4. The Temporal Plastic Self Organising Map

4.1 Introduction

The PSOM (Chapter 2) extends the generic SOM by adding temporality to the links and allowing a network to grow and shrink through time without the restrictions of a 2 dimensional grid. The coarse temporal information encoded implicitly in the links includes some temporal information about the input space but it is not easy to separate from the spatial information. Also, there is likely to be more temporal information about the order in which patterns are presented to the network that is not represented.

In static Neural Networks, the order in which patterns are presented to the network is only a concern when training the network and any temporal structure found therein is ignored. Static Neural Networks tend to require patterns from different classes to be presented in rotation. Furthermore, a detriment in learning has been found if classes are shown in their entirety sequentially (pp.23, Haykin 1999). If shown sequentially, a Static Neural Network will learn the first class presented well but will not learn the later classes as the learning rate will have reduced. Static Neural Networks do not encode this information for use but use it as a preference in training methods. In traditional time series Neural Networks, a window of time is used to represent a pattern. Although this could be seen as some sort of temporal encoding, the temporality is within the pattern itself, not a relationship between the sequence of patterns and thus the Neural Network learns the time sequence as a whole, not how that time sequence relates to the patterns presented before and after.

In many real systems, there can be much useful information stored in the order in which patterns arrive at the input, especially when detecting novelty. If one expects a process to cycle in a certain order, any change in the order is of interest to the user. A static Neural Network will discard this information, concentrating on the

classification or clustering of the pattern space. The PSOM takes some steps towards encoding the order of patterns, but this information is not easily retrieved from the network as the links encode both temporal and spatial information.

The Temporal Plastic Self-Organising Map (TPSOM) attempts to solve this problem by encoding the order in which the patterns arrive. Once this information is encoded, it can be used to predict, given a class, which class is most likely to appear next. This can be used for classification, prediction or the detection of temporal novelty. Novelty, as specified before, is the arrival of a pattern that does not belong to a class that has been seen before. Temporal novelty is where the *sequence* of the classes being shown changes. Spatial novelty is also detected by the arrival of classes new to the input data set. This would be shown by the addition of neurons and links within the PSOM.

4.2 Structure

The TPSOM consists of neurons and links. A neuron contains a vector of equivalent dimension as the input space. This vector holds an example of the input space. Each link represents a scalar probability and a direction (from one neuron to another), which is a difference from the PSOM. A link, c_{ij}, connects neuron x_i to neuron x_j. If x_i is the focus, the link c_{ij} is the probability that x_j will be the next focus. Thus, a link may only connect two neurons and unlike the PSOM, has a direction.

The TPSOM consists of l neurons and m links, both of which may vary over time in number. Each neuron, x_i contains a vector which has the same dimensionality as the input space, E. For the sake of brevity, the neuron x_i and its vector are considered one and the same. Thus $x_i \in E$. In the simplest case, two neurons (x_i and x_j) are connected by a single link, c_{ij}, of probability 1.0. The sum of the probabilities for all links from neuron x_i is 1.0. An example network structure is shown in Figure 4.1.

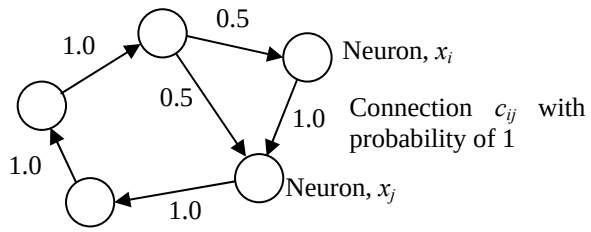

Figure 4.1: The structure of the TPSOM, showing 5 neurons and 6 links.

4.3 TPSOM Algorithm

The algorithm is what sets the TPSOM apart from the PSOM and its variants. The algorithm is simple (seen in Figure 4.2) and should be compared with the PSOM algorithm in Figure 2.2 (page 39). The network is initialised with one neuron that contains a randomly generated point in the input space. This is for implementation reasons, but it is removed early on if it does not represent a class in the input space. For each input, u, presented, the network finds the neuron that is most similar to the input, which is called the *focus*, z. Similarity is measured using Euclidean distance. If

