Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 11.1: Restricted Boltzmann Machines

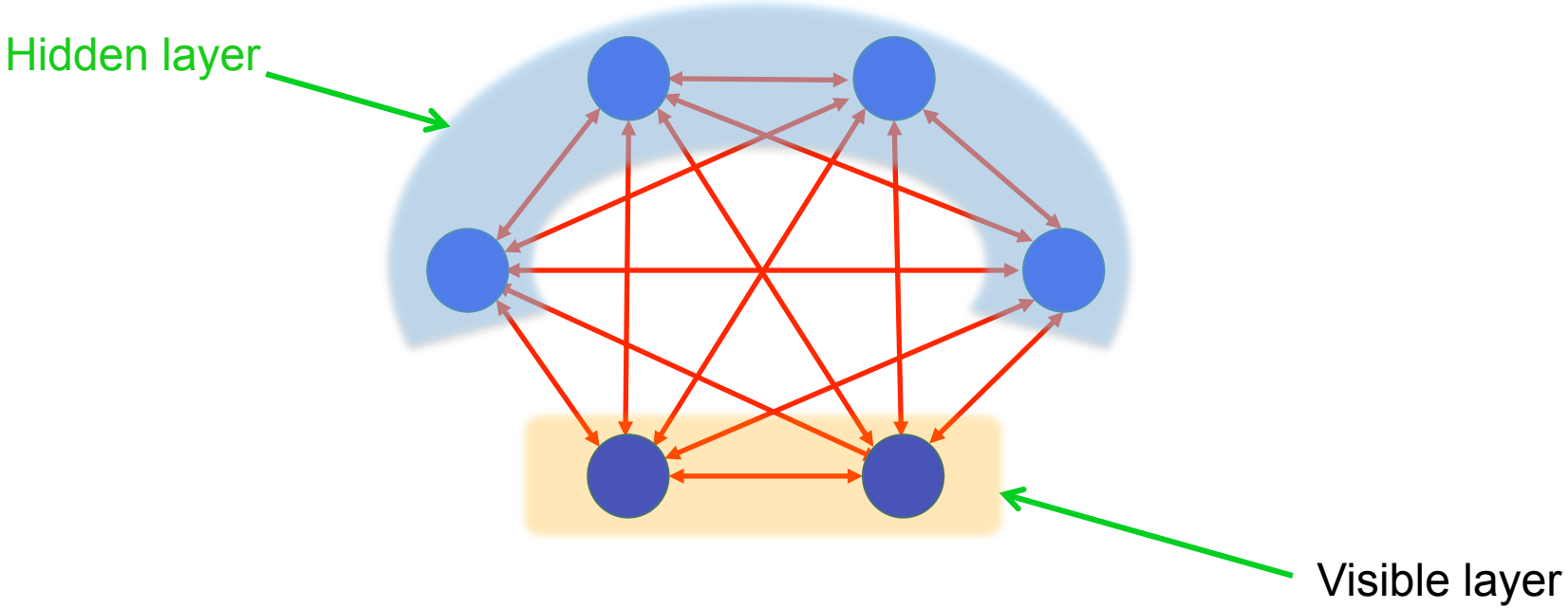
What We've Covered So Far

- Boltzmann machines are essentially stochastic versions of Hopfield networks.
- Stochastic assignment of node states using a binary value.
- Associate a performance metric via the energy or consensus function with network configurations.
- Capability to 'anneal' a BM to solve various optimization problems.

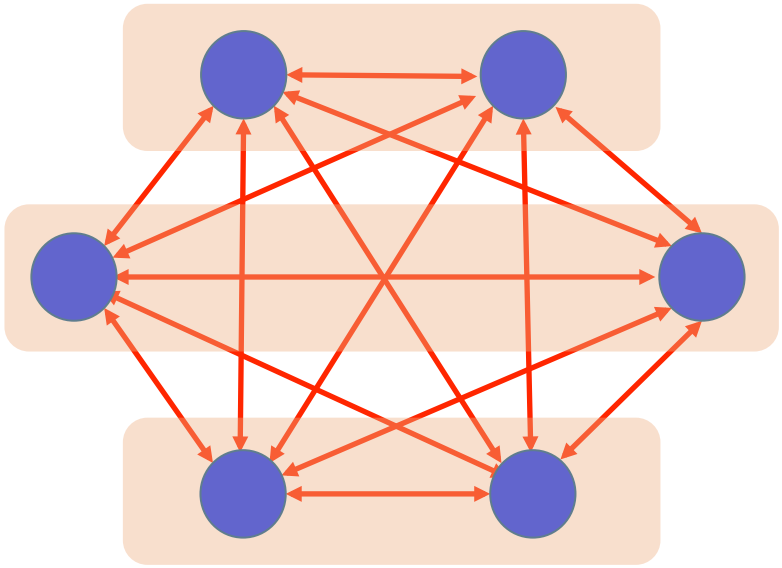
Model Relationships
for Recurrent Networks

Deterministic		Stochastic
Hopfield Networks	→	Boltzmann Machines
Binary Associative Memories	→	Restricted Boltzmann Machines

Restricted Boltzmann Machines



Restricted Boltzmann Machines

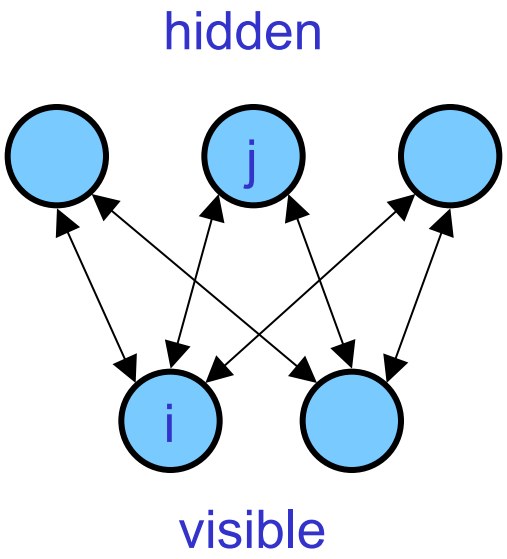


Sort of a multi-layer BAM!

Configured as a *Belief Network*

Restricted Boltzmann Machines

- 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



From Hinton 2007

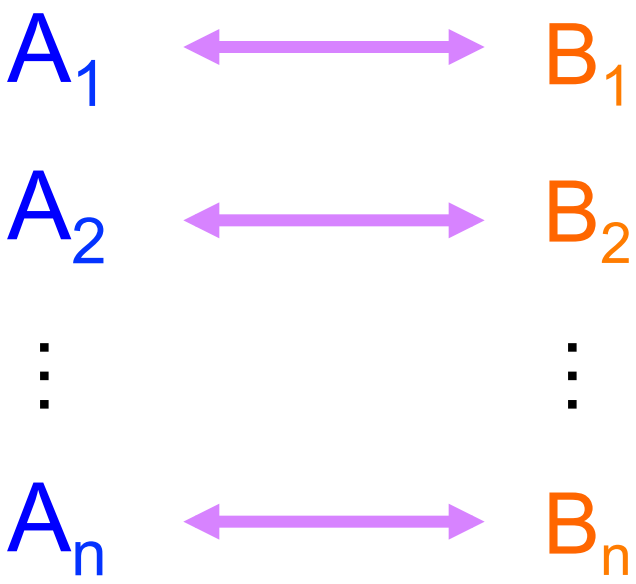
Purpose of RBMs

- Many possible applications.
- Essentially attempts to establish a useful association between visible vectors and hidden vectors similar in nature to how BAMs function.
  - used in factor analysis
  - character recognition
  - many others---active area of research
- Stochastic states allow for greater flexibility and noise

What Do RBMs Do?

- Just like BAMs, they attempt to 'reconstruct' a training vector.
  - In BAMs these were the 'A' vectors or the 'data'.
  - Noisy or variable training vectors probabilistically produce feature detectors in the hidden layer which then probabilistically produce 'reconstructions' of the training vectors.

Binary Associative Memories



Goal: Noisy A_1 produces a correct B_1 which then produces a correct A_1.

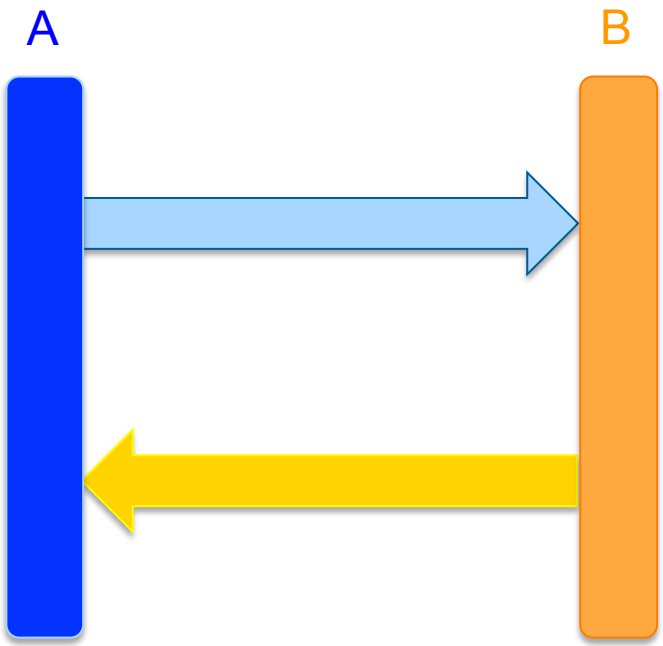
$$\tilde{A}_1 \rightarrow B_1 \rightarrow A_1$$

Binary Associative Memories

- Present a noisy A as input to the A nodes.
- The A nodes produce outputs and are presented to the B nodes.
- The B nodes produce outputs and are presented back to the A nodes.

Binary Associative Memories

Restricted Boltzmann Machines

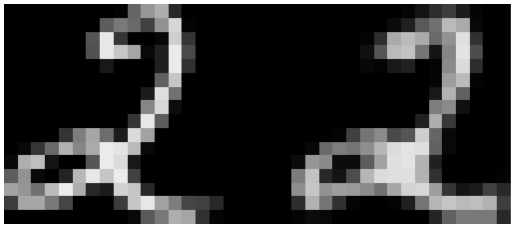


Only Stochastically!

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

Data

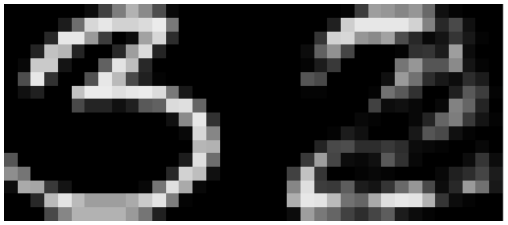
Reconstruction from activated binary features



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

Data

Reconstruction from activated binary features



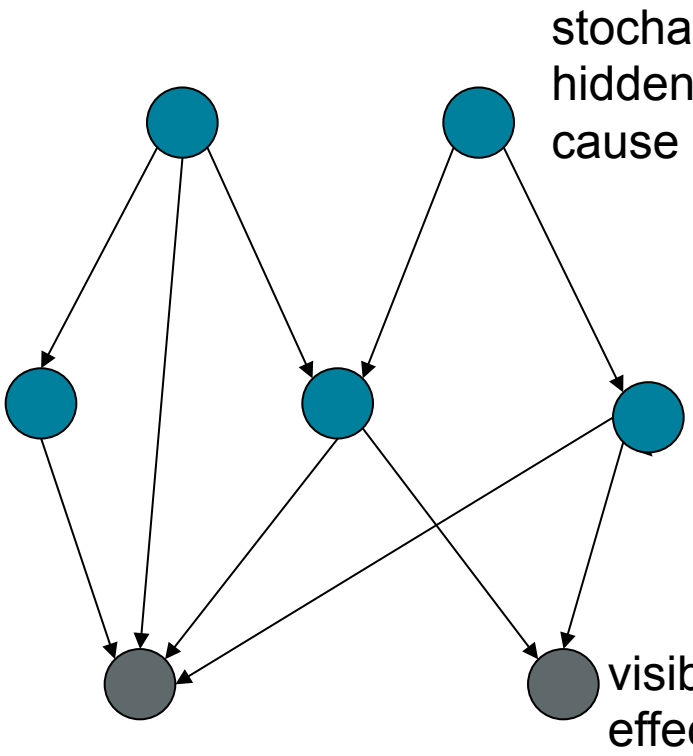
Images from an unfamiliar digit class
(the network tries to see every image as a 2)

From Hinton 2007

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.

From Hinton 2007

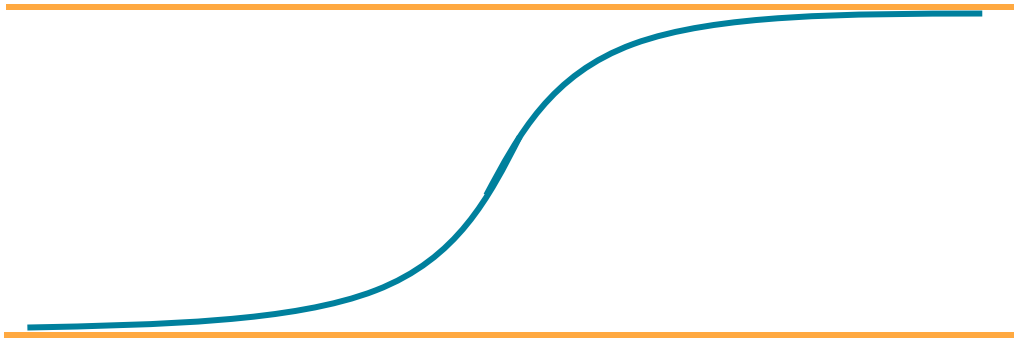


We will use nets composed of layers of stochastic binary variables with weighted connections

Dynamics

Recall the Sigmoid activation function

$$\frac{1}{1+e^{-S_i/T}}$$



Again, we're dealing with stochastic neurons.
What does this curve remind you of?

Dynamics

Set cell activation (state) according to:

$$x_i = \begin{cases} 1 & \text{with probability } p_i = \dfrac{1}{1+e^{-S_i/T}} \\ 0 & \text{with probability } 1-p_i \end{cases}$$

The Energy/Consensus Function

Define the energy function *E*. With the BM we used

$$E = -\sum_{i<j} w_{ij} x_i x_j - \sum_i \theta_i x_i - \sum_j \xi_j x_j$$

In RBMs, only the connections between layers is important.

$$E = -\sum_{i,j} w_{ij} v_i h_j - \sum_i \theta_i v_i - \sum_j \xi_i h_i$$

The Energy of a joint configuration

(ignoring terms with biases)

binary state of visible unit i

binary state of hidden unit j

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

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

weight between units i and j

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

From Hinton 2007 (modified)

Think in terms of consensus

$$C(\mathbf{v},\mathbf{h}) = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} v_i h_j$$

If we want to strengthen an association where both v_i and h_j are 1s,

make w_{ij} a large positive number.

If we want to strengthen an association where v_i and h_j have opposite states,

make w_{ij} a large negative number.

Summary

- RBMs are stochastic versions of BAMs.
  - Two layers: *visible* and *hidden*
- Use probabilistic machinery of Boltzmann machines.
  - Probability of a node's state is based on sigmoid function with activity value based on weighted inputs from 'the other set of nodes'
  - Energy of a (v, h) pair is defined as sum of the weighted products of visible node states and hidden node states.

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 11.2: RBM Mathematics

What We've Covered So Far

- Probabilistic foundations of RBMs.
  - *Energy/Consensus* associated with a visible and hidden pair of vectors.
  - Probability of a node's states

Goal

- Raise the probability that a visible vector with be faithfully reconstructed when a 'hidden' vector is presented to the visible layer.

Question:

How do we train an RBM so that 'reconstructions' are likely to create a reasonable facsimile of the original data?

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).
- The energy of a joint configuration of the visible and hidden units determines its probability:

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

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

From Hinton 2007 (modified)

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

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

partition function

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

From Hinton 2007 (modified)

How Do We Train an RBM?

$$p(\mathbf{v}) = \frac{\sum_{g} e^{-E(\mathbf{v},\mathbf{h}^g)}}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}} \rightarrow \ln p(\mathbf{v}) = \ln \left(\frac{\sum_{g} e^{-E(\mathbf{v},\mathbf{h}^g)}}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}} \right)$$

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

$$\frac{\partial \ln p(\mathbf{v})}{\partial w_{ij}} = \underbrace{\frac{\partial}{\partial w_{ij}} \ln \sum_{g} e^{-E(\mathbf{v},\mathbf{h}^g)}}_{\text{A}} - \underbrace{\frac{\partial}{\partial w_{ij}} \ln \sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}}_{\text{B}}$$

Looking at Term A:

$$\frac{\partial}{\partial w_{ij}} \ln \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} = \frac{1}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} \cdot \frac{\partial}{\partial w_{ij}} \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}$$

$$= \frac{1}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} \cdot \sum_g \frac{\partial e^{-E(\mathbf{v},\mathbf{h}^g)}}{\partial w_{ij}}$$

$$= \frac{1}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} \cdot \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} \cdot \frac{\partial\left(-E(\mathbf{v},\mathbf{h}^g)\right)}{\partial w_{ij}}$$

Looking at Term A:

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

$$\frac{\partial}{\partial w_{ij}} \ln \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} = \frac{1}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} \cdot \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} \cdot \frac{\partial\left(-E(\mathbf{v},\mathbf{h}^g)\right)}{\partial w_{ij}}$$

$$= \frac{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} v_i h_j^g}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} = \sum_g p(\mathbf{h}^g \mid \mathbf{v}) v_i h_j^g = \langle v_i \cdot h_j \rangle_{\mathbf{v}}$$

Where does that conditional probability come from?

$$\frac{e^{-E(\mathbf{v},\mathbf{h}^g)}}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} = p(\mathbf{h}^g \,|\, \mathbf{v})$$

$$p(\mathbf{h}^g \,|\, \mathbf{v}) = \frac{p(\mathbf{v},\mathbf{h}^g)}{p(\mathbf{v})} = \frac{\frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h}^g)}}{\frac{1}{Z}\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} = \frac{e^{-E(\mathbf{v},\mathbf{h}^g)}}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}}$$

Looking at Term B:

$$\frac{\partial}{\partial w_{ij}} \ln \sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)} = \frac{1}{\sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)}} \cdot \frac{\partial}{\partial w_{ij}} \sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)}$$

$$= \frac{1}{\sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)}} \cdot \sum_{u,g} \frac{\partial e^{-E(\mathbf{v}^u, \mathbf{h}^g)}}{\partial w_{ij}}$$

Looking at Term B:

$$\frac{\partial}{\partial w_{ij}} \ln \sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)} = \frac{\sum_{u,g} \frac{\partial e^{-E(\mathbf{v}^u, \mathbf{h}^g)}}{\partial w_{ij}}}{\sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)}}$$

$$= \frac{\sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)} \frac{\partial\left(-E(\mathbf{v}^u, \mathbf{h}^g)\right)}{\partial w_{ij}}}{\sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)}}$$

$$= \frac{\sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)} v_i^u h_j^g}{\sum_{u,g} e^{-E(\mathbf{v}^u, \mathbf{h}^g)}} = \sum_{u,g} p(\mathbf{v}^u, \mathbf{h}^g) v_i^u h_j^g = \langle v_i \cdot h_j \rangle_{\mathbf{vh}}$$

Basis of *Contrastive Divergence*

$$\frac{\partial \ln p(\mathbf{v})}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \ln \sum_{g} e^{-E(\mathbf{v},\mathbf{h}^g)} - \frac{\partial}{\partial w_{ij}} \ln \sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}$$

$$= \langle v_i \cdot h_j \rangle_{\mathbf{v}} - \langle v_i \cdot h_j \rangle_{\mathbf{vh}}$$

Summary

- Showed the derivative of the log probability with respect to weights
- This can serve as the basis of a gradient ascent method for increasing the probability of the reconstructed vector v.

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

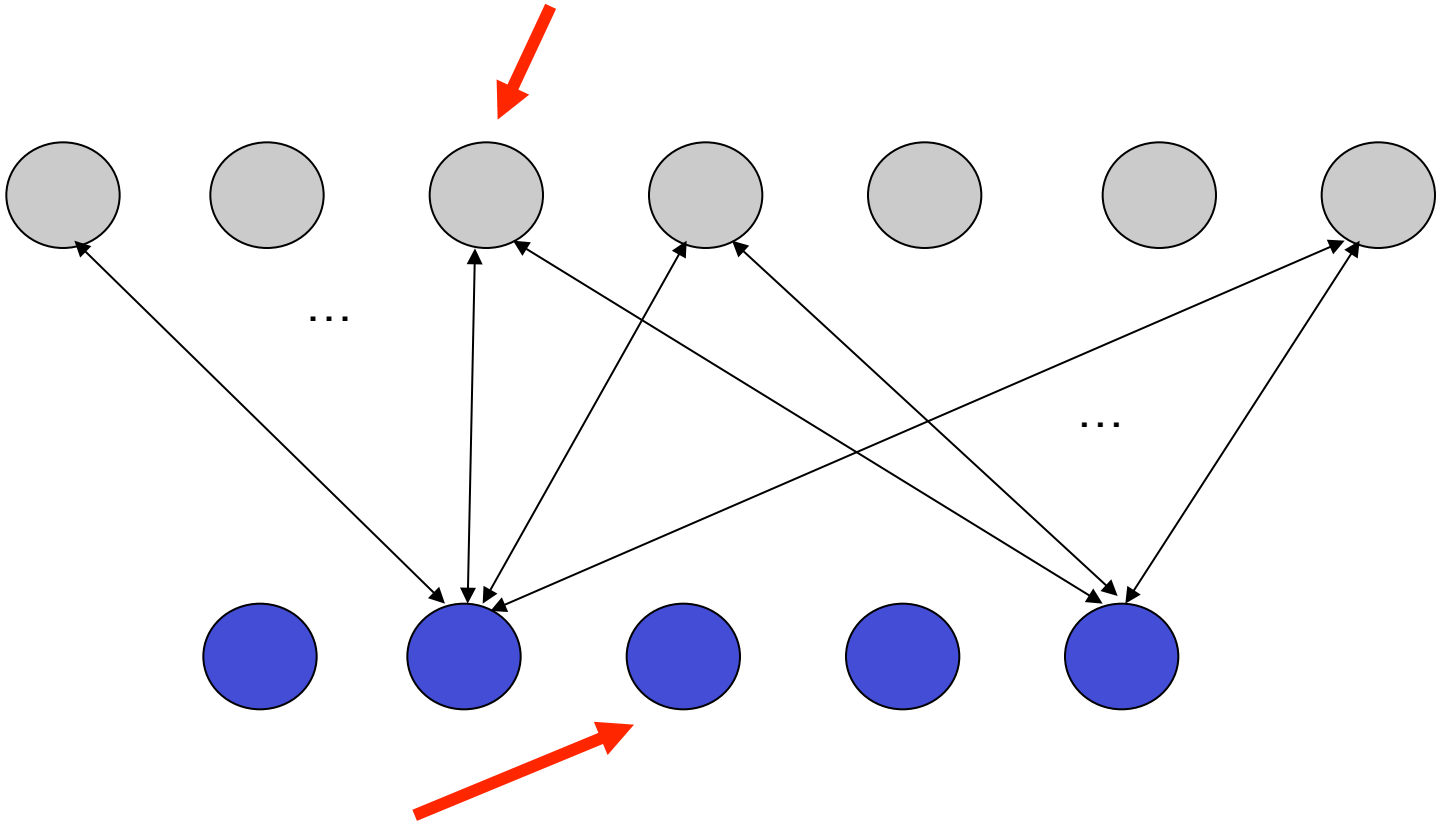
Module 11.3: An RBM Training Example

What We've Covered So Far

- Examined the mathematical foundations for an efficient approach to training RBMs.
- Derivative of the log probability of a vector **v.**
- Two expectations involved.
  - First term based on frequency of $v_i h_j$ over the training set of vectors **v.**
  - Second term based on frequency of $v_i h_j$ over all vectors **v** and **h.**

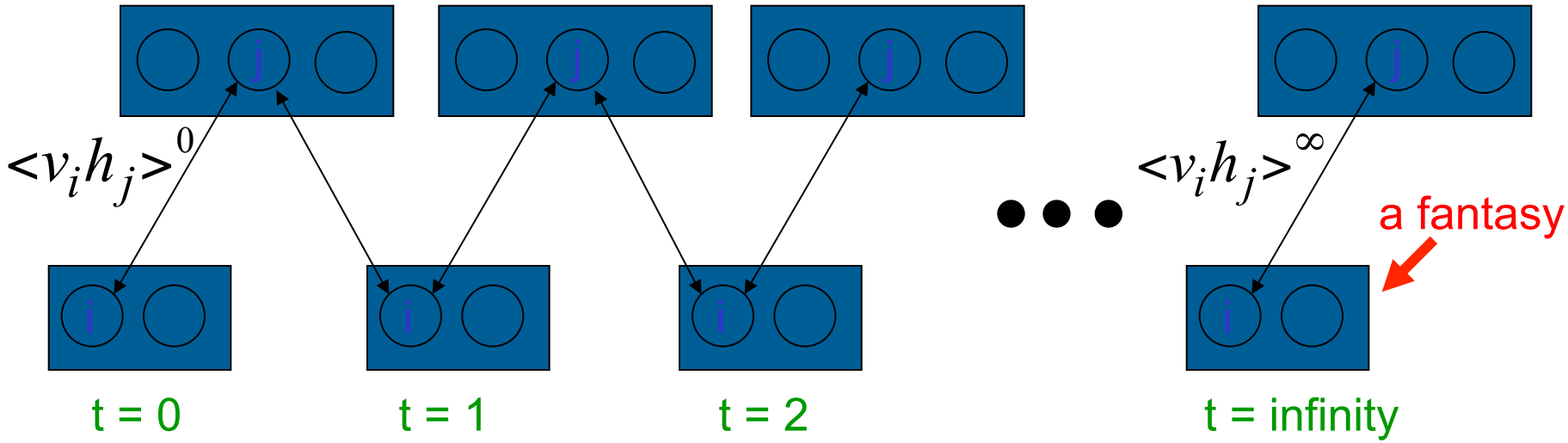
RBM

Hidden Layer of Nodes/Feature Detectors



Visible Layer of Nodes

A picture of the maximum likelihood learning algorithm for an RBM



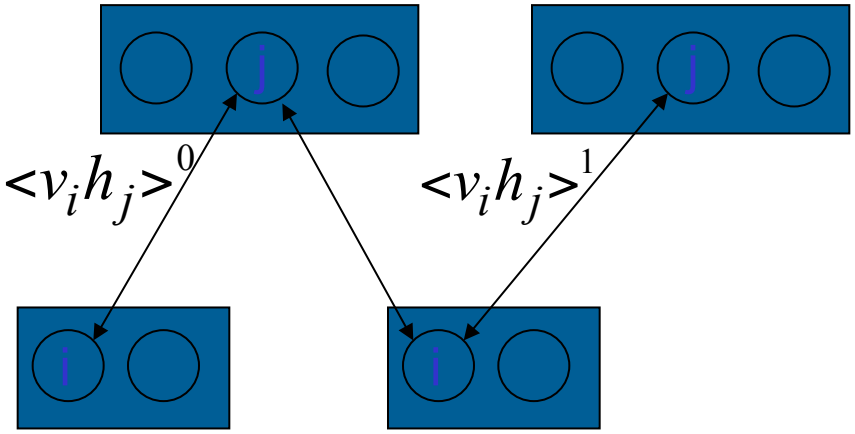
Start with a training vector on the visible units.

