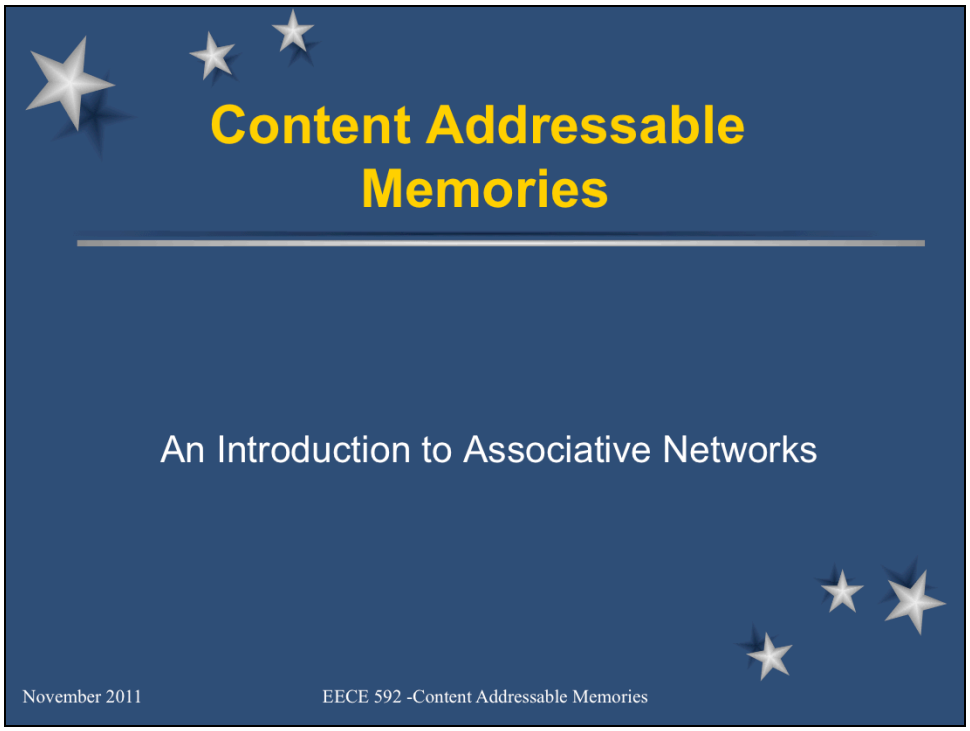
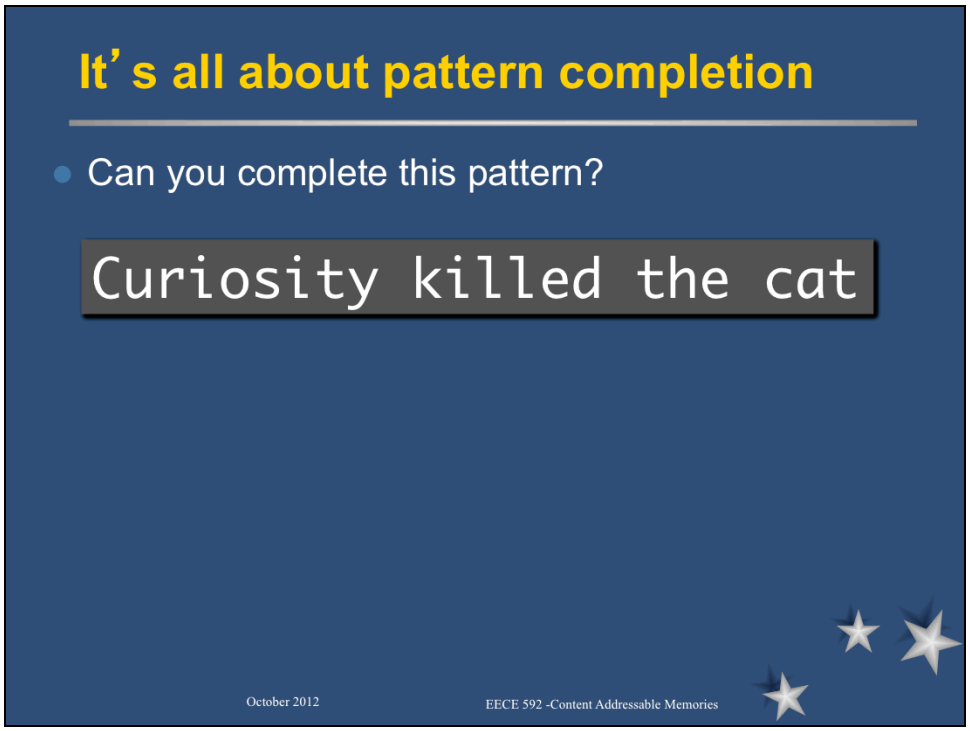


In biological terms, memory refers to the ability of neural systems to store activity patterns and later recall them when required. In humans, association is known to be a prominent feature of memory. In associative or content addressable memory, an output pattern of neural activity is mapped to an input pattern of neural activity. In the retrieval stage, the input can be presented with a noisy or incomplete version of a stored pattern and used as a trigger to recall the original pattern.

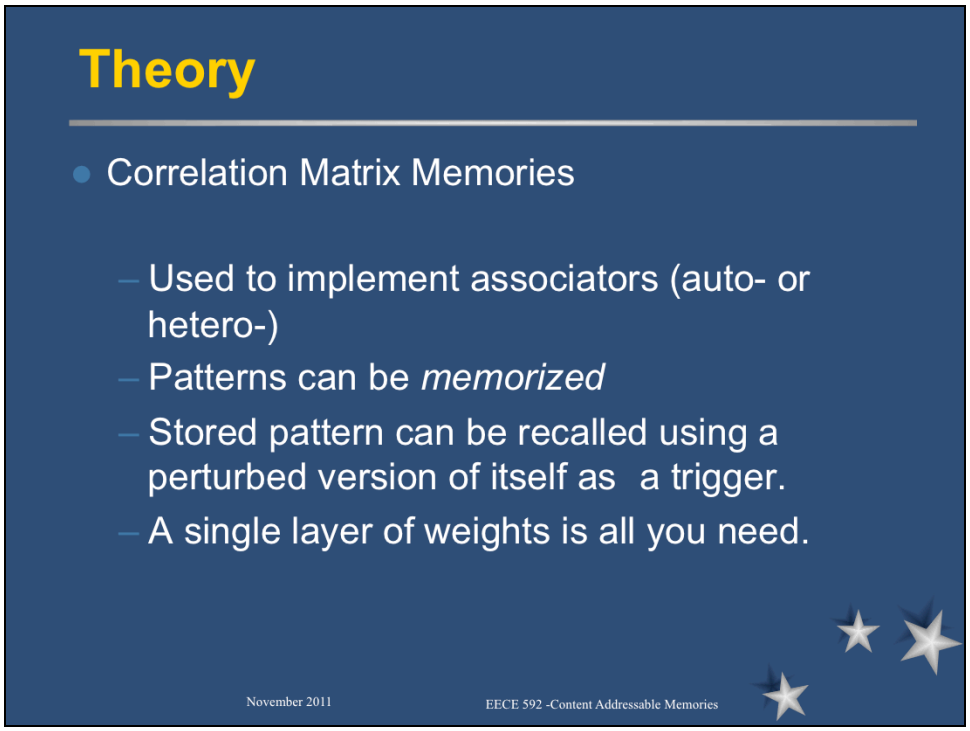
This chapter addresses the theory behind such mechanisms and introduces some of the characteristics of associative networks.



