Self Organizing Feature Maps

Agenda

1) Self-Organization
2) Unsupervised Learning
3) Feature Maps
4) Self Organizing Feature Maps (SOFMs)
    4.1) Network Architecture
    4.2) Network in Operation
5) Network Initialization and Training Techniques
6) Mathematical Foundations
7) Pros & Cons of SOFMs
8) Example Implementations

1) Self-Organization

system - a group of interacting parts functioning as a whole and distinguishable from its surroundings (environment) by recognizable boundaries.

system property - the resultant system no longer solely exhibits the collective properties of the parts themselves ("the whole is more than the sum of its parts")

organization - the arrangement of selected parts so as to promote a specific function

external organization - system organization imposed by external factors

self organization - evolution of a system into an organized form in the absence of external constraints.

Can things self-organize ?

Yes, any system that takes a form that is not imposed from outside (by walls, machines or forces) can be said to self-organize

e.g. crystallization, fish schooling, brain, organism structures, economies, planets, galaxies, universe

Properties:

- absence of centralized control (competition)
- multiple equillibria (possible attractors)
- global order (emergence from local interactions)
- redundancy (insensitive to damage)
- self-maintenance (repair)
- complexity (multiple parameters)
- hierarchies (multiple self-organized levels)

What is an attractor?

A preferred position for the system, such that if the system is started from another state it will evolve until it arrives at the attractor, and will then stay there in the absence of other factors.

Examples:

- point (e.g. swinging pendulum)
- path (e.g. a planetary orbit)
- series of states (e.g. the metabolism of a cell)

Studying self-organization is equivalent to investigating the attractors of the system; a complex system can have many attractors and these can alter with changes to the system interconnections;

2) Unsupervised Learning

information processing system - a system organized in such a way to have a property of processing the input from its surroundings and producing the output

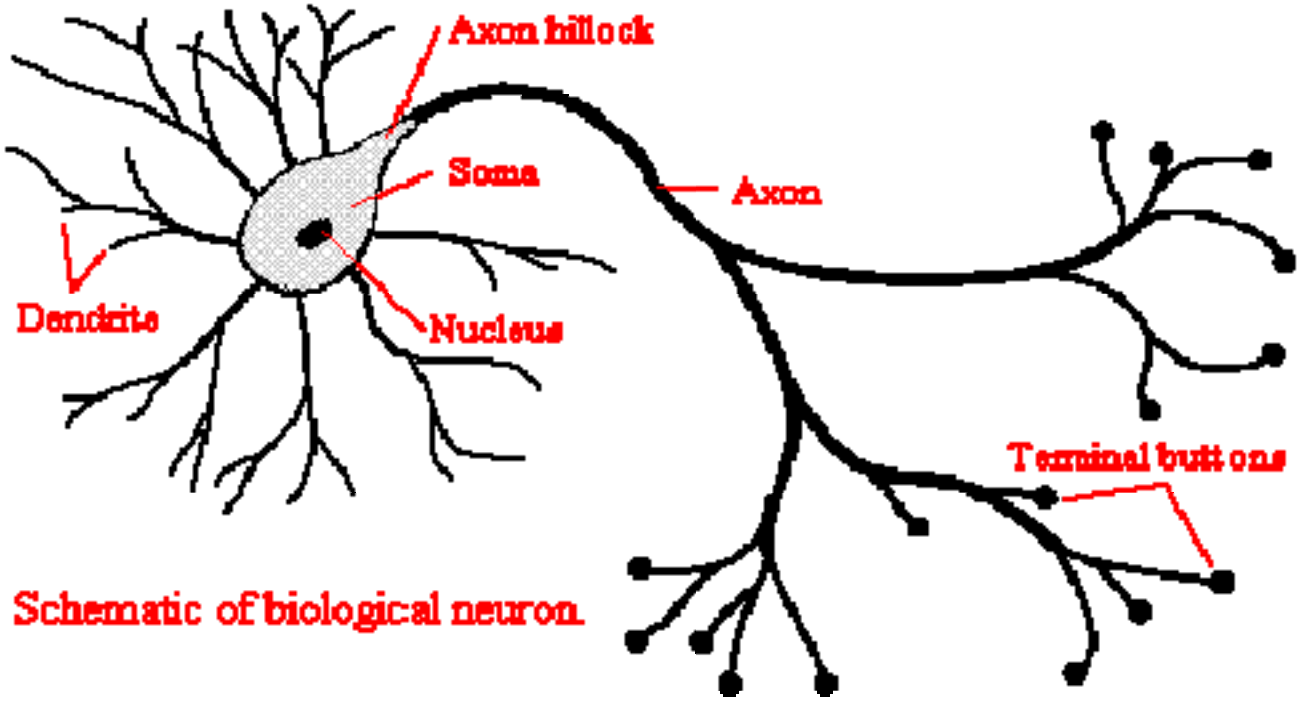
Examples: computer, neural network

computer - external constraints imposed upon its organization; lack of system properties such as intelligence, learning etc.

(unsupervised) neural network - no external constraints; mimics biological neural systems; consists of a collection of neurodes and their interconnections;

Expectations: given sufficient complexity, the same properties that occur in the brain will also occur in the network: ability to learn (possible today), self-awareness, imagination (distant future, if ever)

Neural architecture in biological systems



Schematic of biological neuron

How does brain learn spontaneously, without benefit of a tutor?

Early philosophical approach:

Postulation of homunculus - a little man living inside the brain that acted as decision maker/tutor for learning

Canadian contribution: Donald Hebb's approach:

- explicitly stated the conditions that might allow a change at the synaptic level to reflect learning and memory (1949):

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficacy, as one of the cells firing cell B is increased"

How are these changes achieved?

