Reading Group on Deep Learning
Session 4
Unsupervised Neural Networks

Jakob Verbeek & Daan Wynen

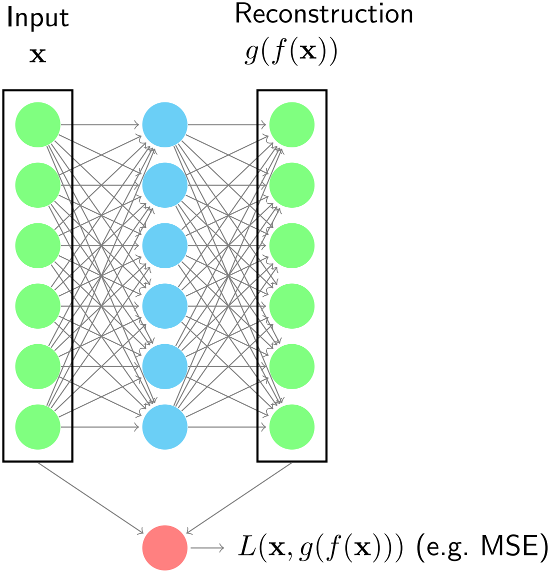
2016-09-22

Outline

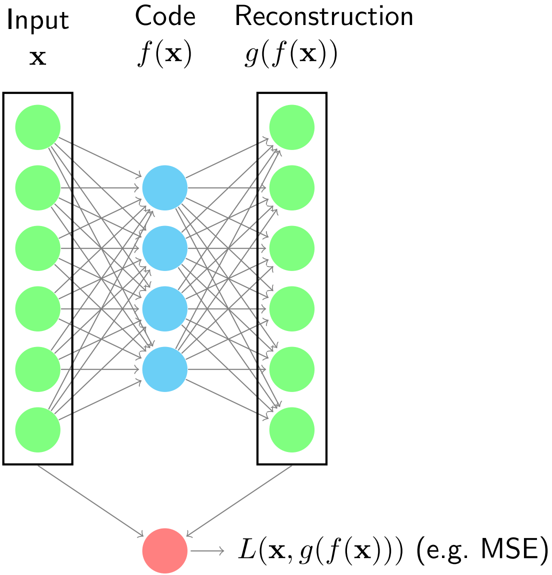
- Autoencoders
- (Restricted) Boltzmann Machines
- Deep Belief Networks

