Seminar: Deep Learning

Deep Belief Networks

Dimitri Thomas Ghouse

LMU München

February 2017

Content

Motivation

Unsupersived Learning:

- only use inputs (e.g. pixel) for learning
- learning without labeled data
- most data exists in unlabeled form
- extract useful features of data

Generative (probabilistic) Model

- generative approach: try to model data distribution $p(x)$
- Use learnt weights of unsupervised model as initialization for a discriminative neural net

Outline

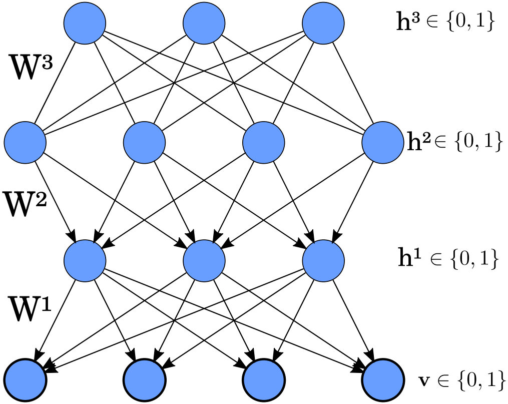


Figure: Deep Belief Network: Structure

Outline

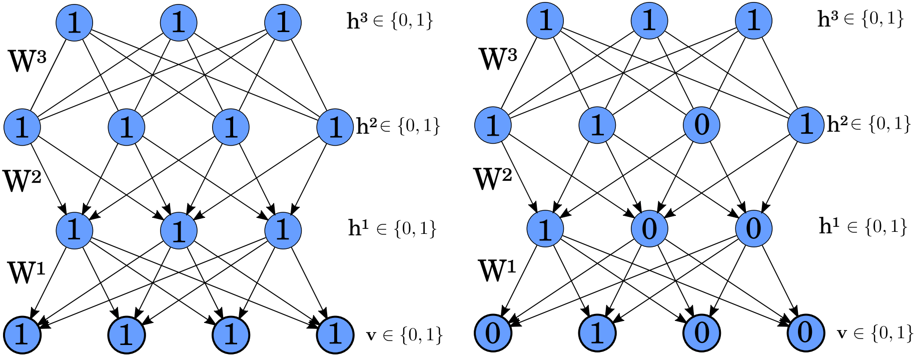


Figure: Deep Belief Network: Generative process

Outline

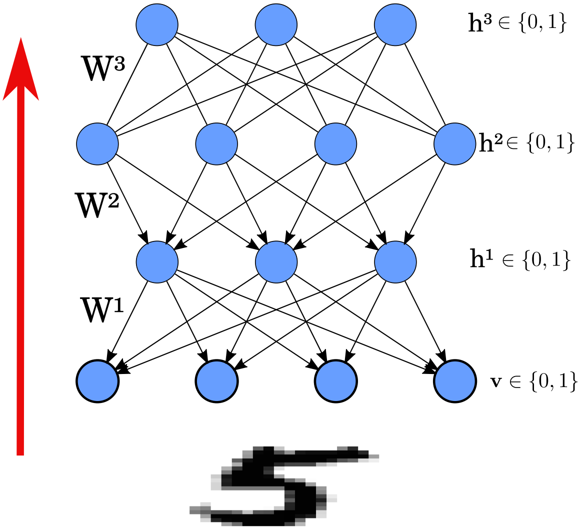


Figure: Deep Belief Network: Learning

Outline