Simple word completion puzzles are characteristic of what content addressable memories can do. As the term suggests, a content addressable memory recalls its contents using the contents as a key. Specifically a partial or noisy version of the contents.

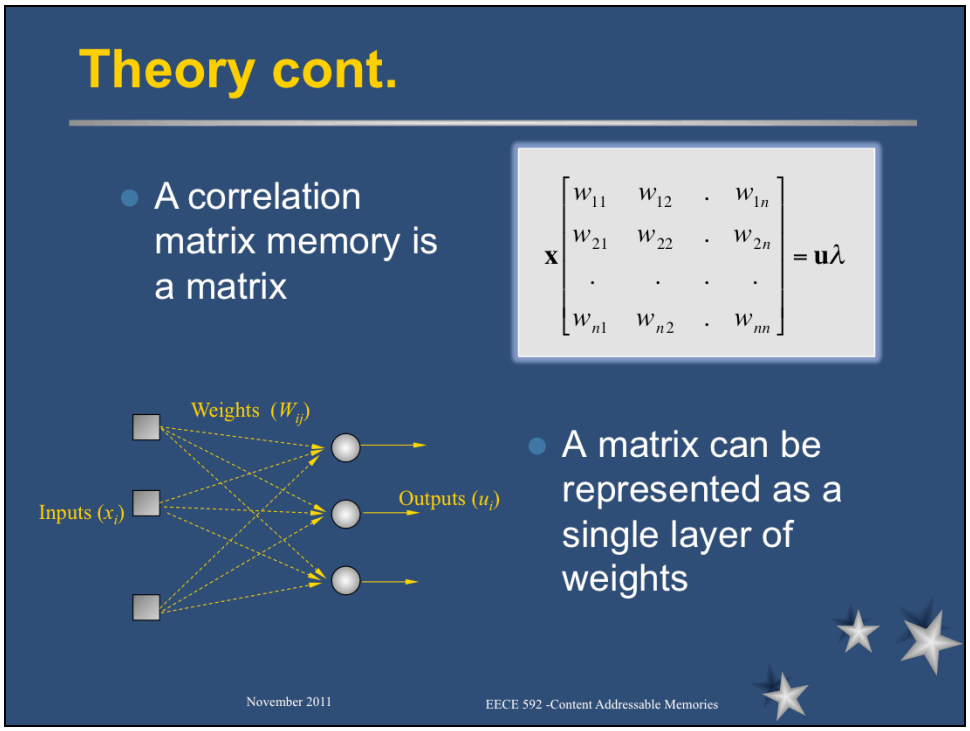
…can be pictures too

… try this too…



The basis of all content addressable memories is the correlation matrix memory.

A correlation matrix memory is the simplest form of content addressable memory. It is a mechanism for *memorizing* patterns and can be represented mathematically. Its main feature is the ability to recall previously memorized patterns from partial or noisy inputs. A single layer of thresholded units can be used to set up a matrix memory.



In matrix terms, the memory can be described as follows :

$$\mathbf{x}\begin{bmatrix} w_{11} & w_{12} & . & w_{1n} \\ w_{21} & w_{22} & . & w_{2n} \\ . & . & . & . \\ w_{n1} & w_{n2} & . & w_{nn} \end{bmatrix} = \mathbf{u}\lambda$$

or

$\mathbf{xw} = \mathbf{u}\lambda$

where ***x*** is the input vector

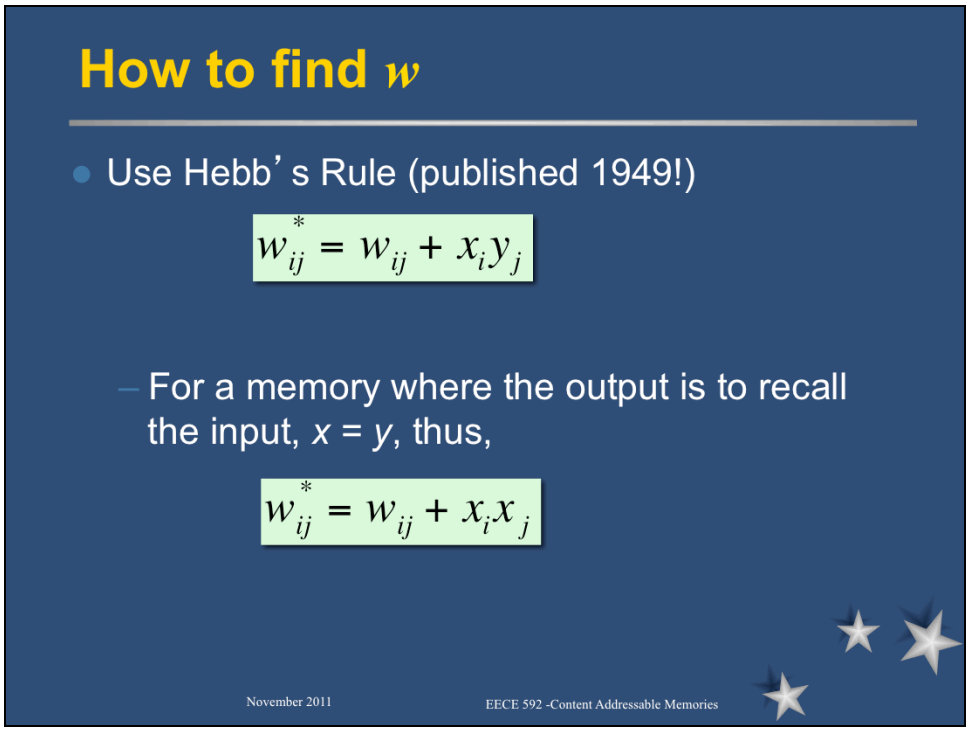
w is the weight matrix

u is the output vector

For a simple memory in which the output is to recall the input, the weight matrix will be square i.e.

w is the correlation matrix.

λ is the eigenvalue and is non-zero for valid solutions.



How to find ***w***?

This is based on Hebb's rule. Published as long ago as 1949, this rule suggested that learning between real biological neurons was a simple matter of reinforcing the connection between two neurons if their outputs were simultaneously coherent and weakening it if not. I.e. if the outputs of neurons *x* and *y* are both positive, or both negative, increase the weight, otherwise, decrease it. The mathematical formulation of this is shown above.

Implementing Hebb's Rule

- Hebb's rule is equivalent to outer products
  - Example to store one pattern $\mathbf{x} = [1 \quad 1 \quad 1 \quad -1]$
  - Compute the outer product $\mathbf{W} = \mathbf{x}^T \mathbf{x}$

$$W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix} \Rightarrow W = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

$w_{ij}^{*} = w_{ij} + x_i x_j$ becomes the weight update step in determining the weight matrix from $\boldsymbol{x}$. This is mathematically equivalent to the outer product rule, which can be used to compute $\boldsymbol{w}$ too.

