Contrastive Divergence and Deep Learning

Jianlin Cheng, PhD

Computer Science Department

University of Missouri, Columbia

Fall, 2013

- **Big Data**
- **Deep Learning**

Deep Learning Network

Google acquires U of T neural networks company

SHARE THIS

Sara Franca

University Professor **Geoffrey Hinton** and two of his graduate students from the Department of Computer Science have sold their startup company to Google Inc.

From left: Ilya Sutskever, Alex Krizhevsky and University Professor Geoffrey Hinton of the University of Toronto's Department of Computer Science (photo by John Guatto)

Google acquired the company, incorporated by **Alex Krizhevsky**, **Ilya Sutskever** and Hinton in 2012, for its research on deep neural networks. Also known as "deep learning" for computers, this research involves helping machines understand context.

Hinton is world-renowned for his work with machine learning and artificial intelligence. His neural networks research has profound implications for areas such as speech recognition, computer vision and language understanding.

"Geoffrey Hinton's research is a magnificent example of disruptive innovation with roots in basic research," said U of T's president, Professor **David Naylor**."The discoveries of brilliant researchers, guided freely by their expertise, curiosity, and intuition, lead eventually to practical applications no one could have imagined, much less requisitioned.

Facebook Launches Advanced AI Effort to Find Meaning in Your Posts

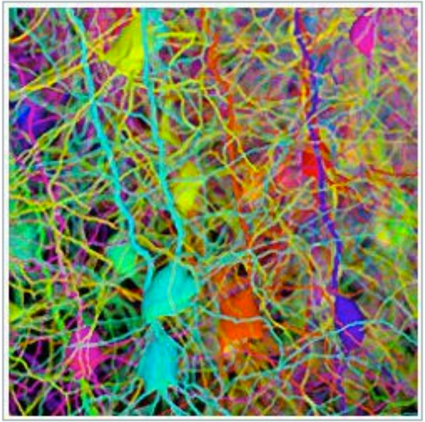
A technique called deep learning could help Facebook understand its users and their data better.

By Tom Simonite on September 20, 2013

Deep Learning Comes of Age

By Gary Anthes
Communications of the ACM, Vol. 56 No. 6, Pages 13-15
10.1145/2461256.2461262
Comments

VIEW AS: SHARE:

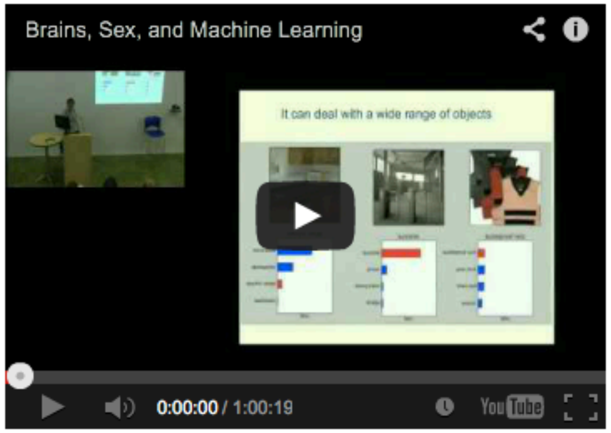


Rainbow brainwaves made from a computer simulation of pyramidal neurons found in the cerebral cortex.

Credit: Hermann Cuntz

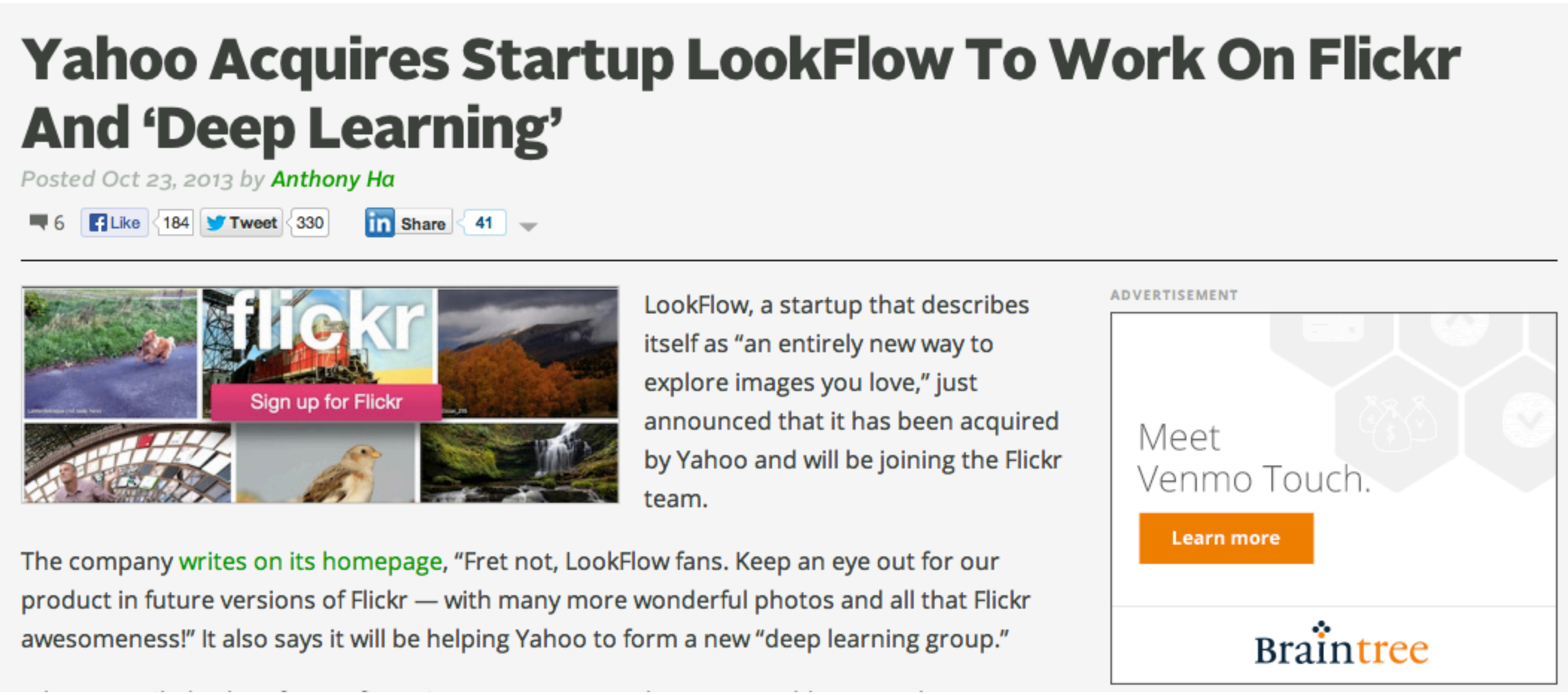
Improvements in algorithms and application architectures, coupled with the recent availability of very fast computers and huge datasets, are enabling major increases in the power of machine learning systems. In particular, multilayer artificial neural networks are producing startling improvements in the accuracy of computer vision, speech recognition, and other applications in a field that has become known as "deep learning."

Artificial neural networks ("neural nets") are patterned after the arrangement of neurons in the brain, and the connections, or synapses, between the neurons. Work on neural nets dates to the 1960s; although conceptually compelling, they proved difficult to apply effectively, and they did not begin to find broad commercial use until the early 1990s.


Brains, Sex, and Machine Learning
It can deal with a wide range of objects
0:00:00 / 1:00:19

Neural nets are systems of highly interconnected, simple processing elements. The behavior of the net changes according to the "weights" assigned to each connection, with the output of any node determined by the weighted sum of its inputs. The nets do not work according to hand-coded rules, as with traditional computer programs; they must be trained, which involves an automated process of successively changing the inter-nodal weights in order to minimize the difference between the desired output and the actual output. Generally, the more input data used for this training, the better the results.

