Energy-Based Models: Boltzmann Machines & Contrastive Divergence

Alexander G. Ororbia II

Biologically-Inspired Intelligent Systems

CSCI-736

3/7/2023

Thanks to Geoffrey Hinton & Fei-Fei Li

Generative Models

Given training data, generate new samples from same distribution

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

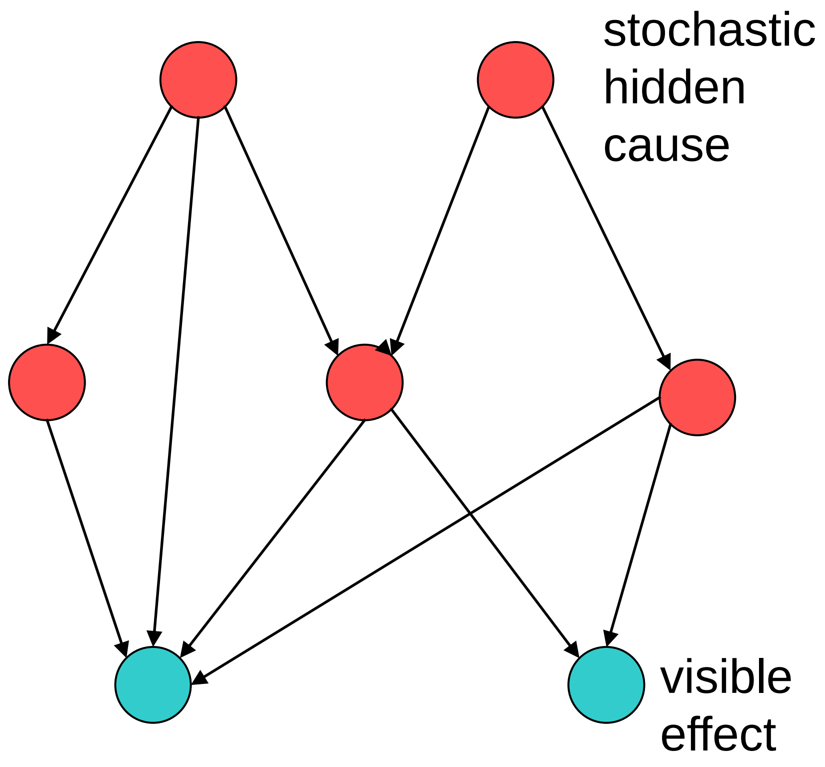
Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

Belief Nets

- A belief net is a directed acyclic graph composed of stochastic variables.
- We get to observe some of the variables and we would like to solve two problems:
- The inference problem: Infer the states of the unobserved variables.
- The learning problem: Adjust the interactions between variables to make the network more likely to generate the observed data.

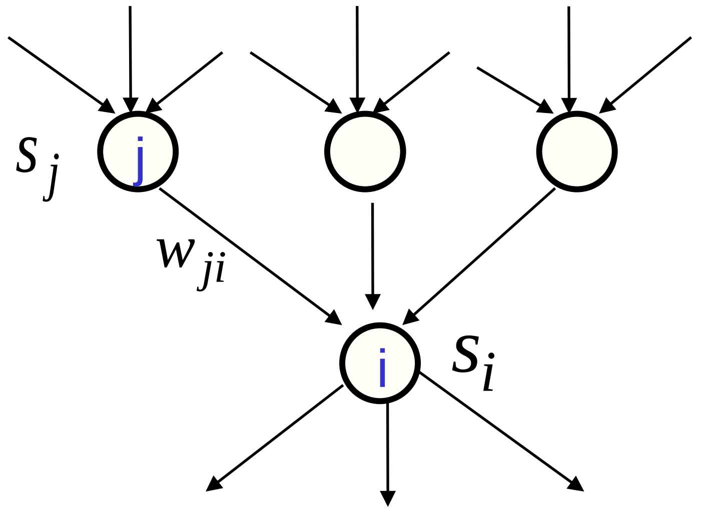


We will use nets composed of layers of stochastic binary variables with weighted connections. Later, we will generalize to other types of variable.

The learning rule for sigmoid belief nets

- Learning is easy if we can get an unbiased sample from the posterior distribution over hidden states given the observed data.

- For each unit, maximize the log probability that its binary state in the sample from the posterior would be generated by the sampled binary states of its parents.

