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.

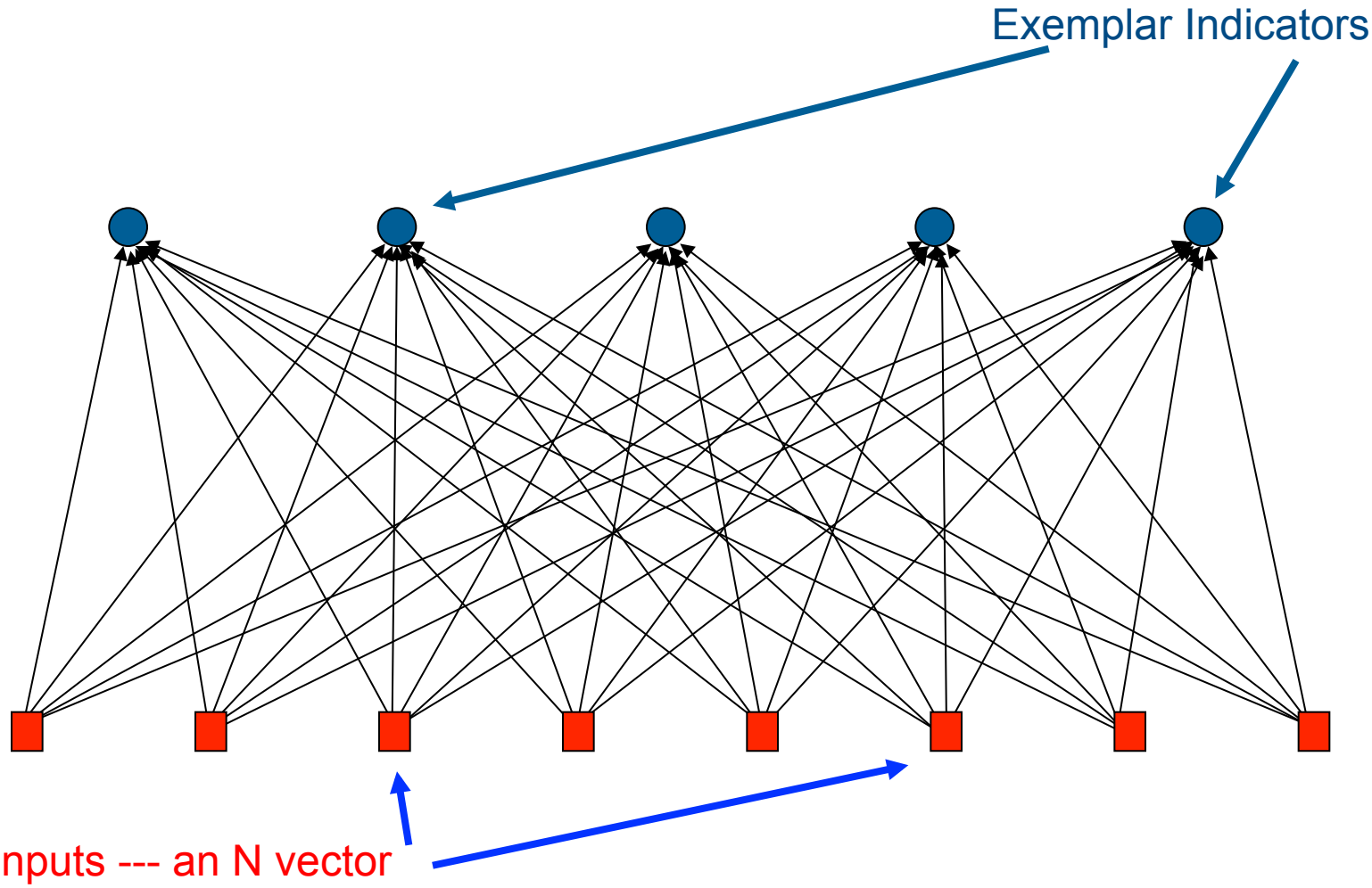
The Hamming Network

- 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



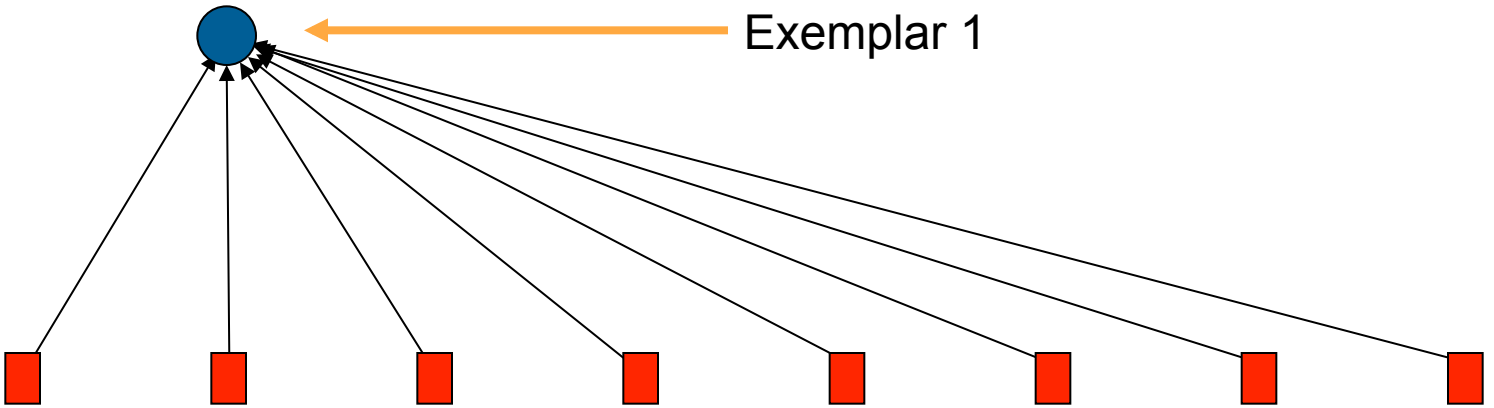
