RBMs, DBN, AutoEncoders

- Restricted Boltzmann Machines (RBMs)
  - energy model of probability distribution
  - training: Contrastive Divergence
  - why does it work?

- Deep Belief Networks
- Auto Encoders
- Denoising Auto Encoders
- Stacked Auto Encoders

References

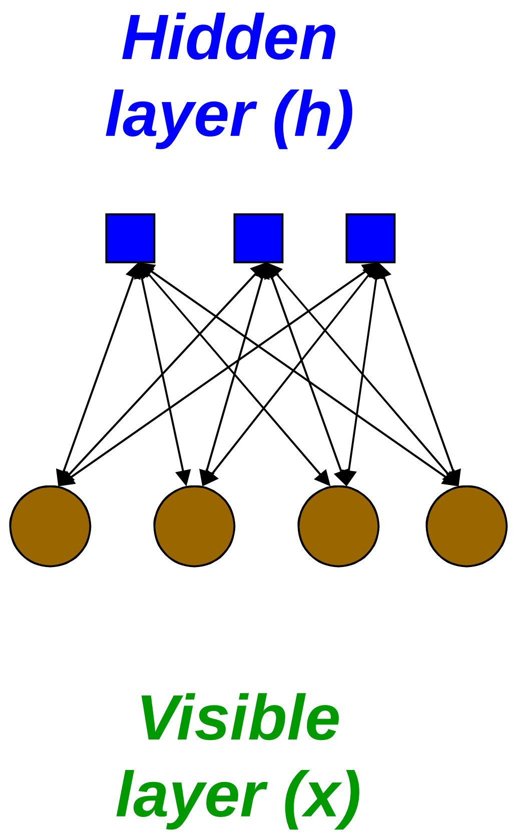
- R.B. Palm, Prediction as a candidate for learning deep hierarchical models of data Deep Belief Networks (MSc. Thesis, 2012)
  github.com/rasmusbergpalm/DeepLearnToolbox

  Sections 1.2 (Methods) and 1.3 (Results) required for exam! (pages 9-27)

- Hinton's tutorial on using RMBs
  www.cs.toronto.edu/~hinton/absps/guideTR.pdf

- Kevin Duh course on Deep Learning (optional)
  http://cl.naist.jp/~kevinduh/a/deep2014/

Restricted Boltzmann Machine



- 2-layered network; **unsupervised!**

- $W=(w_{ij})$: matrix of weights
  $b = (b_i)$: biases of visible layer
  $c = (c_j)$: biases of hidden layer

- nodes take values 0 or 1
- **model of probability distr. of $P(x, h)$**
- non-deterministic behavior:

$$P(h_i = 1|x) = \mathrm{sigm}(c_i + W_i x)$$
$$P(x_i = 1|h) = \mathrm{sigm}(b_i + W_i h)$$

RBM as a model of P(x, h)

P(x, y) is defined as:

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

where energy E and normalization Z are given by:

$$E(x,h) = -b'x - c'h - h'Wx$$

$$Z(x,h) = \sum_{x,h} e^{-E(x,h)}$$

Conditional Probabilities

P(h|x) = ???

P(x, h)=P(h|x)P(x), so P(h|x)=P(x, h)/P(x)

Moreover, P(x)=sum$_h$(P(x, h)), so:

$$P(h|x) = \frac{\exp(b'x + c'h + h'Wx)}{\sum_h \exp(b'x + c'h + h'Wx)}$$

$$P(h|x) = \frac{\prod_i \exp(c_i h_i + h_i W_i x)}{\prod_i \sum_h \exp(c_i h_i + h_i W_i x)}$$

$$P(h|x) = \prod_i \frac{\exp(h_i(c_i + W_i x))}{\sum_h \exp(h_i(c_i + W_i x))}$$

$$P(h|x) = \prod_i P(h_i|x)$$

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

Conditional Probabilities (cont.)

Thus:

$$P(h_i = 1|x) = \frac{\exp(c_i + W_i x)}{1 + \exp(c_i + W_i x)}$$

$$P(h_i = 1|x) = \mathrm{sigm}(c_i + W_i x)$$

and, by symmetry:

$$P(x_i = 1|h) = \mathrm{sigm}(b_i + W_i h)$$

(Note, that

1/(1+exp(-t)) = exp(t)/(1+exp(t))

so the “old” definition is equivalent to the new one)

Training of RBM

- Purpose of training: for a given training set X find weights that maximize the log likelihood of X:

$$L(X)=\text{sum}_{\{x \text{ in } X\}}\log(P(x))$$

- Use the gradient descend algorithm!
- Derivation of the gradient of L (see page 11) leads to a formula that involves two terms:

- expected value of $x_i h_j$ for known x
- expected value of $x_i h_j$ over all vectors x (infeasible!)

- Hinton's trick: approximate the second term by iterating a random process for a few times (1 or 2, perhaps 3 times) => Contrastive Divergence

Derivation of gradients of L (page 11)

$$\frac{\partial}{\partial\theta}(-\log P(x)) = \frac{\partial}{\partial\theta}\left(-\log\sum_h P(x,h)\right)$$

by definition

$$= \frac{\partial}{\partial\theta}\left(-\log\sum_h \frac{\exp(-E(x,h))}{Z}\right)$$

by definition of P(x,h)

$$= -\frac{Z}{\sum_h \exp(-E(x,h))}\left(\sum_h \frac{1}{Z}\frac{\partial \exp(-E(x,h))}{\partial\theta} - \sum_h \frac{\exp(-E(x,h))}{Z^2}\frac{\partial Z}{\partial\theta}\right)$$

log(x)'=1/x); chain rule,
(f*g)'=f'*g+f*g'

$$= \sum_h \left(\frac{\exp(-E(x,h))}{\sum_{\hat{h}} \exp(-E(x,\hat{h}))}\frac{\partial E(x,h)}{\partial\theta}\right) + \frac{1}{Z}\frac{\partial Z}{\partial\theta}$$

$$= \sum_h P(h|x)\frac{\partial E(x,h)}{\partial\theta} - \frac{1}{Z}\sum_{x,h}\exp(-E(x,h))\frac{\partial E(x,h)}{\partial\theta}$$

$$= \sum_h P(h|x)\frac{\partial E(x,h)}{\partial\theta} - \sum_{x,h}P(x,h)\frac{\partial E(x,h)}{\partial\theta}$$

$$= \mu_1\left[\frac{\partial E(x,h)}{\partial\theta}\middle|\, x\right] - \mu_1\left[\frac{\partial E(x,h)}{\partial\theta}\right]$$

$$\frac{\partial}{\partial W}(-\log P(x)) = \mu_1\left[-h'x|\,x\right] - \mu_1\left[-h'x\right]$$

$$\frac{\partial}{\partial b}(-\log P(x)) = \mu_1\left[-x|\,x\right] - \mu_1\left[-x\right]$$