Autoencoders

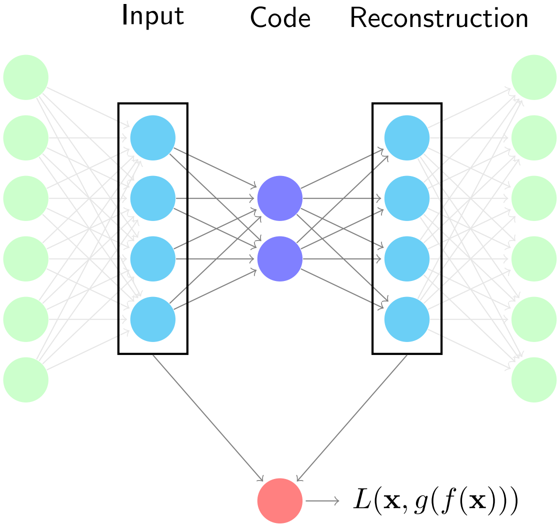
Autoencoder



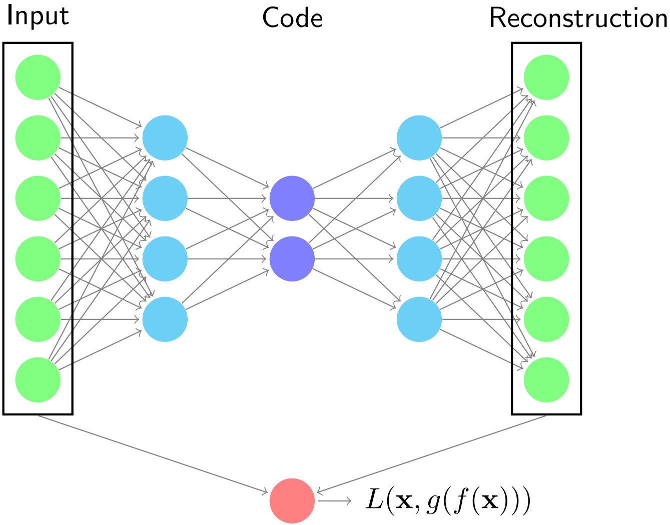
Autoencoder



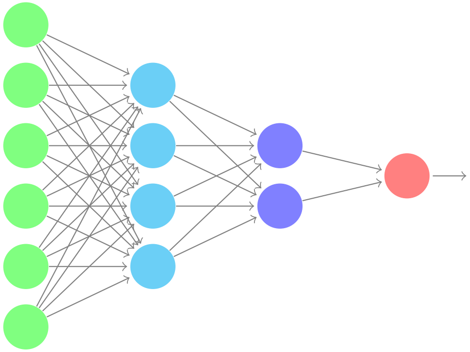
Autoencoder



Autoencoder

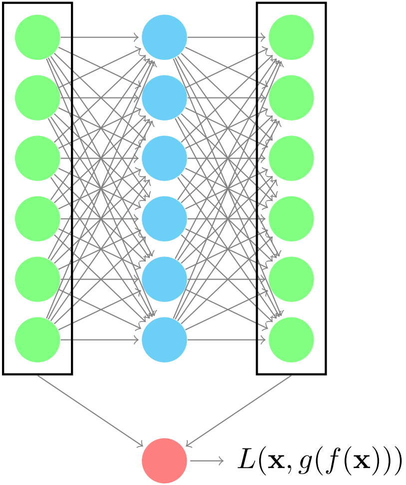


Autoencoder

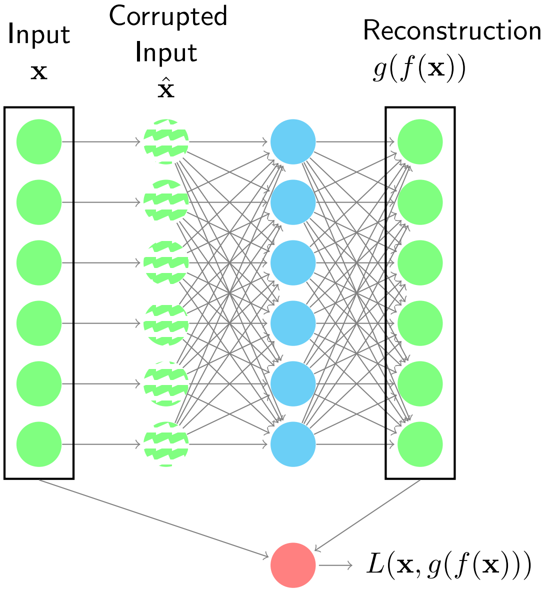


$L(f(\mathbf{x}), \mathbf{y})$

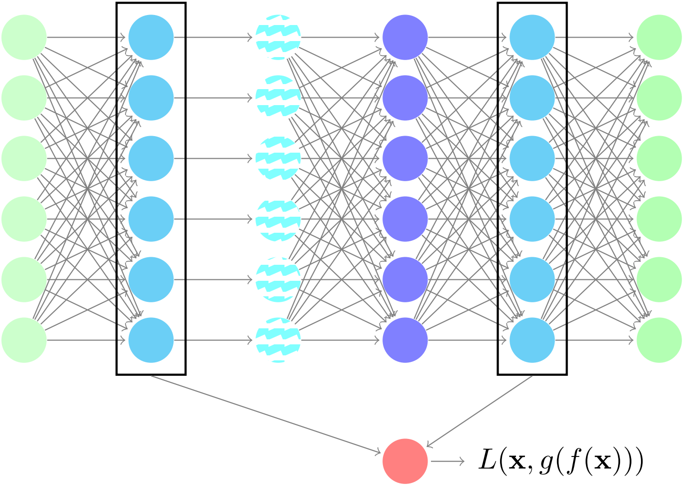
Denoising Autoencoder



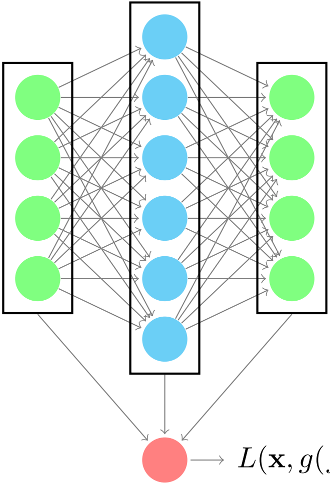
Denoising Autoencoder



Denoising Autoencoder



Sparse Autoencoder

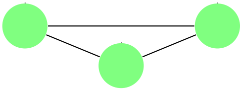


$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(f(\mathbf{x}))$$

(Restricted) Boltzmann Machines

Boltzmann Machine

Binary units $\mathbf{x}$

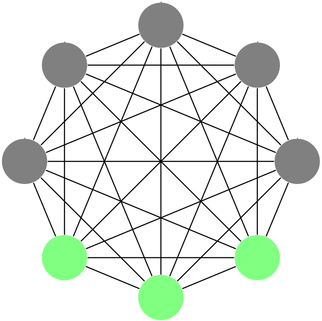


$$P(\mathbf{x}) = \frac{exp(-E(\mathbf{x})}{Z}$$

$$E(\mathbf{x}) = -\mathbf{x}^\top \mathbf{U}\mathbf{x} - \mathbf{b}^\top \mathbf{x}$$

Boltzmann Machine

Binary units $\mathbf{v}, \mathbf{h}$

