Greedy Layer-Wise Training of Deep Networks

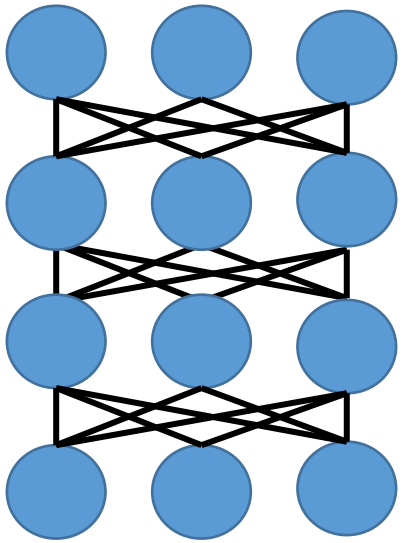
Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle

NIPS 2007

Presented by

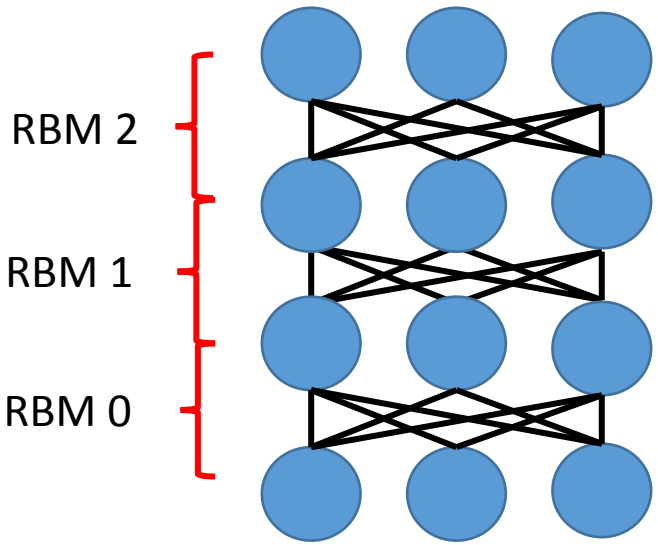
Ahmed Hefny

Story so far …



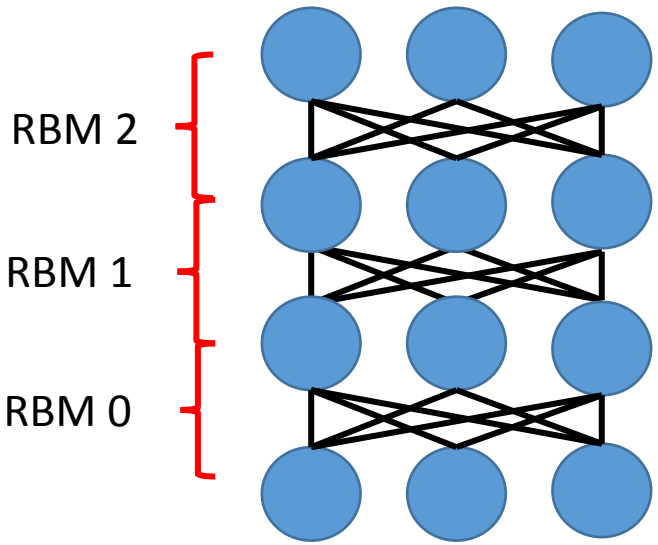
- Deep neural nets are more expressive: Can learn wider classes of functions with less **hidden units (parameters)** and **training examples**.
- Unfortunately they are not easy to train with **randomly initialized** gradient-based methods.

Story so far …



- Hinton et. al. (2006) proposed greedy unsupervised layer-wise training:
  - Greedy layer-wise: Train layers sequentially starting from bottom (input) layer.
  - Unsupervised: Each layer learns a higher-level representation of the layer below. The training criterion does not depend on the labels.
- Each layer is trained as a Restricted Boltzman Machine. (RBM is the building block of Deep Belief Networks).
- The trained model can be fine tuned using a supervised method.