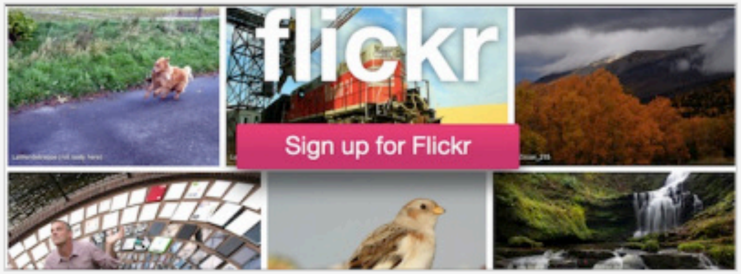
For years, most neural nets contained a single layer of "feature detectors" and were trained mainly with labeled data in a process called "supervised" training. In these kinds of networks, the system is shown an input and told what



Yahoo Acquires Startup LookFlow To Work On Flickr And 'Deep Learning'

Posted Oct 23, 2013 by *Anthony Ha*

6 Like 184 Tweet 330 Share 41



LookFlow, a startup that describes itself as "an entirely new way to explore images you love," just announced that it has been acquired by Yahoo and will be joining the Flickr team.

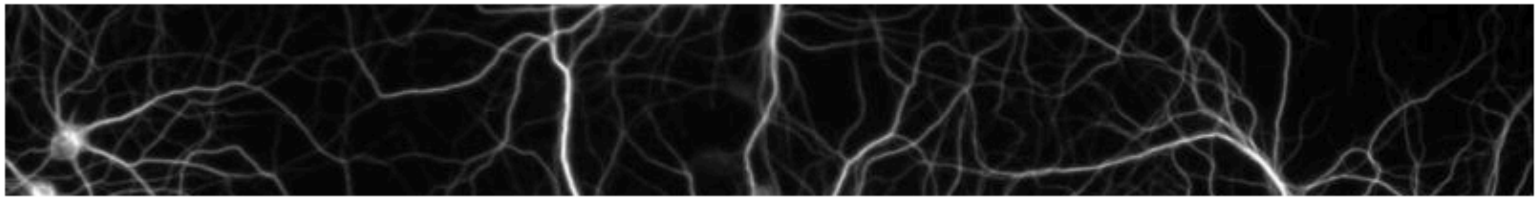
The company writes on its homepage, "Fret not, LookFlow fans. Keep an eye out for our product in future versions of Flickr — with many more wonderful photos and all that Flickr awesomeness!" It also says it will be helping Yahoo to form a new "deep learning group."

ADVERTISEMENT

Meet
Venmo Touch.

Learn more

Braintree



Google’s Large Scale Deep Learning Experiments

Google’s new large-scale learning experimentation using 16000 CPU cores and deep learning as part of google brain project had made a big success on Imagenet dataset. This success had a wide media coverage. Some pointers to the news:

Google official blog, 26 June 2012 http://googleblog.blogspot.com/2012/06/using-large-scale-brain-simulations-for.html

NYT Front page on large scale neural network John Markoff, [...]

Outline

- Motivating factors for study
- RBMs
- Deep Belief Networks
- Applications

The Toolbox

We often reach for the familiar…

For discriminative tasks we have

- o neural networks (~1980's, back-prop)
- o SVM (~1990's, Vapnik)

But is there anything better out there???

Challenges with SVM/NN

Potential difficulties with SVM

- Training time for large datasets
- Large number of support vectors for hard classification problems

Potential difficulties with NN & back-prop

- Diminishing gradient inhibits multiple layers
- Can get stuck in local minimums
- Training time can be extensive

Challenges with SVM/NN

More general "problems" with NNs and SVM…

- Need labeled data (what about unlabeled data?)
- Amount of information restricted by labels (ie, hard to learn a complex model if we are limited by labels)

What if I could use "8"s to learn to recognize "6"s ?

How to respond to these challenges

- Try to model the structure of the sensory input (ie, data), but keep the efficiency and simplicity of a gradient method
  - Adjust the weights to maximize the probability that a generative model would have produced the sensory input.
  - Learn p(data) not p(label | data)
- So instead of learning a label, first learn how to generative your data

Hinton, 2007

How to respond to these challenges

- Try to model the structure of the sensory input (ie, data), but keep the efficiency and simplicity of a gradient method
  - Adjust the weights to maximize the probability that a generative model would have produced the sensory input.
  - Learn p(data) not p(label | data)
- So instead of learning bel, first learn how to generative your data

Immediate benefit in that all data does not have to be label. Also reduces dependency on label.

Hinton, 2007

Recap

So, we are convinced we …

1. recognize some concerns with "standard" tools and would like what other options are out there
2. like the idea of modeling the input first (ie, building a model of our data as oppose to an out right classifier)

Energy Based Models

p(x) – probability of our data; data is represented by feature vector **x**.

$$p(x) = \frac{e^{-E(x)}}{Z}.$$

and

$$Z = \sum_x e^{-E(x)}$$

Attach an energy function (ie, *E*(x)) to score a configuration (ie, each possible input x).

We want desirable data to have low energy. Thus, tweak the parameters of *E*(x) accordingly.

EBMs with Hidden Units

To increase power of EBMs, add hidden variables.

$$P(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x,h)}}{Z}.$$

By using the notation,

Free energy

$$\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)}$$

We can rewrite p(x) in a form similar to the standard EBM,

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \text{ with } Z = \sum_x e^{-\mathcal{F}(x)}.$$

EBMs with Hidden Units

To increase power of EBMs, add hidden variables.

$$P(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x,h)}}{Z}.$$

By using the notation,

Free energy

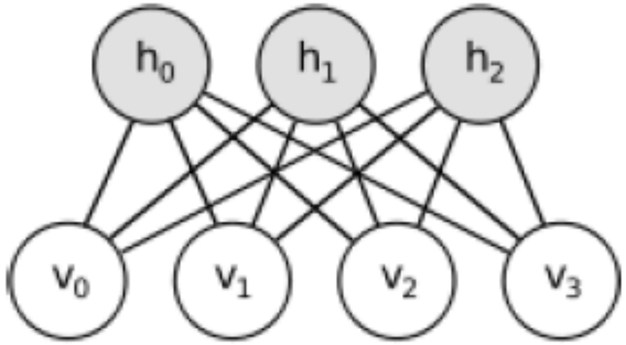
$$\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)}$$

We can rewrite p(x) in a form similar to the standard EBM,

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \text{ with } Z = \sum_x e^{-\mathcal{F}(x)}.$$

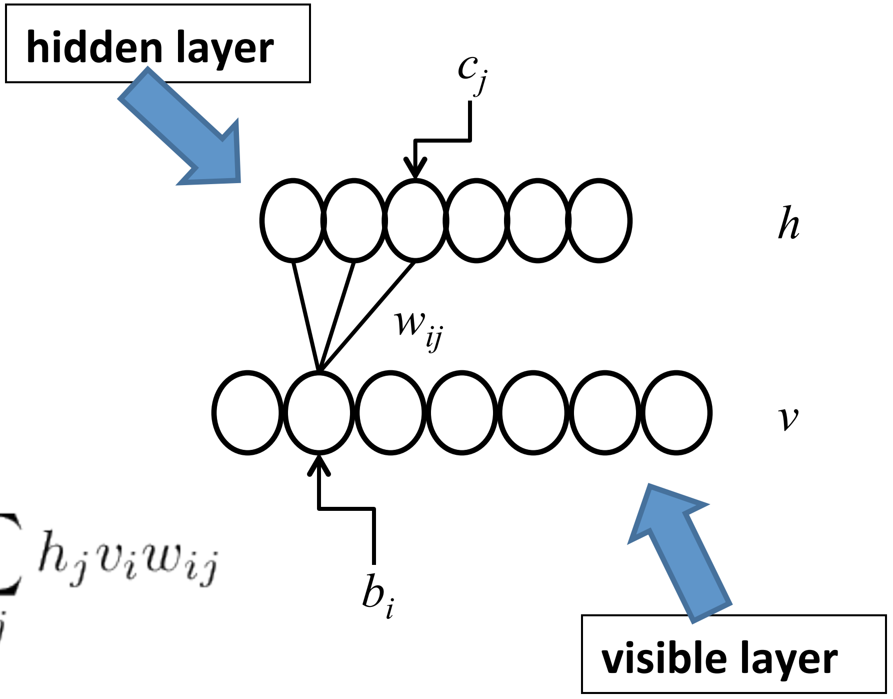
RBMs

- Represented by a bipartite graph, with symmetric, weighted connections
- One layer has visible nodes and the other hidden (ie, latent) variables.
- Notes are often binary , <u>stochastic</u> units (ie, assume 0 or 1 based on probablity)