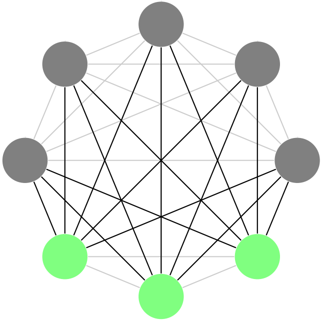


$$P(\mathbf{v}, \mathbf{h}) = \frac{exp(-E(\mathbf{v}, \mathbf{h}))}{Z}$$

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^{\intercal}\mathbf{R}\mathbf{v} - \mathbf{v}^{\intercal}\mathbf{W}\mathbf{h} - \mathbf{h}^{\intercal}\mathbf{S}\mathbf{h} - \mathbf{b}^{\intercal}\mathbf{v} - \mathbf{c}^{\intercal}\mathbf{h}$$

Restricted Boltzmann Machine

Binary units $\mathbf{v}, \mathbf{h}$



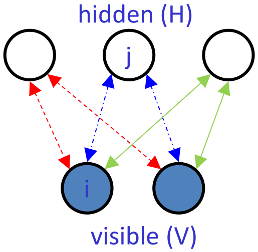
$$P(\mathbf{v}, \mathbf{h}) = \frac{exp(-E(\mathbf{v}, \mathbf{h})}{Z}$$

$$E(\mathbf{v}, \mathbf{h}) = \mathbf{v}^\top \mathbf{R} \mathbf{v} - -\mathbf{v}^\top \mathbf{W} \mathbf{h} - \mathbf{h}^\top \mathbf{S} \mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h}$$

Slides by Honglak Lee:
http://videolectures.net/deeplearning2015_lee_boltzmann_machines

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.



