9 Unsupervised Learning (1.Competitive Learning)

9.1 Introduction to Competitive Learning

Like Hebbian learning, Competitive learning is an example of unsupervised learning. We have seen two ways in which Hebbian learning can be used.

1. The Hopfield network, in which weights trained by Hebbian learning, serves as an associative memory. In such a memory, patterns can be stored and retrieved by cueing the network with an incomplete pattern. Thus a whole neighborhood of incomplete patterns (the basin of the attractor) is mapped onto the stored pattern. In that sense a Hopfield network partitions the input space, wherein each stored patterns represents a partition.
2. When Hebbian learning was used in a feedforward network, the weight vector converged to the eigenvector corresponding to the highest eigenvalue of the autocorrelation matrix of the input data. When Sanger's or Oja's rule, extensions of the basic Hebb's rule, were used, the network was able to extract the top K eigenvectors of the autocorrelation matrix of the input data.

Thus in case 1, Hebbian learning performs a kind of clustering and in case 2, Hebbian learning extracts the principal components of the input data.

Competitive learning is a form of unsupervised learning which performs clustering over the input data. In a competitive learning network with n-output neurons, each output neuron is associated with a cluster. When a data point from a cluster is presented to the network, only the neuron corresponding to that cluster responds, while all other neurons remain silent. The single neuron that responds is often called a "winner" and therefore a competitive learning network of the kind just described is also known as a "winner-take-all" network.

It is easiest to introduce CL mechanism as a slight variation of Hebb's rule.

For a single neuron, whose input-output relationship is given as,

$$y = w.x \tag{9.1.1}$$

weight update by Hebb's rule is defined as,

$$\Delta w = \eta y x \qquad (9.1.2)$$

We have seen that as per the above rule, which simply aims to maximizing squared output, y^2, the weight vector is rotated towards the input pattern. Since the weight vector is normalized, it is only allowed to rotate. The response of the neuron, y, increases as the weight vector aligns itself more and more in the direction of the input vector.

Now consider a data set constrained such that all data exists on the unit sphere, ||x||=1. Now that lengths are constrained, angular deviation and distance are uniquely related, at least within the hemisphere that is centered around the data point, x. Response is lesser when w is farther from x, and is the highest when w = x.

It is possible to present a different neuron model, for which the response

y = f(x; w),

is the highest for w = x, and decreases with increasing distance between w and x.

A simple example of such a function is the Gaussian.

$$y = \exp(-\|x - w\|^2 / \sigma^2) \qquad (9.1.3)$$

Just as in case of Hebbian learning, weight update simply increases the squared output, y^2, in case of the above "Gaussian neuron" too, we can formulate a weight update rule that increases the neuron's response. In other words, the weight vector, w, is changed in the direction of gradient of y, w.r.t. w.

$$\Delta w = \eta \nabla_w y = 2\eta y (x - w) / \sigma^2 \qquad (9.1.4)$$

Since y is always positive, it can be considered as a part of the learning rate η. Thus, we have the following simpler learning rule:

$$\Delta w = \eta (x - w) \qquad (9.1.5)$$

We thus have a ‘Gaussian’ neuron (Fig. 9.2) whose response is given by eqn. (9.1.3). The neuron shows high response for inputs, x, close to the weight vector, w. Neurons which respond selectively to a small part of the input space are said to have ‘tuned responses.’

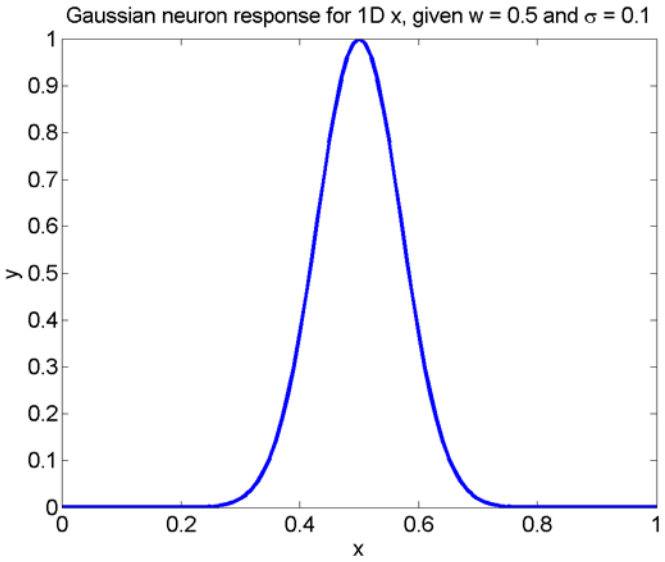

Figure 9.2: Gaussian neuron response

Such neurons are known to exist in various sensory systems:

1. Neurons in the primary visual cortex that respond to a line of a given orientation presented in their receptive fields (Hubel D.H. and Wiesel T.N. (1974) Sequence regularity and geometry of orientation columns in the monkey striate cortex. J. Comp. Neurol., 158:267-294).
2. Neurons in the auditory cortex that respond narrowly to single tones (Philips et al (1995), Factors shaping the tone level sensitivity of single neurons in posterior field of cat auditory cortex.).

