The Self-Generating Model: an Adaptation of the Self-organizing Map for Intelligent Agents and Data Mining

Amy de Buitléir, Mark Daly, and Michael Russell

Faculty of Engineering and Informatics,
Athlone Institute of Technology, Athlone, Ireland
amy@nualeargais.ie,{mdaly,mrussell}@ait.ie

Abstract. We present the Self-Generating Model (SGM), a new version of the Self-organizing Map (SOM) that has been adapted for use in intelligent data mining ALife agents. The SGM sacrifices the topology-preserving ability of the SOM, but is equally accurate, and faster, at identifying handwritten numerals. It achieves a higher accuracy faster than the SOM. Furthermore, it increases model stability and reduces the problem of "wasted" models. We feel that the SGM could be a useful alternative to the SOM when topology preservation is not required.

Keywords: self-organizing map, artificial life, intelligent agents

1 Introduction

Data mining is the extraction of insights from data. In contrast with an ordinary database search or query, where the key features and relationships are known; in data mining, they have to be discovered [1, p. 5].

Data mining includes a wide range of tasks. *Landscape mining* explores the data to find the space of possible inferences (the data's "landscape") and to identify interesting patterns before leaping in with more traditional data analysis tools [2]. *Classification* assigns objects to predefined categories based on the attributes of the objects [3]. *Clustering* (also known as unsupervised classification or exploratory data analysis) partitions items into a set of clusters such that items within a cluster have similar characteristics, and items in different clusters have different characteristics [3, 4]. *Prediction* (also known as forecasting) estimates future (or unknown) data based on present data and past trends, validating hypotheses [3] [5, p. 4]. *Regression* identifies functions which map data objects to prediction variables [3] [5, p. 4]. *Modelling* produces a (typically simpler) representation of the data that captures important features and relationships. Such models can be used for classification, prediction, and to provide insight about the data. *Visualisation* makes insights understandable by humans[3].

The self-organizing map (SOM) provides a way to represent high-dimensional data in fewer dimensions (typically two), while preserving the topology of the

input data [6]. The SOM is a set of models associated with nodes in a regular grid. Patterns that are similar to each other in the high-dimensional space are mapped to models that are near each other on the grid. (There are exceptions to this topology-preserving property, however; see Villmann et al. [7]).

In addition to topology preservation, SOMs have benefits that make them useful for artificial life (ALife) and intelligent agents. They are easy to understand and implement. The SOM models can be inspected directly, which makes it easier to debug problems with the implementation or the learning function. After a SOM has been trained, labels can be assigned to the nodes to allow it to be used for classification. It can also be used to cluster data; a U-matrix (whose elements are the Euclidean distance between neighbouring cells) will have high values at the cluster edges [8].

SOMs have an established place in the data mining tool set, especially for clustering and classification. They have also been used, often with modifications, in ALife [9, 10]. and artificial intelligence [11, 12]. De Buitléir et al. [13] described an ALife species designed for data mining, called *wains*. Wains live in, and subsist on, data; data mining is their survival problem. Their brains use modified SOMs to model their environment and to identify patterns in the data. Wains were originally used with images of handwritten numerals [13], but they have also been applied to the task of speech recognition, identifying audio samples of spoken numerals [14, 15].

In this paper we present the Self-Generating Model (SGM), a new version of the SOM that is modified for use by intelligent agents. However, the modifications may be useful in other applications as well. We will describe the traditional SOM and historical modifications to it (Section 2), discuss the goals of our modifications (Section 3), describe the SGM algorithm (Section 4), discuss the experimental set-up (Section 5), compare the behaviour of the SOM and the SGM (Section 6) and present our conclusions (Section 7).

2 The SOM algorithm

SOM training (see Algorithm 1) is unsupervised. The elements (patterns) of the input data are typically numeric vectors, but they can be any data type so long as we can define a measurement of similarity between two patterns, and a method to make one pattern more similar to another, by an arbitrary amount. The SOM models are arranged on a (typically two-dimensional) grid of fixed size. The models must be initialised.

Step 3 ensures that as additional input patterns are received, nodes that are physically close respond to similar patterns in the input data. Thus, the resulting grid preserves the topology of the original high-dimensional data. SOMs "translate *data similarities* into *spatial relationships*" (emphasis in the original) [16].