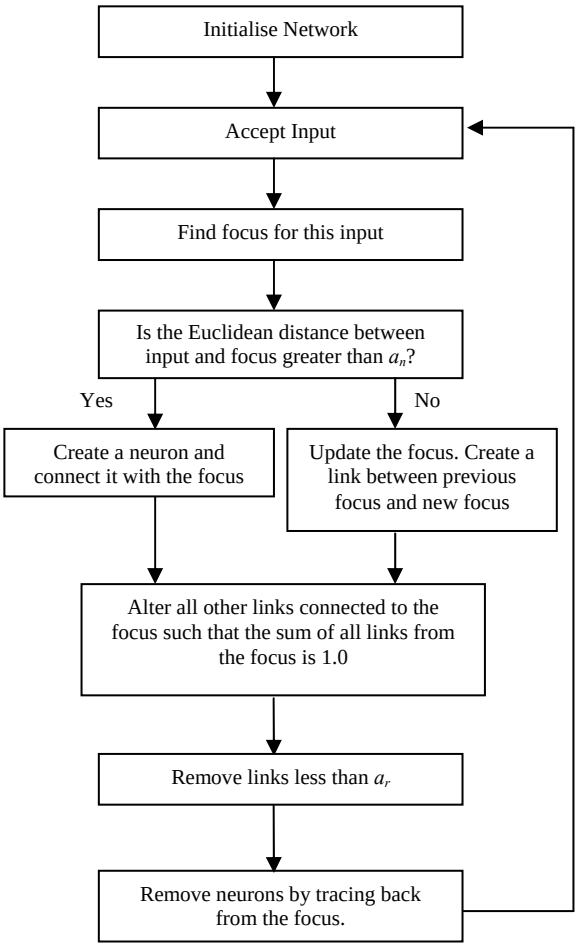

Figure 4.2: The Temporal Self Organising Map learning algorithm.

this distance is larger than the *Node Building Threshold,* a_n, then a new neuron is created with the vector identical to the input. If not, then the focus is updated such that it is made more similar to the input. If a link between the current focus and the previous focus already exists, then it is strengthened using a *learning rate*, b_r, if not, then a new link is created from the previous focus to the current focus. A typical value for b_r is 0.01. Here, the term learning rate is used to show how much effect each new input has on the links of the network. For the link between last focus, y, and the current focus, z, the link, c_{zj}, is updated as follows:

$$\Delta c_{yz} = b_r \cdot c_{yz} \tag{4.1}$$

Once a link has been created or updated, all the other links starting from the last focus must be updated such that the sum of all link probabilities from the last focus is 1. For each link, c_{yj}, from the old focus, y, to each neuron, j, with a total of n links, the update is given as:

$$\Delta c_{yj} = -\left(\frac{\Delta c_{yz}}{n-1}\right) \tag{4.2}$$

After this learning step, the links that are longer than the link removal threshold, a_r, are removed. Thus:

$$\begin{aligned} &c_i > a_r \Rightarrow c_i = 0 \\ &0 \geq i \geq m \end{aligned} \tag{4.3}$$

The loss of this link means that the total probabilities amongst all links from the focus no longer sum to 1. Therefore, they are updated in a similar manner as in Equation 4.2. Assuming the link being removed is between neurons i and j and is therefore denoted by c_{ij}, and assuming there are n other links from i, such that c_{ik} is a link between neuron i and k, then the change in each link is given as:

$$\Delta c_{ik} = \left(\frac{c_{ij}}{n-1}\right) \tag{4.4}$$

Originally, only neurons that were without links were removed, in keeping with the PSOM algorithm. However, an alteration to this algorithm was subsequently adopted. Hindsight allows the new algorithm to be given here, although the reasoning behind it is included later (Section 4.4).

Neuron removal is achieved by tracing back from neuron to neuron along each incoming link, starting from the focus. If a neuron, y, connected to the focus by a link c_{yz}, is not the last focus, $y \neq z\ (t\text{-}1)$, and it has no incoming links, then it is removed. All links from this neuron are then removed. A neuron without any incoming links represents a state which cannot be reached or one that is no longer part of the input set.

The result of the algorithm is a directed graph that self-organises into a temporal structure using only the input data. Given the current input, the next input can be found by following the link from the focus with the highest probability.

4.4 Properties of the TPSOM

The output of the TPSOM is two fold. Firstly, there is the recognition error, i.e. the Euclidean distance between the given input and focus. Secondly, there is the output graph which shows the temporal relationship between different input classes. This output could be used to represent systems which display Markovian-like (see below) processes (Grimmett and Stirzaker, 1994).

A Markov Chain is a process whereby states are connected by probability values or distributions. Thus, the probability of a particular next state can be calculated given the current state. An example of this is given in the technique evaluation (Chapter 5). As the graphical output of the TPSOM is similar to a state transition diagram, the structure of a network can be represented by a state transition matrix. In this manner, a TPSOM neuron is equivalent to a Markovian state. This is desirable as it is temporal (or state) information that the TPSOM is attempting to capture from the system. A state transition matrix is square, each row representing the current state and each column representing each possible next state (Equation 4.5). The element p_{ij}

is the probability that the next state will be *j* given that the current state is *i*. Values in the matrix are probabilities and each row sums to 1 (Equation 4.6).

