2. The Plastic Self Organising Map

2.1 Introduction

The Plastic Self Organising Map (PSOM) is the author's novel extension of Kohonen's Self Organising Map (SOM). The Extended Plastic Self Organising Map, (E)PSOM, is an extension to the PSOM. This Chapter details both of these algorithms.

The motivation for the PSOM was to create a SOM that could learn online using input and class sets that are non-stationary and not limited in size or shape. Traditional SOMs go through a training phase where a selection of patterns is shown to the network. Before the training phase can take place, an input set needs to be constructed with an exemplar pattern from each known class. If these exemplar patterns are not available at the time of training, then at runtime there may be a misclassification or the network may fail to respond at all. Once training is finished, the SOM would then be validated using an alternative input set. Depending on the outcome of the validation process, the SOM would be either used for its intended task or retrained.

To allow a network to learn from non-stationary class sets, it is necessary to reconsider the learning process. With static input sets, the typical train and validate method is a suitable learning scheme, however it is not suitable for dynamic input sets as the learning process should be continuous. For a network to operate on an infinite input set, it would need to treat each pattern identically. Traditional networks use a parameter which decays to allow later patterns to have less effect on the network. This parameter is often called a *learning rate* and is used to aid the network convergence by reducing the amount by which the network responds to later inputs. In a dynamic paradigm, such a parameter is not suitable as patterns arriving later in time would be treated as less important than earlier patterns. If a new class appears for the first time late in the input set, the learning rate may have decayed to such a point that the new pattern is ignored and lost.

Thus a move away from traditional training techniques is required. All training, validation and classification is required to be achieved online and each pattern needs to be treated identically.

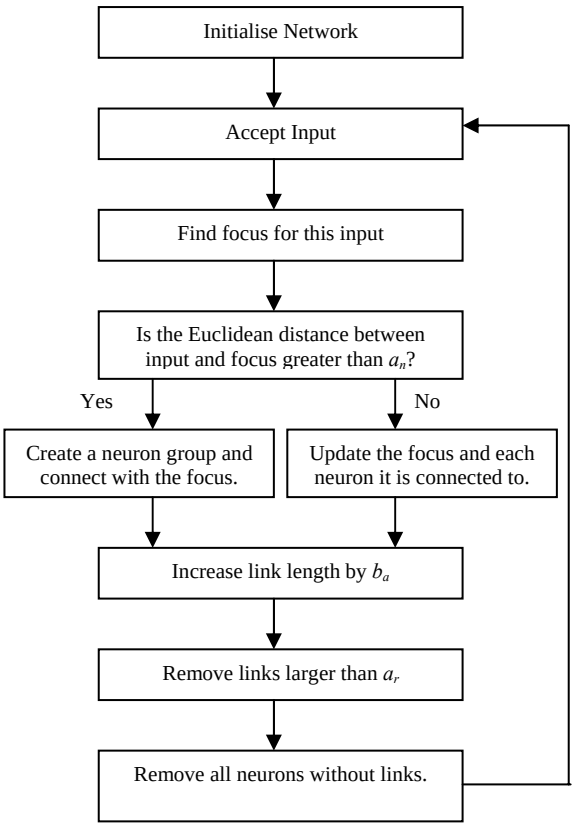
2.2 Structure of the PSOM

The PSOM consists of l neurons and m links (connections between neurons), both of which may vary over time. Each neuron, x_i, is similar to Kohonen's SOM in that it contains a vector (see Equations 1.1, 1.2) which has the same dimensionality, k, as the input space, E. For the sake of brevity, the neuron, x_i and its vector are considered one and the same. Thus, $x_i \in E$. The links are scalars, c_{ij}, which exist between two neurons (x_i and x_j). Each scalar represents the similarity in input space and time (iterations) between vectors contained within the neurons that it connects. A link may only connect two nodes. In the simplest case, neurons with similar vectors are connected by a small value link and those with dissimilar vectors are connected by large values. At each iteration, the links are increased by a value b_a, which means that the links do not only represent the spatial distance between two neurons but due to this ageing process, a temporal factor. The network size (neurons and links) is bounded only by the physical memory required to store and process it. A nomenclature for training parameters is such that *thresholds* are represented by a with a subscript and scaling and *incremental parameters* are represented by a b with a subscript.

