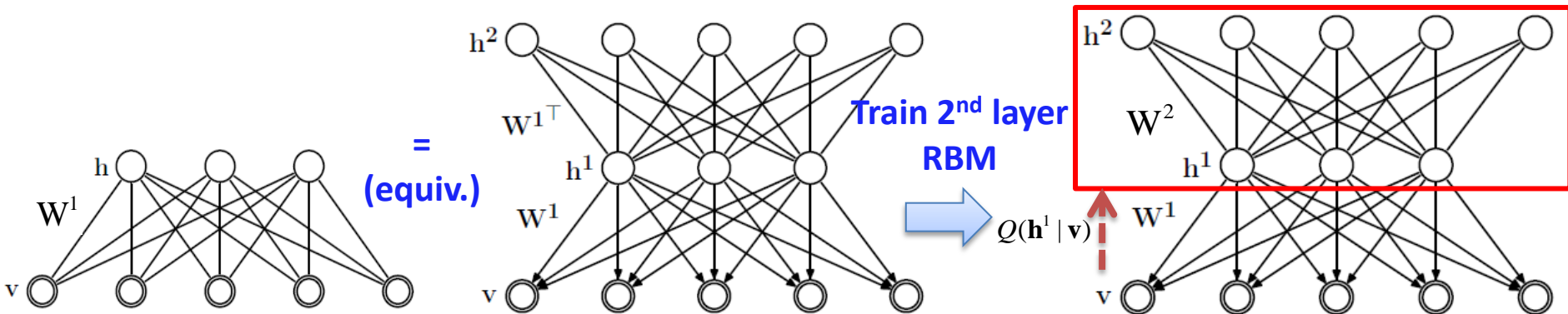
$$\begin{aligned} & \log P_{DBN}(\mathbf{v}; W^1, W^2 = W^{1T}) \\ &= \log P_{RBM}(\mathbf{v}; W^1) \\ &= \sum_{\mathbf{h}^1} Q(\mathbf{h}^1) \log P_{DBN}(\mathbf{h}^1, \mathbf{v}) - \sum_{\mathbf{h}^1} Q(\mathbf{h}^1) \log Q(\mathbf{h}^1) + \underbrace{KL(Q(\mathbf{h}^1) \| P_{DBN}(\mathbf{h}^1|\mathbf{v}))}_{=0} \\ &= \sum_{\mathbf{h}^1} Q(\mathbf{h}^1) \log P_{DBN}(\mathbf{h}^1) + \sum_{\mathbf{h}^1} Q(\mathbf{h}^1) \log P_{DBN}(\mathbf{v}|\mathbf{h}^1) - \sum_{\mathbf{h}^1} Q(\mathbf{h}^1) \log Q(\mathbf{h}^1) \end{aligned}$$

Training the second layer “RBM” increases this “log-likelihood” of first layer hidden units

$$\leq \log P_{DBN}(\mathbf{v}; W^1, W'^2) \quad \text{After training the second layer RBM}$$

Theoretical Justification (summary)

- For the second layer DBN, the log-likelihood actually improves by greedy training. (Hinton et al., 2006)
- Variational lower-bound of the log-likelihood improves with greedy layerwise training of RBMs (for all layers)



$$\log P(\mathbf{x}) \geq H_{Q(\mathbf{h}|\mathbf{x})} + \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{x}) (\log P(\mathbf{h}) + \log P(\mathbf{x}|\mathbf{h}))$$

Trained by the second layer RBM

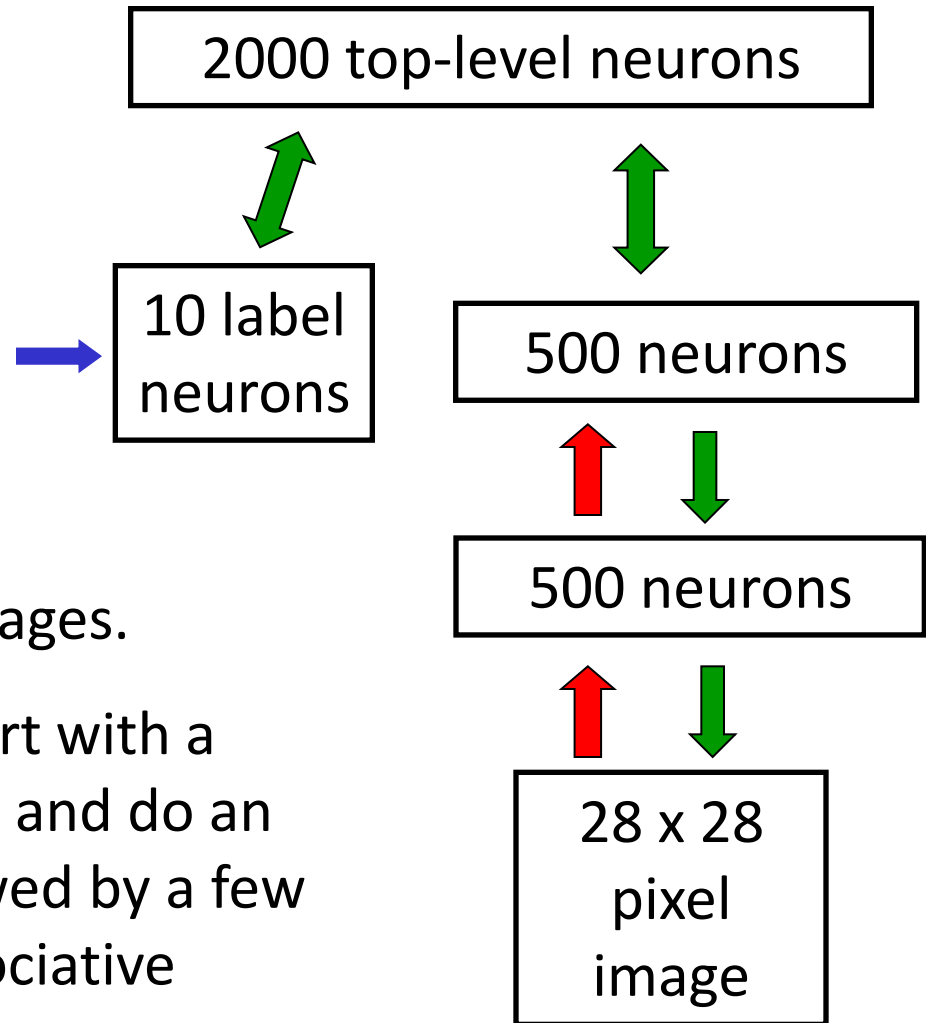
DBN and supervised fine-tuning

- Discriminative fine-tuning
 - Initializing with neural nets + backpropagation
 - Maximizes $\log P(Y | X)$ (X: data Y: label)
- Generative fine-tuning
 - Up-down algorithm
 - Maximizes $\log P(Y, X)$ (joint likelihood of data and labels)
- Hinton et al. used supervised + generative fine-tuning in their Neural Computation paper. However, it is possible to use unsupervised + generative fine-tuning as well.

A model for digit recognition

The top two layers form an associative memory

The energy valleys have names



The model learns to generate combinations of labels and images.

To perform recognition we start with a neutral state of the label units and do an up-pass from the image followed by a few iterations of the top-level associative memory.

Generative fine-tuning via Up-down algorithm

After pre-training many layers of features, we can fine-tune the features to improve generation.