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

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

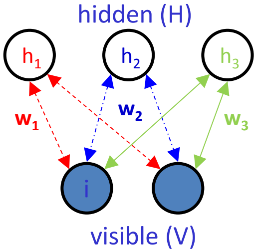
$$= -\mathbf{v}^T W \mathbf{h} - \mathbf{b}^T \mathbf{h} - \mathbf{c}^T \mathbf{v}$$

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

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

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.



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

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

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

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

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

Conditional Probabilities (RBM with binary-valued input data)

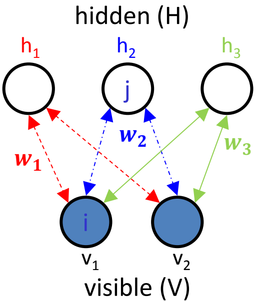
- Given $\mathbf{v}$, all the h_j are conditionally independent

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

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

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

  – P(**h**|**v**) can be used as "features"

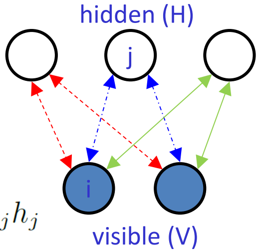


- Given $\mathbf{h}$, all the v_i are conditionally independent

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

RBMs with real-valued input data

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



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

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

RBMs with real-valued input data

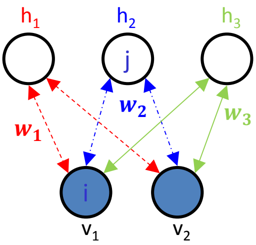
- Given $\mathbf{v}$, all the h_j are conditionally independent

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

$$= sigmoid(\frac{1}{\sigma}\sum_i W_{ij}v_i + b_j)$$

$$= sigmoid(\frac{1}{\sigma}\mathbf{w}_j^T\mathbf{v} + b_j)$$

  – P(**h**|**v**) can be used as “features”



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

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

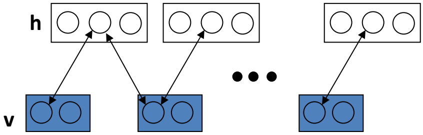
$$P(\mathbf{v}|\mathbf{h}) = \mathcal{N}(\sigma\mathbf{W}\mathbf{h} + \mathbf{c}, \sigma^2\mathbf{I})$$

Inference

- Conditional Distribution: P(v|h) or P(h|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(v,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,…) {
    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

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

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

Contrastive Divergence

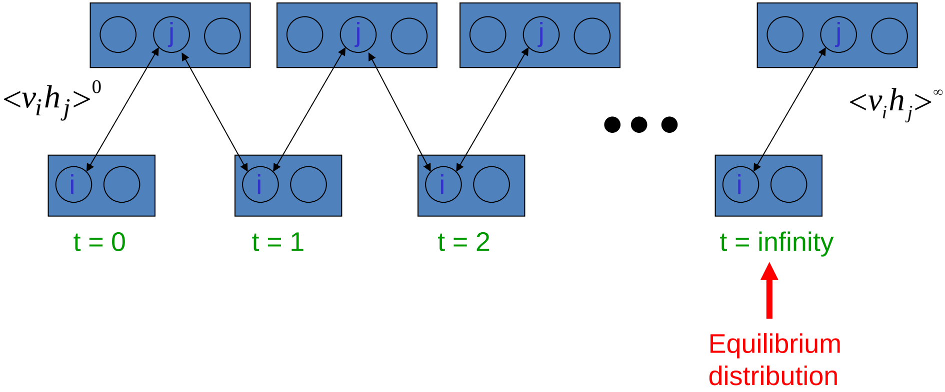
- Approximation of the log-likelihood gradient
  1. Replace the average over all possible inputs by samples