Training: Single neuron case:

Let us consider what happens when we train a single Gaussian neuron using the learning rule of eqn. (9.1.5). What does w converge to?

Let x be drawn from a distribution, p(x). If we iterate eqn. (9.1.5), at convergence we have,

$$< \Delta w >= 0 = \int (x - w)\, p(x) dx \tag{9.1.6}$$

$$\int w p(x) dx = \int x p(x) dx$$

or,

$$w = \int x p(x) dx \tag{9.1.7}$$

Therefore, w is the mean of the data with probability density, p(x).

Now let us consider the result of training with multiple neurons. Let us begin with 2 neurons.

Two-neuron case:

Let us consider two neurons with response functions given as,

$$y_1 = \exp(-\|x - w_1\|^2 / \sigma^2) \tag{9.1.8}$$

$$y_2 = \exp(-\|x - w_2\|^2 / \sigma^2) \tag{9.1.9}$$

where w_1 and w_2 are the weight vectors of the two neurons respectively (Fig. 9.1.2).

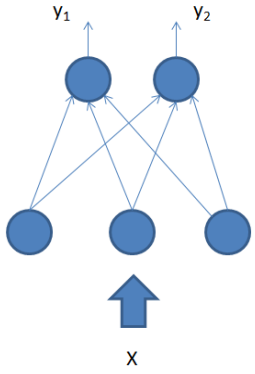

Figure 9.1.2: 2 neuron case

A straightforward extension of training to 2-neuron case would be to train both the weight vectors using the training rule of eqn. (9.1.5) as follows,

$$\Delta w_1 = \eta(x - w_1) \qquad (9.1.10)$$

$$\Delta w_2 = \eta(x - w_2) \qquad (9.\ 1.11)$$

But that gives us a trivial result since both w_1 and w_2 converge to the same point - the mean of the input data.

To obtain a more interesting outcome, let us introduce the element of *competition* into learning. Since the Gaussian neurons only respond to a specific part of the input space (defined by the centroid w and width s), let us train the neurons such they respond to non-overlapping portions of the input space. Thus either neuron "specializes" over a certain part of the input space.

This intuitive idea can be formalized in the following learning rule,

Learning rule:

For a given x,

If $y_1 > y_2$, (9. 1.12)

$$\Delta w_1 = \eta(x - w_1)$$

else

$$\Delta w_2 = \eta(x - w_2)$$

In the above training process, the two neurons are said to "compete" with each other to "win" over a certain input vector. Thus for a given input, the neuron with higher response is said to be the "winner." The above learning rule is known as competitive learning.

Let us consider the consequences of the above learning rule using a simple example.

Example:

Consider a data set, S, consisting of 4 points (A, B, C and D) located at the corners of the unit square (Fig. 9.1.3). Let the weights w_1 and w_2 be initialized to (0,0.1) and (0, -0.1) respectively.

At the end of training, neuron 1 is the winner for data from the upper (A and B) half of the unit square, while neuron 2 is the winner for the lower half. Thus the two neurons partition the input space. Two important lessons can be drawn from this exercise.

a) The border between the two partitions is the perpendicular bisector of the line joining w_1 and w_2. (Fig. 9.1.3)

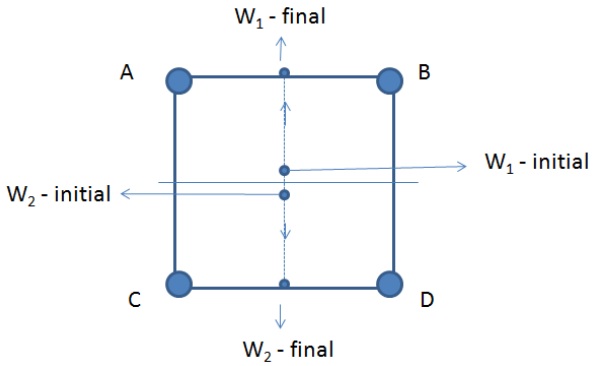

Figure 9.1.3: 2 neuron partition of the input space

The responses of the two neurons are the same on the border between the two partitions. Or,

$$\exp(-\|x-w_1\|^2/\sigma^2) = \exp(-\|x-w_2\|^2/\sigma^2) \qquad (9.\ 1.13)$$

$$\|x-w_1\|^2 = \|x-w_2\|^2$$

b) The final partitions obtained are sensitive to the initial conditions of w_1 and w_2. Consider the final weights obtained with the following initial conditions w_1 = (-0.1,0) and w_2=(0.1,0).