1) by increasing the # of transmitters released at synaptic cleft
2) by increasing the size of the synaptic cleft
3) by forming new synapses

Conclusion:

Brain is a self-organizing system that can learn by itself by changing(adding, removing, strengthening) the interconnections between neurons.

3) Feature maps

What is the result of brain's self-organization?

- formation of feature maps in the brain that have a linear or planar topology (that is, they extend in one or two dimensions)

Examples:

- tonotopic map - sound frequencies are spatially mapped into regions of the cortex in an orderly progression from low to high frequencies.
- retinotopic map - visual field is mapped in the visual cortex (occipital lobe) with higher resolution for the centre of the visual field
- somatosensory map - mapping of touch

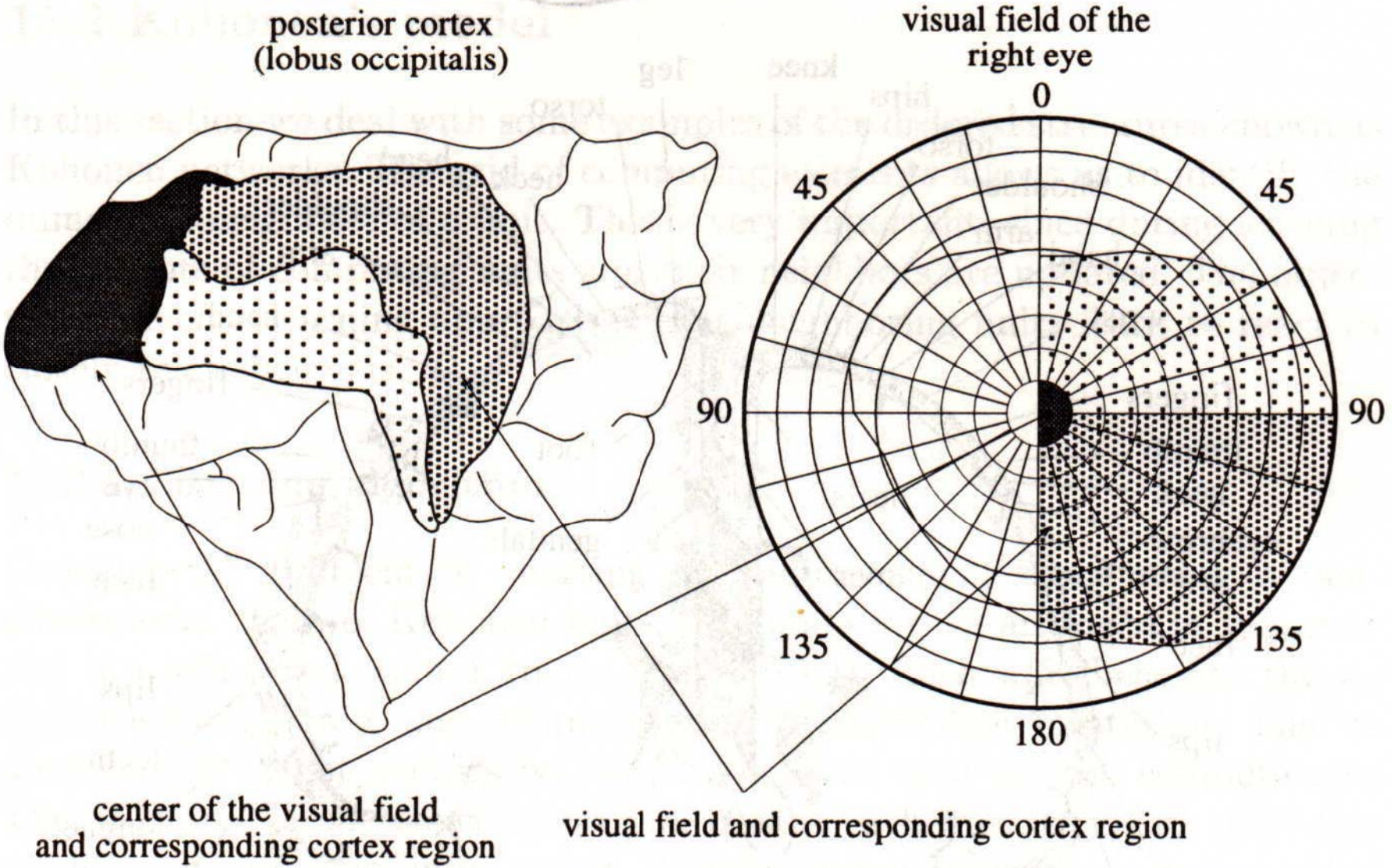


Fig. 15.2. Mapping of the visual field on the cortex

Why are feature maps important?

Sensory experience is multidimensional
E.g. sound is characterised by pitch, intensity, timbre, noise etc.,

The brain maps the external multidimensional representation of the world (including its spatial relations) into a similar 1 or 2 - dimensional internal representation.