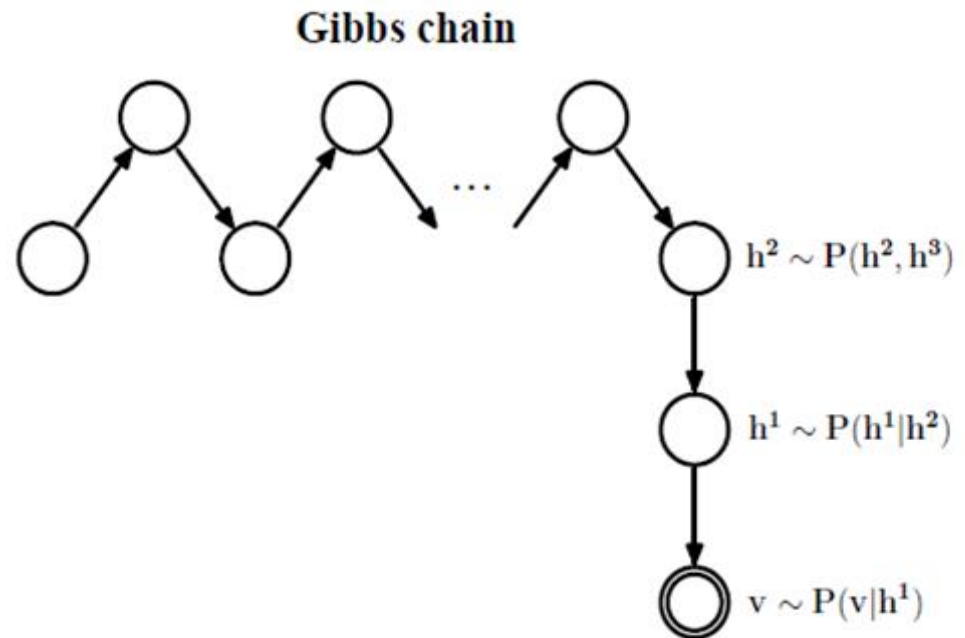
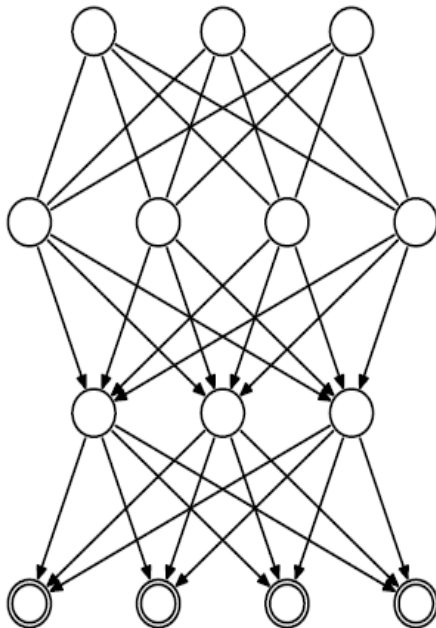
1. Do a stochastic bottom-up pass
 - Adjust the top-down weights to be good at reconstructing the feature activities in the layer below.
2. Do a few iterations of sampling in the top level RBM
 - Adjust the weights in the top-level RBM.
3. Do a stochastic top-down pass
 - Adjust the bottom-up weights to be good at reconstructing the feature activities in the layer above.

Generating sample from a DBN

- Want to sample from

$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \dots, \mathbf{h}^l) = P(\mathbf{v} | \mathbf{h}^1)P(\mathbf{h}^1 | \mathbf{h}^2) \dots P(\mathbf{h}^{l-2} | \mathbf{h}^{l-1})P(\mathbf{h}^{l-1}, \mathbf{h}^l)$$

- Sample \mathbf{h}^{l-1} using Gibbs sampling in the RBM
- Sample the lower layer \mathbf{h}^{i-1} from $P(\mathbf{h}^{i-1} | \mathbf{h}^i)$



Generating samples from DBN



Figure 9: Each row shows 10 samples from the generative model with a particular label clamped on. The top-level associative memory is initialized by an up-pass from a random binary image in which each pixel is on with a probability of 0.5. The first column shows the results of a down-pass from this initial high-level state. Subsequent columns are produced by 20 iterations of alternating Gibbs sampling in the associative memory.

Result for supervised fine-tuning on MNIST

- Very carefully trained backprop net with one or two hidden layers (Platt; Hinton) 1.6%
- SVM (Decoste & Schoelkopf, 2002) 1.4%
- Generative model of joint density of images and labels (+ generative fine-tuning) 1.25%
- Generative model of unlabelled digits followed by gentle backpropagation (Hinton & Salakhutdinov, Science 2006) 1.15%

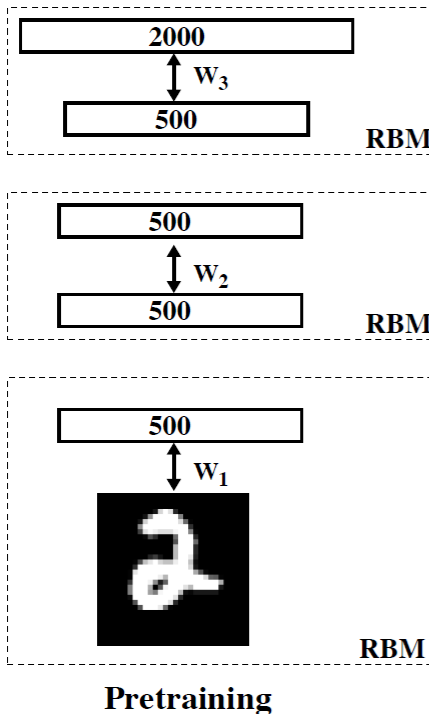
- More details on up-down algorithm:
 - Hinton, G. E., Osindero, S. and Teh, Y. (2006) “A fast learning algorithm for deep belief nets”, Neural Computation, 18, pp 1527-1554.
<http://www.cs.toronto.edu/~hinton/absps/ncfast.pdf>
- Handwritten digit demo:
 - <http://www.cs.toronto.edu/~hinton/digits.html>

Stacking of RBMs as Deep Neural Networks

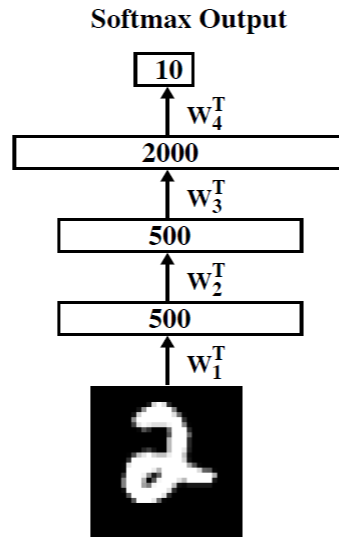
Using Stacks of RBMs as Neural Networks

- The feedforward (approximate) inference of the DBN looks the same as the sigmoid deep neural networks
- Idea: use the DBN as an initialization of the deep neural network, and then do fine-tuning with back-propagation