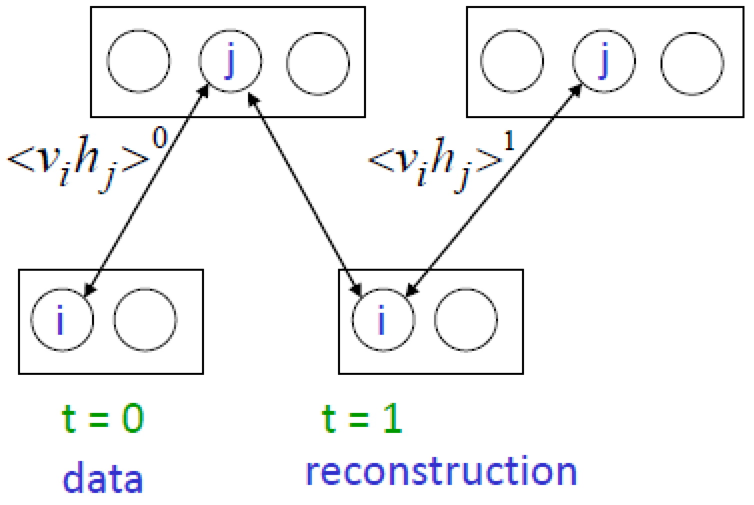
$$\frac{\partial}{\partial c}(-\log P(x)) = \mu_1\left[-h|\,x\right] - \mu_1\left[-h\right]$$

$$E(x,h) = -b'x - c'h - h'Wx$$

$$Z(x,h) = \sum_{x,h} e^{-E(x,h)}$$

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

Approximation of gradients



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.

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

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

Motivation: page 12

- At each iteration the entire layer is updated. To get unbiased samples, we should initialize the model at some arbitrary state, and sample n times, n being a large number.
- To make this efficient, we'll do something slightly different. We'll initialize the model at a training sample, iterate one step, and use this as our negative sample. This is the contrastive divergence algorithm as introduced by Hinton [HOT06] with one step (CD-1).
- The logic is that, as the model distribution approaches the training data distribution, initializing the model to a training sample approximates letting the model converge.

Training of RBM

Algorithm 1 Contrastive Divergence 1

```
for all training samples as $t$ do
    $x^{(0)} \leftarrow t$
    $h^{(0)} \leftarrow sigm(x^{(0)}W + c) > rand()$
    $x^{(1)} \leftarrow sigm(h^{(0)}W^T + b) > rand()$
    $h^{(1)} \leftarrow sigm(x^{(1)}W + c) > rand()$
    $W \leftarrow W + \alpha(x^{(0)}h^{(0)} - x^{(1)}h^{(1)})$
    $b \leftarrow b + \alpha(x^{(0)} - x^{(1)})$
    $c \leftarrow c + \alpha(h^{(0)} - h^{(1)})$
end for
```

More details

- To increase the accuracy (esp. in the final stage of the training process) iterate “up”-”down” updates several times
- In some cases, instead of using of $x_i h_j$ it is better to use of $x_i p(h_j|x)$
- Use mini-batches: process your data in small batches, updating weights after the whole mini-batch is processed:
  - “controlling randomness”: speeding up convergence
  - distributed implementations (GPU’s)
- more tips: A practical guide to training RBMs (Hinton): www.cs.toronto.edu/~hinton/absps/guideTR.pdf

Deep Belief Networks

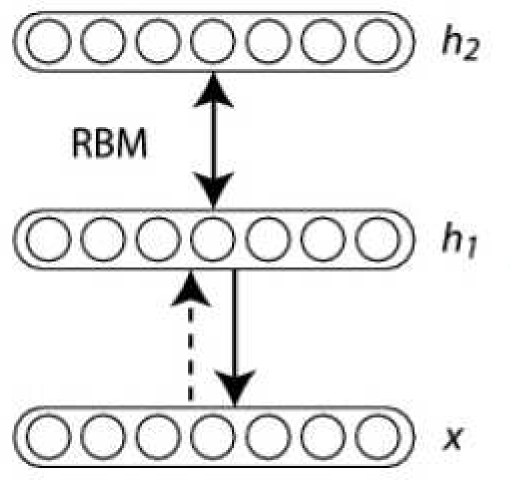
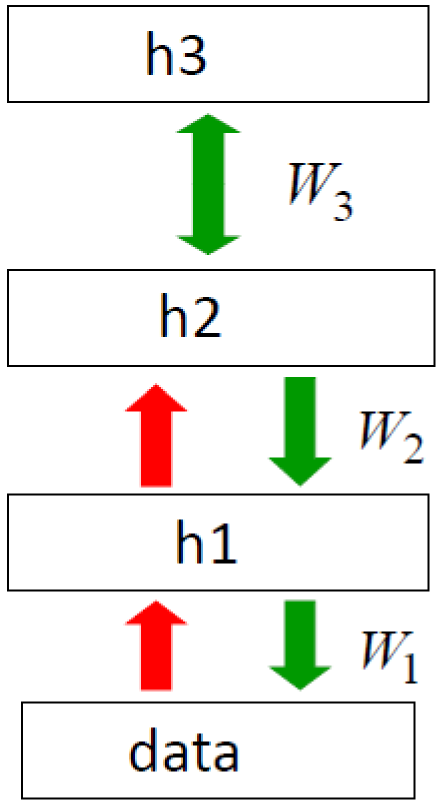
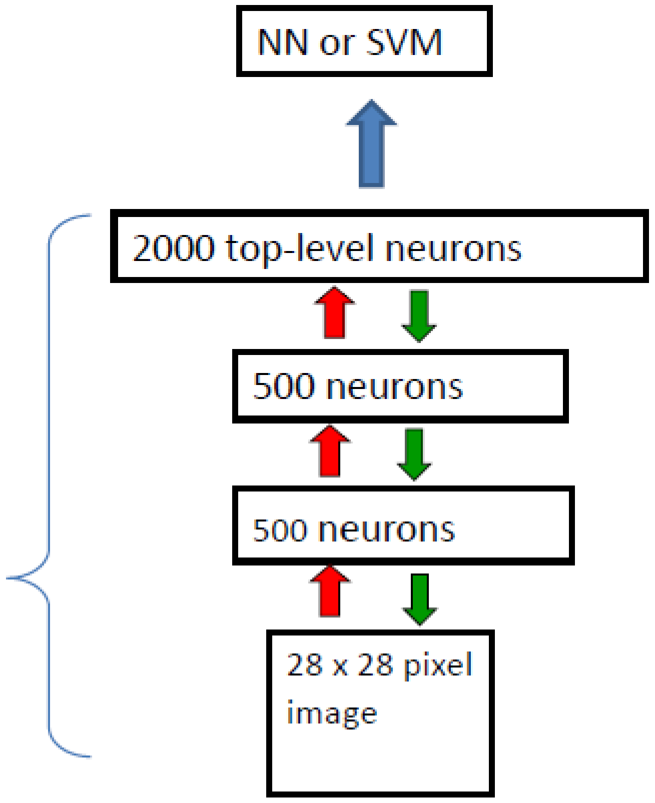


