Deep Belief Network and Deep Boltamann Machines

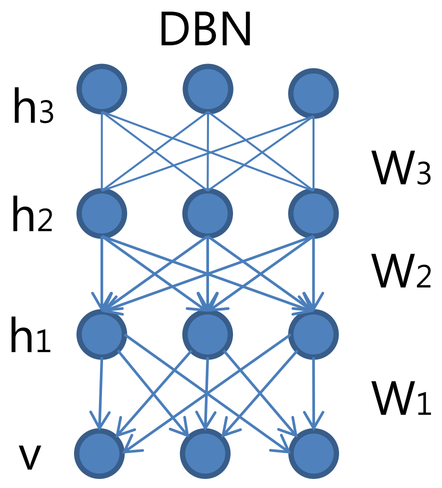
Yonghao He
NLPR, Institute of Automation,
Chinese Academy of Sciences

Content

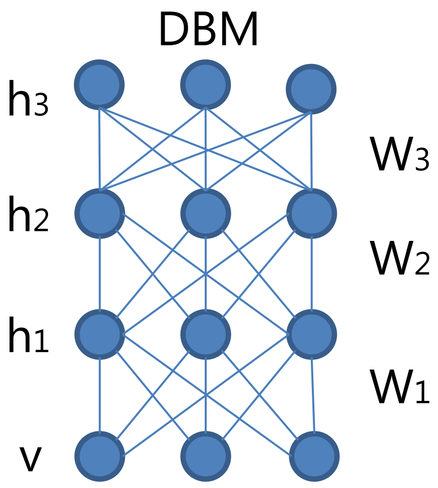
- Basic concepts
- Training procedure
- Applications
- Some open problems
- References

Content

Basic concepts



Mixed: directed and undirected

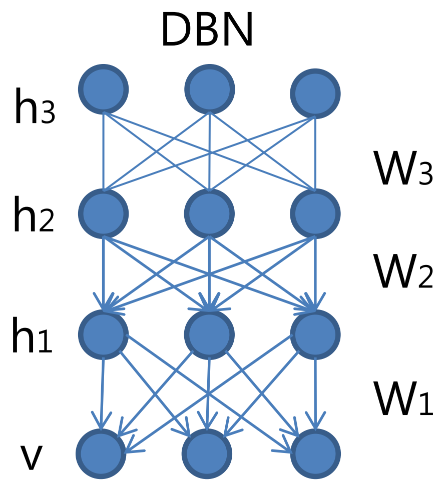


undirected

Here, all nodes are stochastic binary variables.

From the perspective of the traditional neural network, both networks belong to deep generative model and have tons of parameters. If we use BP algorithm to solve all parameters, we should need a huge amount of labeled data for training. An alternative way is a maximum likelihood estimation(a generative model always results in this way) of the observed data without too much supervised learning.(a small amount of labeled data can be used to fine-tune.)

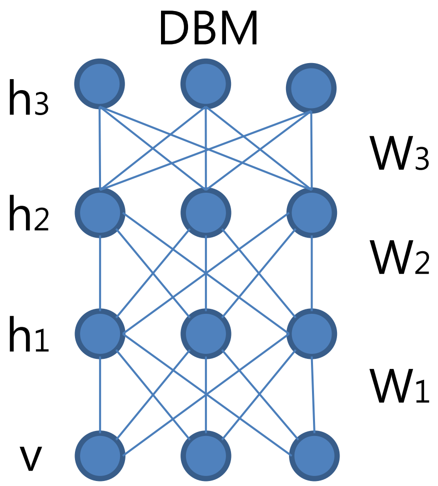
Basic concepts(DBN)



Model joint distribution[6]:

$$p(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \mathbf{v}|\boldsymbol{\theta}) = \prod_i \mathrm{Ber}(v_i|\mathrm{sigm}(\mathbf{h}_1^T \mathbf{w}_{1i}) \prod_j \mathrm{Ber}(h_{1j}|\mathrm{sigm}(\mathbf{h}_2^T \mathbf{w}_{2j}) \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{kl} h_{2k} h_{3l} W_{3kl}\right)$$

Basic concepts(DBM)



Model joint distribution[6]:

$$p(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \mathbf{v}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{ij} v_i h_{1j} W_{1ij} + \sum_{jk} h_{1j} h_{2j} W_{2jk} + \sum_{kl} h_{2k} h_{3l} W_{3kl} \right)$$

Basic concepts(DBN)

In directed nets, there exists a classic problem: explaining away.[1]

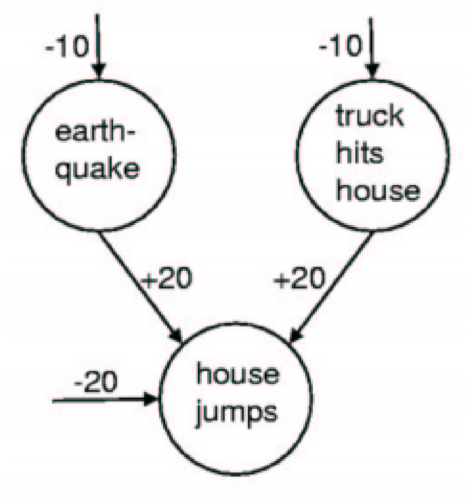


Figure 2: A simple logistic belief net containing two independent, rare causes that become highly anti-correlated when we observe the house jumping. The bias of -10 on the earthquake node means that, in the absence of any observation, this node is e^{10} times more likely to be off than on. If the earthquake node is on and the truck node is off, the jump node has a total input of 0 which means that it has an even chance of being on. This is a much better explanation of the observation that the house jumped than the odds of e^{-20} which apply if neither of the hidden causes is active. But it is wasteful to turn on both hidden causes to explain the observation because the probability of them both happening is $e^{-10} \times e^{-10} = e^{-20}$. When the earthquake node is turned on it "explains away" the evidence for the truck node.