Restricted Boltzmann Machine (RBM)

- A model for a distribution over binary vectors
- Probability of a vector, ***v***, under the model is defined via an "energy"



$$E(v,h) = -\sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} h_j v_i w_{ij}$$

$$Z = \sum_v \sum_h e^{-E(v,h)}$$

$$p(v) = \sum_h \frac{e^{-E(v,h)}}{Z}$$

Training a RBM – Maximum Likelihood Approach

$$l(\theta) = \frac{1}{n}\sum_i \log\left(\sum_h e^{-E(v_i,h)}\right) - \log Z$$

$$\frac{\partial l(\theta)}{\partial \theta_j} = \frac{1}{n}\sum_i \frac{\sum_h e^{-E(v_i,h)}}{\sum_h e^{-E(v_i,h)}}\frac{-\partial E}{\partial \theta_j} - \frac{1}{Z}\sum_v \sum_h e^{-E(v,h)}\frac{-\partial E}{\partial \theta_j}$$

$$= \frac{1}{n}\sum_i \sum_h \frac{p(v_i,h)}{p(v_i)}\left(\frac{-\partial E}{\partial \theta_j}\right) - E\left[\frac{-\partial E}{\partial \theta_j}\right]_{p^\infty}$$

$$= \frac{1}{n}\sum_i \sum_h p(h|v_i)\left(\frac{-\partial E}{\partial \theta_j}\right) - E\left[\frac{-\partial E}{\partial \theta_j}\right]_{p^\infty}$$

$$= E\left[\frac{-\partial E}{\partial \theta_j}\right]_{p^0} - E\left[\frac{-\partial E}{\partial \theta_j}\right]_{p^\infty}$$

Training a RBM - Contrastive Divergence (CD)

Instead of attempting to sample from joint distribution p(v,h) (i.e. p∞), sample from p^1(v,h).

$$\Delta\theta_j \propto E\left[\frac{-\partial E}{\partial\theta_j}\right]_{p^0} - E\left[\frac{-\partial E}{\partial\theta_j}\right]_{p^\infty}$$

$$\Delta\theta_j \propto E\left[\frac{-\partial E}{\partial\theta_j}\right]_{p^0} - E\left[\frac{-\partial E}{\partial\theta_j}\right]_{p^1}$$

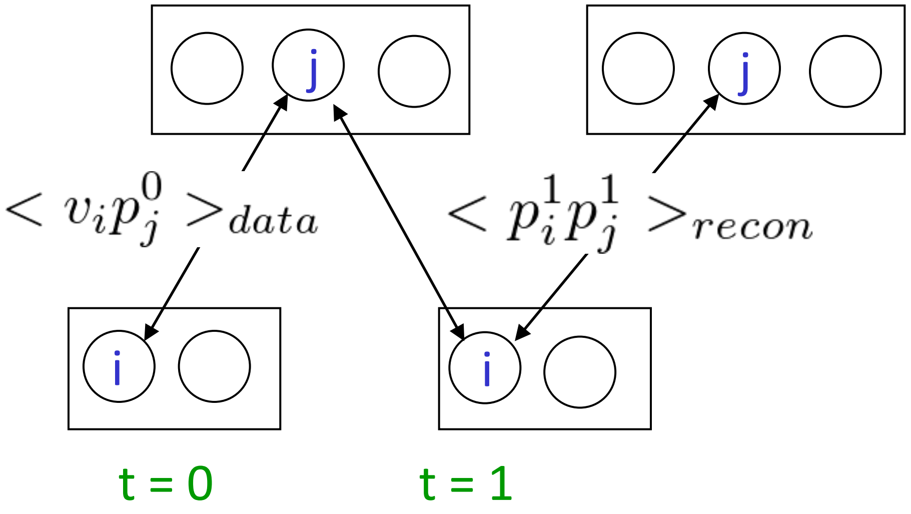
Faster and lower variance in sample.

Hinton, *Neural Computation*(2002)

Training a RBM

Partials of E(v, h) easy to calculate.

$$\frac{\partial E}{\partial w_{ij}} = v_i h_j$$



$$p_j^{(0)} = \sigma(\sum_i v_i w_{ij} + c_j)$$

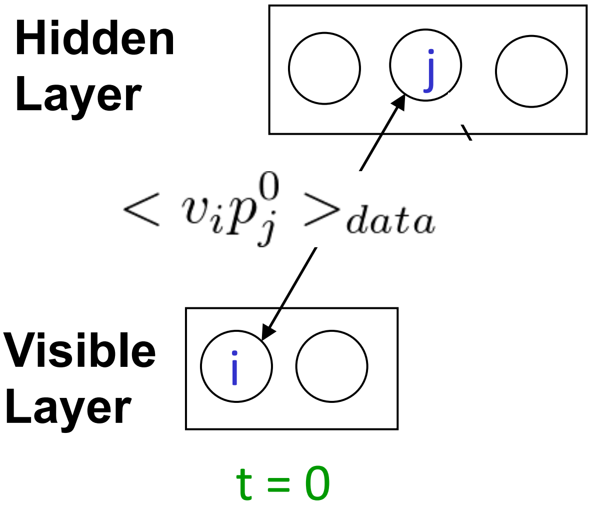
$$p_i^{(1)} = \sigma(\sum_j h_j w_{ij} + b_i)$$

$$p_j^{(1)} = \sigma(\sum_i p_i^1 w_{ij} + c_j)$$

Hinton, *Neural Computation*(2002)

Training a RBM via Contrastive Divergence

Gradient of the likelihood with respect to w_{ij} ≈ the difference between interaction of v_i and h_j at time 0 and at time 1.

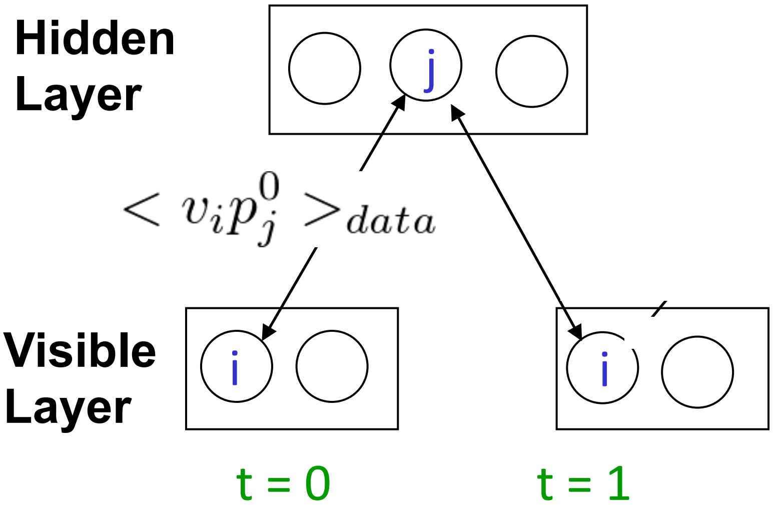


$$p_j^{(0)} = \sigma(\sum_i v_i w_{ij} + c_j)$$

Hinton, *Neural Computation*(2002)

Training a RBM via Contrastive Divergence

Gradient of the likelihood with respect to w_{ij} ≈ the difference between interaction of v_i and h_j at time 0 and at time 1.



