Unsupervised Learning

What is Unsupervised Learning?

- Most simply, it can be thought of as learning to recognise and recall things
  - Recognition – “I’ve seen that before”
  - Recall – “I’ve seen that before and I can recall more about it from memory”.
- There is no feedback or reward like there is with reinforcement learning
- There is no given answer like there is in supervised learning

In Context

- Supervised learning:
  - These are apples, these are pears
  - What is this new thing (apple or pear?)
- Reinforcement learning:
  - Like the warmer, colder game – actions are given rewards or feedback, which is learned for future use
- Unsupervised learning:
  - Remembering the route home in day light and still being able to do it at night, or in the snow when things look different, or when parked cars have moved

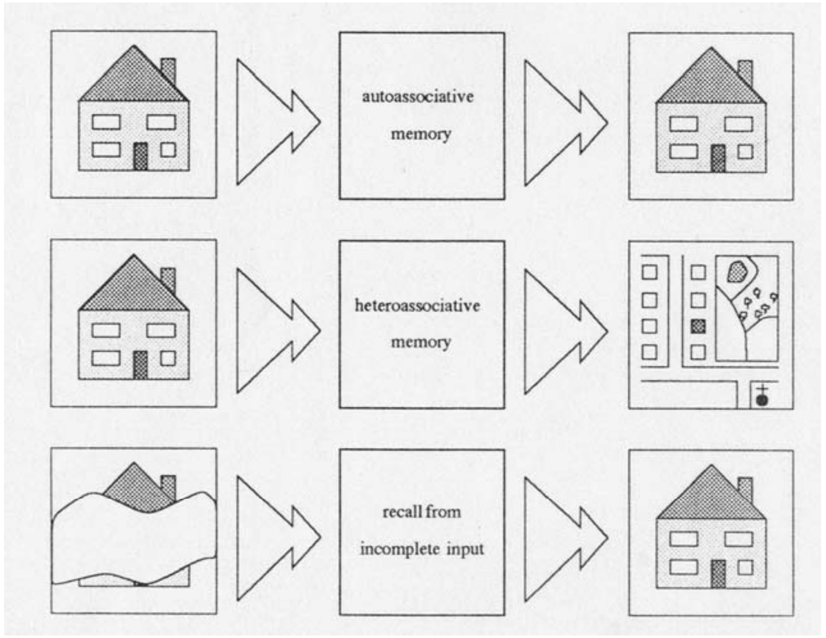
Content Addressable Memory

- One way to think about unsupervised learning is as a content addressable memory
- That is, you look things up, not by searching, but by describing some aspects of the thing you are looking for and having that trigger other things about it

Associative Patterns

- The important aspect of associative memory is that it stores patterns, or things that go together
- Just as an exotic smell might evoke memories of a holiday, associative memories work by completing partial patterns

Associative Memory



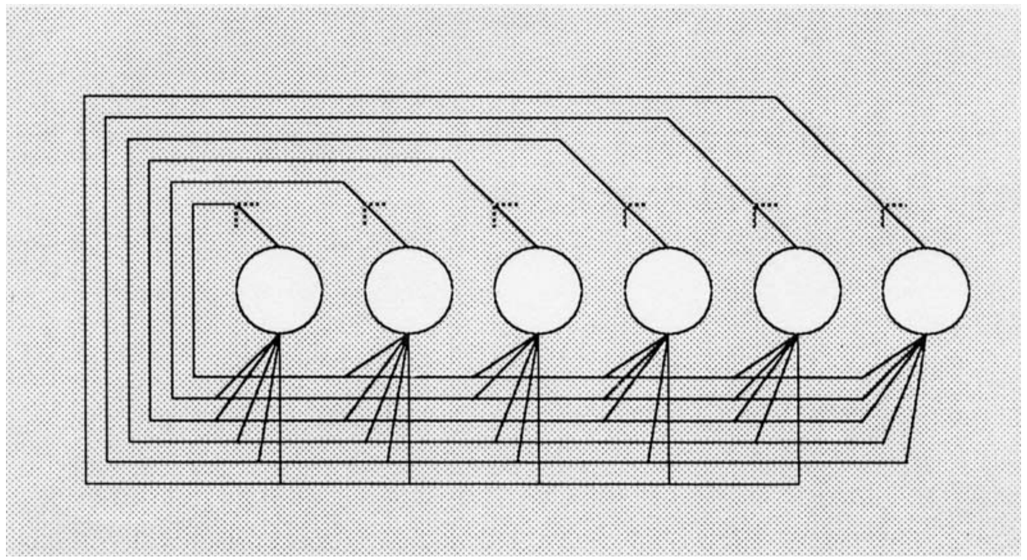
A Simple Associative Memory
The Hopfield Network