The traditional SOM has been adapted and extended in many ways. Common modifications include use of grids in non-euclidean spaces [16], dynamically increasing the size of the grid [17], replacing the grid with a hierarchical arrange-

Algorithm 1 SOM algorithm.

For each input pattern,

1. Compare the input pattern to all models in the SOM. The node with the model that is most similar to the input pattern is called the *winning node*.
2. The winning node's model is adjusted to make it slightly more similar to the input pattern. The amount of adjustment is determined by the learning rate, which typically decays over time.
3. The models of all nodes within a given radius of the winning node are also adjusted to make them slightly more similar to the input pattern, by an amount which is smaller the further the node is from the winning node.

ment of nodes [18], and combining with principal component analysis [19]. There is extensive literature on SOMs with two or more of these modifications [20–27].

3 Adapting the SOM for intelligent data mining ALife agents

The requirements for a classifier used in intelligent data mining ALife agents are rather different than for more common applications. For example, in recognising handwritten or spoken numerals, it is not necessary to preserve the topology of the input data set. (We may not be interested in knowing whether a particular '3' is more similar to an '8' or a '6'.) In an early implementation of wains, De Buitléir et al. [13] made a small modification to the SOM to improve performance. By updating only the winning node, the topology-preserving ability of the SOM was sacrificed in favour of speed [13].

If we dispense with topology preservation, what is the cost? Consider that in addition to a SOM-like classifier, the brain of an intelligent data mining ALife agent might include a mechanism that uses the information provided by the classifier to determine what response to take. This is the approach used by de Buitléir et al[13]. For convenience, we'll call this mechanism the *decider*. Suppose that the classifier assigns the label a to the current scenario, and the decider does not know a good response to a. If the classifier preserves the topology of the input data, the decider can look for the nearest neighbour of a for which it *does* know a good response, and choose that (see Figure 1). If a good response to a neighbour of a is likely to be a good response to a, this tactic could benefit the agent's survival.

However, there may be other ways to achieve the same result. The classifier could report the similarity of the scenario to *all* models, including the model labelled a. (This information is calculated anyway as part of the SOM algorithm.) Without needing to know anything about the topology used by the classifier, the decider can look for known responses to models that are similar, and choose a response that is known to be good for a similar model (see Figure 2). Thus, we can sacrifice topology preservation in favour of other goals.

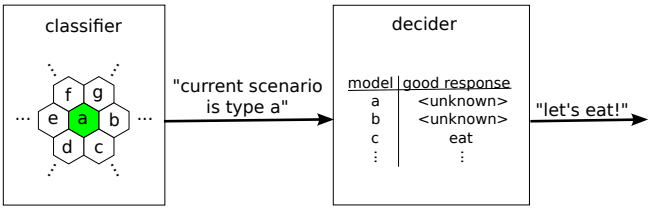

Fig. 1. Decision-making using a classifier that preserves topology.

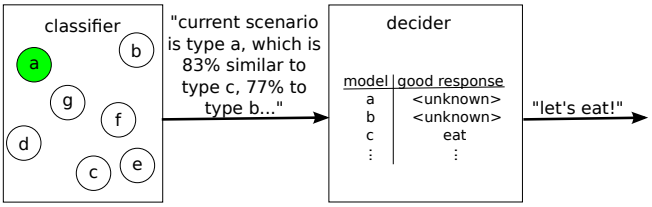

Fig. 2. Decision-making using a classifier that uses disconnected models and does not preserve topology.

One advantage of the SOM for intelligent agents is that the models can be extracted from the classifier, making it easier to understand how an agent perceives the object, and evaluate any decisions the agent makes in response. This is a feature we wanted to keep. In a traditional neural net, it can be difficult or even impossible to analyse why the net makes certain classifications.

Many SOM modifications are motivated by a desire for greater accuracy in classifying; however, this may not be necessary for some agent implementations. In a multi-agent system one can ask the same question of multiple agents, each with a different set of lifetime experiences, to get independent opinions. By averaging the responses, a "wisdom of the crowd" effect could produce greater accuracy than a single agent could achieve. Thus, increasing accuracy was not one of our goals.

However, we did have a goal of **early accuracy**. Agents cannot wait until they have a full, final set of models (they continue to learn throughout their lives) to begin learning rules for survival. Agents need to be able to "hit the ground running". An agent should have a useful, if small, set of models early in life; this will allow it to experiment with possible responses to objects in their environment, and to learn from the results.