For simplicity, E denotes "earthquake'", T denotes "truck hits house" and H denotes "house jumps".

E or T	0	1
	0.9999546	0.0000454

E T	H=0	H=1
0 0	0.99999999799	0.00000000201
1 0	0.5	0.5
0 1	0.5	0.5
1 1	0	1

$$P(T=1|H=1) = \frac{P(T=1,H=1)}{P(H=1)} = \frac{\sum_E P(E,T=1,H=1)}{\sum_{E,T} P(E,T,H=1)}$$

$$= \frac{0.0000227026}{0.00004540406} = 0.5$$

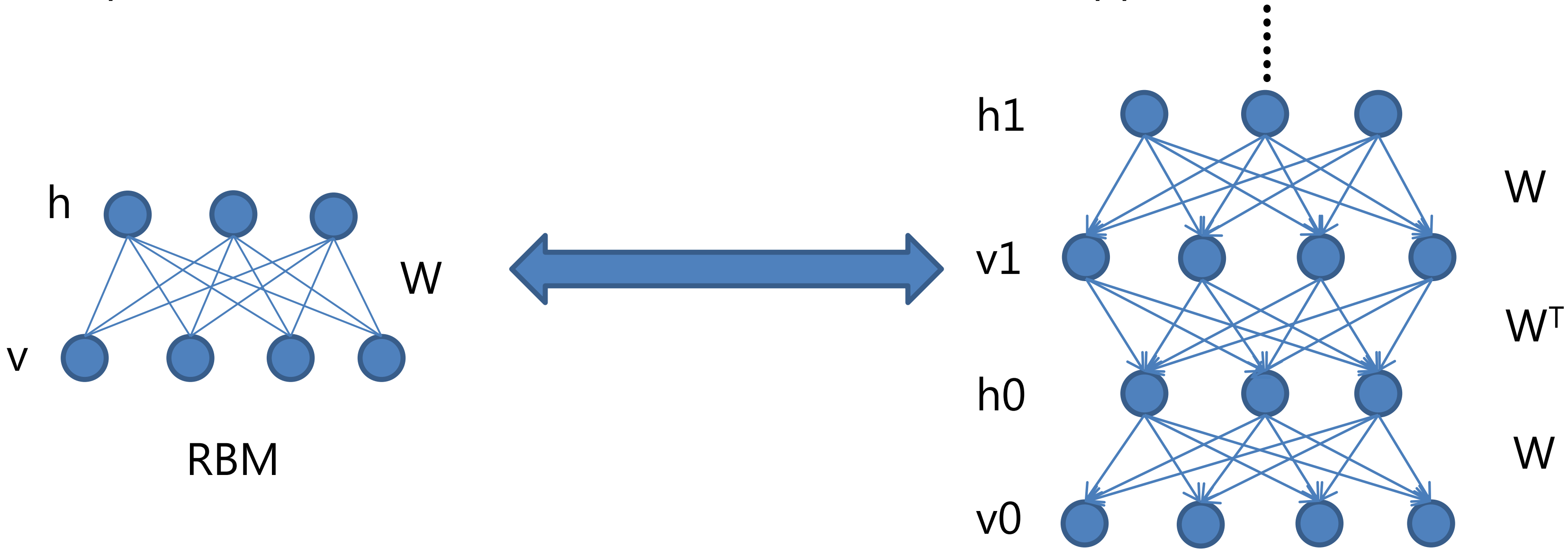
$$P(T=1|H=1,E=1) = \frac{P(T=1,H=1,E=1)}{P(H=1,E=1)}$$

$$= \frac{P(T=1,H=1,E=1)}{\sum_T P(T,H=1,E=1)} = \frac{0.00000000206}{0.0000227026} = 0.0000908$$

Basic concepts(DBN)[1]

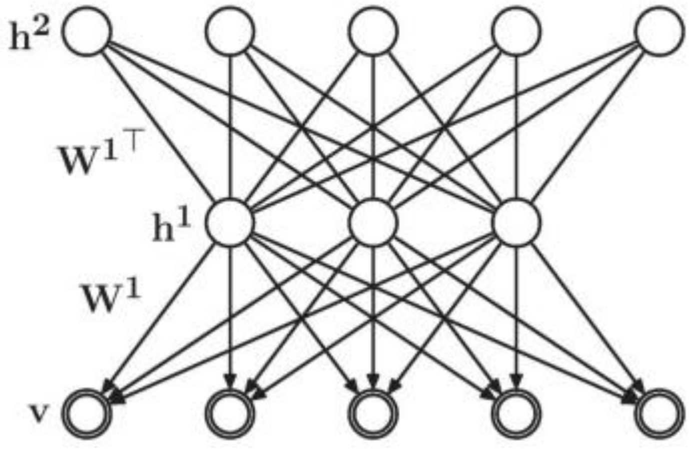
According to the "explaining away", we know that once we get an observation and one of its causes, then the other causes are hardly turned on. This is so-called "explaining away". From the graph, E and T are originally independent. But when we observed a state of H, E and T are not independent any more. So this makes it difficult to do inference in directed belief nets.