- Stores patterns in an associative memory
- Can recall a complete pattern when given only a part of that pattern as input
- Robust under noise – will recall the nearest pattern it has to the input stimulus
- Robust under damage – remove a few of the connections and it still works

Characteristics of a Hopfield Network

- A collection of nodes, which we will call neurons (though they are really just simple mathematical functions)
- Each neuron is connected to every other neuron in the network (but not itself) – we call the connections synapses
- Synapses have a weight that is either excitatory (+ve) or inhibitory (-ve)
- Weights are symmetrical: $W_{ij} = W_{ji}$
- Neurons can be either on or off – represented as an output value of +1 or -1
- The neurons are of the McCulloch and Pitts type that we have already seen.

A Hopfield Network



Recall in a Hopfield Network

- The output (+1 or -1) from a neuron is calculated based on the incoming weights and the values carried along those weights
- The output from each neuron is multiplied by the weight on each connection leaving that neuron to contribute to the input to its destination node
- So the input to a neuron is the sum of the product of the incoming weights and their values

The Value of a Hopfield Neuron

$$u_i = \sum_{i \neq j} w_{ij} v_j + I_i$$

$$v_i = \begin{cases} 1 & u_i \geq 0 \\ -1 & u_i < 0. \end{cases}$$

u is the sum of the product of the weights, *w* and the outputs from their pre-synaptic neurons, *v* plus the input to the neuron itself (*I*)

v, the value of the neuron (its output) is either +1 or -1 depending on the sign of *u* (+ve or –ve)

Convergence

- You might think that as each neuron’s output value changes, it affects the input too, and so the output from other neurons, which change all other neurons and the whole system keeps changing forever
- But it doesn’t
- Given an input pattern, a Hopfield network will always settle on a fixed state after a number of iterations
- This state is known as the *attractor* for the given input and it represents the pattern that has been recalled

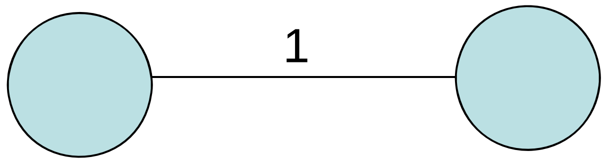
As the Network Settles

- As the network settles in its steady state, output from the neurons becomes consistent with the weights
- This consistency can be measured by multiplying the values of each pair of neurons with the weight of the synapse between them and summing the result:

$$\sum_i \sum_j w_{ij} v_i v_j$$

Simple Example

- Take a single pair of neurons:



- With a connection strength of 1:

  1 x 1 x 1 = 1

  -1 x -1 x 1 = 1

  -1 x 1 x 1 = -1

Stable States

- The sum of the product of the weights and the values gets larger as the network approaches an attractor
- In the previous example, 1,1 and -1,-1 are attractors for the given network
- Set them as inputs, and the network will not change
- Set -1, 1 or 1, -1 as inputs, and it will change to 1,1 or -1,-1

Energy Function

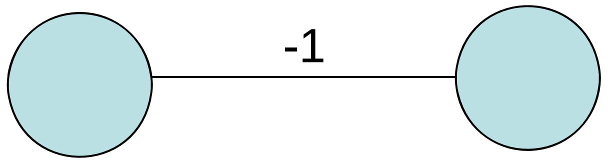
- A network in a stable state is said to have low energy
- A network in transition has higher energy
- So energy is somehow the opposite of the consistency measure we saw earlier
- In fact, if we make that measure negative so that low values correspond with stable states, we get

$$E = -\frac{1}{2}\sum_i \sum_j w_{ij} v_i v_j$$

- We halve the value because we are counting the bi-directional synapses twice
- So, recall in a Hopfield network is the same as minimising the energy function