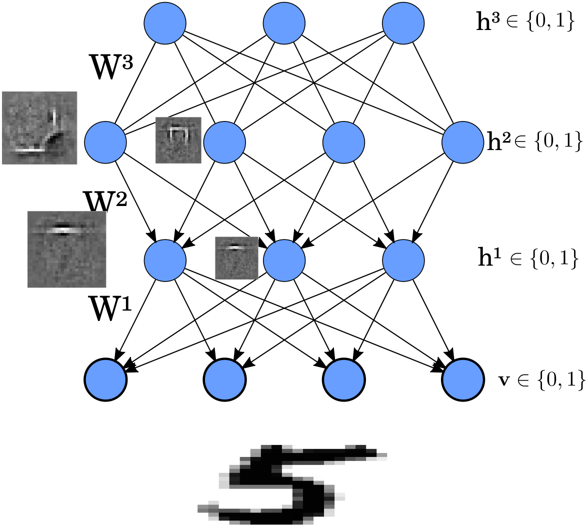


Figure: Deep Belief Network: Hierarchical features

Outline

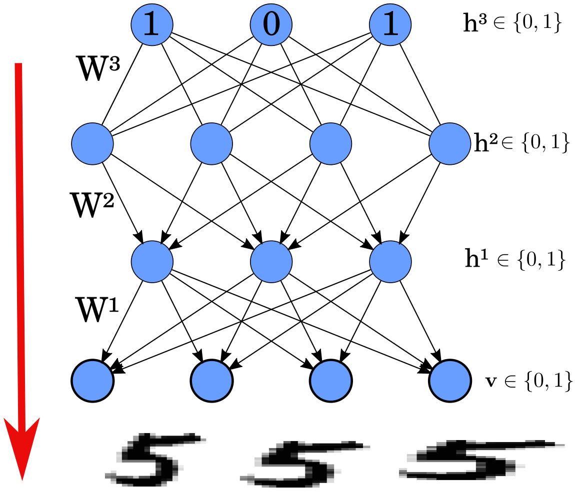


Figure: Deep Belief Network: Generate Pixels

Outline

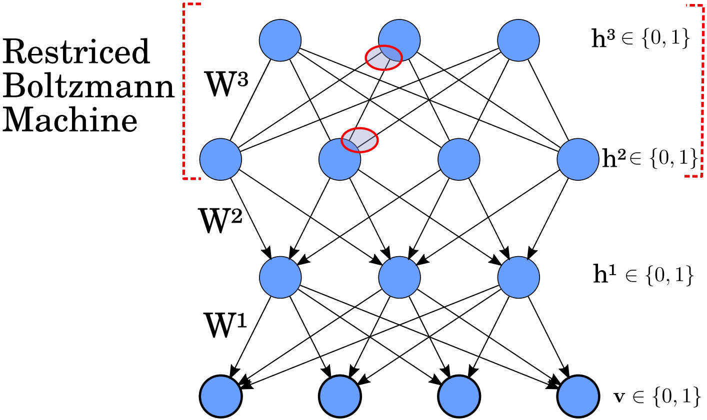


Figure: Deep Belief Network: Structure

Outline

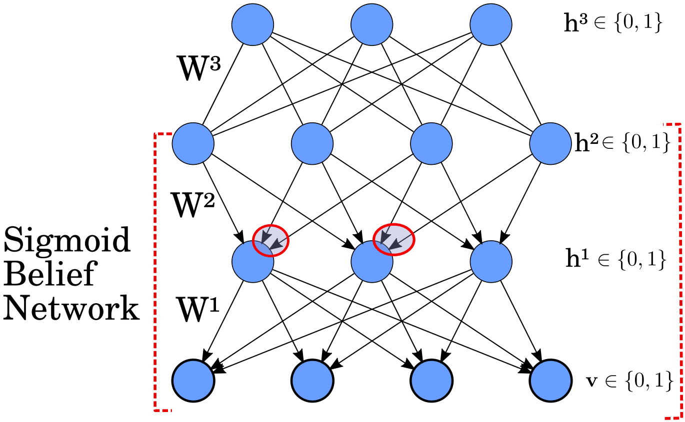


Figure: Deep Belief Network: Structure

Sigmoid Belief Network

- Causal model (Directed Acyclical Graph)
- Binary stochastic neuron
  - $p\left(v_i = 1|\mathbf{h^1}\right) = \dfrac{1}{1+exp\left(-b_i - \sum_j W_{ij}^1 h_j^1\right)}$
  - $v_i = \mathbf{1}\left\{p(v_i = 1|\mathbf{h^1}) > U\left[0,1\right]\right\}$
  - $p(\mathbf{v}|\mathbf{h}^1) = \prod_i p\left(v_i|\mathbf{h^1}\right)$
- Easy to generate a sample (top-down)

