Self Organizing Maps: Fundamentals

Neural Computation : Lecture 16

1. What is a Self Organizing Map?
2. Topographic Maps
3. Setting up a Self Organizing Map
4. Kohonen Networks
5. Components of Self Organization
6. Overview of the SOM Algorithm

What is a Self Organizing Map?

So far we have looked at networks with ***supervised training*** techniques, in which there is a target output for each input pattern, and the networks learn to produce the required outputs.

We now turn to ***unsupervised training***, in which the networks learn to form their own classifications of the training data without external help. This involves assuming that class membership is broadly defined by the input patterns sharing ***common features***, and that the networks will be able to identify those features across the range of input patterns.

One particularly interesting class of unsupervised system is based on ***competitive learning***, in which the output neurons compete amongst themselves to be activated, with the result that only one is activated at any one time. This activated neuron is called a ***winner-takes-all neuron*** or simply the ***winning neuron***. Such competition can be induced/implemented by having ***lateral inhibition connections*** (negative feedback paths) between the neurons. The result is that the neurons are forced to organise themselves. For obvious reasons, such a network is called a ***Self Organizing Map (SOM)***.

Topographic Maps

Neurobiological studies indicate that different sensory inputs (motor, visual, auditory, etc.) are mapped onto corresponding areas of the cerebral cortex in an ***orderly fashion***.

This form of map, known as a ***topographic map***, has two important properties:

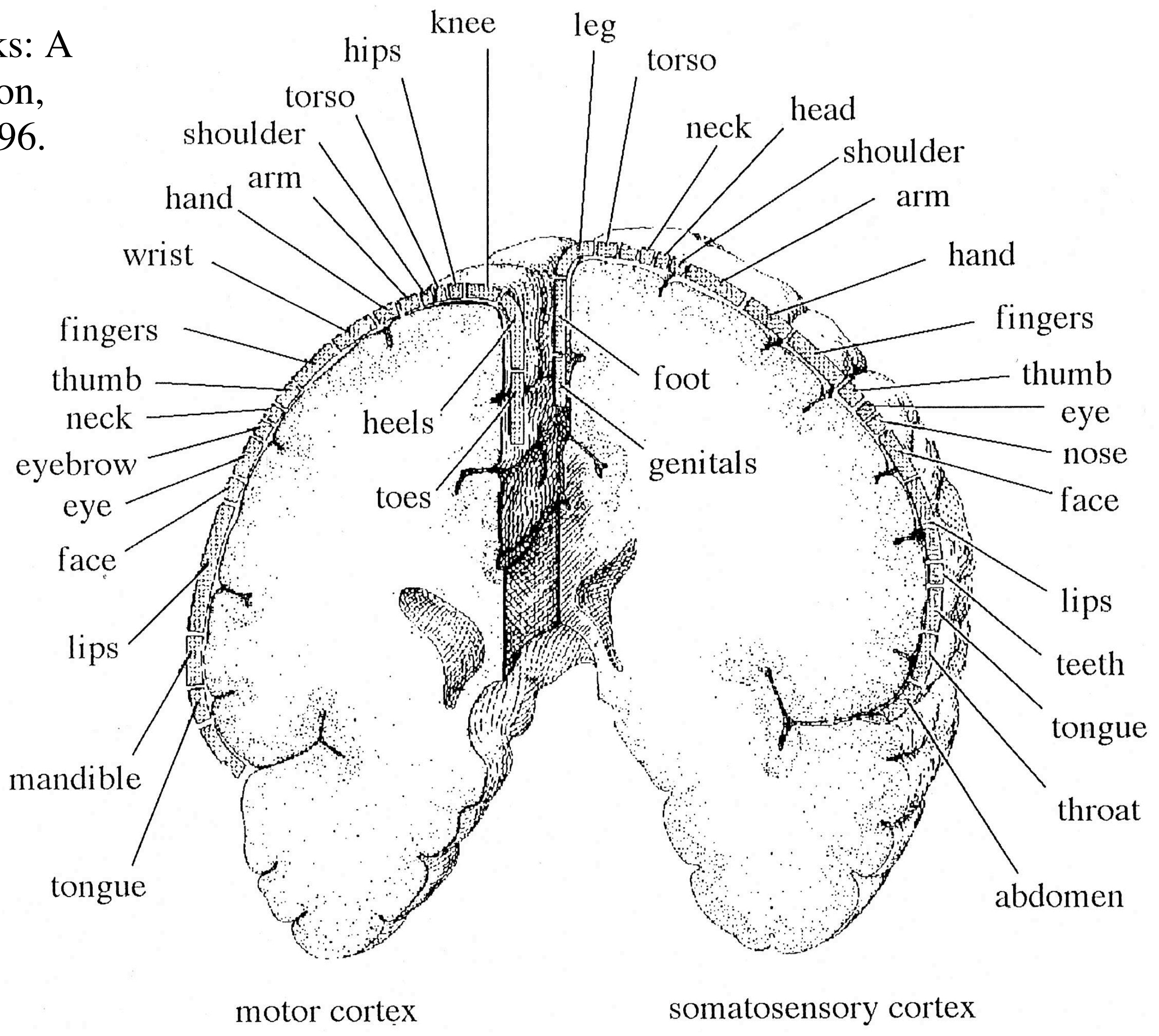
1. At each stage of representation, or processing, each piece of incoming information is kept in its proper context/neighbourhood.
2. Neurons dealing with closely related pieces of information are kept close together so that they can interact via short synaptic connections.