Then, we want to eliminate the "explaining away". A feasible way is to use "complementary prior". This makes the posterior distribution of hidden layers fully factorized, and "complementary prior" seems like a mere trick for making directed models equivalent to a undirected ones.(details are in the Appendix A of [1])



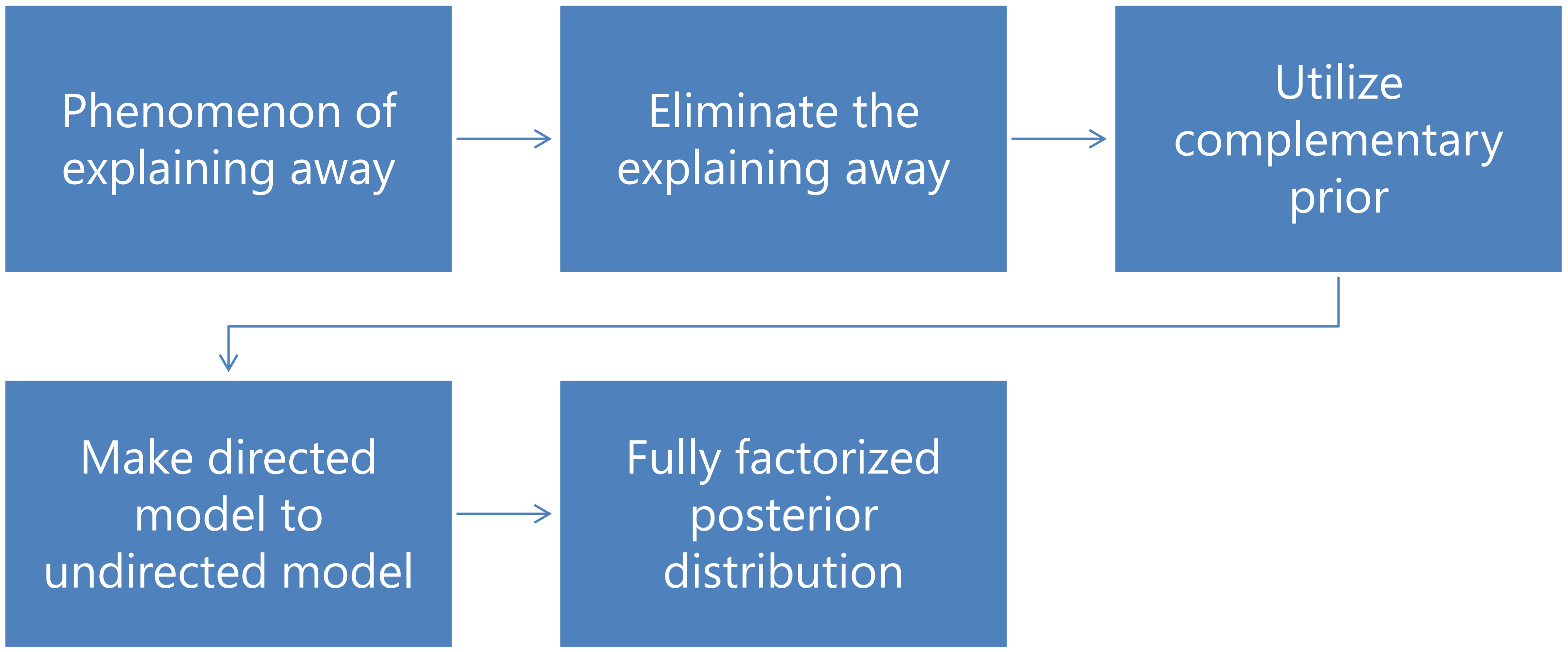
A infinite logistic belief net with tied weights

Basic concepts(DBN)



In[6], We can show that $p(\boldsymbol{h}_1, \boldsymbol{v}|\boldsymbol{W}_1) = \sum_{\boldsymbol{h}_2} p(\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{v}|\boldsymbol{W}_1) = \frac{1}{Z(\boldsymbol{W}_1)} \exp(\boldsymbol{v}^T \boldsymbol{W}_1 \boldsymbol{h}_1)$, which is equivalent to a RBM without biases.