This paper



- Extends the concept to:
  - Continuous variables
  - *Uncooperative* input distributions
  - Simultaneous Layer Training

- Explores variations to better understand the training method:
  - What if we use greedy **supervised** layer-wise training ?
  - What if we replace RBMs with auto-encoders ?

Outline

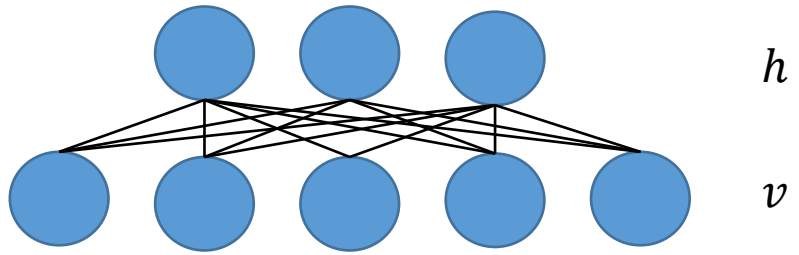
- Review
  - Restricted Boltzman Machines
  - Deep Belief Networks
  - Greedy layer-wise Training
- Supervised Fine-tuning
- Extensions
  - Continuous Inputs
  - Uncooperative Input Distributions
  - Simultaneous Training
- Analysis Experiments

Outline

- Review
  - Restricted Boltzman Machines
  - Deep Belief Networks
  - Greedy layer-wise Training
- Supervised Fine-tuning
- Extensions
  - Continuous Inputs
  - Uncooperative Input Distributions
  - Simultaneous Training
- Analysis Experiments

Restricted Boltzman Machine

Undirected bipartite graphical model with connections between **visible** nodes and **hidden** nodes.

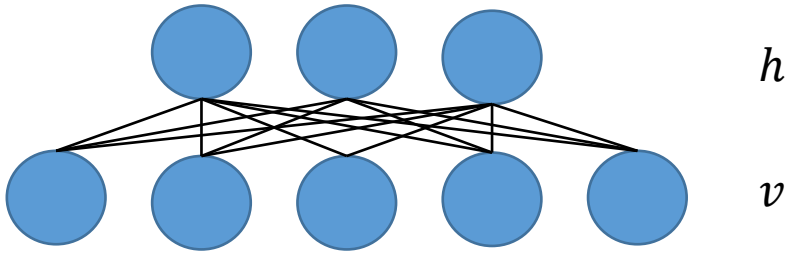


Corresponds to joint probability distribution

$$P(v,h) = \frac{1}{Z}\exp(-energy(v,h))$$
$$= \frac{1}{Z}\exp(v'Wh + b'v + c'h)$$

Restricted Boltzman Machine

Undirected bipartite graphical model with connections between **visible** nodes and **hidden** nodes.



Corresponds to joint probability distribution

$$P(v,h) = \frac{1}{Z}\exp(h'Wv + b'v + c'h)$$

Factorized Conditionals

$$Q(h|v) = \prod_j P(h_j|v)$$

$$P(v|h) = \prod_k P(v_k|h)$$

$$Q(h_j = 1|v) = sigm(c_j + \sum_k W_{jk}v_k)$$

$$P(v_k = 1|h) = sigm(b_k + \sum_j W_{jk}h_j)$$

Restricted Boltzman Machine (Training)

- Given input vectors V_0, adjust $\theta = (W, b, c)$ to increase $\log\ P(V_0)$

$$\log P(v_0) = \log \sum_h P(v_0, h) = \log \sum_h \exp\big(-energy(v_0, h)\big) - \log \sum_{v,h} \exp\big(-energy(v, h)\big)$$

$$\frac{\partial \log P(v_0)}{\partial \theta} = -\sum_h Q(h|v_0) \frac{\partial energy(v_0, h)}{\partial \theta} + \sum_{v,h} P(v, h) \frac{\partial energy(v, h)}{\partial \theta}$$

$$\frac{\partial \log P(v_0)}{\partial \theta_k} = -\sum_h Q(h|v_0) \frac{\partial energy(v_0, h)}{\partial \theta_k} + \sum_v P(v) \sum_{h_k} Q(h_k|v) \frac{\partial energy(v, h)}{\partial \theta_k}$$