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

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

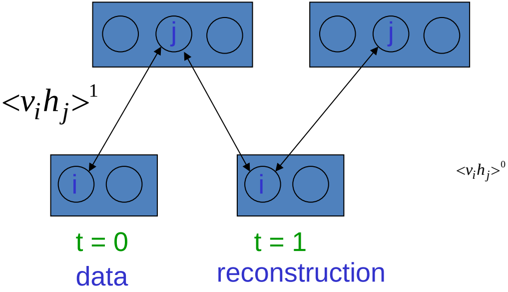
```
Initialize with v^0 = v
Sample h^0 from P(h|v^0)

For t = 1,…,k {
        Sample v^t from P(v^t|h^{t-1})
        Sample h^t from P(h|v^t)
}
```

Maximum likelihood learning for RBM



Slide Credit: Geoff Hinton

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.

Slide Credit: Geoff Hinton

Update rule: Putting together

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

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

  - 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

Deep Belief Networks

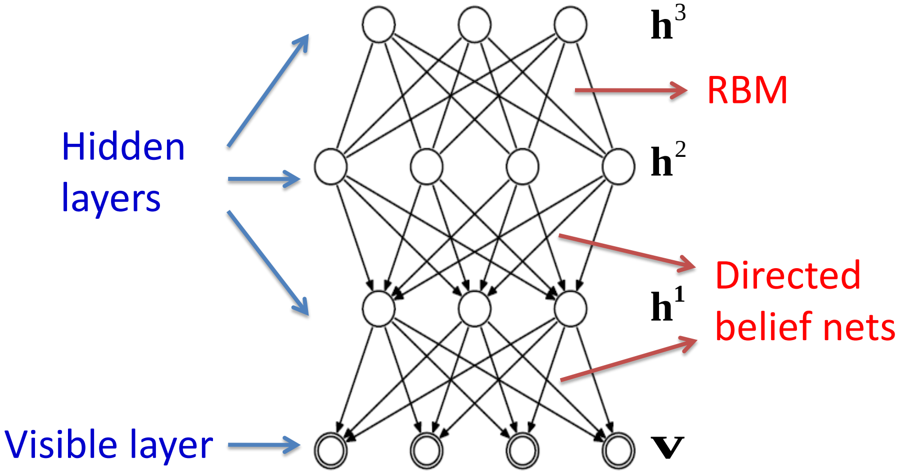
Deep Belief Networks (DBNs)

[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)

Related work: Deep Boltzmann Machine (Salakhutdinov and Hinton, 2009)

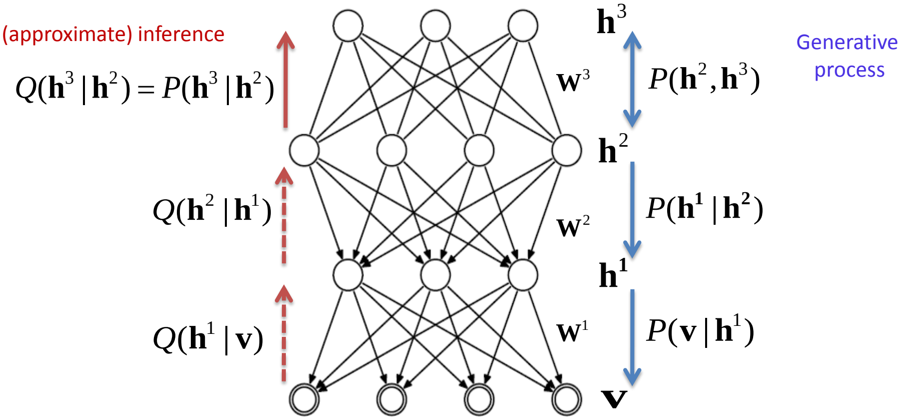
Deep Belief Networks (DBN)