Storing p patterns

- To store a single pattern x:

$$\mathbf{W} = \mathbf{x}^T\mathbf{x}$$

- To store p patterns x_p :

$$\mathbf{W} = \sum_p \mathbf{x}_p^T\mathbf{x}_p$$

The weight matrix to store a single pattern is simply the outer product of x. This is x transposed as a column vector multiplied with x as a row vector.

To compute the weight matrix that stores multiple patterns, simply compute the outer product for each pattern and then sum the resulting weight matrices.

Some points to bear in mind.

- Neurons used in output layer use a linear activation function.
- Bipolar representation is assumed. If not, patterns will not be of equal length and we cannot use linear threshold.

Storage Capacity

- Mutually orthogonal means (scalar product)

$$\mathbf{x}_k \mathbf{x}_p^T = 0$$

  - for all $p \neq k$

- For n dimensions
  - n -1 mutually orthogonal patterns can be stored
  - attempting to store n patterns results in a singular matrix (0)

November 2013 EECE 592 -Content Addressable Memories

Example - A singular matrix

$n = 4$ Attempt to store 4 mutually orthogonal patterns

Pattern	**Weight Matrix**
$[+1 \quad +1 \quad -1 \quad -1]$	$\begin{bmatrix} 0 & +1 & -1 & -1 \\ +1 & 0 & -1 & -1 \\ -1 & -1 & 0 & +1 \\ -1 & -1 & +1 & 0 \end{bmatrix}$
$[-1 \quad +1 \quad +1 \quad -1]$	$\begin{bmatrix} 0 & -1 & -1 & +1 \\ -1 & 0 & +1 & -1 \\ -1 & +1 & 0 & -1 \\ +1 & -1 & -1 & 0 \end{bmatrix}$
$[-1 \quad +1 \quad -1 \quad +1]$	$\begin{bmatrix} 0 & -1 & +1 & -1 \\ -1 & 0 & -1 & +1 \\ +1 & -1 & 0 & -1 \\ -1 & +1 & -1 & 0 \end{bmatrix}$
$[+1 \quad +1 \quad +1 \quad +1]$	$\begin{bmatrix} 0 & +1 & +1 & +1 \\ +1 & 0 & +1 & +1 \\ +1 & +1 & 0 & +1 \\ +1 & +1 & +1 & 0 \end{bmatrix}$
	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

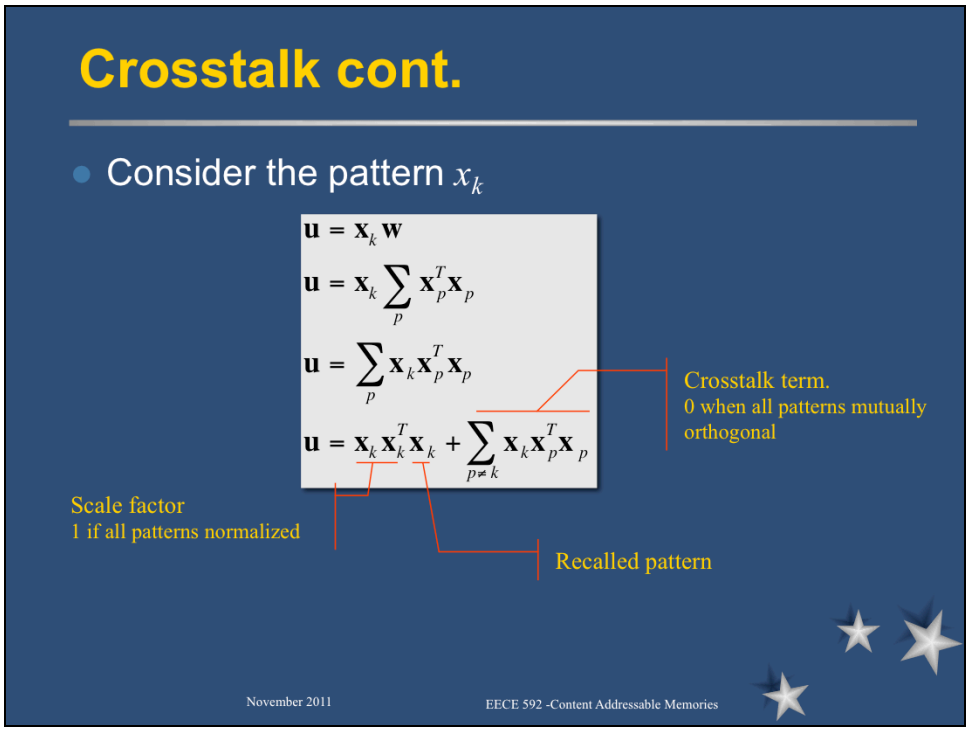
Crosstalk

- ….or what happens if patterns are not mutually orthogonal
- For non-thresholded units:

$$\mathbf{u} = \mathbf{x}_k \mathbf{w}$$

- Matrix constructed via:

$$\mathbf{w} = \sum_p \mathbf{x}_p^T \mathbf{x}_p$$



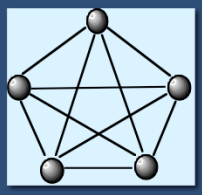
The second term is the *crosstalk* term. If $\mathbf{x}_k \mathbf{x}_p^T = 0$ for all p≠q (i.e. all patterns are mutually orthogonal) then the crosstalk term is 0 and $\mathbf{u} = \mathbf{x}_k \mathbf{x}_k^T \mathbf{x}_k$ where $\mathbf{x}_k \mathbf{x}_k^T$ is simply a scale factor applied to the target $\mathbf{x}_k$

If all patterns have been normalized to unit length, then obviously the scale factor will always be 1. In this case, the outputs do not need an activation function (or rather activation function is linear).

Typically, binary (or bipolar) patterns will not be of unit length. In this case, a thresholding function is necessary to reproduce the stored pattern. Note that thresholding can help when crosstalk is small.

Autoassociation

- Autoassociator cells are both inputs and outputs.
  - Typically used in an iterative manner.



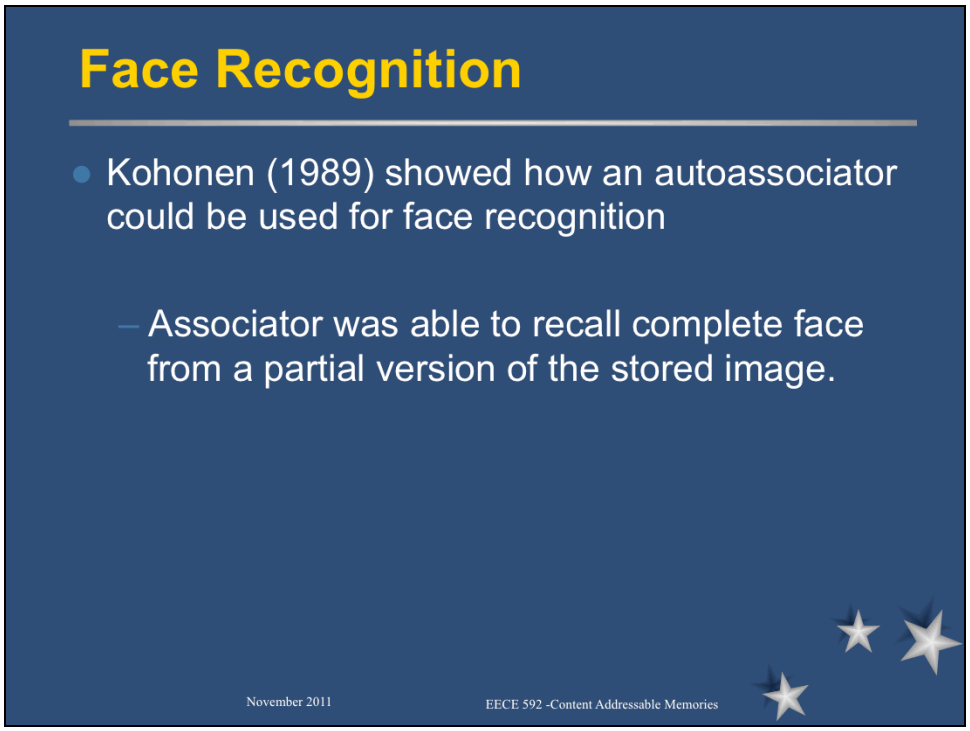
- Autoassociators are *content-addressable*.