That is, the brain processes the external signals in a topology-preserving way

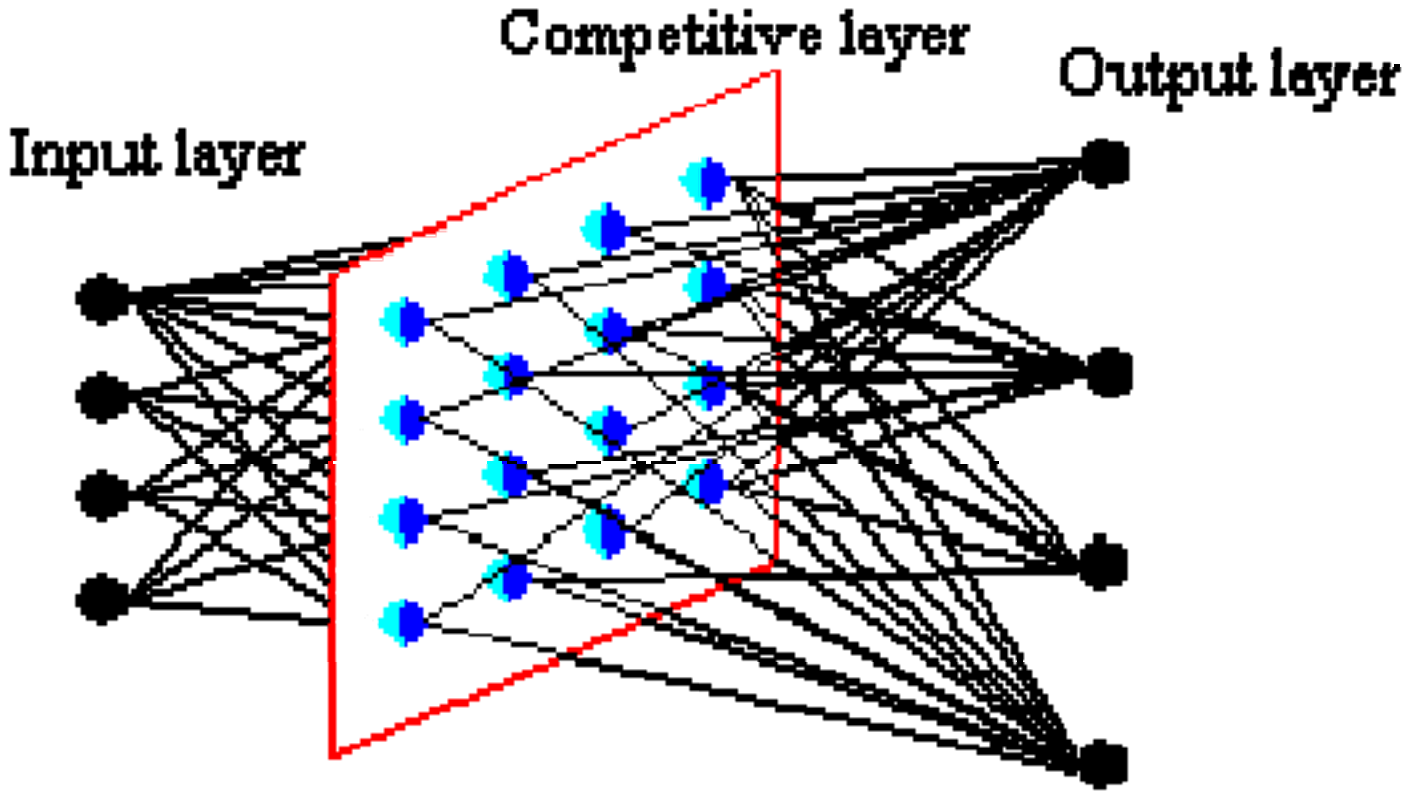
So, if we are to have a hope of mimicking the way the brain learns, our system should be able to do the same thing.

4.) Self-Organizing Feature Maps (SOFMs)

a.k.a as Kohonen networks, competitive filter associative memories

- represents the embodiment of the ideas we have discussed so far
- named after Dr. Eng. Teuvo Kohonen

4.1) Network Architecture



Input Layer

- accepts multidimensional input pattern from the environment

- an input pattern is represented by a vector.

    e.g. a sound may consist of pitch, timbre, background noise, intensity, etc.

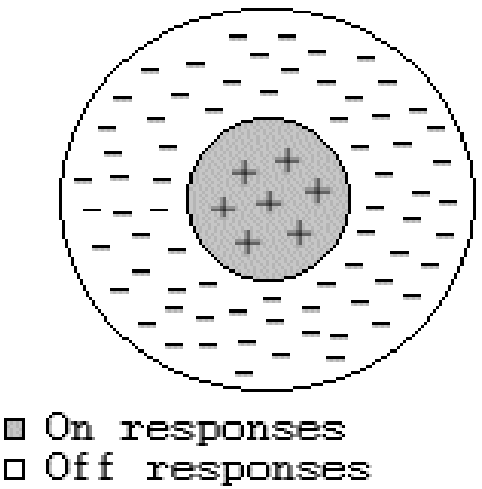
- each neurode in the input layer represents one dimension of the input pattern