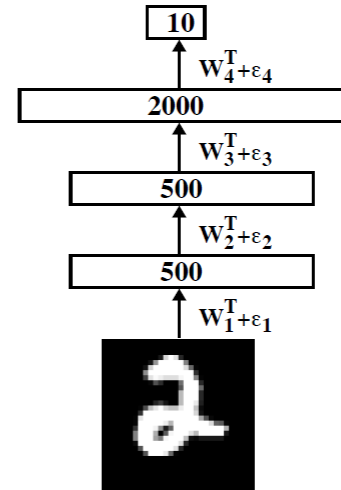
DBN for classification



Pretraining



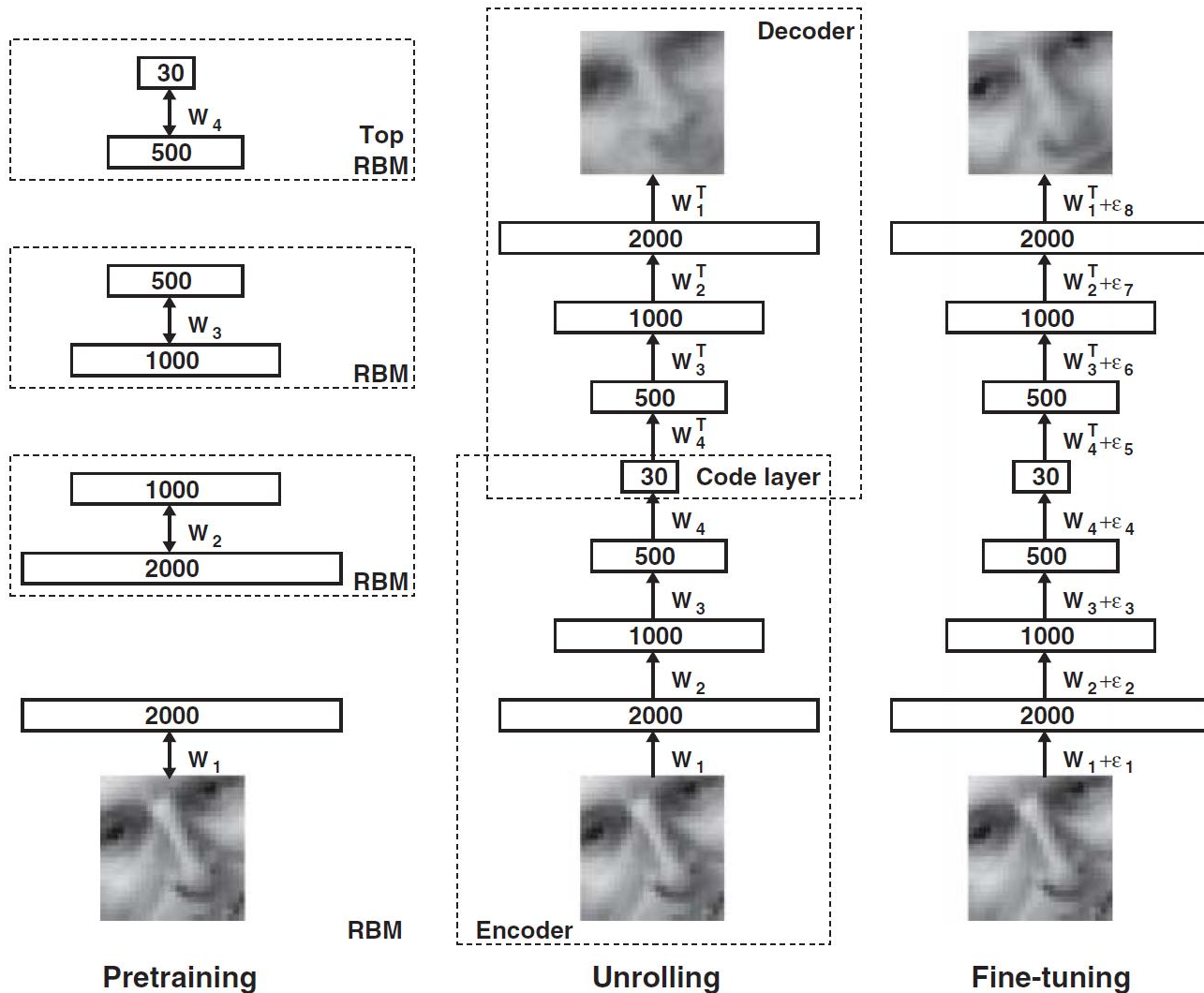
Unrolling



Fine-tuning

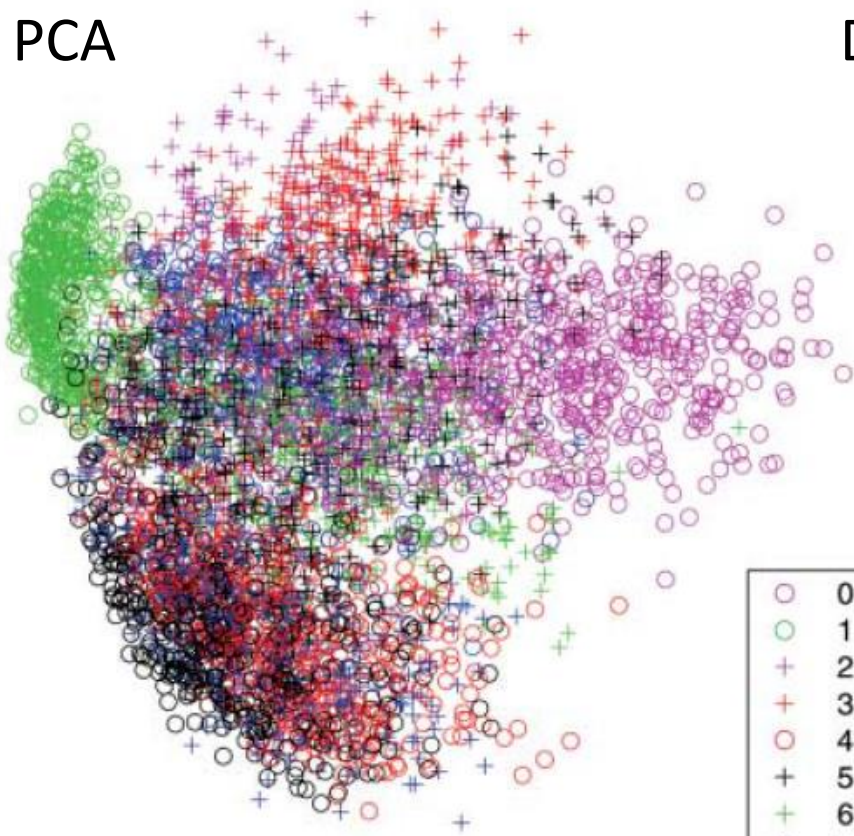
- After layer-by-layer **unsupervised pretraining**, discriminative fine-tuning by backpropagation achieves an error rate of 1.2% on MNIST. SVM's get 1.4% and randomly initialized backprop gets 1.6%.
- Clearly unsupervised learning helps generalization. It ensures that most of the information in the weights comes from modeling the input data.

Stacks of RBMs as deep autoencoders

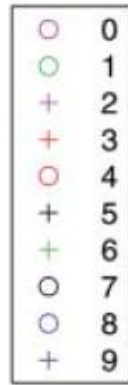
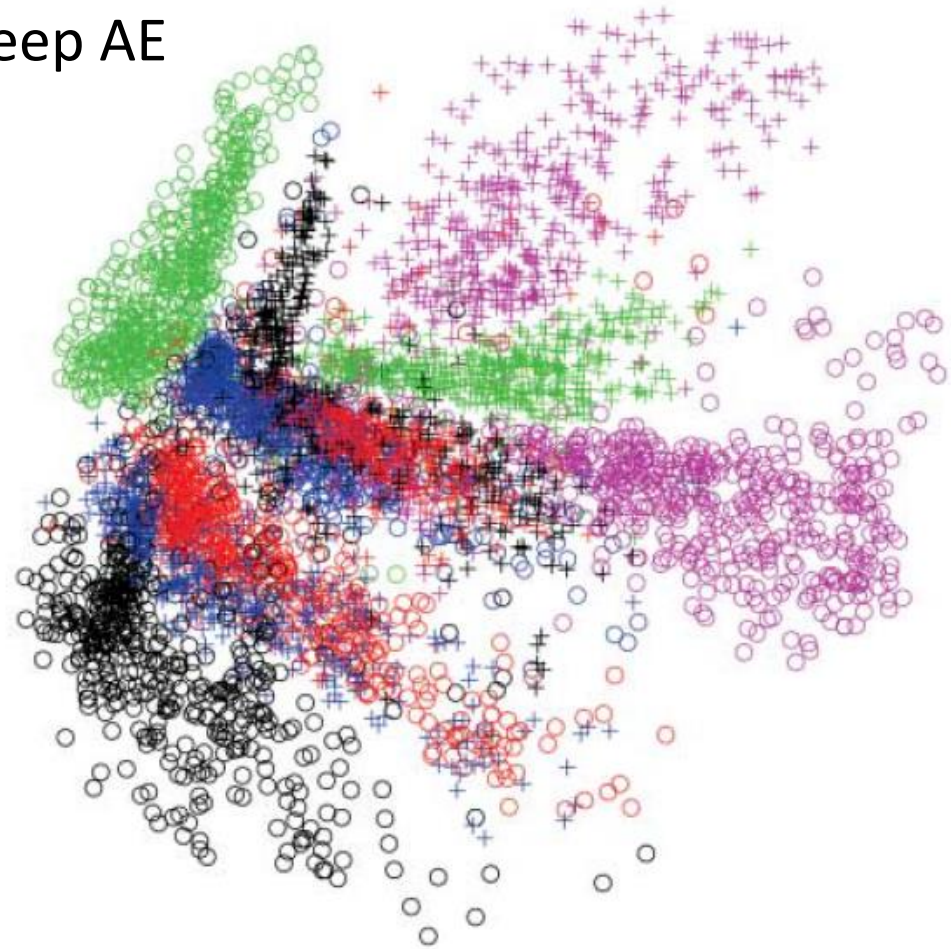