November 2011 EECE 592 -Content Addressable Memories

Autoassociator cells are both input and output cells and are usually fully interconnected, even to themselves. By saying that the cells are both input and output cells, means that the current input is used for computation of the next activation. The cells are recurrent. Iterations can be synchronous where all outputs are updated at the same time, or asynchronous where a single output is updated one at a time. This output then becomes available for computing the next update.

Autoassociators are trained to store a set of patterns in such a way that when presented with a new pattern, the network responds by producing whichever one of the stored patterns most closely resembles the input.

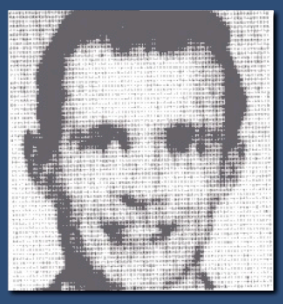
Effectively, autoassociators are *content-addressable* memories that can recover a full memory from a partial or noisy version of it presented at the inputs.



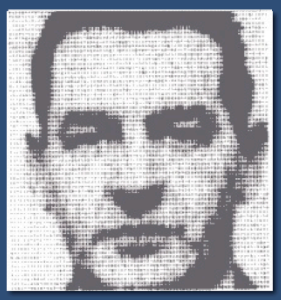
The most famous work illustrating their use was performed by Kohonen (1988) who used the technique to retrieve faces given just parts of the image of the face. In this application, a large grid of cells were used such that the position and output of a cell represented the position and value of one of many pixels belonging to the image.

The initial activation of the grid of 'pixels' is set to represent the input pattern; typically an incomplete portion of a stored face. The activations of all pixels are recomputed based on these initial values. Several activation cycles may be necessary to fully recall the complete face, where each cycle updates activations of all cells.

Face Recognition cont.



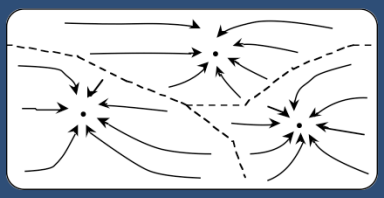
- Recall via partial version of stored pattern



- Stored patterns

Autoassociation cont.

- Trained patterns act as attractors



The above diagram illustrates storage of "memorized" patterns by an associator. Each stored pattern can be thought of as an "energy well" or attractor. Here the three points each represent a trained pattern acting as an attractor.

Linear associators

- Simplest form when activation function is linear

$$u_i = S_i = \sum_j w_{ij} u_j$$

- Each training vector must be of unit length.

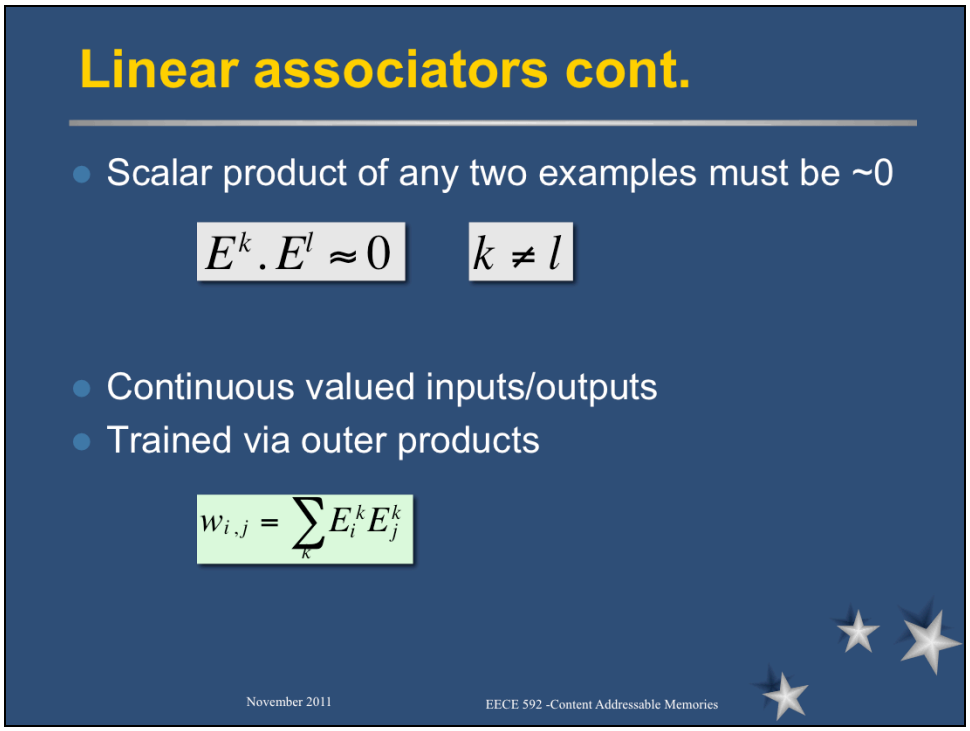
When the output of a cell is not thresholded its activation becomes linear. This permits patterns involving continuous outputs and hence continuous inputs to be learned!

$$u_i = S_i = \sum_j w_{ij} u_j$$

for which a unit may have a connection to itself, i.e.

w_{ij} may be non-zero.

However, we have to now worry about the length of the input vectors. Can you think why?