Our interest is in building artificial topographic maps that learn through self-organization in a neurobiologically inspired manner.

The aim is to follow the ***principle of topographic map formation***: "The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature drawn from the input space".

The Somatosensory and Motor Cortex

From: Neural Networks: A Systematic Introduction, R. Rojas, Springer, 1996.



Setting up a Self Organizing Map

The principal goal of a SOM is to ***transform*** an incoming signal pattern of arbitrary dimension into a one or two dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion.

It is natural therefore to set up the SOM by placing neurons at the nodes of a one or two dimensional lattice. Higher dimensional maps are also possible, but not so common.

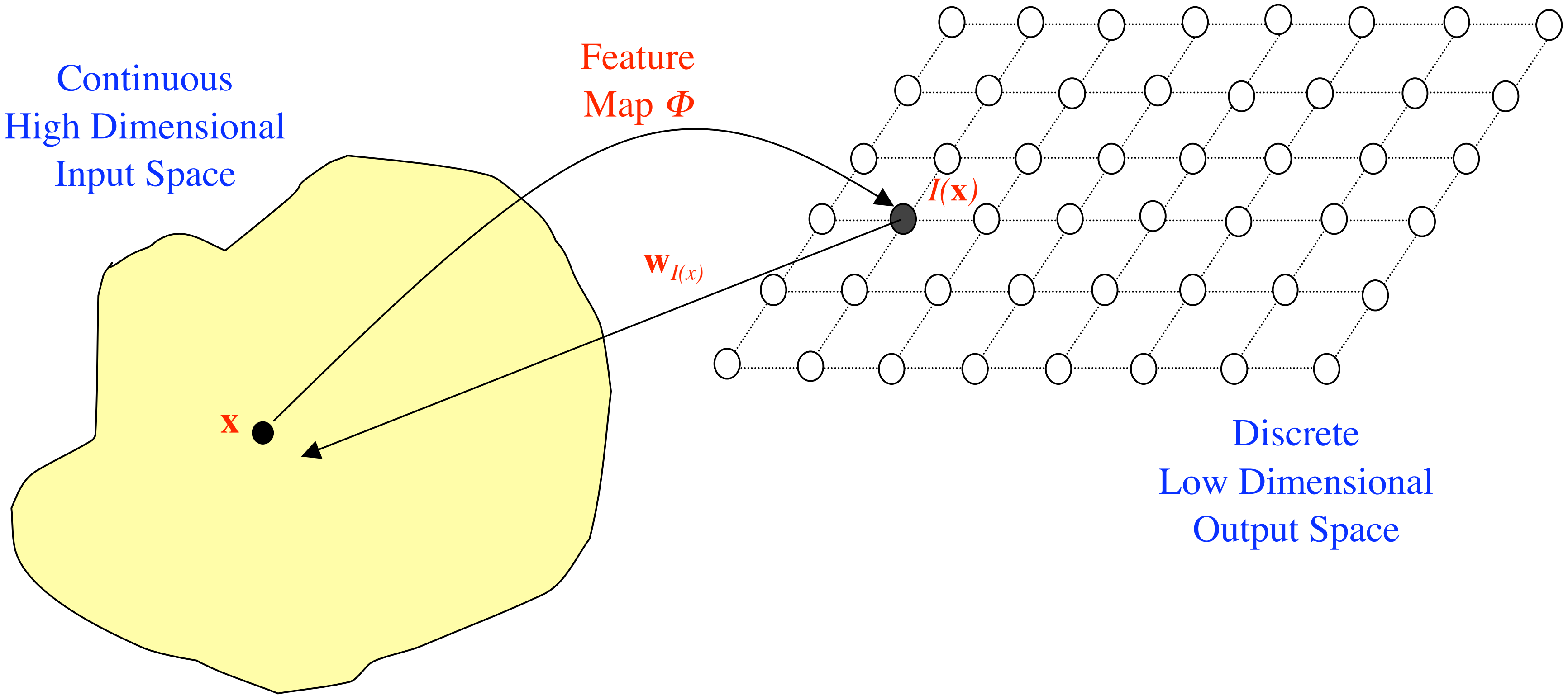
The neurons become ***selectively tuned*** to various input patterns (stimuli) or classes of input patterns during the course of the competitive learning.