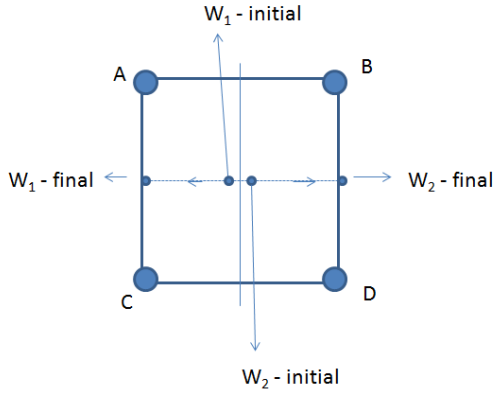

Figure 9.1.4 2 neuron partition of the input space : sensitive to the initialization

In this case, neuron 1 wins over the left half of the input space, while neuron 2 over the right half.

n-neuron case:

In a network with 'n' neurons, with responses given as,

$$y_i = \exp(-\|x - w_i\|^2 / \sigma^2) \qquad (9.\ 1.14)$$

the winner i* is the one for which

$y_{i*} > y_i \quad \forall i \neq i*.$

Only the weight of the winner is updated as usual:

$$\Delta w_{i*} = \eta(x - w_{i*}) \qquad (9.\ 1.15)$$

In this case, the network partitions the input space into n partitions, where each partition is bounded by hyperplanes. This partitioning is known as Voronoi tessellation (Fig. 9.1.5).

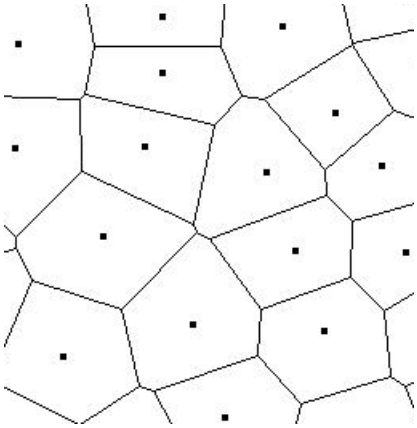

Figure 9.1.5: Voronoi tessellation

Thus competitive learning gives a way of partitioning a given data set which brings us to the problem of data clustering.

9.2 Data Clustering

The problem of data clustering consists of associating every data point, x, in a data set, S, to a cluster, c. This mapping from x to c is often done so as to optimize certain constraints.

Two commonly used criteria for clustering are:

(i) points within a cluster are close to each other.

(ii) points in different clusters are not so similar/close to each other.

Data clustering is a vast area and cannot be discussed at any length here. Our present aim is only to show that a certain kind of clustering, known as K-means clustering, is closely related to competitive learning.

In K-means clustering, the clusters are defined by a representative vector or a mean vector, w_i. A data point x_p, is associated to a mean vector, w_i, only if w_i is the nearest mean vector. We presently use Euclidean distance as the distance measure, though more sophisticated distance measures are also used.

Criterion (i) can be satisfied mathematically in this fashion.

Consider the cost function,

$$E = \sum_p \| x_p - w_{i(p)} \|^2 \qquad (9.2.1)$$

x_p – p'th data point

$w_{i(p)}$ – mean vector nearest to x_p.

The mean vectors, wi, can be updated so as to minimize E, by performing gradient descent on E, as follows:

$$\Delta w_{i(p)} = \eta(x_p - w_{i(p)}) \qquad (9.2.2)$$

The above update can also expressed as follows. If w_i is the nearest mean vector (the "winner") to x_p, update w_i as,

$$\Delta w_i = \eta(x_p - w_i) \qquad (9.2.3)$$

The update rule of Eqn. (9.2.3) is known as *K-means clustering*. Note that it is also identical to the *competitive learning rule*, which was inspired by *Hebbian learning*.

9.3 Self-organizing Map

Information is often represented spatially in the two-dimensional neuronal sheets in the brain, in both the cortex and subcortical structures. We have learnt about the somtosensory, motor and visual maps in the corresponding sensory cortices in the brain. A map, in its ordinary sense, denotes a two-dimensional representation of a real-world domain, such that nearby points in the domain are mapped onto nearby points in the map.

Due to this "adjacency-preserving" property, these maps are also called *topographic* maps. Self-organizing maps (SOM) are models of the topographic maps of the brain, first proposed by Teovo Kohonen.