2.3 PSOM Training Algorithm

2.3.1 Initialisation

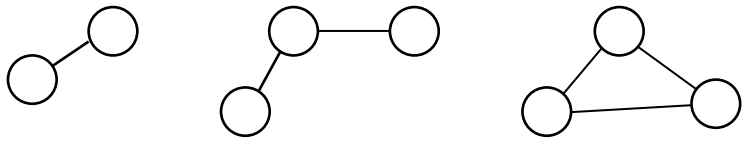
Figure 2.1 The Plastic Self Organising Map learning algorithm.



The training algorithm is as shown in Figure 2.1. After initialisation, the PSOM performs an iterative process that accepts an input, finds the most similar neuron, decides upon a course of action (whether to create a neuron or not), ages links and then removes neurons and links above a threshold. A number of parameters are introduced in this section, which are listed in Table 2.1.

Table 2.1 Parameters used in the PSOM.

Parameter	Name	Purpose	Value or Range
a_n	Node Building Parameter	Controls the ease with which neurons are created.	0 – 1.0 typically 0.29.
a_{cl}	Cluster Threshold	Controls whether a neuron is pushed away from the input or toward it.	0 – 1.0 typically 0.23.
a_r	Maximum Link Length	The upper bound on link lengths	1 – 100
b_c	Neuron update parameter	Either +ive or –ive depending on a_{cl}. Set algorithmically.	+0.01 -0.01
b_a	Link ageing parameter	How much links are aged each iteration.	0 – 1.0 Typically 0.01.



A B C

Figure 2.2 Different starting topologies that were briefly investigated. It was discovered that of these topologies, 'B' was the most successful.

The network is initialised with a small number of neurons and links, with random vectors and link lengths. Intuitively, the initial neurons in the network should not persist and should be removed. This is because the initial positions in the input space are randomly generated and thus are unlikely to be within the space of a class, especially in the case of a sparse[1] input space. The initial neurons are regarded as a structure onto which the useful parts of the network (those representing classes) will grow and are removed when the network has some useful neurons in it. A number of initial topologies were tested (Figure 2.2) and it was found empirically that three neurons with two links performed the best (B). With initial topology 'A', the link was easily broken and the neurons lost too quickly. Topology 'C' was too strong a group and the network often had difficultly removing this highly connected group. Thus, 'B' was a suitable balance between the two. Initial topologies with more neurons were considered, however the purpose of the network is to determine the optimal topology and thus the initial state is considered as being arbitrary.

[1] A sparse input space is one where the classes do not cover much of the space.

2.3.2 Finding and updating the Focus

For each input vector, u, presented, the network will find the neuron most similar to the input (calculated by Euclidean distance between the input vector and each neuron vector, Equation 1.4). The most similar neuron is called the *focus, z*. If the distance between the input and the focus is larger than the *node building parameter*, a_n, then a group of neurons is created with vectors similar to the input and attached to the focus. If the distance is less than a_n then the focus is made more similar to the input by a scaling value b_v (which is equivalent to a learning rate) using Equation 2.1. This learning rate was arbitrarily set to 0.9 and is a constant value that is not altered during any of the operations.

$$\Delta z(x) = b_v(u - z) \tag{2.1}$$

2.3.3 Neighbourhood Update

After the focus is updated, the neighbourhood around the focus is updated. The neighbourhood of the focus is defined as a neuron that is directly connected to the focus. This update is achieved in two steps. Firstly, the links between the focus and the neurons within the neighbourhood are altered such that they are a multiple of the Euclidean distance between the focus and that neuron. Secondly, the neurons are updated to either make them more or less similar to the input. This is done to a lesser degree than was applied to the focus.