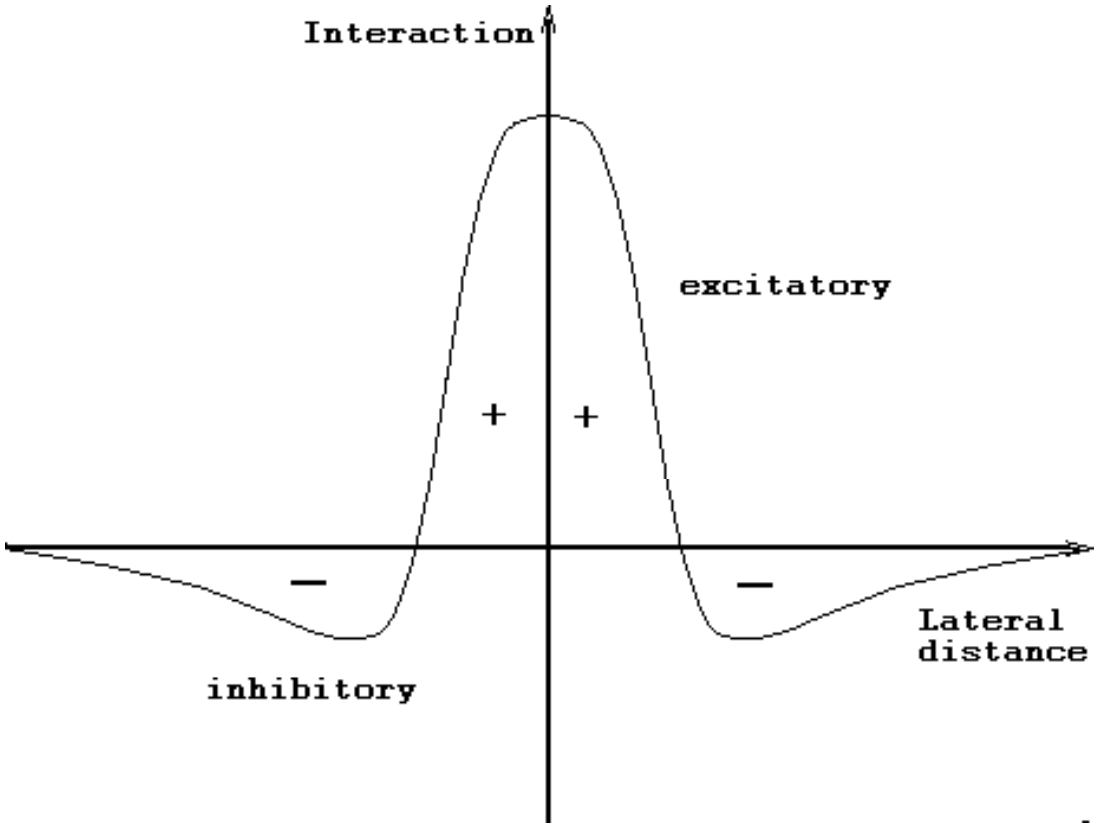
- an input neurode distributes its assigned element of the input vector to the competitive layer.

Competitive layer

- each neurode in the competitive layer receives a sum of weighted inputs from the input layer
- every neurode in the competitive layer is associated with a collection of other neurodes which make up its 'neighbourhood’
- competitive Layer can be organized in 1 dimension, 2 dimensions, or ... n dimensions
- typical implementations are 1 or 2 dimensions.
- upon receipt of a given input, some of the neurodes will be sufficiently excited to fire.
- this event can have either an inhibitory, or an excitatory effect on its neighborhood
- the model has been copied from biological systems, and is known as 'on-center, off-surround' architecture, also known as lateral feedback / inhibition.

Lateral feedback / inhibition





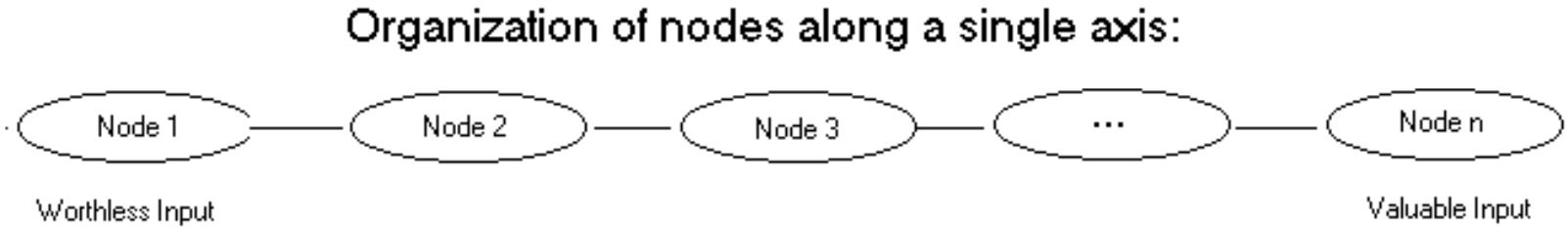
Output layer

- organization of the output layer is application-dependent

- strictly speaking, not necessary for proper functioning of a Kohonen network

- *the "output" of the network is the way we choose to view the interconnections between nodes in the competitive layer*

- if nodes are arranged along a single dimension, output can be seen as a continuum:



Example: **Organization in 2 Dimensions**

Suppose we work for SETI. We want to be able to analyze incoming radio waves and determine with some reasonable probability, whether the waves are of an intelligent and extra-terrestrial origin.

A possible input vector makeup…

- radio wave frequency, amplitude, angle of incidence with the receiver, number of repetitions in the signal encountered so far

Possible classifications of inputs…

- background noise (e.g. cosmic radiation)
- coherent, but unintelligent (e.g. pulsar)
- intelligent, terrestrial (e.g. broadcast)
- intelligent, extra-terrestrial, but man-made (e.g. artificial satellite, Voyager)
- intelligent, extra-terrestrial, unknown origin

4.2) Network in Operation

Competition in a SOFM (*The Emergence of Order from Chaos)*

- each neurode in the competitive layer receives a (complex) mixture of excitatory and inhibitory signals from the neurodes in its neighbourhood, and from the input layer.
- lateral inhibition is used to stabilize the network and prevent "meltdown" due to over - excitation in the competitive layer, or starvation due to poor selection of the threshold value.
- for a given input, the neurode which responds most strongly will be permitted to adjust the weights of the neurodes which make up its neighbourhood, including itself.
- this is a "**winner takes all**" strategy to the learning process.

- **neurodes in this layer are competing to 'learn' the input vectors.**

5) Network Initialization and Training Techniques

Initialization

- originally was thought acceptable to randomize input weights and neighbourhood associations
- this may lead to the rise of **dead vectors*:*** neurodes which will never fire…
- if we know something about the distribution of inputs, it may be useful to initialize the neurodes to mirror this distribution
- ideally, we want each neurode to be able to 'win', or be in the neighbourhood of a winning neurode, for at least one input from the training set

Selection of Network Size

Q: "How many neurodes do I need in the competitive Layer?”

A: "How much error can you live with?”

- generally speaking, more neurodes equals less error.

- higher numbers of neurodes will support finer-grained classification of the inputs.

- the size of the competitive layer is governed by the number of parameters represented in the input vector

- caveat: the more neurodes present, the longer training will take...