Stacks of RBMs as deep autoencoders

PCA

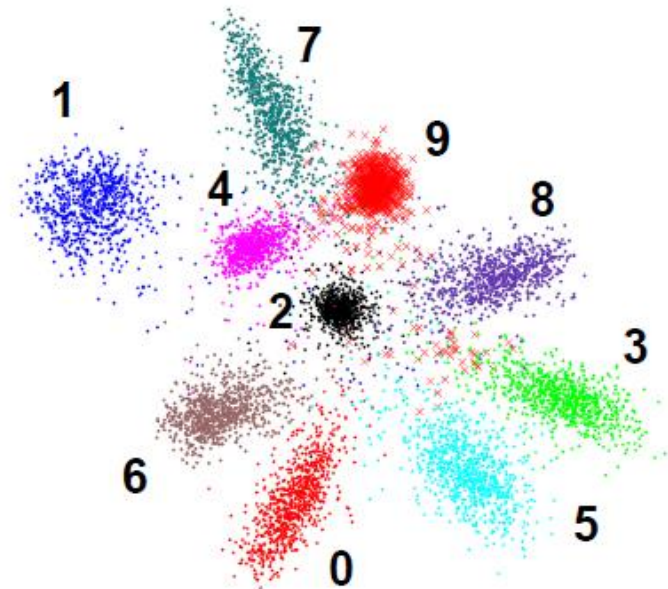
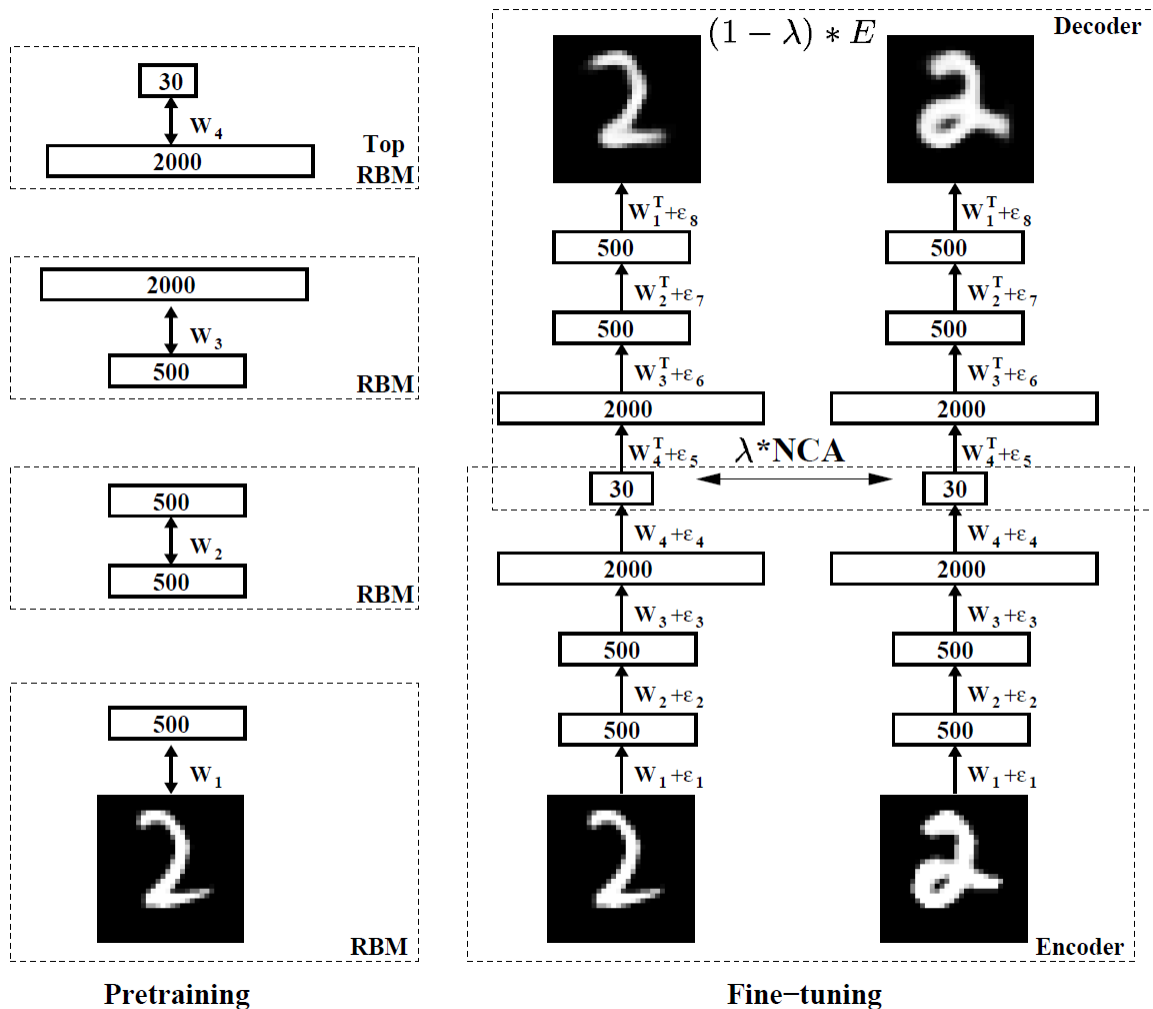


Deep AE



Learning similarity metric

- Stacks of RBMs as initialization for DNN

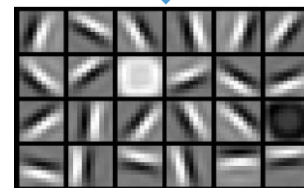
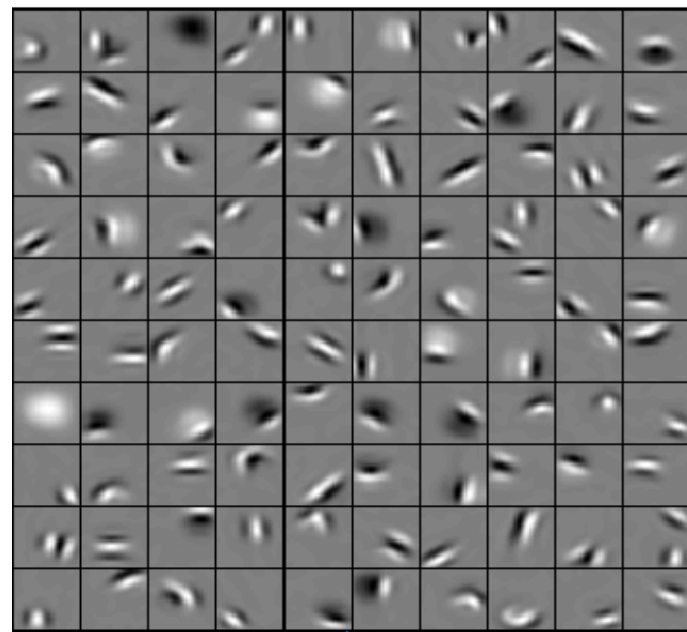
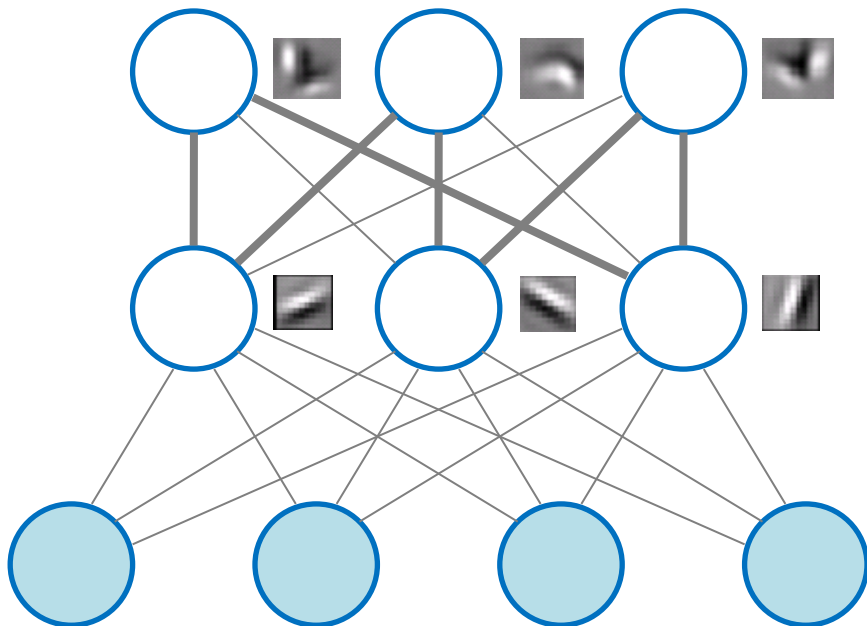


Salakhutdinov and Hinton, 2007

Learning feature hierarchy for images

Pixels \rightarrow edges \rightarrow contours \rightarrow ...