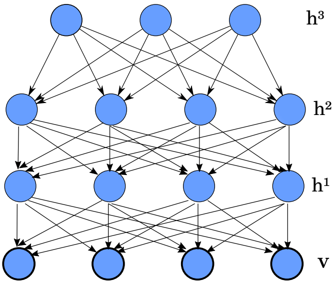


Figure: Sigmoid Belief Net

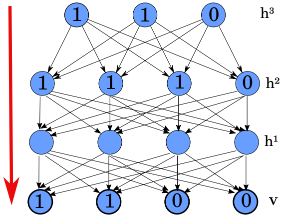
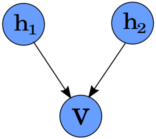


Figure: Sigmoid Belief Net

SBN - Explaining Away Problem

Joint distribution:

$$p(h_1, h_2, v) = p(h_1)p(h_2)p(v|h_1, h_2)$$



Independence properties:

$$\begin{aligned} p(h_1, h_2) &= \sum_v p(h_1)p(h_2)p(v|h_1, h_2) \\ &= p(h_1)p(h_2) \end{aligned}$$

Conditionally dependent: (explaining away)

$$\begin{aligned} p(h_1, h_2|v) &= \frac{p(h_1, h_2, v)}{p(v)} \\ &= \frac{p(h_1)p(h_2)p(v|h_1, h_2)}{p(v)} \end{aligned}$$

Sigmoid Belief Network

Inference Problem:

- Posterior distribution for hidden given data hard to compute
- $p\left(\mathbf{h^1}|\mathbf{v}\right) \rightarrow$ explaining away
- $p(\mathbf{h}^1|\mathbf{v})$ is not factorial

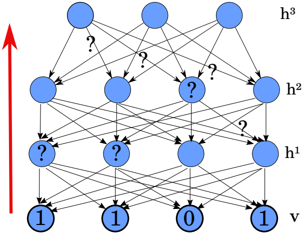


Figure: Sigmoid Belief Net: Learning

Learning Problem:

- Adjust w_{ij} so SBN generates more likely training data $\mathbf{v}$
- Maximizing $log\ p(v)$ difficult

RBM - Example

$$E(\mathbf{v},\mathbf{h};\theta) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j \qquad p\left(\mathbf{v},\mathbf{h};\theta\right) = \frac{1}{Z(\theta)} e^{-E(v,h;\theta)} \qquad Z(\theta) = \sum_{\mathbf{v}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)}$$

S	h_1	h_2	v_1	v_2	$-E(v,h)$	$e^{-E(v,h)}$	$p(v,h)$	$p(v)$
1			0	0				
2			0	0				
3			0	0				
4			0	0				
5			1	0				
6			1	0				
7			1	0				
8			1	0				
9			0	1				
$\vdots$			$\vdots$	$\vdots$				
15			1	1				
16			1	1				

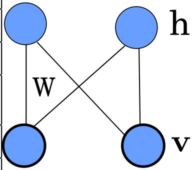


Tabelle: RBM, $w_{ij} = 1$

RBM - Example

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j \qquad p(\mathbf{v}, \mathbf{h}; \theta) = \frac{1}{Z(\theta)} e^{-E(v,h;\theta)} \qquad Z(\theta) = \sum_{\mathbf{v}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)}$$

S	h_1	h_2	v_1	v_2	$-E(v,h)$	$e^{-E(v,h)}$	$p(v,h)$	$p(v)$
1	0	0	0	0				
2	1	0	0	0				
3	0	1	0	0				
4	1	1	0	0				
5	0	0	1	0				
6	1	0	1	0				
7	0	1	1	0				
8	1	1	1	0				
9	0	0	0	1				
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$				
15	0	1	1	1				
16	1	1	1	1				

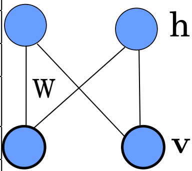


Tabelle: RBM, $w_{ij} = 1$

RBM - Example

$$E(\mathbf{v},\mathbf{h};\theta) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j \qquad p(\mathbf{v},\mathbf{h};\theta) = \frac{1}{Z(\theta)} e^{-E(v,h;\theta)} \qquad Z(\theta) = \sum_{\mathbf{v}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)}$$