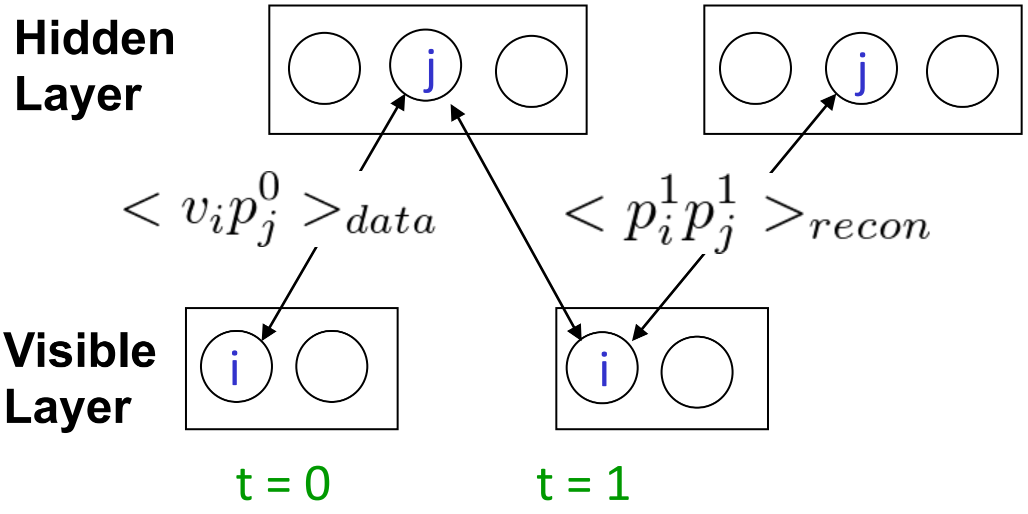
$$p_j^{(0)} = \sigma(\sum_i v_i w_{ij} + c_j)$$

$$p_i^{(1)} = \sigma(\sum_j h_j w_{ij} + b_i)$$

Hinton, *Neural Computation*(2002)

Training a RBM via Contrastive Divergence

Gradient of the likelihood with respect to w_{ij} ≈ the difference between interaction of v_i and h_j at time 0 and at time 1.



$$p_j^{(0)} = \sigma\left(\sum_i v_i w_{ij} + c_j\right)$$

$$p_i^{(1)} = \sigma\left(\sum_j h_j w_{ij} + b_i\right)$$

$$p_j^{(1)} = \sigma\left(\sum_i p_i^1 w_{ij} + c_j\right)$$

$$\Delta w_{i,j} = <v_i p_j^0> - <p_i^1 p_j^1>$$

Hinton, *Neural Computation*(2002)

Weight/Bias Updates

$$\Delta^{(n)} w_{ij} = \epsilon\{(< v_i p_j^{(0)} > - < p_i^{(1)} p_j^{(1)} >) - \eta w_{ij}\} \\ + \nu w_{ij}^{(n-1)}$$

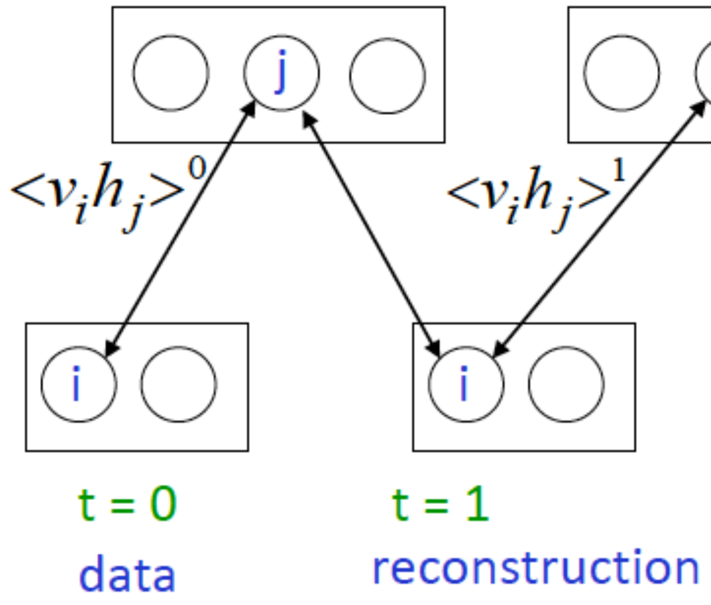
$$\Delta^{(n)} b_i = \epsilon\{(< v_i > - < p_i^{(1)} >)\} - \nu b_i^{(n-1)}$$

$$\Delta^{(n)} c_j = \epsilon\{(< p_j^{(0)} > - < p_j^{(1)}) >\} - \nu c_i^{(n-1)}$$

ε is the learning rate, η is the weight cost, and *υ* the momentum.

$\downarrow$ **Gradient** $\downarrow$ **Smaller Weights** $\downarrow$ **Avoid Local Minima**

A quick way to learn an 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”.

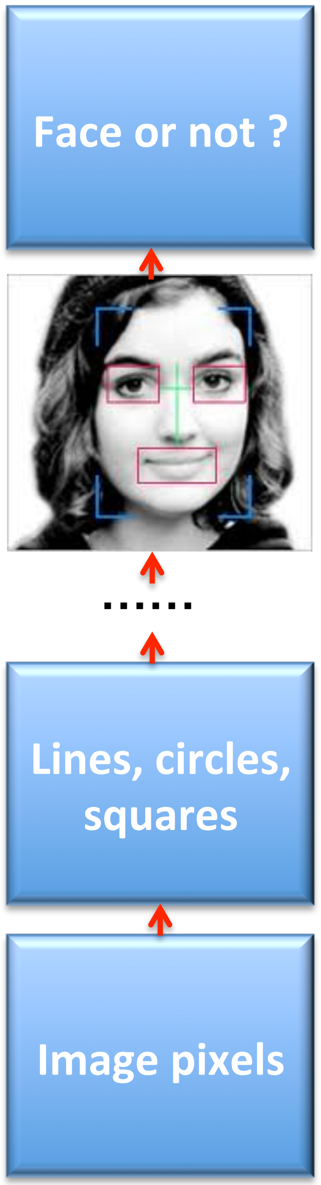
Update the hidden units again.

$$\Delta w_{ij} = \varepsilon \left(<v_i h_j>^0 - <v_i h_j>^1 \right)$$

This is not following the gradient of the log likelihood. But it works well. It is approximately following the gradient of another objective function (Carreira-Perpinan & Hinton, 2005).

Slide modified from Hinton, 2007

Why Deep Learning? – A Face Recognition Analogy



Brain Learning

Training a RBM – A Maximum Likelihood Approach

Objective of Unsupervised Learning:

Find $w_{i,j}$ to maximize the likelihood $p(v)$ of visible data

Iterative Gradient Descent Approach:

Adjust $w_{i,j}$ to increase the likelihood according to gradient

Why ???

Okay, we can model p(x).

But how to…

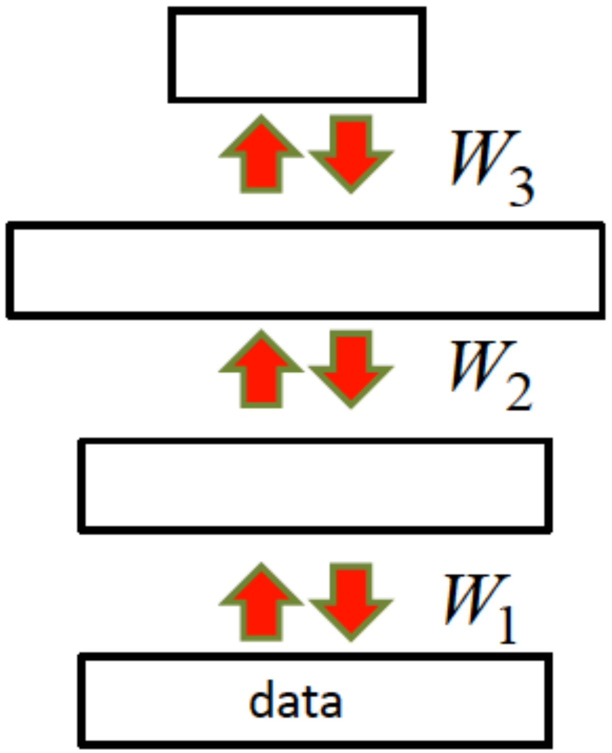
1. **Find p(label|x). We want a classifier!**
2. Improve the model for p(x).