The locations of the neurons so tuned (i.e., the winning neurons) become ordered and a meaningful ***coordinate system*** for the ***input features*** is created on the lattice. The SOM thus forms the required topographic map of the input patterns.

This can be viewed as a non-linear generalization of principal component analysis (PCA).

Organization of the Mapping

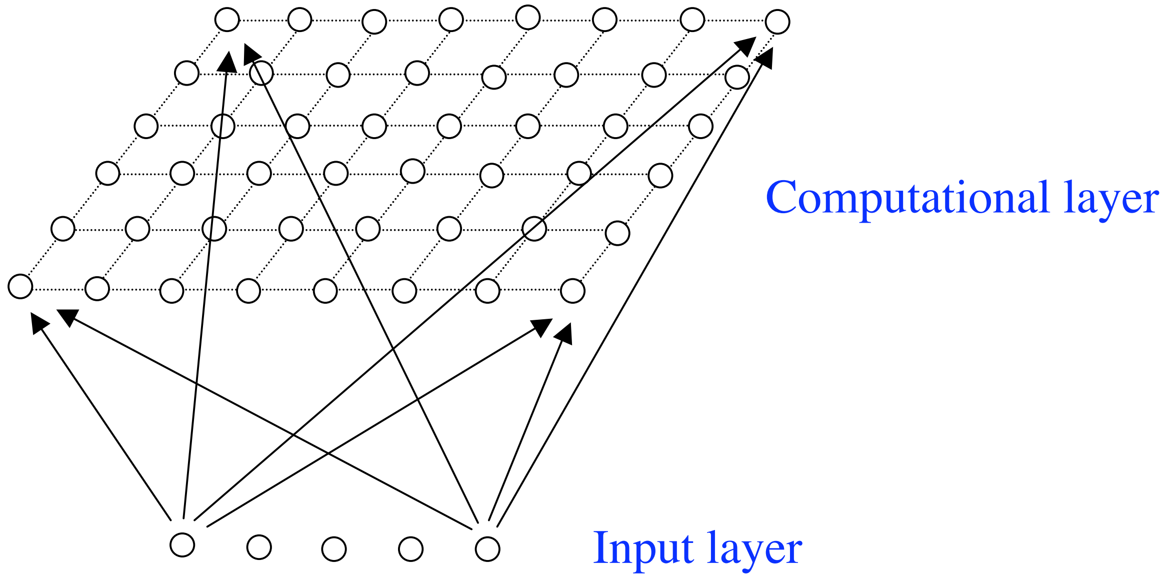
Each point **x** in the input space maps to a corresponding point *I*(**x**) in the output space:



Each point *I* in the output space maps to a corresponding point **w**(*I*) in the input space.