The Training Set

- most important element of the network planning process

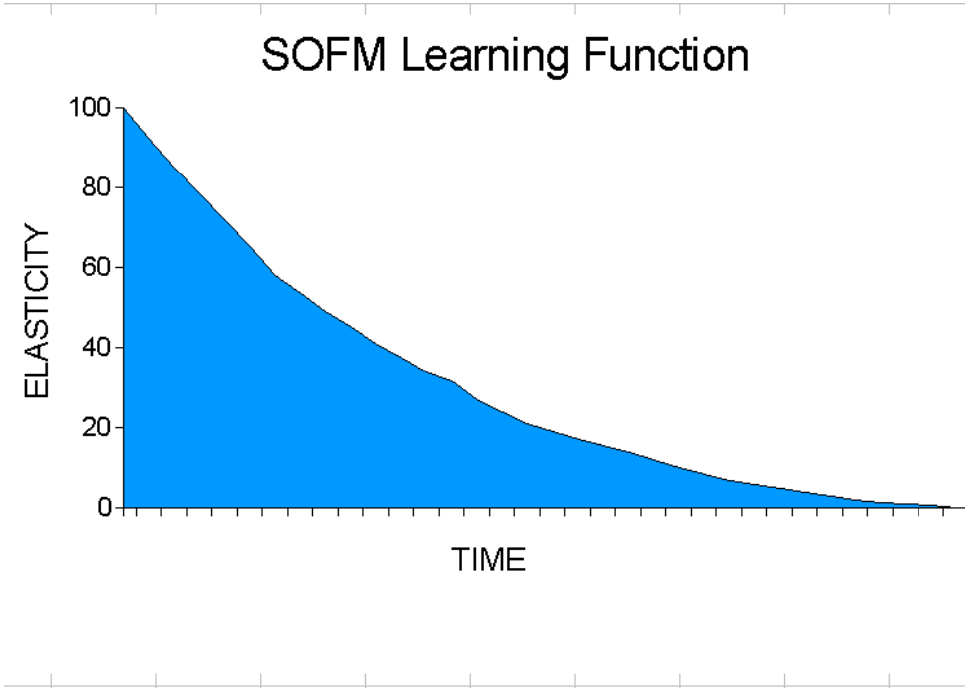
- we want the training set to mirror the probability distribution of inputs in the environment

- within the training set, we should randomize the presentation of input classes

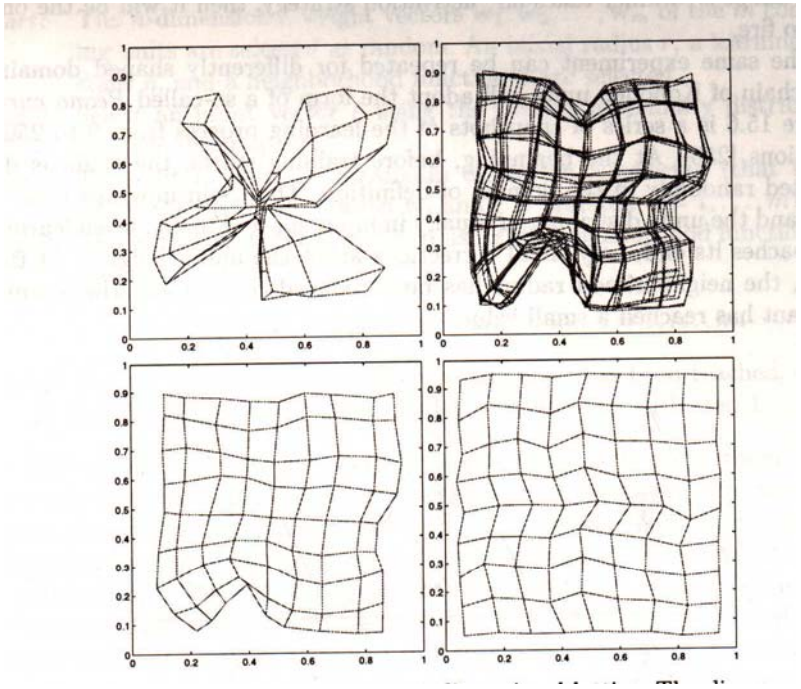
- we may need to adjust the elasticity of the learning rate to accommodate large sets

Note: No guarantee of convergence for neural networks with more than one dimension in the competitive layer