Another goal in adapting the SOM was to have **stable models.** Agents make decisions based on the patterns that they encounter, and the mental categories (node labels) associated with the patterns. Agents experiment by trying different actions in response to each cluster of patterns. Through trial and error, each agent develops rules that select the appropriate action to take in response to each pattern cluster. If models change to such a degree that they no longer

match patterns that they used to match, agents may need to "unlearn" existing rules and replace them with new ones.

As will be shown, SOM models can be very unstable. This can make it more difficult for agents to learn appropriate responses for their environment. For example, suppose the environment has both edible and poisonous berries; and an agent can distinguish by some characteristic such as colour. We would expect an agent to develop at least one model that matches edible berries but not poisonous ones. The agent has a better chance of surviving if it learns to eat objects that match this model. Now imagine that the model changes so much that it now matches the poisonous berries. The eating response that the agent has learned is now dangerous. In order to survive, the agent must "unlearn" the eating response and learn a more appropriate action. The situation is even worse if the model changes from matching poisonous berries to edible ones. The agent may have ruled out eating anything that matches this model, and may never try eating the edible berries.

Once those goals are met, there are additional features that would be desirable in a modified SOM. We want a **faster algorithm**; this can be achieved if we **minimise what we call "wasted" models**. Models that will can not be used to classify future patterns are wasted; the computational effort to create and update those models is unnecessary. This is especially important because instead of working with a single SOM, we may require a population of agents with SOMs in their brains, thus amplifying any inefficiencies in the algorithm.

Finally, we hope to use the modified SOM in a variety of data mining applications. Therefore, we wanted a **generic algorithm**, not one that was tailored to a specific type of data such as images or audio samples.

Why not modify the SOM, when other classifier algorithms are available that are also capable of unsupervised learning? It is often impractical for an agent to keep a copy of every data input it has encountered during its life; fortunately SOMs only require that we keep the models. Contrast this with an algorithm such as k-means which requires that we re-calculate the centroid at each step, accessing all of the data seen previously [4]. Particle Swarm Optimisation (PSO) similarly iterates over all the data, making it unsuitable for this application [28].

Learning Classifier Systems (LCSs) [29, Section 3.9] learn the best action to take in response to a set of conditions. As such, the LCS might be suitable as a replacement for *both* the classification and decision-making components in a wain (as we will discuss in Section 7). However, it seems overkill to replace just the classifier with an LCS. Finally, as mentioned earlier, SOM models can be inspected directly. A trained neural network stores what it has learned as weights[4]; making it difficult to extract the models.

4 Self-Generating Model

To satisfy the above goals, we adapted the basic SOM algorithm to produce the SGM algorithm (see Algorithm 2). The SGM can be initially empty, or it can be initialised with a set of (possibly random) models. Step 2, adjusting the winning

node, has been modified to allow the classifier to grow as needed and produce models that are useful as soon as they are created. In addition, Step 3 of the SOM algorithm, adjusting models in the neighbourhood of the winning node, has been eliminated in an attempt to improve performance, and to minimise wasted models. The *difference threshold* helps to ensure that models do not change too much during the lifetime of the SGM, providing model stability Like the SOM, the SGM design is generic; it has not been tailored to a specific kind of data.

Algorithm 2 SGM algorithm.

For each input pattern,

1. Compare the input pattern to all models in the SGM. The node with the model that is most similar to the input pattern is called the *winning node*.
2. If the difference between the input pattern and the winning node's model is greater than the *difference threshold*, and the SGM is not at capacity (number of models $<$ maximum), a new model is created that is identical to the input pattern. Otherwise, the winning node's model is adjusted to make it slightly more similar to the input pattern. The amount of adjustment is determined by the learning rate, which typically decays over time.

5 Experimental set-up

The experiments described in this paper used the MNIST database, which is a collection of images of hand-written numerals from Census Bureau employees and high-school students [30]. The training set contains 60,000 images, while the test set contains 10,000 images. All images are 28x28 pixels, and are grey-scale as a result of anti-aliasing. The centre of pixel mass of the numeral has been placed in the centre of the image. Sample images are shown in Figure 3. We used the database images without modification.

Fig. 3. Sample images from the MNIST database [30].