The new length of the link, c_{zj}, is a product of the maximum link length, a_r, and the Euclidean distance between the updated focus, z and the neuron it connects, x_j. Therefore the new link length is given by Equation 2.2.

$$c_{zj} = a_r \cdot \left\| z - x_j \right\| \tag{2.2}$$

The change in neuron vector, x_j, is given by Equation 2.3 and is a product of the updated link between it and the focus, c_{zj} and the difference between the focus, z and x_j.

$$\begin{aligned} &\Delta\overline{x}_j = b_c \cdot c_{zj} \cdot \left(\overline{u} - \overline{x}_j\right) \\ &if \left\| z - x_j \right\| > a_{cl} \Rightarrow b_c = b_c \cdot -1 \end{aligned} \tag{2.3}$$

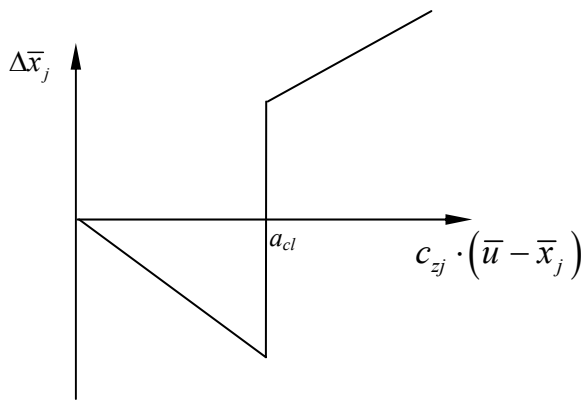


Figure 2.3 The non-linear, discontinuous neighbourhood function.

As link lengths are positive integers between 0 and a_r, the scalar, a_r, was set to 100 which makes b_c simply a scaling parameter of 0.01. The link lengths were made positive integers in this way such that they could be easily drawn on the network representation in the implementation. If the Euclidean distance between z and x_j is above the *clustering threshold,* a_{cl}, b_c is set to be negative. This gives the effect of making dissimilar neurons more dissimilar and similar neurons more similar (a rudimentary clustering process). The degree by which they are altered is dependent on the length of the link that connects them, c_{zj}. The function that this method produces is non-linear and discontinuous (Figure 2.3) where a_{cl} marks the transition whereby a neuron update is either positive (made more similar to the input) or negative (made more dissimilar). This discontinuous function was used to allow a more compact and computationally efficient implementation.

2.3.4 Network Ageing

After the focus neuron and its neighbouring neurons have been updated, all links within the network are then increased by scalar b_a, which represents the ageing process. Any links larger than threshold a_r are removed. Finally, any neurons that are without links are removed. The next input is presented and the process repeats *ad infinitum*.

The continual training process depends on values of thresholds for neuron construction, a_n, and for link removal, a_r. A small value of a_n will lead to patterns being represented by more neurons than absolutely required, leading to a reduction in efficiency. A large value of a_n would give a smaller network, but with the possibility of two similar classes being classed as one. Thus, a_n controls the granularity of the representation which is the amount of space each neuron covers. The link removal threshold, a_r, controls the rate at which the network forgets data. A large a_r will cause the PSOM to forget patterns but rejects noise more easily; a small a_r holds patterns for longer but loses some noise rejection capability. This decay of information is not to be confused with the learning rate decay of the static network. In static networks, the decay of the learning rate leads to later patterns having less significance whereas all patterns in the PSOM are treated with identical importance. The 'decay' in the PSOM is a forgetting factor of information already held by the network. Currently these thresholds are set either empirically or using heuristics as given in Section 2.3.6.

2.3.5 Classification

The network can be used for classification by labelling each neuron with a class label. Full identification can then be performed by comparison of a neuron's vector with input vectors contained within some library of test input data. When new neurons are added, the identification number is incremented and each neuron in the new group is given that class number. For an input vector already represented by a neuron within the network, the focus neuron is found, and the neurons around it are updated. If the link between the focus and a neighbouring neuron is less than a_{cl}, the neighbouring neuron will take the same class identification number as the focus.