Another Example

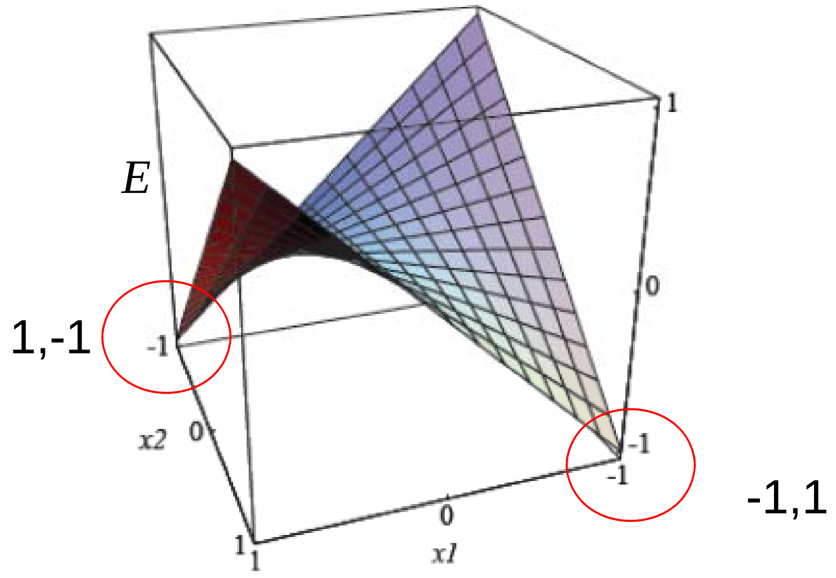
- Look at the simple network below



- Stable states are 1,-1 and -1,1
- Put it in 1,1 or -1,-1 and one neuron will flip
- This is a so-called flip-flop network

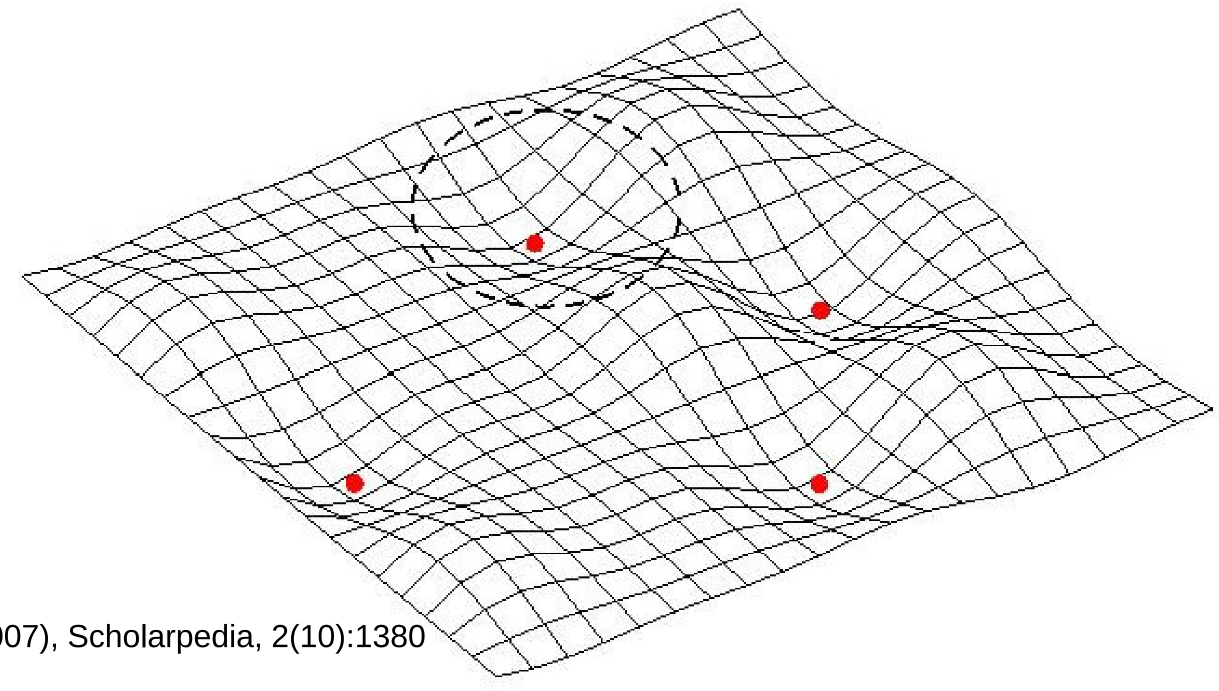
Flip Flop

- It's energy is $E = -\frac{1}{2}(-v_1 v_2) + (-v_2 v_1)$
- Which is $E = -(-v_1 v_2) + (-v_2 v_1)/2 = v_1 v_2$
- Which is plotted below:



Pattern Recall

- So a Hopfield network minimises its energy function and settles into an attractor state, which represents the full pattern that the inputs are closest to



Chris Eliasmith (2007), Scholarpedia, 2(10):1380

Learning in a Hopfield Network

- Now we turn to the question of learning in a Hopfield network – how do the weights get set to store the memories?
- The goal is to ensure that the energy of the network at each of the patterns is locally minimal (i.e. the lowest near that pattern)

Hopfield Learning

- To minimise the energy, we present a pattern p to a network and update the weights thus:

$$w_{ij} \leftarrow w_{ij} + p_i p_j$$

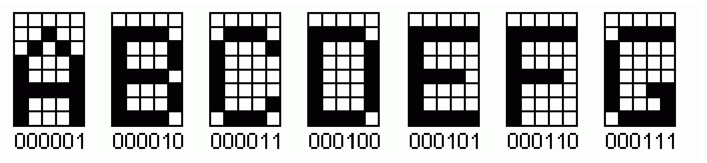
- Simply update the weight between two neurons by the product of its two neurons' values in each pattern
- So neurons that are both present in many patterns have larger weights!

Learning

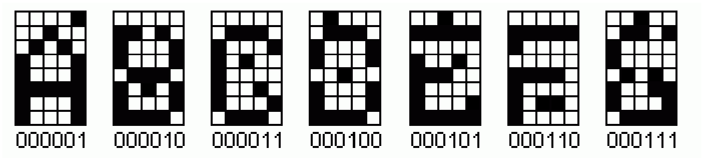
- Finally, when all the patterns have been loaded, we divide each weight by the number of patterns there are to normalise the weights
- We also make sure the leading diagonal of the weights matrix contains zeros – to ensure no neuron links to itself
- What does this remind you of?
- Hebb’s rule – when two neurons are on, strengthen the weight between them

Example – Learning Images

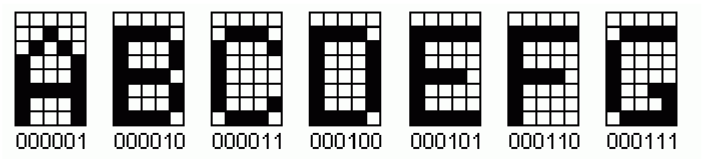
- Train a network with clean images:



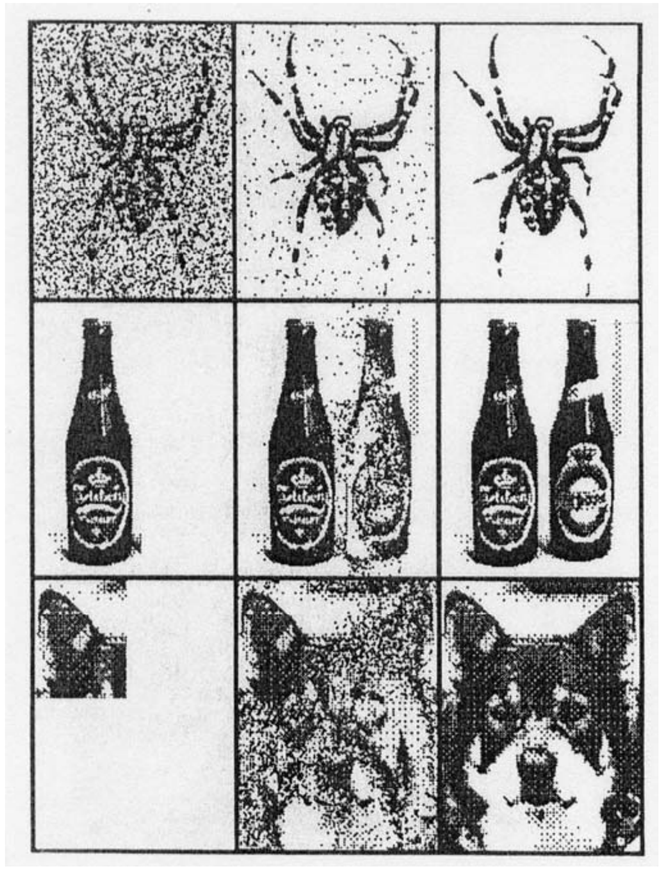
- Present corrupted text as inputs:



- Network produces clean characters as outputs:



Example – Learning Images



Hertz, Krogh & Palmer, 1991