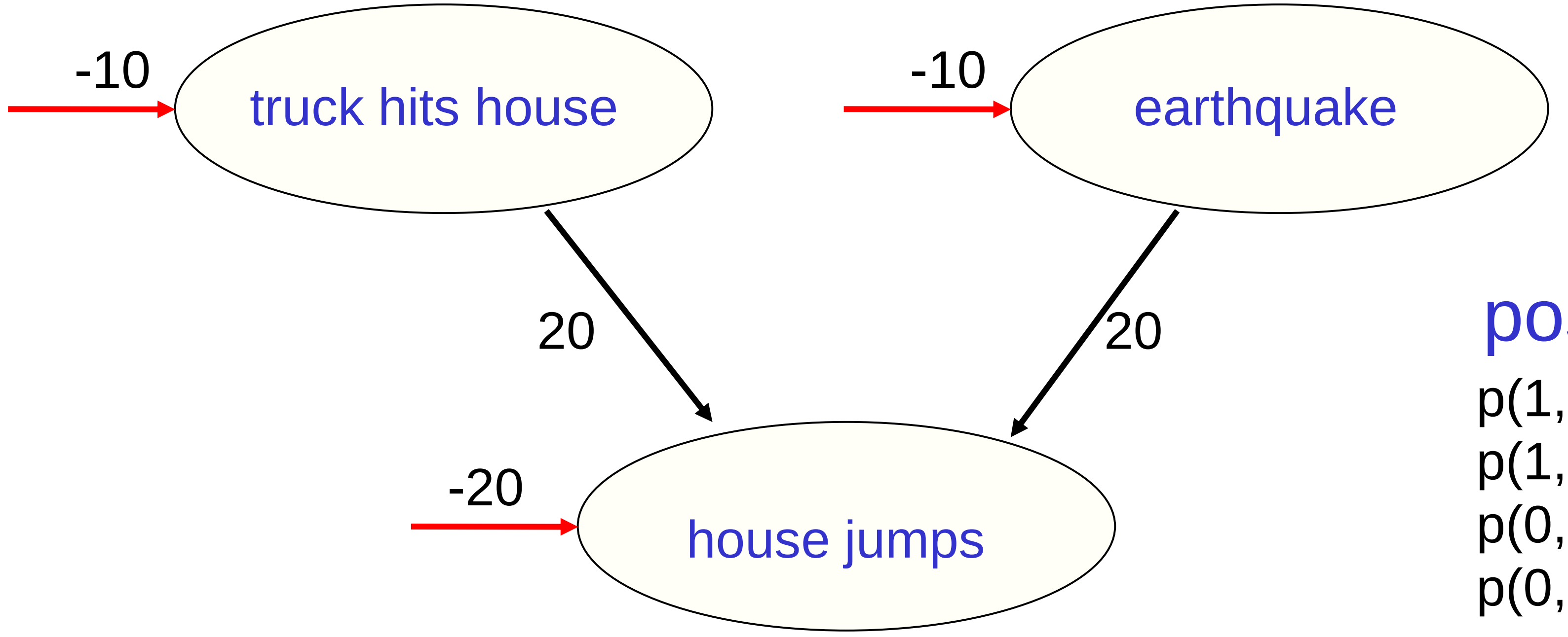


$$p_i \equiv p(s_i = 1) = \frac{1}{1 + \exp(-\sum_j s_j w_{ji})}$$

$$\Delta w_{ji} = \underset{\substack{\Uparrow \\ \text{learning rate}}}{\varepsilon}\, s_j (s_i - p_i)$$

Explaining away (Judea Pearl)

- Even if two hidden causes are independent, they can become dependent when we observe an effect that they can both influence.
  - If we learn that there was an earthquake it reduces the probability that the house jumped because of a truck.



posterior

p(1,1)=.0001

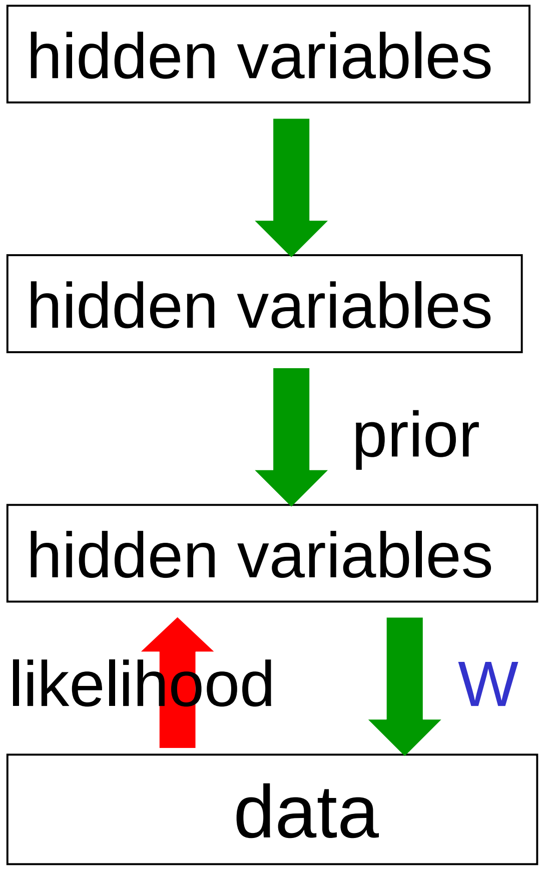
p(1,0)=.4999

p(0,1)=.4999

p(0,0)=.0001

Why it is usually very hard to learn sigmoid belief nets one layer at a time

- To learn W, we need the posterior distribution in the first hidden layer.
- Problem 1: The posterior is typically complicated because of “explaining away”.
- Problem 2: The posterior depends on the prior as well as the likelihood.
  - So to learn W, we need to know the weights in higher layers, even if we are only approximating the posterior. All the weights interact.
- Problem 3: We need to integrate over all possible configurations of the higher variables to get the prior for first hidden layer. Yuk!



Two Types of Generative Neural Networks

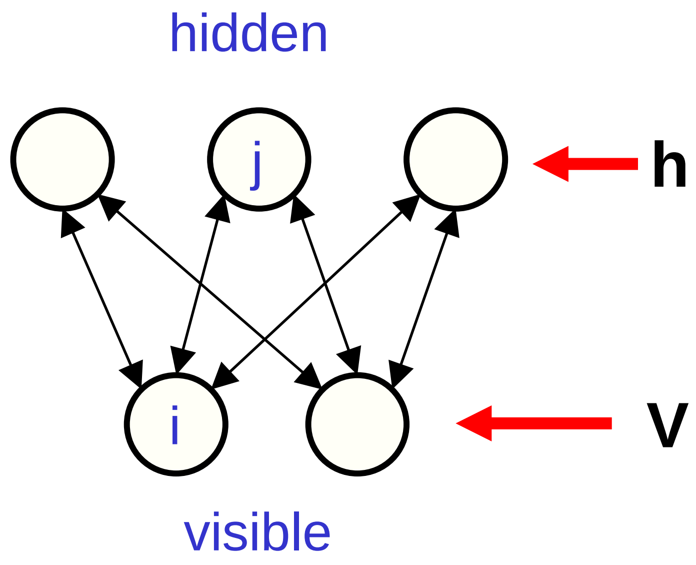
- If we connect binary stochastic neurons in a directed acyclic graph we get a Sigmoid Belief Net (Radford Neal 1992).

- If we connect binary stochastic neurons using symmetric connections we get a Boltzmann Machine (Hinton & Sejnowski, 1983).
  - If we restrict the connectivity in a special way, it is easy to learn a Boltzmann machine.