Deep Belief Nets

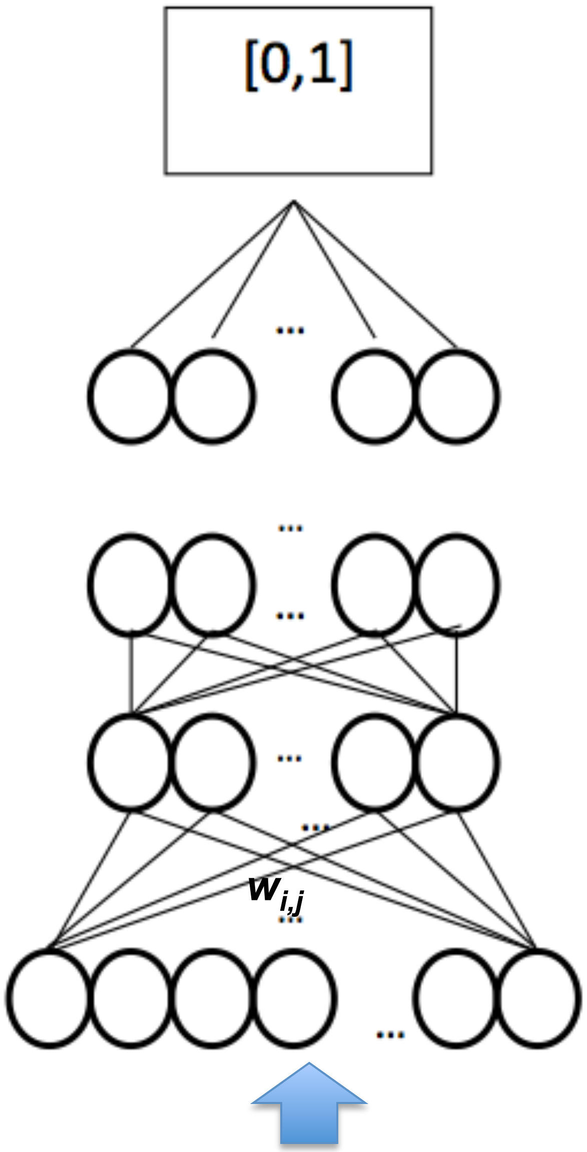
RBMs are typically used in stack

- Train them up one layer at a time
- Hidden units become visible units to the next layer up

If your goal is a discriminator, you train a classifier on the top level representation of your input.



Deep Learning Network Architecture



~350 nodes

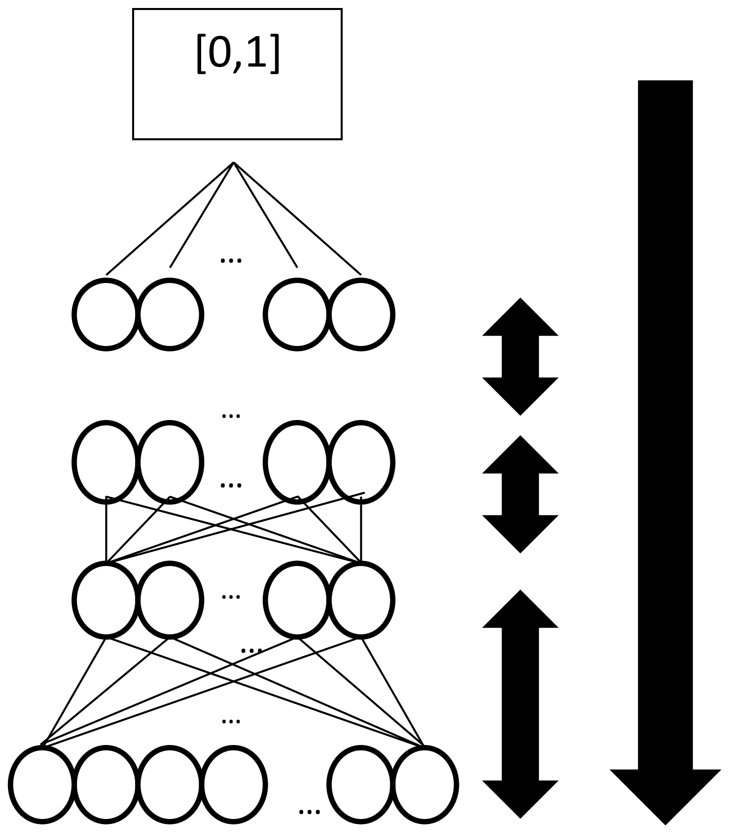
~500 nodes

~500 nodes

~400 input nodes

A Vector of ~400 Features (numbers between 0 and 1)

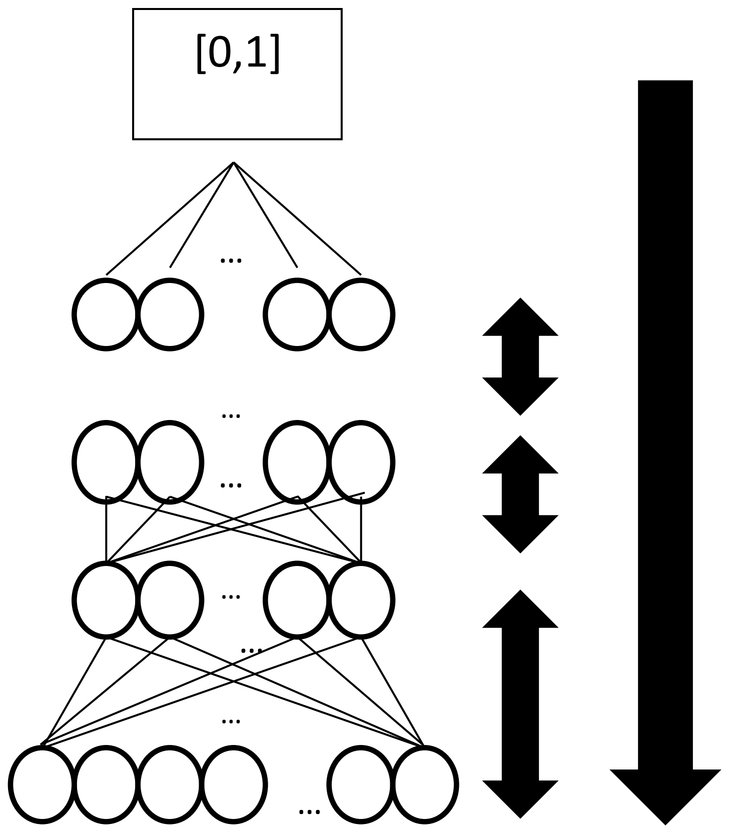
Training a Deep Network



1. Weights are learned layer by layer via <u>unsupervised learning</u>.
2. Final layer is learned as a <u>supervised neural network</u>.
3. All weights are fine-tuned using <u>supervised back propagation</u>.

Hinton and Salakhutdinov, *Science,* 2006

Training a Deep Network



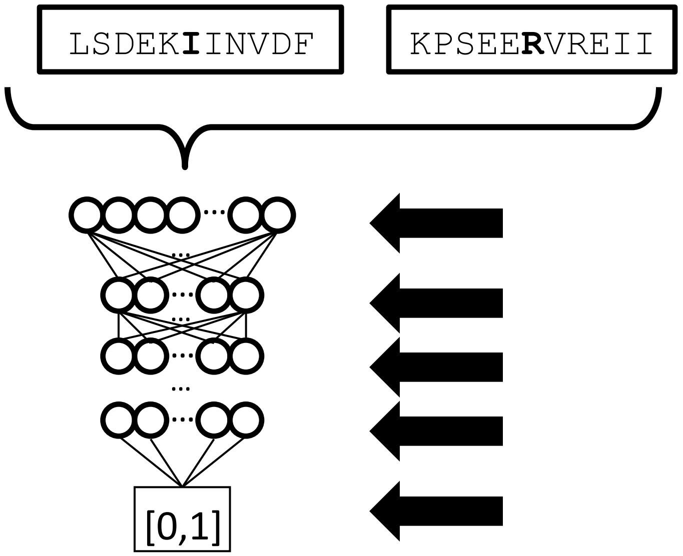
1. Weights are learned layer by layer via <u>unsupervised learning</u>.
2. Final layer is learned as a <u>supervised neural network</u>.
3. All weights are fine-tuned using <u>supervised back propagation</u>.