S	h_1	h_2	v_1	v_2	$-E(v,h)$	$e^{-E(v,h)}$	$p(v,h)$	$p(v) = \sum_h p(v,h)$
1	0	0	0	0	0	1,000		
2	1	0	0	0	0	1,000		
3	0	1	0	0	0	1,000		
4	1	1	0	0	0	1,000		
5	0	0	1	0	0	1,000		
6	1	0	1	0	1	2,718		
7	0	1	1	0	1	2,718		
8	1	1	1	0	2	7,389		
9	0	0	0	1	0	1,000		
⋮	⋮	⋮	⋮	⋮	⋮	⋮		
15	0	1	1	1	2	7,389		
16	1	1	1	1	4	54,598		
						102,028		

Tabelle: RBM, $w_{ij} = 1$

RBM - Example

$$E(\mathbf{v},\mathbf{h};\theta) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j \qquad p(\mathbf{v},\mathbf{h};\theta) = \frac{1}{Z(\theta)} e^{-E(v,h;\theta)} \qquad Z(\theta) = \sum_{\mathbf{v}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)}$$

S	h_1	h_2	v_1	v_2	$-E(v,h)$	$e^{-E(v,h)}$	$p(v,h)$	$p(v) = \sum_h p(v,h)$
1	0	0	0	0	0	1,000	0,010	**0,04**
2	1	0	0	0	0	1,000	0,010	
3	0	1	0	0	0	1,000	0,010	
4	1	1	0	0	0	1,000	0,010	
5	0	0	1	0	0	1,000	0,010	**0,14**
6	1	0	1	0	1	2,718	0,027	
7	0	1	1	0	1	2,718	0,027	
8	1	1	1	0	2	7,389	0,072	
9	0	0	0	1	0	1,000	0,010	**0,14**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	0	1	1	1	2	7,389	0,072	**0,68**
16	1	1	1	1	4	54,598	0,535	
						102,028	$\sum 1$	

Tabelle: RBM, $w_{ij} = 1$

Restricted Boltzmann Machine

Goal: Model a probability distribution over a set of random variables

Energy-based models

- Captures dependencies between random variables through an energy-function
- $\mathbf{v} \in \{1,0\}^m \qquad \mathbf{h} \in \{1,0\}^n$
- RBM has in total $(\mathbf{v},\mathbf{h}) \in \{1,0\}^{m+n}$ possible states
- Every state has a associated scalar value
- State with a lower energy is more likely

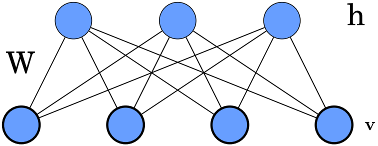


Figure: RBM: 3 hidden and 4 visible layers

Energy function :

$$E(\mathbf{v},\mathbf{h};W,b,c) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

RBM - Probability Distribution

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \qquad \theta = (W, b, c)$$

Get a probability distribution for every possible configuration of $(\mathbf{v}, \mathbf{h})$

$$p\left(v, h; \theta\right) = \frac{1}{Z(\theta)} e^{-E(v,h;\theta)} \qquad Z(\theta) = \sum_{v}\sum_{h} e^{-E(v,h;\theta)}$$

RBM - Properties

Conditional Distribution

$p(\mathbf{h}|\mathbf{v}) = \prod_{i=1}^{n} p(h_i|\mathbf{v}) \quad p(\mathbf{v}|\mathbf{h}) = \prod_{j=1}^{m} p(v_j|\mathbf{h})$

$\Rightarrow$ Sampling all h_i given v at once possible

RBMs as a (stochastic) neural network

$$p(h_i = 1|\mathbf{v}) = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$

$$h_i = \mathbf{1}\{p(h_i = 1|\mathbf{v}) > U[0,1]\}$$

$$p(v_j = 1|\mathbf{h}) = \sigma\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$

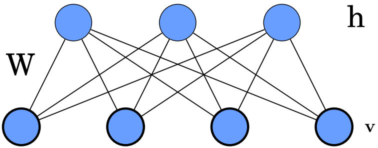


Figure: RBM: 3 hidden and 4 visible layers

RBM - Log-Likelihood

Goal: Adjust weights & bias terms so that $p(v)$ equals more to the data distribution