The most useful outputs of the network are:

- The Euclidean distance between the focus and the input. This is also known as the Recognition Error. A small distance means that the network has successfully recognised the input, a large error suggests that the network has not recognised the input.
- The class for the last input presented. This is a classification output.
- The network will self-organise into a visual graph structure, displaying high dimensional temporal data in a two dimensional format.

These outputs can then be interpreted either by a user or by adding a look up table to interpret the self-assigned class numbers as known patterns.

2.3.6 Heuristics for setting network parameters

Setting the node building and clustering threshold parameters is vital for successful operation of the network. A heuristic was found for finding initial values for these parameters but it should be noted that the resultant parameters are not optimal and may need further tuning.

For this heuristic to be used correctly, the following requirements must me met:

- The maximum radius of a class must be known (which is not always the case).
- The input space must be normalised.

Normalisation is the process whereby the largest input is set to 1.0 and all the other inputs are linearly scaled between 0 and 1.0 as a proportion of the largest input. The motivation for Normalisation is so that every input is treated equally. For example, if one input had a range between 0 and 5000 and another input had a range between 0 and 2, then a change in the first output of 1 is less significant than a change in the second input of 1. Scaling all inputs between 0 and 1 removes this problem. The maximum radius of the class, u_r, is the distance from the notional centre of the class to the furthermost point of the class. The Node Building Parameter, a_n, can then be set such that it is larger than the maximum radius of the largest class. To make sure a_n is larger than u_r, a factor of u_r is added, as seen in Equation (2.4).

$$a_n \geq u_r + \left(u_r \cdot 0.1\right) \tag{2.4}$$

The Clustering Threshold, a_{cl}, can be then set smaller than a_n. A reasonable rule of thumb for a_{cl} is 80% of a_n (obtained empirically).

An ad-hoc method for selecting the ageing parameter is given in Chapter 6, Equation 6.3.

2.4 Extended Plastic Self Organising Map (EPSOM)

The EPSOM grew from modifications to the PSOM algorithm. As most of the operation is identical to the PSOM, only the extensions and alterations will be discussed here. Each modification is dealt with separately. A numerical analysis identical to that of the PSOM was conducted but is not included for the sake of brevity.

The motivation for the EPSOM was to remove the c*lustering parameter,* a_{cl} by using a continuous function and link lengths scaled by the *node building parameter,* a_n. It was found (Chapter 3) for the PSOM that a_{cl} was not as effective on the operation of the network as a_n on the artificial datasets and so it might be removed. This was found by noting the effect on the network's error and ability to classify correctly over varying values of a_{cl}. Removing a parameter from the operation of the network is desirable because it reduces the number of heuristics required to initially set the parameters and reduces the amount of tuning required by the user.

2.4.1 Continuous Neighbourhood Function

The EPSOM differs from the PSOM by having a continuous neighbourhood function. The PSOM's original discontinuous function was simple and computationally inexpensive. However, the discontinuity in the function led to similar patterns being classified differently. Conversely, incorrectly setting a_{cl} could mean very different network structures and outputs given very similar input data sets. Also, two similar neurons connected to the focus that were either side of the clustering threshold would be treated very differently. Intuitively, this is not a desirable effect, as one would expect similar neurons to be treated in a similar manner. To remove this sensitivity (or this parameter altogether), the discontinuous neighbourhood function was replaced by a continuous *sinc* function, similar to that of a typical SOM.

For an input x, the *sinc* function $s(x)$ is defined as:

$$s(x) = \frac{\sin(x)}{x} \tag{2.5}$$

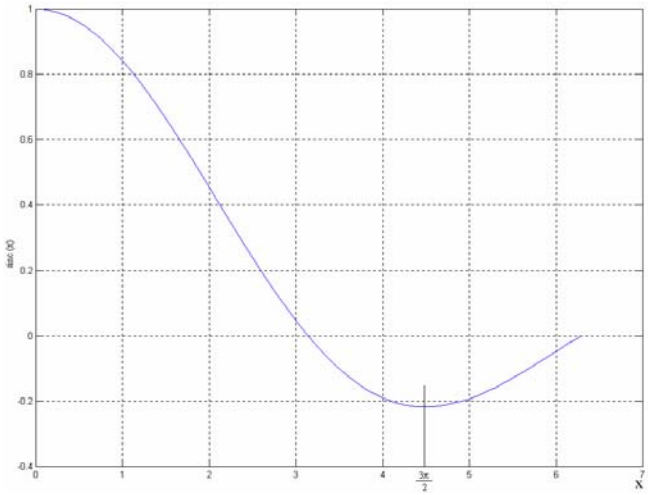


Figure 2.4 The sinc function showing the outermost bound that is used.

Graphically, a *sinc* function is represented in Figure 2.4. The range of the function that is used is between 0 and 3π/2, as marked in Figure 2.4. This range is used because it is required that similar patterns are made more similar and dissimilar patterns made more so. If a wider range is used then the sinusoidal oscillations would also become apparent in the structure of the network as neurons representing dissimilar patterns that happen to be 2π apart would not be pushed apart. This is shown in Figure 2.4 where the result of the *sinc* function is 0 at 6.28, or 2π. The range of 0-3π/2 can be selected by scaling the link length, c_{fj}, between 0 and 1 and then between 0 and 3π/2. Using this function over the range of 0 to a_n, the change in neuron x_j is as shown in Equation 2.7.

$$\Delta\bar{x}_j = \mathrm{s}\left(\frac{c_{fj}\pi}{2}\right)\left(\bar{u} - \bar{x}_j\right) \tag{2.6}$$

2.4.2 Scaled Link Lengths

The link length in the PSOM represents not only the spatial similarity of the neurons it connects, but also the temporal relationship between them. When a link is connected to the focus, it is updated to be the Euclidean Distance between the focus and the neuron it connects multiplied by the maximum link length, a_r. For the PSOM, this update is totally independent of the node building parameter (Equation 2.2).

However, the Euclidean Distances between neurons are very rarely 1.0 as the input is normalised (See Chapter 3, PSOM Test 1). When the neighbourhood function sets the lengths of the links, even links between two very dissimilar neurons will not be long enough to break. For example, if the maximum link length, a_r, is 90 and the distance is 0.6, the link length is only 54, which is much less than the link breaking threshold of 90 and so far too short to be broken.

By adding a factor involving a_n, the length of links then becomes dependent on a_n (Equation 2.8).

$$c_{zj} = a_r \cdot \left\| x_j - z \right\| \cdot \frac{\left\| x_j - z \right\|}{a_n} \tag{2.7}$$

which becomes

$$c_{zj} = \frac{a_r \cdot \left\| x_j - z \right\|^2}{a_n} \tag{2.8}$$

This simple addition of a scaling factor means that when the neighbourhood function is applied to the links, the dissimilar neurons are pushed apart such that links can be broken purely through neighbourhood rather than relying on the ageing process.

2.5 Comparisons with SOM

This section illustrates the similarities and differences between the PSOM and EPSOM (shortened to (E)PSOM) and Kohonen's SOM. The largest similarity is the construction of each neuron; both the SOM and the (E)PSOM have a reference vector (Equations 1.1 and 1.2) and, both algorithms also use random initialisations. The next similarity between the standard SOM and the (E)PSOM is the definition of a focus and the manner in which it is found (Equation 1.4).