The SOM model can be presented as an extension of the competitive learning model described in the previous section. It is constructed by adding a biologically-relevant feature that is not originally present in the competitive learning network.

A key property of the SOM is that nearby or similar inputs activate nearby neurons in the map. The competitive learning network does not have this property.

Consider a hypothetical competitive learning network with 3 output neurons. The input space is two-dimensional. The weight vectors w1, w2, w3 lie on a line as shown in Fig. 9.3.1, with w1 in between w2 and w3. Note that such an arrangement is possible since there is no relation between the spatial position of the weight vectors and their indices.

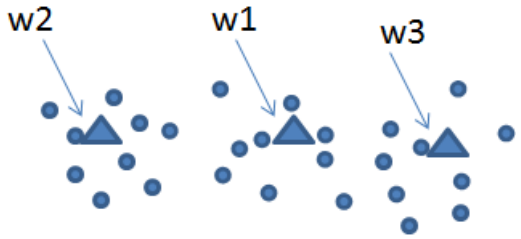

Figure 9.3.1: weight vectors and their indices when not related

Now consider an input vector, x, that is continuously displaced from w2 on the left to w3 on the right extreme. The output neurons are activated in the following order: 2, 1, 3. If we imagine that the three output neurons are on a 1-dimensional grid, and their indices represent cardinal spatial

positions on the grid, we now have a violation of the adjacency principle. Continuous changes in the input result in big jumps in the location of the neuron that is activated. Nearby inputs do not activate nearby neurons in the output layer.

In order to achieve the adjacency principle, the competitive learning must be altered. In the original learning rule of Eqns. (9.1.5, 9. 1.15), each weight vector moves independently. There is nothing that constrains them to be ordered in any fashion. The essence of the modification proposed in the SOM model, is a mechanism that ensures that the weight vectors remain spatially ordered, while they also move towards the data points that activate them maximally.

Learning rule for the Self-organizing map:

Unlike a competitive learning network, which consists of a single row of output neurons, a SOM consists of a m-dimensional grid of neurons. Usually two-dimensional SOMs are studied since SOMs were originally inspired by the two-dimensional maps in the brain. The topology of the grid is usually rectangular (Fig. 9.3.2), though sometimes hexagonal topologies (Fig. 9.3.2) are also considered.

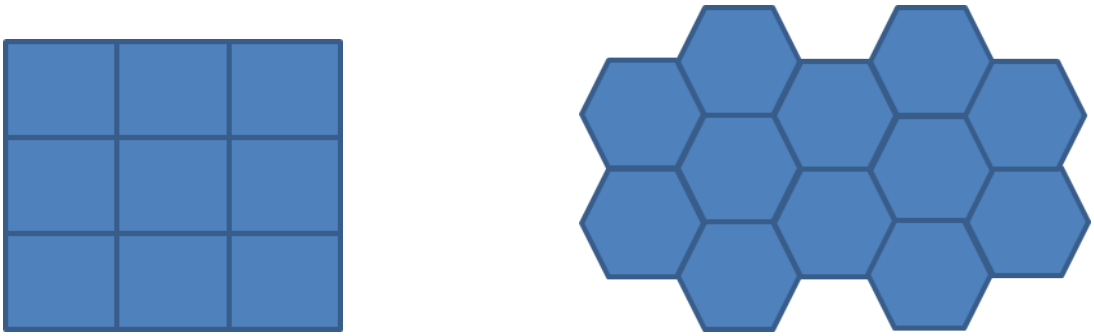

Figure 9.3.2: Rectangular and hexagonal trajectories

As in the competitive learning network, the input, x, is presented to every neuron in the SOM. In a SOM with rectangular topology, the response of a neuron at location (i,j) may be expressed as,

$$y_{ij} = \exp(-\left\|x - w_{ij}\right\|^2 / \sigma^2) \qquad (9.3.1)$$

Where w_{ij} is the weight vector that connects the input vector, x, to the neuron at (i,j)

The weights are initialized either randomly, or by drawing from the input data set itself.

Training loop proceeds as follows:

Loop over input data, x:

1. Present x to all neurons in the SOM and calculate outputs using Eqn. (9.3.1)

2. Find the winner (i*, j*) such that,

$$y_{i^* j^*} > y_{ij} \quad \forall (i, j) \neq (i^*, j^*)$$

3. As in the case of competitive learning, the weight vector of the winner is moved towards the input, x. But addition, neurons close to the winner in the SOM are also moved towards the input, x, but with a lesser learning rate. Neurons that are nearby in the SOM are defined by a neighborhood $\mathbb{N}$ (Fig. 9.3.3).