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

DBN structure

Hinton et al., 2006



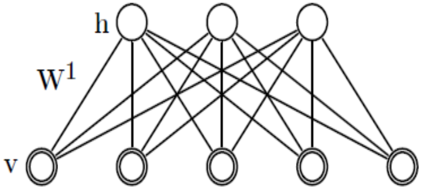
$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^l) = P(\mathbf{v} \mid \mathbf{h}^1) P(\mathbf{h}^1 \mid \mathbf{h}^2) \ldots P(\mathbf{h}^{l-2} \mid \mathbf{h}^{l-1}) P(\mathbf{h}^{l-1}, \mathbf{h}^l)$$

$$Q(\mathbf{h}^i \mid \mathbf{h}^{i-1}) = \prod_j sigm(\mathbf{b}_j^{i-1} + \mathrm{W}_j^i \mathbf{h}^{i-1}) \qquad P(\mathbf{h}^{i-1} \mid \mathbf{h}^i) = \prod_j sigm(\mathbf{b}_j^i + \mathrm{W}_{\cdot j}^{i}{}' \mathbf{h}^i)$$

DBN Greedy training

Hinton et al., 2006

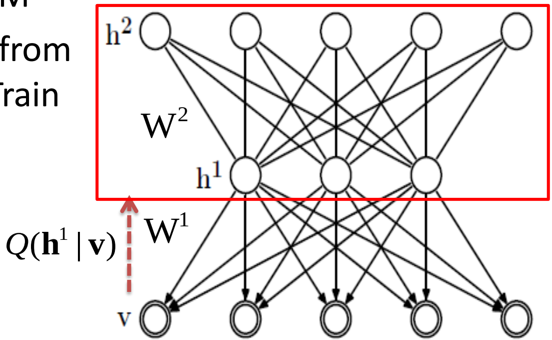
- First step:
  - Construct an RBM with an input layer **v** and a hidden layer **h**
  - Train the RBM



DBN Greedy training

Hinton et al., 2006

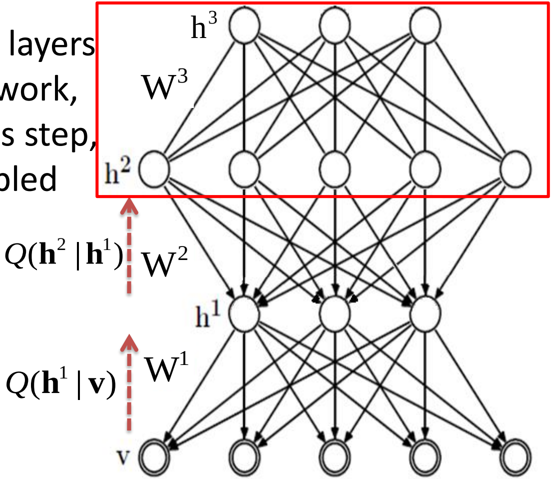
- Second step:
  - Stack another hidden layer on top of the RBM to form a new RBM
  - Fix W^1, sample $\mathbf{h}^1$ from $Q(\mathbf{h}^1 | \mathbf{v})$ as input. Train W^2 as RBM.



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…



DBN and supervised fine-tuning

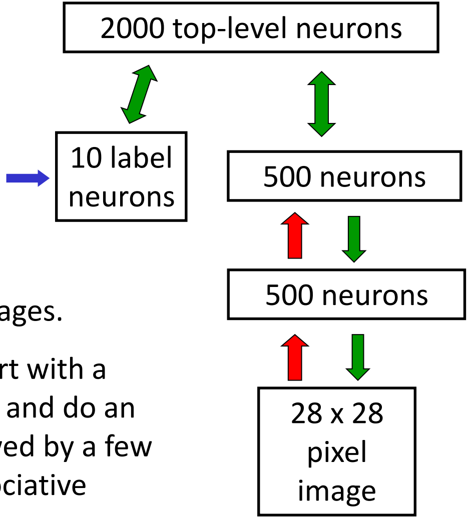
- Discriminative fine-tuning
  - Initializing with neural nets + backpropagation
  - Maximizes $\log P(Y \mid X)$ (X: data Y: label)
- Generative fine-tuning
  - Up-down algorithm
  - Maximizes $\log P(Y, X)$ (joint likelihood of data and labels)