Kohonen Networks

We shall concentrate on the particular kind of SOM known as a ***Kohonen Network***. This SOM has a feed-forward structure with a single computational layer arranged in rows and columns. Each neuron is fully connected to all the source nodes in the input layer:



Clearly, a one dimensional map will operate in the same way but just have a single row (or a single column) in the computational layer.

Components of Self Organization

The self-organization process involves four major components:

Initialization: All the connection weights are initialized with small random values.

Competition: For each input pattern, the neurons compute their respective values of a *discriminant function* which provides the basis for competition. The particular neuron with the smallest value of the discriminant function is declared the winner.

Cooperation: The winning neuron determines the spatial location of a topological neighbourhood of excited neurons, thereby providing the basis for cooperation among neighbouring neurons.

Adaptation: The excited neurons decrease their individual values of the discriminant function in relation to the input pattern through suitable adjustment of the associated connection weights, such that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced.

The Competitive Process

If the input space is D dimensional (i.e. there are D input units), we can write the input patterns as $\mathbf{x} = \{x_i : i = 1, \ldots, D\}$ and the connection weights between the input units i and the neurons j in the computation layer can be written $\mathbf{w}_j = \{w_{ji} : j = 1, \ldots, N;\ i = 1, \ldots, D\}$ where N is the total number of neurons.

It then makes sense to define the ***discriminant function*** to be the squared Euclidean distance between the input vector $\mathbf{x}$ and the weight vector $\mathbf{w}_j$ for each neuron j

$$d_j(\mathbf{x}) = \sum_{i=1}^{D} (x_i - w_{ji})^2$$

In other words, the neuron whose weight vector comes closest to the input vector (i.e., is most similar to it) is declared the winner.

In this way, the continuous input space can be mapped to the discrete output space of neurons by a simple process of competition between the neurons.

The Cooperative Process

Neurobiological studies have found that there is ***lateral interaction*** within a set of excited neurons. When one neuron fires, its closest neighbours tend to get excited more than those further away. There is a ***topological neighbourhood*** that decays with distance.

It is natural to define a similar topological neighbourhood for the neurons in our SOM. If S_{ij} is the lateral distance between neurons i and j on the grid of output neurons, we take

$$T_{j,I(\mathbf{x})} = \exp(-S^2_{j,I(\mathbf{x})} / 2\sigma^2)$$

as the topological neighbourhood, where $I(\mathbf{x})$ is the index of the winning neuron. This has several important properties: it is maximal at the winning neuron, it is symmetrical about that neuron, it decreases monotonically to zero as the distance goes to infinity, and it is translation invariant (i.e. independent of the location of the winning neuron).

The self-organization process will work best if the size σ of the neighbourhood decreases with time. A popular time dependence is an exponential decay: $\sigma(t) = \sigma_0 \exp(-t / \tau_\sigma)$.

The Adaptive Process

Clearly the SOM must involve some kind of adaptive, or learning, process by which the outputs become self-organised and the ***feature map*** between inputs and outputs is formed.

The point of the topographic neighbourhood is that not only the winning neuron gets its weights updated, but its neighbours will have their weights updated as well, although by not as much as the winner itself. In practice, the appropriate weight update equation is

$$\Delta w_{ji} = \eta(t) \,.\, T_{j,I(\mathbf{x})}(t) \,.\, (x_i - w_{ji})$$

in which we have a time (epoch) t dependent learning rate $\eta(t) = \eta_0 \exp(-t / \tau_\eta)$, and the updates are applied for all the training patterns $\mathbf{x}$ over many epochs.

The effect of each learning weight update is to move the weight vectors $\mathbf{w}_i$ of the winning neuron and its neighbours towards the input vector $\mathbf{x}$. Repeated presentations of the training data thus leads to topological ordering.