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1s} \\ p_{21} & p_{22} & \cdots & p_{2s} \\ \cdots & \cdots & \cdots & \cdots \\ p_{s1} & p_{s2} & \cdots & p_{ss} \end{bmatrix} \tag{4.5}$$

where

$$\left(\sum_{i=1}^{s} p_{ij} \right) = 1 \qquad 0 > j > s \tag{4.6}$$

This matrix allows the tracking of the probabilistic structure of the network. Previously, the only error measure that was useful was the Euclidean distance between a given input, *u*, and the focus, *z*. However, for a given model, of known transition matrix, $\bar{P}$, one can calculate the model error, e_m, between $\bar{P}$ and the network's transition matrix, *P* (Equation 4.7).

$$e_m = \left\| \bar{P} - P \right\| \tag{4.7}$$

This error can be used to measure how well the TPSOM tracks the given model, producing an error over time.

Although the TPSOM works with Markovian Chains, there are a few distinct differences with existing algorithms, such as:

- Markovian processes can be non-stationary. This means that the contents of the state transition matrix – representing the probability of a transition from one state to another – can change over time. Most typical Markovian techniques assume that this matrix is static.
- No prior assumption of states. Most Markovian techniques assume the number of states in the system. As the TPSOM adds neurons to represent

states, there is no need to assume how many states will be required, which is similar to estimating topology structures in Neural Networks.

- Non-Markovian statistical processes may be represented by a TPSOM. This is because a Markovian process assumes that the number of states is fixed and that the probability between two states is static.

It should be noted that although the theories within the field of Markovian dynamics presented here are well established, the work presented here arrived at a similar point through the slight alteration of the PSOM.

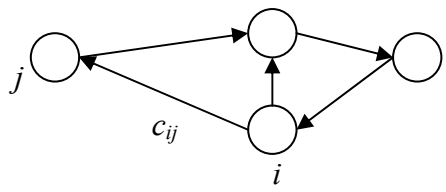

Figure 4.3: A typical TPSOM layout before link removal. Assume that link c_{ij} is approaching removal ($c_{ij} \approx a_r$).

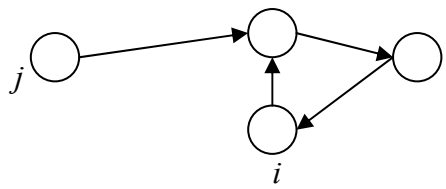

Figure 4.4: After the removal of link c_{ij}. This would occur if the other link from *i* approached 1.

Another property of the vanilla TPSOM algorithm is that some neurons are never removed. This is because neurons are only removed when all the links associated with that neuron are removed. It is quite possible that the following situation can occur:

Figure 4.3 and Figure 4.4 show a situation where the link c_{ij} has been removed. After the link removal, there is no method for the link from j to be altered as this would require the state j to be excited. Assuming that state j is no longer excited by the system providing the input, then the link from this neuron would not be altered and the neuron would remain in the system. It is undesirable to have neurons representing

states that are no longer part of the input state. Therefore, a new neuron removal criterion is required for the TPSOM.

One possible solution is that a neuron is removed if no links lead to it. Intuitively, this suggests that states that can not be excited given the current graph are to be removed. In the above example, as soon as link c_{ij} is removed, neuron j will also be removed. This added rule has difficulties when the network is young and the number of neurons is small. Therefore, neuron removal can only be accomplished when the number of neurons, m, is larger than 3, an arbitrarily small value. This method is not suitable for one class of models, where the input system builds long chains (Figure 4.). This is a valid system output, especially for systems with many states without repeating any steps. An example of this type of system can be found in a production line where the product goes through many states before its finish. No step is repeated but the state matrix is valid. However, given the above alteration to the algorithm, the neuron i would be removed on the first pass and then neurons along the chain would be subsequently removed until only 3 neurons are left. This is not entirely desirable for the construction of long chains or for transient systems.

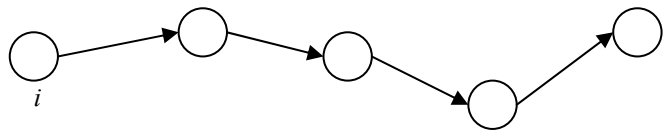

Figure 4.5: A possible system chain. The neuron *i* represents the first neuron in the chain.