- 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.

http://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.ppt

Slide Credit: Geoff Hinton

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.

http://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.ppt

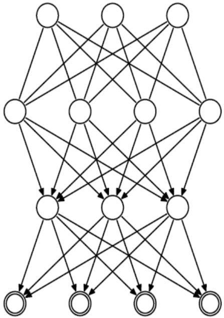
Slide Credit: Geoff Hinton

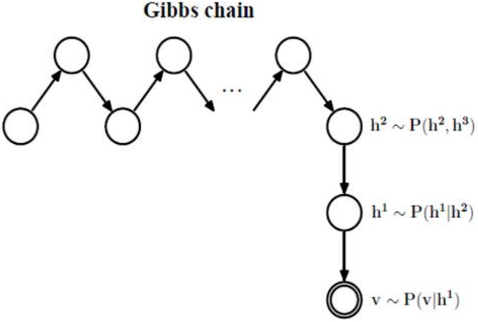
Generating sample from a DBN

- Want to sample from

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

  - 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.

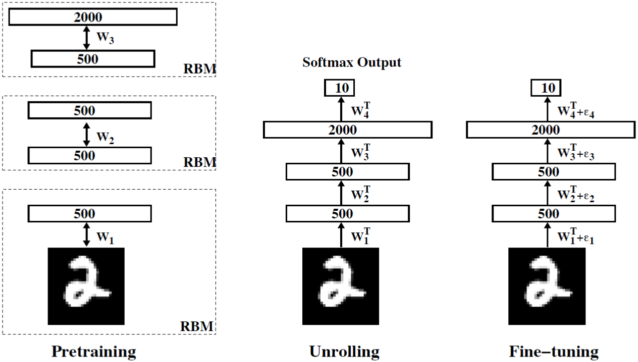
Hinton et al, A Fast Learning Algorithm for Deep Belief Nets, 2006

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



- 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.

Slide credit: Russ Salakhutdinov

Deep Boltzmann Machine

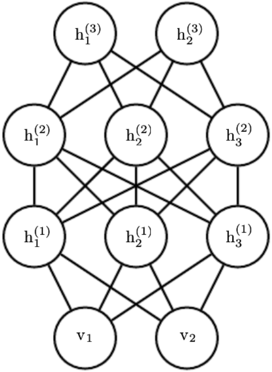


Figure: from `http://www.deeplearningbook.org`

- Specialization of Boltzmann Machine

Deep Boltzmann Machine

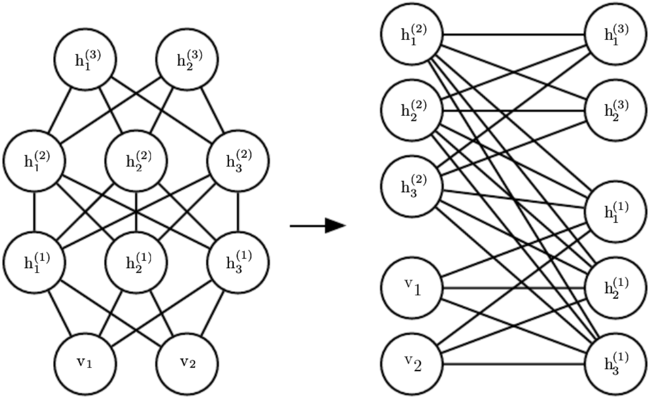


Figure: from `http://www.deeplearningbook.org`

- Specialization of Boltzmann Machine
- Also specialization of *Restricted* Boltzmann Machine!

DBM - Properties

- Allows for up-down interactions between layers
- Compared to DBNets, $Q(\mathbf{h}|\mathbf{v})$ can be tighter to $P(\mathbf{h}|\mathbf{v})$
- Also makes inference and training harder
  - MCMC across all layers...