Restricted Boltzman Machine (Training)

- Given input vectors V_0, adjust $\theta = (W, b, c)$ to increase $\log\ P(V_0)$

$$\log P(v_0) = \log \sum_h P(v_0, h) = \log \sum_h \exp\big(-energy(v_0, h)\big) - \log \sum_{v,h} \exp\big(-energy(v, h)\big)$$

$$\frac{\partial \log P(v_0)}{\partial \theta} = -\sum_h Q(h|v_0) \frac{\partial energy(v_0, h)}{\partial \theta} + \sum_{v,h} P(v, h) \frac{\partial energy(v, h)}{\partial \theta}$$

$$\frac{\partial \log P(v_0)}{\partial \theta_k} = -\sum_h Q(h|v_0) \frac{\partial energy(v_0, h)}{\partial \theta_k} + \sum_v P(v) \sum_{h_k} Q(h_k|v) \frac{\partial energy(v, h)}{\partial \theta_k}$$

Restricted Boltzman Machine (Training)

- Given input vectors V_0, adjust $\theta = (W, b, c)$ to increase $\log P(V_0)$

$$\log P(v_0) = \log \sum_h P(v_0, h) = \log \sum_h \exp\left(-energy(v_0, h)\right) - \log \sum_{v,h} \exp\left(-energy(v, h)\right)$$

$$\frac{\partial \log P(v_0)}{\partial \theta} = -\sum_h Q(h|v_0) \frac{\partial energy(v_0, h)}{\partial \theta} + \sum_{v,h} P(v, h) \frac{\partial energy(v, h)}{\partial \theta}$$

$$\frac{\partial \log P(v_0)}{\partial \theta_k} = -\sum_h Q(h|v_0) \frac{\partial energy(v_0, h)}{\partial \theta_k} + \sum_v P(v) \sum_{h_k} Q(h_k|v) \frac{\partial energy(v, h)}{\partial \theta_k}$$

Sample h_0 given v_0

Sample v_1 and h_1 using Gibbs sampling

Restricted Boltzman Machine (Training)

- Now we can perform stochastic gradient descent on data log-likelihood

- Stop based on some criterion

  (e.g. reconstruction error $-\log P(v_1 = x | v_0 = x)$

Deep Belief Network

- A DBN is a model of the form

$$P\left(x, g^1, g^2, \ldots, g^l\right) = P(x|g^1)\ \mathrm{P}(g^1|g^2) \ldots P\left(g^{l-2}\middle|g^{l-1}\right)P(g^{l-1}, g^l)$$

$x = g^0$ denotes input variables

g denotes hidden layers of causal variables

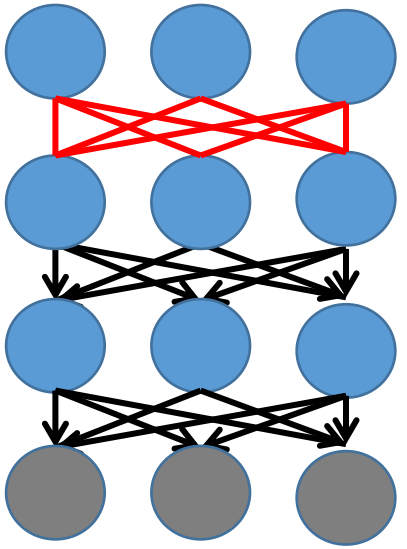
Deep Belief Network

- A DBN is a model of the form

$$P\left(x, g^1, g^2, \ldots, g^l\right) = P(x|g^1)\,\mathrm{P}(g^1|g^2)\ldots P(g^{L-2}|g^{L-1})P(g^{L-1}, g^L)$$