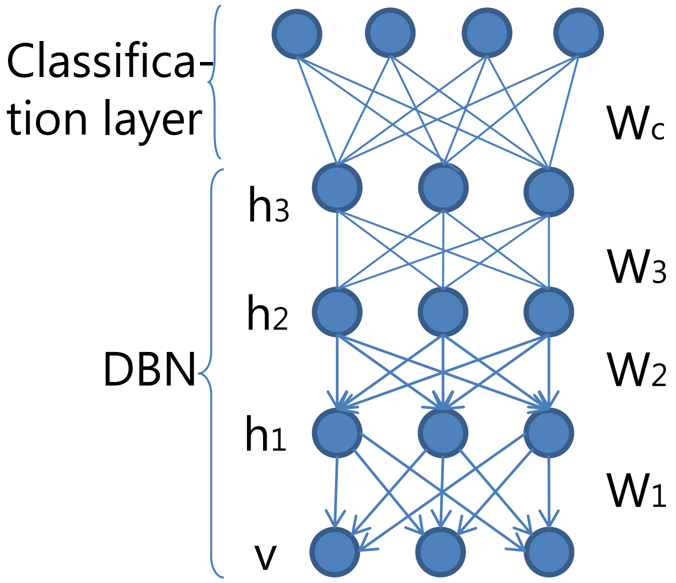
Basic concepts(DBN)


Phenomenon of explaining away
Eliminate the explaining away
Utilize complementary prior
Make directed model to undirected model
Fully factorized posterior distribution

This procedure leads to a very efficient greedy layer-wise training method ---- when we strictly train multiple common RMBs and then stack them from bottom to top, we will obtain a DBN(not a DBM!!).

Content

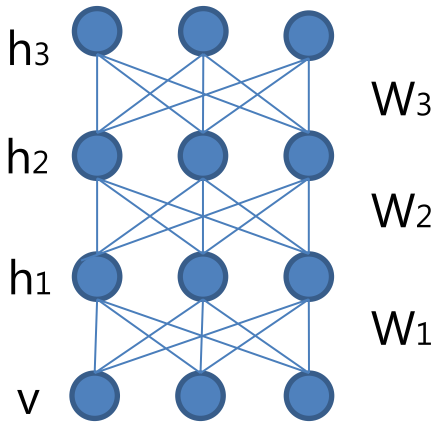
Training procedure(DBN)[1]



How to train a DBN shown on the left?

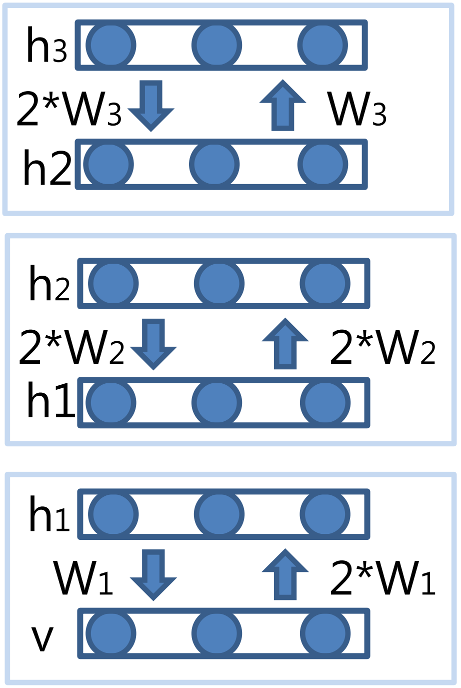
1. **Layer-wise pretraining**: train three common RBMs progressively. The lower RBM's output becomes the input of the higher RBM. These RBMs are stacked.
2. **Fine-tuning**: This step needs labeled data. Assume that it's a classification task. So we need to add an extra layer to indicate the labels. Back propagation algorithm is always used to tune all these weights to data.

Training procedure(DBM)[2]



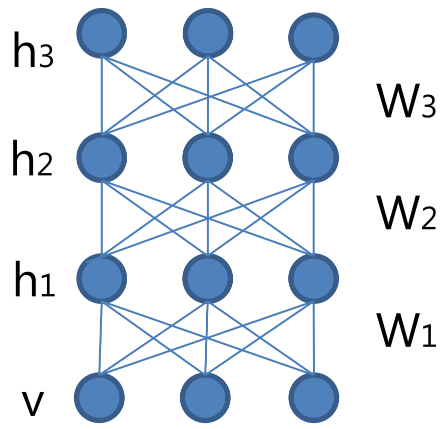
How to train a DBM shown on the left?

1. **Layer-wise pretraining**: Two modified RBMs are trained for the first and the last layers. This is shown as follows:



Why do like this? And we do so, they form a DBM.(not a DBN!!)

Training procedure(DBM)[2]



How to train a DBM shown on the left?

2. **Joint layers training and fine-tuning**: It's special here,
   we need to construct another multilayer neural network to fine tune the DBM. This is shown below:

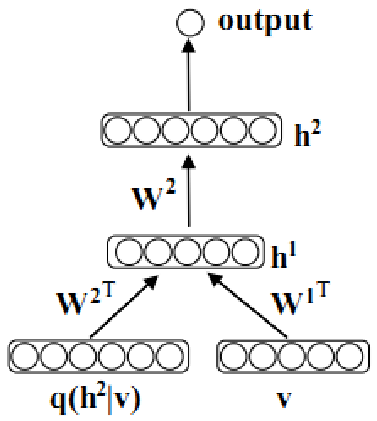


Figure 3: After learning, DBM is used to initialize a multilayer neural network. The marginals of approximate posterior $q(h_j^2 = 1|\mathbf{v})$ are used as additional inputs. The network is fine-tuned by backpropagation.