SOFM elasticity



Network convergence



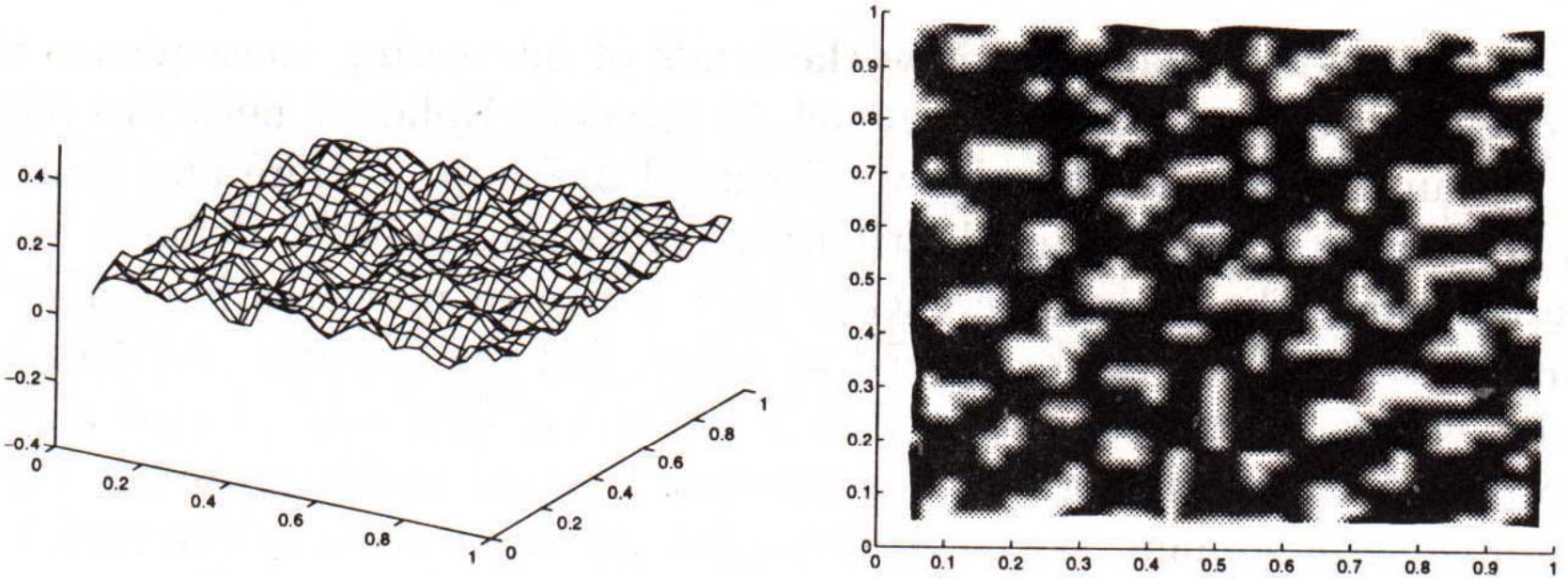
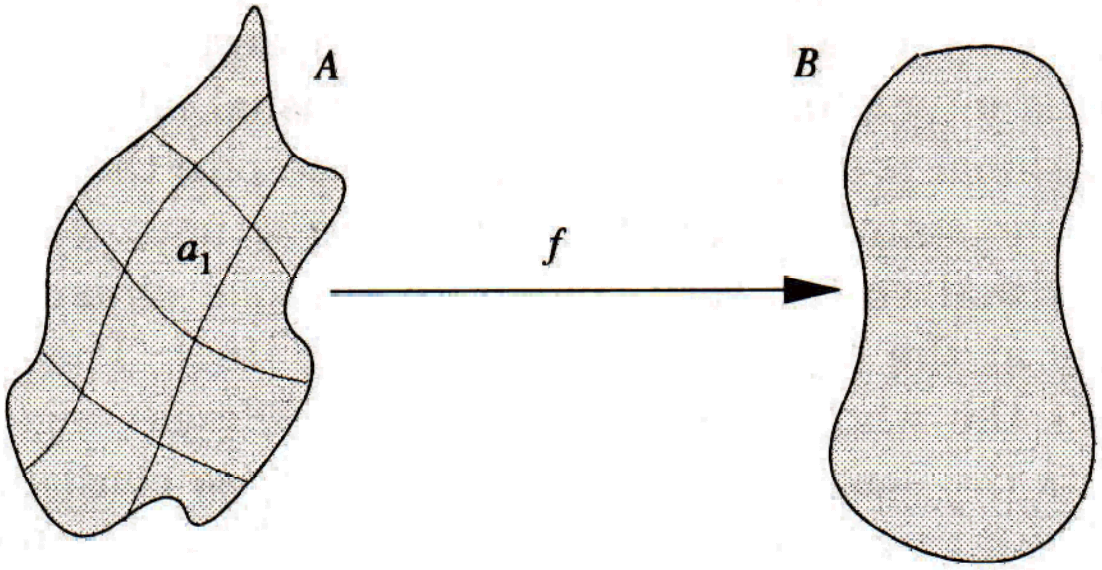


Fig. 15.9. Two-dimensional map of a three-dimensional region

6) Mathematical Model



- a neural network computes a function f from input space A to output space B
- domain of f is covered by Kohonen network in such a way when, for example, an input vector is selected from the region a1, only one unit in the network fires

- input vector is represented by scalar signals v_1 to v_n:

  $v = (v_1 \ldots v_n)$

- every unit "i" in competitive layer has a weight vector associated with it, represented by variable parameters w_{i1} to w_{in}:

  $w = (w_{i1} \ldots w_{in})$

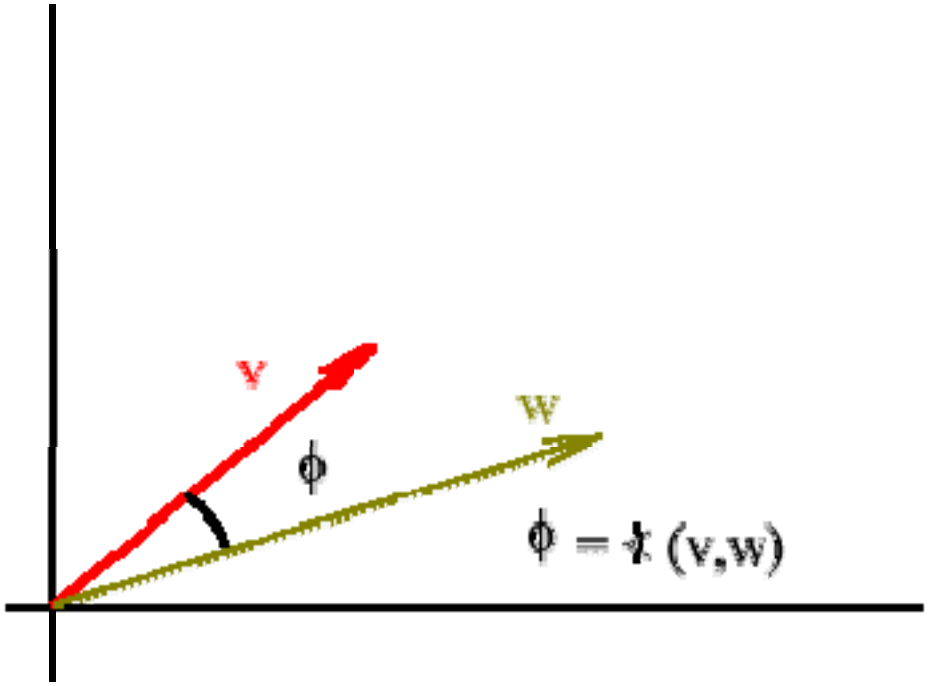
- we compute the total input to each neurode by taking the weighted sum of the input signal:

$$s_i = \sum_{j=1}^{n} w_{ij} v_j$$

- every weight vector may be regarded as a kind of image that shall be matched or compared against a corresponding input vector; our aim is to devise adaptive processes in which weight of all units converge to such values that every unit "i" becomes sensitive to a particular region of domain

- geometrically, the weighted sum is simply a dot (scalar) product of the input vector and the weight vector:

$v*w = v_1 w_1 + ... + v_n w_n$



- the dot product can also be defined as:

  (def. 1.0) $v*w = \|v\| \|w\| \cos(v,w)$

  where ||...|| is the magnitude of the vector which can be calculated using Pythagorean Theorem

- geometrically the dot product (i.e. the weighted sum) is equivalent to taking the projection of the input vector onto the weight vector and multiplying by the the length of the weight vector:

  $\cos(v,w) = (v*w) / (\|v\| \|w\|)$ (from def. 1.0)

  $\cos(v,w) = \text{proj}_w v / \|v\|$ (from def. of cosine)

  $\Rightarrow v*w = \text{proj}_w v \ \|w\|$

- we would like to correlate the dot product to the angle (v,w);

  Why? We need to a mechanism of determining the winning neurode

- in order to get the angle, we need to take into accounts the lengths of two vectors; this is bad because if we have a vector with magnitude much larger than the magnitude of the other, the dominating vector will dominate the dot product

  Why is this bad? Remember, there is no central authority (i.e. domination) in a self-organized system

- the solution is to normalize the two vectors: we collapse the two vectors into unit vectors, while preserving the angle between them:

  $v' = v / \|v\| \qquad w' = w / \|w\|$

- so now we get a nicely behaved pair of vectors and a relationship between the dot product and the angle:

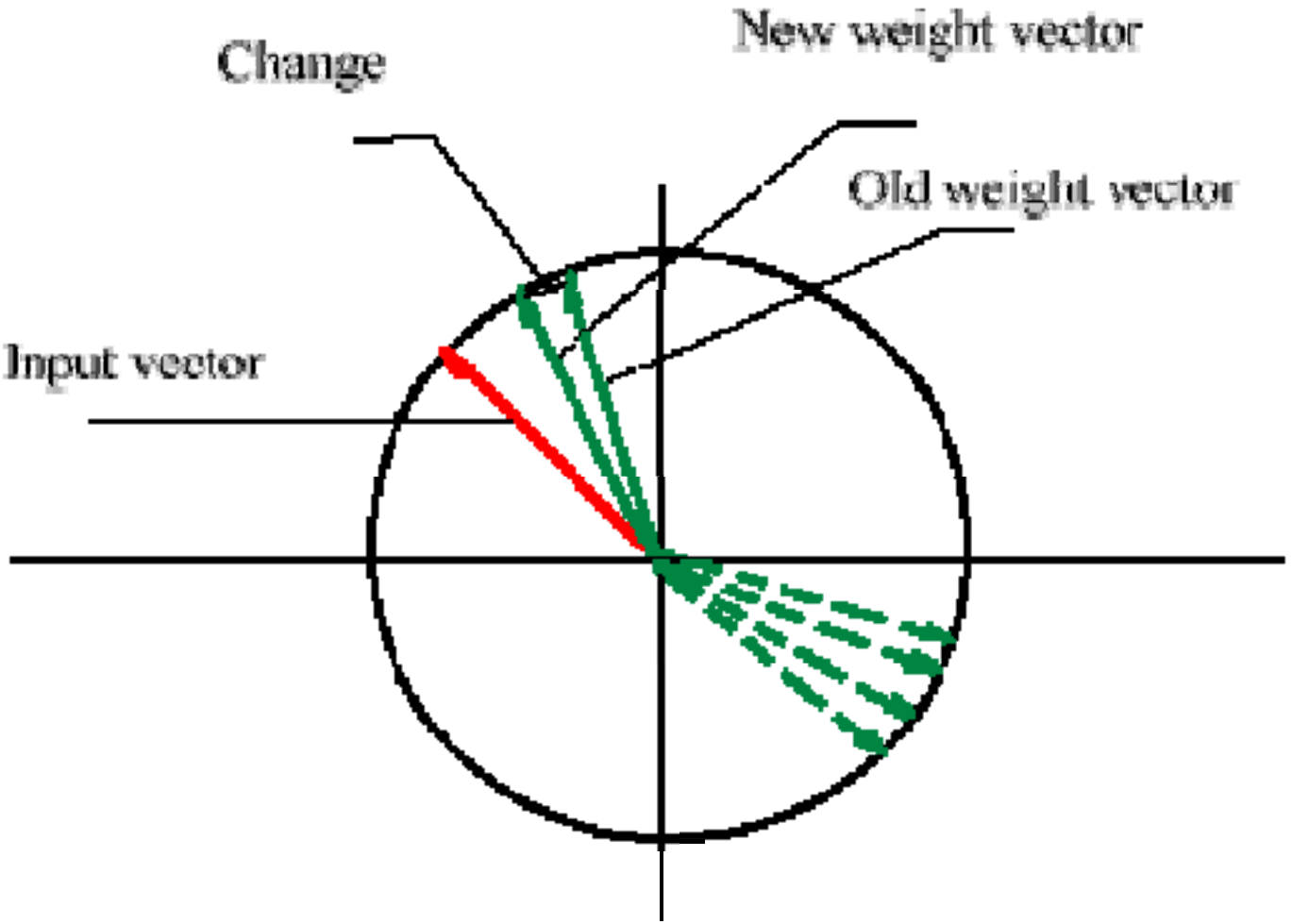
v'*w' = ||v'|| ||w'|| cos (v,w) = cos(v,w)