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

From Hinton 2007

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

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.

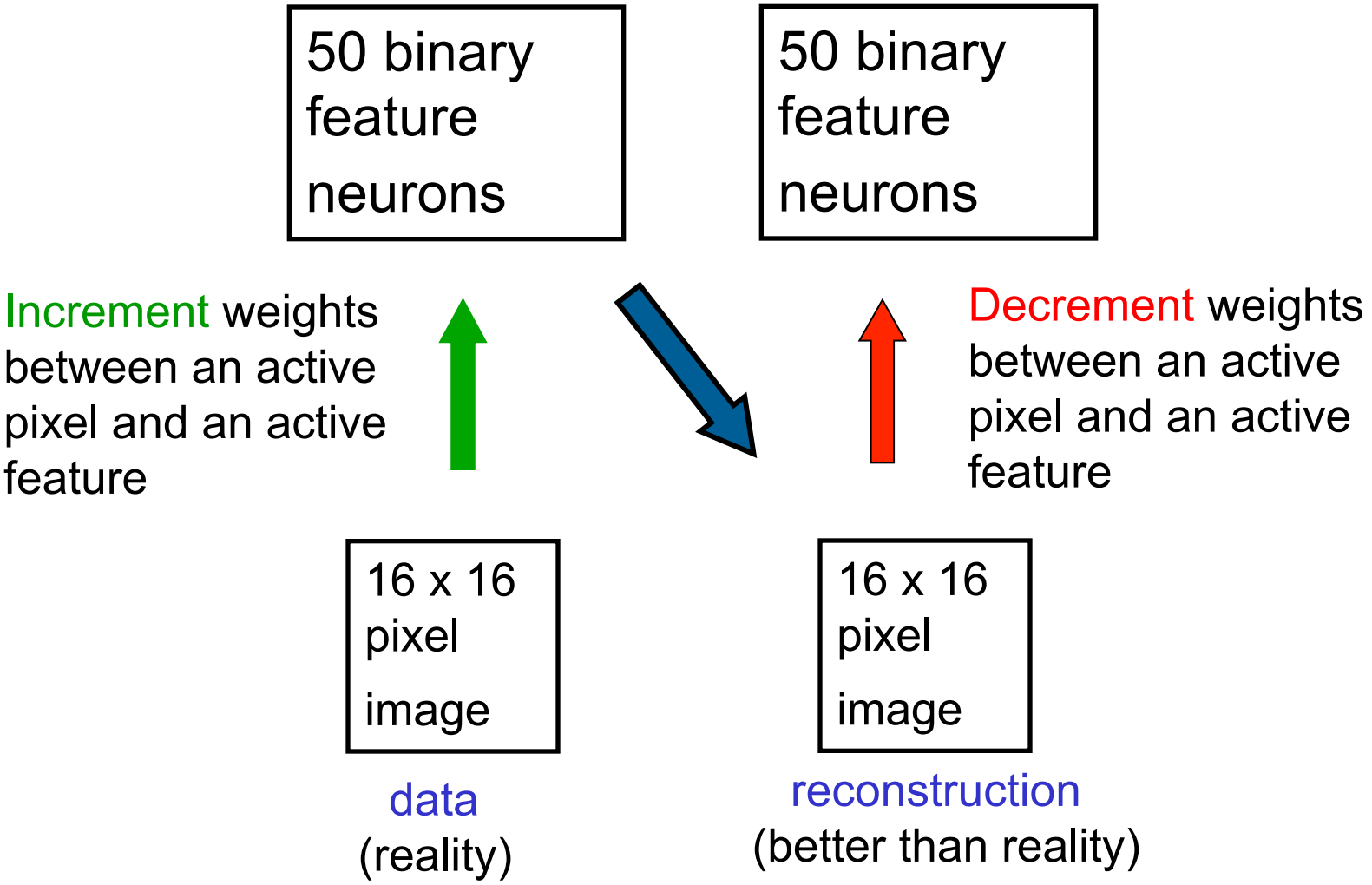
$$\Delta w_{ij} = \varepsilon \left(<v_i h_j>^0 - <v_i h_j>^1 \right)$$

This is not following the gradient of the log likelihood. But it works well.

It is approximately following the gradient of another objective function.

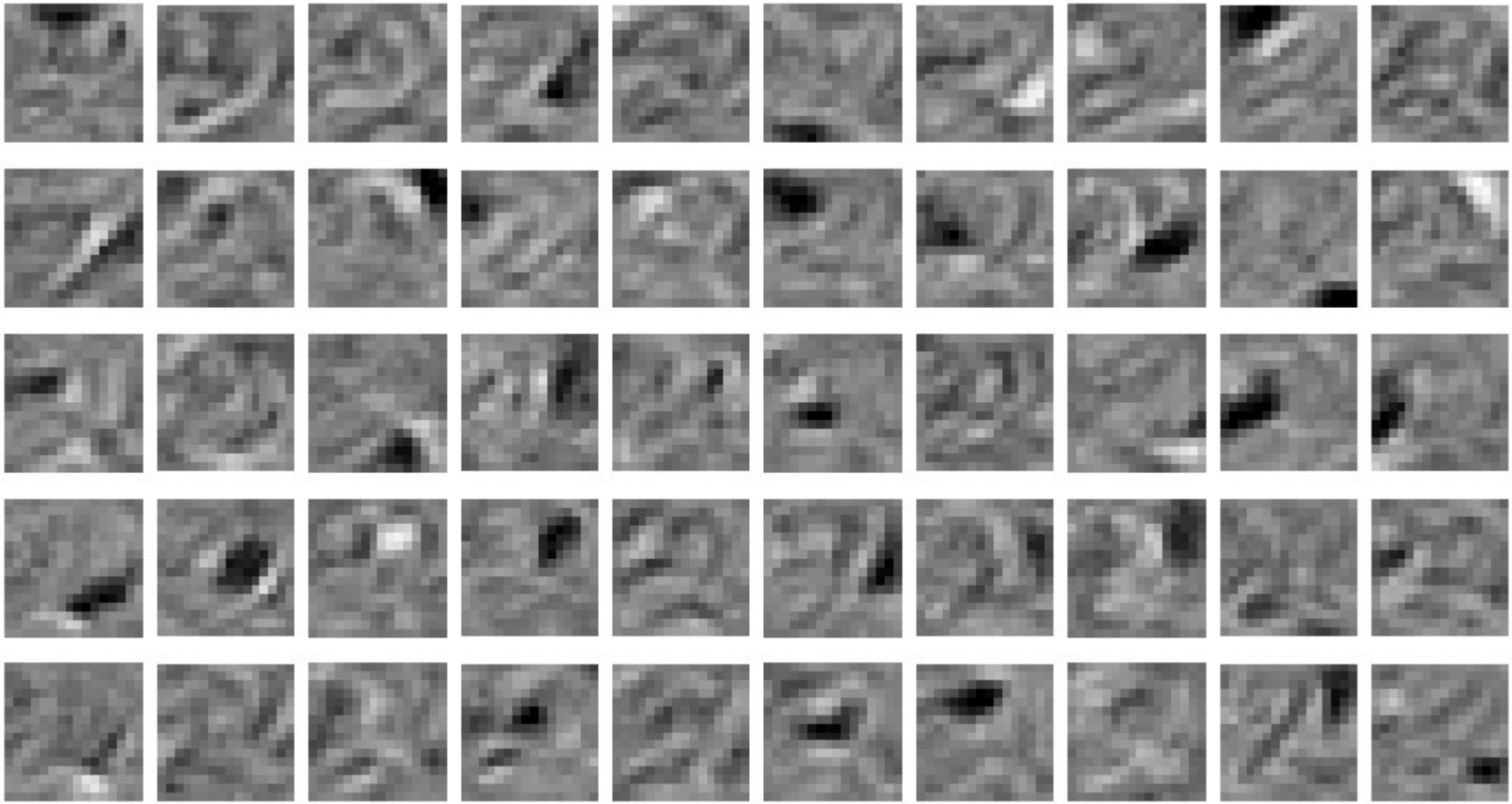
From Hinton 2007

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



From Hinton 2007

The final 50 x 256 weights



Each neuron grabs a different feature.

From Hinton 2007

Training a deep network

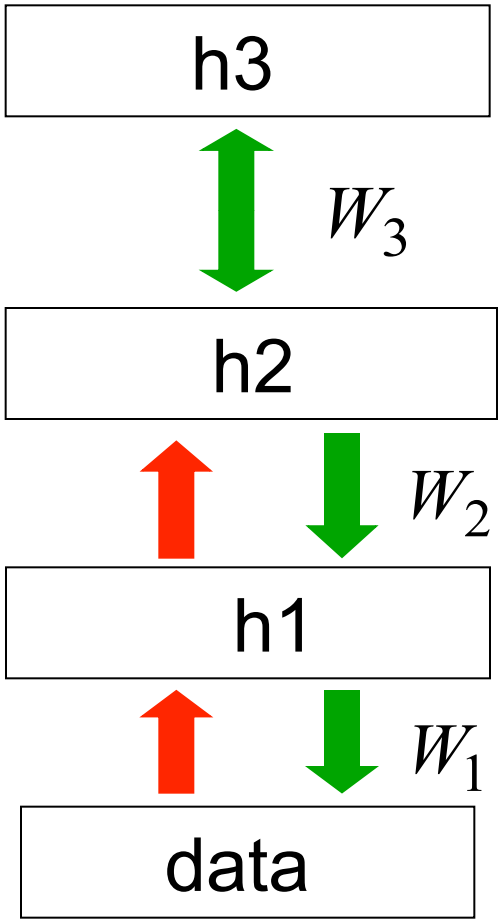
- First train a layer of features that receive input directly from the pixels.
- Then treat the activations of the trained features as if they were pixels and learn features of features in a second hidden layer.
- It can be proved that each time we add another layer of features we improve a variational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

From Hinton 2007

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.
  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.



From Hinton 2007

Why does greedy learning work?

The weights, W, in the bottom level RBM define p(v|h) and they also, indirectly, define p(h).

So we can express the RBM model as

$$p(v) = \sum_h p(h)\ p(v \mid h)$$

If we leave p(v|h) alone and improve p(h), we will improve p(v).

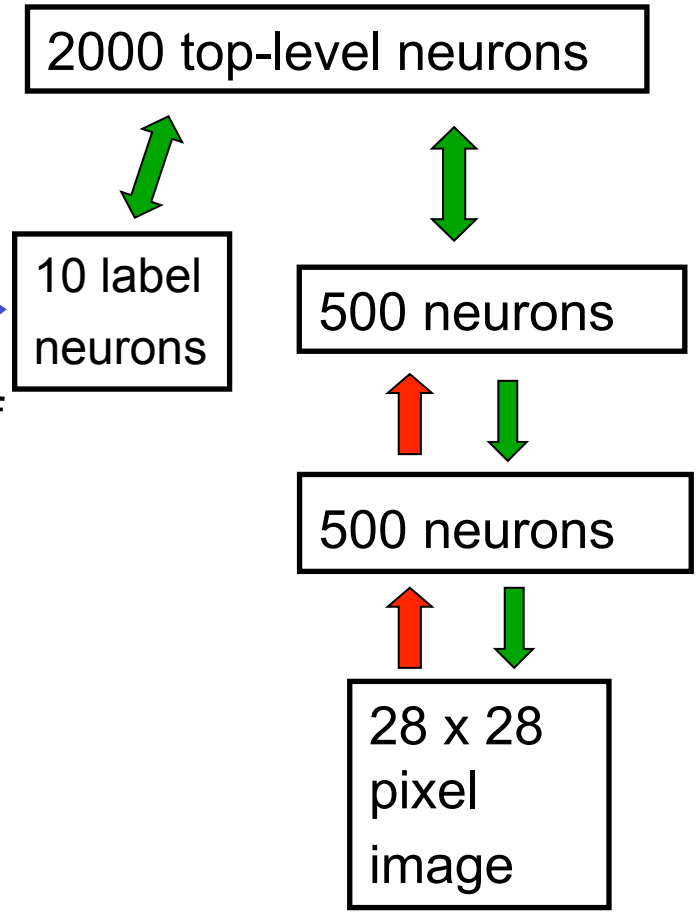
To improve p(h), we need it to be a better model of the aggregated posterior distribution over hidden vectors produced by applying W to the data.

From Hinton 2007

A neural 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.

From Hinton 2007

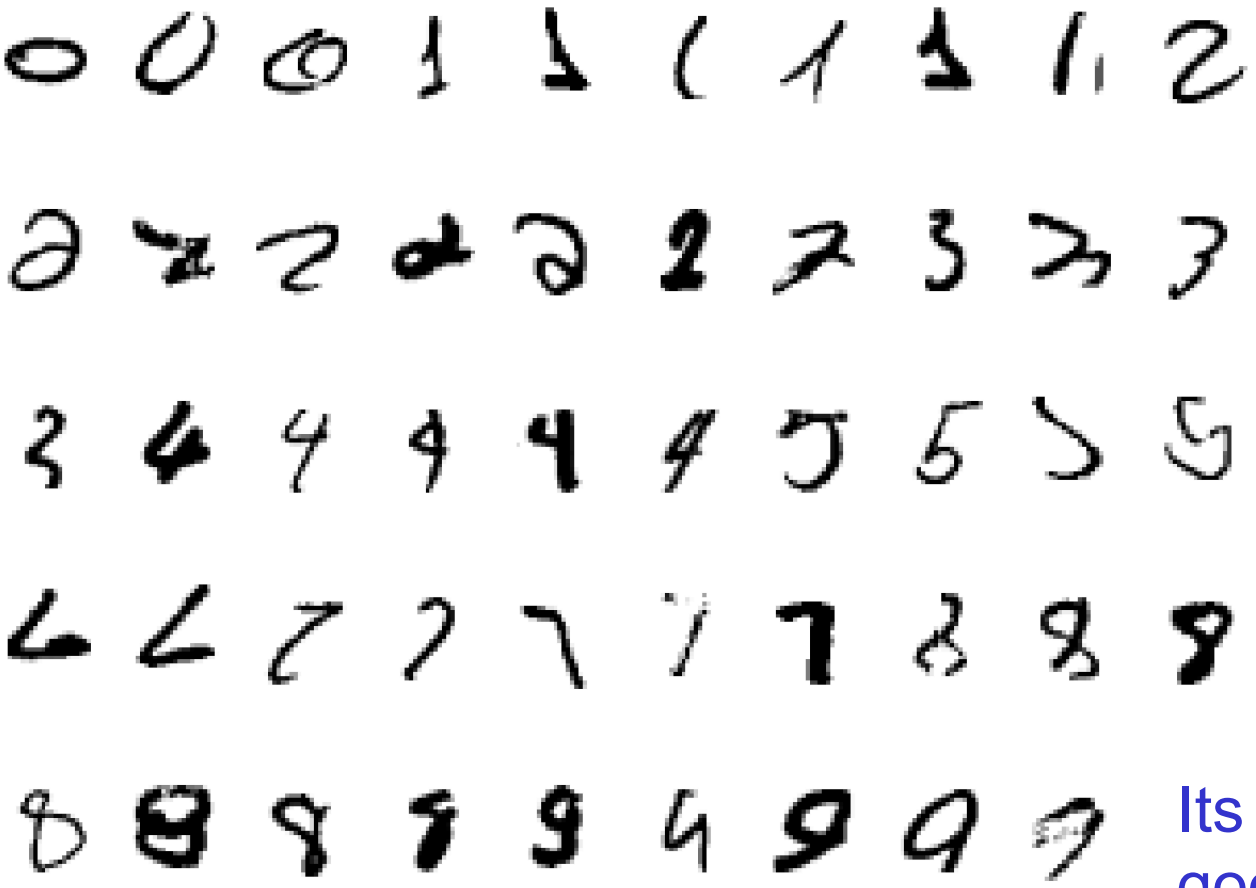
Fine-tuning with a contrastive divergence version of the "wake-sleep" algorithm

- 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
  - Use CD learning to improve the 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.

From Hinton 2007

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



From Hinton 2007

Its very good

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%

- 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.

From Hinton 2007

Summary

- Examined an application using RBMs as deep belief networks to recognize handwritten digits.
- Showed an efficient approximation—*contrastive divergence*—of the derivative of the log of the probability.
  - Involved calculating the average of <v,h> after the initial presentation, and
  - Calculating the average of <v,h> after the first reconstruction.
  - Using their difference multiplied by a learning parameter as the basis of a gradient ascent scheme.

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 11.4.1: RBM Mathematics, Insights and Getting Your Head Around It!

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What We've Covered So Far

- Probabilistic foundations of RBMs.
  - *Energy/Consensus* associated with a visible and hidden pair of vectors.
  - Probability of a node's states

Goal

- Raise the probability that a visible vector with be faithfully reconstructed when a 'hidden' vector is presented to the visible layer.

Question:

How do we train an RBM so that 'reconstructions' are likely to create a reasonable facsimile of the original data?

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

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).
- The energy of a joint configuration of the visible and hidden units determines its probability:

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

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

From Hinton 2007 (modified)

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

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

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

partition function

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

From Hinton 2007 (modified)

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

How Do We Train an RBM?

$$p(\mathbf{v}) = \frac{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}} \rightarrow \ln p(\mathbf{v}) = \ln\left(\frac{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}}\right)$$

$$= \ln \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} - \ln \sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}$$

$$\frac{\partial \ln p(\mathbf{v})}{\partial w_{ij}} = \underbrace{\frac{\partial}{\partial w_{ij}} \ln \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}}_{\text{A}} - \underbrace{\frac{\partial}{\partial w_{ij}} \ln \sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}}_{\text{B}}$$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Looking at Term A:

$$\frac{\partial}{\partial w_{ij}} \ln \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} = \frac{1}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} \cdot \frac{\partial}{\partial w_{ij}} \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}$$

$$= \frac{1}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} \cdot \sum_g \frac{\partial e^{-E(\mathbf{v},\mathbf{h}^g)}}{\partial w_{ij}}$$

$$= \frac{1}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} \cdot \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} \cdot \frac{\partial\left(-E(\mathbf{v},\mathbf{h}^g)\right)}{\partial w_{ij}}$$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Looking at Term A:

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

$$\frac{\partial}{\partial w_{ij}} \ln \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} = \frac{1}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} \cdot \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} \cdot \frac{\partial\left(-E(\mathbf{v},\mathbf{h}^g)\right)}{\partial w_{ij}}$$

$$= \frac{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)} v_i h_j^g}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} = \sum_g p(\mathbf{h}^g \mid \mathbf{v}) v_i h_j^g = \langle v_i \cdot h_j \rangle_{\mathbf{v}}$$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Where does that conditional probability come from?

$$\Pr\{A \mid B\} = \frac{\Pr\{A \cap B\}}{\Pr\{B\}}$$

$$p(\mathbf{h}^g \mid \mathbf{v}) = \frac{p(\mathbf{v},\mathbf{h}^g)}{p(\mathbf{v})} = \frac{\frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h}^g)}}{\frac{1}{Z} \sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} = \frac{e^{-E(\mathbf{v},\mathbf{h}^g)}}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}}$$

$$\frac{e^{-E(\mathbf{v},\mathbf{h}^g)}}{\sum_g e^{-E(\mathbf{v},\mathbf{h}^g)}} = \frac{p\left(\mathbf{v},\mathbf{h}^g\right)}{p\left(\mathbf{v}\right)} = p(\mathbf{h}^g \mid \mathbf{v})$$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Looking at Term B:

$$\frac{\partial}{\partial w_{ij}} \ln \sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)} = \frac{1}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}} \cdot \frac{\partial}{\partial w_{ij}} \sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}$$

$$= \frac{1}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}} \cdot \sum_{u,g} \frac{\partial e^{-E(\mathbf{v}^u,\mathbf{h}^g)}}{\partial w_{ij}}$$

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Looking at Term B:

$$\frac{\partial}{\partial w_{ij}} \ln \sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)} = \frac{\sum_{u,g} \frac{\partial e^{-E(\mathbf{v}^u,\mathbf{h}^g)}}{\partial w_{ij}}}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}}$$

$$= \frac{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)} \frac{\partial\left(-E(\mathbf{v}^u,\mathbf{h}^g)\right)}{\partial w_{ij}}}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}}$$

$$= \frac{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)} v_i^u h_j^g}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}} = \sum_{u,g} p(\mathbf{v}^u,\mathbf{h}^g) v_i^u h_j^g = \langle v_i \cdot h_j \rangle_{\mathbf{vh}}$$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Basis of *Contrastive Divergence*

$$\frac{\partial \ln p(\mathbf{v})}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \ln \sum_{g} e^{-E(\mathbf{v},\mathbf{h}^g)} - \frac{\partial}{\partial w_{ij}} \ln \sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}$$
$$= \langle v_i \cdot h_j \rangle_{\mathbf{v}} - \langle v_i \cdot h_j \rangle_{\mathbf{vh}}$$

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Using Contrastive Divergence

- Use the derivative to perform *stochastic* gradient ascent!

$$\frac{\partial \ln p(\mathbf{v})}{\partial w_{ij}} \quad \propto \quad \Delta w_{ij} = \eta\left(\langle v_i \cdot h_j \rangle_{\mathbf{v}} - \langle v_i \cdot h_j \rangle_{\mathbf{vh}}\right)$$

But why?

Engineering for Professionals

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Introduction to Neural Networks

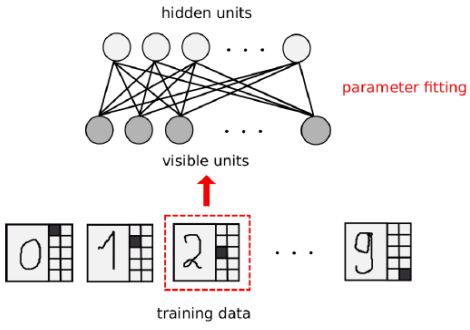
Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 11.4.2: RBM Mathematics, Insights and Getting Your Head Around It!

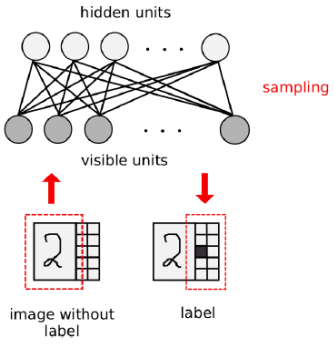
JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Some Motivation

learning with labels



classification



From Fischer, Asja, "Training Restricted Boltzmann Machines", Doctoral Thesis, Faculty of Science, University of Copenhagen, 2014 at p.19

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Our Goal

- Strengthen the probability of reconstructed visible vectors so that they correspond to their probability of occurring in a training set of data.