Hinton and Salakhutdinov, *Science,* 2006

Specific Implementation on GPU

Speed up training by CUDAMat and GPUs

Train DNs with over 1M parameters in about an hour

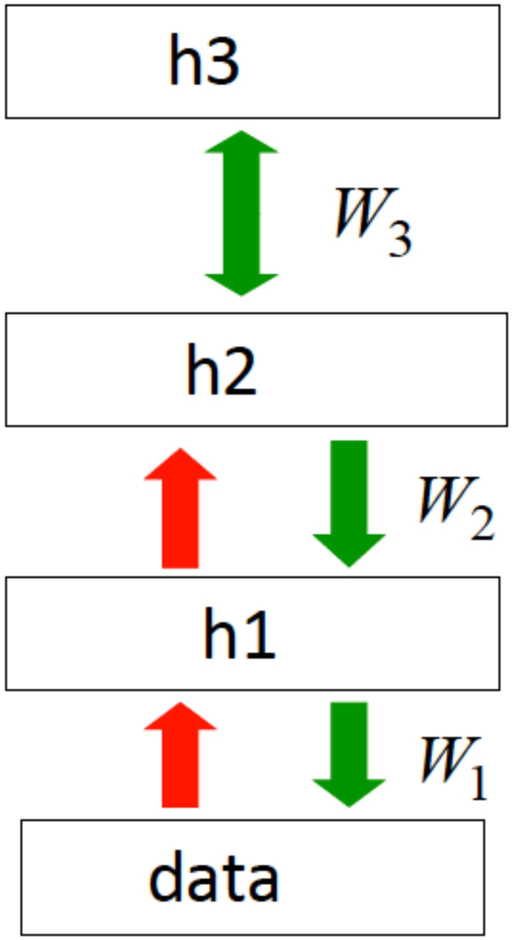


How to generate from the model

Bonus when modeling p(x), we can see what the model believes in

- To generate data:
  - Get an equilibrium sample from the top-level RBM by performing alternating Gibbs sampling for a long time.
  - Perform a top-down pass to get states for all the other layers.

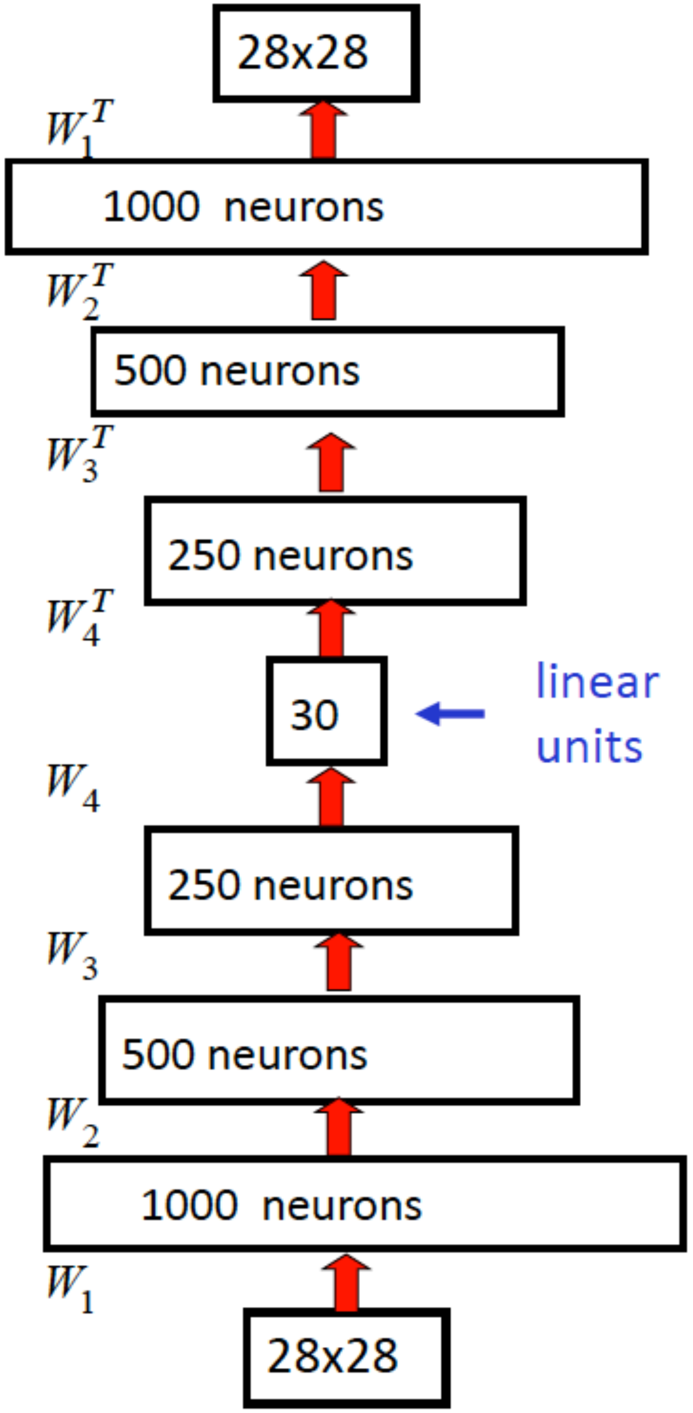
So the lower level bottom-up connections are not part of the generative model. They are just used for inference.



Slide modified from Hinton, 2007

Deep Autoencoders

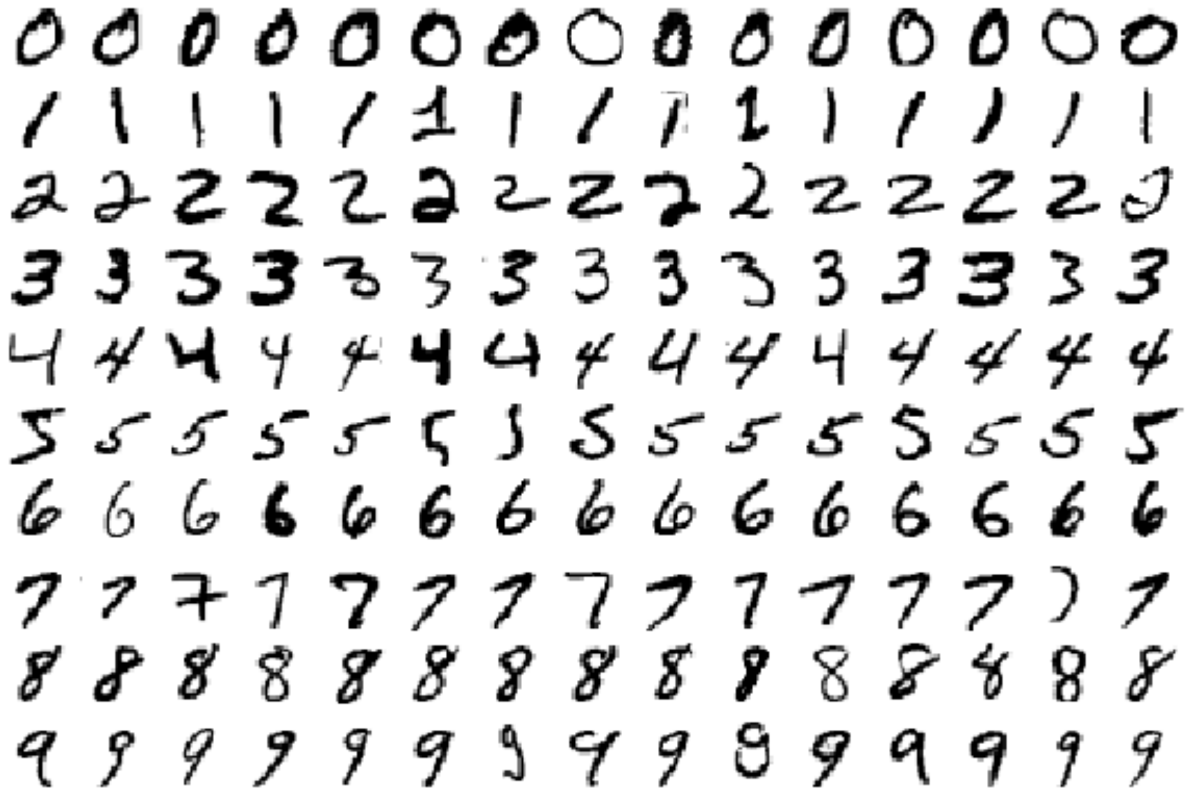
- They always looked like a really nice way to do non-linear dimensionality reduction:
  - But it is very difficult to optimize deep autoencoders using backpropagation.
- We now have a much better way to optimize them:
  - First train a stack of 4 RBM's
  - Then "unroll" them.
  - Then fine-tune with backprop.