- the cos(v,w) gets larger as the angle gets smaller

=> the neurode with weight vector making the smallest angle with the input vector has the largest dot product

=> the weight vector pointing most nearly in the same direction as the input vector is the winner of the competition

- during training, the small weight changes in the learning cycle correspond to rotations of the weight vector and the net effect of the weight change rule is to push w closer to v:

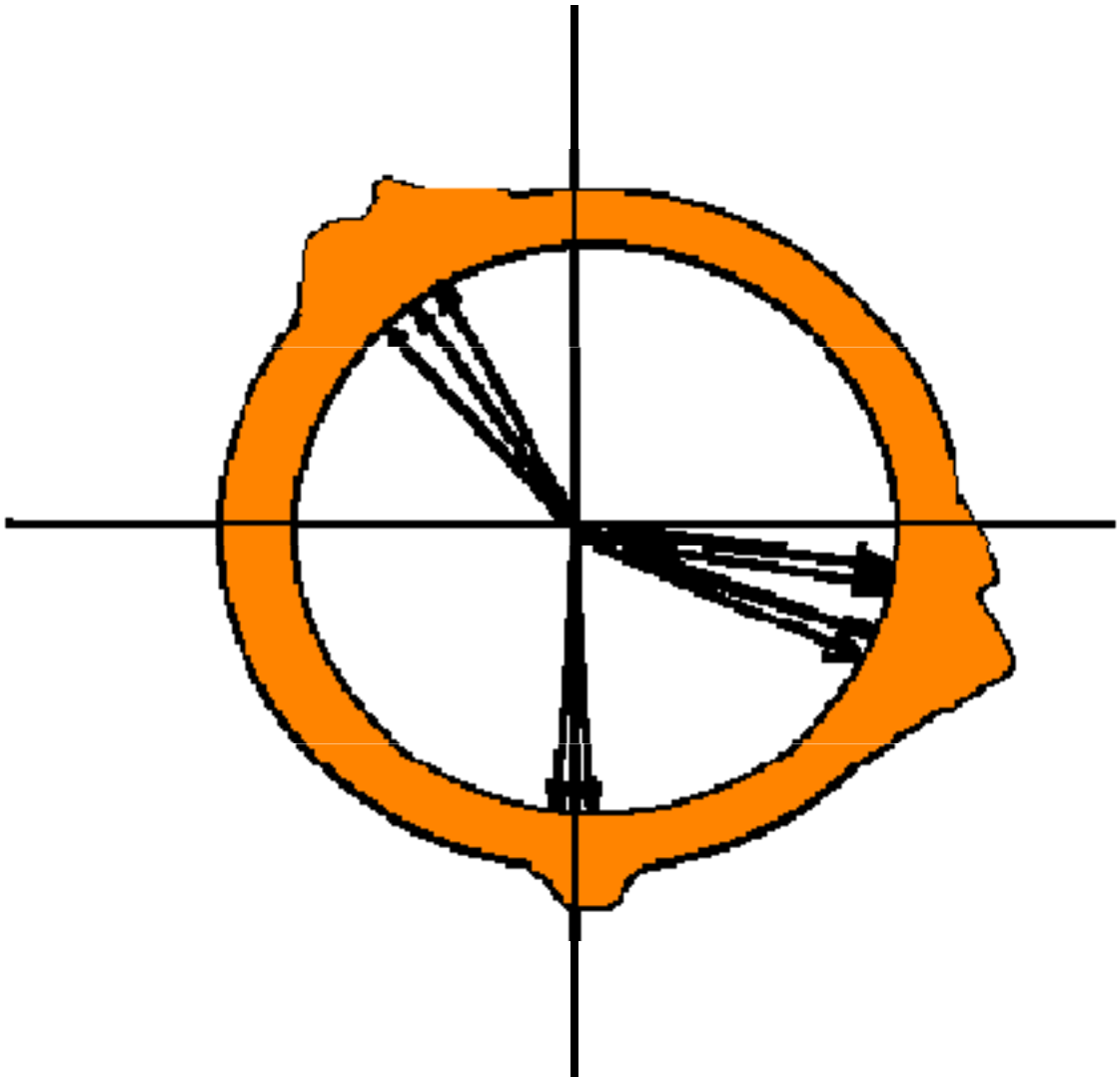


- for vectors with more than two dimensions, unit circle is replaced by unit sphere (3D) or unit hypersphere (>3D)

- after applying a number of input patterns, the network models its distribution of weight vectors around the unit circle based on the input patterns presented to it

- the weight vectors become clustered in those regions with a high probability of having an input vector, and fewer pointing to those regions having a low probability of having an input vector

Clustering of weight vectors (extrapolation of input distribution)



7) Pros & Cons of SOFMs

Pros:

- excellent for classification problems

- can greatly reduce computational complexity

- high sensitivity to frequent inputs

- new ways of associating related data

- no need of supervised learning rules

Cons:

- system is a black box

- error rate may be unacceptable

- no guarantee of network convergence for higher dimension networks
- many problems can't be effectively represented by a SOFM

- a large training set may be required

- for large classification problems, training can be lengthy

- can be difficult to come up with the input vector.

- associations developed by SOFM not always easily understood by people.