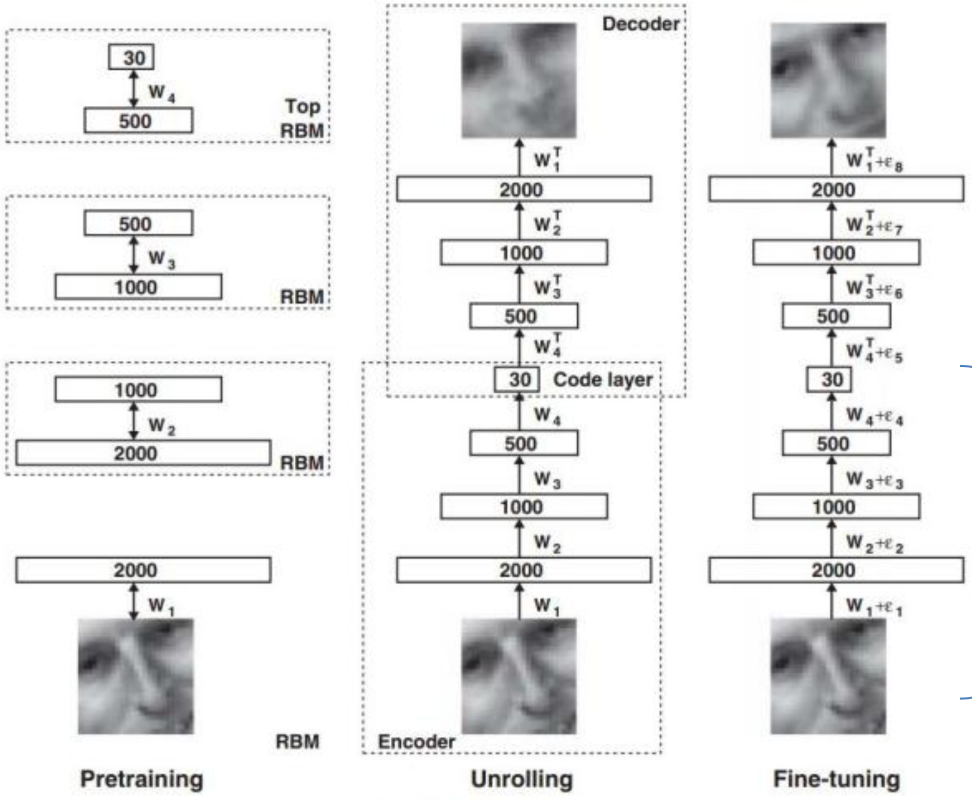
This is from [2] with only two hidden layers. q(h2|v) is obtained by using mean-field inference.

If we have 3 hidden layers, how to construct this network?
I'm not clear about this step.

Content

Applications(DBN)[4]

A brief review of Autoencoder:



Auto Encoder

This process can be viewed as dimensionality reduction or feature learning.
In hinton's Science Paper, he also showed the super ability of the DBN to do unsupervised clustering and dimensionality reduction simultaneously.
(this will be shown in next slide.)

Actually, this part is a DBN.

Applications(DBN)[4]