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

The Energy/Probability Relationships

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

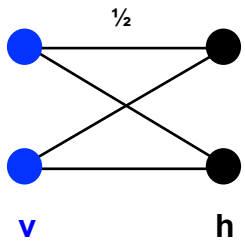
$$\Pr(\mathbf{v},\mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{u,g} e^{-E(\mathbf{v}^u,\mathbf{h}^g)}}$$

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

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

A Numerical Example



We are going to increase the probability of the vector v = [0, 1].

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So What is the Generative Model?

v	h	Joint Probability
0 0	0 0	
0 0	0 1	
0 0	1 0	
0 0	1 1	
0 1	0 0	
0 1	0 1	
0 1	1 0	
0 1	1 1	

v	h	Joint Probability
1 0	0 0	
1 0	0 1	
1 0	1 0	
1 0	1 1	
1 1	0 0	
1 1	0 1	
1 1	1 0	
1 1	1 1	

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So What is the Generative Model?

	v1	v2	h1	h2	E	e^{-E}	Probability
1	0	0	0	0	0	1	0.031390208
2	0	0	0	1	0	1	0.031390208
3	0	0	1	0	0	1	0.031390208
4	0	0	1	1	0	1	0.031390208
5	0	1	0	0	0	1	0.031390208
6	0	1	0	1	-0.5	1.648721271	0.051753703
7	0	1	1	0	-0.5	1.648721271	0.051753703
8	0	1	1	1	-1	2.718281828	0.085327431
9	1	0	0	0	0	1	0.031390208
10	1	0	0	1	-0.5	1.648721271	0.051753703
11	1	0	1	0	-0.5	1.648721271	0.051753703
12	1	0	1	1	-1	2.718281828	0.085327431
13	1	1	0	0	0	1	0.031390208
14	1	1	0	1	-1	2.718281828	0.085327431
15	1	1	1	0	-1	2.718281828	0.085327431
16	1	1	1	1	-2	7.389056099	0.231944006
						31.8570685	1

All energy and probability values are based solely on the weights and biases!

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What Does This Tell Us?

- The different configurations will occur with the indicated probability.
- How?
  - Present (activate) a initial visible vector onto the visible nodes (set their states).
  - Stochastically assign states to the hidden vector nodes.
  - Let the hidden vector nodes stochastically influence the assignment of states to the visible nodes, and so on.

If we do this back and forth for a very large number of cycles and count the occurances of the different vectors (configurations), they will occur with the frequency from the preceding table!

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What is the Frequency of Occurrence of Visible Vector [0,1]?

From the table, we can calculate the marginal probability.

	v1	v2	h1	h2	E	e^{-E}	Probability
5	0	1	0	0	0	1	0.031390208
6	0	1	0	1	-0.5	1.648721271	0.051753703
7	0	1	1	0	-0.5	1.648721271	0.051753703
8	0	1	1	1	-1	2.718281828	0.085327431

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

This sums to about 0.22022505.

All based on the energy values and Boltzmann distribution function.

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Let's See How Stochastic Update Functions are consistent with the table values.

- Thus, given a visible vector, the hidden vectors are assigned states with certain probabilities based on the activity function.
- Remember?

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 11.4.3: RBM Mathematics, Insights and Getting Your Head Around It!

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What are the probabilities of h given v?

$$\Pr\left\{h_1 = 1 \middle| \mathbf{v}\right\} = \frac{1}{1 + e^{-S_{h_1}/T}}$$

If S = 0, then this probability is 1/2.

Can you determine the first row of the table?

$$\Pr\left\{\mathbf{v}, \mathbf{h}\right\} = \Pr\left\{\mathbf{h} \middle| \mathbf{v}\right\} \times \Pr\left\{\mathbf{v}\right\}$$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What are the probabilities of h given v?

$$\Pr\left\{h_1 = 1 \middle| \mathbf{v}\right\} = \frac{1}{1+e^{-S_{h_1}/T}}$$

Let's assume for now that **v** = [0, 0]

So, $$S_{h_1} = \sum_i w_i v_i + \theta_i$$

$$= \tfrac{1}{2}\cdot 0 + \tfrac{1}{2}\cdot 0 + 0 = 0$$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

The Generative Model

Since $S_{h1} = 0$, then

$$\Pr\left\{h_1 = 1 \middle| \mathbf{v}\right\} = \frac{1}{1+e^{-S_{h_1}/T}} = \frac{1}{1+1} = \frac{1}{2}$$

But for the first row of the table, we want to know the probability of $h_1 = 0$ (not 1). Also, we want to determine the probability of the **vector h given the vector $\mathbf{v}_1$**

Probability of **h**, given that $v_1 = [0, 0]$, is ¼.

This is because h_1 is independent of h_2.

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So What is the Generative Model?

	v	h	Joint Probability
P_{v1}	0 0	0 0	$P_{v1} / 4$
	0 0	0 1	$P_{v1} / 4$
	0 0	1 0	$P_{v1} / 4$
	0 0	1 1	$P_{v1} / 4$
P_{v2}	0 1	0 0	
	0 1	0 1	
	0 1	1 0	
	0 1	1 1	

	v	h	Joint Probability
P_{v3}	1 0	0 0	
	1 0	0 1	
	1 0	1 0	
	1 0	1 1	
P_{v4}	1 1	0 0	
	1 1	0 1	
	1 1	1 0	
	1 1	1 1	

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What are the probabilities of h given v?

$$\Pr\left\{h_1 = 1 \middle| \mathbf{v}_2\right\} = \frac{1}{1 + e^{-S_{h_1}/T}}$$

Now, **v** = [0, 1]

So, again, $S_{h_1} = \sum_i w_i v_i + \theta_i$

$$= \tfrac{1}{2} \cdot 0 + \tfrac{1}{2} \cdot 1 = \tfrac{1}{2}$$

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What are the probabilities of h given v?

$$\Pr\left\{h_1 = 1 \middle| \mathbf{v}_2\right\} = \frac{1}{1+e^{-1/2}} = 0.622459331$$

So, given that **v** = [0, 1], Pr { **h** = [0,0] | $\mathbf{v}_2$ } = 0.3775 × 0.3775 = 0.1425.

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So What is the Generative Model?

	v	h	Joint Probability
P_{v1}	0 0	0 0	P_{v1} / 4
	0 0	0 1	P_{v1} / 4
	0 0	1 0	P_{v1} / 4
	0 0	1 1	P_{v1} / 4
P_{v2}	0 1	0 0	P_{v2} • 0.1425
	0 1	0 1	
	0 1	1 0	
	0 1	1 1	

	v	h	Joint Probability
P_{v3}	1 0	0 0	
	1 0	0 1	
	1 0	1 0	
	1 0	1 1	
P_{v4}	1 1	0 0	
	1 1	0 1	
	1 1	1 0	
	1 1	1 1	

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What are the probabilities of h given v?

$$\Pr\left\{h_1 = 1 \middle| \mathbf{v}_2\right\} = \frac{1}{1+e^{-1/2}} = 0.622459331$$

So, given that **v** = [0, 1], Pr { **h =** [0,1] | $\mathbf{v}_2$ } = 0.3775 × 0.6224 = 0.2350.

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So What is the Generative Model?

	v	h	Joint Probability
P_{v1}	0 0	0 0	P_{v1} / 4
	0 0	0 1	P_{v1} / 4
	0 0	1 0	P_{v1} / 4
	0 0	1 1	P_{v1} / 4
P_{v2}	0 1	0 0	P_{v2} • 0.1425
	0 1	0 1	P_{v2} • 0.2350
	0 1	1 0	P_{v2} • 0.2350
	0 1	1 1	

	v	h	Joint Probability
P_{v3}	1 0	0 0	
	1 0	0 1	
	1 0	1 0	
	1 0	1 1	
P_{v4}	1 1	0 0	
	1 1	0 1	
	1 1	1 0	
	1 1	1 1	

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What are the probabilities of h given v?

$$\Pr\left\{h_1 = 1 \middle| \mathbf{v}_2\right\} = \frac{1}{1+e^{-1/2}} = 0.622459331$$

So, given that **v** = [0, 1], Pr { **h =** [1,1] | $\mathbf{v}_2$ } = 0.6224 × 0.6224 = 0.3874.

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So What is the Generative Model?

	v	h	Joint Probability
P_{v1}	0 0	0 0	P_{v1} / 4
	0 0	0 1	P_{v1} / 4
	0 0	1 0	P_{v1} / 4
	0 0	1 1	P_{v1} / 4
P_{v2}	0 1	0 0	P_{v2} • 0.1425
	0 1	0 1	P_{v2} • 0.2350
	0 1	1 0	P_{v2} • 0.2350
	0 1	1 1	P_{v2} • 0.3874

	v	h	Joint Probability
P_{v3}	1 0	0 0	
	1 0	0 1	
	1 0	1 0	
	1 0	1 1	
P_{v4}	1 1	0 0	
	1 1	0 1	
	1 1	1 0	
	1 1	1 1	

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So What is the Generative Model?

	v	h	Joint Probability
P_{v1}	0 0	0 0	P_{v1} / 4
	0 0	0 1	P_{v1} / 4
	0 0	1 0	P_{v1} / 4
	0 0	1 1	P_{v1} / 4
P_{v2}	0 1	0 0	P_{v2} • 0.1425
	0 1	0 1	P_{v2} • 0.2350
	0 1	1 0	P_{v2} • 0.2350
	0 1	1 1	P_{v2} • 0.3874

	v	h	Joint Probability
P_{v3}	1 0	0 0	P_{v2} • 0.1425
	1 0	0 1	P_{v2} • 0.2350
	1 0	1 0	P_{v2} • 0.2350
	1 0	1 1	P_{v2} • 0.3874
P_{v4}	1 1	0 0	
	1 1	0 1	
	1 1	1 0	
	1 1	1 1	

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

What are the probabilities of h given v?

$$S_{v_4} = \tfrac{1}{2}\cdot 1 + \tfrac{1}{2}\cdot 1 = 1$$

$$\Pr\left\{h_1 = 1 \middle| \mathbf{v}_4\right\} = \frac{1}{1+e^{-1}} = 0.7310$$

So, given that **v** = [1, 1], Pr { **h** = [0,0] | $\mathbf{v}_2$ } = 0.2689 × 0.2689 = 0.0723.

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So What is the Generative Model?

	v	h	Joint Probability
P_{v1}	0 0	0 0	P_{v1} / 4
	0 0	0 1	P_{v1} / 4
	0 0	1 0	P_{v1} / 4
	0 0	1 1	P_{v1} / 4
P_{v2}	0 1	0 0	P_{v2} • 0.1425
	0 1	0 1	P_{v2} • 0.2350
	0 1	1 0	P_{v2} • 0.2350
	0 1	1 1	P_{v2} • 0.3874

	v	h	Joint Probability
P_{v3}	1 0	0 0	P_{v3} • 0.1425
	1 0	0 1	P_{v3} • 0.2350
	1 0	1 0	P_{v3} • 0.2350
	1 0	1 1	P_{v3} • 0.3874
P_{v4}	1 1	0 0	P_{v4} • 0.0723
	1 1	0 1	P_{v4} • 0.1966
	1 1	1 0	P_{v4} • 0.1966
	1 1	1 1	P_{v4} • 0.5344

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

So, based on stochastic updating…

- What is the probability of v_2?
- Let's look at the 5th row of the preceding slide.

P_{v2} • 0.142536957 = 0.031390208

From the spreadsheet For the probability of the configuration [0,1], [0, 0]

Solving for P_{v2} = 0.220225047!

As expected, stochastic updating is consistent with the energy/ probability functions defined earlier ---- that was the basis of stochastic updating afterall!

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 11.4.4: RBM Mathematics, Insights and Getting Your Head Around It!

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Let's Increase the Probability that v = [0, 1] occurs

$$\Delta w_{ij} = \eta\left(\langle v_i \cdot h_j \rangle_{\mathbf{v}} - \langle v_i \cdot h_j \rangle_{\mathbf{vh}}\right)$$

	v1	v2	h1	h2	E	e^{-E}	Probability
5	0	1	0	0	0	1	0.031390208
6	0	1	0	1	-0.5	1.648721271	0.051753703
7	0	1	1	0	-0.5	1.648721271	0.051753703
8	0	1	1	1	-1	2.718281828	0.085327431

Recall that the first term is ... $= \sum_g p(\mathbf{h}^g \mid \mathbf{v}) v_i h_j^g = \langle v_i \cdot h_j \rangle_{\mathbf{v}}$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Let's Do Some Calculations

$$\langle v_i \cdot h_j \rangle_{\mathbf{v}} = \sum_g p(\mathbf{h}^g \,|\, \mathbf{v}) v_i h_j^g = \sum_g \left(\frac{p(\mathbf{h}^g, \mathbf{v})}{p(\mathbf{v})} \right) v_i h_j^g$$

5	0	1	0	0	0	1	0.031390208
6	0	1	0	1	−0.5	1.648721271	0.051753703
7	0	1	1	0	−0.5	1.648721271	0.051753703
8	0	1	1	1	−1	2.718281828	0.085327431

So for Δw_{11}, this term is:

$$\langle v_1 \cdot h_1 \rangle_{\mathbf{v}} = \left(\frac{0.0313}{0.2202} \right) \bullet 0 \bullet 0 + \left(\frac{0.0517}{0.2202} \right) \bullet 0 \bullet 0 + \left(\frac{0.0517}{0.2202} \right) \bullet 0 \bullet 1 + \left(\frac{0.0853}{0.2202} \right) \bullet 0 \bullet 1 = 0$$

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Now for the Second Term

Recall that $\sum_{u,g} p(\mathbf{v}^u, \mathbf{h}^g) v_i^u h_j^g = \langle v_i \cdot h_j \rangle_{\mathbf{vh}}$

	v1	v2	h1	h2	E	e^−E	Probability
1	0	0	0	0	0	1	0.031390208
2	0	0	0	1	0	1	0.031390208
3	0	0	1	0	0	1	0.031390208
4	0	0	1	1	0	1	0.031390208
5	0	1	0	0	0	1	0.031390208
6	0	1	0	1	−0.5	1.648721271	0.051753703
7	0	1	1	0	−0.5	1.648721271	0.051753703
8	0	1	1	1	−1	2.718281828	0.085327431
9	1	0	0	0	0	1	0.031390208
10	1	0	0	1	−0.5	1.648721271	0.051753703
11	1	0	1	0	−0.5	1.648721271	0.051753703
12	1	0	1	1	−1	2.718281828	0.085327431
13	1	1	0	0	0	1	0.031390208
14	1	1	0	1	−1	2.718281828	0.085327431
15	1	1	1	0	−1	2.718281828	0.085327431
16	1	1	1	1	−2	7.389056099	0.231944006
						31.8570685	1

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

The Second Term for v_1 h_1

So, $\sum_{u,g} p(\mathbf{v}^u, \mathbf{h}^g) v_1^u h_1^g = \langle v_1 \cdot h_1 \rangle_{\mathbf{vh}} = 0.45435257$

$$\begin{aligned}\Delta w_{11} &= \eta\left(\langle v_1 \cdot h_1 \rangle_{\mathbf{v}} - \langle v_1 \cdot h_1 \rangle_{\mathbf{vh}}\right) \\ &= 0.1\left(0 - 0.45435257\right) \\ &= -0.045435257\end{aligned}$$

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Doing the Same Calculations for all the other weights, we get …

$$\langle v_1 \cdot h_1 \rangle_{\mathbf{v}} = \left(\frac{0.0313}{0.2202}\right)\bullet 0 \bullet 0 + \left(\frac{0.0517}{0.2202}\right)\bullet 0 \bullet 0 + \left(\frac{0.0517}{0.2202}\right)\bullet 0 \bullet 0 + \left(\frac{0.0853}{0.2202}\right)\bullet 0 \bullet 0 = 0$$

$$\langle v_1 \cdot h_2 \rangle_{\mathbf{v}} = \left(\frac{0.0313}{0.2202}\right)\bullet 0 \bullet 0 + \left(\frac{0.0517}{0.2202}\right)\bullet 0 \bullet 1 + \left(\frac{0.0517}{0.2202}\right)\bullet 0 \bullet 0 + \left(\frac{0.0853}{0.2202}\right)\bullet 0 \bullet 1 = 0$$

$$\langle v_2 \cdot h_1 \rangle_{\mathbf{v}} = \left(\frac{0.0313}{0.2202}\right)\bullet 1 \bullet 0 + \left(\frac{0.0517}{0.2202}\right)\bullet 1 \bullet 0 + \left(\frac{0.0517}{0.2202}\right)\bullet 1 \bullet 1 + \left(\frac{0.0853}{0.2202}\right)\bullet 1 \bullet 1 = 0.622459331$$

$$\langle v_2 \cdot h_2 \rangle_{\mathbf{v}} = \left(\frac{0.0313}{0.2202}\right)\bullet 1 \bullet 0 + \left(\frac{0.0517}{0.2202}\right)\bullet 1 \bullet 1 + \left(\frac{0.0517}{0.2202}\right)\bullet 1 \bullet 0 + \left(\frac{0.0853}{0.2202}\right)\bullet 1 \bullet 1 = 0.622459331$$

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Now for all the Second Terms

$$\langle v_1 \cdot h_1 \rangle_{\mathbf{vh}} = 0.0517 + 0.0853 + 0.0853 + .2319 = 0.454352573$$
$$\langle v_1 \cdot h_2 \rangle_{\mathbf{vh}} = 0.0517 + 0.0853 + 0.0853 + .2319 = 0.454352573$$
$$\langle v_2 \cdot h_1 \rangle_{\mathbf{vh}} = 0.0517 + 0.0853 + 0.0853 + .2319 = 0.454352573$$
$$\langle v_2 \cdot h_2 \rangle_{\mathbf{vh}} = 0.0517 + 0.0853 + 0.0853 + .2319 = 0.454352573$$

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Updated Weights

$$\Delta w_{11} = \eta\left(\langle v_1 \cdot h_1 \rangle_{\mathbf{v}} - \langle v_1 \cdot h_1 \rangle_{\mathbf{vh}}\right) = 0.1\left(0 - 0.45435257\right) = -0.045435257$$
$$\Delta w_{12} = 0.1\left(0 - 0.45435257\right) = -0.045435257$$
$$\Delta w_{21} = 0.1\left(0.622459331 - 0.45435257\right) = 0.016810676$$
$$\Delta w_{22} = 0.1\left(0.622459331 - 0.45435257\right) = 0.016810676$$

Engineering for Professionals

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

The Updated Configurations

	v1	v2	h1	h2	E	e^-E	Probability
1	0	0	0	0	0	1	0.032197018
2	0	0	0	1	0	1	0.032197018
3	0	0	1	0	0	1	0.032197018
4	0	0	1	1	0	1	0.032197018
5	0	1	0	0	0	1	0.032197018
6	0	1	0	1	-0.516810676	1.676671664	0.053983827
7	0	1	1	0	-0.516810676	1.676671664	0.053983827
8	0	1	1	1	-1.033621352	2.811227869	0.090513154
9	1	0	0	0	0	1	0.032197018
10	1	0	0	1	-0.454564743	1.575487492	0.050725999
11	1	0	1	0	-0.454564743	1.575487492	0.050725999
12	1	0	1	1	-0.909129486	2.482160837	0.079918176
13	1	1	0	0	0	1	0.032197018
14	1	1	0	1	-0.971375419	2.641575235	0.085050845
15	1	1	1	0	-0.971375419	2.641575235	0.085050845
16	1	1	1	1	-1.942750838	6.97791972	0.224668205
						31.05877721	1

So now the total probability Pr{v=[0,1]} = 0.230677826

If eta = 1, the probability goes up to 0.332961042!

Recall, it was 0.22022505

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Hinton's Approximation in vector form

1. Take a training sample v, compute the probabilities of the hidden units and sample a hidden activation vector h from this probability distribution.
2. Compute the outer product of **v** and **h** and call this the positive gradient.
3. From **h**, sample a reconstruction **v'** of the visible units, then resample the hidden activations **h'** from this. (Gibbs sampling step)
4. Compute the outer product of **v'** and **h'** and call this the negative gradient.
5. Let the update to the weight matrix *W* be the positive gradient minus the negative gradient, times some learning rate: $\Delta W = \epsilon (\mathbf{v}\mathbf{h}^{\mathsf{T}} - \mathbf{v'}\mathbf{h'}^{\mathsf{T}})$.
6. Update the biases *a* and *b* analogously: $\Delta a = \epsilon(\mathbf{v} - \mathbf{v'})$, $\Delta b = \epsilon (\mathbf{h} - \mathbf{h'})$.

From Wikipedia.

Summary

- Showed the derivative of the log probability with respect to weights
- This can serve as the basis of a gradient ascent method for increasing the probability of the reconstructed vector v.

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 12.1: The Hamming Network

What We've Covered So Far

- Examined approaches to unsupervised learning methods
  - Recurrent networks: Hopfield networks, Boltzmann Machines
  - BAMs and RBMs
  - Data itself 'trains' the network

In This Module We Will Cover:

- Competitive Learning
  - The Hamming Net
  - The MAXNET algorithm
  - Self-Organizing Maps
  - Data values ‘compete’ in some fashion

A Basic Problem in Communications

- In Hopfield Net, the network converged to an exemplar from a ‘noisy’ version of the exemplar.
- Let’s try a different approach based on ‘classification’.

What to do with a noisy exemplar?

- Compare it to all possible patterns and pick the one it is ‘closest’ to.
- For binary information, we use the ‘Hamming distance’.
- Recall the notion of distance from the material on metric spaces.

Hamming Distance

- For two binary strings, the Hamming Distance is the number of corresponding bits (vector elements) that are different.
- A Hamming Distance of zero implies the two strings are the same.
- Example:
  - x = {0 0 1 1 0 1 0 1}
  - x′ = {0 1 1 1 0 1 0 0}

$$(x, y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{otherwise} \end{cases}$$

- HD = $\sum_{i=1}(x_i, x'_i)$ = ?

Using the Hamming Distance

- Suppose we want a *larger* number to correspond to better matching?
  - i.e., score ∝ similarity
- Define:

$$H(\mathbf{x}, \mathbf{x}') = N - \sum_{i=1}^{N} (x_i, x_i')$$

- We could use this to determine the nearest exemplar to determine the best match.

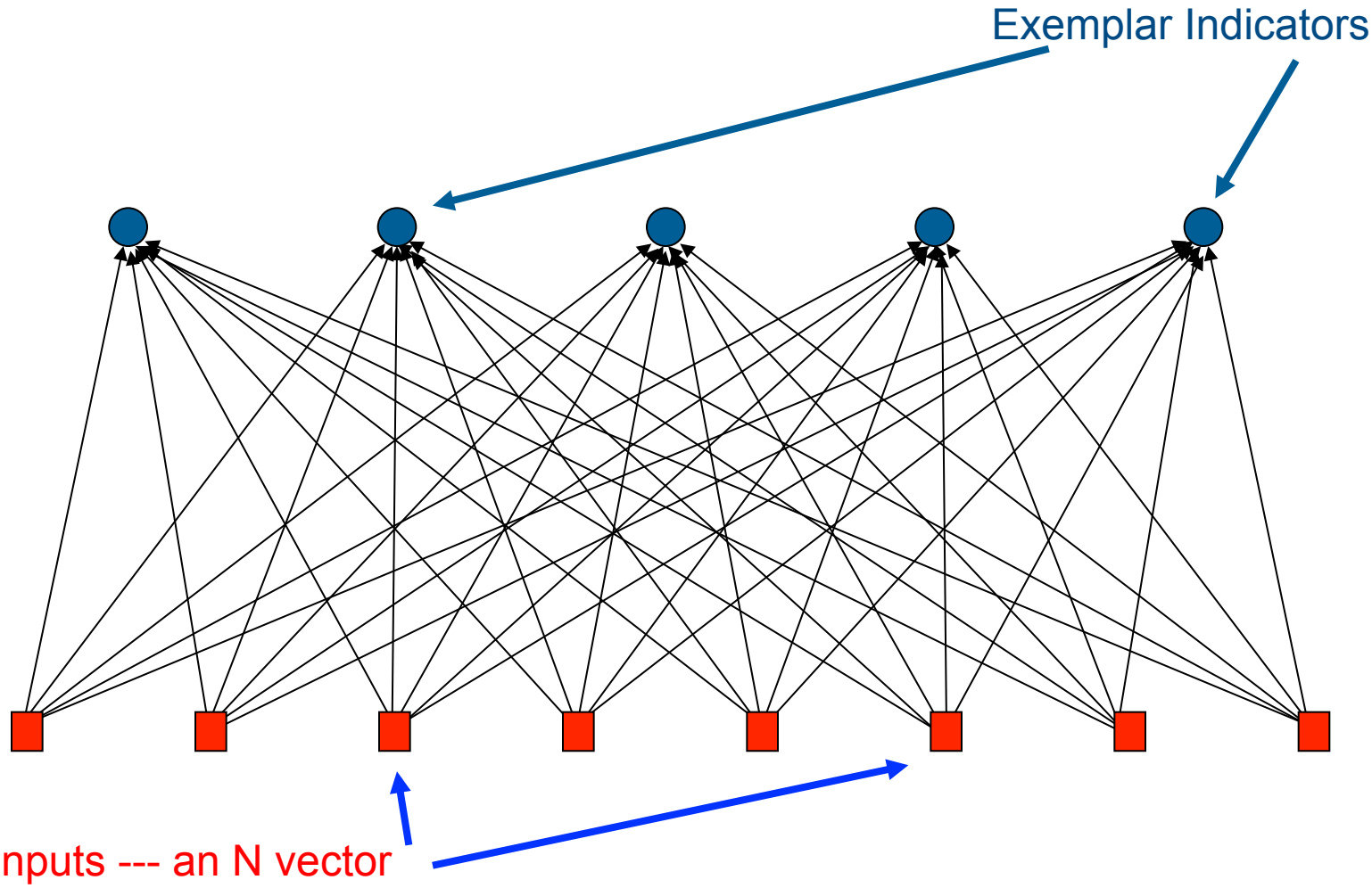
The Hamming Network

- This is more interesting.
- Let a network decide which exemplar is the best match to a network input.
- Let a node designate a particular exemplar which ‘fires’ a particular node when the network is presented with a noisy input that is closest to it.
- Use a competitive learning approach.