The Hamming Network

- 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 …

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

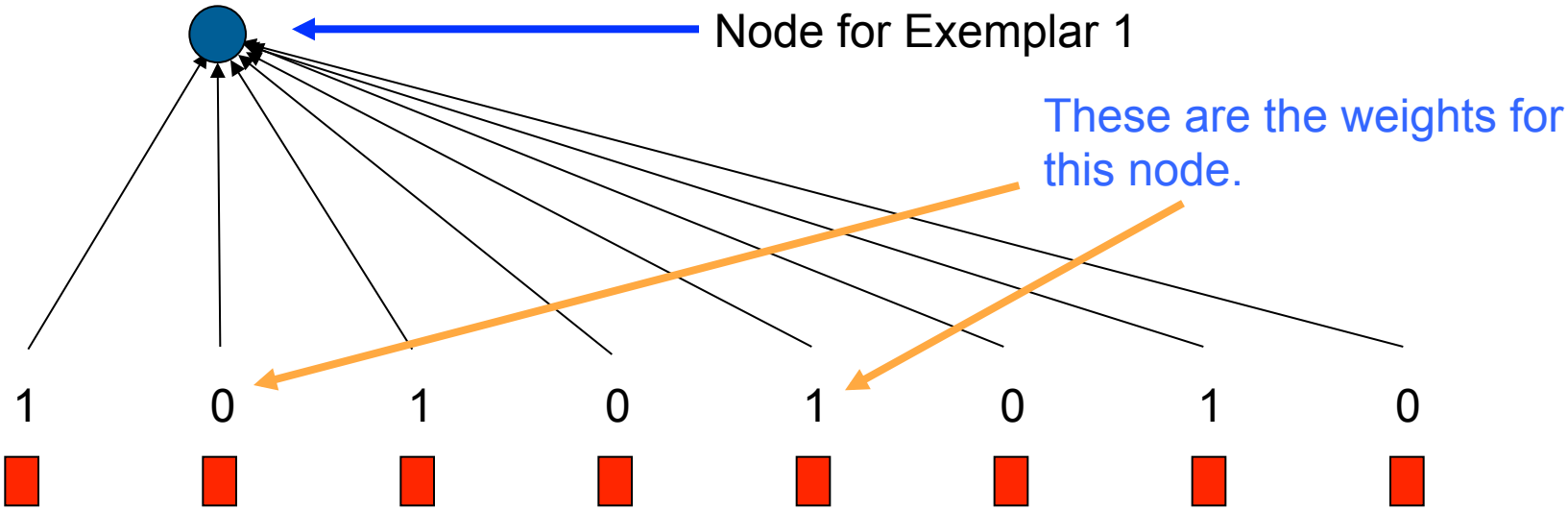
where x_i^j is the i^{th} element of exemplar j.