Restricted Boltzmann Machines (RBMs)

(Smolensky ,1986, called them “harmoniums”)

- We restrict the connectivity to make learning easier.
  - Only one layer of hidden units.
    - We will deal with more layers later
  - No connections between hidden units.
- In an RBM, the hidden units are conditionally independent given the visible states.
  - So we can quickly get an unbiased sample from the posterior distribution when given a data-vector.
  - This is a big advantage over directed belief nets



The Energy of a Joint Configuration

(ignoring terms related to biases)

binary state of visible unit i

binary state of hidden unit j

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

Energy with configuration v on the visible units and h on the hidden units

weight between units i and j

$$-\frac{\partial E(v,h)}{\partial w_{ij}} = v_i h_j$$

Weights → Energies → Probabilities

- Each possible joint configuration of the visible and hidden units has an energy
  - The energy is determined by the weights and biases (as in a Hopfield net)
- Energy of a joint configuration of the visible and hidden units determines its probability:

$$p(v,h) \propto e^{-E(v,h)}$$

- Probability of a configuration over the visible units is found by summing the probabilities of all the joint configurations that contain it

$$p(v) = \frac{1}{Z}\sum_{\{h\}} \exp(-E(v,h))$$

Using Energies to Define Probabilities

- The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energy of all other joint configurations.

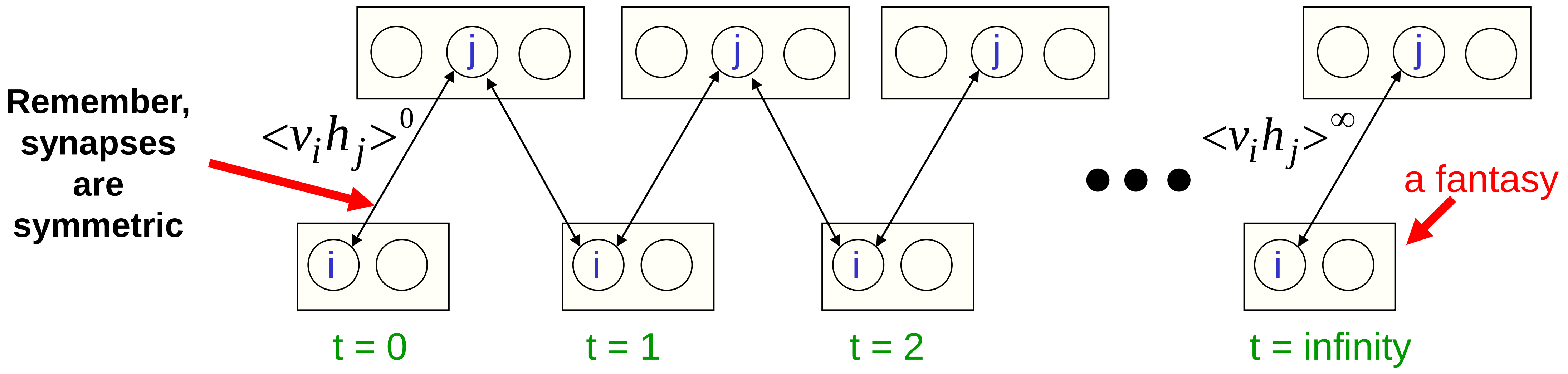
$$p(v,h) = \frac{e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}$$

partition function

- The probability of a configuration of the visible units is the sum of the probabilities of all the joint configurations that contain it.

$$p(v) = \frac{\sum_{h} e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}$$

A Picture of the Maximum Likelihood Learning Algorithm for an RBM

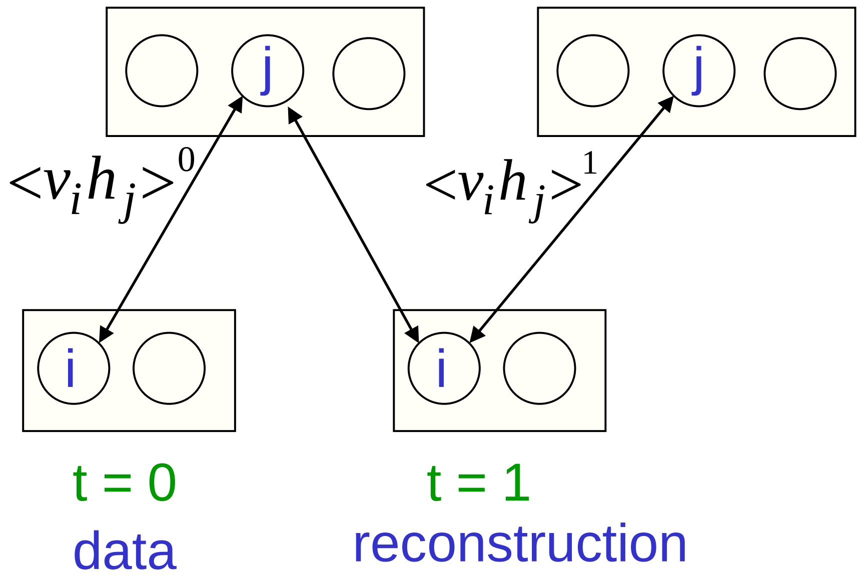


Start with a training vector on the visible units.

Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

$$\frac{\partial \log p(v)}{\partial w_{ij}} = <v_i h_j>^0 - <v_i h_j>^\infty$$

A Quick Way to Learn an RBM



Start with a training vector on the visible units.

Update all the hidden units in parallel

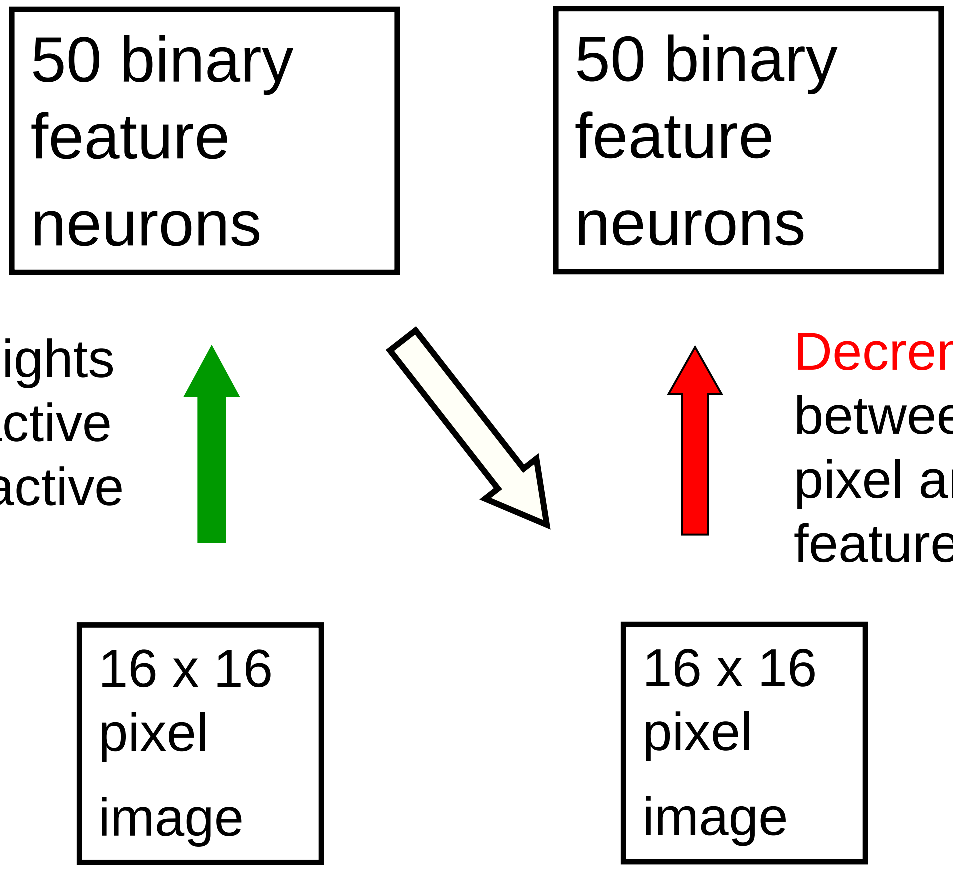
Update the all the visible units in parallel to get a “reconstruction”.