$x = g^0$ denotes input variables

g denotes hidden layers of causal variables



Deep Belief Network

- A DBN is a model of the form

$$P\left(x, g^1, g^2, \ldots, g^l\right) = P(x|g^1)\, \mathrm{P}(g^1|g^2) \ldots P\left(g^{l-2}\middle|g^{l-1}\right)P(g^{l-1}, g^l)$$

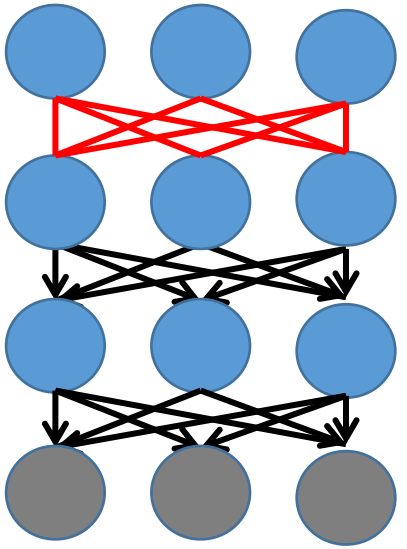
$x = g^0$ denotes input variables

g denotes hidden layers of causal variables

$P(g^{l-1}, g^l)$ is an RBM

$P\left(g^i\middle|g^{i+1}\right) = \prod_j P(g_j^i|g^{i+1})$

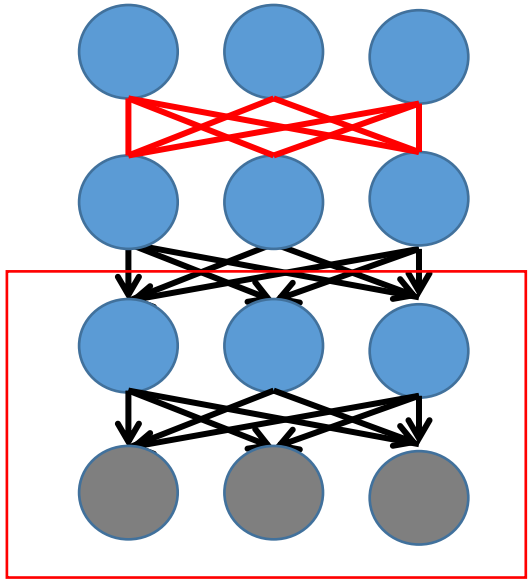
$P\left(g_j^i\middle|g^{i+1}\right) = sigm(b_j^i + \sum_k^{n^{i+1}} W_{kj}^i g_k^{i+1})$



RBM = Infinitely Deep network with tied weights

Greedy layer-wise training

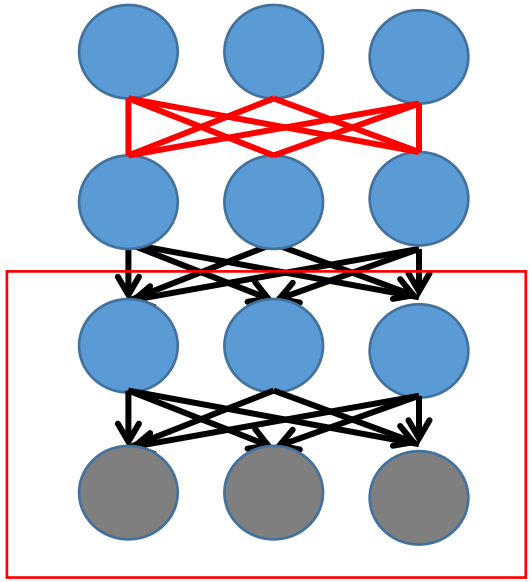
- $P(g^1|g^0)$ is intractable
- Approximate with $Q(g^1|g^0)$
  - Treat bottom two layers as an RBM
  - Fit parameters using contrastive divergence