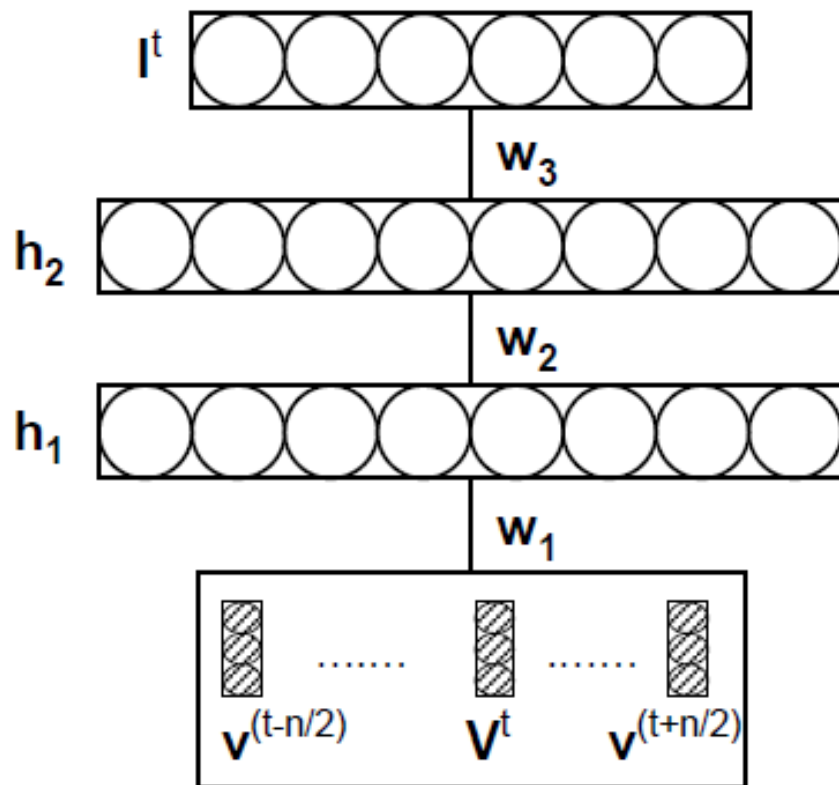
[Lee et al., 2007 & 2009;
Ranzato et al., 2007; etc.]



Speech recognition using DBNs

(Dahl et al., 2010)

- Pre-training RBMs followed by fine-tuning with back propagation



Learning feature hierarchy for speech

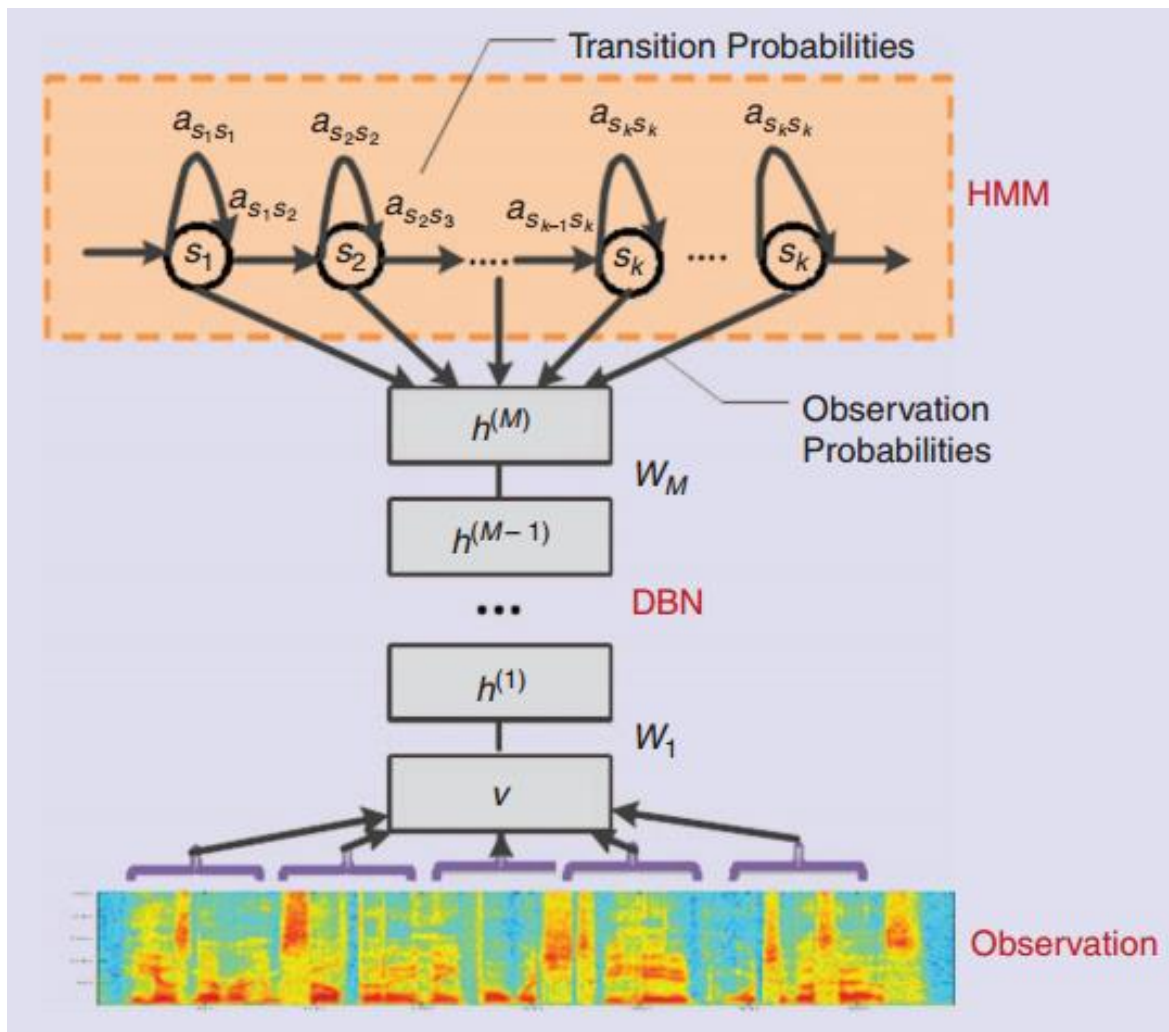


Image from: Dong Yu and Li Deng, 2012

Related work: Dahl et al., 2010; Hinton et al., 2012; and others

Other applications

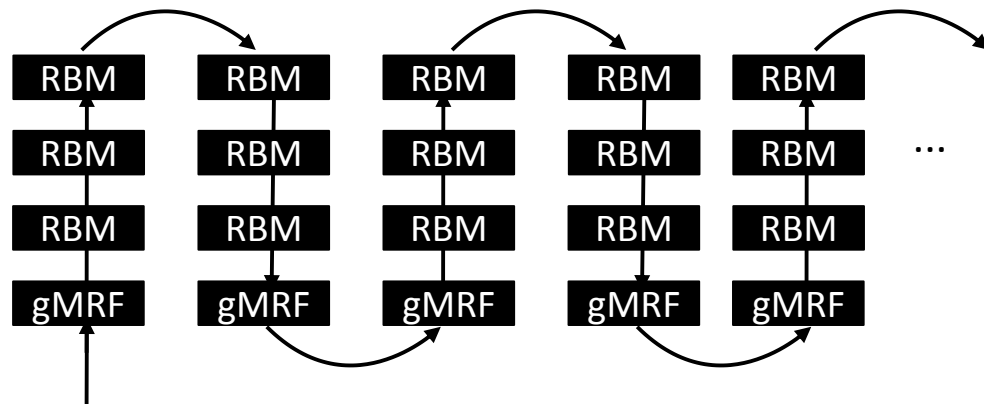
- Human pose/motion modeling with MOCAP data (Taylor et al., 2007; Taylor et al., 2009; ..)
- Multimodal deep learning: audio-visual speech recognition (Ngiam et al., 2011)
- Audio-visual emotion recognition (Kim et al., 2013)
- Facial Expression recognition (Ranzato et al., 2011)
- Face verification (Huang et al., 2012)
- Many others....

Remarks

- RBM is a generative model with distributed representation
 - Can handle well high dimensionality
- Stacks of RBMs can be used for initializing deep belief networks or deep neural networks
- For classification, pretraining (stacks of RBMs) helps when the amount of labeled data is not huge.
- DBN can be used for performing approximate probabilistic inference

DBN inference example: Image Inpainting for Facial Expression Recognition

- 7 synthetic occlusions
- use generative model to fill-in (conditional on the known pixels)



originals



occlusion: top half



Restored images



Another Example of a hybrid graphical model:
Learning Output Representation
for Structured Prediction

Learning Output Representations

- Most work in deep learning so far have focused on **learning input representations** (constructing input features)
 - Reasonable for **complex input data** (statistical dependencies, high dimensionality, etc.) and **labels with simple structures** (e.g., classification)
- However, relatively much less work has been done for **learning output representations**
 - Challenging when output space exhibits complex dependencies (e.g., structured output prediction: Joachims et al., Taskar et al.; Lafferty et al., ...)