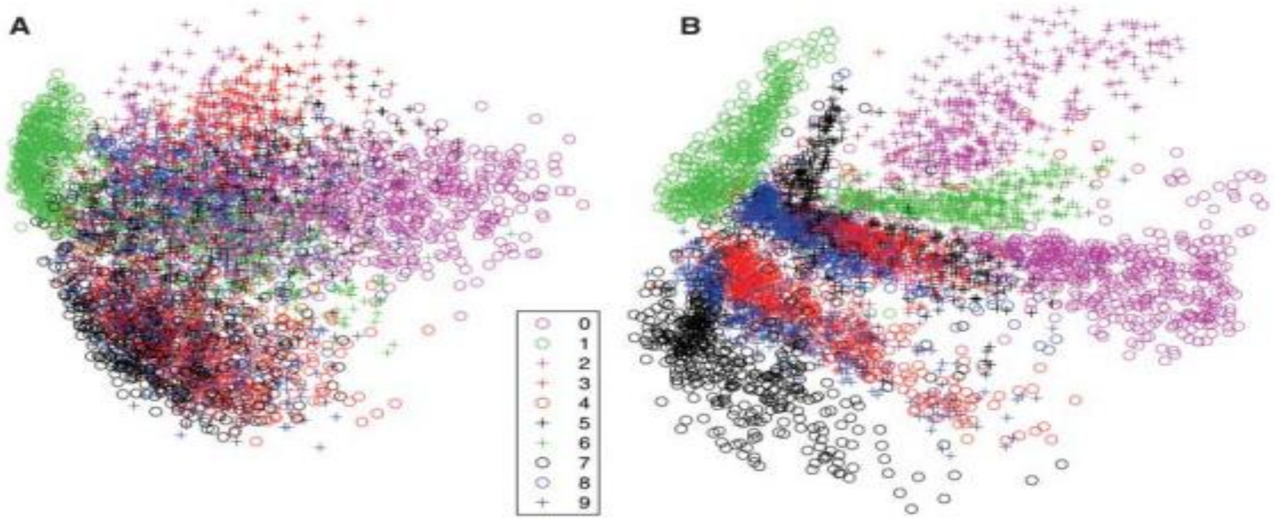


Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

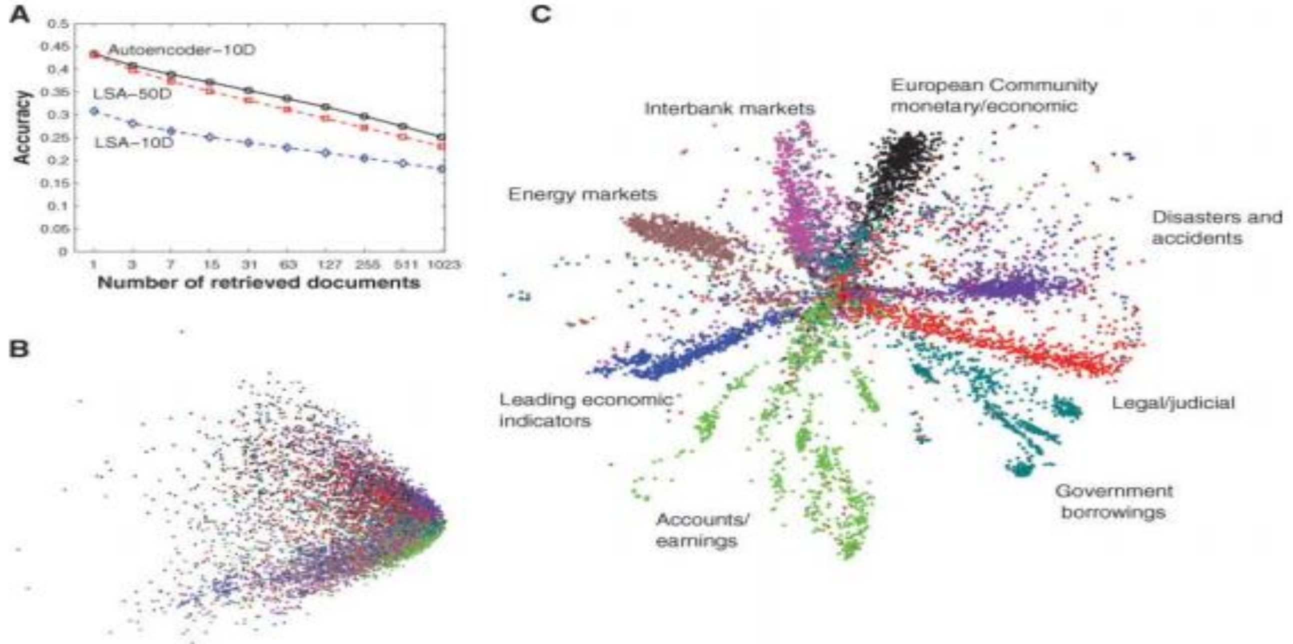


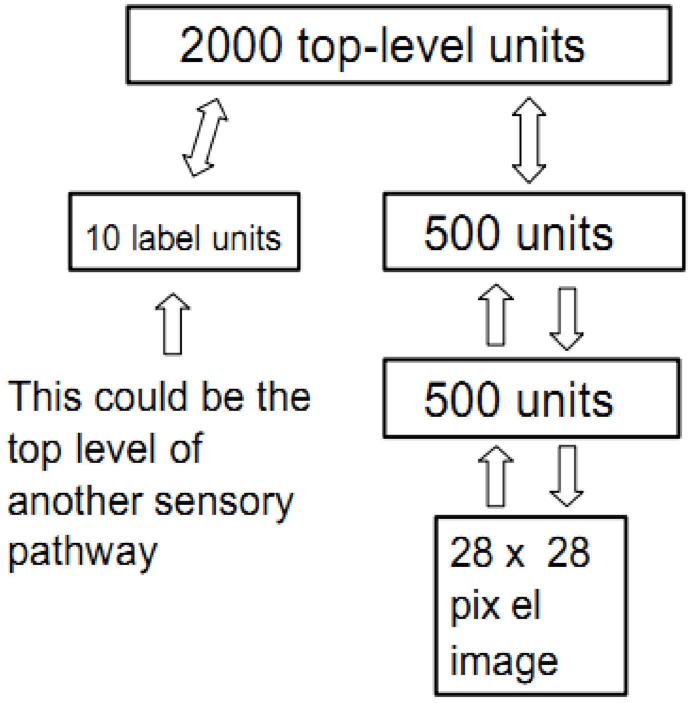
Fig. 4. (**A**) The fraction of retrieved documents in the same class as the query when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries. (**B**) The codes produced by two-dimensional LSA. (**C**) The codes produced by a 2000-500-250-125-2 autoencoder.

Applications(DBN)[1]

Hand-written digit recognition using DBN on the MNIST database

60000 training images and 10000 test images.

Only 44000 training images are used and they are divided into 440 mini-batches each containing 100 examples of each digit class.
The generalization performance is 1.25% error rate.