Figure 1.5: Deep Belief Network. Taken from



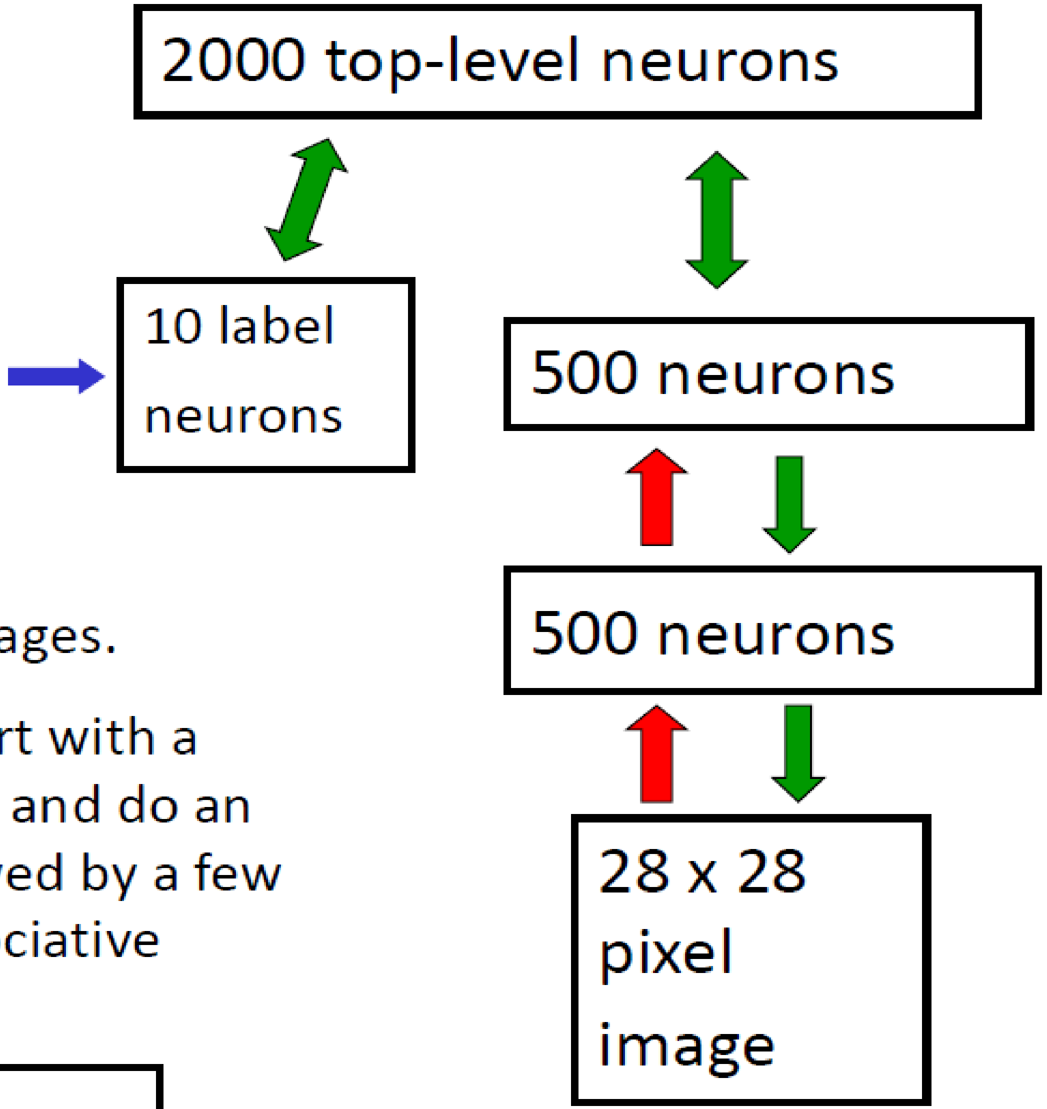


Applications: A model of digit recognition

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

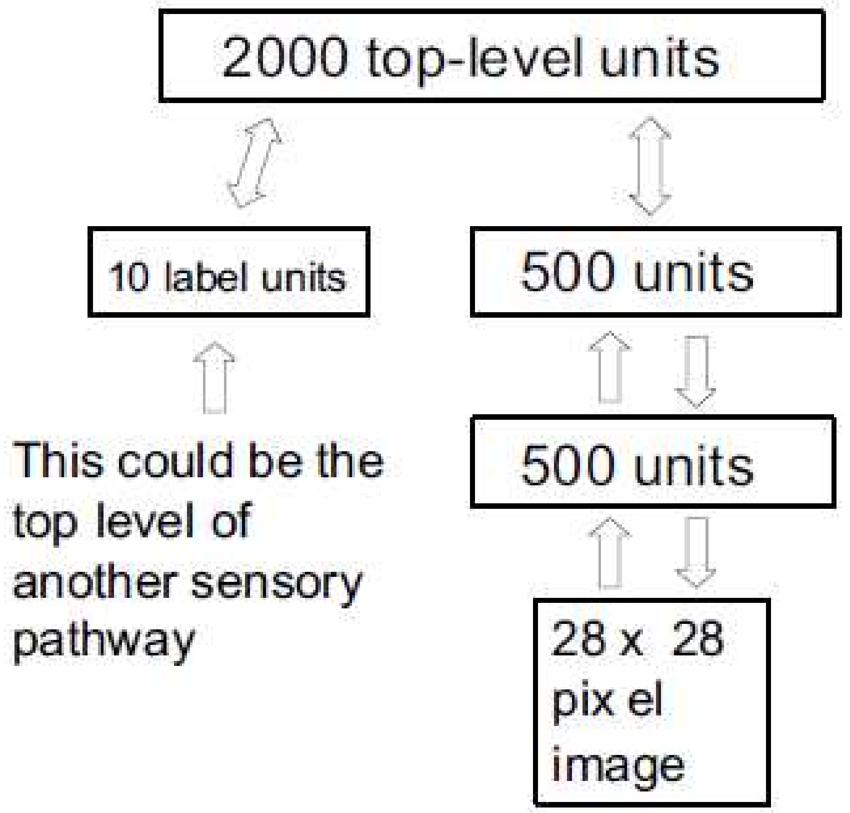
Hinton’s demo

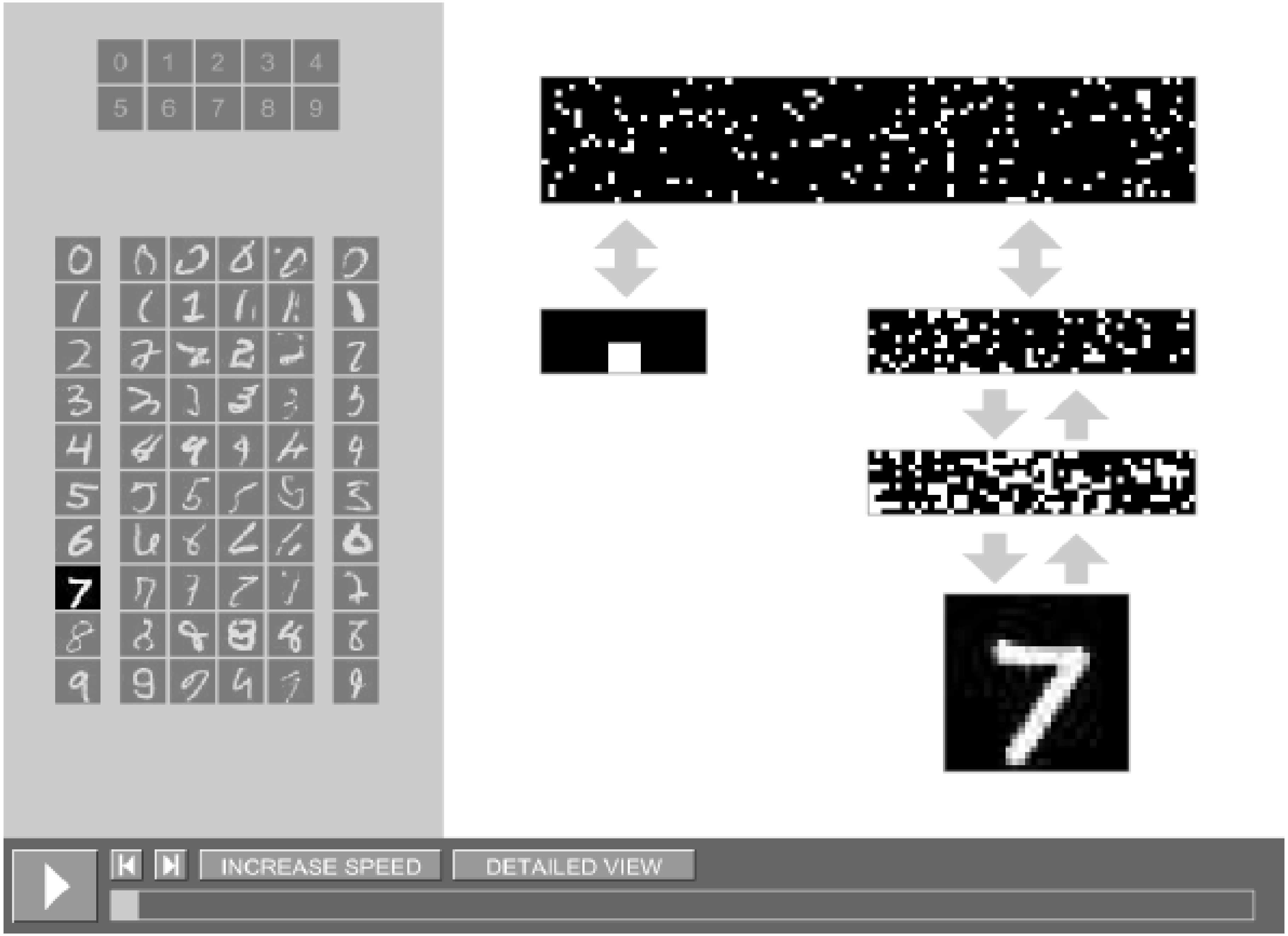
http://www.cs.toronto.edu/~hinton/adi/index.htm

http://www.cs.toronto.edu/~hinton/absps/fastnc.pdf

DNN as a classifier

DNN as a generator




0
1
2
3
4
5
6
7
8
9
INCREASE SPEED
DETAILED VIEW

More…

- Why DBN is better than MLP?
  - in MLP “delta’s” vanish (or explode) from layer to layer

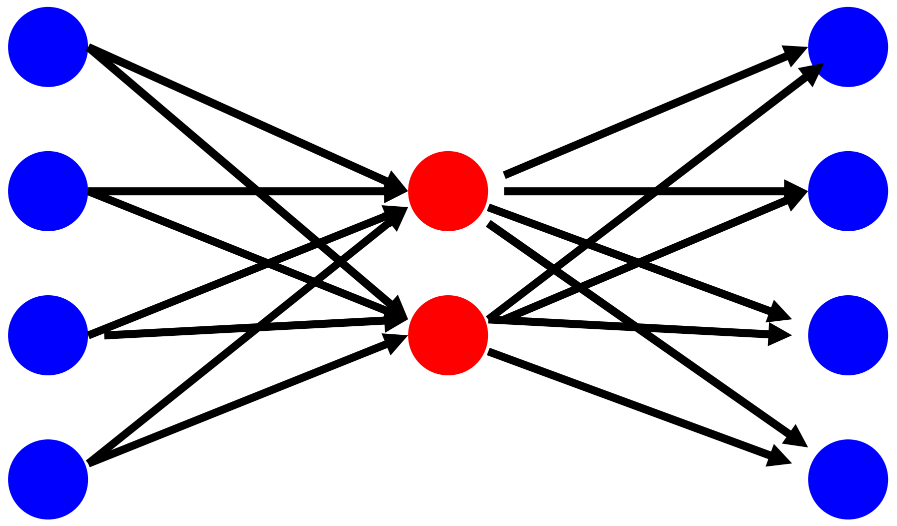
    => very few layers
  - RBMs “discover” intrinsic features of data

    => a hierarchy of features
  - RBMs “pre-train” MLP to avoid local minima!
  - RBM is trained on unlabeled data