For all experiments, the SOM used the learning function given by Equation 1,

$$f(d,t) = re^{-\frac{d^2}{2w^2}}, \tag{1}$$

where

$$r = r_0\left(\frac{r_f}{r_0}\right)^a, \quad w = w_0\left(\frac{w_f}{w_0}\right)^a, \quad \text{and} \quad a = \frac{t}{t_f}.$$

The parameter r_0 is the initial learning rate, r_f is the learning rate at time t_f, w_0 is the radius of the initial neighbourhood, and w_f is the radius of the neighbourhood at time t_f. For the winning node, $d = 0$, and Equation 1 reduces to Equation 2,

$$f(t) = r = r_0 \left(\frac{r_f}{r_0} \right)^a . \tag{2}$$

We used Equation 2 as the learning function for the SGM in all experiments. Thus, at all times the learning rate of the SGM matches the learning rate of the winning node in the SOM. This permits a fairer comparison of the SOM and the SGM.

We use the Mean of Absolute Differences (MAD) as a measure of difference between two images. The absolute difference between each pair of corresponding pixels in the two images is calculated and the mean taken, to obtain a number between 0 (identical) and 1 (maximally dissimilar). As all the images in the MNIST database have the same size, viewing direction (normal to the plane of the image, from above), and comparable intensity, the MAD is an appropriate difference metric.

The models for each SOM were initialised with images containing random low pixel values similar to the background of the MNIST images. Each SGM was initially empty, having no nodes or models.

Once a classifier has been trained, the nodes must be labelled with the numeral represented by the associated model before the classifier can be used for testing. To do this, we counted the number of times each node was the winning node for each numeral during the training phase. The node was then labelled with the numeral it most often matched.

5.1 Experiment 1: Early accuracy

Recall that agents cannot wait until they have a full, final set of models to begin learning appropriate responses. This experiment determines how long it takes to develop a useful, if small, set of models. For this experiment, we used a SOM and SGM of similar size. After 25 training images, chosen at random, had been presented to a classifier, we tested its accuracy with the entire test set, presented in random order. We repeated the process with various amounts of training, from 50 up to 500 images.

Table 1 shows the configuration of the classifiers for this experiment. The values r_0 and r_f were chosen so that the learning rate would start at maximum and be near zero by the end of training. The values w_0 and w_f were chosen through experimentation. The value of t_f is the number of training images.

Recall that if the difference between an input pattern and the winning node's model is greater than the difference threshold, and the SGM is not at capacity, a new model is created. Once capacity is reached, the SGM will always update the most similar model, which increases the chance of a model eventually representing a different numeral than it was created for. We wanted to compare the SGM with a small (10x10) SOM. However, an SGM may not create the maximum

Table 1. Configuration of SOM and SGM in Experiment 1

variable	SOM	SGM
node count	100	96
grid type	rectangular	unconnected nodes
r_0	1	1
r_f	1×10^{-4}	1×10^{-4}
w_0	2	not applicable
w_f	1×10^{-4}	not applicable
t_f	60000	60000
difference threshold	not applicable	0.165

number of models. In order to maximise model stability, we used an SGM with a maximum capacity of 2000 models, and relied on the difference threshold to indirectly control the number of models created.

To find a reasonable value for this difference threshold, we selected a random set of 500 images and measured the MAD between all pairs of images. The results are shown in Table 2. Experimenting with the values near the two means, we discovered that a threshold of 0.165 resulted in the SGM creating 96 models, which was useful for comparison with the 100 models in the SOM.

Table 2. Analysis of MAD between MNIST images, based on a sample of 500 images. The first column contains the *mean of the mean* absolute difference; the second, the *standard deviation of the mean.*

	mean	std. dev.
same numeral	0.135	0.0436
different numerals	0.171	0.0374

5.2 Experiment 2: Full training run

To compare the overall accuracy of the SOM and SGM, we created a randomised list of all 60,000 images in the MNIST training set. The training images were then presented, in this order, to a small and large SOM, and a small and large SGM. Next, we created a randomised list of all 10,000 images in the test set, and presented those to the SOM and the SGM for classification. This allowed us to compare the accuracy, speed, model stability and number of wasted models for the two classifiers.

Table 3 shows the configuration of the classifiers for this experiment. In preliminary trials, we found that the accuracy of both the SOM and the SGM

depends strongly on the number of models, weakly on r_0, and very weakly on the other configuration parameters. Therefore, for this experiment we chose to vary the classifier size, while keeping r_0 and r_f constant. The values r_0 and r_f were chosen so that the learning rate would start at maximum and be near zero by the end of training. The values w_0 and w_f were chosen through experimentation. The value of t_f is the number of training images.