The Hamming Network

- Suppose we have M exemplars of vectors each with N elements.
- We want only one of the M neurons to fire corresponding to the exemplar closest to the noisy input.

The Hamming Network



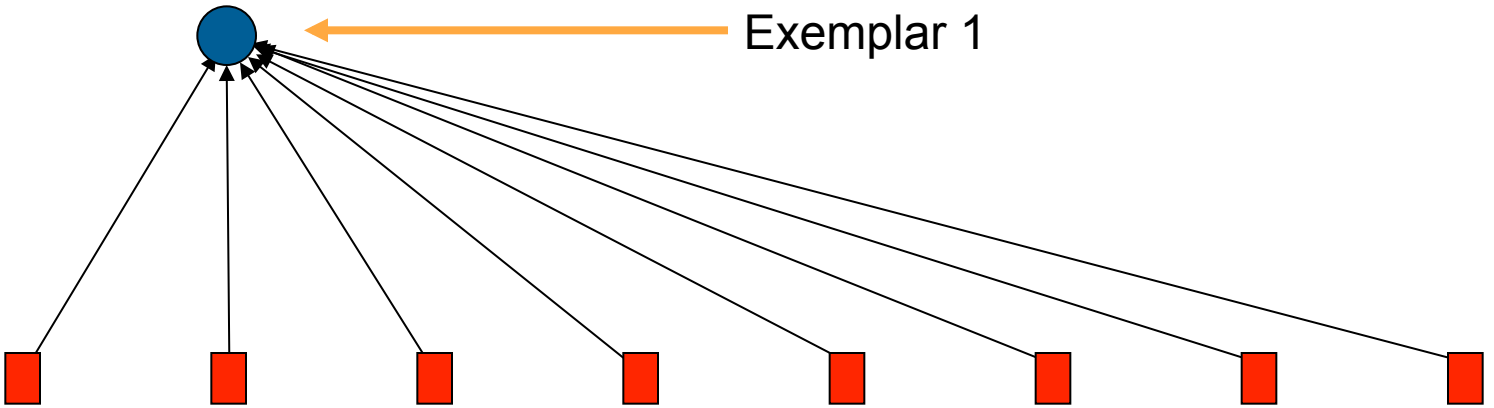
The Hamming Network

- Each output has N weight connections --- one for each input.
- We need to set the weights so that one exemplar that an input is closest to will have the highest value of H.

Setting Weights

5 Exemplars:

1: 1 0 1 0 1 0 1 0
2: 0 0 0 0 1 1 1 1
3: 1 1 1 1 0 0 0 0
4: 0 0 1 1 1 1 0 0
5: 1 1 0 0 0 0 1 1



The Hamming Network

- Set weights w_{ij} for input i to exemplar j according to the exemplar pattern. One approach is …

$$w_{ij} = \frac{x_i^j}{2}, \quad \theta_j = \frac{N}{2}$$

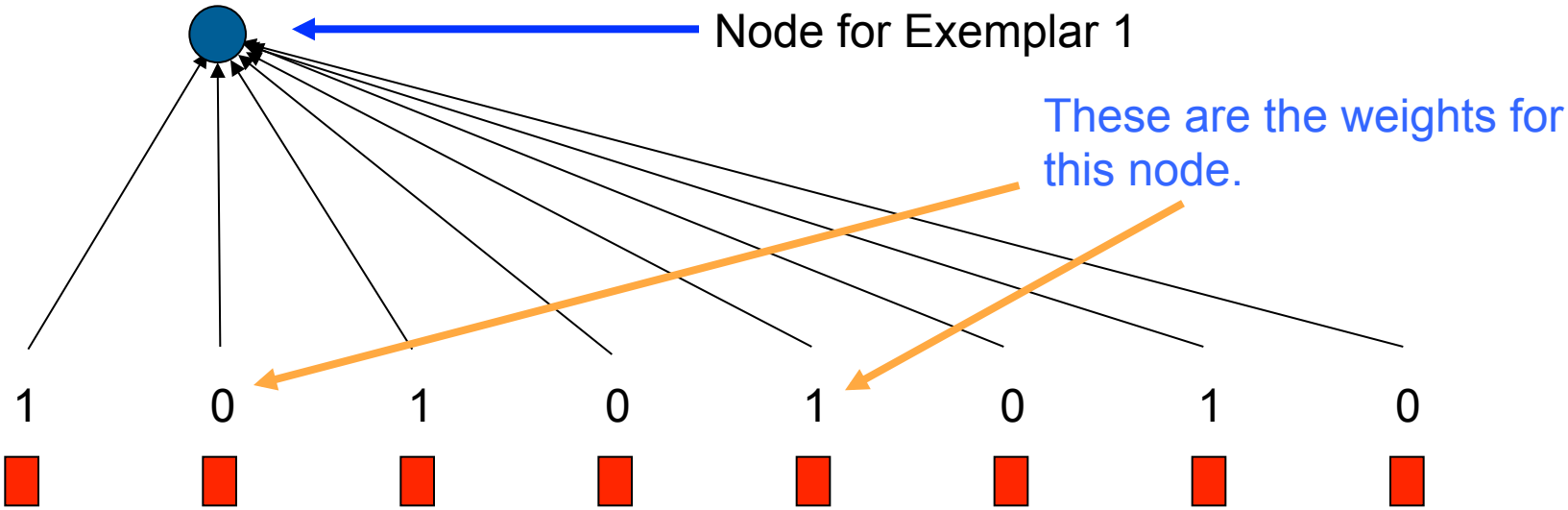
where x_i^j is the i^{th} element of exemplar j.

Another approach is to simply use the exemplar pattern itself and then calculate the value of H.

Setting Weights

5 Exemplars:

1: 1 0 1 0 1 0 1 0
2: 0 0 0 0 1 1 1 1
3: 1 1 1 1 0 0 0 0
4: 0 0 1 1 1 1 0 0
5: 1 1 0 0 0 0 1 1



Using the Weights

- Each of the output nodes calculates the value of H based on its weight vector and the input vector.
- The node with the greatest value is the one most similar to the input!
- So ….?

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 12.2: Competitive Learning--MAXNET

In This Module We Will Cover:

- Competitive Learning
  - The Hamming Net
  - The MAXNET algorithm
  - Self-Organizing Maps
  - Data values ‘compete’ in some fashion

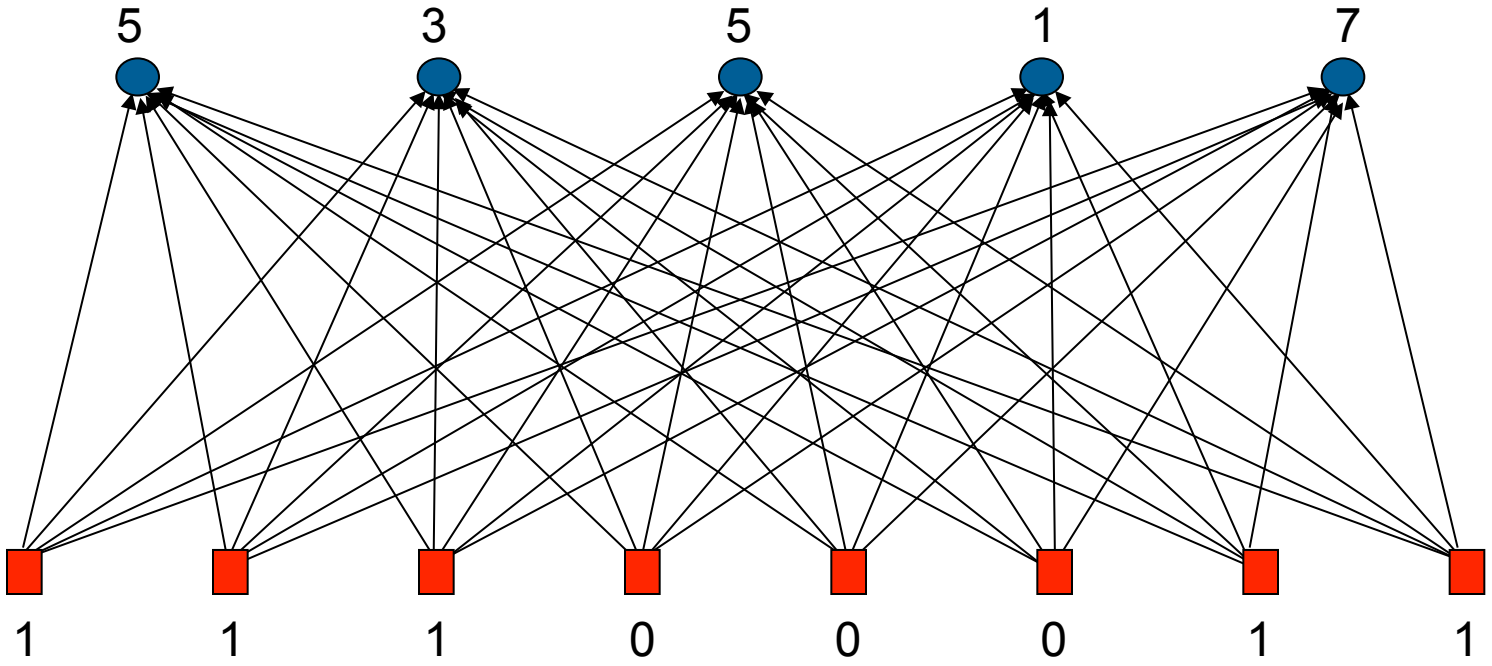
Using the Weights

- Each of the output nodes calculates the value of H based on its weight vector and the input vector.
- The node with the greatest value is the one most similar to the input!
- So ….?

The Hamming/MAXNET Network

5 Exemplars:

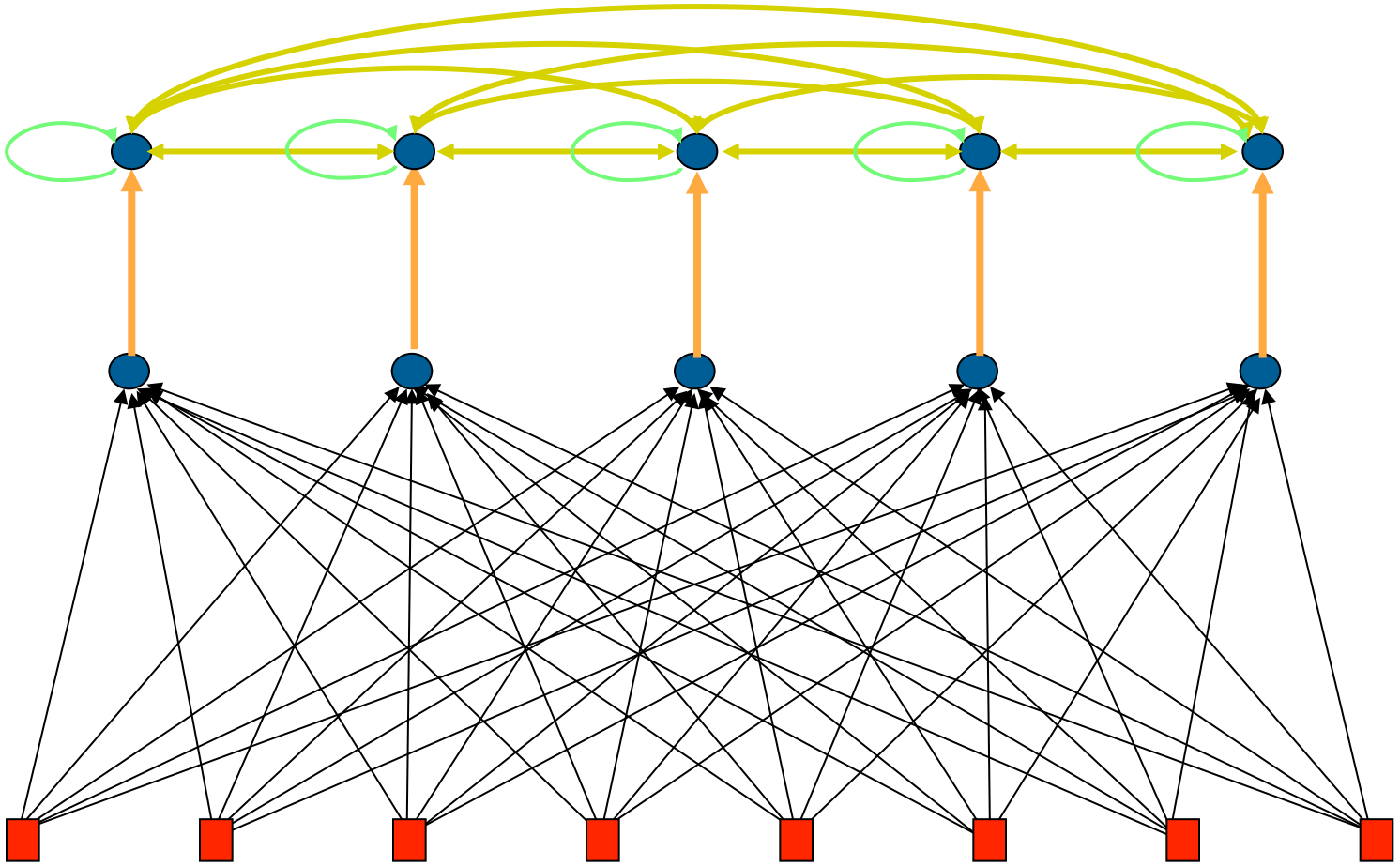
1: 1 0 1 0 1 0 1 0
2: 0 0 0 0 1 1 1 1
3: 1 1 1 1 0 0 0 0
4: 0 0 1 1 1 1 0 0
5: 1 1 0 0 0 0 1 1



The Hamming/MAXNET Network

- We want the network to "tell us" which node has the highest value of H.
- These nodes will have to fight it out!
- Competitive Learning!
- Add MAXNET to the Hamming Net.

The Hamming/MAXNET Network



The Hamming/MAXNET Network

- The MAXNET is a fully connected Hopfield type network.
- Weights w_{jk} are set by:

$$w_{jk} = \begin{cases} 1 & \text{if } j = k \\ -\varepsilon & \text{otherwise} \end{cases}$$

where $\varepsilon < 1/M$

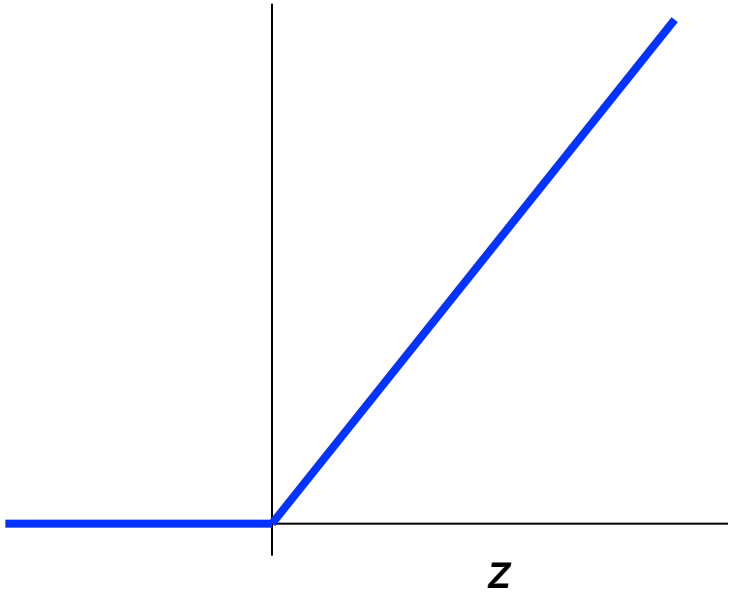
MAXNET

Each node calculates its activity value as usual and uses the hard-limiting function for its activation function:

$$A_j(t) = \sum_{k=1}^{M} w_{jk} x_k(t)$$

$$x_j(t+1) = f_t(A_j(t)) \quad \text{where}$$

$$f_t(z) = \begin{cases} az & \text{for } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



MAXNET

1. Calculate H using Hamming Net.
2. Present values of H to MAXNET.
3. Iterate MAXNET until convergence.

$$A_j(t) = \sum_{k=1}^{M} w_{jk} x_k(t)$$

$$x_j(t+1) = f_t(A_j(t))$$

$$= f_t\left[x_j(t) - \varepsilon \sum_{\substack{k=1 \\ k \neq j}}^{M} x_k(t) \right]$$

MAXNET

Time 0: 5 3 5 1 7

A for Node 1 = 5 – (1/6)(3 + 5 + 1 + 7) = 2.3333 ← Decreased by 2 2/3

A for Node 2 = 3 – (1/6)(5 + 5 + 1 + 7) = 0

A for Node 3 = 5 – (1/6)(5 + 3 + 1 + 7) = 2.3333

A for Node 4 = 1 – (1/6)(5 + 3 + 5 + 7) = -2.3333

A for Node 5 = 7 – (1/6)(5 + 3 + 5 + 1) = 4.6666 ← Decreased by 2 1/3!

Time 1: 2.333 0 2.333 0 4.666

MAXNET

Time 1:	2.333	0	2.333	0	4.666

A for Node 1 = 2.333 – (1/6)(0 + 2.333 + 0 + 4.666) = 1.16666

A for Node 2 = 0 – (1/6)(2.333 + 2.333 + 0 + 4.666) = -1.16666

A for Node 3 = 2.333 – (1/6)(2.333 + 0 + 0 + 4.666) = 1.16666

A for Node 4 = 0 – (1/6)(2.333 + 0 + 2.333 + 4.666) = -1.5555

A for Node 5 = 4.666 – (1/6)(2.333 + 0 + 2.333 + 0) = 3.8888

Time 2:	1.1666	0	1.1666	0	3.888

MAXNET

The keys to understanding it:

$$A_j(t) = \sum_{k=1}^{M} w_{jk} x_k(t)$$

$$x_j(t+1) = f_t(A_j(t))$$

$$= f_t\left[x_j(t) - \varepsilon \sum_{\substack{k=1 \\ k \neq j}}^{M} x_k(t) \right]$$

$$f_t(z) = \begin{cases} az & \text{for } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

MAXNET

Considerations:

- It can be **proven** that MAXNET always converges and find the node with maximum value so long as $\varepsilon < 1/M$.
- It has **advantages** over Hopfield net. It implements the optimum minimum error classifier when bit errors are random and independent.
- Tends to require **fewer** connections than in Hopfield. E.g., with 100 inputs and 10 classes, only 1100 connections are needed whereas with Hopfield with 100 inputs, 10,000 connections are needed. This difference increases as the number of inputs increase.

Summary

- The Hamming Distance measure for discrete vectors was modified to define a metric that measures how well a discrete vector matches another discrete vector.
- Defined an iterative scheme such that a node that has the greatest activity value will be the only node in a set of nodes that has a positive value.
- Competitive Learning.
- Can be used in Self-Organized Maps. Stay tuned!

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 12.3: Competitive Learning and Self-Organized Maps

In This Module We Will Cover:

- Competitive Learning
  - The Hamming Net
  - The MAXNET algorithm
  - Self-Organizing Maps
  - Data values ‘compete’ in some fashion

Self-Organizing Maps

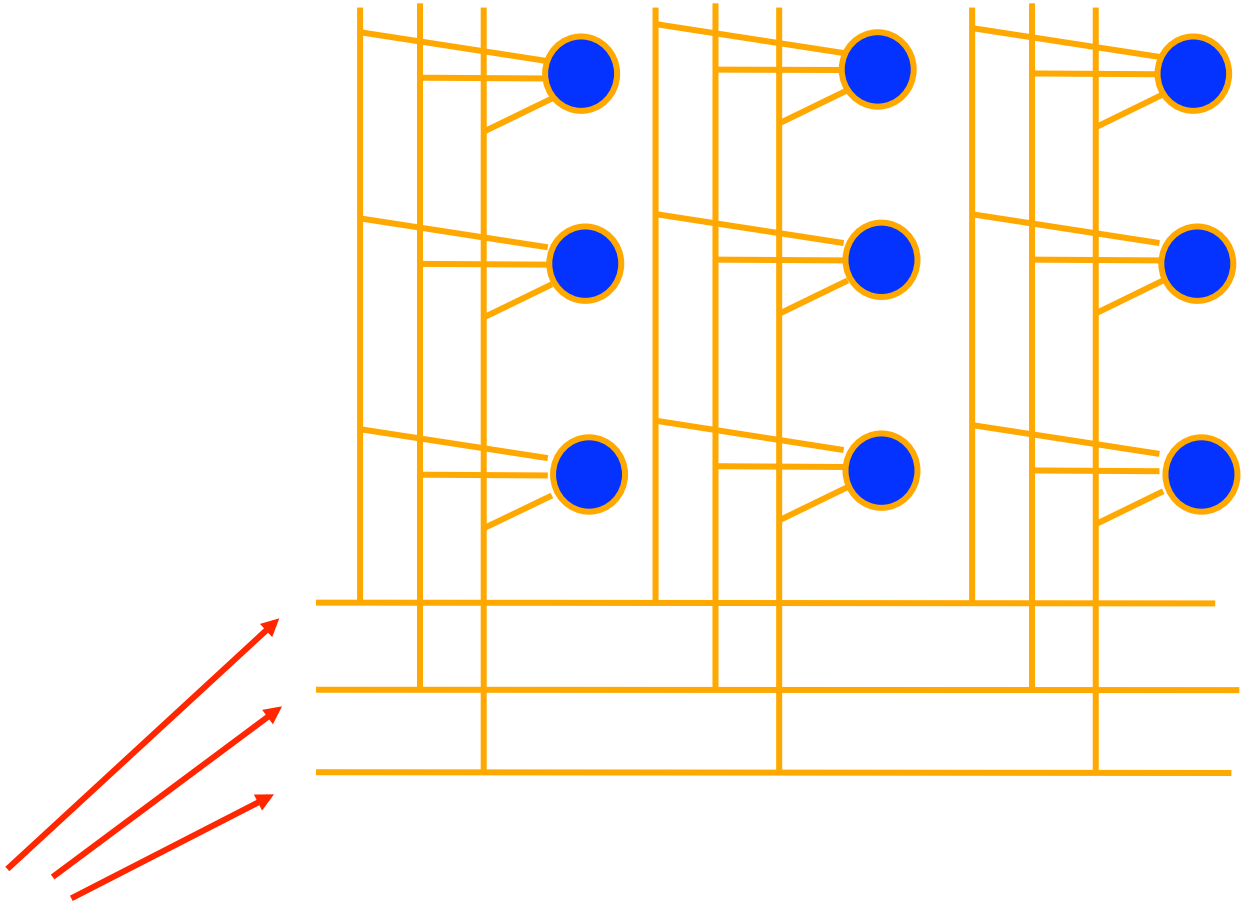
- Self-Organizing Maps (a.k.a. Kohonen Nets)
  - Unsupervised learning.
  - Data causes network to learn.
    - Data itself trains the net.
  - Uses Hamming Net and MAXNET.
  - Tries to create a *topology preserving map* where "near" inputs lead to "near" outputs.
  - Performs a type of feature extraction.
  - Akin to *operant conditioning.*
  - Data dimensionality reduction.
  - Displays a natural clustering behavior.

Self-Organizing Maps

What is the benefit of this?

- It extracts important features and segregates these features (see examples).
- It in effect allows for redundancy and fault tolerance.
- Reduces dimensions and displays similarities in input data. It can thus represent high-dimension data with smaller storage. The lack of dimension information is compensated in effect by a type of association topology, i.e., the feature extraction.

SOM Topology—an example



Converts 3D data to a 2D array!