    it’s easy to get unlabeled data; labeled data is difficult to get

Encoders, PCA, Compression

- Consider a 4:2:4 MLP (ENCODER):



- Inputs: (0 0 0 1), (0 0 1 0), (0 1 0 0), (1 0 0 0)
- Outputs: (0 0 0 1), (0 0 1 0), (0 1 0 0), (1 0 0 0)

Encoders

- Encoder network "learns" the concept of binary representation of integers!
- General case: $2^n:n:2^n$ networks
- Somehow the network is able to "extract" the most concise representation of the input!
- The same trick works for "real numbers": PCA networks
- It can be used for dimensionality reduction, data compression, and visualization.

Training Encoders: variants

- Backpropagation (good for shallow architectures)

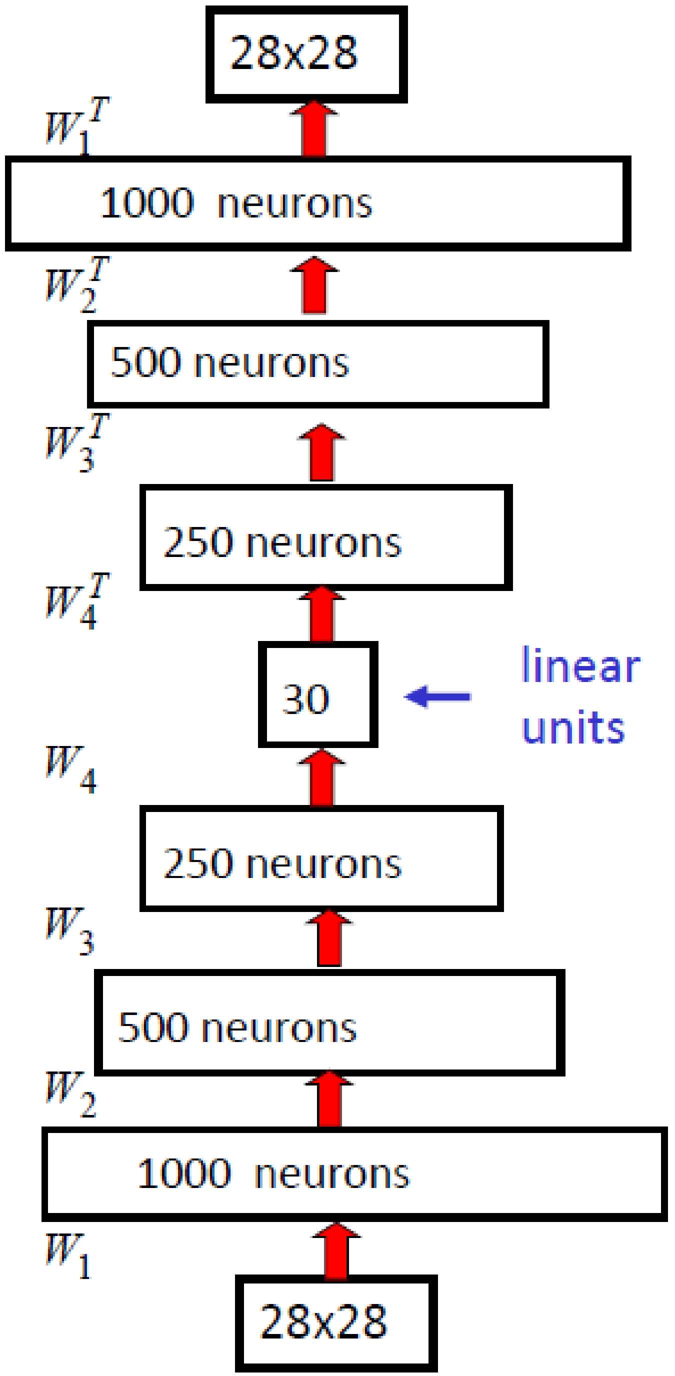
- Enforced "symmetry of weights"
(upper part = mirror of lower part)