Ordering and Convergence

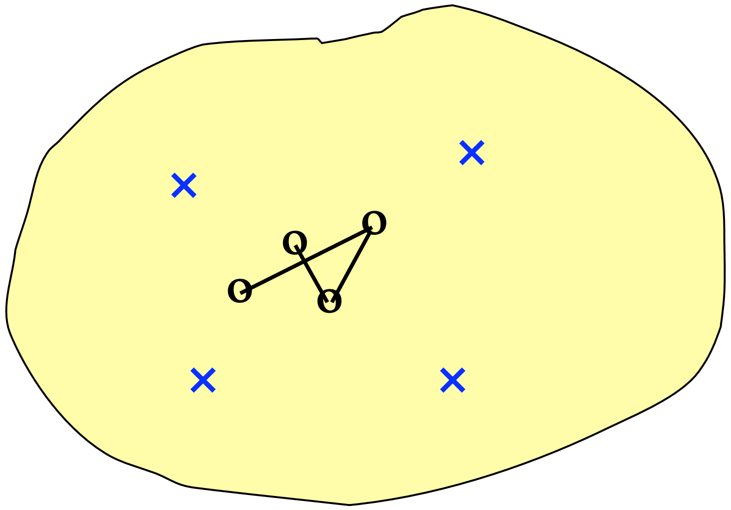
Provided the parameters (σ_0, τ_σ, η_0, τ_η) are selected properly, one can start from an initial state of complete disorder, and the SOM algorithm will gradually lead to an organized representation of activation patterns drawn from the input space. However, it is possible to end up in a ***metastable state*** in which the feature map has a topological defect.

There are two identifiable phases of this adaptive process:

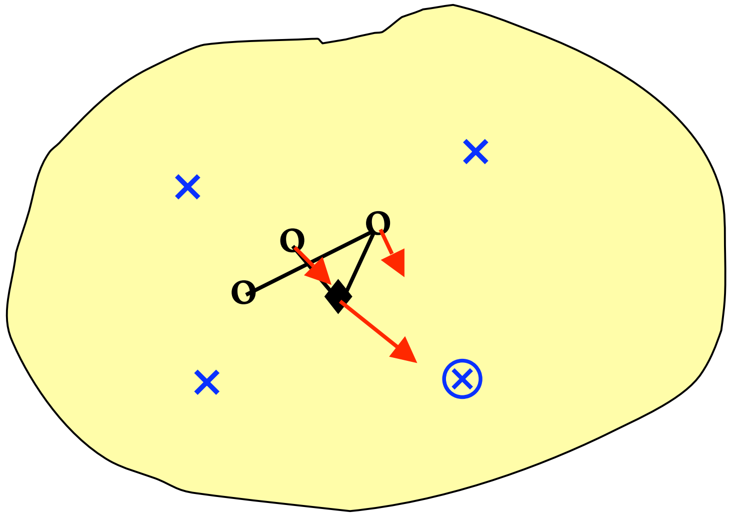
1. **Ordering or self-organizing phase** – during which the topological ordering of the weight vectors takes place. Typically this will take as many as 1000 iterations of the SOM algorithm, and careful consideration needs to be given to the choice of neighbourhood and learning rate parameters.

2. **Convergence phase** – during which the feature map is fine tuned and comes to provide an accurate statistical quantification of the input space. Typically the number of iterations in this phase will be at least 500 times the number of neurons in the network, and again the parameters must be chosen carefully.

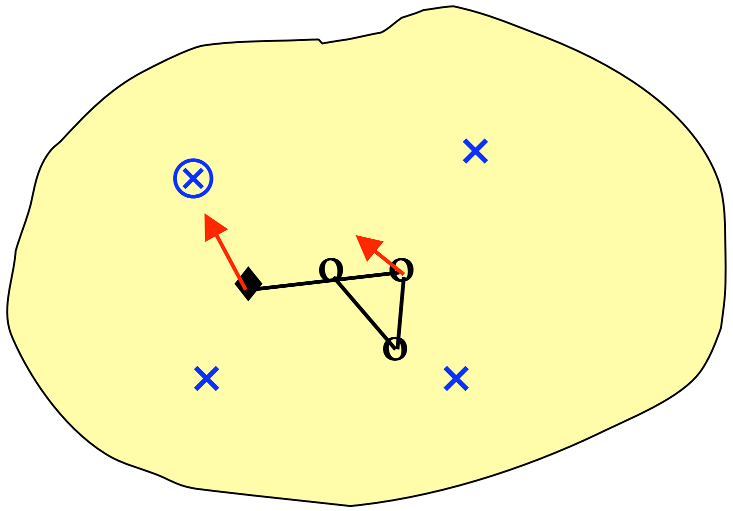
Visualizing the Self Organization Process