Structured output prediction: Examples



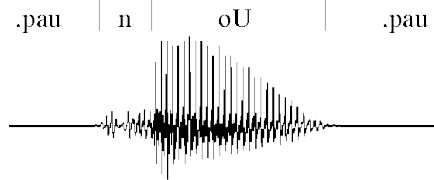
image segmentation
and labeling



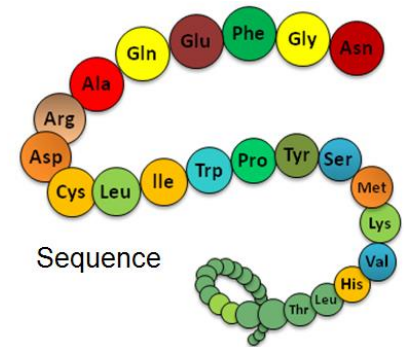
object detection



human pose estimation



phoneme recognition



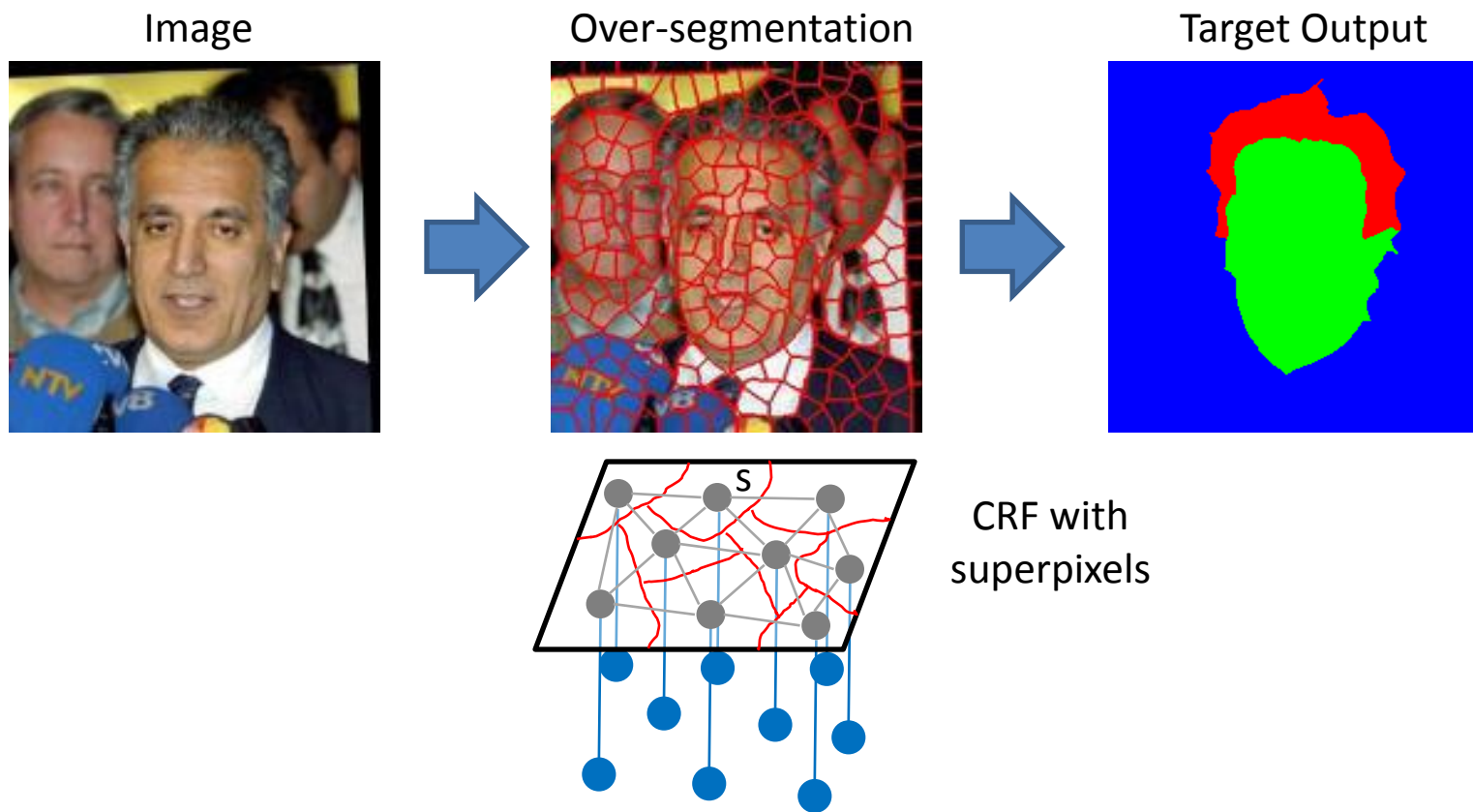
Sequence

protein structure
prediction

Combining Global and Local Consistencies for Structured Output Prediction

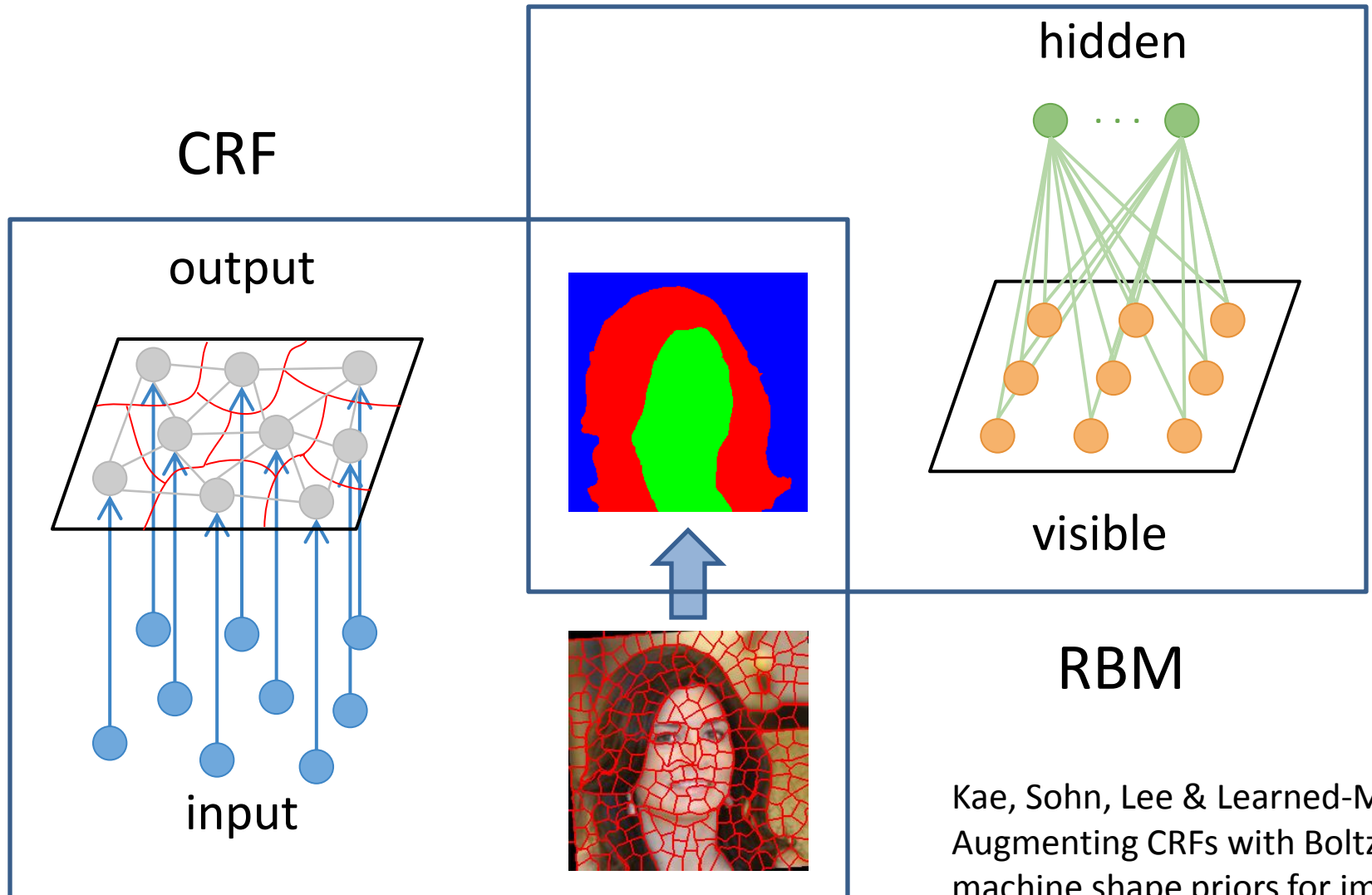
- Task: scene segmentation

(Kae et al., CVPR 2013)