- **Denoising Autoencoders**: the input layers gets a "noisy version of true x"; the target contains the original x ("noise": e.g., 10% of randomly selected pixels set to 0)

- **Deep Autoencoders**: the first "half" is trained unsupervised as a DBN (a stack of RBMs); the upper part is the "mirror" of the lower part; final phase trained by backpropagation

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.

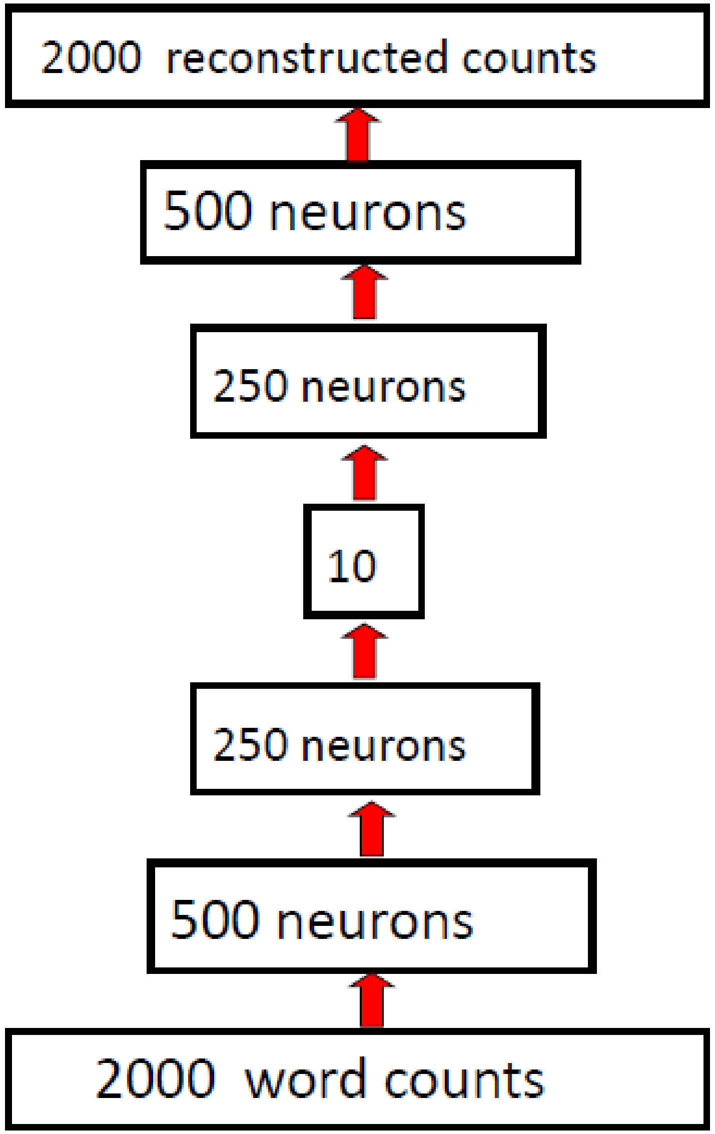


Hinton & Salakhutdinov, 2006; slide form Hinton UCL tutorial

Applications: Classifying text documents

- A document can be characterized by the frequency of words that appear (ie, word counts for some dictionary become feature vector)
- Goals…
  1. Group/cluster similar documents
  2. Find similar documents

How to compress the count vector

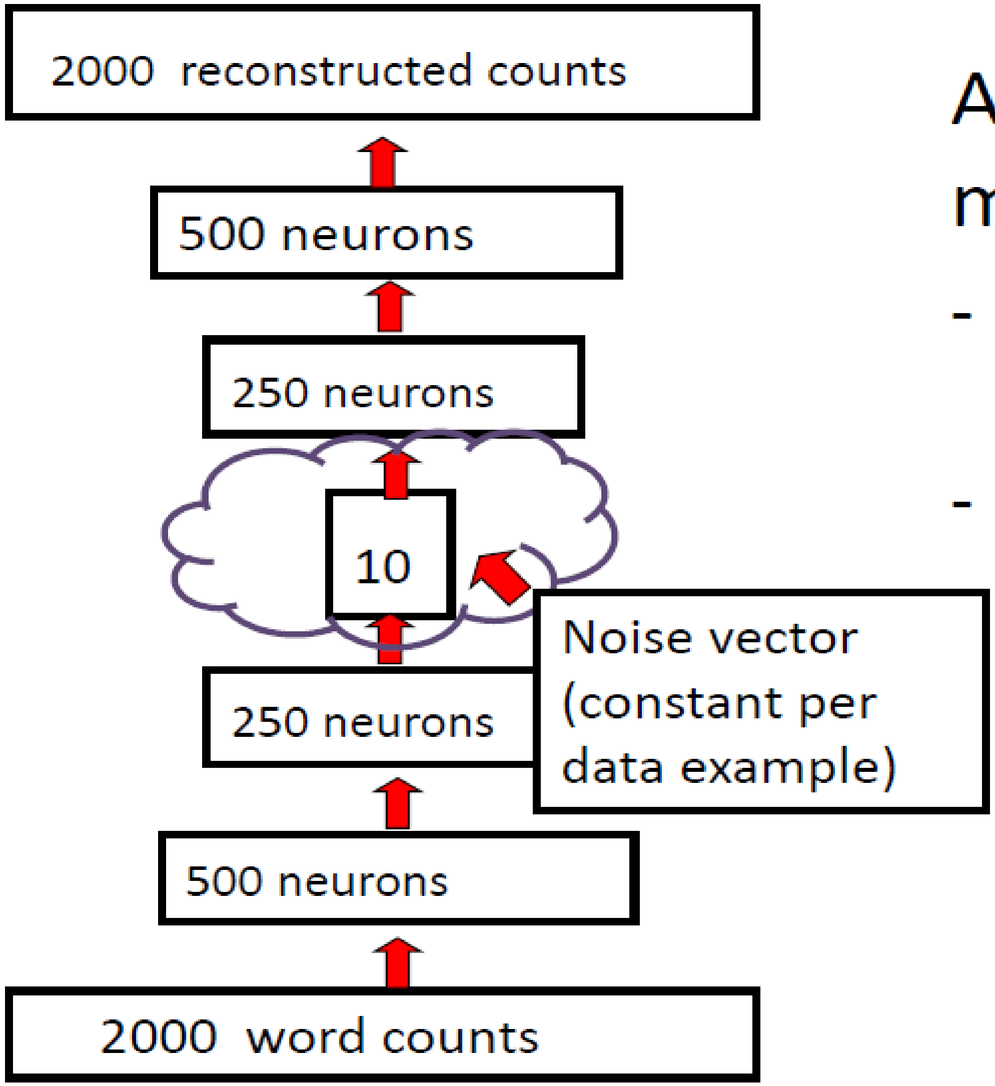


<u>Multi-layer auto-encoder</u>

- Train a model to reproduce its input vector as its output
- This setup forces as much information as possible be compressed and passed thru the 10 numbers in the central bottleneck.
- These 10 numbers are then a good way to compare documents.