However, a requirement is that each training *vector* must be of unit length and that the scalar or inner product of any two examples E^k and E^l be close to zero. Restricting inputs to unit length ensures that the $\mathbf{x}_k \mathbf{x}_k^T$ term that scales the output remains at unity.

ie. $E^k . E^l \approx 0$

for $k \neq l$

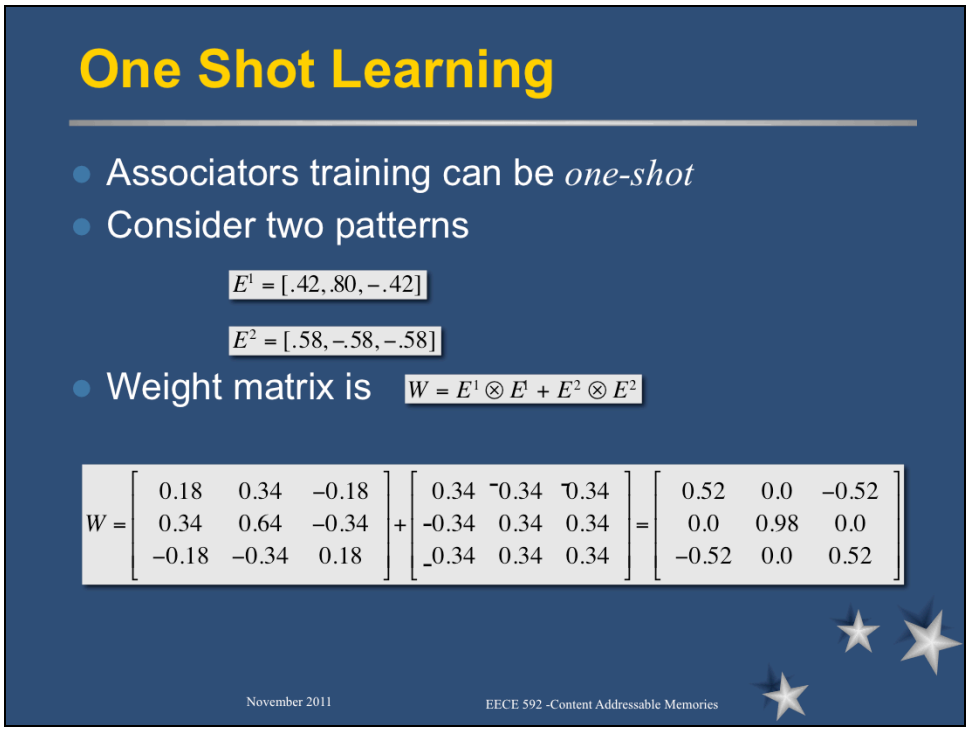
The cells have continuous values and are trained by summing the outer product of each training vector with itself to produce a matrix of connection weights W.

so:

$$W = E^1 \otimes E^1 + E^2 \otimes E^2 + \ldots + E^n \otimes E^n$$

or

$$w_{i,j} = \sum_k E_i^k E_j^k$$



The learning approach for content addressable memories is also referred to as "one shot learning" because it is non-iterative and the required set of weights can be found in one step.

Consider two patterns:

$E^1 = [.42, .80, -.42]$

$E^2 = [.58, -.58, -.58]$

For which $\|E^1\| \approx \|E^2\| \approx 1$ and $|E^1 . E^2| \approx 0.2 \approx 0$

(Remember, since the outputs are not thresholded, the unit length must be 0, otherwise actual output will be scaled.)

One Shot Learning cont.

- Now attempt recall with the weight matrix

$$W = \begin{bmatrix} 0.52 & 0.0 & -0.52 \\ 0.0 & 0.98 & 0.0 \\ -0.52 & 0.0 & 0.52 \end{bmatrix}$$

$$W.E^1 = [.44, .78, -.44] \approx E^1$$

$$W.E^2 = [.60, -.57, -.60] \approx E^2$$

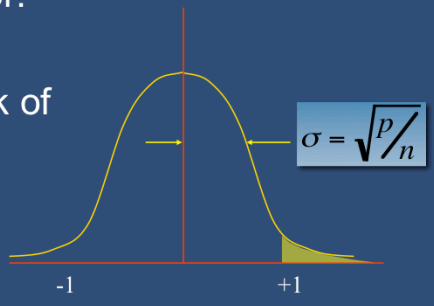
- Try it, it works!

More on Storage Capacity

- Realistically, stored patterns unlikely to be mutually orthogonal.
  - I.e. crosstalk will be non-zero.
  - Storage capacity will be less than n-1
    - (where n is the dimensionality)
- So what is the storage capacity?
  - Based upon statistics

More on storage capacity cont.

- Have to accept a non-zero probability of recall error.
- Assuming that crosstalk of training set follows a gaussion distribution
  - p is # patterns
  - n is dimensionality
- Can define a curve for the set of patterns based upon p & n



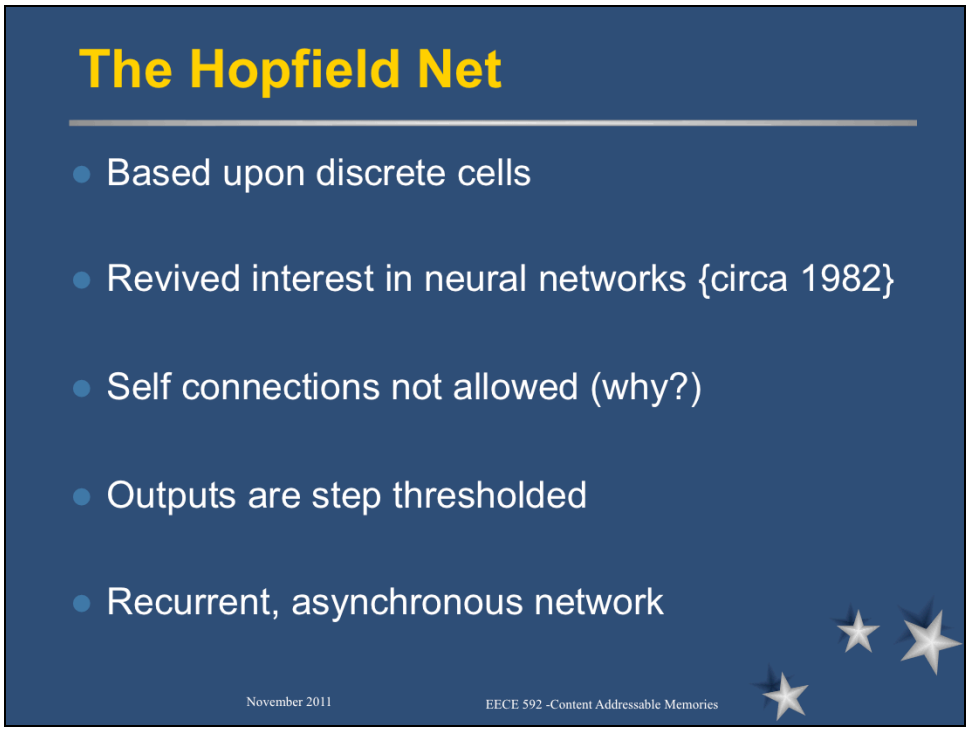
Can attempt to determine what the storage capacity is given an acceptable probability of error in the recalled pattern. Determination is based upon the gaussian distribution of the crosstalk. Ideally, we want the area under the curve to be as small as possible. The shaded area represents the area which is believed to result in an acceptable probability of error.

More on storage capacity cont.

- Customary to accept a less than 1% chance of error in a bit

Probability of error	p_{max} / n
0.001	0.105
0.0036	0.138
0.01	0.185
0.05	0.37
0.1	0.61

- This gives p_{max} = 0.138 n
  - E.g., if dimensionality is 100
  - Max storage capacity is <14 patterns.



Hopfield worked on an autoassociator based upon discrete cells. As a Nobel prize winner, his reputation helped spark interest in neural networks in the mid '80s. As a physicist, not surprisingly, the Hopfield model involved the notion of an energy function.