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

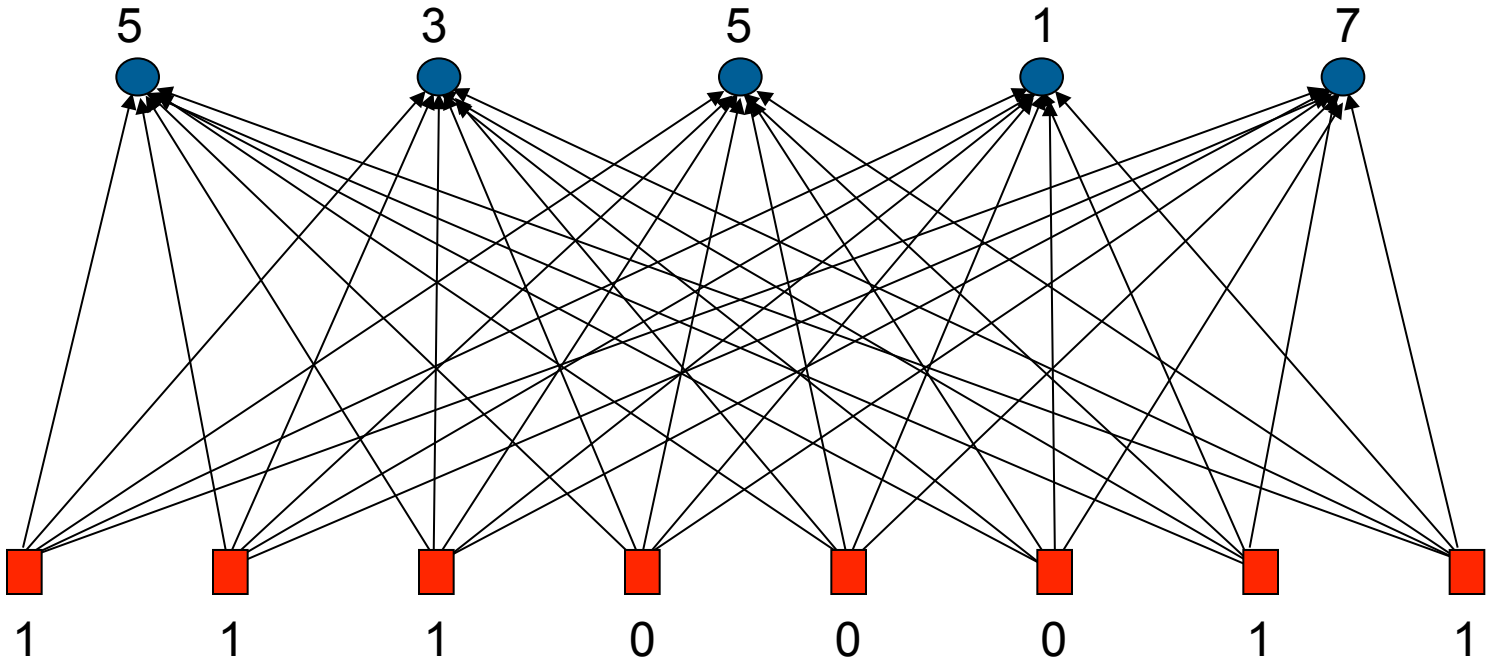
Using the Weights

- 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:

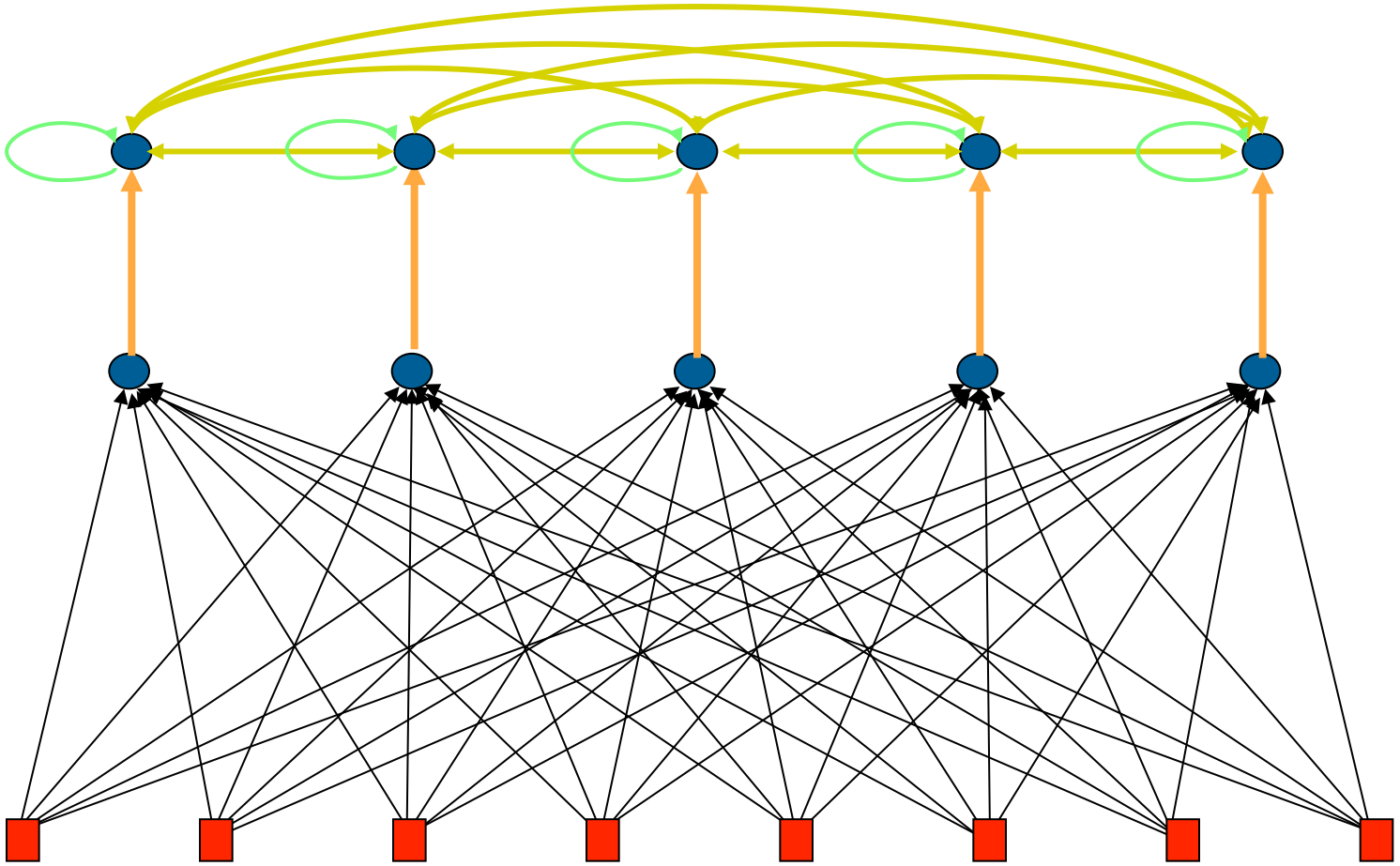
1: 1 0 1 0 1 0 1 0
2: 0 0 0 0 1 1 1 1
3: 1 1 1 1 0 0 0 0
4: 0 0 1 1 1 1 0 0
5: 1 1 0 0 0 0 1 1