Slide modified from Hinton, 2007

Search


2000 reconstructed counts
500 neurons
250 neurons
10
Noise vector (constant per data example)
250 neurons
500 neurons
2000 word counts

Add noise to input to middle layer

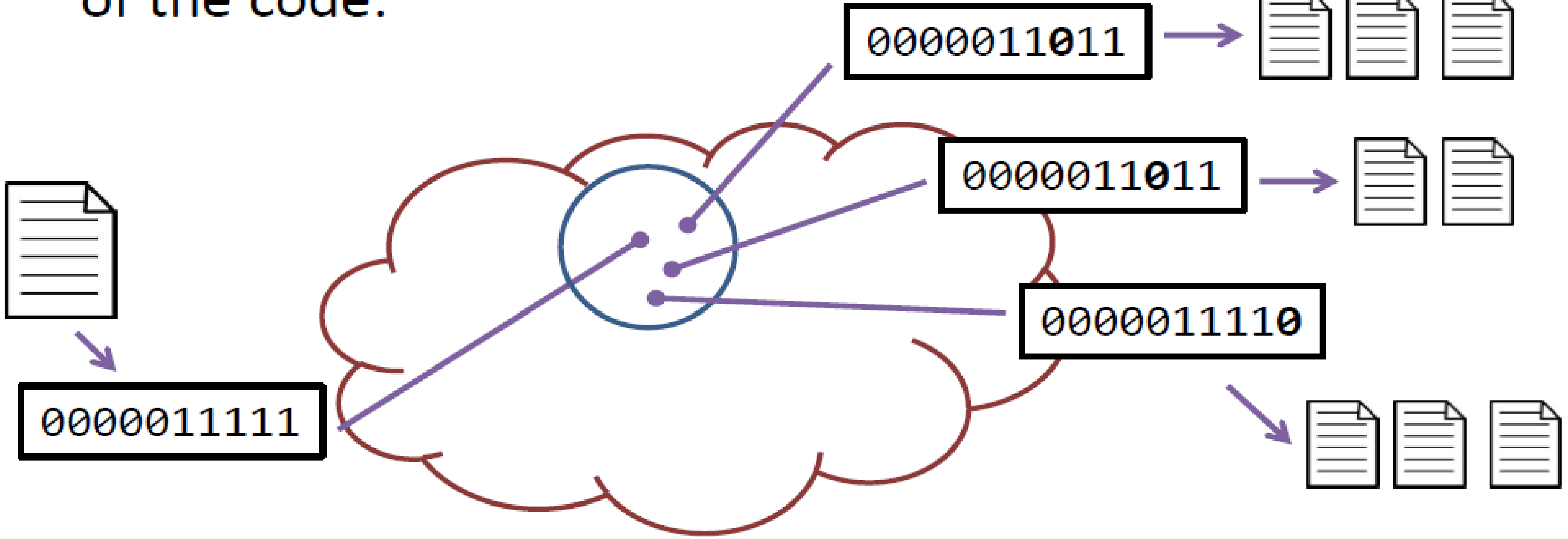
- Forces output to become bimodal
- Round values to 0 or 1 to form a binary vector (ie, code)

Hinton, 2007

Search

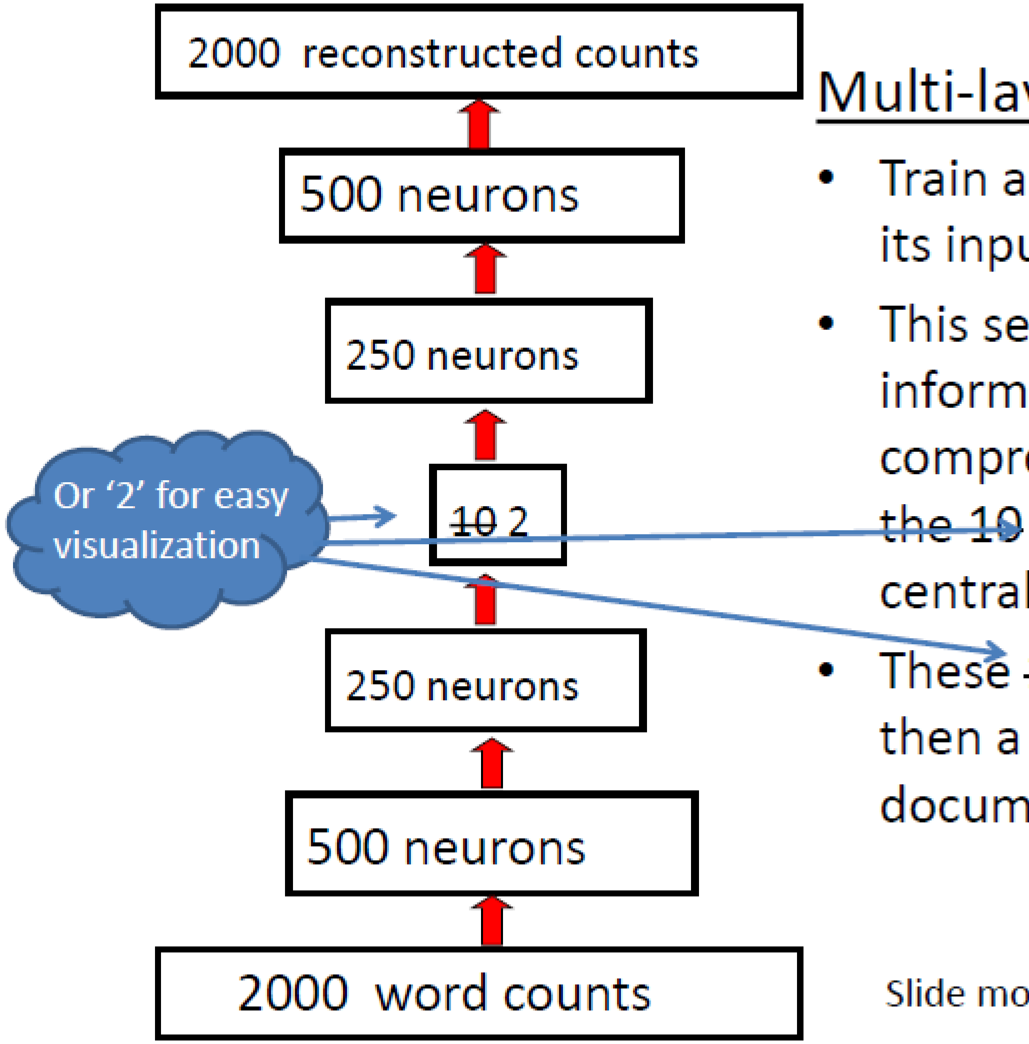
Use the binary codes as a key/hash documents

To find a similar document, calculate binary code and then retrieve documents that correspond to small deviations of the code.