Greedy layer-wise training

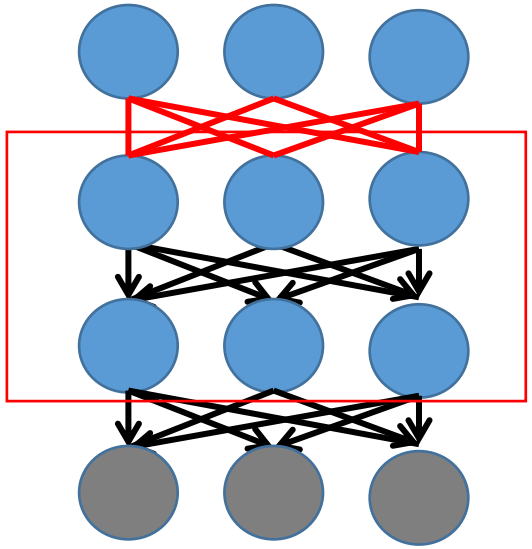
- $P(g^1|g^0)$ is intractable
- Approximate with $Q(g^1|g^0)$
  - Treat bottom two layers as an RBM
  - Fit parameters using contrastive divergence

- That gives an approximate $\hat{P}\ (g^1)$
- We need to match it with $P(g^1)$



Greedy layer-wise training

- Approximate $P\left(g^l \middle| g^{l-1}\right) \approx Q(g^l | g^{l-1})$
  - Treat layers $l-1, l$ as an RBM
  - Fit parameters using contrastive divergence
  - Sample g_0^{l-1} recursively using $Q\left(g^i \middle| g^{i-1}\right)$ starting from g^0



Outline

- Review
  - Restricted Boltzman Machines
  - Deep Belief Networks
  - Greedy layer-wise Training
- Supervised Fine-tuning
- Extensions
  - Continuous Inputs
  - Uncooperative Input Distributions
  - Simultaneous Training
- Analysis Experiments

Supervised Fine Tuning (In this paper)

- Use greedy layer-wise training to initialize weights of all layers except output layer.

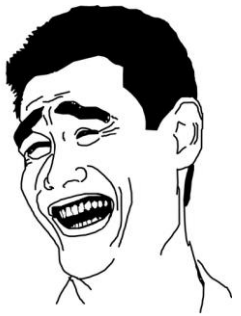
- For fine-tuning, use stochastic gradient descent of a cost function on the outputs where the conditional expected values of hidden nodes are approximated using mean-field.

$$E\left(g^i \mid g^{i-1} = \mu^{i-1}\right) = \mu^i = sigm(b^i + W^i \mu^{i-1})$$

Supervised Fine Tuning (In this paper)

- Use greedy layer-wise training to initialize weights of all layers except output layer.

- Use backpropagation



Outline

- Review
  - Restricted Boltzman Machines
  - Deep Belief Networks
  - Greedy layer-wise Training
- Supervised Fine-tuning
- Extensions
  - Continuous Inputs
  - Uncooperative Input Distributions
  - Simultaneous Training
- Analysis Experiments

Continuous Inputs

- Recall RBMs:
- $Q(h_j|v) \propto Q(h_j, v) \propto \exp(h_j w'v + b_j h_j) \propto \exp((w'v + b_j) \quad h_j) = \exp(a(v)h_j)$

- If we restrict $h_j \in I = \{0,1\}$ then normalization gives us binomial with p given by sigmoid.
- Instead, if $I = [0, \infty]$ we get exponential density
- If I is closed interval then we get truncated exponential

Continuous Inputs (Case for truncated exponential [0,1])

- Sampling

  For truncated exponential, inverse CDF can be used

  $$\mathrm{h_j} = F^{-1}(U) = \frac{\log(1-U\times(1-\exp(a(v))}{a(v)}$$

  where U is sampled uniformly from [0,1]

- Conditional Expectation

  $$E[h_j|v] = \frac{1}{1-\exp(-a(v))} - \frac{1}{a(v)}$$

Continuous Inputs

- To handle Gaussian inputs, we need to augment the energy function with a term quadratic in h.
- For a diagonal covariance matrix

$$P\left(h_j \middle| v\right) = a(v)h_j + d_j h_j^2$$

Giving

$$E\left[h_j \middle| z\right] = a(x)/2d^2$$

Continuous Hidden Nodes ?