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 $\varepsilon < 1/M$

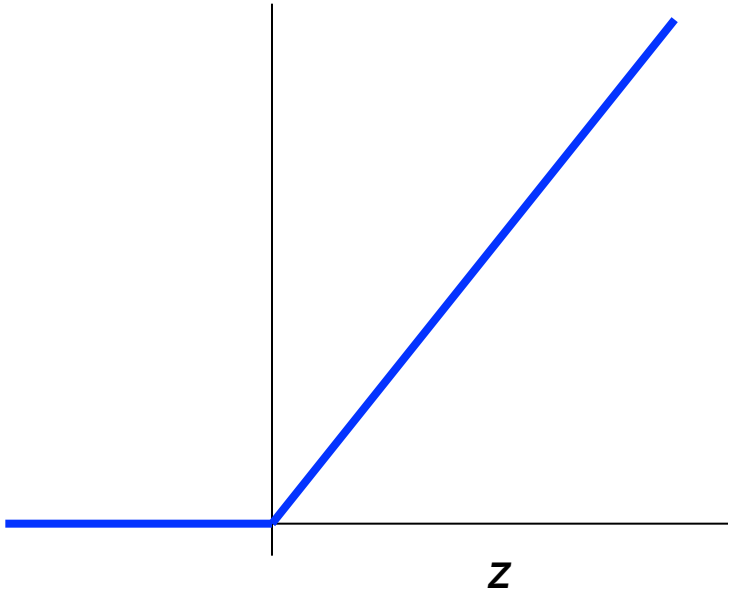
MAXNET

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

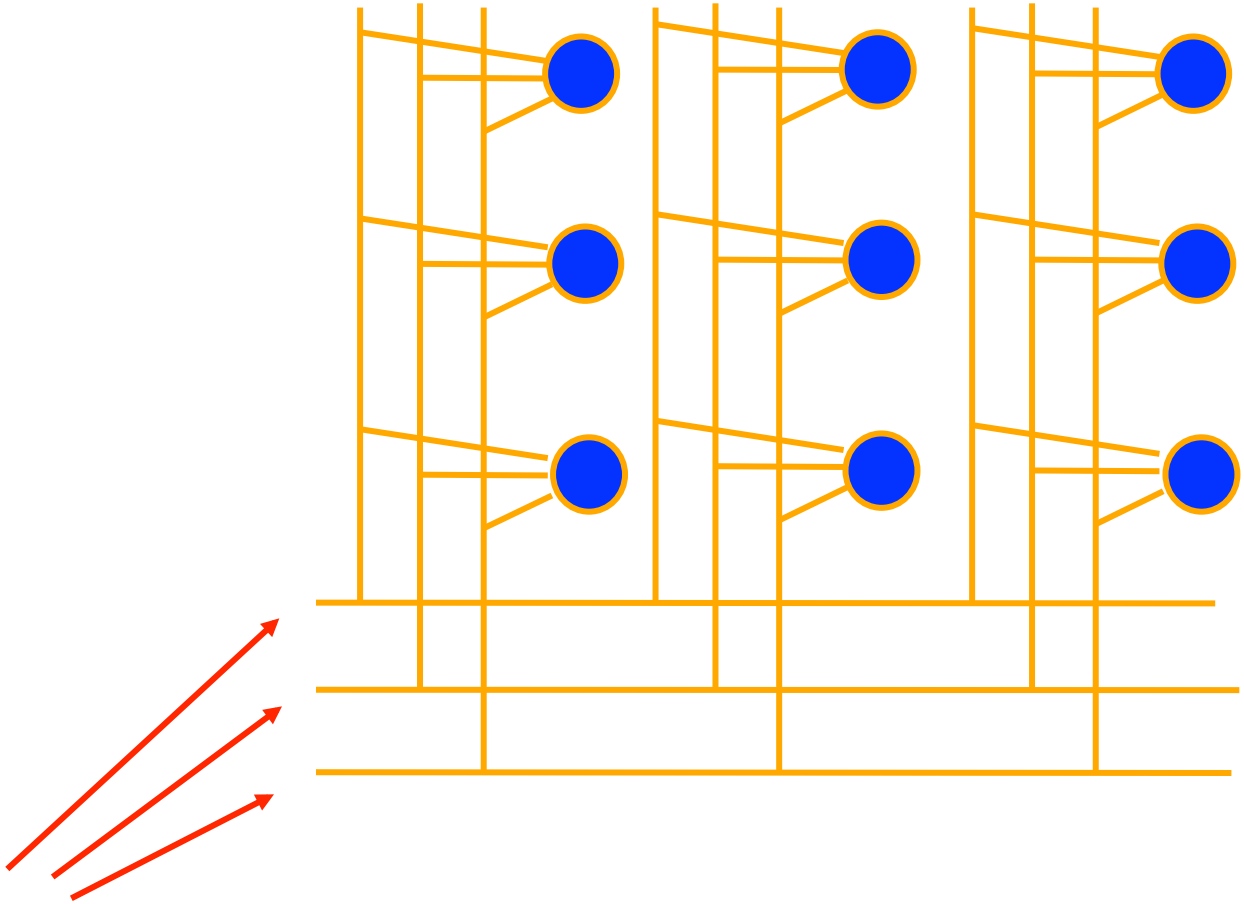
- Self-Organizing Maps (a.k.a. Kohonen Nets)
  - Unsupervised learning.
  - Data causes network to learn.
    - Data itself trains the net.
  - Uses Hamming Net and MAXNET.
  - Tries to create a *topology preserving map* where "near" inputs lead to "near" outputs.
  - Performs a type of feature extraction.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

