Unsupervised Learning

- Learning without a teacher
  - No targets for the outputs
  - Networks which discover patterns, correlations, etc. in the input data
  - This is a *self organisation*
- Self organising networks
  - An important parallel with neuronal networks in the brain
  - Target outputs are often a totally meaningless concept in the brain
- An obvious question to ask is -

What meaning can be ascribed to the outputs of an unsupervised network?

"Meaning" of Outputs in Unsupervised Networks

- We need to be able to produce a semantics for the outputs
- There are a number of possibilities depending on the architecture of the network and the learning algorithm employed
- Hertz, Krogh & Palmer suggest the following -
  - Familiarity
  - Principal Component Analysis
  - Clustering
  - Prototyping
  - Encoding
  - Feature Mapping

Output Meanings (I)

1. Familiarity
   - A continuously-valued output could indicate how similar a new pattern is to a typical or an average pattern
   - 1 might mean very similar and 0 very different with a gradation in between
   - Gradually the network could learn what a typical pattern is
2. Principal Component Analysis
   - A number of familiarity measures on a number of output nodes provides a facility for measuring similarity on a number of different metrics
   - Each metric would be a principal component which the network would learn as new patterns were presented

Output Meanings (II)

3. Clustering
   - A number of binary-valued outputs could indicate which of a number of categories (one per output) a particular input pattern belonged to
   - Highly correlated patterns would be categorised together
4. Prototyping
   - The output could be a representative pattern from a particular class
   - Clustering would be performed first and then a prototype pattern produced on the output nodes
   - This would be an associative memory but with the memories being discovered rather than burnt in

Output Meanings (III)

5. Encoding
   - If there were fewer outputs than inputs then the network could perform data compression in which as much distinguishing information as possible was preserved
   - An inverse decoding network could work in tandem with an encoding network for low bandwidth comms
6. Feature Mapping
   - If the outputs represented a multi-dimensional array then the network could map input patterns onto single elements of the array in such a way that similar input patterns were mapped to "nearby" elements

Output Meanings (IV)

Here are some examples of these possible meanings from our own human experience -

- We learn at a very early age that cats are more similar to dogs than they are to bricks (1, 2, 6)
- We learn the concepts of "animal", "mammal", etc. long before we get a sound definition of them (3)
- We can also produce examples of certain types of animals without knowing their precise definitions (4)
- Massive data compression **must** occur somewhere in the brain somehow, or our memories would be very much poorer than they are (5)

Types of Unsupervised Learning

We shall look at two types of unsupervised learning -

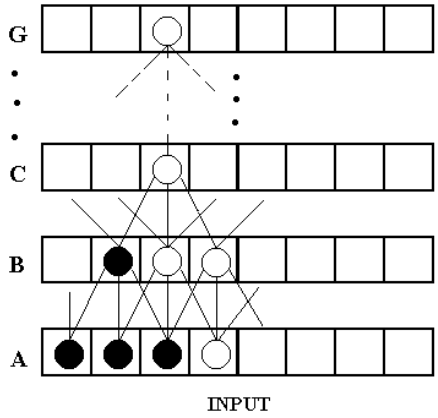
- Hebbian learning
  - The Hebb Rule is an inherently unsupervised learning rule and can readily be applied as such
  - Nodes which are active together increase their connection weight
- Competitive learning
  - In competitive learning active nodes attempt to inhibit other nodes
  - Nodes compete and successful ones prevent their peers from firing

Self Organising Feature Extraction

- An example of Hebbian learning due to Linsker (1986)
- Linsker produced a model of the visual system of the cat which was able to learn a lot a features which it is known can be identified by the visual cortex of the cat - and other mammals
- He used a 7 layer feed-forward network but with *neighbourhood* rather than total connectivity between layers
- Each layer should be regarded as a two-dimensional pixel array or visual field

Linsker's Network

- Each layer receives inputs from a particular receptive field in the previous layer



Linsker's Equations

- The output of a node in Linsker's network is given by

$$\mathbf{o_i} = \mathbf{a} + \sum_{\mathbf{j=1}}^{\mathbf{k}} \mathbf{w_{ij}} \mathbf{o_j}$$

where *a* is a threshold and the other variables take their usual meanings

- The weights are adjusted with a modified version of the Hebb Rule

$$\Delta \mathbf{w_{ij}} = \eta\left(\mathbf{o_i o_j} + \mathbf{b o_i} + \mathbf{c o_j} + \mathbf{d}\right)$$