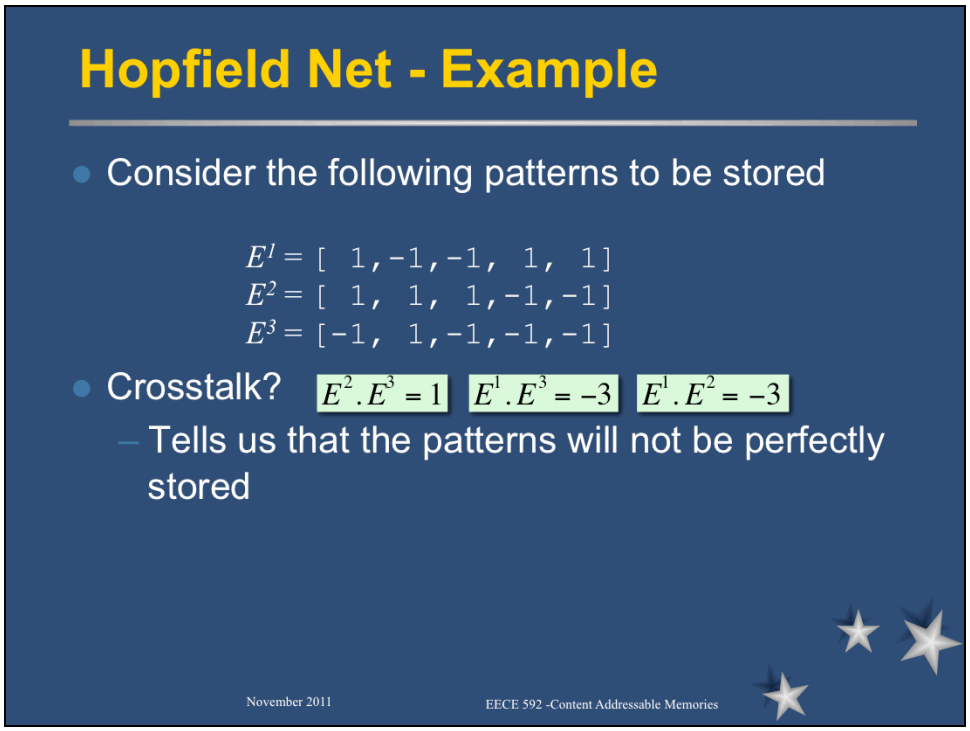
Learning is again by the outer-product rule but in this case, self connections are not allowed. In effect, this means simply setting the leading diagonal of the weight matrix to zero. One potential benefit of this is to avoid the possibility of encountering the identity matrix which of course would be useless.

$$w_{i,i} = 0 \quad \text{for all } i$$

$$S_i(t) = \sum_j w_{i.j} u_j(t) \quad \text{is the weighted sum at time t}$$

and $$u_i(t) = \begin{Bmatrix} +1 & \text{if } S_i \geq 0 \\ -1 & \text{if } S_i < 0 \end{Bmatrix}$$

is used to compute the activations



An example of one-shot learning for the Hopfield Model

Consider three training vectors (patterns to be memorized)

$E^1 = [\ 1,-1,-1,\ 1,\ 1]$

$E^2 = [\ 1,\ 1,\ 1,-1,-1]$

$E^3 = [-1,\ 1,-1,-1,-1]$

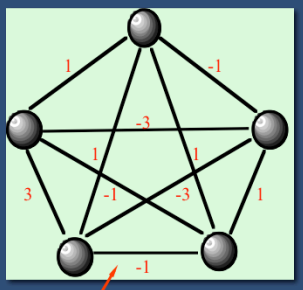
The patterns are not all mutually orthogonal which tells us that they cannot be all perfectly stored.

$\| E^k \|^2 = p = 5$ for all training examples.

Hopfield example cont.

- The weight matrix W is:

$$W = \begin{bmatrix} 0 & -1 & -1 & 1 & 1 \\ -1 & 0 & 1 & -1 & -1 \\ -1 & 1 & 0 & -1 & -1 \\ 1 & -1 & -1 & 0 & 1 \\ 1 & -1 & -1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 1 & -1 & -1 \\ 1 & 0 & 1 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 \\ -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 1 & 1 & 1 \\ -1 & 0 & -1 & -1 & -1 \\ 1 & -1 & 0 & 1 & 1 \\ 1 & -1 & 1 & 0 & 1 \\ 1 & -1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 & 1 & 1 \\ -1 & 0 & 1 & -3 & -3 \\ 1 & 1 & 0 & -1 & -1 \\ 1 & -3 & -1 & 0 & 3 \\ 1 & -3 & -1 & 3 & 0 \end{bmatrix}$$



Notice here that the leading diagonal is zero. This is specified by the Hopfield net which prohibits self connections.

Hopfield Energy

- Hopfield described his model in terms of energy
- Energy of a Hopfield net is:

$$\mathbf{E}(t) = -\sum_{i<j} u_i(t)u_j(t)w_{i,j}$$

- Stored memory patterns, or attractors are low energy *wells*.
- Influenced development of the *Boltzmann Machine*.

November 2011 EECE 592 -Content Addressable Memories

Hopfield's most significant contribution was the idea of an energy function for neural net theory.

The energy E is $\mathbf{E}(t) = -\sum_{i<j} u_i(t)u_j(t)w_{i,j}$

It can be useful to think of energy as a landscape comprised of rolling hills and valleys. The *attractors* (memorized patterns) are then the local minima or valleys of this landscape. Hopfield notion of energy had an important influence on the development of the Boltzmann machine.

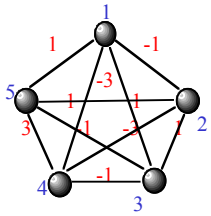
Computation of the energy is simple. It's the outputs multiplied by the weight connecting them summed over the whole net. Consider the network is in the following state (I.e. this is the pattern on the outputs) :

$E^2 = [1, 1, 1, -1, -1]$

Energy $E = -(W_{12}+W_{13}-W_{14}-W_{15}+W_{23}-W_{24}-W_{25}-W_{34}-W_{35}+W_{45})$

E= - (-1 +1 -1 -1 +1 +3 +3 +1 +1 +3)

E= -10



Note that the connections are bi-directional and only need to be summed in one direction.

Hopfield Energy cont.

- Following a state change, energy changes given by:

$$\Delta \mathbf{E} = \mathbf{E}(t+1) - \mathbf{E}(t)$$

$$= -(u_k(t+1) - u_k(t)) \sum_{j \neq k} u_j(t) w_{j,k}$$

$$= -2u_k(t+1) S_k(t)$$

- Thus energy change is -ve indicating that a state change always moves towards a stored pattern.