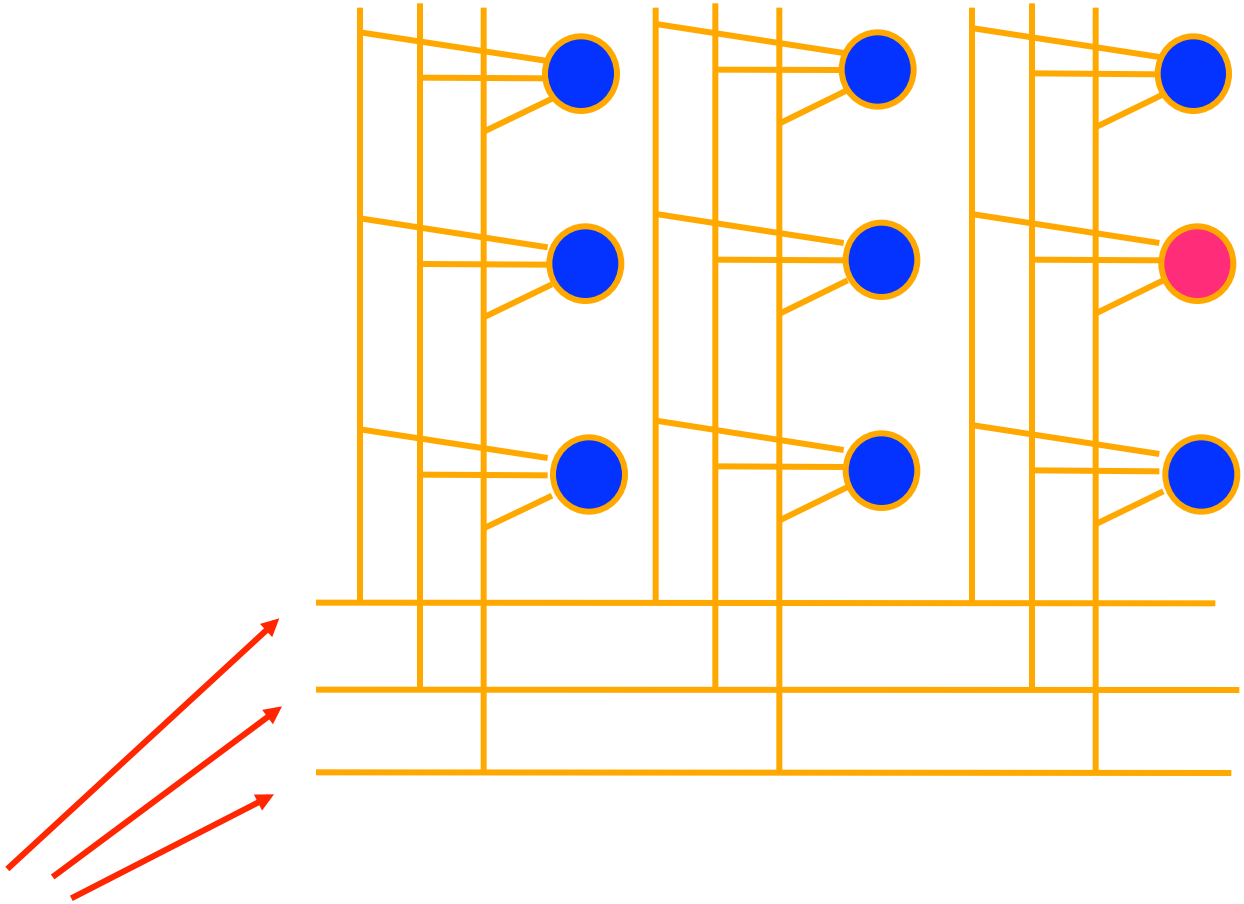
- Given this topology, each node has the 3 inputs **x** (shared with all other nodes) and a weight vector **w** (unique for each node) equal in size to the inputs. Basic idea is that for a given input (here a 3-tuple of possibly real values), with randomly assigned weights **w** for each node), determine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