Continuous Hidden Nodes ?

- Truncated Exponential

$$E\left[h_j \middle| v\right] = \frac{1}{1 - \exp(-a(v))} - \frac{1}{a(v)}$$

- Gaussian

$$E\left[h_j \middle| v\right] = a(v)/2d^2$$

Uncooperative Input Distributions

- Setting

$$x \sim p(x)$$
$$y = f(x) + noise$$

- No particular relation between *p* and *f,* (e.g. Gaussian and sinus)

Uncooperative Input Distributions

- Setting

$$x \sim p(x)$$
$$y = f(x) + noise$$

- No particular relation between *p* and *f,* (e.g. Gaussian and sinus)

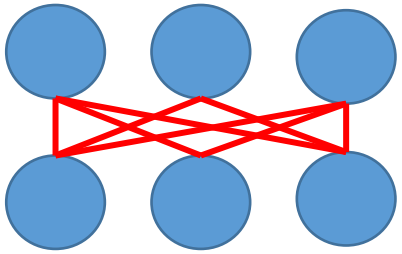
- Problem: Unsupvervised pre-training may not help prediction

Outline

- Review
  - Restricted Boltzman Machines
  - Deep Belief Networks
  - Greedy layer-wise Training
- Supervised Fine-tuning
- Extensions
- Analysis Experiments

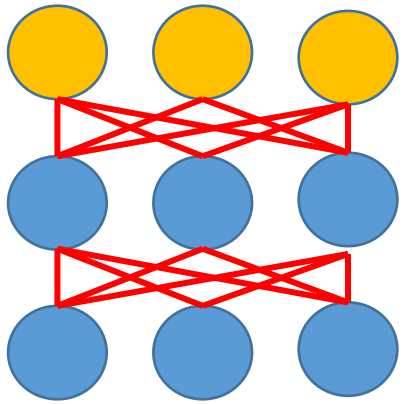
Uncooperative Input Distributions

- Proposal: Mix unsupervised and supervised training **for each layer**



Stochastic Gradient of input log likelihood
by Contrastive Divergence

Temp. Ouptut Layer



Stochastic Gradient of prediction error

Combined Update

Simultaneous Layer Training

- Greedy Layer-wise Training
- For each layer
  - Repeat Until Criterion Met
    - Sample layer input (by recursively applying trained layers to data)
    - Update parameters using contrastive divergence

Simultaneous Layer Training

- Simultaneous Training
- Repeat Until Criterion Met
  - Sample input to all layers
  - Update parameters of all layers using contrastive divergence

- Simpler: One criterion for the entire network
- Takes more time

Outline

- Review
  - Restricted Boltzman Machines
  - Deep Belief Networks
  - Greedy layer-wise Training
- Supervised Fine-tuning
- Extensions
  - Continuous Inputs
  - Uncooperative Input Distributions
  - Simultaneous Training
- Analysis Experiments

Experiments

- Does greedy unsupervised pre-training help ?
- What if we replace RBM with auto-encoders ?
- What if we do greedy *supervised* pre-training ?