Update the hidden units again.

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

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).

How to learn a set of features that are good for reconstructing images of the digit “2”



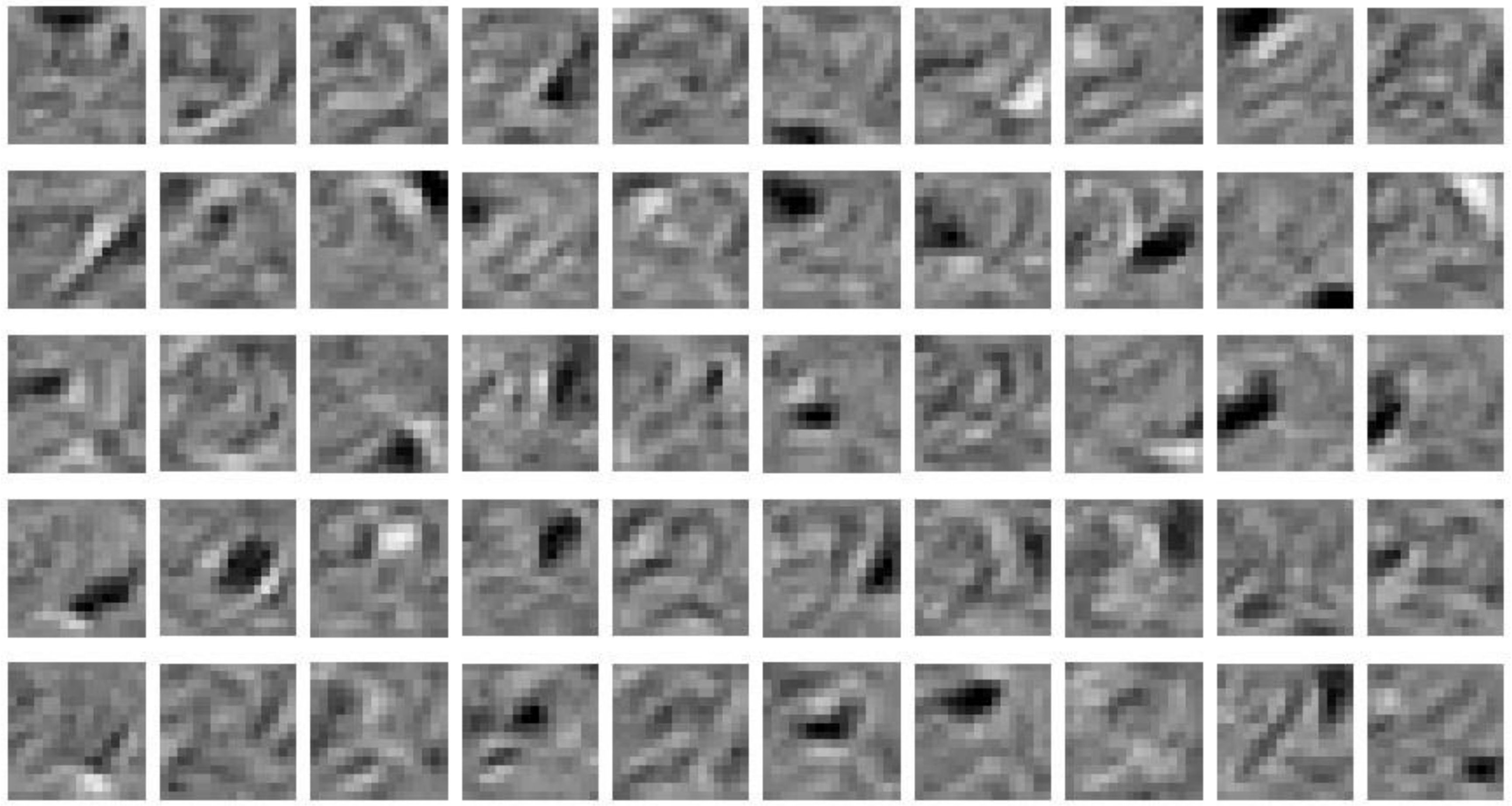
Increment weights between an active pixel and an active feature

Decrement weights between an active pixel and an active feature

data
(reality)

reconstruction
(better than reality)

The Final 50 x 256 Weights (after RBM training)

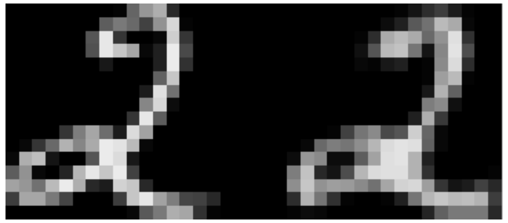


Each neuron grabs a different feature.

How well can we reconstruct the digit images from the binary feature activations?