Self-Organizing Maps

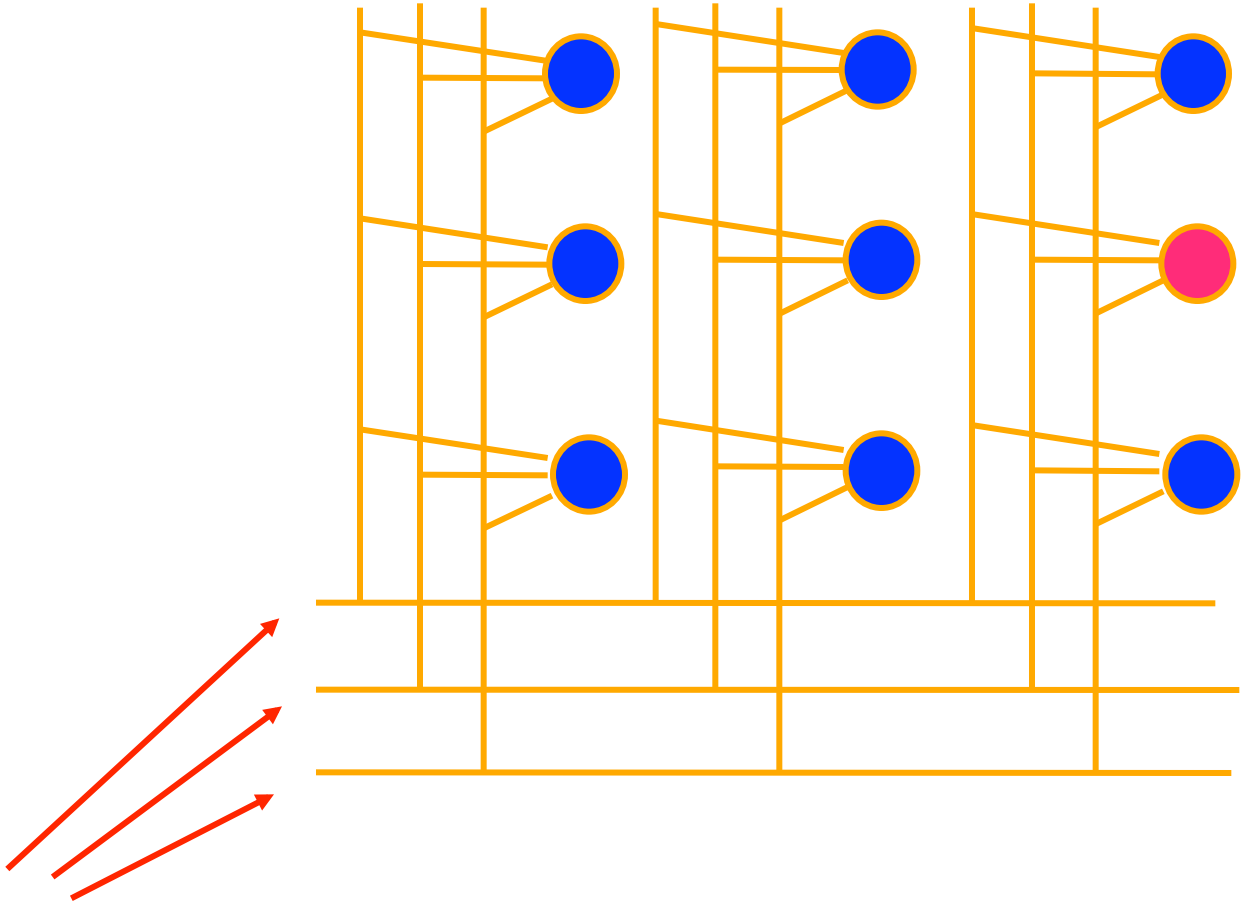
- Given this topology, each node has the 3 inputs **x** (shared with all other nodes) and a weight vector **w** (unique for each node) equal in size to the inputs. Basic idea is that for a given input (here a 3-tuple of possibly real values), with randomly assigned weights **w** for each node), determine which node's weights are 'closest' to the input vector.

Self-Organizing Maps

- Use competitive learning ala Hamming/ MAXNET to identify "winning" node.
- Use Hamming Distance, H, or other metrics: e.g.,

$$D(\mathbf{x}, \mathbf{w}) = \left(\sum_{i=1}^{N} (x_i - w_i)^2 \right)^{1/2}$$

SOM Topology—an example

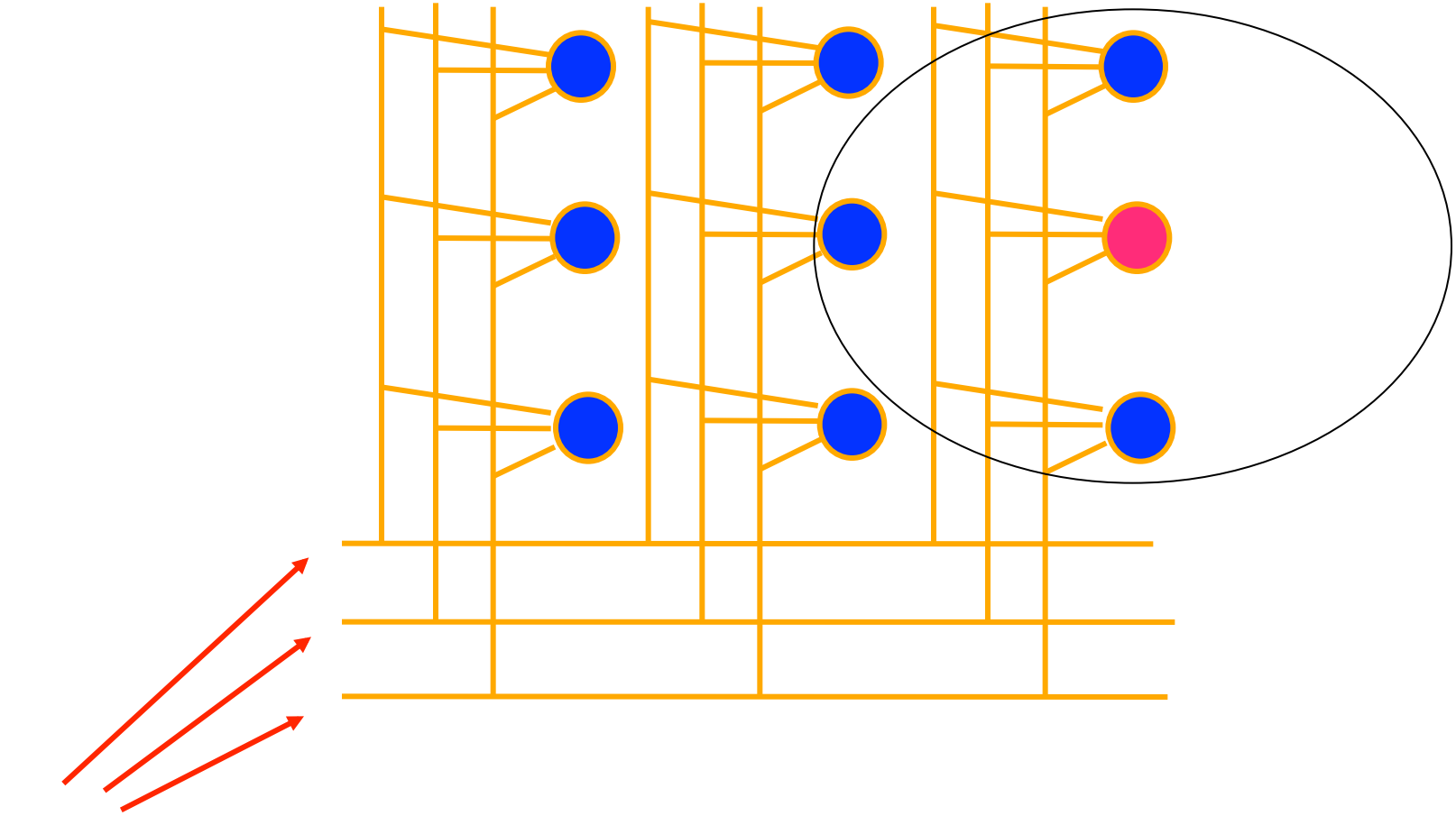


SOM Dynamics

- Select a neighborhood size σ and a neighborhood function $N(i,k)$ associated with the winning node.
- This function determines the magnitude of changes to the weight vectors of neighboring nodes. For example, for a node i,

$$N(i,k) = e^{-|r_k - r_i|^2/(2\sigma^2)}$$

SOM Topology—an example



SOM Dynamics

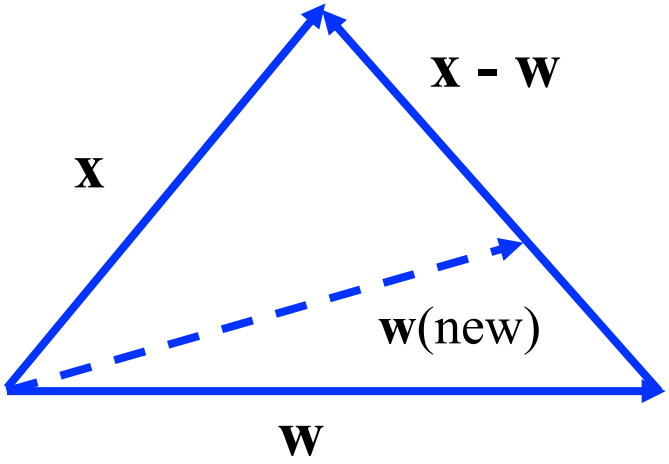
- Next we update all the weights of the neighboring (possibly all) nodes *k* in the array:

$$\mathbf{w}_k(\text{new}) = \mathbf{w}_k(\text{old}) + \mu N(i,k)(\mathbf{x} - \mathbf{w}_k)$$

What's going on here?
What does this update function do?

SOM Dynamics

- The weight vector is "moved" towards the input vector to a degree influenced by the topology (the nearness of the neighboring nodes):

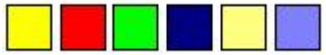


SOM Dynamics

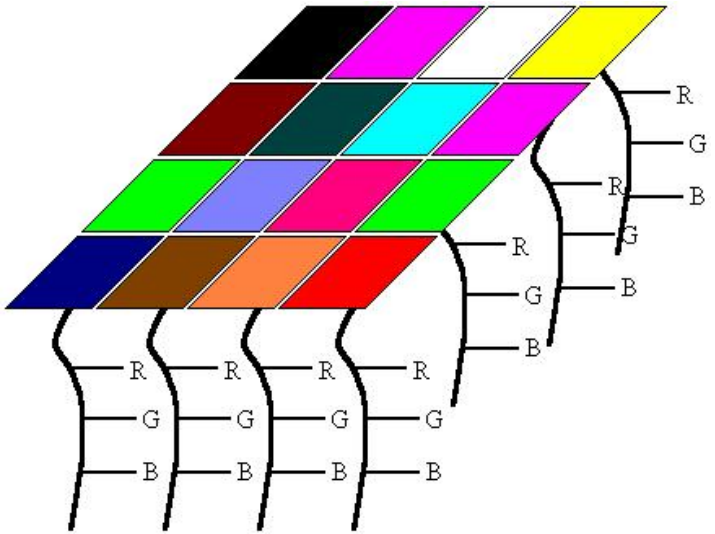
- Other similar update functions are possible, but the overall effect is to make the vector of weights closer to a given input pattern for the winning node and its neighbors.
- Overtime, the neighborhood weighting may lessen, the learning parameter also may decrease.
- Eventually, new input data gets mapped to a particular region of nodes with little if any effect on the nodes.
- Wanna see?

First some more background…

- **Example:** Use a 3 dimensional vector to represent colors in RGB:



- The network topology could be illustrated as:



First, Randomly Assign Weights

- In this example, this means randomly assign color values:

Other updating rules…

- Now update the nodes using an updating rule.
- Another, simpler rule is

$\mathbf{w}_{NEW} = \mathbf{w}_{OLD}\,(1 - \lambda) + \mathbf{x}(\lambda)$ where λ is initially 1

Eventually, this array becomes

Let's see it in action…

http://davis.wpi.edu/~matt/courses/soms/applet.html

Other Examples….

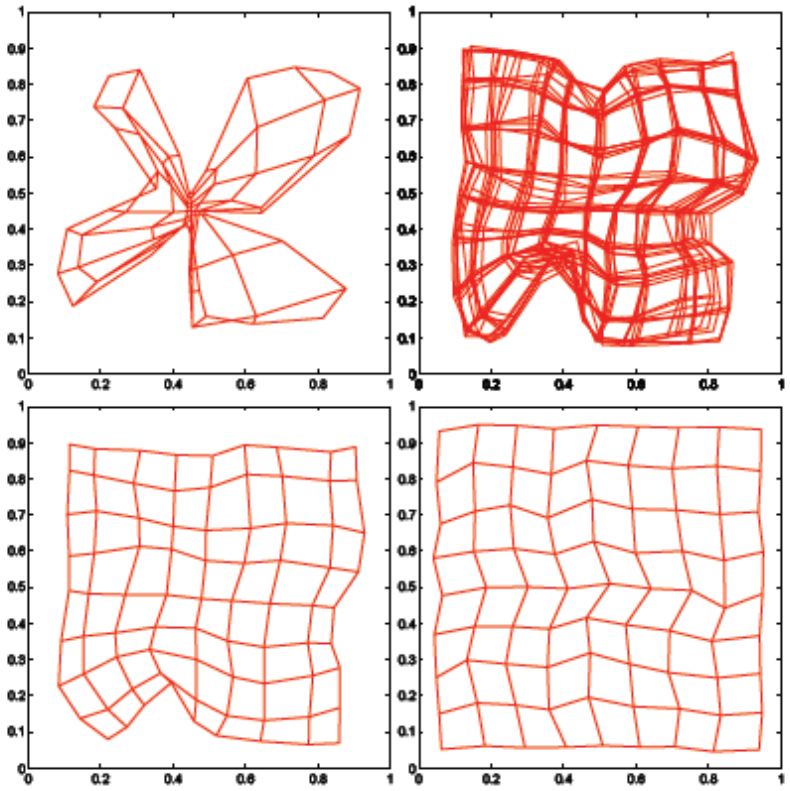


Fig. 15.7. Mapping a square with a two-dimensional lattice. The diagram on the upper right shows some overlapped iterations of the learning process. The diagram below it is the final state after 10000 iterations.

From Rojas.

Summary

- The Hamming Distance measure for discrete vectors was modified to define a metric that measures how well a discrete vector matches another discrete vector.
- Defined an iterative scheme such that a node that has the greatest activity value will be the only node in a set of nodes that has a positive value.
- Competitive Learning.
- Applied 3 dimensional color values to a 2 dimensional array.
- Applied 2 dimensional ‘lattice values’ to display a uniform distribution of input vectors.

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 13.1: Clustering

In This Module We Will Cover:

- Clustering:
  - The nature of the problem
  - Defining it
  - Modeling it as an optimization problem
- Radial Basis Functions
  - Their connection to clustering

What is the Clustering Problem

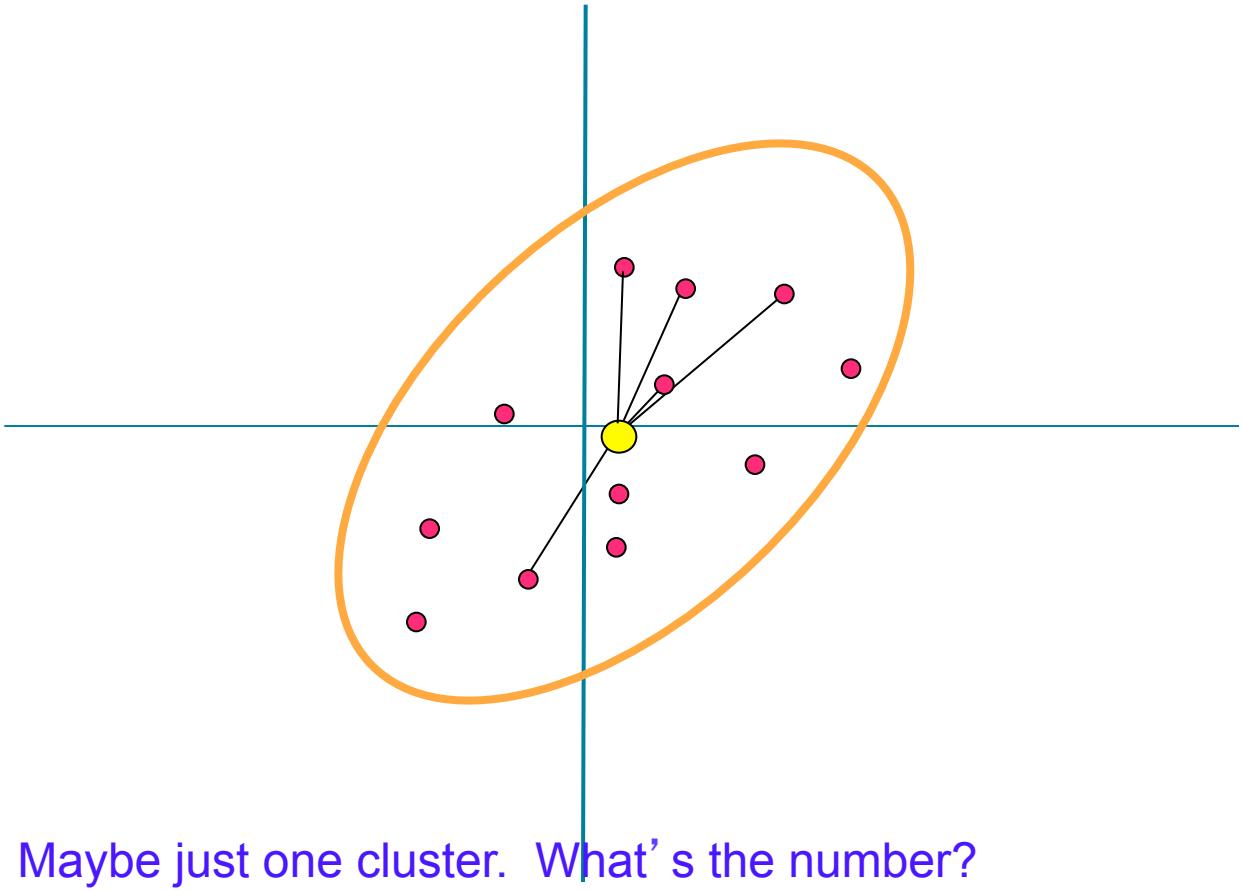
- What is a cluster?
  - One definition: A set of points that are within some defined distance of a 'centroid'.
- Questions:
  - Which 'cluster' does a point belong? How can we classify a point to be a member of a 'cluster'?
  - How many clusters should there be?

Cluster Definitions

- Many possible ways to define a cluster.
- Can require that a cluster ‘centroid’ be an element of the population.
- Or, we can require it to be defined by some function.

How Many Clusters?

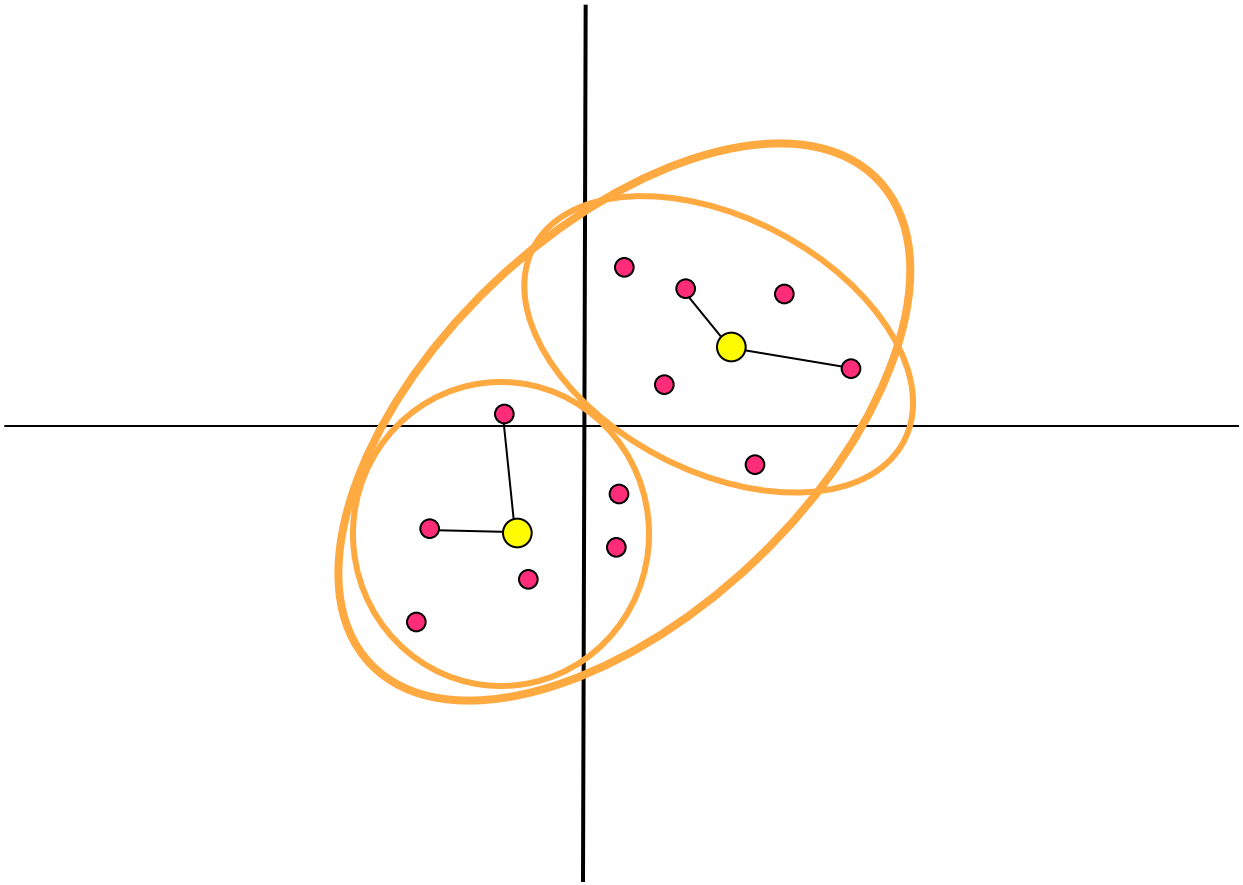
There are 12 points. What are the clusters?



Maybe just one cluster. What's the number?

How Many Clusters?

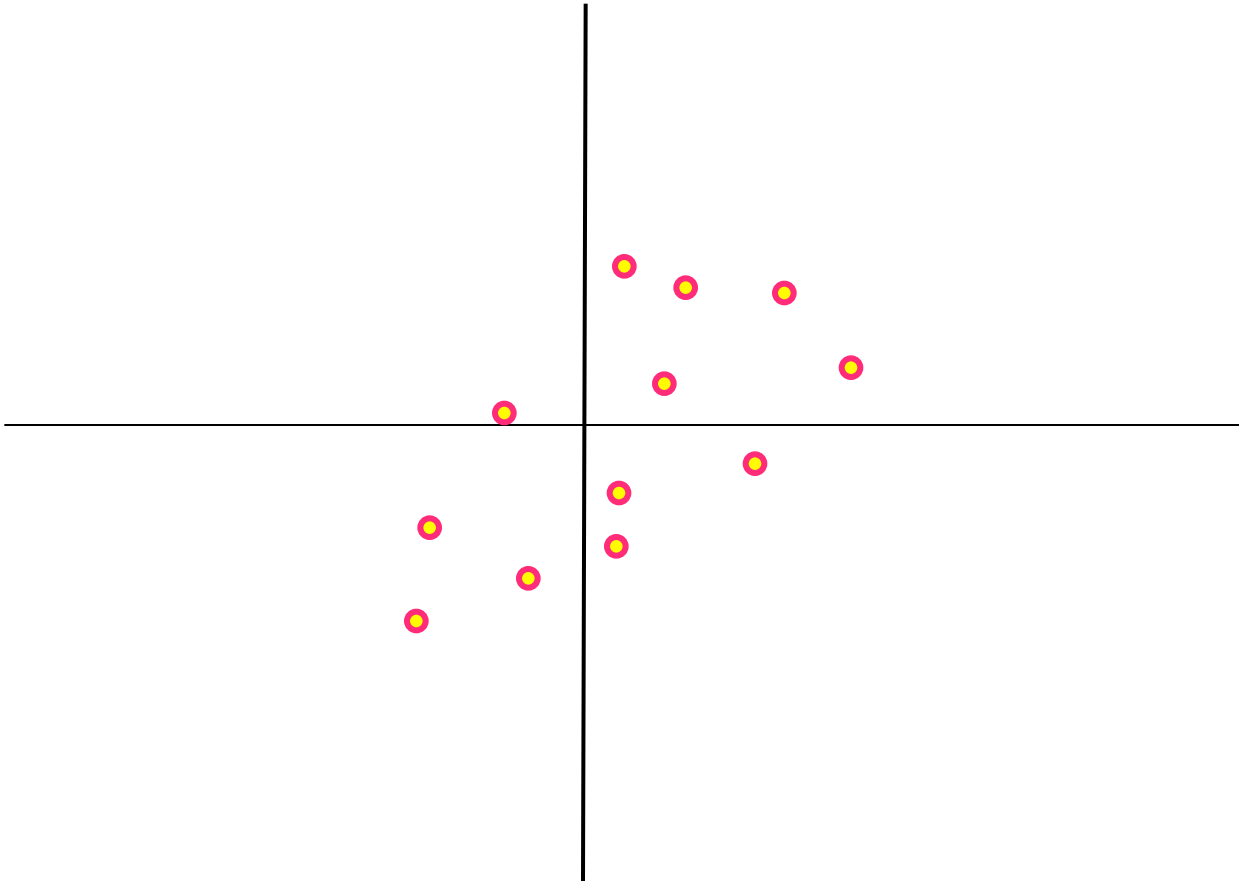
There are 12 points. What are the clusters?