Table 3. Configuration of SOM and SGM in Experiment 2

variable	SOM	SGM
grid size	4×4, 6×6, 8×8, 10×10, 15×15, 20×20, 25×25, 30×30, 35×35, 40×40, 45×45, 70×70	initially empty, grows as needed
grid type	rectangular	unconnected nodes
r_0	0.1	0.1
r_f	1×10^{-4}	1×10^{-4}
w_0	2	not applicable
w_f	1×10^{-4}	not applicable
t_f	60000	60000
difference threshold	not applicable	0.09, 0.1, 0.105, 0.11, 0.115, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2, 0.21

6 Results and interpretation

6.1 Experiment 1: Early accuracy

Figure 4 compares the accuracy of the SOM and SGM during the early part of training. The SGM reaches a usable level of accuracy faster than the SOM.

6.2 Experiment 2: Full training run

Figures 5 and 6 show one pair of small classifier models after all of the training images have been presented to the small classifiers. From Figure 5 we can see that many of the models are blurry combinations of more than one numeral. The topology of the input data has been partially preserved; models of the same numeral tend to be near each other.

There are four shaded models in Figure 5. They were not winning nodes at any point during testing, were not used to classify testing images and are counted as "wasted". An unmatched model could be assigned the same label as was assigned to a majority of its neighbours. However, this would result in the left pair of unmatched models (shaded) being assigned labels for the numeral '1',

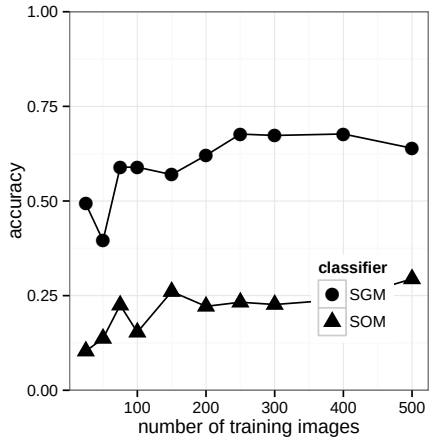

Fig. 4. Early accuracy comparison

even though they are clearly better matches for '0'. The right pair of unmatched models are very ambiguous; it may be better not to use them.

Figure 6 shows the small SGM after training. We can see that the topology has not been preserved. Unfortunately, there are still many ambiguous models, perhaps due to the small size of the classifier.

Figure 7 compares model stability for the SOM and SGM. To measure this, we noted the first numeral matched by the model. (In the case of an SGM, this is the numeral the model was created in response to.) We compared this to the numeral used to label the model's node (at the end of training). If the numerals were the same, we counted the model as stable. The SGM consistently achieved higher model stability.

Figure 8 compares model usage. A model is counted as "used" if it was the winning node at any point during testing, otherwise it is considered "wasted". The SGM consistently used more of its models, reducing the problem of wasted models.

Figure 9 shows the time required for training and testing the SOM and SGM. For all but the smallest classifiers, the SGM is considerably faster than the SOM. We believe the reduction in processing time occurs primarily because the SGM only updates one model during training, while the SOM updates the models in the neighbourhood of the winning node. In addition, the SGM has fewer models during the early part of training, and therefore does not need to make as many comparisons as the SOM does.

Figure 10 compares the accuracy of the classifiers. The accuracy is the number of times that an image was correctly identified, divided by the total number of images. The accuracy of the two methods appears to be comparable. For all but the smallest SOMs, a small fraction of the nodes were not winning nodes at any point during training. We could have labelled these nodes to match the majority

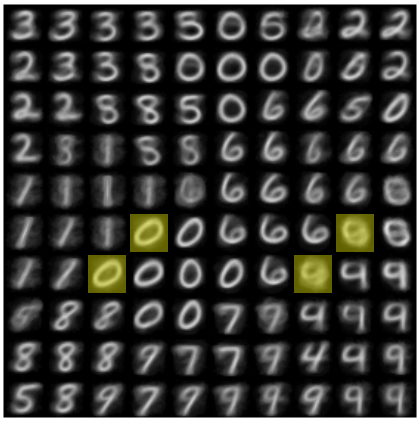

Fig. 5. Small SOM after all 60,000 training images have been presented. Models are arranged in a grid. Wasted models are shaded.