Data

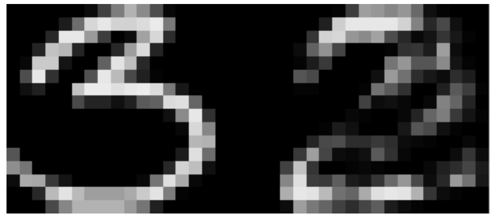
Reconstruction from activated binary features



New test images from the digit class that the model was trained on

Data

Reconstruction from activated binary features



Images from an unfamiliar digit class (the network tries to see every image as a 2)

Three ways to combine probability density models (an underlying theme of the tutorial)

- **Mixture:** Take a weighted average of the distributions.
  - It can never be sharper than the individual distributions. It's a very weak way to combine models.
- **Product:** Multiply the distributions at each point and then renormalize.
  - Exponentially more powerful than a mixture. The normalization makes maximum likelihood learning difficult, but approximations allow us to learn anyway.
- **Composition:** Use the values of the latent variables of one model as the data for the next model.
  - Works well for learning multiple layers of representation, but only if the individual models are undirected.

Training a Deep Network

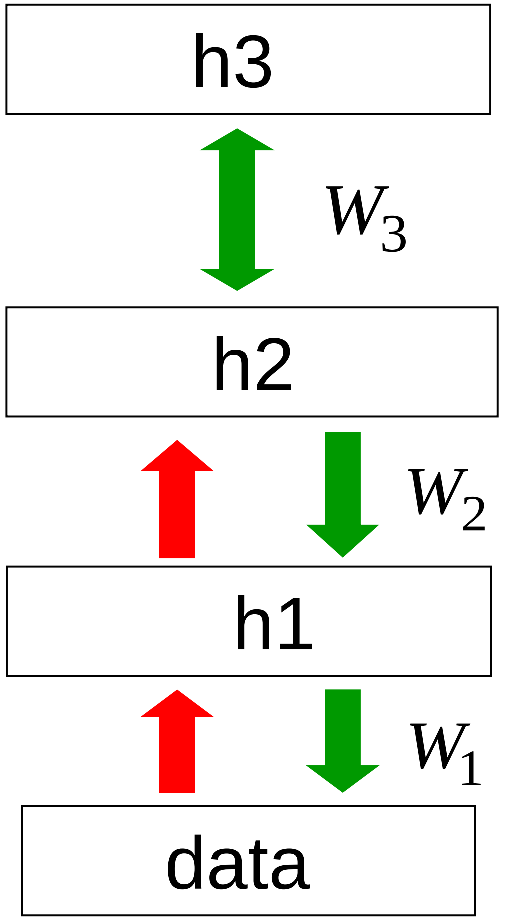
(the main reason RBM's are interesting)

- First train a layer of features that receive input directly from the pixels.
- Then treat the activations of the trained features as if they were pixels and learn features of features in a second hidden layer.
- It can be proved that each time we add another layer of features we improve a variational lower bound on the log probability of the training data.
  - Based on a neat equivalence between an RBM and a deep directed model (through unfolding)

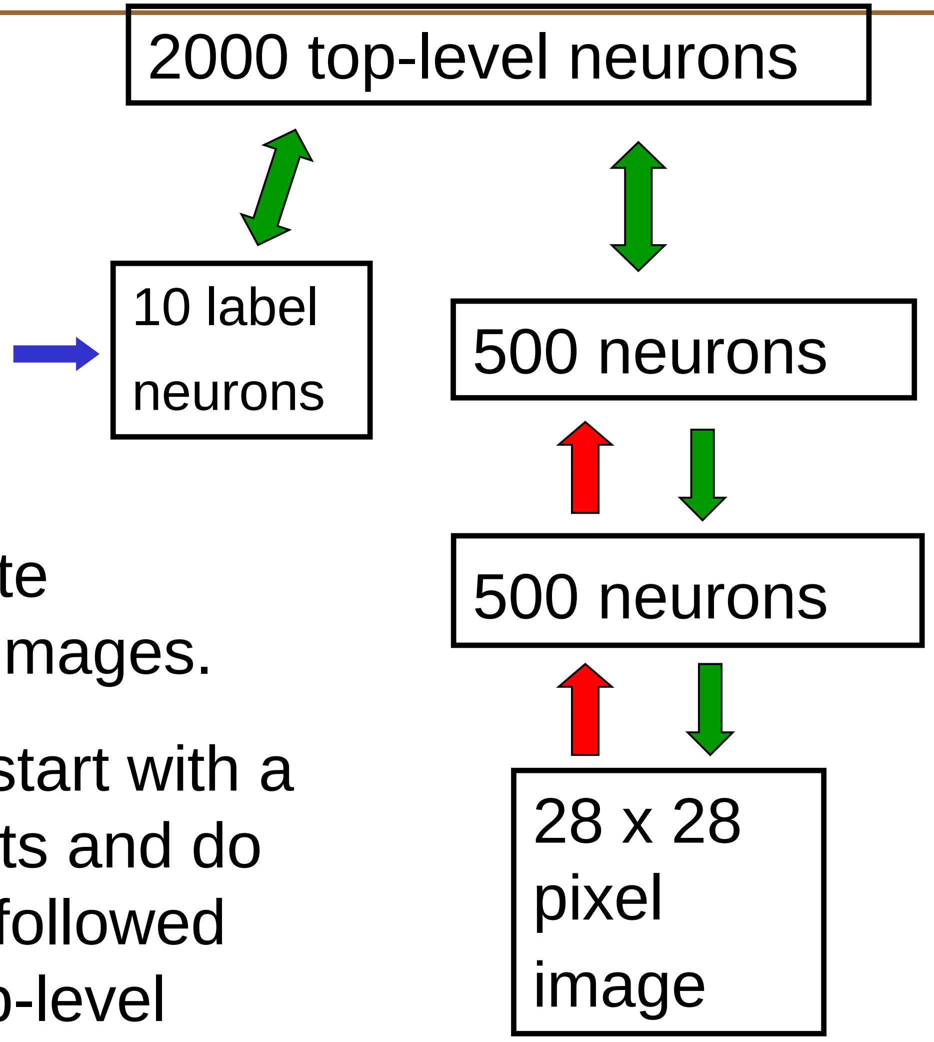
The Generative Model after Learning 3 Layers

- To generate data:
  1. Get an equilibrium sample from the top-level RBM by performing alternating Gibbs sampling for a long time.
  2. 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.