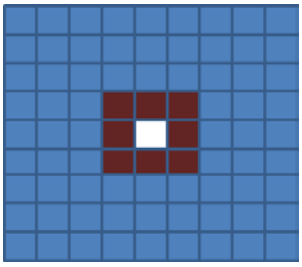

Figure 9.3.3: For the neuron in white (center) the neurons in red represent the neighborhood if we consider the neighborhood radius to be 1

Weights corresponding to all the neurons in the neighborhood, $\mathbb{N}$, of the winner, are moved towards the input as follows:

$$\Delta w_{i(p)} = \eta (x_p - w_{i(p)}) \qquad (9.3.2)$$

$$\Delta w_{ij} = \eta \Lambda (i, j; i^*, j^*)(x - w_{ij}) \qquad (9.3.3)$$

Where $(i, j) \in N$, and

$$\Lambda (i, j; i^*, j^*) = \exp(-(i - i^*)^2 - (j - j^*)^2) \qquad (9.3.4)$$

Note that $\Lambda (i, j; i^*, j^*)$ is the highest for (i,j)=(i*,j*) and decreases gradually with increasing distance between (i,j) and (i*, j*).

Neighborhood size is large in the early stages, and is decreased gradually as training progresses.

4. Loop over data until all weights converge.

End loop

Let us consider the stages in SOM training with the help of an elementary example.

Example:

Let us train a 2-dimensional SOM consisting of 15X15 neurons on 2-dimensional input data (600 patterns) uniformly distributed over the unit square bounded by (0,0) and (1,1). The weights are initialized by small random values from the interval (-0.2,0). Note that there is no ordering among the initial weights. The ordering is expected to emerge out of SOM training.

Consider the distribution of weights in various stages of training.

Evolution of the map is divided into two phases.

1) Ordering phase:

In this phase, the map orders itself in such a way that nearby neurons on the map respond to nearby points in the data set (see figs. 9.3.4 and 9.3.5).

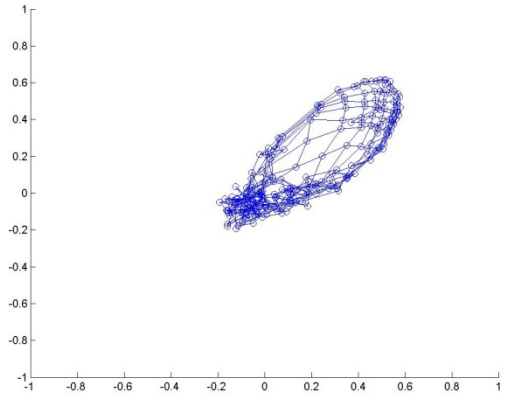

Figure 9.3.4: Ordering phase. After Epoch 1, pattern 100.

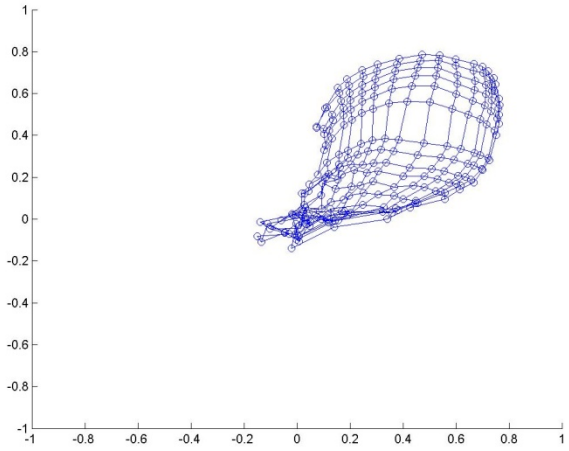

Figure 9.3.5: Ordering phase. After Epoch 1, pattern 300.

2) Settling phase:

In the settling phase, the map adjusts itself such that the density of neurons allocated to a given part of the input space reflects the local density of points in the input data set.

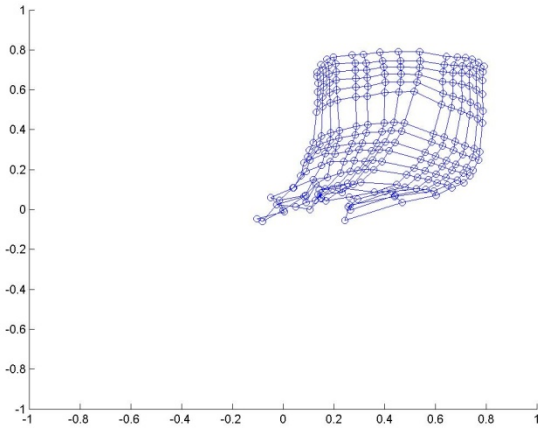

Figure 9.3.6: Settling phase. After Epoch 1, pattern 300.