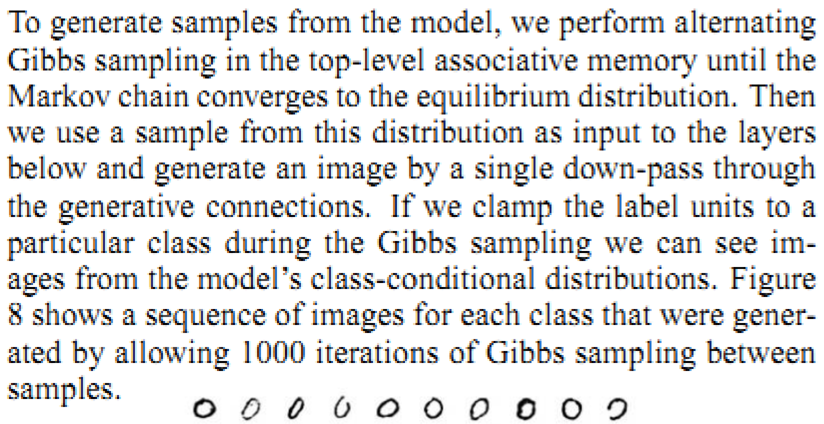
Some **tricks** in the training procedure:
1. In pretraining, visible layer of each RBM has real-valued activities. But hidden layer uses stochastic binary values. (error rate 2.49%?)
2. In fine-tuning, first 10000 images from the remainder of the training set is used. Then they use all 60000 images in the training set to fine-tune until achieving 1.25% error rate.

Applications(DBN)[1]

Another two interesting experiments about DBN

We can use this trained DBN to generate fake samples.

The first experiment:



To generate samples from the model, we perform alternating Gibbs sampling in the top-level associative memory until the Markov chain converges to the equilibrium distribution. Then we use a sample from this distribution as input to the layers below and generate an image by a single down-pass through the generative connections. If we clamp the label units to a particular class during the Gibbs sampling we can see images from the model's class-conditional distributions. Figure 8 shows a sequence of images for each class that were generated by allowing 1000 iterations of Gibbs sampling between samples.

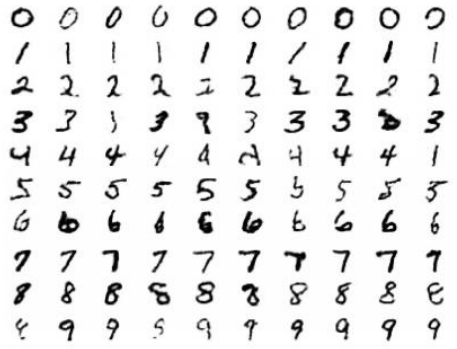
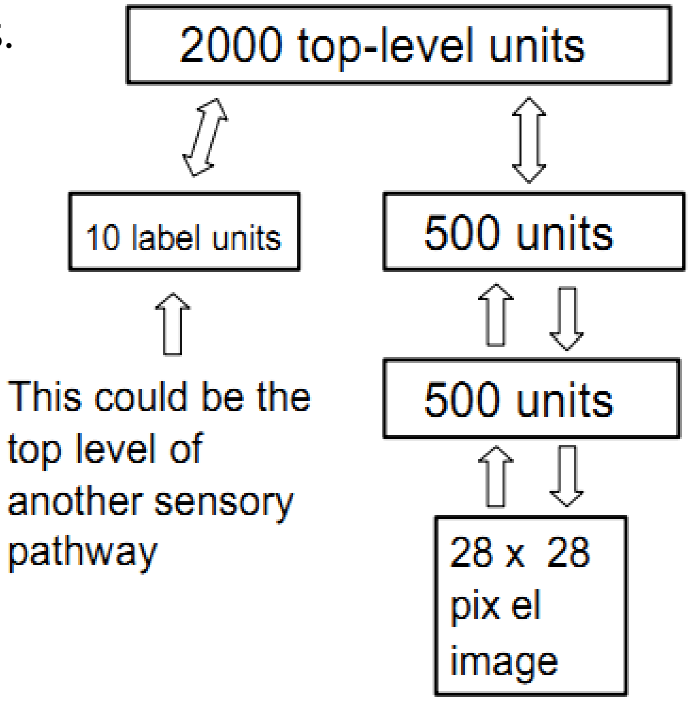


Figure 8: Each row shows 10 samples from the generative model with a particular label clamped on. The top-level associative memory is run for 1000 iterations of alternating Gibbs sampling between samples.



Applications(DBN)[1]

The second experiment:

We can also initialize the state of the top two layers by providing a random binary image as input. Figure 9 shows how the class-conditional state of the associative memory then evolves when it is allowed to run freely, but with the label clamped. This internal state is "observed" by performing a down-pass every 20 iterations to see what the associative memory has in mind.