Salakhutdinov and Hinton, 2007

How to compress the count vector

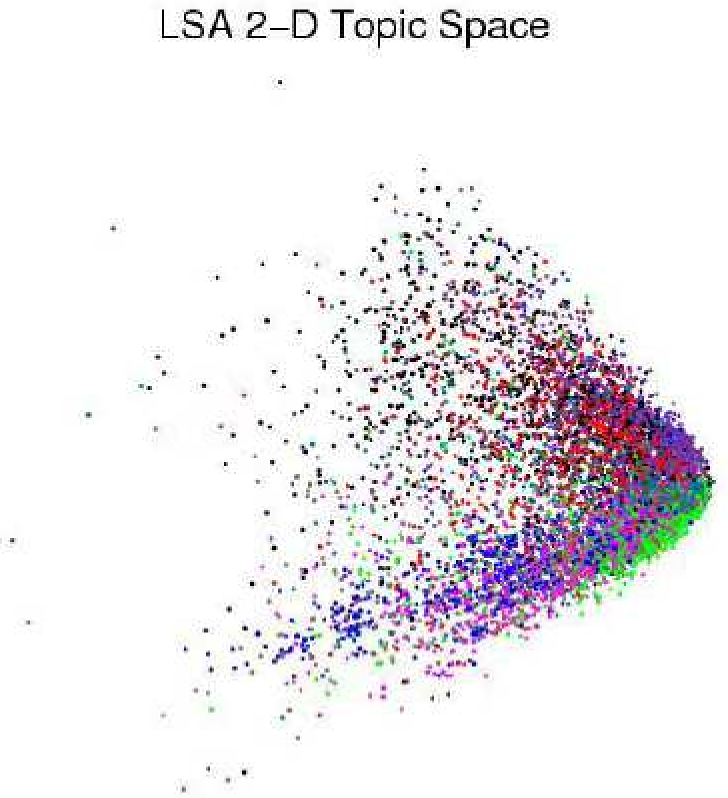


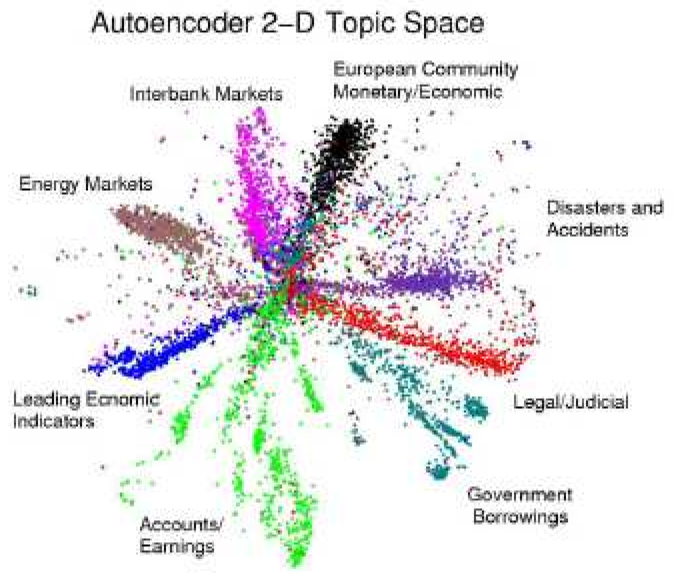
Multi-layer auto-encoder

- Train a model to reproduce its input vector as its output
- This setup forces as much information as possible be compressed and passed thru the ~~10~~ 2 numbers in the central bottleneck.
- These ~~10~~ 2 numbers are then a good way to compare documents.

Slide modified from Hinton, 2007

Clusters

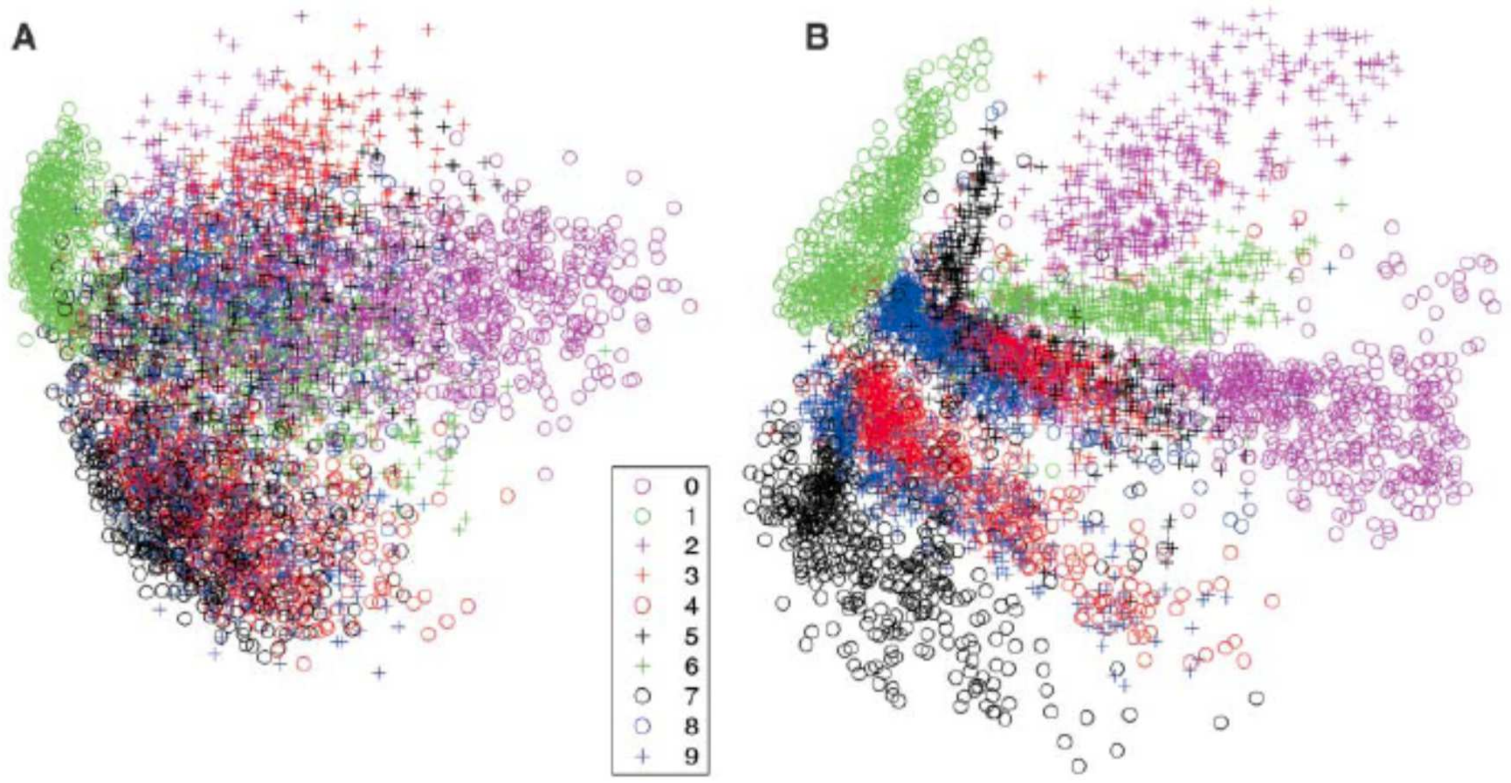




Images from Hinton, 2007

http://www.cs.toronto.edu/~rsalakhu/papers/science.pdf

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*).