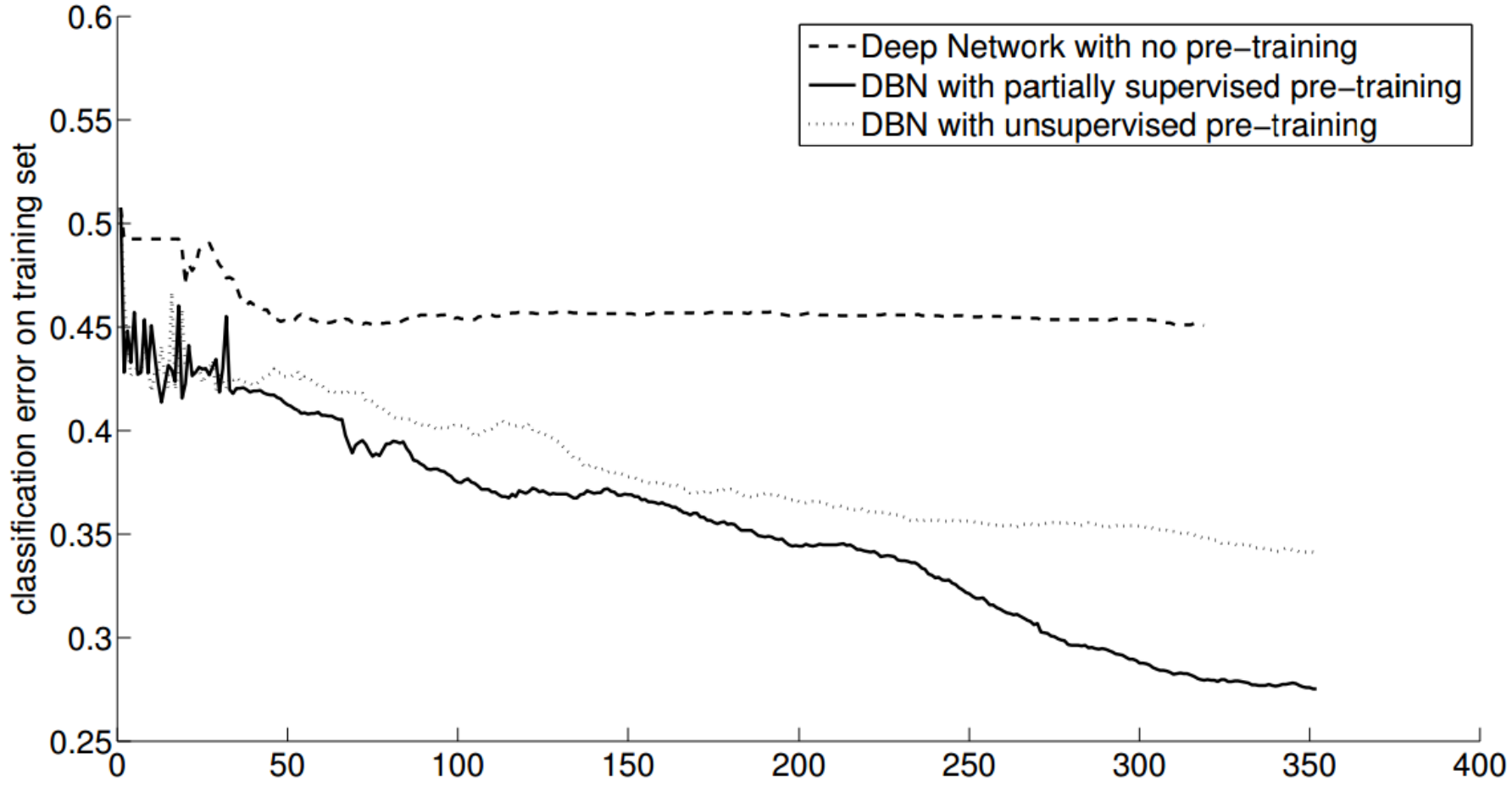
- Does continuous variable modeling help ?
- Does partially supervised pre-training help ?

Experiment 1

- **Does greedy unsupervised pre-training help ?**
- What if we replace RBM with auto-encoders ?
- What if we do greedy *supervised* pre-training ?

- **Does continuous variable modeling help ?**
- **Does partially supervised pre-training help ?**

Experiment 1



Experiment 1

	Abalone			Cotton		
	train.	valid.	test.	train.	valid.	test.
1. Deep Network with no pre-training	4.23	4.43	4.2	45.2%	42.9%	43.0%
2. Logistic regression	.	.	.	44.0%	42.6%	45.0%
3. DBN, binomial inputs, unsupervised	4.59	4.60	4.47	44.0%	42.6%	45.0%
4. DBN, binomial inputs, partially supervised	4.39	4.45	4.28	43.3%	41.1%	43.7%
5. DBN, Gaussian inputs, unsupervised	4.25	4.42	4.19	35.7%	34.9%	35.8%
6. DBN, Gaussian inputs, partially supervised	4.23	4.43	4.18	27.5%	28.4%	31.4%