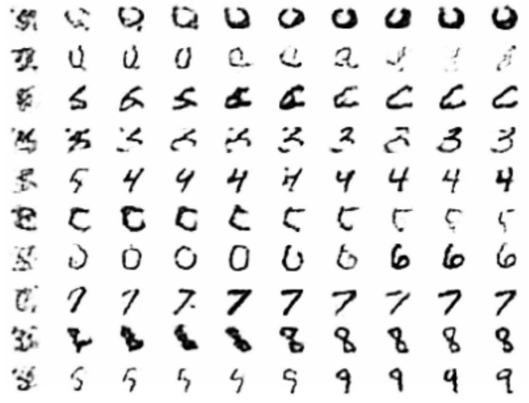
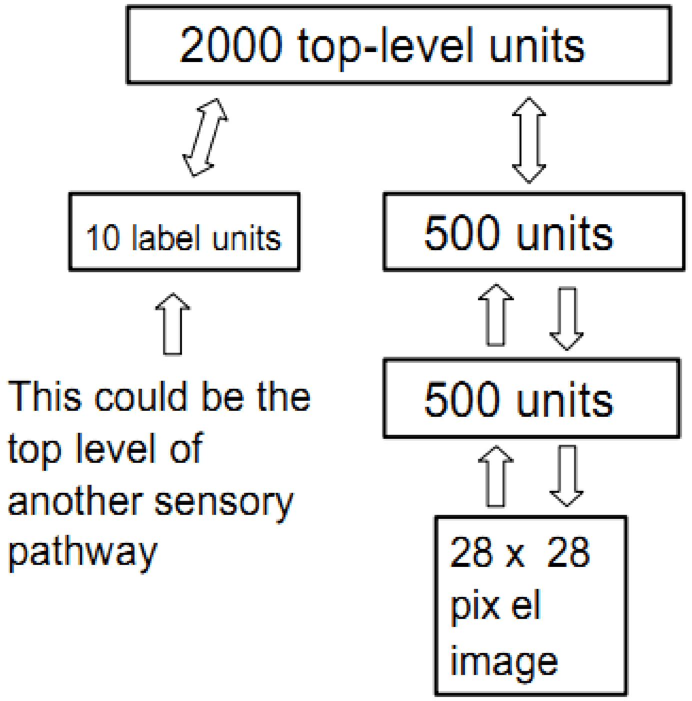


Figure 9: Each row shows 10 samples from the generative model with a particular label clamped on. The top-level associative memory is initialized by an up-pass from a random binary image in which each pixel is on with a probability of 0.5. The first column shows the results of a down-pass from this initial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



Applications(DBM)[2]

Hand-written digit recognition and generating using DBM on the MNIST database

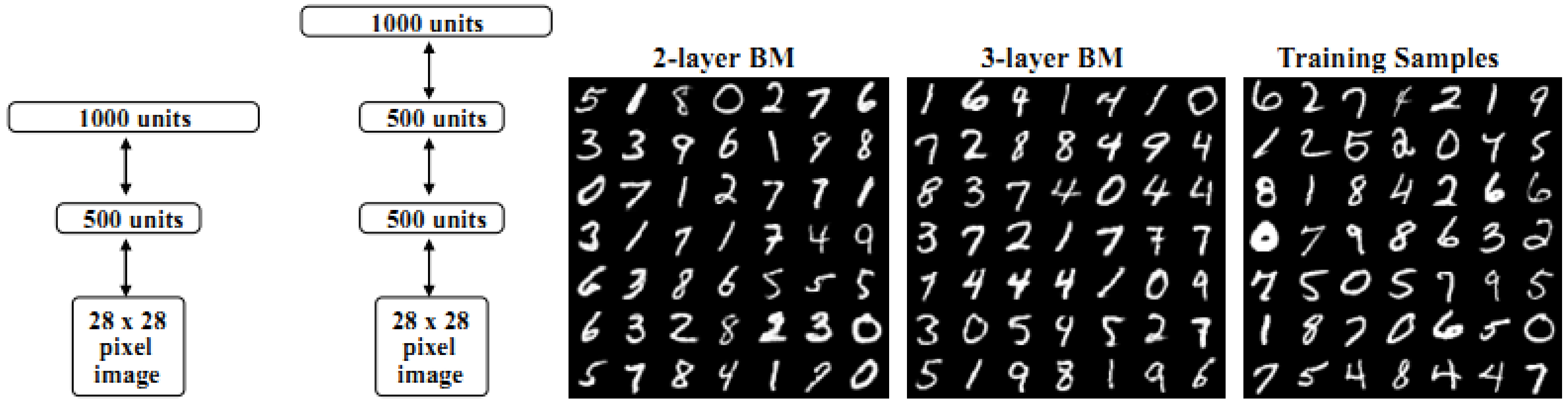


Figure 4: **Left:** Two deep Boltzmann machines used in experiments. **Right:** Random samples from the training set, and samples generated from the two deep Boltzmann machines by running the Gibbs sampler for 100,000 steps. The images shown are the *probabilities* of the binary visible units given the binary states of the hidden units.

For recognition task, the two-layer BM achieves an error rate of 0.95% and the three-layer BM gains a error rate of 1.01%. Note that this is better than DBN.

I'm so clear about how to generate the fake samples here.

Content

Some open problems

- Deep learning, how deep?
- DBN vs DBM, which is better dominantly? Or DBN is fit for some tasks and DBM is suitable for some others?
- How to specify the number of hidden units?
- When we encounter the big data, how to training the terrible DBN or DBM?
  Ask GPUs or distributed computing for help?
- If we want to make a big progress in such kind of neural networks, what entry points will lead to surprising breakthroughs? Change graphical structure based on the latest discoveries in Brain Science? Develop a much faster and efficient algorithms for training?

Content

- Basic concepts
- Training procedure
- Applications
- Some open problems
- References

References

[1] G. Hinton, S. Osindero and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006
[2] R. Salakhutdinov and G. Hinton. Deep Boltzmann Machines. *ICAIS*, 2009
[3] R. Salakhutdinov and G. Hinton. A Better Way to Pretrain Deep Boltzmann Machines. *NIPS*, 2013
[4] G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 2006
[5] G. Hinton. Learning multiple layers of representation. *Cognitive Science*, 2007
[6] Kevin P. Murphy. Machine Learning: A Probabilistic Perspective. Chapter 28 : Deep Learning. The MIT Press, 2012

Thank you