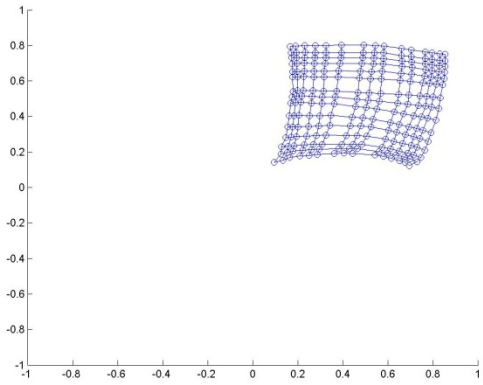

Figure 9.3.7: Settling phase. After Epoch 2, pattern 500.

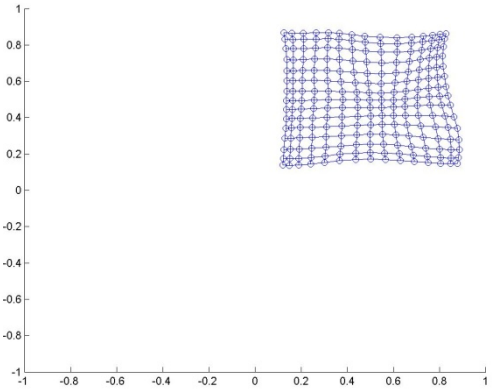

Figure 9.3.8: Settling phase. After Epoch 10, pattern 500.

Applications of SOM model in neuroscience:

Modeling orientation-sensitive maps in the visual cortex:

There are neurons in the primary visual cortex that respond specifically to oriented bars, lines or edges presented in their receptive fields. This response property suddenly emerges at the level of visual cortex and does not exist in lower stages of the visual system.

Light that enters the eye through the pupil is converted into electrical signals by an array photoreceptors called rods and cones located in the retina. These signals are then propagated through two layers of neurons – bipolar layer and ganglion cell layer – both located in the retina (Fig. 9.3.9).

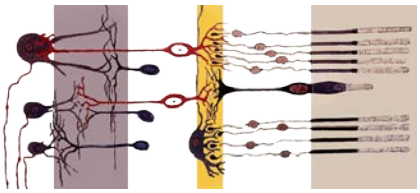

Figure 9.3.9: Retinal Bipolar and Ganglion cells

The output of the ganglion cell layer projects to a nucleus known as the Lateral Geniculate Nucleus (LGN) located in the thalamus. LGN neurons in turn project to the primary visual cortex, the first cortical stopover of visual information entering the brain.

A neuron in a given layer is connected to only a small "window" of neurons in the previous layer. This localized, or 'pyramidal' connectivity gives rise to the concept of a Receptive Field. Each neuron is able to receive information only from a small part of the external visual space. A photoreceptor receives information only a small visual angle. Same is the case with a bipolar cell since it receives inputs from a small set of photoreceptors. We have the same situation with the ganglion cell which has a RF of size 1^{o}. As you higher in the visual hierarchy, naturally, the RF size increases.

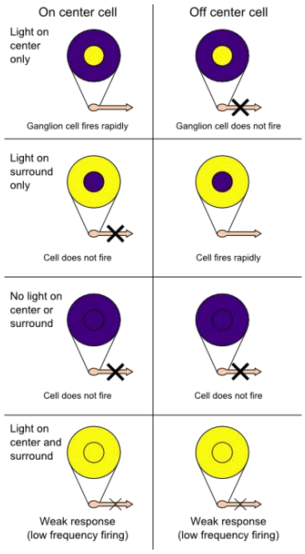

Figure 9.3.10: Receptive fields of ganglion cells

In addition to have larger RFs, neurons in higher stages of visual hierarchy also respond to more complex patterns compared to neurons in lower layers. For example, bipolar cells respond simple dot-like features – a bright dot with a dark background, or a black dot with a white background.

Neurons that respond to circularly symmetric, dot-like patterns are thought to have circularly symmetric, "center-surround" RFs (Fig. 9.3.10). Neurons that respond to a bright dot with a dark background have "ON-center, OFF-surround" RFs, while those that respond to a black dot have "OFF-center, ON-surround" RFs.

Neurons in bipolar and ganglion layers of the retina, and those of the many layers of LGN are known to have center-surround type of RFs. But in the primary visual cortex have RFs which do not have this simple circular symmetry. An important class of such cells consists of neurons that respond to oriented lines, a feature that can be easily modeled by a SOM.

Showing how orientation sensitivity arises in a network naturally by unsupervised learning is only part of the answer. The Linsker's model, which we encountered in the last chapter, achieves it. A more challenging question is: how is the orientation sensitivity spatially distributed in the cortex. If every neuron in the visual cortex is mapped onto an orientation, θ, (varying between 0 and π), what is the shape of that map?