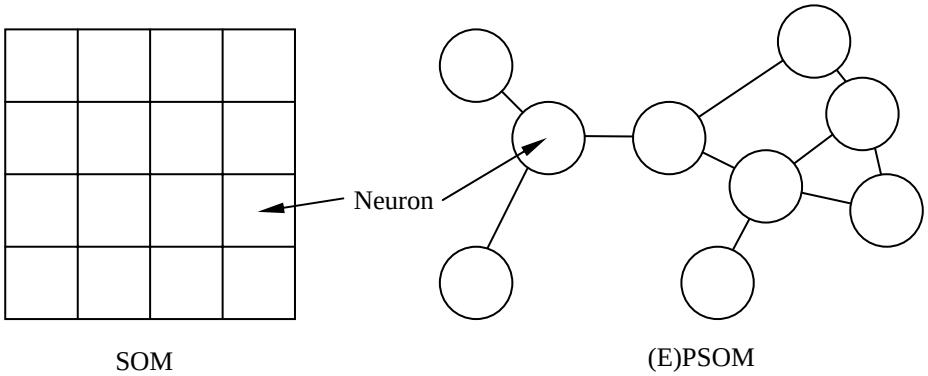


Figure 2.5 Comparison of network architectures. The SOM is an ordered grid of neurons where location in the grid defines neighbourhood (may also be hexagonal). The (E)PSOM algorithms are graphs that uses weighted links to define neighbourhood.

The first difference in algorithms is in architecture. Figure 2.5 compares the SOM and (E)PSOM architectures. The SOM is an ordered grid of neurons where the neighbourhood is decided by the location in the grid. The (E)PSOM uses a graph structure where the neighbourhood is decided by the weight links. The 2D grid of the SOM (either square or hexagonal) may not be the best manner in which to show the data.

The next difference between the algorithms is in the learning rules. The classic SOM uses a Gaussian for the neighbourhood function (Equation 1.7) with a learning rate (α) that deals with the focus and the neighbourhood. The PSOM uses a discontinuous function (Equation 2.3, Figure 2.3) and a separate, linear update for the focus (Equation 2.1). The EPSOM uses a smooth *sinc* function (Equation 2.6) and a separate, linear update for the focus (Equation 2.1).

The differences continue with the update for the links. The SOM uses a decaying parameter to reduce the size of the neighbourhood where the (E)PSOM algorithms have no decaying parameter. Instead, the neighbourhood is first controlled by the spatial difference between connected neurons and secondly by the link ageing process. Thus, the SOM shows purely spatial information about the input set whereas the (E)PSOM algorithms encode a spatio-temporal relationship between patterns in the input space.

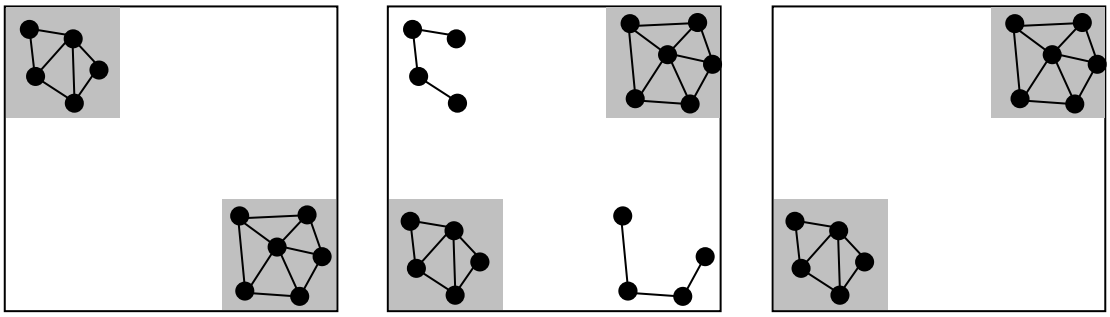
The final difference is in the removal of links and neurons. The SOM has a static topology: the number of neurons does not change and thus neither does the number of links. However, the number of neurons and links in the (E)PSOM algorithms are always changing, depending on the input set.

2.6 Analytical Comparisons with GNG-U

The GNG-U algorithm (as described in Chapter 1) anticipated some elements of the initial versions of the PSOM algorithm. However, the algorithm was found after the PSOM was created. This section demonstrates the analytical differences between the GNG-U and PSOM algorithm. A numerical comparison can be found in Chapter 3.

2.6.1 Purpose

The two methods differ in a subtle manner. Both algorithms are intended for the tracking of non-stationary signal spaces. The GNG-U algorithm tracks a moving distribution, either slow moving or quick moving (by jumping). Figure 2.5 shows the GNG-U algorithm tracking two separate distributions. The PSOM result for this problem is given in Chapter 3.



a) 20000 Signals b) 25000 Signals c) 40000 Signals

Figure 2.5: GNG-U tracks two classes in the input space of an unknown probability distribution. a) Situation before the distribution changes. b) State of the network 5000 steps after the distribution changes. c) After 40000 steps, all the units have adapted to the new distribution.