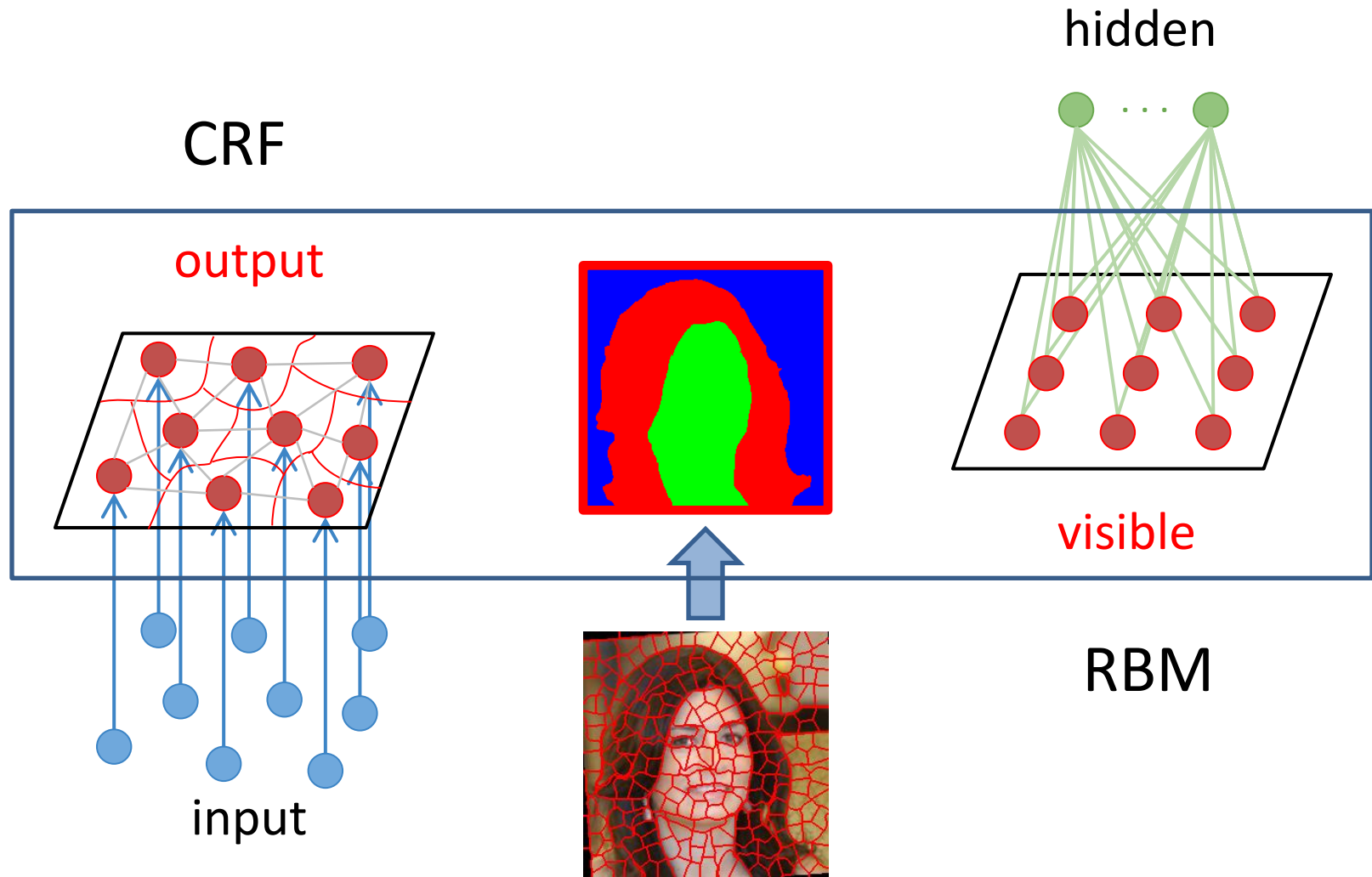
- Problem of standard CRFs: only enforces local consistency
- Approach: a hybrid graphical model (with a high-order potential) can enforce both local and global consistency

RCRF: generative model of face shape



Kae, Sohn, Lee & Learned-Miller
Augmenting CRFs with Boltzmann
machine shape priors for image
labeling. CVPR 2013.

Output of CRF = Visible of RBM



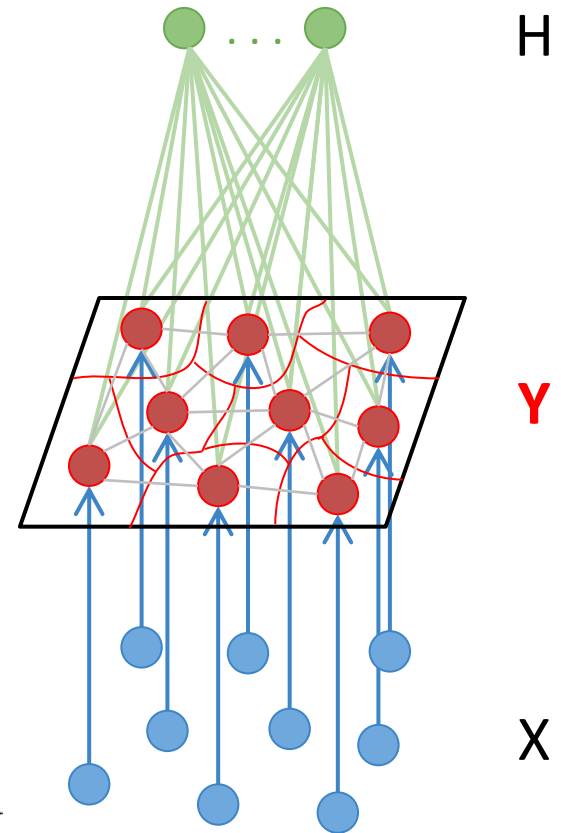
Kae, Sohn, Lee & Learned-Miller. Augmenting CRFs with Boltzmann machine shape priors for image labeling. CVPR 2013.

Formulation

- Energy function

$$E(\mathcal{Y}, \mathcal{X}, \mathcal{H}) =$$

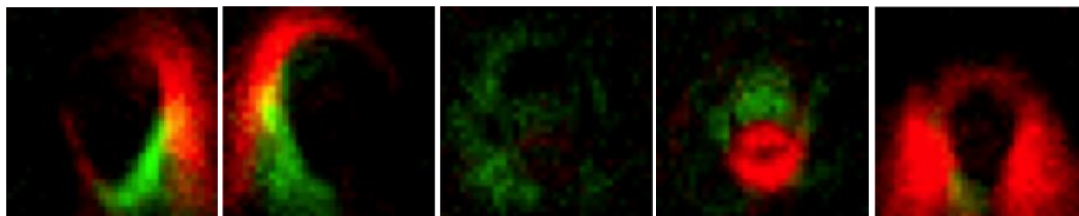
$$\underbrace{E^u(\mathcal{Y}, \mathcal{X}) + E^p(\mathcal{Y})}_{\text{CRF}} + \underbrace{E^{rbm}(\mathcal{Y}, \mathcal{H})}_{\text{RBM}}$$