Starting with the pioneering work of Huber and Wiesel who discovered orientation maps in the visual cortex, a lot of subsequent experimental work was directed towards study of orientation maps. A summary of some of the salient properties of orientation maps is as follows:

- The maps of orientation sensitivity are highly repetitive
- Orientation changes continuously as a function of cortical location except at isolated points.
- These isolated points where orientation changes drastically are mathematical singularities. They are also known as 'pinwheels' in the jargon of visual science.
- Orientation changes by 180 deg around the pinwheels
- Both types of pinwheels appear in equal numbers
- There exist line-like regions (fractures), across which orientation preferences change rapidly with distance.

Simple SOM-based models can be used to model all the above properties of orientation maps. A natural way to generate orientation maps with a SOM is to train it on images consisting of oriented lines.

An even simpler model of orientation map is possible, by simplifying the training data. Orientation data is parameterized by a single parameter – orientation, θ, - with periodicity of π. Instead of presenting whole images of oriented bars, we may present,

$x = [\cos(2\theta), \sin(2\theta)]$,

as input data.

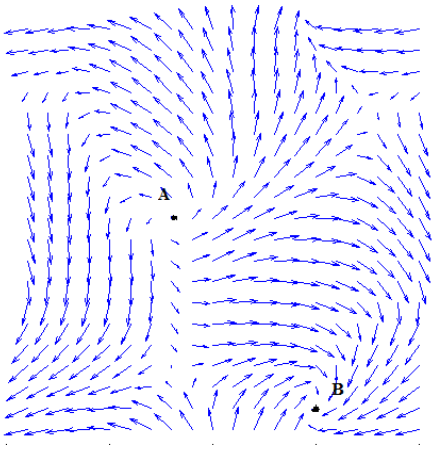

Figure 9.3.11: Orientation map constructed using a SOM trained on data given by $x = [\cos(2\theta), \sin(2\theta)]$.

Figure 9.3.11 shows an orientation map constructed using a SOM trained on data given by $x = [\cos(2\theta), \sin(2\theta)]$. Two pinwheels A (anticlockwise) and B (clockwise) can be seen in the figure. It is interesting that such a simple algorithm is capable of explaining an apparently complex neural response pattern of the primary visual cortex.

A model of somatosensory map:

Located in the post-central gyrus, the somatosensory map is a tactile map of the entire body surface (Fig. 9.3.12). Touching a point on the body surface activates a corresponding pool of neurons in the somtosensory map.

Figure 9.3.12: Pictorial Representation of somatosensory map devised by Wilder Penfield

Although the map satisfies for the most part, the adjacency principle, there are certain "fractures" in the map where the adjacency principle is violated. For example, note that the parts of the map that respond to the hand and the head are close to each other, though the hand and the head are not adjacent anatomically.

The somatosensory map is not "drawn to scale." There is a disproportionate allocation of cortical real-estate to the body surface. The map areas corresponding to the hands and mouth area are much bigger than the areas that process the abdomen. Thus more than the anatomical size of the body part that is serviced, the amount of tactile information that pours in from the said organ, seems to determine the cortical area allocated to the body part in the somatosensory map.

Kaas and colleagues studied the dynamic nature of somatotopic map in an adult ape. Using electrophysiological recordings, they mapped the five fingers of one hand of the animal. The map reveals a nearly linear ordering of the five regions corresponding to the five fingers (Fig. 9.3.13). The middle finger of the animal was then amputated and the resulting changes in the map were observed. Several weeks after the amputation, the region of the map that earlier corresponded to middle finger, started responding to the adjacent fingers – index and ring fingers. Since the middle finger was missing, the region numbered 3 in Fig. 9.3.13 had nothing to respond to. Due to dynamic remapping and rewiring, neurons of area 3 now respond to adjacent fingers. Therefore, regions 2 and 4 now expand and encroach into what was region 3 earlier.

Such dynamic reorganization of somatotopic map has been modeled by Ritter and Schulten (1986). The map model consists of a 30 X 30 array of neurons.

Inputs: Points from the image of a "hand", a two-dimensional area, parts of which designated as the five fingers (see Fig. 9.3.13).

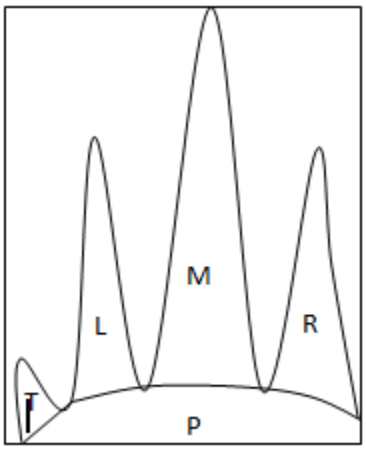

Figure 9.3.13: somatotopic map containing five fingers of an adult ape hand.