Hinton & Salakhutdinov, 2006; slide form Hinton UCL tutorial

Applications: A model of digit recognition

- Classify digits (0 – 9)
- Input is a 28x28 image from MNIST (training 60k, test 10k examples)

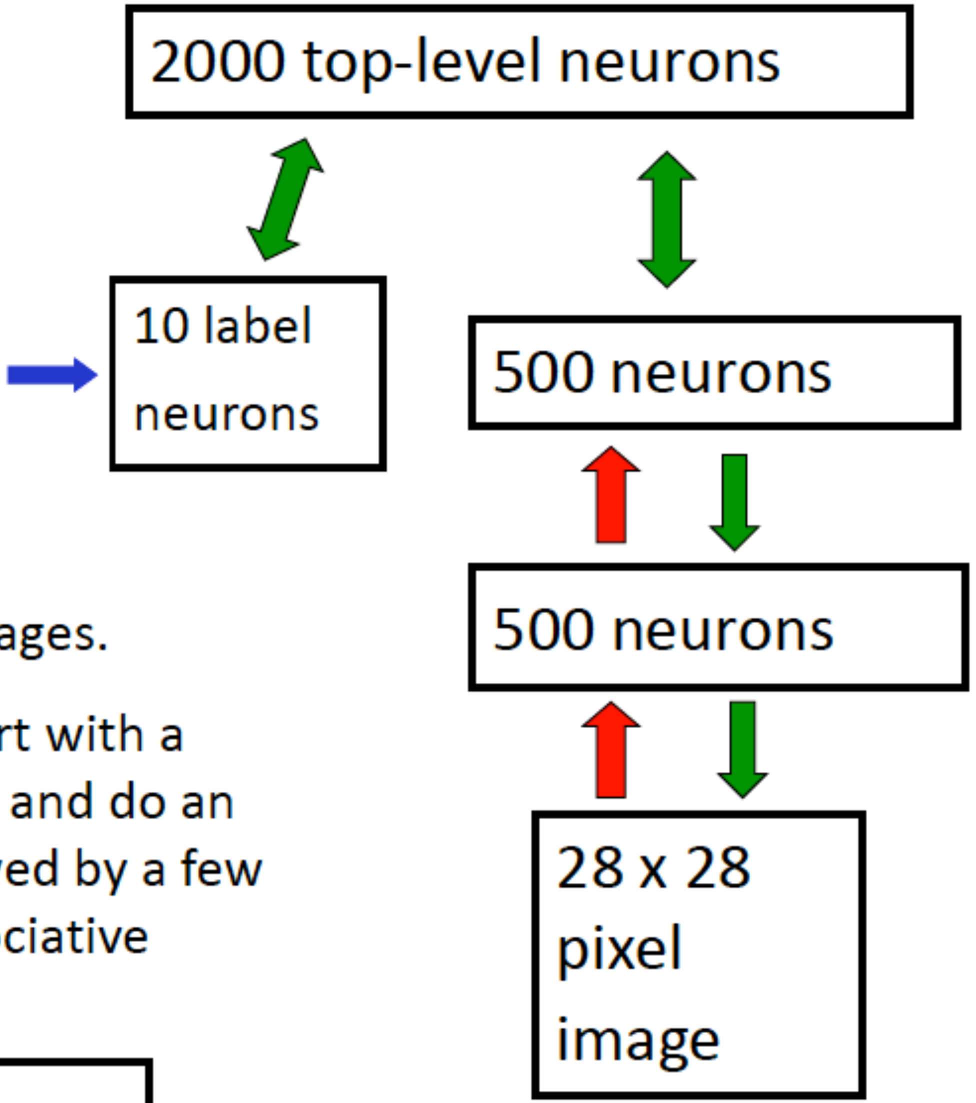


Applications: A model of digit recognition

This is work from Hinton et al., 2006

The top two layers form an associative memory whose energy landscape models the low dimensional manifolds of the digits.

The energy valleys have names



The model learns to generate combinations of labels and images.

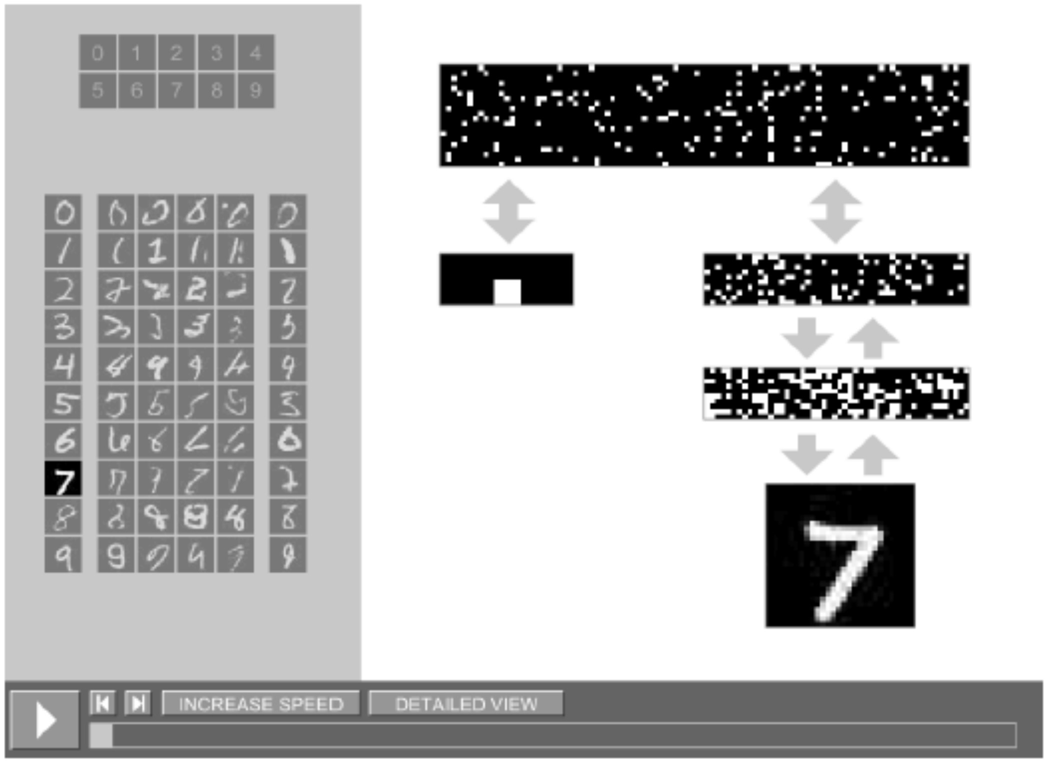
To perform recognition we start with a neutral state of the label units and do an up-pass from the image followed by a few iterations of the top-level associative memory.