Likelihood-Approach:

$$log \mathcal{L}(\theta|v) = log\, p(v;\theta) = log \sum_h p(v,h;\theta) \quad = \quad log\, \frac{1}{Z(\theta)} \sum_h e^{-E(v,h;\theta)}$$

$$= log \sum_h e^{-E(v,h;\theta)} \qquad - \qquad log \sum_v \sum_h e^{-E(v,h;\theta)}$$

$$\frac{\partial log \mathcal{L}(\theta|v)}{\partial w_{ij}} \quad = \quad \frac{\partial}{\partial w_{ij}} \left(log \sum_h e^{-E(v,h)} \right) \quad - \quad \frac{\partial}{\partial w_{ij}} \left(log \sum_v \sum_h e^{-E(v,h)} \right)$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$-\sum_h p(h|v) \quad \frac{\partial E(v,h)}{\partial w_{ij}} \quad + \quad \sum_v \sum_h p(v,h) \quad \frac{\partial E(v,h)}{\partial w_{ij}}$$

RBM - Gradient of Log-Likelihood

$$\frac{\partial E(\mathbf{v},\mathbf{h};\theta)}{\partial w_{ij}} = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

$$= -h_i v_j$$

$$\frac{\partial log \mathcal{L}(\theta|\mathbf{v})}{\partial w_{ij}} = -\sum_{h_i \in \{0,1\}} p(h|\mathbf{v}) \frac{\partial E(v,h;\theta)}{\partial w_{ij}} + \sum_{v}\sum_{h} p(v,h) \frac{\partial E(v,h;\theta)}{\partial w_{ij}}$$

$$= p\left(h_i = 1|\mathbf{v}\right) v_j - \sum_{v} \underset{\substack{\uparrow \\ Problem}}{p\left(v\right)} p\left(h_i = 1|\mathbf{v}\right) v_j$$

$$p(h_i = 1|\mathbf{v}) = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$

RBM - Contrastive Divergence

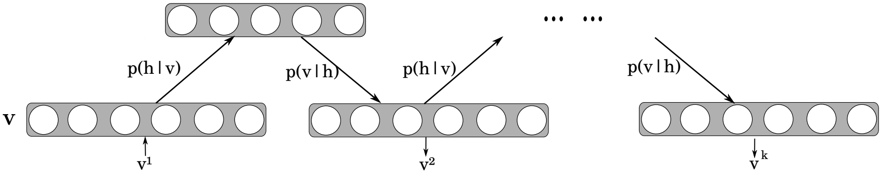


Figure: Contrastive Divergence (CD)

Contrastive divergence:

1: For a given training example $\mathbf{v} \equiv \mathbf{v^1}$

1. Start at $\mathbf{v^1}$. Generate a $\mathbf{v^k}$ sample using k steps of Block-Gibbs sampling
2. Update parameters: $w_{ij} \leftarrow w_{ij} + p\left(h_i = 1|\mathbf{v^1}\right) v_j^1 - p\left(h_i = 1|\mathbf{v^k}\right) v_j^k$

2: Repeat until stopping criteria

RBM: Negative - Positive - Phase

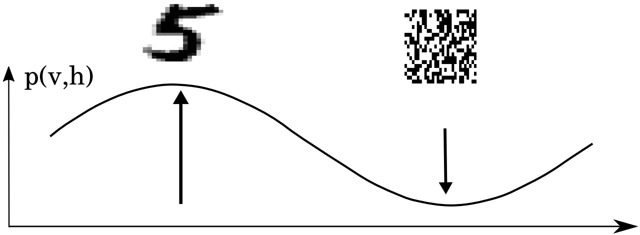


Figure: Positive and Negative Phase of Learning

$$
\begin{aligned}
\frac{\partial \log \mathcal{L}(\theta|v)}{\partial w_{ij}} &= -\sum_h p(h|v) \frac{\partial E(v,h;\theta)}{\partial w_{ij}} + \sum_v \sum_h p(v,h) \frac{\partial E(v,h;\theta)}{\partial w_{ij}} \\
&= \mathbb{E}_{p(h|v)} [h_i v_j] - \mathbb{E}_{p(v,h)} [h_i v_j] \\
&= \mathbb{E}_{P_{Data}} [h_i v_j] - \mathbb{E}_{P_{Model}} [h_i v_j]
\end{aligned}
$$