Neurons are Gaussian neurons and therefore exhibit tuned responses. But the weights are not ordered. Adjacency principle is not preserved. Therefore nearby neurons do not respond to the same finger or to nearby fingers.

Map organization at iteration 0:
Responses have random organization initially.

Map organization after 500 iterations:
A coarse map can be seen to have formed (Fig. 9.3.14). Continuous stretches of neurons in the map now seem to respond to the same finger. However, it may be noted that some neurons at this stage do not respond to any point on the hand surface (gaps in the map).
(Fig. 9.3.14).

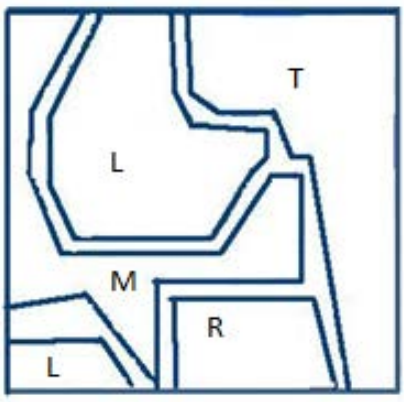

Figure 9.3.14: SOM model neurons after 500 iterations

Map organization after about 20,000 iterations:

A well-ordered map can be observed. The top right part of the map has neurons that respond to the palm. Below the palm region, there are four nearly parallel strips which correspond to the four fingers. Nearby neurons now respond to the same or adjacent finger quite consistently.

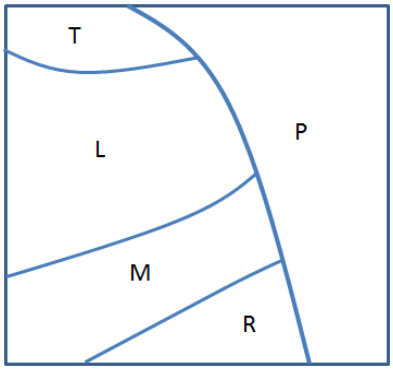

Figure 9.3.15: SOM model neurons after 20,000 iterations

Simulating amputation: inputs from the middle finger are omitted, and the map training is continued.

Map organization after another 50,000 iterations of training: the regions corresponding to the left and right fingers may be seen to encroach into what was earlier the "middle finger region."

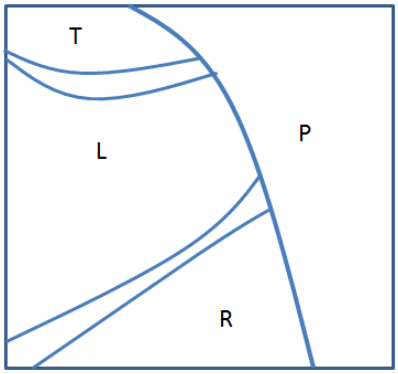

Figure 9.3.16: SOM model neurons after 50,000 iterations

Kaas JH, Nelson RJ, Sur M, Lin CS, Merzenich MM, *(1979) Multiple representations of the body within the primary somatosensory cortex of primates. Science 204:521–523*

Nelson RJ, Sur M, Felleman DJ, Kaas JH, *(1980) Representations of the body surface in postcentral parietal cortex of Macaca fascicularis. J Comp Neurol 192:611–643.*

Pons TP, Garraghty PE, Cusick CG, Kaas JH, *(1985) The somatotopic organization of area 2 in macaque monkeys. J Comp Neurol 241:445–466.*

Pons TP, Wall JT, Garraghty PE, Cusick CG, Kaas JH, *(1987) Consistent features of the representation of the hand in area 3b of macaque monkeys. Somatosens Res 4:309–331*

Ritter, H., Martinetz, T., and Schulten, K. (1990). *Neuronale Netze*. Addison-Wesley, Bonn, Germany.

Ritter, H. J. (1993). Parametrized self-organizing maps. In Gielen, S. and Kappen, B., (Eds.), *Proceedings of the International Conference on Artificial Neural Networks*, pages 568-575. Springer, Berlin.

Ritter, H. J., Martinetz, T. M., and Schulten, K. J. (1989). Topology-conserving maps for learning visuo-motor-coordination. *Neural Networks*, **2**, 159-168.

Ritter, H. J. and Schulten, K. J. (1986). Topology conserving mappings for learning motor tasks. In Denker, J. S., (Ed.), *Neural Networks for Computing*, volume 151, pages 376-380, Snowbird, UT. AIP Conference Proceedings.