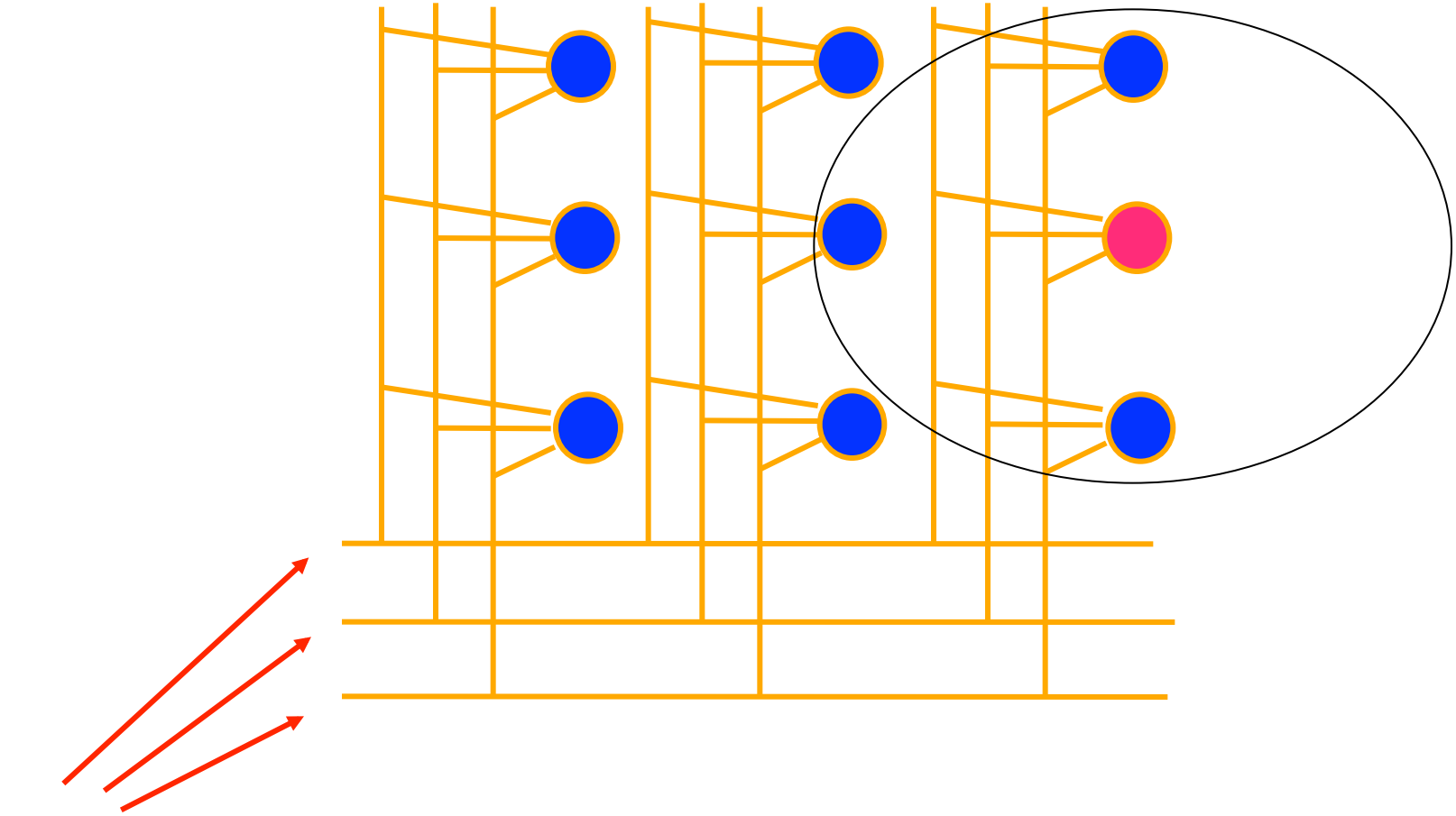


SOM Dynamics

- 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

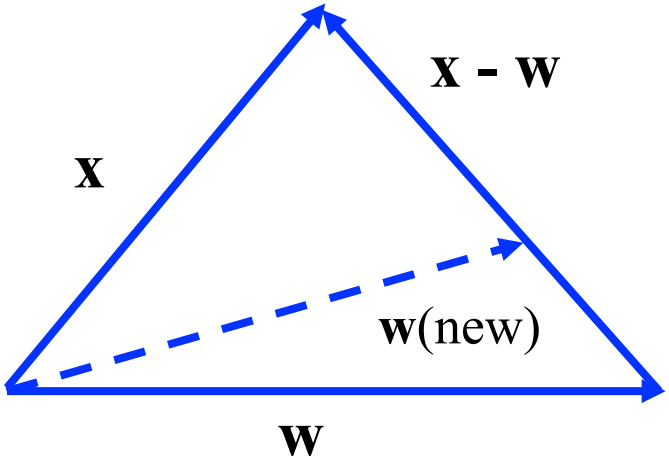
- Next we update all the weights of the neighboring (possibly all) nodes *k* in the array:

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

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):

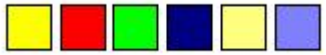


SOM Dynamics

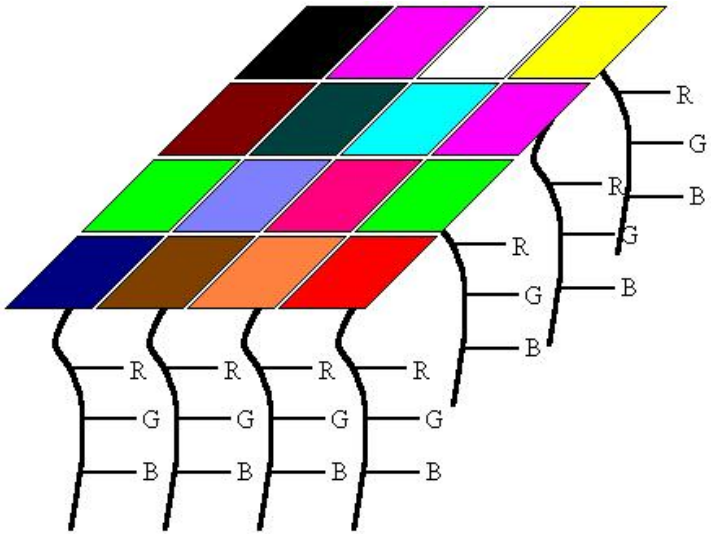
- Other similar update functions are possible, but the overall effect is to make the vector of weights closer to a given input pattern for the winning node and its neighbors.
- Overtime, the neighborhood weighting may lessen, the learning parameter also may decrease.
- Eventually, new input data gets 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….

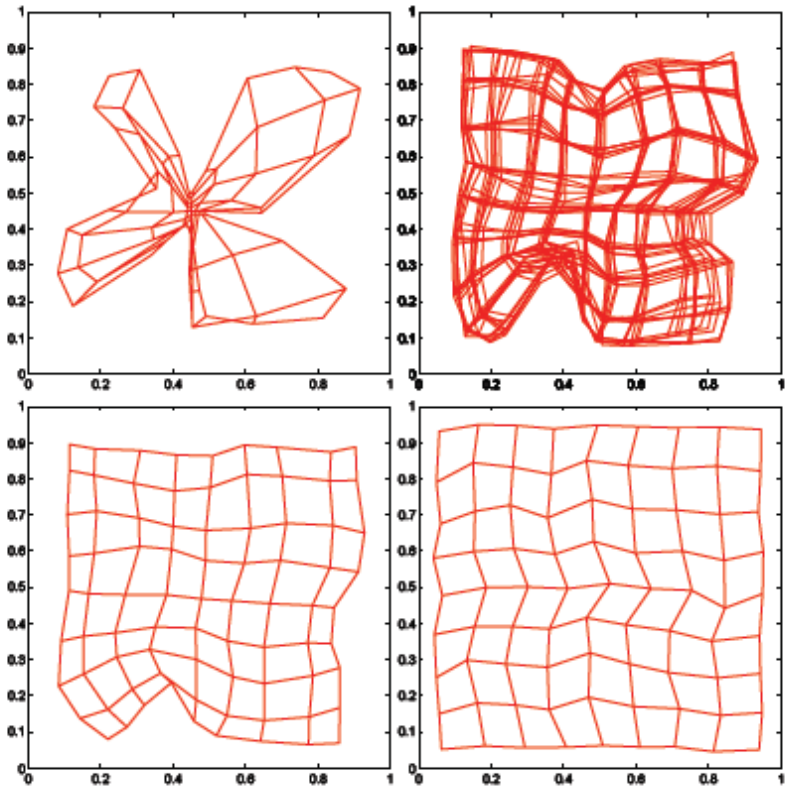


Fig. 15.7. Mapping a square with a two-dimensional lattice. The diagram on the upper right shows some overlapped iterations of the learning process. The diagram below it is the final state after 10000 iterations.

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.