A Model of Digit Recognition



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.

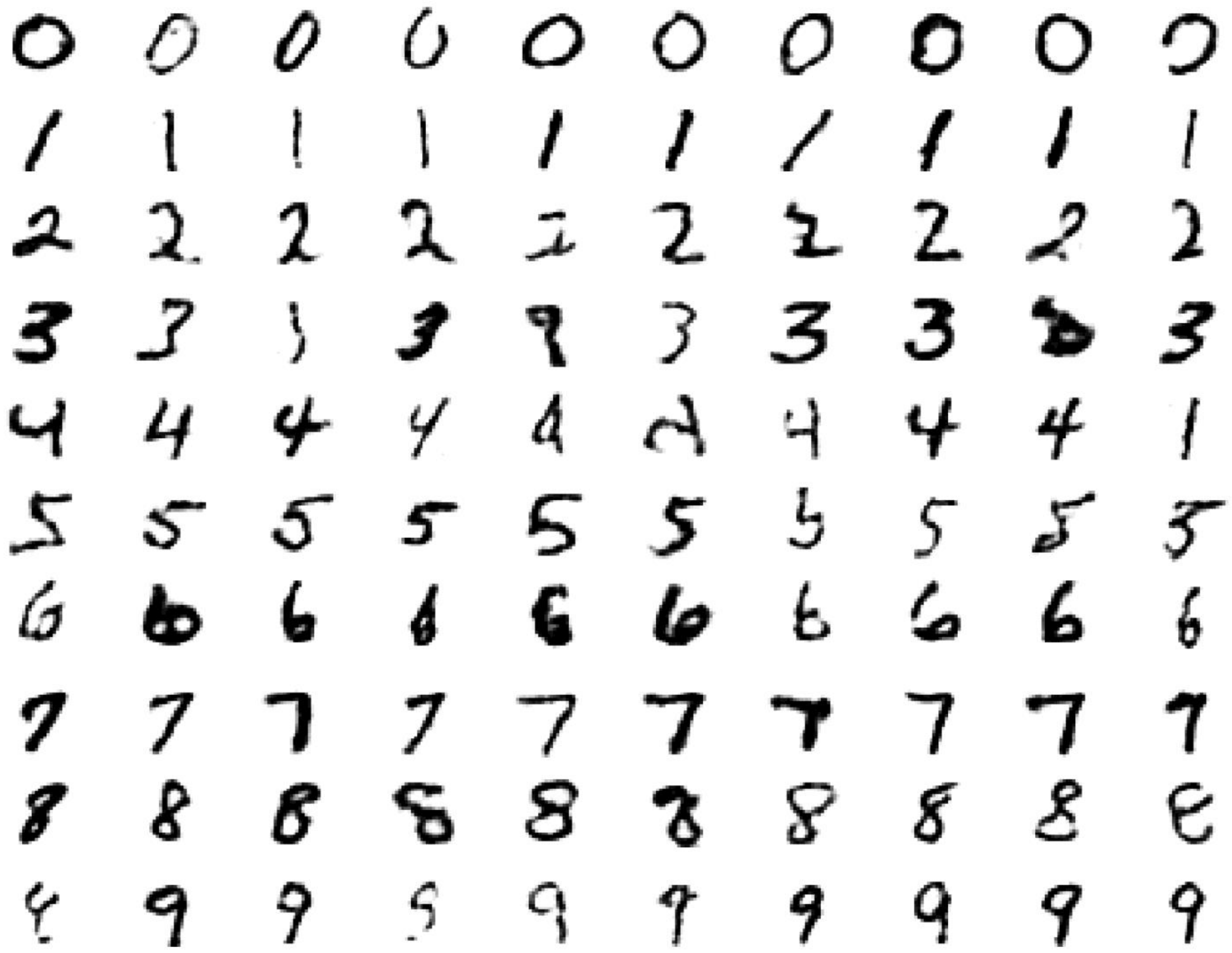
To perform recognition, we start with a neutral state of the label units and do an up-pass from the image followed by a few iterations of the top-level associative memory.

Fine-tuning with a Contrastive Version of the “Wake-Sleep” Algorithm

After learning many layers of features, we can fine-tune the features to improve generation.

1. Do a stochastic bottom-up pass
   - Adjust the top-down weights to be good at reconstructing the feature activities in the layer below.
2. Do a few iterations of sampling in the top level RBM
   -- Adjust the weights in the top-level RBM.
3. Do a stochastic top-down pass
   - Adjust the bottom-up weights to be good at reconstructing the feature activities in the layer above.

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



How well does it discriminate on MNIST test set with no extra information about geometric distortions?

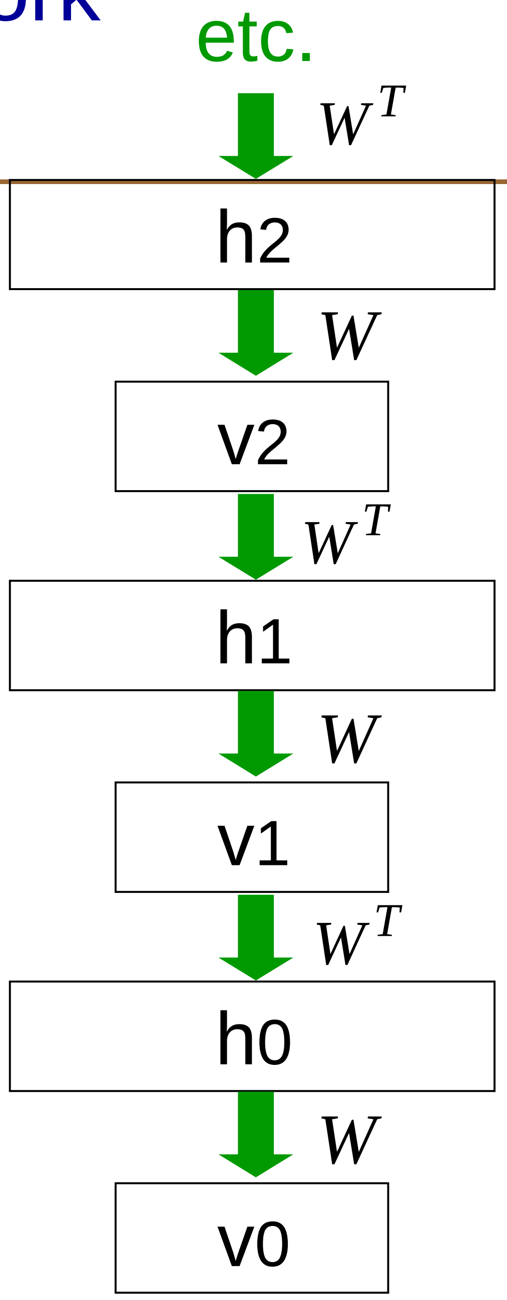
• Generative model based on RBM’s	1.25%
• Support Vector Machine (Decoste et. al.)	1.4%
• Backprop with 1000 hiddens (Platt)	~1.6%
• Backprop with 500 -->300 hiddens	~1.6%
• K-Nearest Neighbor	~ 3.3%