Another possibility is to add a link age onto the links. This additional parameter would increment every time an input was shown to the network (on each iteration). When this age parameter becomes large (in comparison to some other threshold) the link would be removed. If a link joins the previous focus and the current focus, then it is set to zero. This would allow the non-cyclic chain (Figure 4.5) to exist but not for infinite time (as before). This system is very much like the GNG algorithm of Fritzke where age information is collected on the links for the purpose of removing them. The problem with this technique is that two more parameters are required. One parameter would set how much the links age with each iteration and a threshold parameter would set the calculation for link removal. Although the iterative ageing parameter could be set to 1, there is still the matter of setting the threshold. By

setting this parameter, one is defeating the objective of the network: to find temporal structures in the data without any prior knowledge. Being able to set this parameter would imply there is prior knowledge of the temporal structure. Temporality could be built into the link values themselves (creating spatio-temporal link like the PSOM) but this would reduce the usefulness of the resultant graph as links would no longer sum to 1.

Another possible solution is to use a self-tuning process to set the value of a threshold. The smaller the threshold, the more rapidly links will be removed, the larger the threshold, the more slowly links will be removed. By setting a desirable number of links, the ageing threshold parameter becomes a function of this number. Intuitively, it is much more viable to set a desirable number of links as this value can be determined from hardware or operational considerations. There is no external imposition on the temporal constructions.

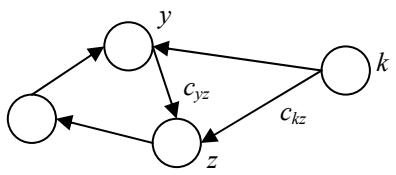

Figure 4.6: An example of the focus triggering a scan of the incoming links. Here, neuron *k* would be removed.

The final possible solution (and the one chosen) uses the focus, z, to trigger a scan of incoming links. After all the other link updates for the new focus have been completed, the algorithm would look back along the links that lead to the focus (except the one that has just been updated). If y is the last focus, then a neuron k which has a link c_{kz} (where $k \neq y$) can be checked for incoming links (c_{*k}). If this neuron does not have any incoming links, then it can be removed. Figure 4.6 shows an example using the previous notation. In this example the new focus, z, has two input links (c_{yz} and c_{kz}) and one output link. As c_{yz} was the link that was just updated, neuron y is not checked. As neuron k has a link to the current focus (c_{kz}) it is examined for incoming links. As neuron k has no incoming links, it would be removed from the network.

Figure 4.7: Two groups of neurons. Group A is a group that is now persistent in the network. Group B consists of states that are no longer produced by the system.

There are instances where this algorithm would not work. Figure 4.7 shows a network structure that consists of two groups that have been separated by the input process. Group A is still a functioning group and Group B consists of neurons that represent states that are no longer part of the input set. This could happen if the two groups were initially linked and a heavy bias to the states in Group A severed the links. In this case, the neurons in group B would never be investigated as none of the links travel back from group A to group B. Given that group A consists of all the active foci, a link is required from group B to group A for the removal process to be triggered. Although this problem exists, it would be rare and easily detected by a separate algorithm that would remove neurons based on how often the state is visited. Another situation where neurons will remain in the network even after the state has been removed is in the construction of long chains. Figure 4.3 shows a case for which a long chain has been created. Here, neuron *k* represents a state that is no longer present in the system. As it is not directly connected to the focus, it cannot be removed from the network.

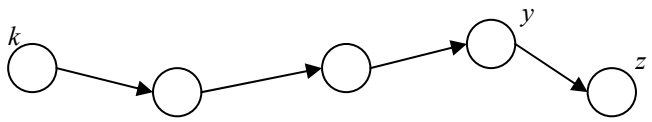

Figure 4.3: A long chain is constructed, neuron *k* represents a state that is no longer part of the input system.

This method of neuron removal is preferable to the others as it does not require any threshold, will allow systems to grow from 1 neuron without removing any neurons early on, will allow the creation of long chains and is computationally inexpensive: only those neurons connected directly though unidirectional links are investigated for removal. Perhaps the best solution would be to create a hybrid technique consisting of the ones presented here and using the best qualities of each algorithm to remove neurons in different situations. Difficulties would arise in determining which

algorithm to use and whether removing such neurons as k in Figure 4.3 is actually desirable.

The resulting structure of the TPSOM can be used to predict the next state of the input environment. This is achieved by knowing the current focus in the network and then analysing the link from the focus. The link with the highest probability leads to the next most likely state in the input environment. Temporal novelty, a change in the order of events, is detected by the network by the addition of new links or neurons.

4.5 Chapter Summary

This chapter has presented the Temporal Plastic Self Organising Map (TPSOM) Neural Network. This alteration to the PSOM algorithm (Chapter 2) allows the TPSOM to represent temporal input systems by constructing Markov-like chains. A TPSOM can be used to predict the next state, classify the current state and detect changes in the order of presentation (temporal novelty).