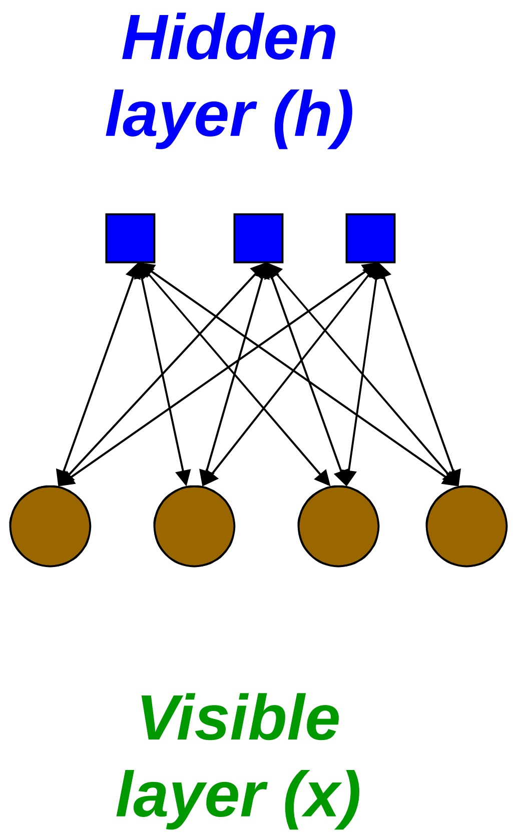
The same trick applied to "digits"

Some references

- Hinton, G. *2007 NIPS Tutorial on: Deep Belief Nets* [PowerPoint slides]. Retrieved from http://www.cs.utoronto.ca /~hinton/.
- Hinton, G. *UCL Tutorial on: Deep Belief Nets* [PowerPoint slides]. Retrieved from http://www.cs.utoronto.ca /~hinton/.
- Hinton, G., Osindero, S. and The, Y. (2006) A fast learning algorithm for deep belief nets. *Neural Computation*, **18**, pp 1527-1554.
- Hinton, G. and Salakhutdinov, R. (2006) Reducing the dimensionality of data with neural networks. *Science*, **313**:5786, pp. 504 – 507.
- Salakhutdinov, R and Hinton, G. (2007) Semantic Hashing. *Proceedings of the SIGIR Workshop on Information Retrieval and applications of Graphical Models*, Amsterdam.
- *Restricted Boltzmann Machines (RBM).* Retrieved from http://deeplearning.net/ tutorial/rbm.html#rbm
- Carreira-Perpinan, M and Hinton, G. (2005) On Contrastive Divergence Learning. *Artificial Intelligence and Statistics*, Barbados.

From: boltzmann_machine.pdf by Jesse Eickholt

RBM: a broader context



RBM is a model of p(x, h)
p(x), p(h|x), p(x|h), …

Autoencoder (feature extractor)

Buliding block for
DeepAutoencoders

Why so confusing?

- Strange terminology
  - energy, partition function, Gibbs sampling, mini batches, …
- Various activation functions (“up” and “down”)
  - logistic, gaussian, softmax, …
- Various training schemes
  - log likelihood
  - energy minimization
  - MCMC (Monte Carlo Markov Chains)
  - Contrastive Divergence
- Various contexts in which RBMs are used
  - feature extraction
  - dimensionality reduction
  - collaborative filtering

A historical perspective

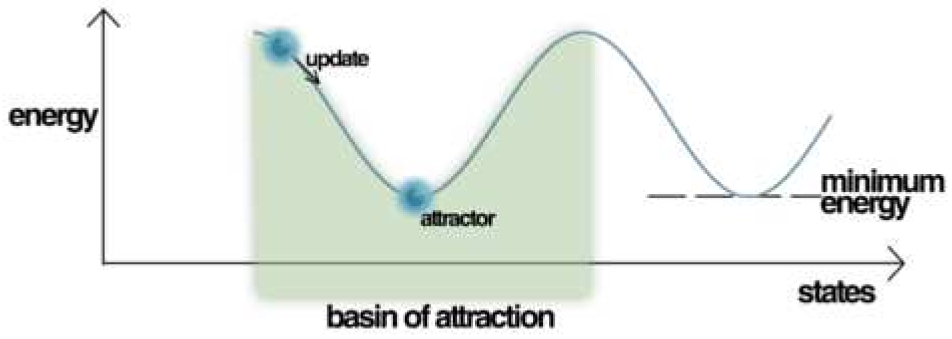
- Origins of RBMs
  - Hebb’s rule and Hopfield Networks
  - Statistical Physics (Ising model)
  - Probabilistic Graphical Models
    - Baysian Networks
    - Markov Models
    - Gibbs Distributions
- Various training schemes
  - energy minimization (Hopfield Networks, Lyapunov Energy)
  - Partitioning Functions, LogLikelihood
  - MCMC (Monte Carlo Markov Chains)
  - Contrastive Divergence: a heuristic simplification of MCMC

Hopfiled Networks (1982)

- A fully connected network with n nodes
- Connections are symmetric ($w_{ij}=w_{ji}$)
- Each node acts as a “0 or 1” or “-1 or +1” perceptron
- Dynamic system: asynchronous updates
- Energy: a level of disagreement
- Every update makes the energy smaller
- Convergence to local minima (stable states)
- http://en.wikipedia.org/wiki/Hopfield_network

Hopfiled Networks

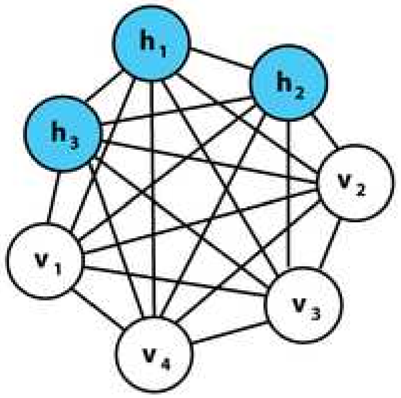
- “Energy Optimizer”



- Memory Model (a learning problem):
  “for a given collection of “training vectors” (e.g., images), find weights such that “stable states” (minimal energy states) correspond to the training examples
- Hebb’s rule:
  “the connection strength between two nodes should be proportional to the correlation of their activations”

From Hopfiled Networks to Boltzmann Machines

- Limited applicability of Hopfield networks
  - very limited number of patterns that could be "memorized"
  - many "spurious states" (unwanted local minima)
  - expensive training process



- Boltzmann Machine: Hopfield Network but some nodes are "hidden"

- **Restricted Boltzmann Machine: the connections are allowed only between "visible" and "hidden" nodes (layers)**

Probabilistic Graphical Models

- Bayesian Networks
- Pairwise Markov Networks
- General Gibbs Distributions
- Partition Functions
- Ising Model

Check the course of Daphne Koller:
Probabilistic Graphical Models
www.coursera.org/course/pgm