The energy change

Following a change in the activation, the energy **E** of the net changes as follows:-

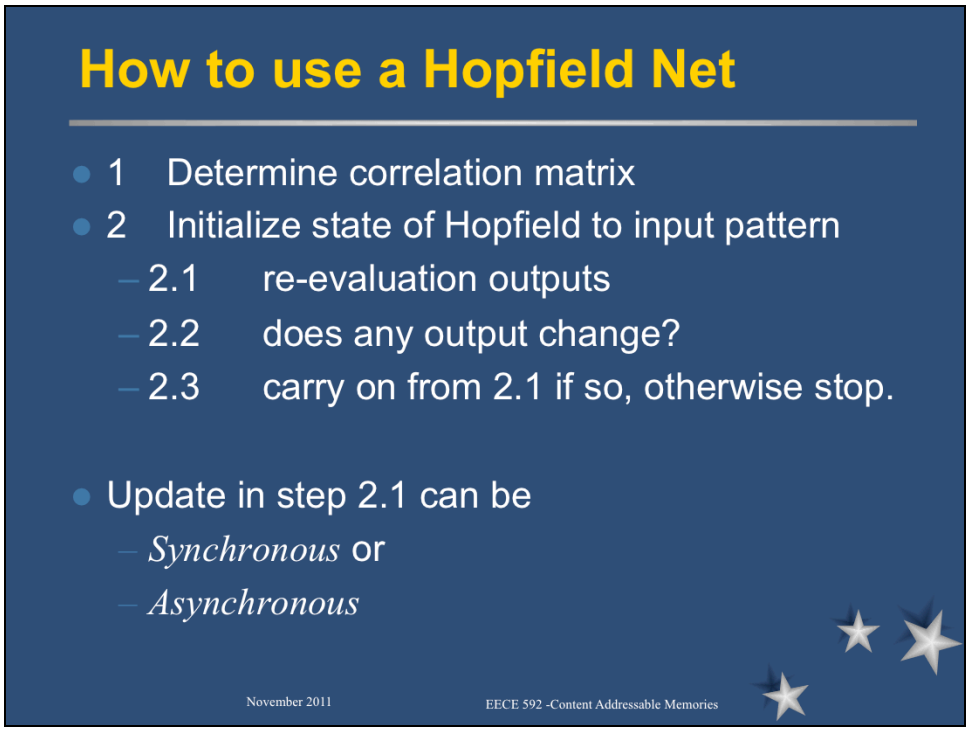
$$\Delta \mathbf{E} = \mathbf{E}(t+1) - \mathbf{E}(t)$$

$$= -(u_k(t+1) - u_k(t)) \sum_{j \neq k} u_j(t) w_{j,k}$$

$$= -2u_k(t+1) S_k(t)$$

$$< 0$$

Showing that the energy decreases at each step. Thus the net is moving towards one of its *attractors* (i.e. memorized pattern).

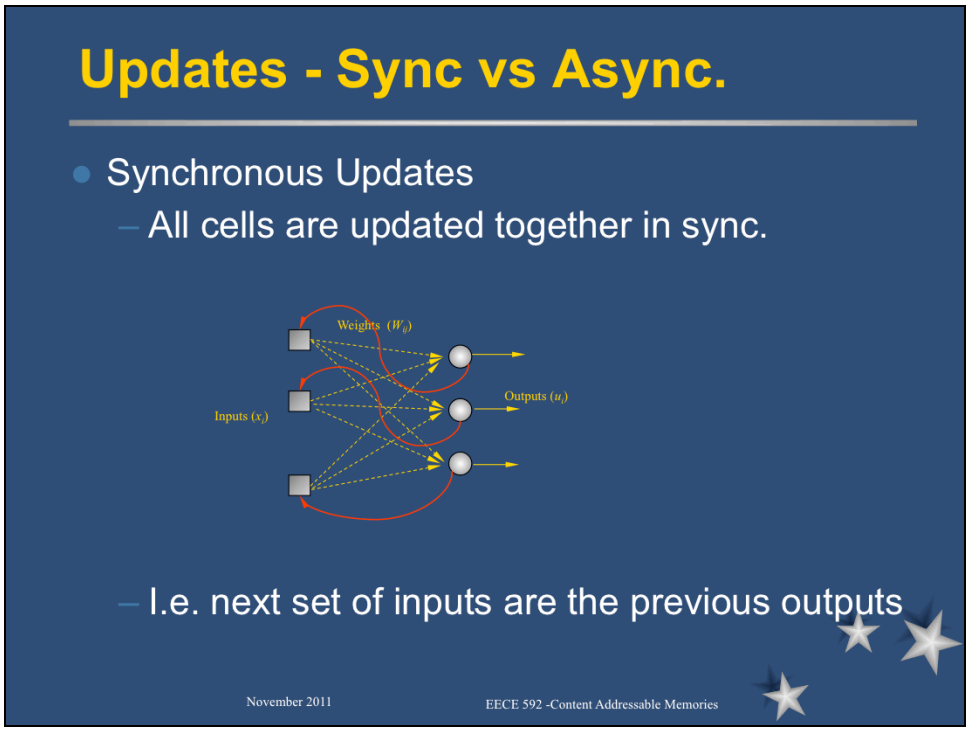


Using a Hopfield Net.

Once a weight matrix is obtained for the set of patterns to be memorized, the net can be presented with input patterns and its outputs re-evaluated.

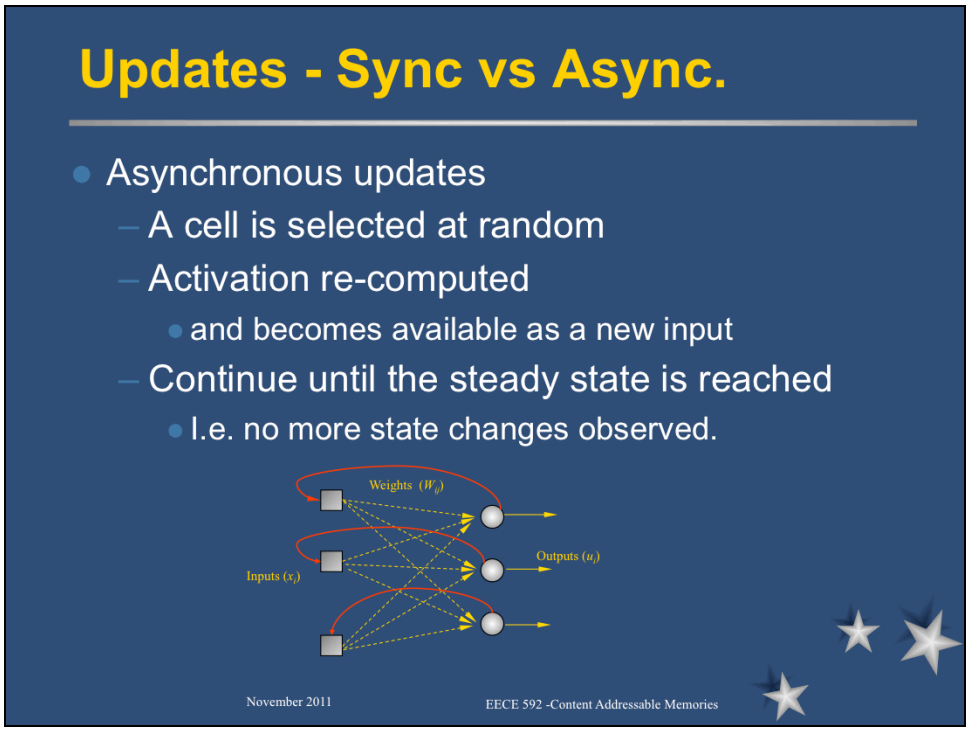
The general idea is that the initial input pattern applied to the Hopfield net is the starting state. The Hopfield net will then try to converge on one of the stored patterns using this initial pattern as a starting state. State changes will cease when a stored pattern has been recalled (a low energy state). Hopfield theory is that each new state reached has a lower energy level than the previous.