We can add more clusters. How many?

How Many Clusters?

There are 12 points. What are the clusters?



The Problem with Clusters

- Points can appear to have ‘natural’ clustering
  - Not a very scientific or objective way of describing or determining clusters
  - How can we objectively determine the number of cluster centers?
- We need some metric associated with the number of clusters and where the cluster centers are located

Posing an Optimization Problem

- Establishes objective criteria for how to define clusters
- Could capture issues associated with costs and benefits of clusters
- Main idea: A point should be closest to 'its' cluster center
- What is the tradeoff?

The Optimization Problem

The Tradeoff:

- If there are more cluster centers, a point will tend to be closer to its center
- More centers, less distance
- Fewer centers, greater distance
- Our objective function should entail both elements:
  - 1) the number of cluster centers;
  - 2) the ‘distance’ of each cluster member to its center.

The Optimization Model

- Definitions:
  - Let P be the set of all points.
  - Let C be the set of all cluster centers
  - A point p is a member of cluster c_i if its distance to the centroid (or center) of c_i is less than the distance to any other cluster center.
  - The distance metric associated with cluster c_i is some function of the distance from the center of c_i to all of its members

The Optimization Model

A set of points in some space: $P = \{p_1, p_2, \ldots, p_n\}$

A set of clusters/centers: $C = \{c_1, c_2, \ldots, c_M\}$

Membership criteria: $p' \in c_j \text{ iff } \rho(p', c_j) < \rho(p', c_i) \; \forall \, i \neq j$

The Optimization Model

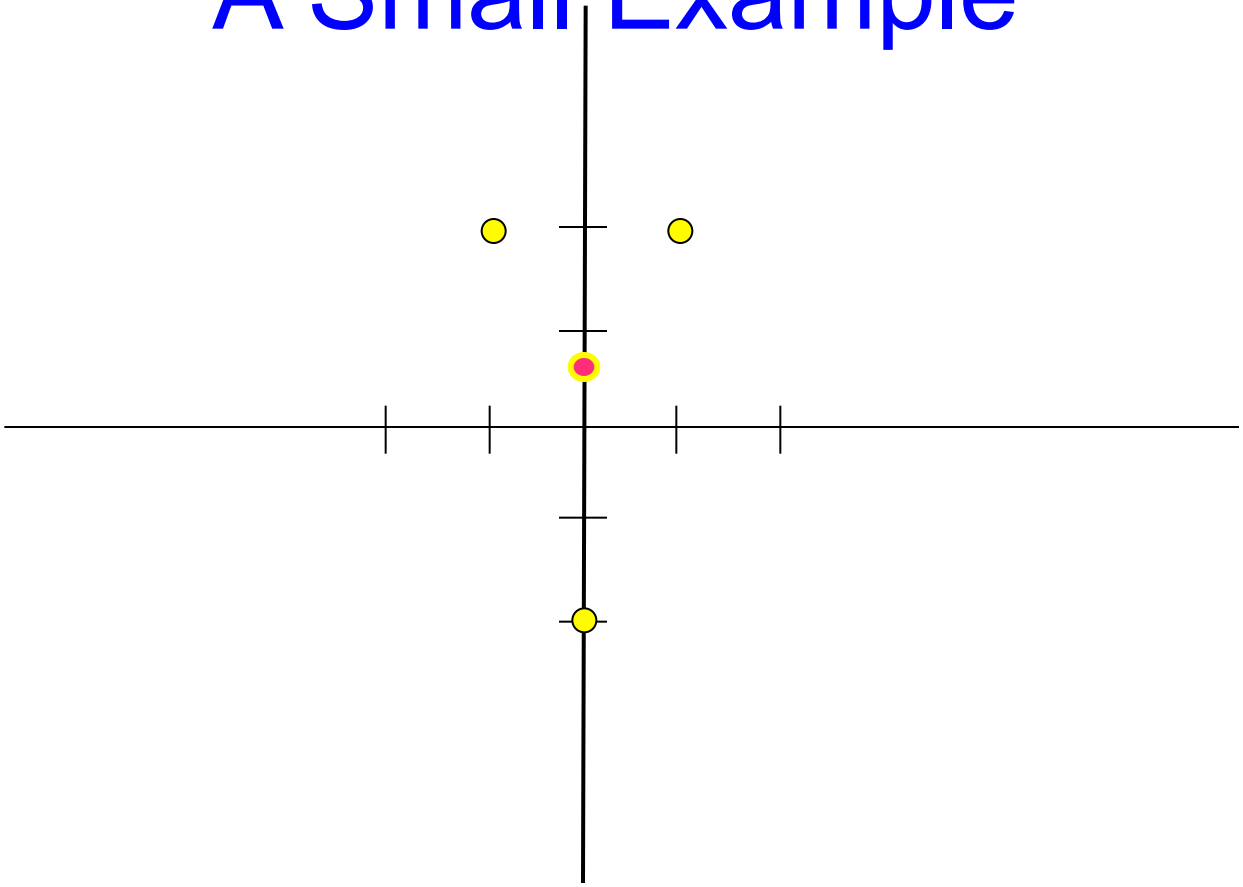
The number of cluster centers = $|C|$

A distance metric for each cluster center:

$$D_{c_j} = \sum_{p_i \in c_j} \rho\left(p_i, c_j\right)$$

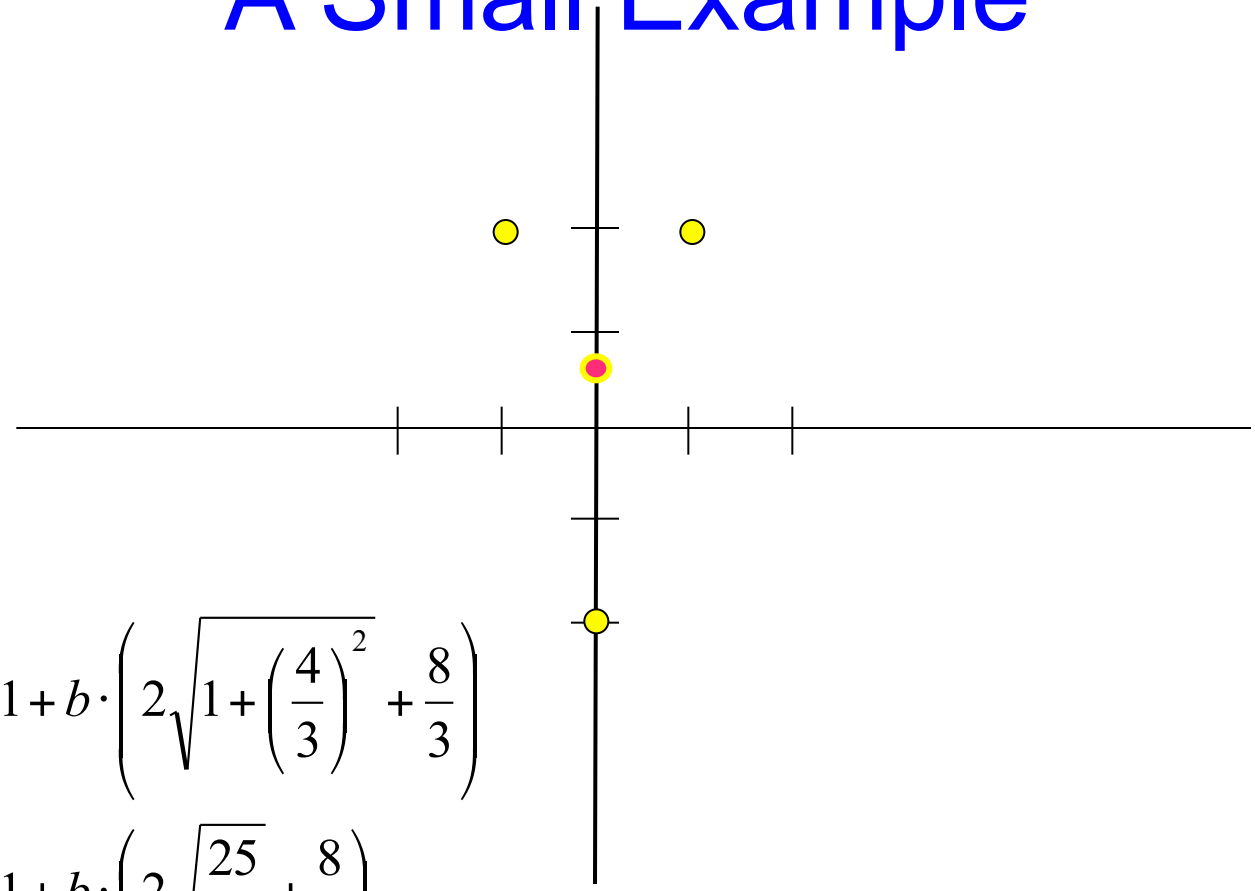
$$f(\rho, C) = a|C| + b\sum_{c_i \in C} D_{c_i}$$

A Small Example



$$f(\rho, C) = a \cdot |C| + b \cdot D_{c_1}$$

A Small Example



$$f(\rho, C) = a \cdot 1 + b \cdot \left(2\sqrt{1 + \left(\frac{4}{3}\right)^2} + \frac{8}{3} \right)$$

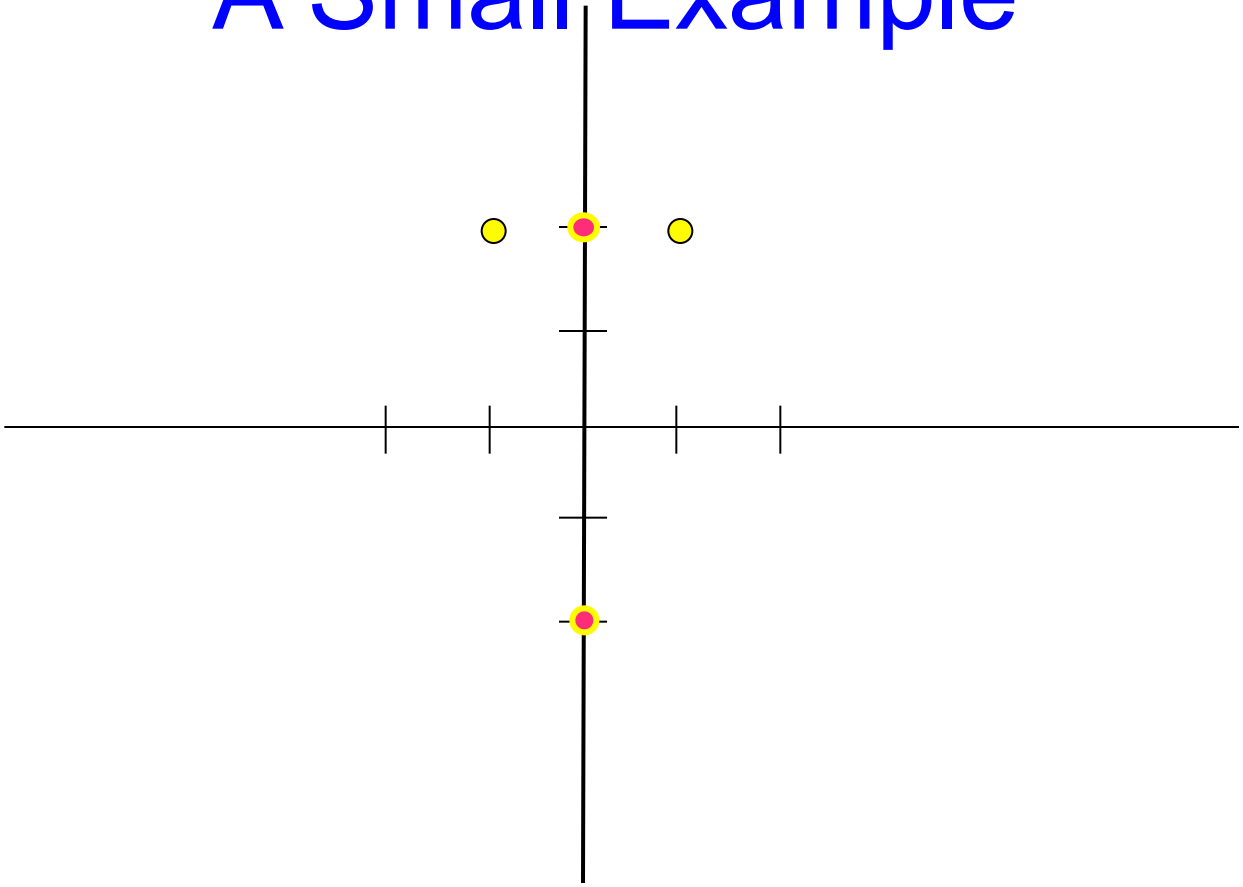
$$= a \cdot 1 + b \cdot \left(2\sqrt{\frac{25}{9}} + \frac{8}{3} \right)$$

$$= a \cdot 1 + b \cdot 6$$

A Small Example

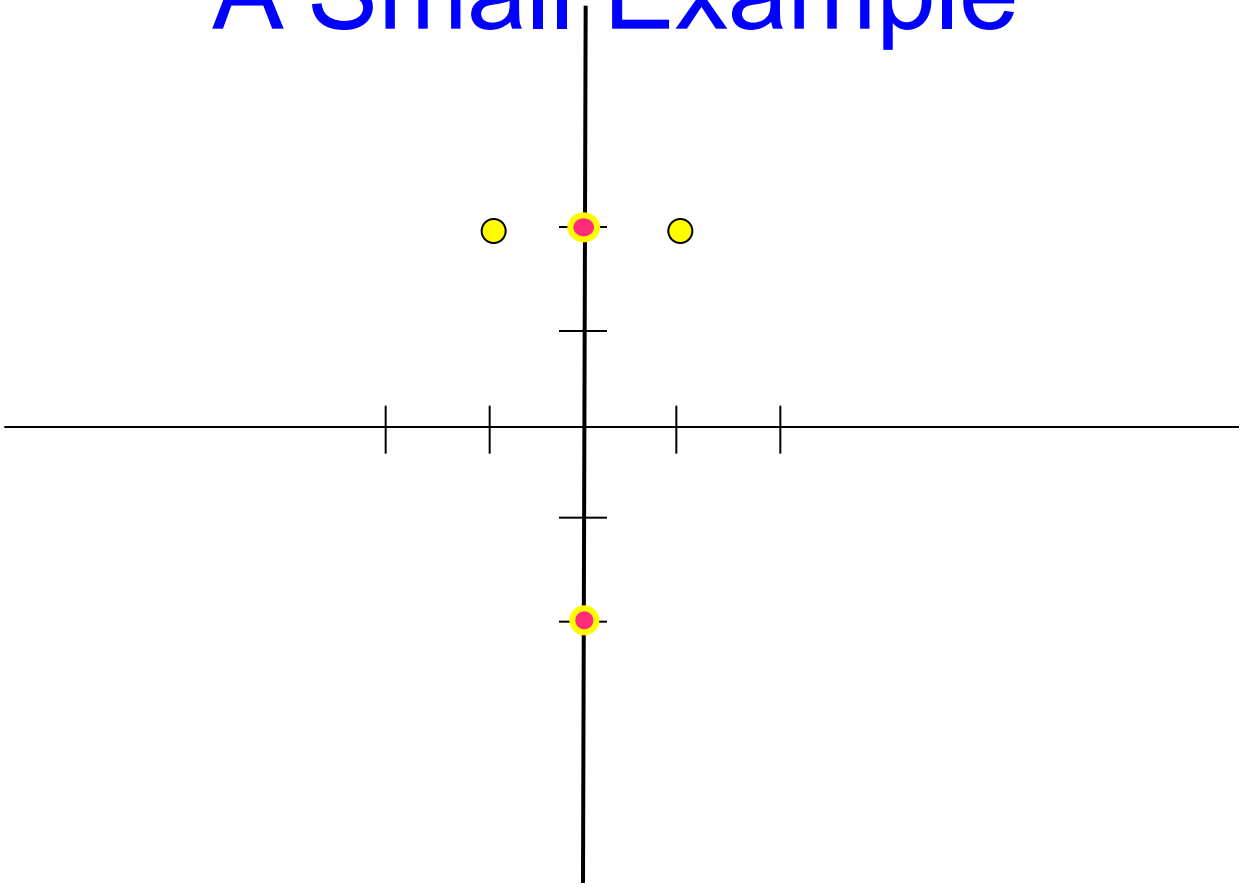
- So if we weight the distance metric as more costly, say b = 10, than the number of cluster centers where a = 1, then the ‘cost’ function is:
- 61

A Small Example



$$f(\rho, C) = a \cdot |C| + b \cdot D_{c_1}$$

A Small Example



$$f(\rho, C) = a \cdot 2 + b \cdot (2 + 0)$$

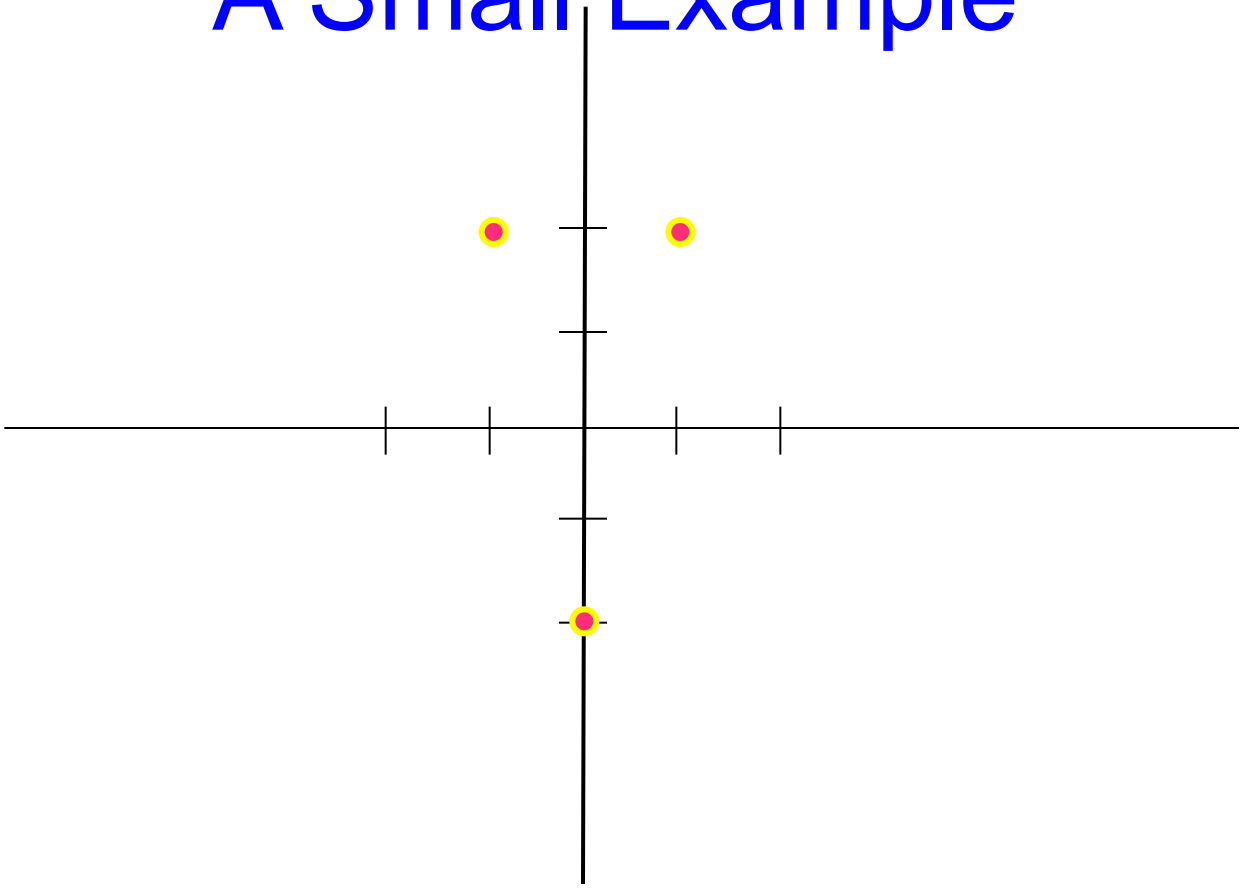
A Small Example

- So with a = 1 and b = 10, the ‘cost’ is: 22.

A Small Example

- What if the relative costs were different? Let a = 10, b = 1:
- Then the first arrangement has value:
- $10 \cdot 1 + 1 \cdot (6) = 16$
- And the second arrangement = 22

A Small Example



$$f(\rho, C) = a \cdot 3 + b \cdot (0)$$

A Small Example

- Now the cost = 3.

If a = 10, b = 1, then cost is 30

Summary of Examples

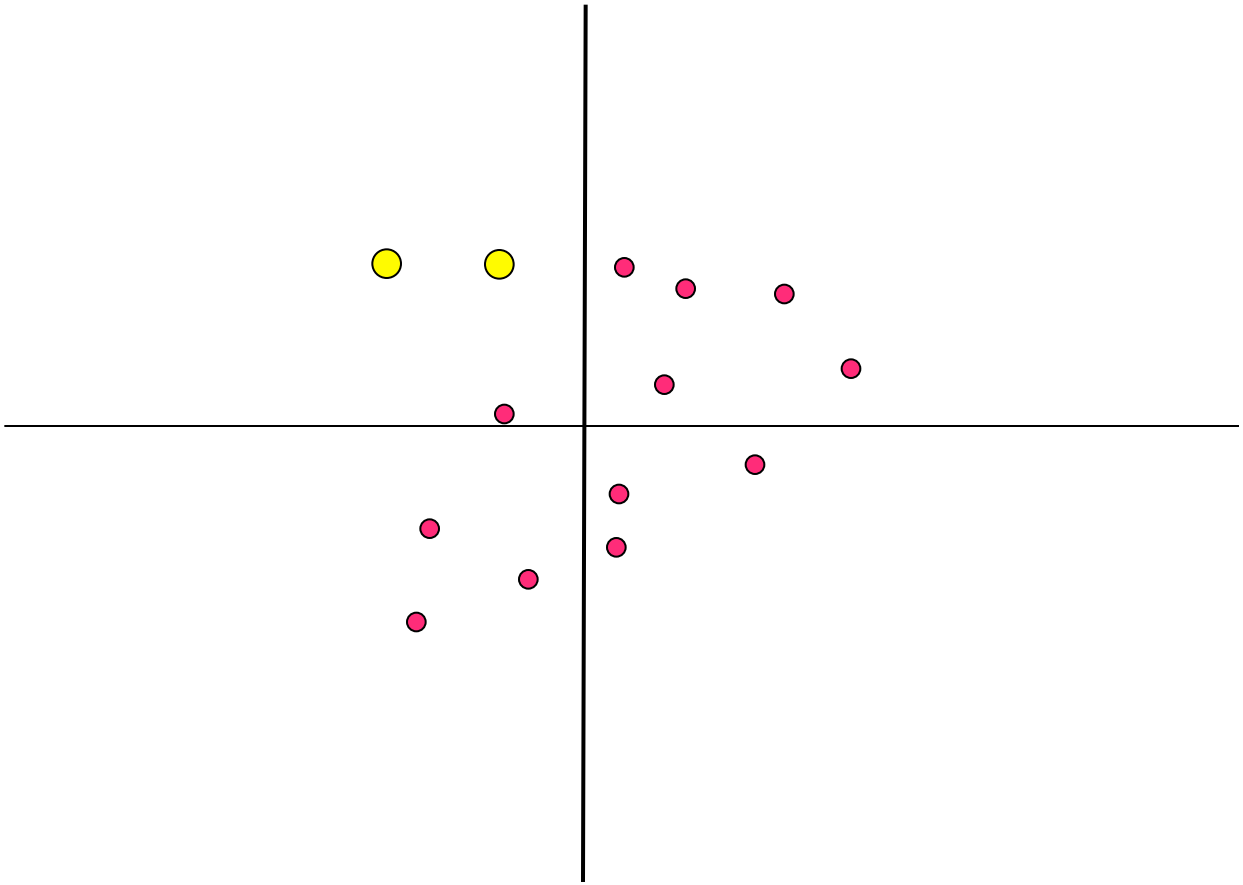
Number of Clusters	a =1, b = 10	a =10, b = 1
1	61	16
2	22	22
3	3	30

The Optimization Model

- Once a metric is chosen, the function *f* is a function of the cluster centers, and the number of them.
- We want to minimize *f.*
- How hard is this problem?
- Really, really hard if we want to be totally objective!