Experiment 1(MSE and Training Errors)

	Abalone			Cotton		
	train.	valid.	test.	train.	valid.	test.
1. Deep Network with no pre-training	4.23	4.43	4.2	45.2%	42.9%	43.0%
2. Logistic regression	·	·	·	44.0%	42.6%	45.0%
3. DBN, binomial inputs, unsupervised	4.59	4.60	4.47	44.0%	42.6%	45.0%
4. DBN, binomial inputs, partially supervised	4.39	4.45	4.28	43.3%	41.1%	43.7%
5. DBN, Gaussian inputs, unsupervised	4.25	4.42	4.19	35.7%	34.9%	35.8%
6. DBN, Gaussian inputs, partially supervised	4.23	4.43	4.18	27.5%	28.4%	31.4%

Partially Supervised < Unsupervised Pre-training < No Pre-training

Gaussian < Binomial

Experiment 2

- **Does greedy unsupervised pre-training help ?**
- **What if we replace RBM with auto-encoders ?**
- **What if we do greedy *supervised* pre-training ?**

- Does continuous variable modeling help ?
- Does partially supervised pre-training help ?

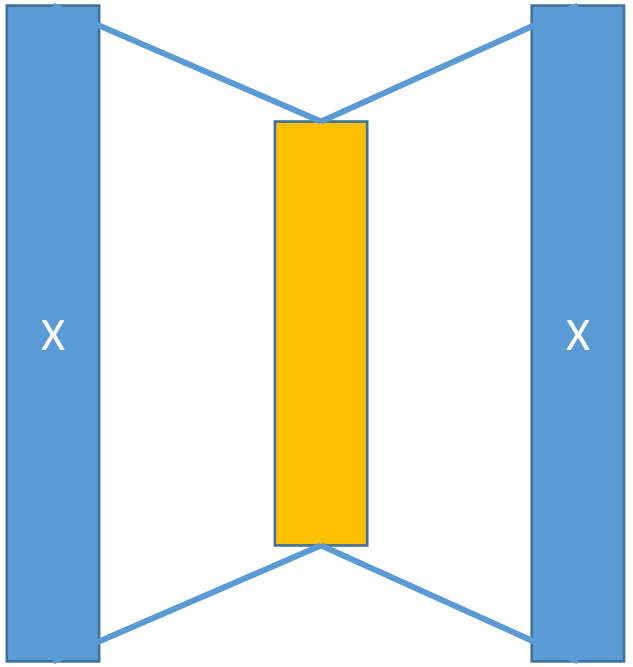
Experiment 2

- Auto Encoders
- Learn a compact representation to reconstruct X

$$p(x) = sigm\big(c + Wsigm(b + W'x)\big)$$

- Trained to minimize reconstruction cross-entropy

$$R = -\sum_i x_i \log p(x_i) + (1 - x_i) \log p(1 - x_i)$$



Experiment 2

(500~1000) layer width

20 nodes in last two layers

	Experiment 2			Experiment 3		
	train.	valid.	test	train.	valid.	test
DBN, unsupervised pre-training	0%	1.2%	1.2%	0%	1.5%	1.5%
Deep net, auto-associator pre-training	0%	1.4%	1.4%	0%	1.4%	1.6%
Deep net, supervised pre-training	0%	1.7%	2.0%	0%	1.8%	1.9%
Deep net, no pre-training	.004%	2.1%	2.4%	.59%	2.1%	2.2%
Shallow net, no pre-training	.004%	1.8%	1.9%	3.6%	4.7%	5.0%

Experiment 2

- Auto-encoder pre-training outperforms supervised pre-training but is still outperformed by RBM.

- Without pre-training, deep nets do not generalize well, but they can still fit the data if the output layers are wide enough.

Conclusions

- Unsupervised pre-training is important for deep networks.
- Partial supervision further enhances results, especially when input distribution and the function to be estimated are not closely related.
- Explicitly modeling conditional inputs is better than using binomial models.

Thanks