- See Le Cun et. al. 1998 for more results

- Its better than backprop and much more neurally plausible because the neurons only need to send one kind of signal, and the teacher can be another sensory input.

An Infinite Sigmoid Belief Network that is equivalent to an RBM

- The distribution generated by this infinite directed net with replicated weights is the equilibrium distribution for a compatible pair of conditional distributions: p(v|h) and p(h|v) that are both defined by W
  - A top-down pass of the directed net is exactly equivalent to letting a restricted Boltzmann machine settle to equilibrium
  - So this infinite directed net defines the same distribution as an RBM



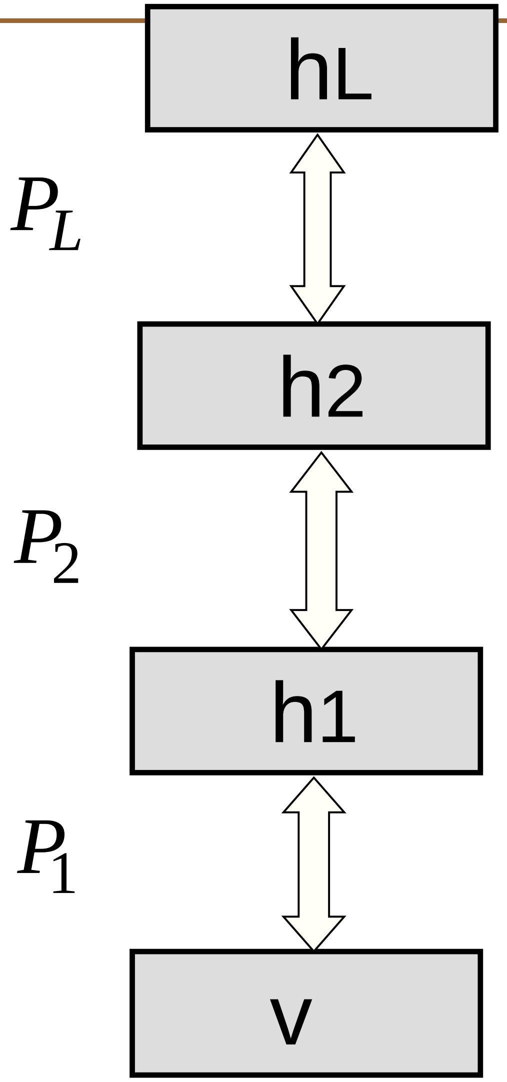
A Stack of RBM's

- Each RBM has the same subscript as its hidden layer.
- Each RBM defines its own distribution over its visible vectors

$$P_l(h_{l-1}) = \frac{\sum_{h_l} \exp(-E(h_{l-1}, h_l))}{Z_l}$$

- Each RBM defines its own distribution over its hidden vectors

$$P_l(h_l) = \frac{\sum_{h_{l-1}} \exp(-E(h_{l-1}, h_l))}{Z_l}$$



Summary so far

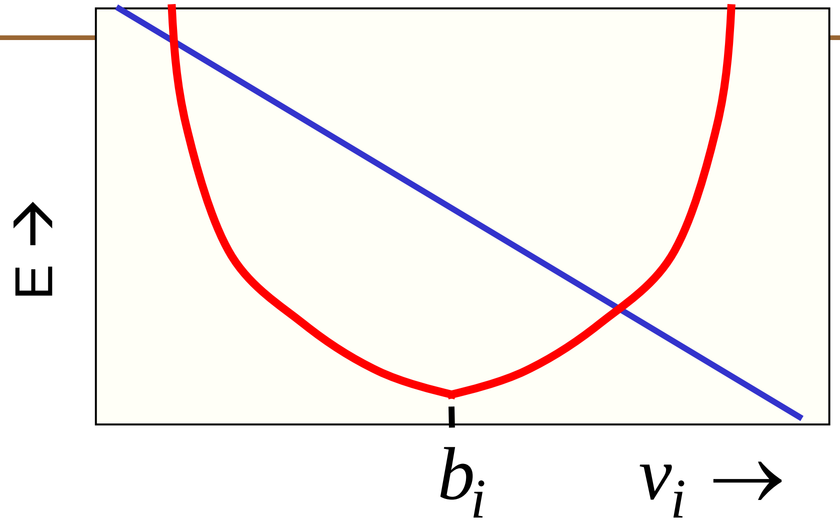
- Restricted Boltzmann machines provide a simple way to learn a layer of features without any supervision
  - Maximum likelihood learning is computationally expensive because of the normalization term, but contrastive divergence learning is fast and usually works well
- Many layers of representation can be learned by treating the hidden states of one RBM as the visible data for training the next RBM (a composition of experts).
- This creates good generative models that can then be fine-tuned
  - Contrastive wake-sleep can fine-tune generation

Modeling Real-Valued Data

- For images of digits it is possible to represent intermediate intensities as if they were probabilities by using “mean-field” logistic units.
  - We can treat intermediate values as the probability that the pixel is inked.
- This will not work for real images.
  - In a real image, the intensity of a pixel is almost always almost exactly the average of the neighboring pixels.
  - Mean-field logistic units cannot represent precise intermediate values.