Illustrating the Algorithm

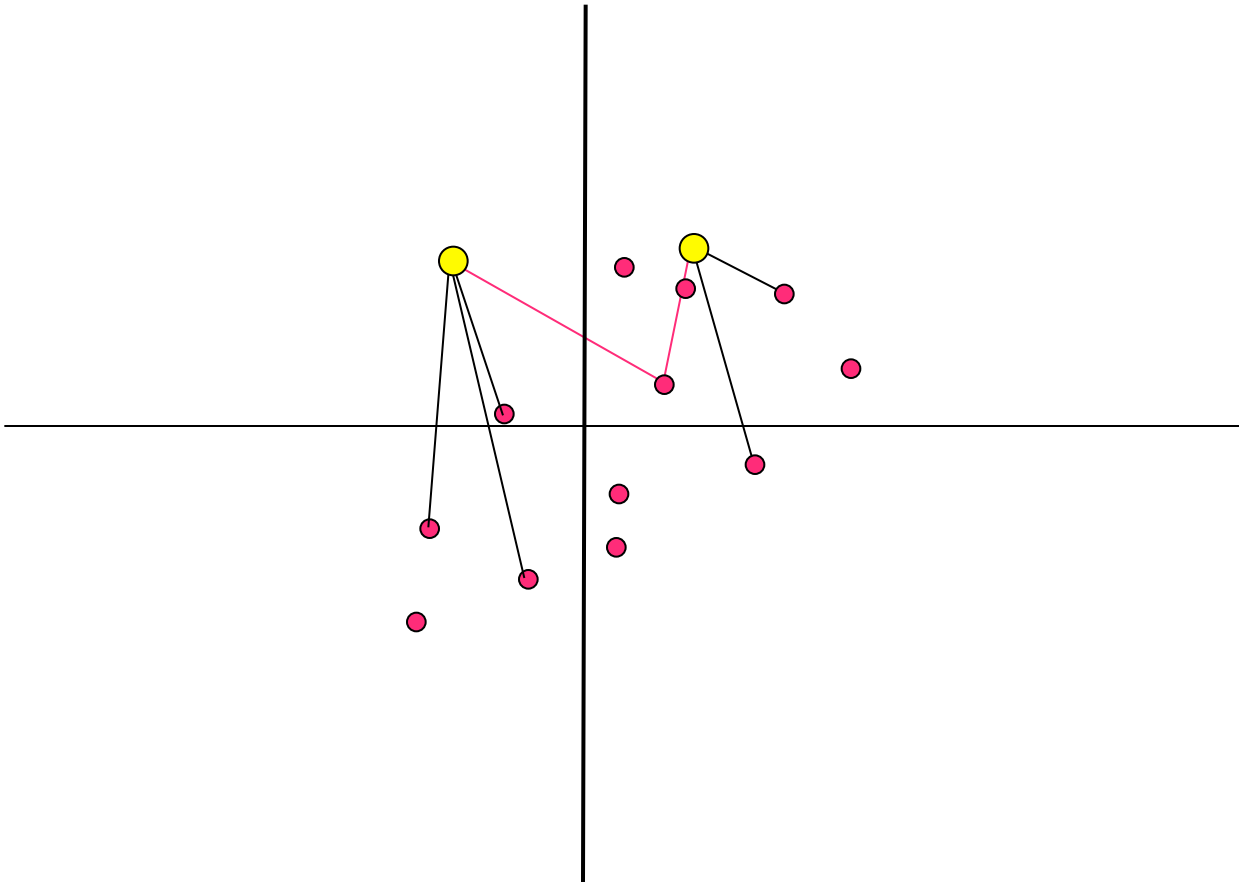
Say we are now considering two cluster centers



We can add more clusters. How many?

Illustrating the Algorithm

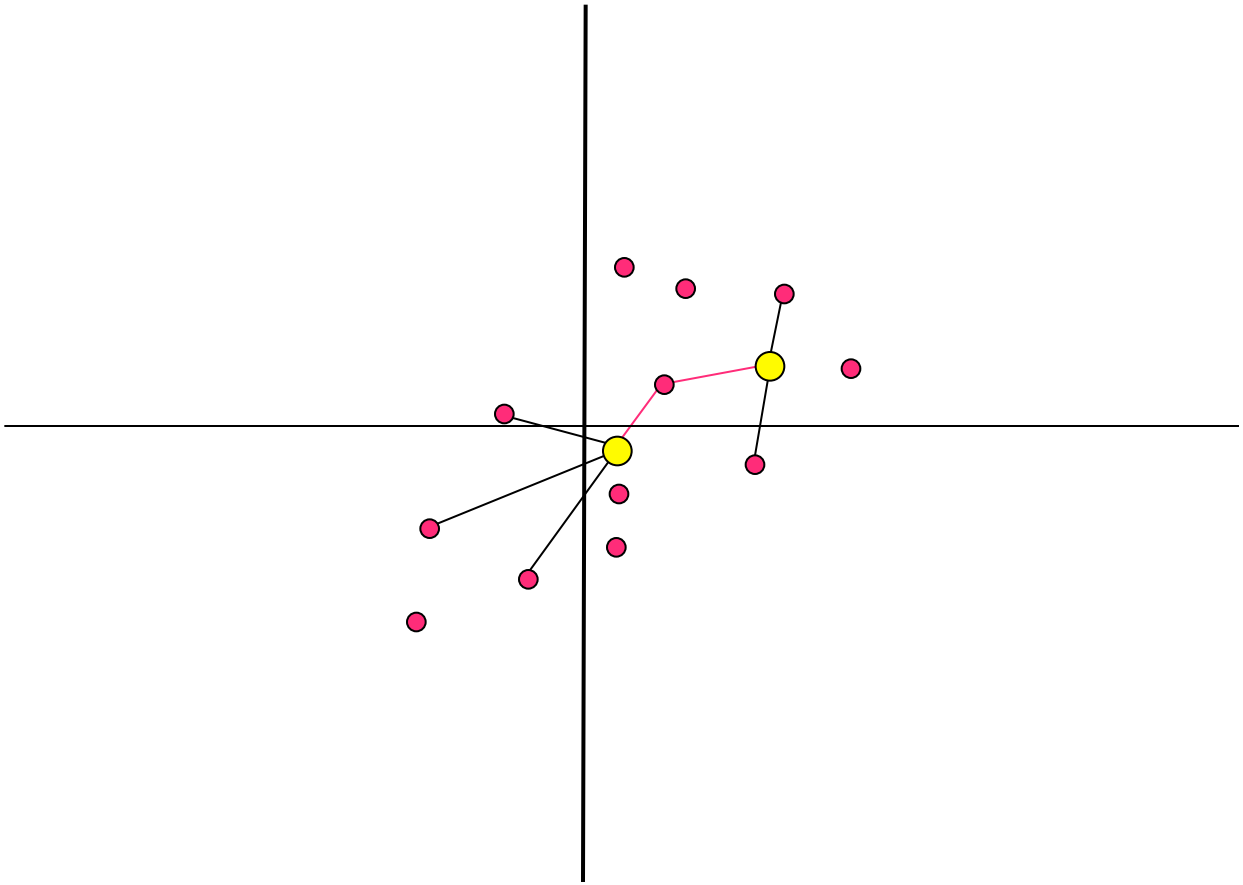
Say we are now considering two cluster centers



Determine cluster location, membership and distances.

Illustrating the Algorithm

Say we are now considering two cluster centers



Determine cluster location, membership and distances.

The Optimization Problem

1. Initialize *a* and *b.*
2. Add another cluster center to the mix.
3. Place the cluster centers.
4. Determine point membership based on cluster center placement.
5. Calculate the function *f.*
6. If *f* is less than the best so far, save it.
7. Goto 3 to reposition cluster centers.
8. If after all cluster center positions have been used, goto step 2.

Heuristics to the Rescue!

- We can make intelligent guesses as to initializing cluster center locations.
- Then add additional cluster centers up to some number.
- How can neural networks be used here?

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 13.2: Radial Basis Functions

What We've Covered and Will Cover:

- Clustering:
  - The nature of the problem
  - Defining it
  - Modeling it as an optimization problem
- Radial Basis Functions
  - Their connection to clustering
  - An Application of RBFs

Heuristics to the Rescue!

- We can make intelligent guesses as to initializing cluster center locations.
- Then add additional cluster centers up to some number.
- How can neural networks be used here?

Basis Functions

The first level of processing regarding weights and inputs.

Up to now, we've only really considered a Linear Basis Function for a perceptron:

$$A_i = \sum_j w_{ij} x_j + \theta_i$$

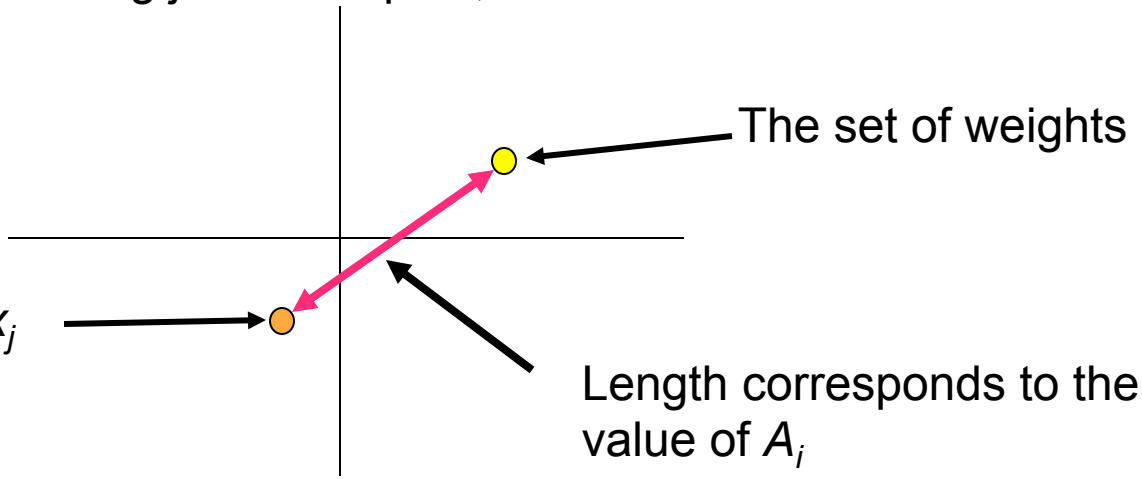
but there are others!

Radial Basis Functions

An Example:

$$A_i = \sqrt{\sum_{j=1}^{n} \left(w_{ij} - x_j \right)^2}$$

With a node having just two inputs, it looks like this:



Radial Basis Functions

- Another similar basis function, but a bit more general:

$$A_i = \sqrt{\sum_{j=1}^{n} \alpha_{ij} \left(w_{ij} - x_j \right)^2}$$

This is an elliptic basis function.

Observations

- The closer the inputs are to the weights, the smaller the value of A_i
- If we want a perceptron to 'fire' when inputs are close to the weights, what kind of activation function should we have?
- We could use a Gaussian function as the activation function.
- We could also use some threshold post-processing to make the output 1 when the inputs are within a certain radius of the weights.

Some Variations of its use

$$f_i(\mathbf{x}) = \frac{e^{-\sum_j (x_j - w_j)^2 / 2\sigma^2}}{\sum_i e^{-\sum_j (x_j - w_j)^2 / 2\sigma^2}}$$

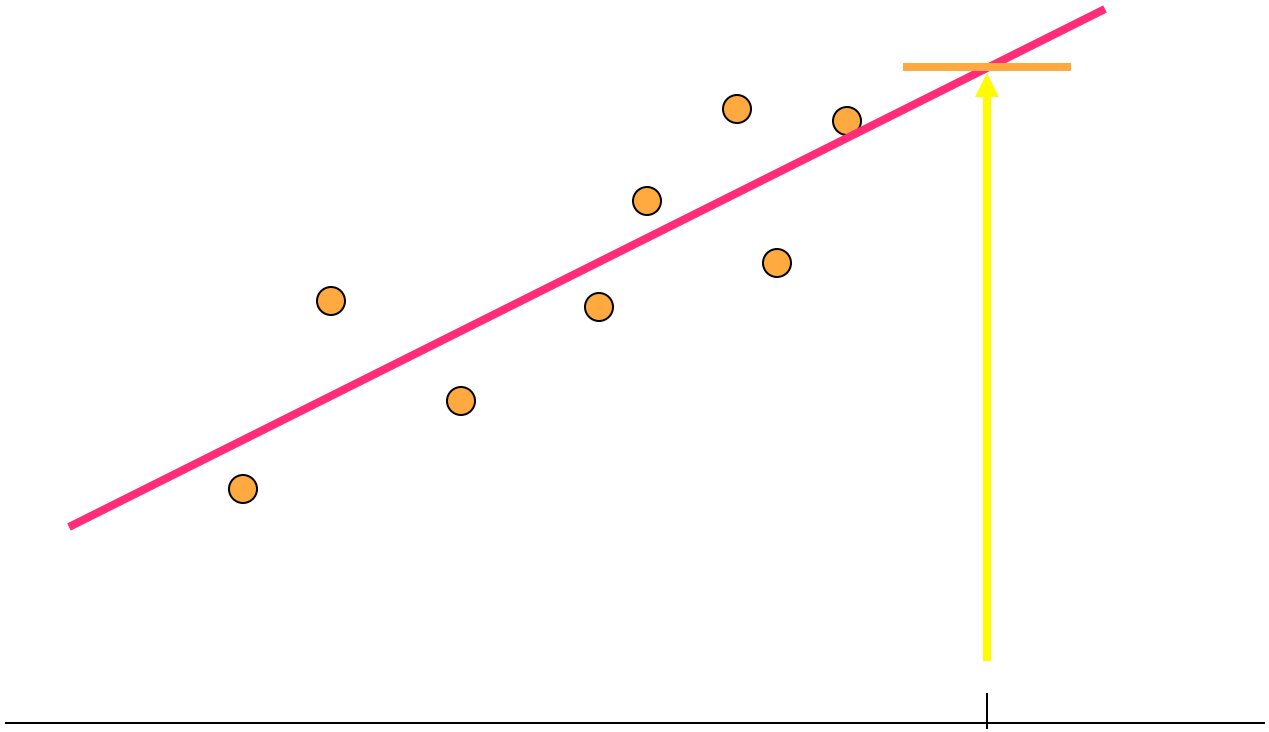
Utility of RBFs

- Can be used to approximate many functions.
- From Rojas: "The main difference between networks made of radial basis functions and networks of sigmoidal units is that the former use locally concentrated functions as building blocks whereas the latter use smooth steps."

Regression

- Regression is useful for prediction
- Also useful for function approximation
- We assume data is produced by some underlying model and perturbed by random variation (noise)
- Can NNs be useful here? Can different basis functions be useful here?

Regression—a Review



Generalized Regression Neural Network

- Attempts to perform generalized regression calculations
- Useful for many types of regression problems
- Uses four layers:

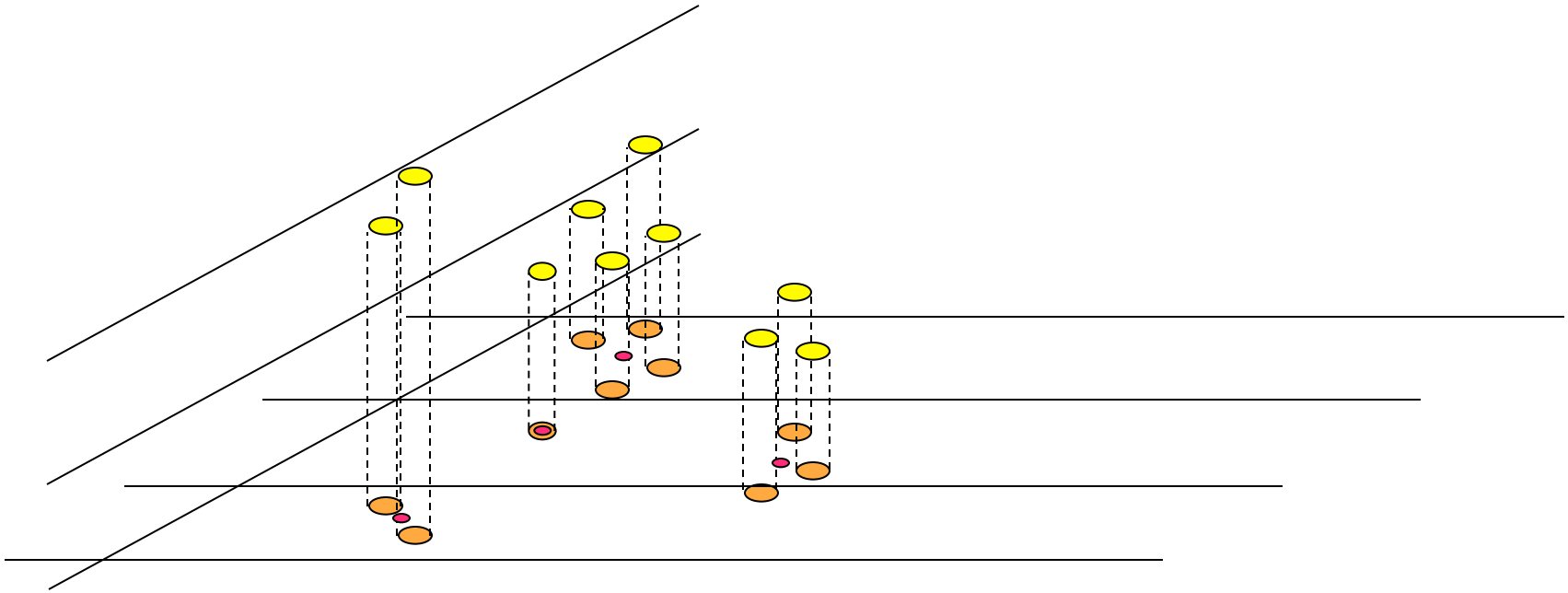
Generalized Regression Neural Network

$$E\left[y|X\right]=\frac{\int_{-\infty}^{\infty} yf(X,y)dy}{\int_{-\infty}^{\infty} f(X,y)dy}$$

$$D_i^2=\left(\mathbf{X}-\mathbf{X}_i\right)^{\mathbf{T}}\left(\mathbf{X}-\mathbf{X}_i\right)$$

$$\hat{Y}(\mathbf{X})=\frac{\sum_{i=1}^{n}\mathbf{Y}_i\exp\left(-\frac{D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^{n}\exp\left(-\frac{D_i^2}{2\sigma^2}\right)}$$

Generalized Regression Neural Network



Generalized Regression Neural Network

- From Specht:

Represent exemplars or cluster centers. They could be based on RBFs with Gaussian activation.

Performs a weighted sum of the Pattern Unit outputs. Weights correspond to number of observations associated with that cluster.

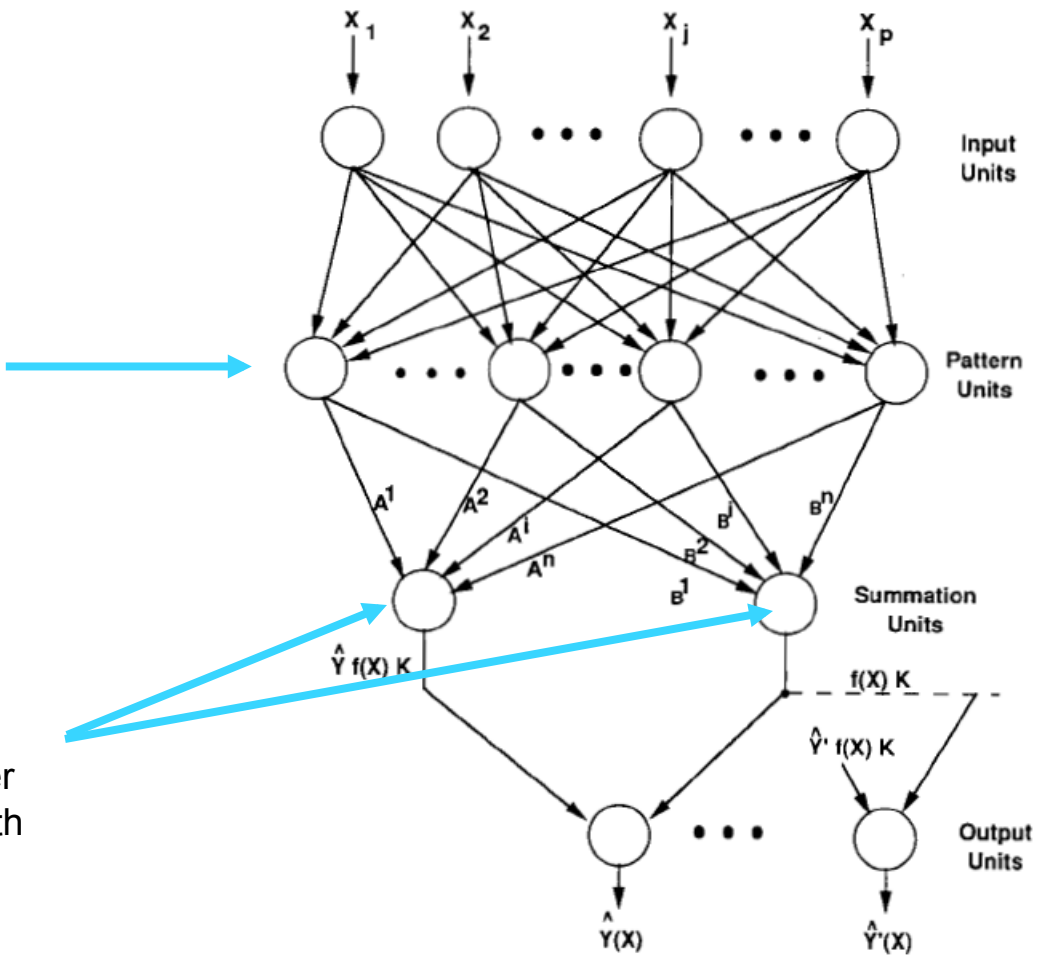


Fig. 1. GRNN block diagram.

Generalized Regression Neural Network

- Two Summation Units:

  f(**X**)*K:* "sums the outputs of the pattern units weighted by the number of observations each cluster center represents."

$\hat{\mathbf{Y}}f(\mathbf{X})K$:This unit "multiplies each value for a Pattern Unit by the sum of samples [Y_i] associated with Cluster center [X_i]."

Generalized Regression Neural Network

- Applications of GRNNs:
  - General regression problems,
  - Adaptive control,
  - Filtering,
  - …

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 14.1: Cellular Neural Networks

My Brain

In This Module We Will Cover:

- State-Based Representation of Linear Systems
- Cellular Automata—brief intro
- Cellular Neural Networks

Linear Systems

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}$$

State Equation Representation of a
Continuous, Time Invariant, Linear System.

Given a set of initial conditions, the trajectories (values through time) of $\mathbf{x}(t)$ and $\mathbf{y}(t)$ can be determined.

A good foundation for thinking about and modeling complex systems!

Cellular Automata

- Explored in the 1940s by Stanislaw Ulam and John von Neumann
- Exhibits complex behavior
- Sensitive dependence on initial conditions
- Paradigm for exploring how simplicity evolves to complexity

Game of Life

- Conway formulated a simple 2-state CA.
- Very simple rules govern the evolution of states.
- All states updated synchronously.