Matlab/Octave code available at http://www.cs.utoronto.ca/~hinton/

Slide modified from Hinton, 2007

Model in action

Hinton has provided an excellent way to view the model in action…

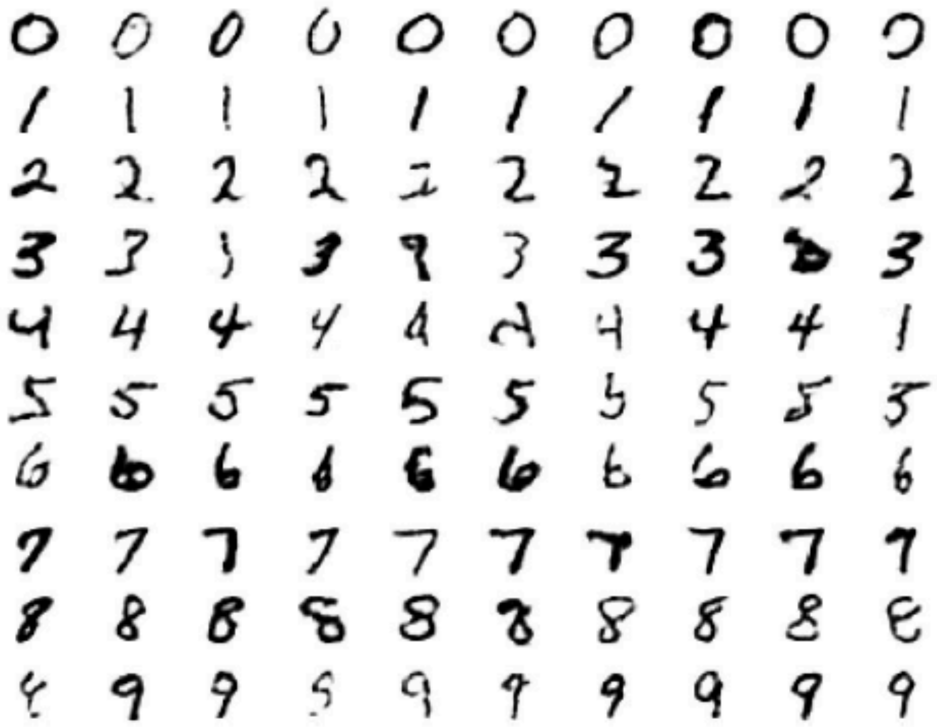


Demo:

http://www.cs.toronto.edu/~hinton/digits.html

More Digits

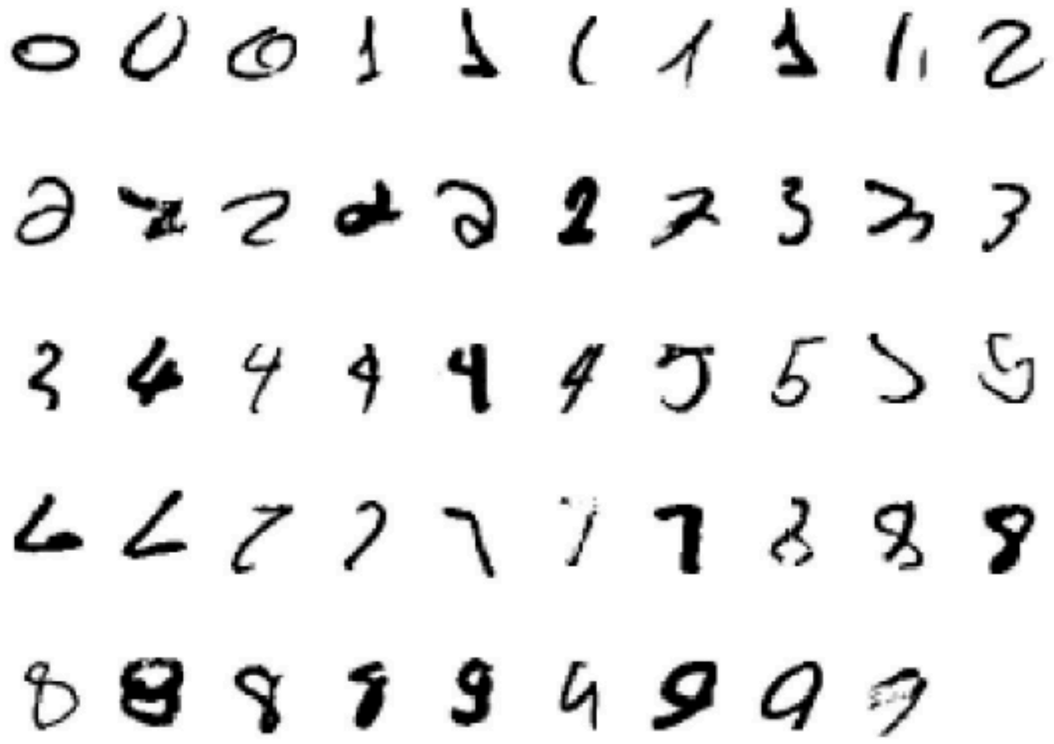
Samples generated by letting the associative memory run with one label clamped. There are 1000 iterations of alternating Gibbs sampling between samples.



Slide from Hinton, 2007

Even More Digits

Examples of correctly recognized handwritten digits that the neural network had never seen before



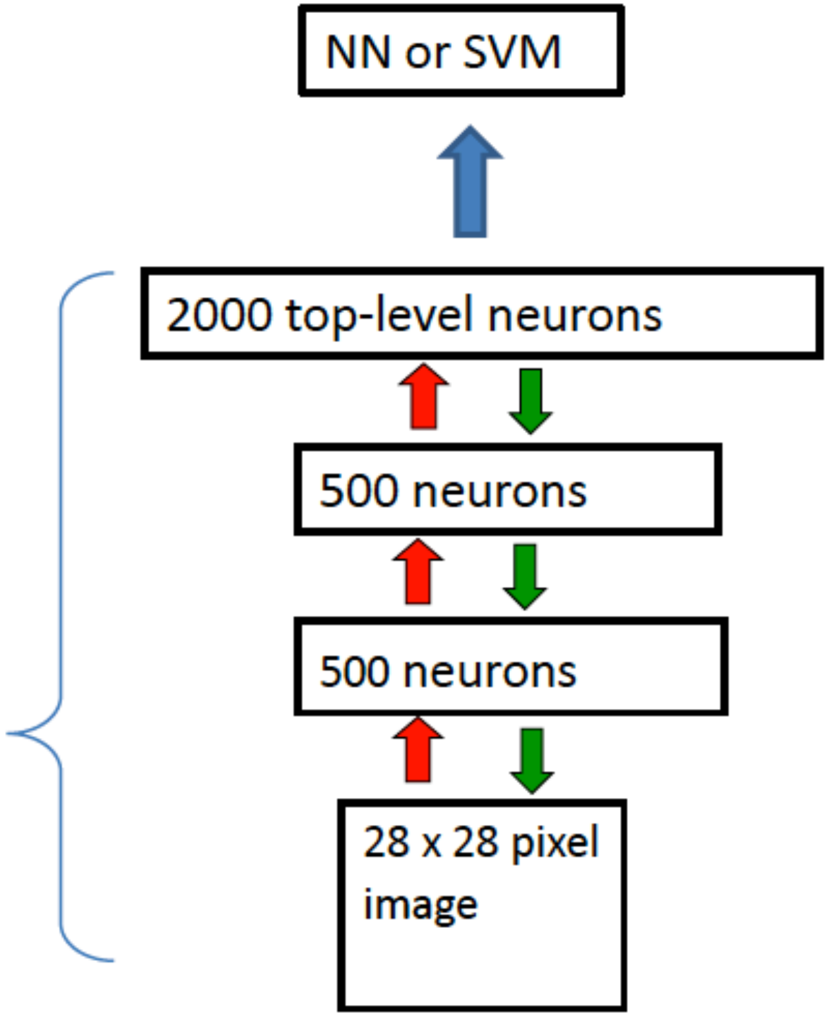
Slide from Hinton, 2007

Extensions

Do classification.

One way (probably no the best), train generative model with labeled/unlabeled data

Then train a NN on higher dimensional representation.



Acknowledgements

- Greg Hinton’s slides
- Jesse Eickholt’s slides