Unlike feedforward nets, the Hopfield net uses recurrent connections. Updates to the activation of the cells can be performed either *synchronously* or *asynchronously*.



Synchronous updating.

Synchronous updates require that that all cells be updated together in sync. with a timing signal. Typically however, updates are performed asynchronously for the Hopfield net.



Asynchronous updating.

A cell is selected at random and its activation re-computed as follows:

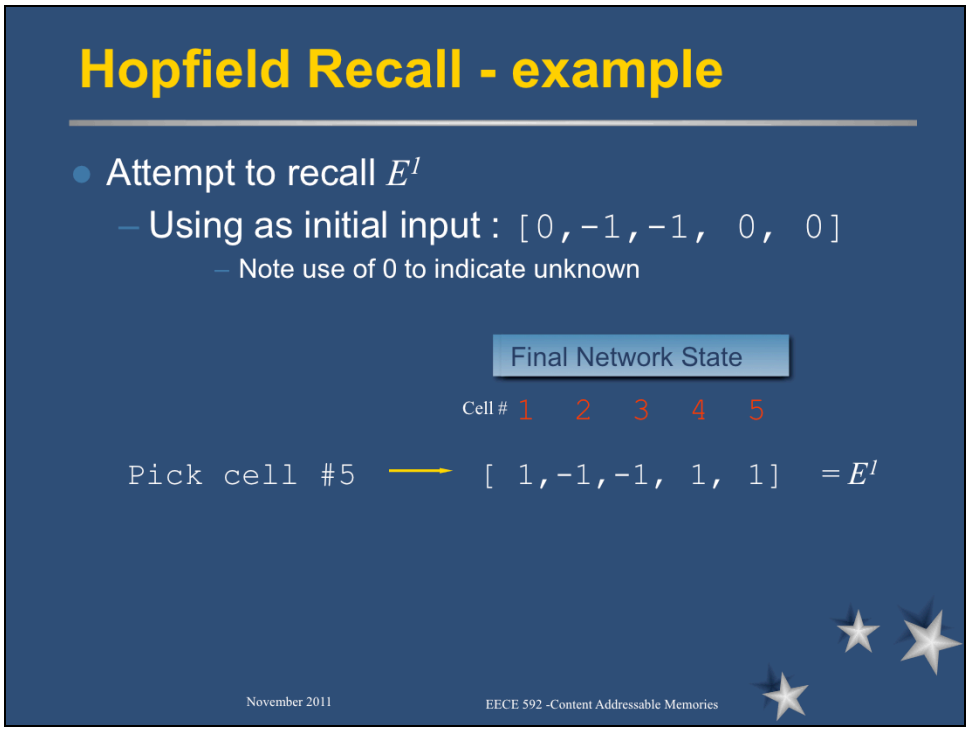
The cell will keep its current value unless :-

$u_k(t) = -1$ and $S_k(t) \geq 0$ then $u_k(t+1) \rightarrow +1$

or

$u_k(t) = +1$ and $S_k(t) < 0$ then $u_k(t+1) \rightarrow -1$

These steps are performed for all cells, in random order, until they all reach the steady state.



Using a Hopfield Net.

In the previous example, the weights to memorize three 5 dimensional inputs were computed. The process of recall can be illustrated by following the steps involving an incomplete pattern. The pattern chosen is a partial version of :

$E^{initial}$= [0,-1,-1, 0, 0]

cell# 1 2 3 4 5

Note the use of 0 values to indicate "unknown" variables. (A benefit of bi-polar representation over standard boolean). This partial version of is used to set the initial activations of the cells in the networks.

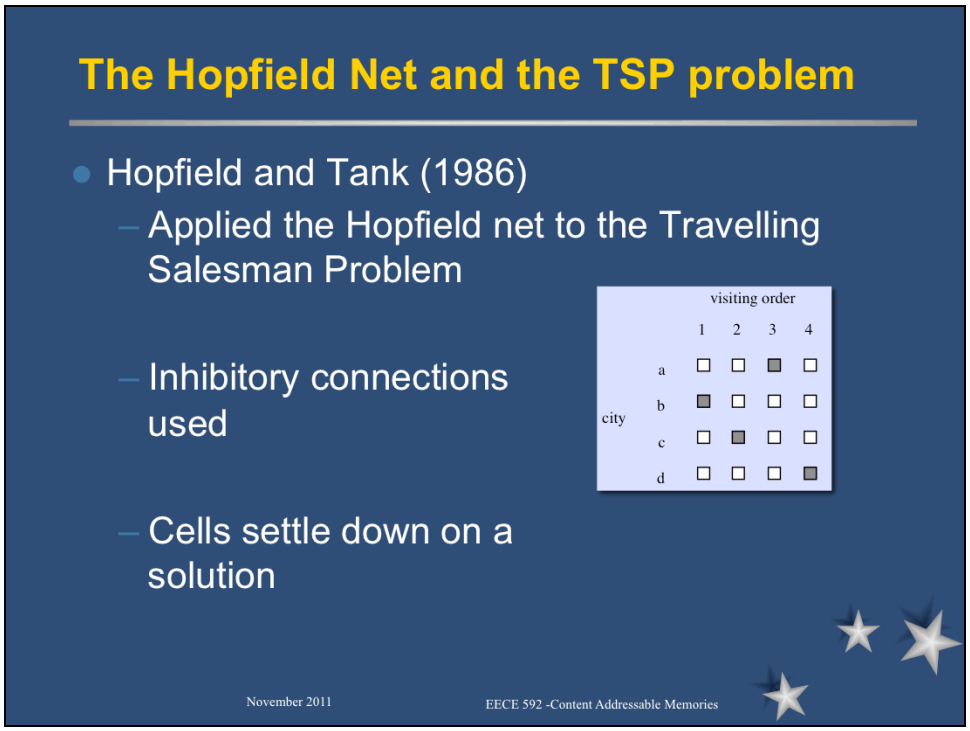
Initial network state = [0,-1,-1, 0, 0]

Let's randomly pick and re-compute the output of neuron (for output) 1.

The output of the cells now becomes → network state = [1,-1,-1, 0, 0]

Now pick neuron 4 → network state = [1,-1,-1, 1, 0]

And now pick neuron 5 → network state = [1,-1,-1, 1, 1] = $E^{final} = E^1$



The Travelling Salesman Problem

Hopfield and Tank (1986) describe how a net may be used to solve the travelling salesman problem.

An n by n array of neurons can be used to code the solution to the problem, such that columns indicate the order in which that city is to be visited, and the position in the column refers to a particular city. A suitable arrangement of inhibitory connections within rows and within columns are used to ensure that one and only one neuron fires in a given row or column.

The output of a neuron reflects a current hypothesis, that a city is visited in a particular order, and the magnitude of the output represents the strength of that hypothesis. During convergence several conflicting solutions or propositions will be considered simultaneously and this is what gives the net the ability to find optimal solutions with only a few iterations. Hopfield and Tank (1986) point out that, if binary values are adopted instead of continuously variable values, the solutions found are little more than random.