An RBM with Real-Valued Visible Units

- Using ***Gaussian visible units*** we can get much sharper predictions and alternating Gibbs sampling is still easy, though learning is slower



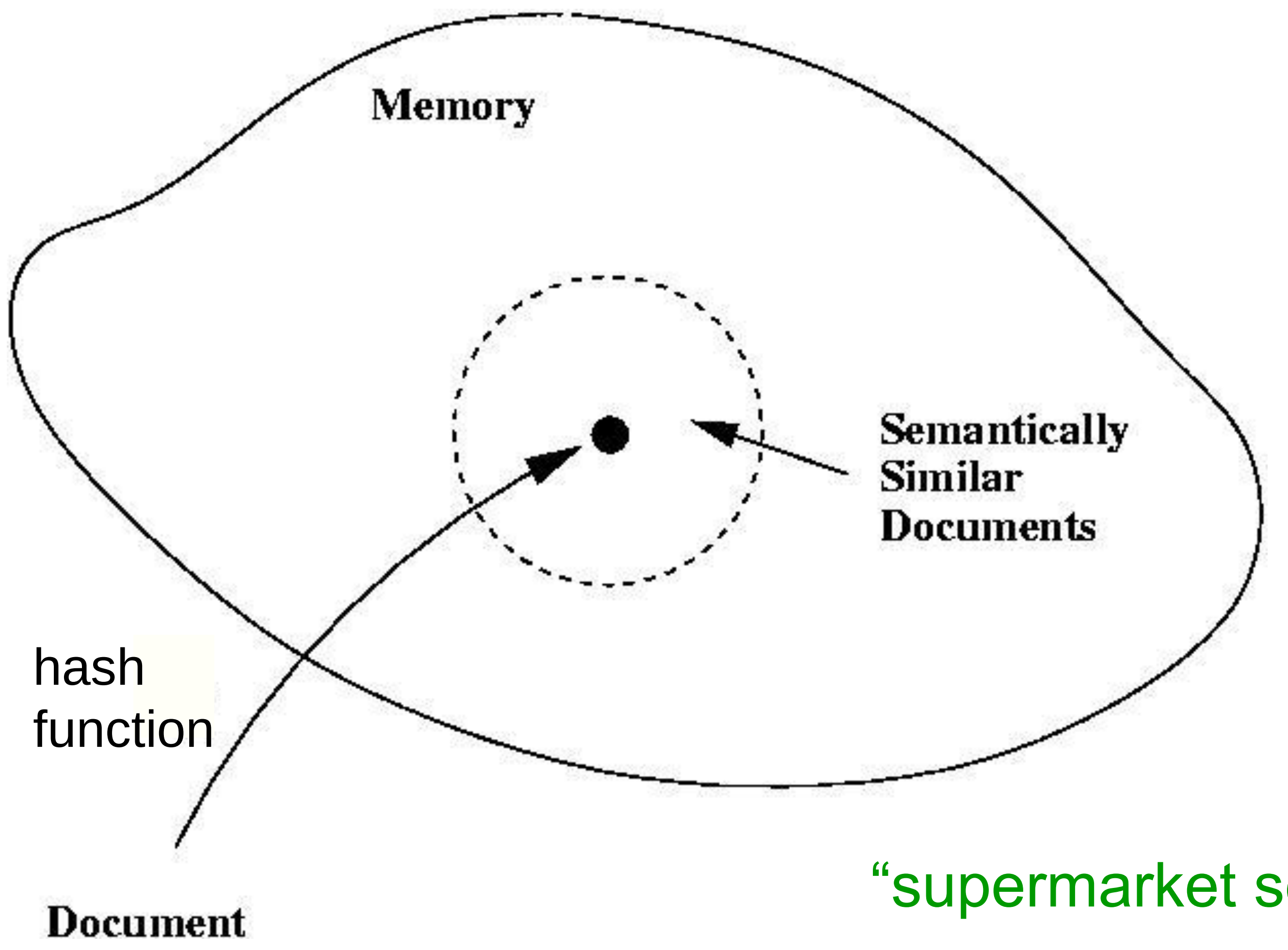
parabolic containment function

energy-gradient produced by the total input to a visible unit

$$E(\mathbf{v},\mathbf{h}) = \sum_{i\,\varepsilon\,vis} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j\,\varepsilon\,hid} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j\, w_{ij}$$

Welling et. al. (2005) show how to extend RBM's to the exponential family. See also Bengio et. al. 2007)

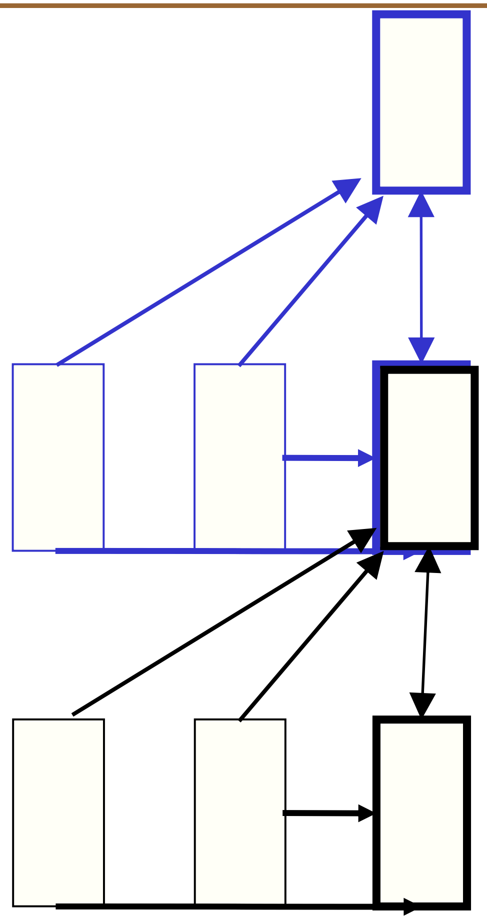
Semantic hashing: Using a deep autoencoder as a hash-function for finding approximate matches (Salakhutdinov & Hinton, 2007)



“supermarket search”

Stacking temporal RBM's

- Treat the hidden activities of the first level TRBM as the data for the second-level TRBM.
  - So when we learn the second level, we get connections across time in the first hidden layer.
- After greedy learning, we can generate from the composite model
  - First, generate from the top-level model by using alternating Gibbs sampling between the current hiddens and visibles of the top-level model, using the dynamic biases created by the previous top-level visibles.
  - Then do a single top-down pass through the lower layers, but using the autoregressive inputs coming from earlier states of each layer.



Questions??