2.6.2 Structure

The structures of the GNG-U and (E)PSOM have similarities, both consisting of an undirected graph, but there are some very critical differences summarised in Table 2.2 below.

Table 2.2: Comparisons between GNG-U and (E)PSOM Structure

GNG-U	(E)PSOM
Neurons contain reference vector, accumulated error and accumulated utility. No class number stored.	Neurons contain only reference vector and class reference number. No error or utility is stored or calculated.
Links hold ageing parameter but this does not have any association with length.	Links hold a connection value that represents both spatial and temporal information.
Input space no larger than 3 dimensions.	Input space can be any dimension.

2.6.3 Parameters

The only similarity between the parameters within the networks is the link removal threshold (a_{max} in GNG-U and a_r in (E)PSOM). For GNG-U, when the link age is larger than a_{max}, the link is removed. However, this link age is only temporal and does not have any bearing on the topology of the network. In (E)PSOM, a link is removed when the length of the link is larger than a_r. However, the difference here is in link semantics. The link length of a link in (E)PSOM is both temporal and spatial. Links are removed when either the link has aged or the neurons that it connects are dissimilar.

2.6.4 Training Algorithm

The largest differences between the two methods lie in the training algorithm. Both similarities and differences are summarised below in Table 2.3.

Table 2.3: Comparisons between GNG-U and (E)PSOM Algorithm

GNG-U	(E)PSOM
Focus is found by finding minimum distance between input and node	The same
Second winner is found	Not required.
Neurons in neighbourhood are moved toward the input by some fraction	Neurons in neighbourhood moved toward input by a function
First and second winner connected by a link	Not applicable.
If first and second winner connected by a link, set age to zero	All links connected to the focus are set to the correct topographical length (counter the effect of ageing)
Remove links with age larger than a threshold	The same
Only one node added at a time.	Small clusters of neurons added.
Neurons only added after a number of iterations	Neurons may be added on any iteration, depending on density of input data
Update node utility	No utility, so update not required.
Remove only the node with smallest utility	Remove nodes that are unlinked. May remove more than one
Degrade node utility and error	Not applicable

Increase link age only on links connected to focus (local process)	All links aged (global process)
No class numbers	Class number distributed from focus to local neighbourhood

Analytically, the differences are quite considerable. In general, GNG-U relies on information being stored in both neurons and links for the topology of the network to change. These are two separate processes. The link process removes the links when they reach a certain age (without activation) whereas the neurons are removed when their cumulative utility-error ratio drops below a threshold. With the (E)PSOM, the emphasis is on the removal of links. When links are removed, neurons are made more vulnerable until unlinked neurons are removed. Also, GNG-U does not keep track of the different classes as the PSOM does. This is a minor difference as the GNG-U algorithm would not need much alteration for this function to be added.

The two algorithms' structures provide different types of information. GNG-U shows purely spatial information. However, the PSOM shows a mix of spatial and temporal information. A large, spaced out group in the GNG-U shows that the neurons are covering a large area of the input space. In the PSOM, this represents a class that does not appear very often in the search space or two different classes.

2.7 Chapter Summary

This chapter has presented the algorithms of the Plastic Self Organising Map (PSOM) and the Extended Plastic Self Organising Map (EPSOM). The algorithms are inspired from Kohonen's Self Organising Map. Unlike their static counterparts, these dynamic structures are able to alter their topology to accept new classes and reject old classes from the input data set. This allows the network to detect novelty and to detect when a pattern pertaining to a new class is seen by the network for the first time. Finally, an analytical comparison with the only other similar existing method, Growing Neural Gas with Utility (GNG-U), is presented that shows significant differences in the semantics of the topology, the method of neuron creation/removal and the ability to classify.