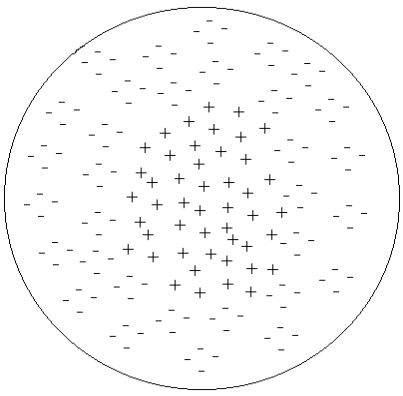
where b, c and d are parameters which permit different types of behaviour to be produced

Linsker's Simulations (I)

- Linsker used random noise as input
- Each layer had the weight update rule applied to it in turn
- The first layer of weights saturated to their maximum values (all weights were constrained to prevent them from growing inexorably)
- The nodes in Layer B simply averaged the inputs in their receptive fields (Linsker actually used a modified version of his modified Hebb Rule in practice which is why this happened)

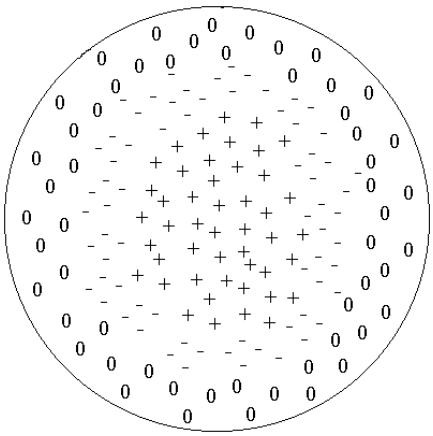
Linsker's Simulations (II)

- The nodes in Layer C developed into "centre-surround" detectors responding most to a bright spot in a dark field and vice versa



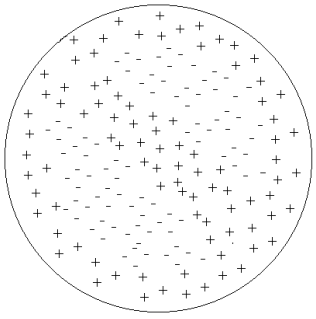
Linsker's Simulations (III)

- Some nodes in Layer C and more and more in Layers D to F became "Mexican Hat" detectors



Linsker's Simulations (IV)

- In Layer G Linsker increased the size of the receptive fields and, by varying the parameters in the weight update rule, produced detectors for alternating bands of light and dark - these were orientation-specific nodes



Linsker's Simulations (V)

- By adding connections between the nodes in Layer G he produced results very similar to the orientation columns found in the visual cortex of the cat … and monkeys … and us!
- These cortical columns contain neurons which respond to different orientations of dark bars on a light field and vice versa
- They are one of the few places in the brain where one can actually identify precisely what a neuron is representing when it fires

Self Organising Feature Mapping

- An example of competitive learning due to Kohonen (1982)
- Feature mapping is concerned with the geometric arrangement of the outputs
  - Mapping input vectors onto a line, plane, cube, hypercube of outputs
- The closer the outputs are to each other (in the Euclidean metric) the more similar are the input vectors that activate them
- Kohonen used a single layered network and a "winner takes all" learning rule based on a neighbourhood function

"Winner Takes All"

- Each node competes to respond to an input vector, *p*, say
- The node whose weight vector is closest to *p* gets the highest net input and wins the competition
  - This node outputs 1
  - All other nodes output 0
- The weights of the winning node are adjusted using the Kohonen learning rule
  - If the *i*th node wins then the elements of the *i*th row of the input matrix are adjusted
- The Kohonen rule allows the weights of a node to learn an input vector

Kohonen's Neighbourhood Function

- Instead of updating only the winning node, Kohonen had all of the nodes within a certain neighbourhood of the winning node to be updated
- The neighbourhood $N_i(d)$ contains all of the nodes that lie within a radius *d* of the winning node

$$N_i(d) = \{j, d_{ij} \leq d\}$$