Deep Belief Network

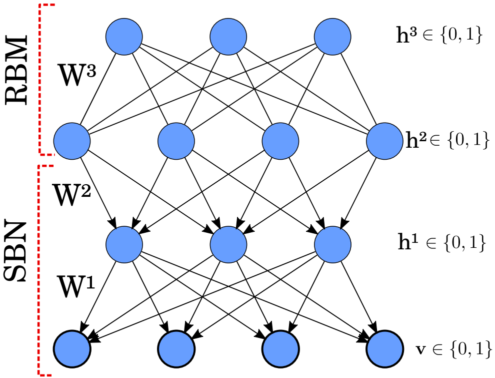
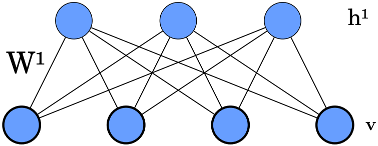


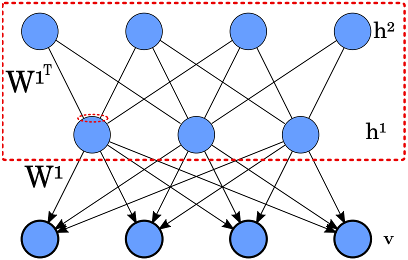
Figure: Deep Belief Network

$$p\left(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3; \theta\right) = \underbrace{p\left(\mathbf{v}|\mathbf{h}^1; W^1\right) p\left(\mathbf{h}^1|\mathbf{h}^2; W^2\right)}_{SBN} \underbrace{p\left(\mathbf{h}^2, \mathbf{h}^3; W^3\right)}_{RBM}$$

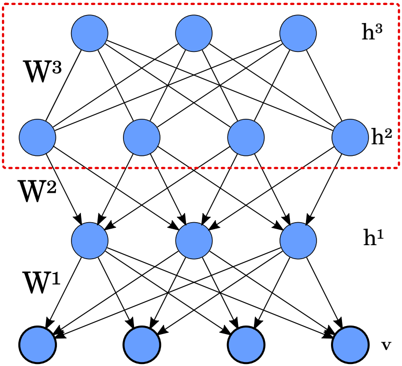
DBN - Layer-wise Training



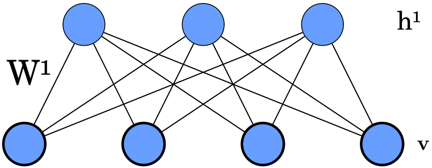
DBN - Layer-wise Training

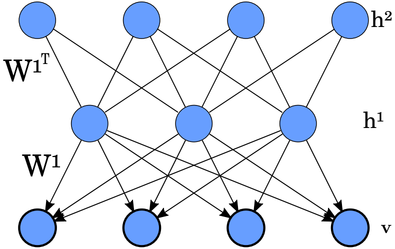


DBN - Layer-wise Training



DBN - Layer-wise Training





Why does this kind of learning work?

RBM and DBN with identical weights $W^1 = {W^1}^T$ have the same joint distribution.

DBN: $p\left(\mathbf{v}, \mathbf{h}^1; W^1, {W^1}^T\right) = \sum_{\mathbf{h}^2} p\left(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; W^1, {W^1}^T\right)$

RBM: $p\left(\mathbf{v}, \mathbf{h}^1; W^1\right)$

$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{h^1}) p(\mathbf{v}|\mathbf{h^1})$

Idea: leave $p(\mathbf{v}|\mathbf{h^1})$ fixed and try to improve $p(h^1)$ via the second RBM (learning a better model of $p(h^1; W^2)$)

Example: MNIST - Deep Belief Network

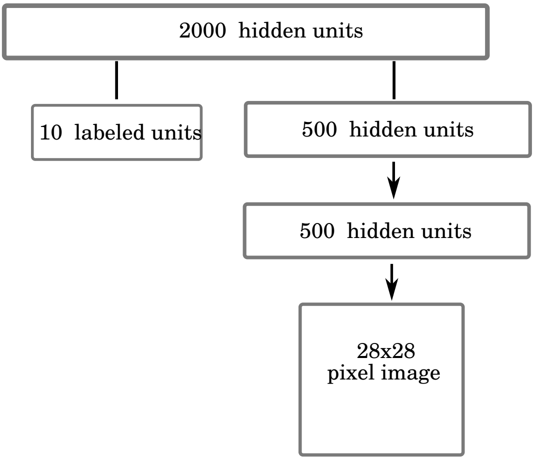


Figure: "A fast learning algorithm for deep belief nets" (Hinton et al., 2006)

Thank you for your attention!