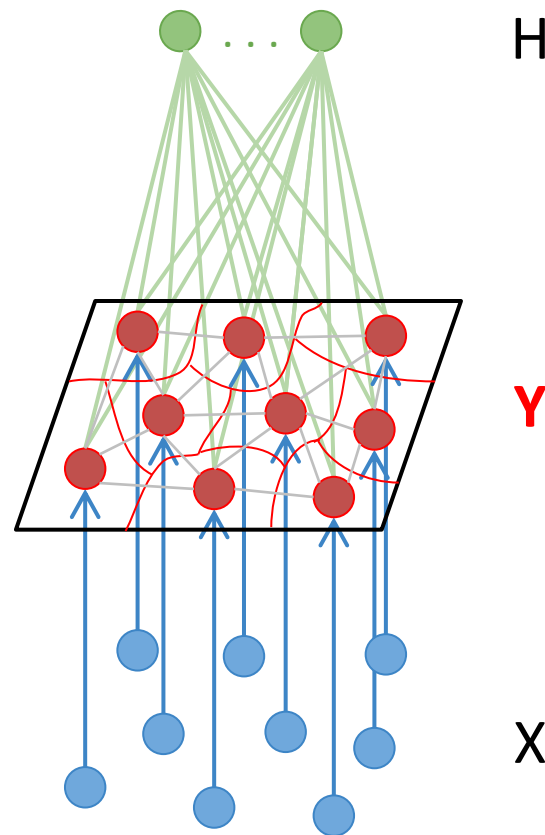
$$P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z} \sum_{\mathbf{h}} \exp \{-E(\mathbf{X}, \mathbf{Y}, \mathbf{h}; I)\}$$

$$E(\mathbf{X}, \mathbf{Y}, \mathbf{h}; I) = E_{\text{crf}}(\mathbf{X}, \mathbf{Y}) + E_{\text{rbm}}(\mathbf{Y}, \mathbf{h})$$

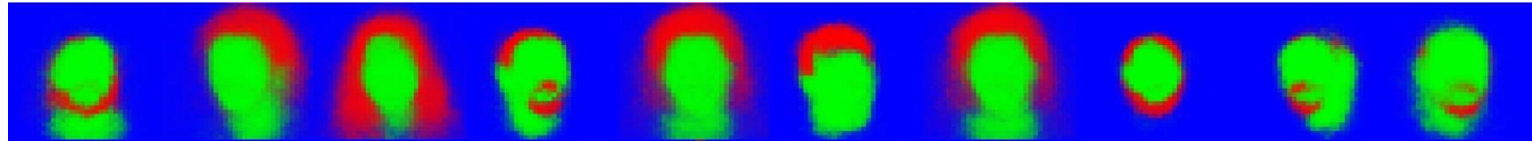
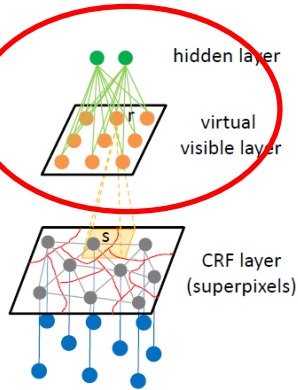
H learns shape patterns



RBM filter visualization (red: hair, green: skin, black: bg)



Generated Samples from RBM prior



- Top: generated samples (for outputs) from RBM prior
- Bottom: Closest training example to each generated sample above

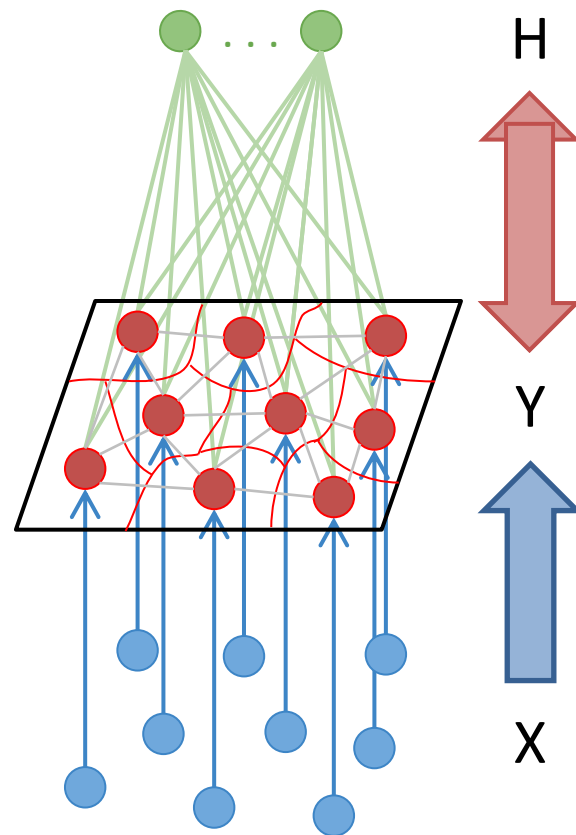
Prediction: mean-field approximation

- Infer Y given H, X
 - augmented unary potential:

$$\begin{aligned} E(\mathcal{Y}, \mathcal{X}, \mathcal{H}) &= E^u(\mathcal{Y}, \mathcal{X}) + E^{rbm}(\mathcal{Y}, \mathcal{H}) + E^p(\mathcal{Y}) \\ &= \bar{E}^u(\mathcal{Y}, \mathcal{X}, \mathcal{H}) + E^p(\mathcal{Y}) \end{aligned}$$

Note: Red arrows in the original image point from the E^u term in the second line to the \bar{E}^u term in the third line.

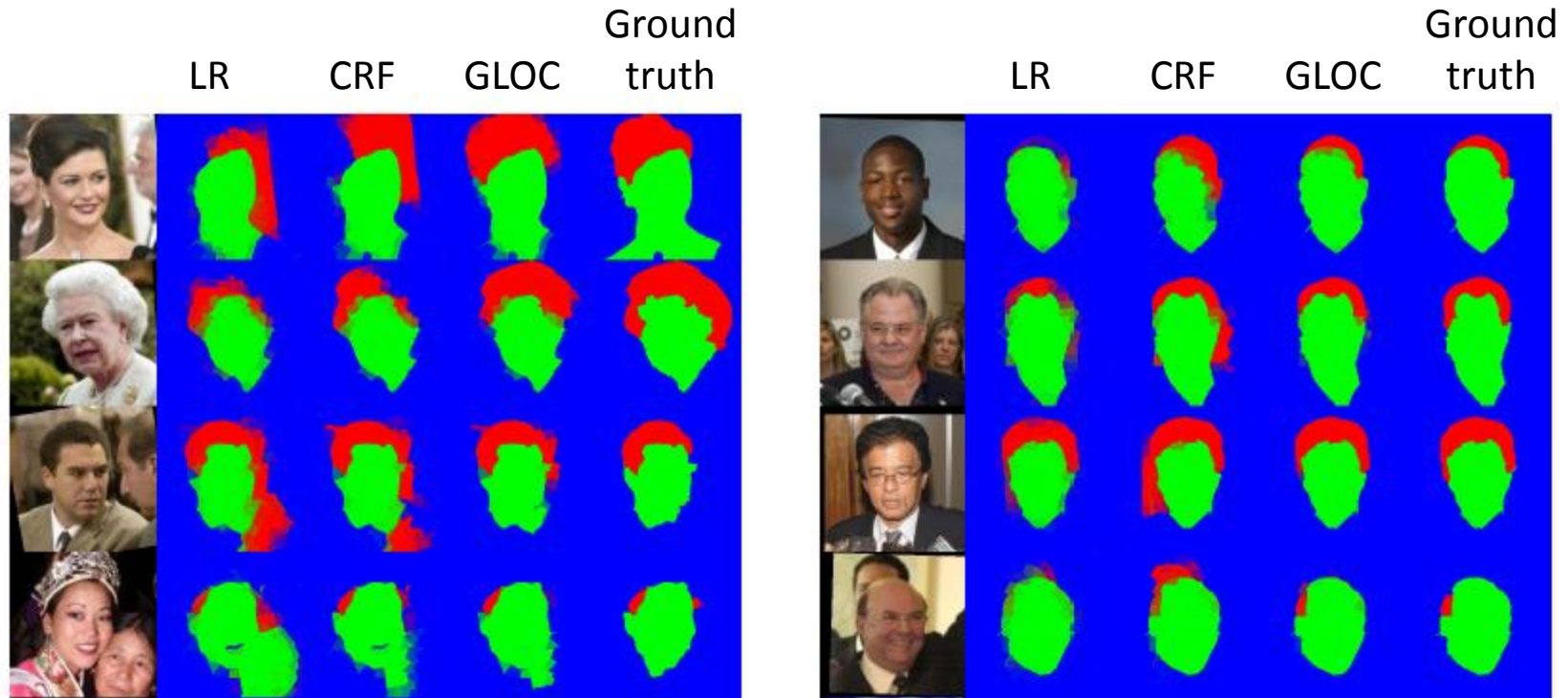
Any CRF inference algorithm (e.g., LBP, graph cut, ...) can be used.



Experimental results

- Visualization of segmentation

(Kae et al., CVPR 2013)



- LR: singleton potential
- CRF: singleton + pairwise potential
- GLOC: singleton + pairwise + RBM potential

Quantitative Evaluation on Labeling



~17% relative error reduction over spatial CRF