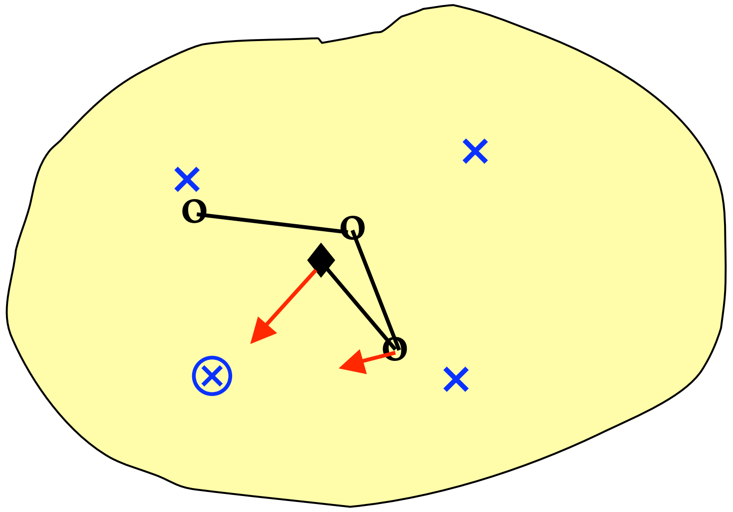
Suppose we have four data points (crosses) in our continuous 2D input space, and want to map this onto four points in a discrete 1D output space. The output nodes map to points in the input space (circles). Random initial weights start the circles at random positions in the centre of the input space.



We randomly pick one of the data points for training (cross in circle). The closest output point represents the winning neuron (solid diamond). That winning neuron is moved towards the data point by a certain amount, and the two neighbouring neurons move by smaller amounts (arrows).



Next we randomly pick another data point for training (cross in circle). The closest output point gives the new winning neuron (solid diamond). The winning neuron moves towards the data point by a certain amount, and the one neighbouring neuron moves by a smaller amount (arrows).



We carry on randomly picking data points for training (cross in circle). Each winning neuron moves towards the data point by a certain amount, and its neighbouring neuron(s) move by smaller amounts (arrows). Eventually the whole output grid unravels itself to represent the input space.

Overview of the SOM Algorithm

The algorithm operates on a spatially ***continuous input space*** in which the input vectors live. The aim is to map from this high dimensional space to a low dimensional spatially ***discrete output space***, the topology of which is formed by arranging a set of neurons in a grid. The SOM provides such a non-linear transformation called a ***feature map***.

The stages of the SOM algorithm can be summarised as follows:

1. ***Initialization*** – Choose random values for the initial weight vectors $\mathbf{w}_j$.
2. ***Sampling*** – Draw a sample training input vector $\mathbf{x}$ from the input space.
3. ***Matching*** – Find the winning neuron $I(\mathbf{x})$ with weight vector closest to input vector.
4. ***Updating*** – Apply the weight update equation $\Delta w_{ji} = \eta(t)\, T_{j,I(\mathbf{x})}(t)\, (x_i - w_{ji})$.
5. ***Continuation*** – Keep returning to step 2 until the feature map stops changing.

In the next lecture, we shall explore the properties of the resulting feature map and look at some simple examples of its application.

Overview and Reading

1. We began by defining what is meant by the terms Self Organizing Map (SOM) and Topographic Map.
2. We then looked at how to set up a SOM and at the components of self organisation: competition, cooperation, and adaptation. We saw that the self organization has two identifiable stages: ordering and convergence.
3. We ended with an overview of the SOM algorithm and its five stages: initialization, sampling, matching, updating, and continuation.

Reading

1. Haykin-2009: Sections 9.1, 9.2, 9.3
2. Beale & Jackson: Sections 5.1, 5.2, 5.3, 5.4, 5.5
3. Gurney: Sections 8.1, 8.2, 8.3
4. Hertz, Krogh & Palmer: Sections 9.4, 9.5
5. Callan: Sections 3.1, 3.2, 3.3, 3.4, 3.5