Game of Life

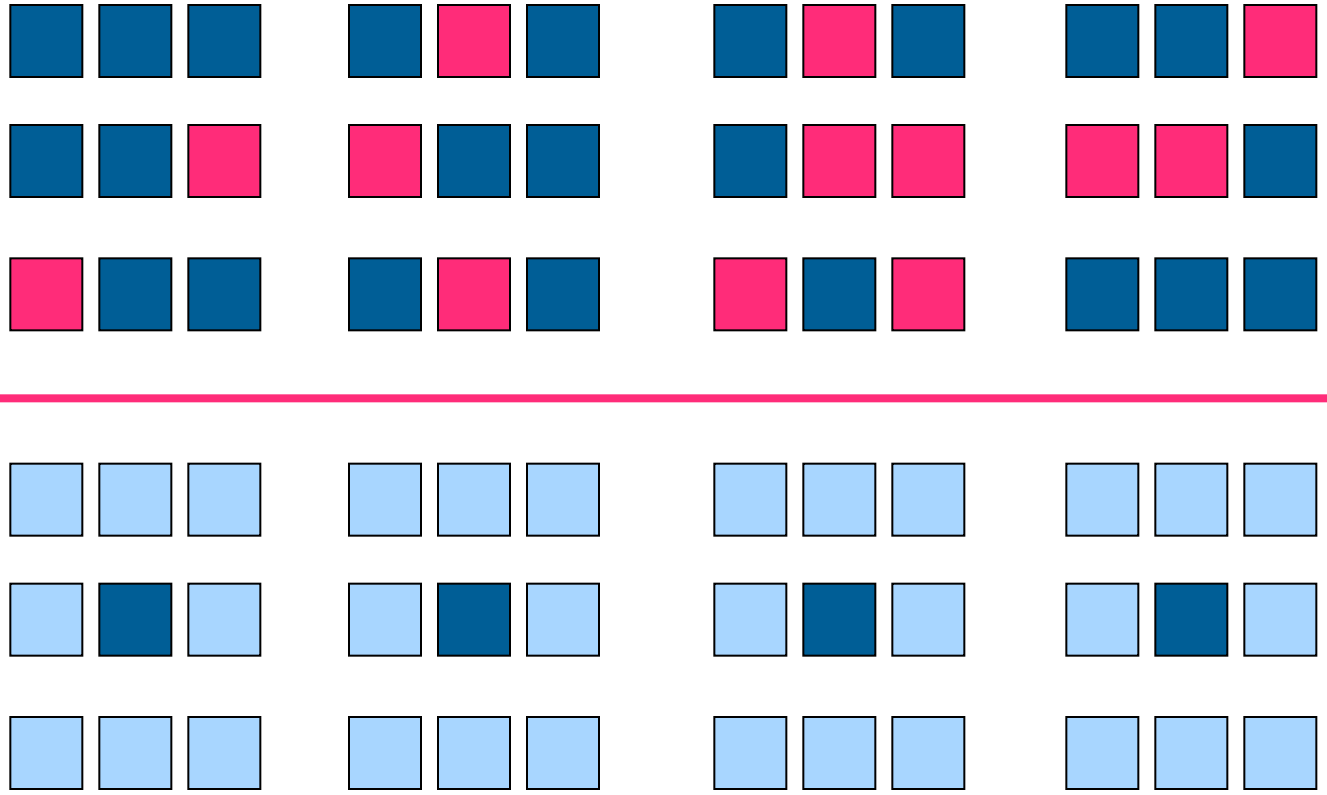
- 2 dimensional array of “cells”
- Each cell can be in one of two states: “alive” or “dead”.
- Each cell’s state in the next iteration depends on how “crowded” it is within its neighborhood.

Game of Life

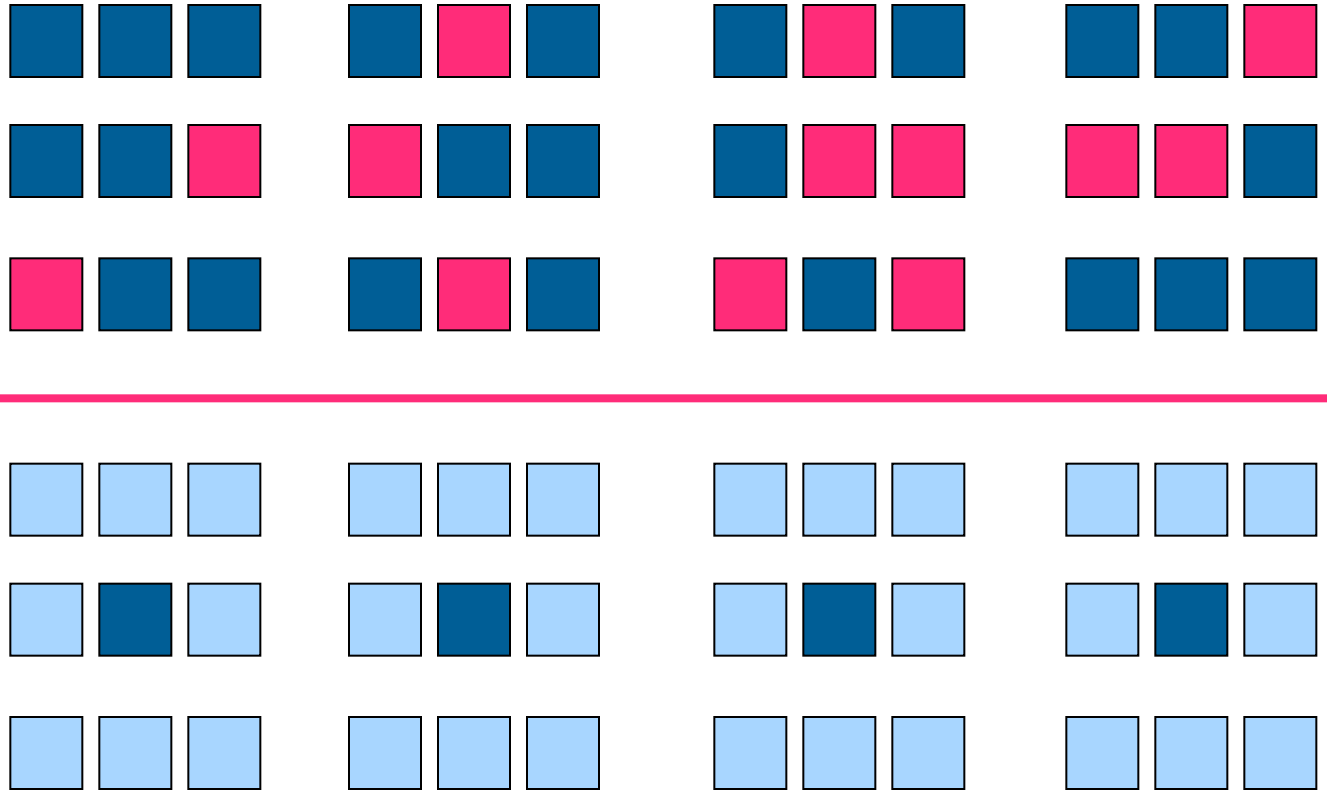
Simple rules to determine state in next iteration:

- If current state of a cell is 'dead', it becomes 'alive' (i.e., turns 'on') if and only if 3 neighbors are alive.
- If current state of a cell is 'alive',
  - a cell with 0 or 1 neighbors alive dies
  - a cell with 2 or 3 neighbors alive lives
  - a cell with 4 or more neighbors alive dies

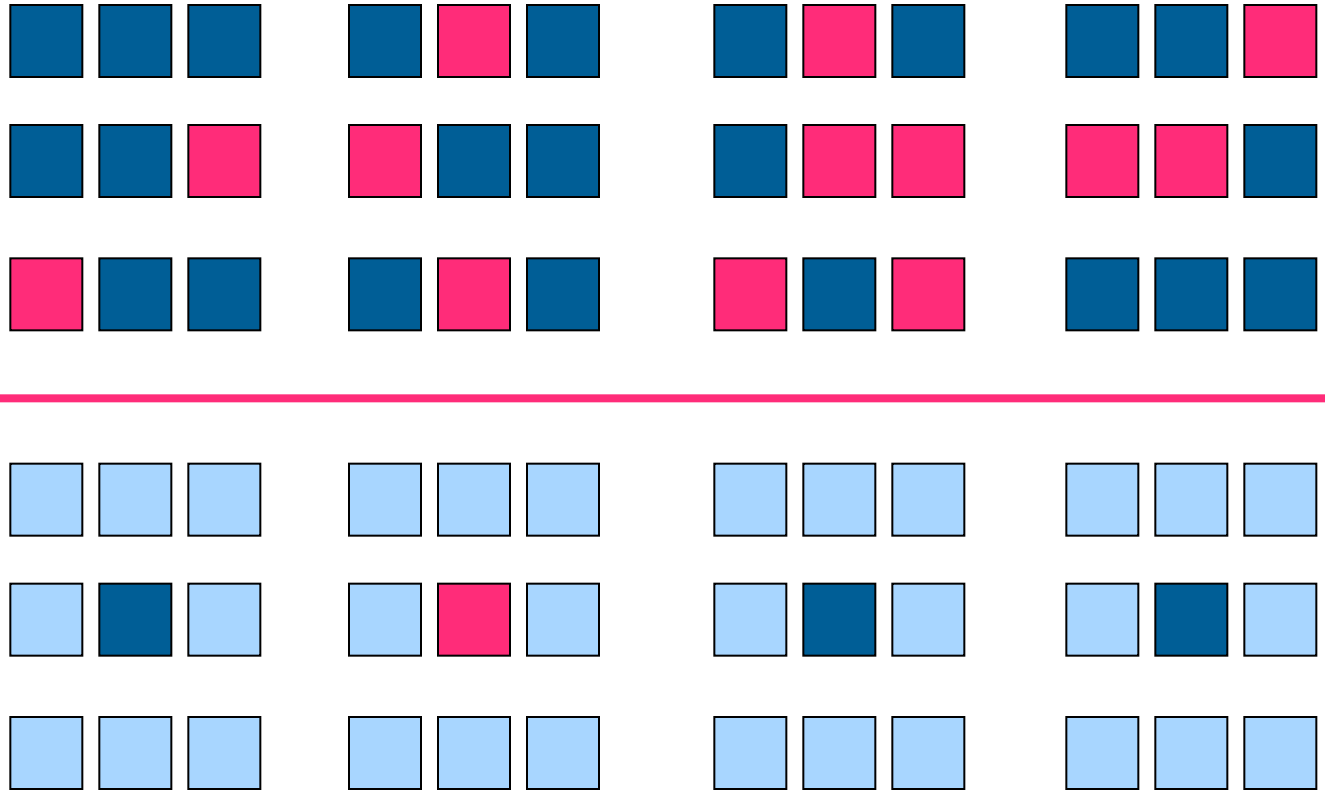
Game of Life



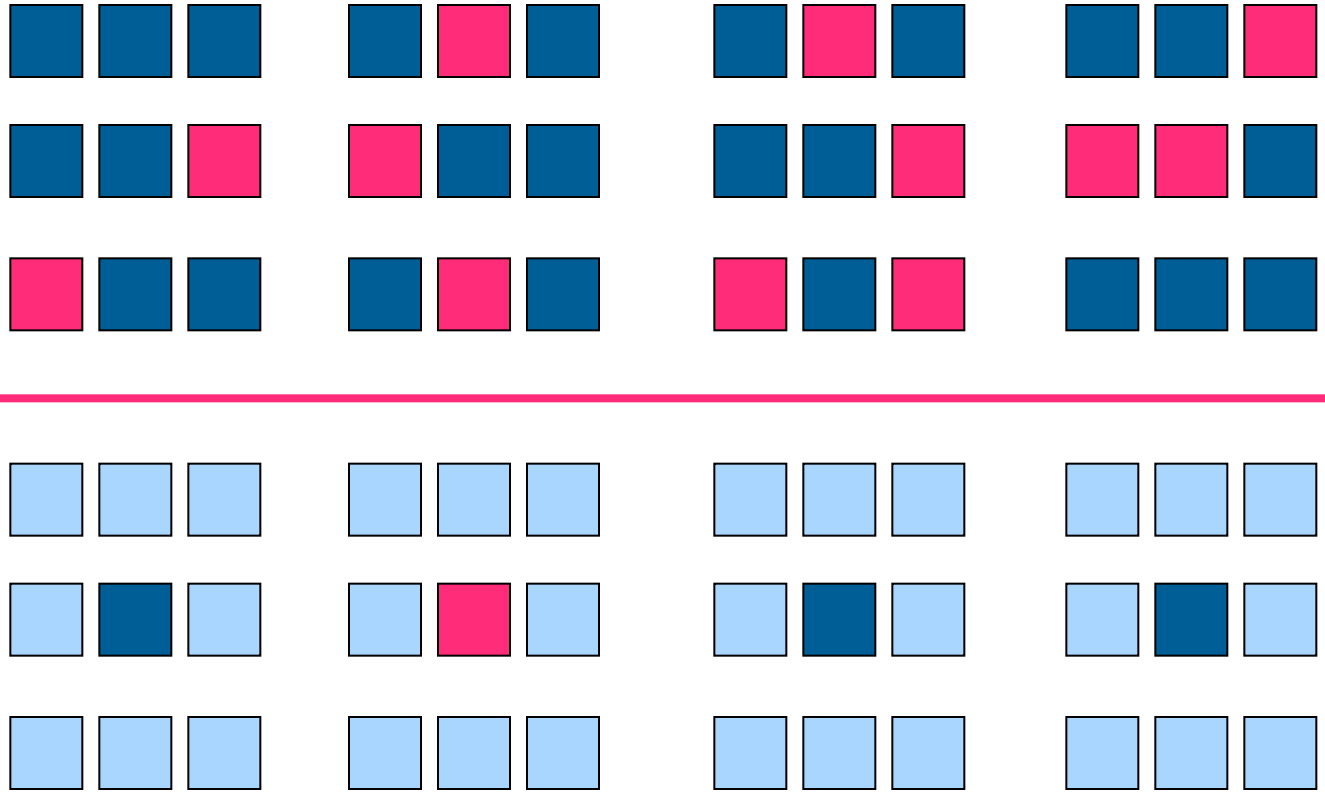
Game of Life



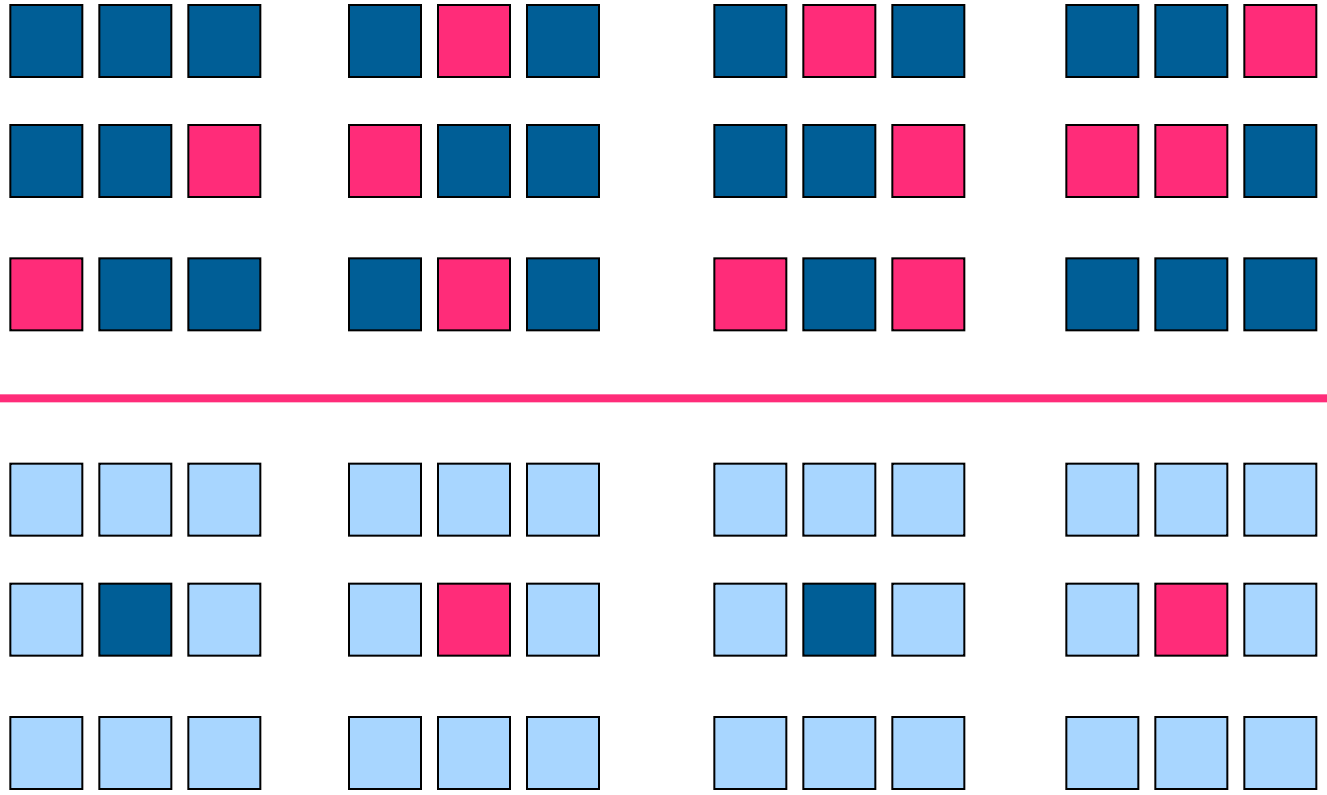
Game of Life



Game of Life



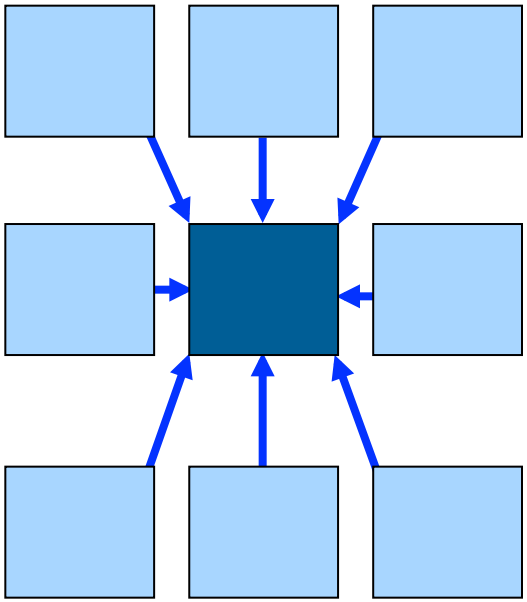
Game of Life

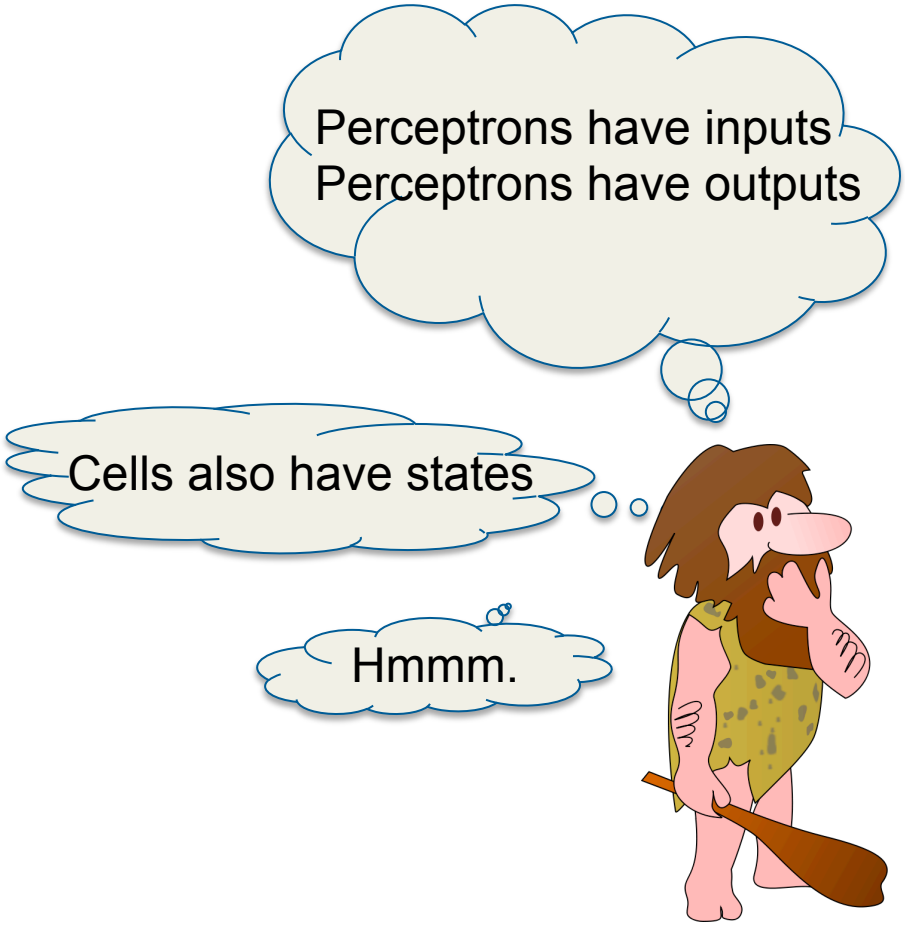


Interesting Patterns Emerge

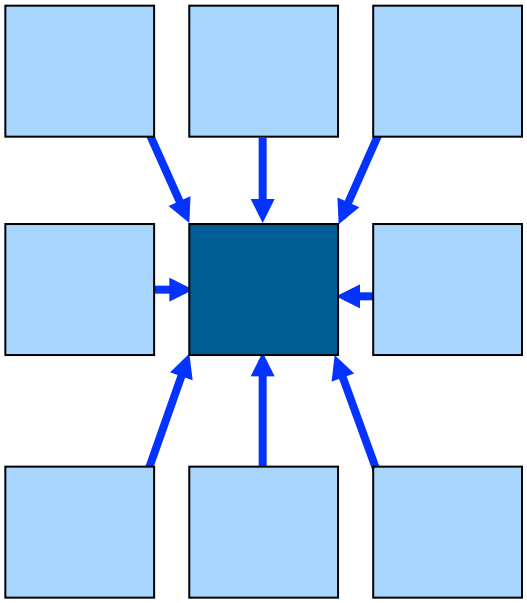
- Glider Guns --- patterns that repeat themselves but are displaced a few ‘pixels’. Thus, they appear to move.
- Space Ships --- another pattern that moves.
- Can perform gating logic.
- Can constitute a Universal Turing Machine.

Suppose a ‘cell’ was a perceptron?

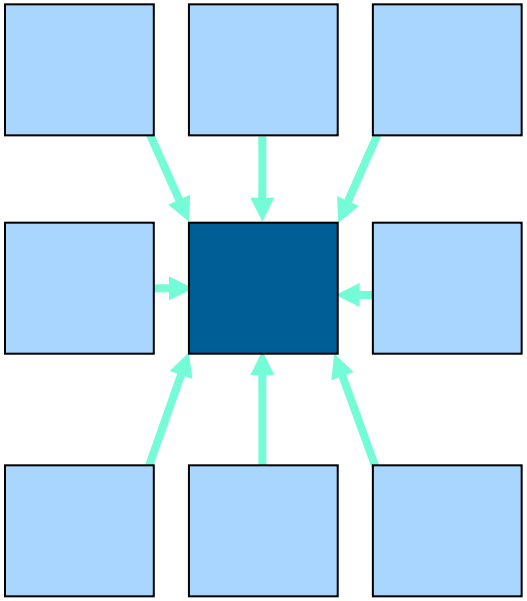




Suppose a ‘cell’ was a perceptron?



A: weighted
state values



B: weighted
output values

Neighborhoods

$$S_r(i, j) = \{C(k, l) | \max_{1 \le k \le M, 1 \le l \le N} \{|k - i|, |l - j|\} \le r\}$$

Cambridge University Press
0521652472 - Cellular Neural Networks and Visual Computing: Foundations and Applications
Leon O. Chua and Tamas Roska

Cell Equations of State

$$\dot{x}_{ij} = -x_{ij} + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y_{kl} + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} + z_{ij}$$

state

output

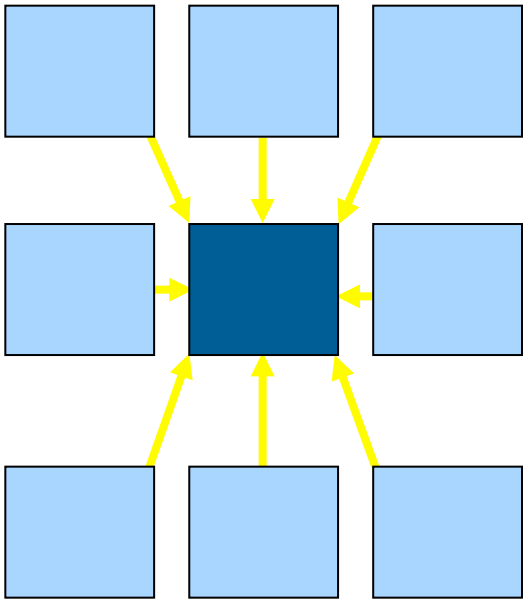
input

Bias/
Threshold

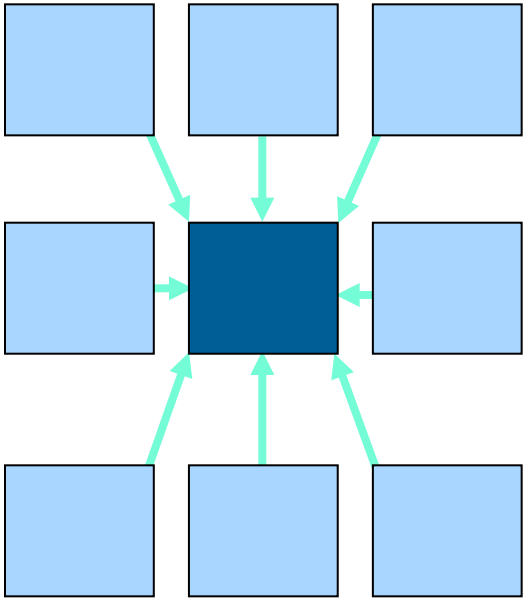
Cambridge University Press
0521652472 - Cellular Neural Networks and Visual Computing: Foundations and Applications
Leon O. Chua and Tamas Roska

How Many CNNs Possible?

A template



B template



If only 2 values are permissible for each link…2^{16} possible CNNs.

Let's Take a Look

http://www.isiweb.ee.ethz.ch/haenggi/CNN_web/CNNsim_adv.html