Fig. 6. Small SGM after all 60,000 training images have been presented. Models are unconnected; they are shown here in the order they were created. There were no wasted models.

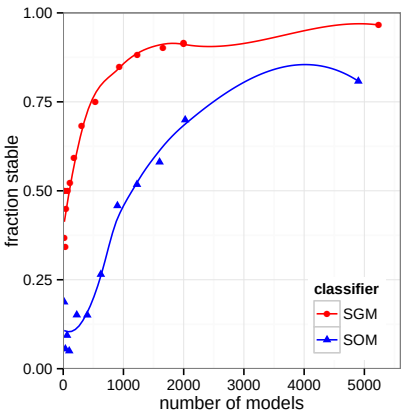

Fig. 7. Model stability. Larger values are better. The lines show a loess (local polynomial regression) data fit.

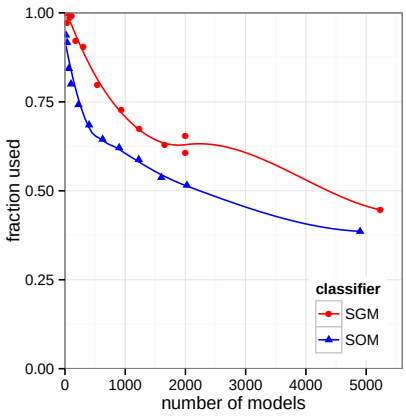

Fig. 8. Model usage. Larger values are better. The lines show a loess (local polynomial regression) data fit.

of their neighbours. However, there were not enough to significantly impact the accuracy, so we counted them as correct answers.

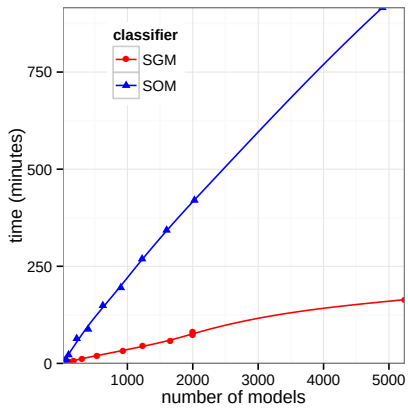

Fig. 9. Processing time. Smaller values are better. The lines show a loess (local polynomial regression) data fit.

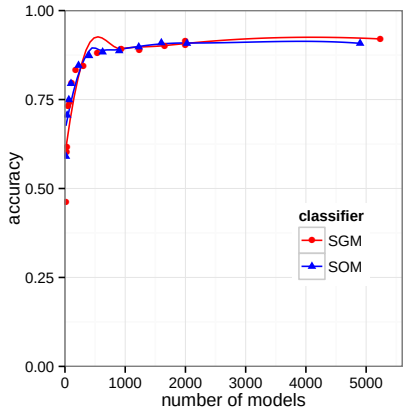

Fig. 10. Accuracy. Larger values are better. The lines show a loess (local polynomial regression) data fit.

The code and results for this experiment are open access [31, 32]

7 Conclusions

The overall accuracy of the two classifiers is comparable, but the SGM achieves a higher accuracy faster. This could allow an agent to make good survival decisions with less training. In the SGM, model stability was higher. Furthermore, the SGM significantly reduces wasted models, making it faster than the SOM. We feel that the SGM could be a useful component for implementing intelligent agents. Furthermore, it may be useful for other clustering or classification applications.

Areas for future research include comparing the accuracy of the SOM and SGM on other types of data (e.g., audio), and designing a brain based on the SGM that would allow an agent to learn to survive in an environment through experimentation. The new brain design could be compared to both the existing design for the wain, and to an LCS.

References

1. Gorunescu, F.: Data mining concepts, models and techniques (2011), `http://site.ebrary.com/id/10454853`
2. Menzies, T.: Beyond data mining. IEEE Software 30(3), 92 (2013)

3. Goebel, M., Gruenwald, L.: A survey of data mining and knowledge discovery software tools. ACM SIGKDD Explorations Newsletter 1(1), 20–33 (1999), http://www.lcb.uu.se/users/janko/data/goebel{_}sigkddexp99.pdf
4. Xu, R., Wunsch, D., I.: Survey of clustering algorithms. Neural Networks, IEEE Transactions on 16(3), 645–678 (May 2005), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1427769
5. Fayyad, U.M.: Advances in Knowledge Discovery and Data Mining (American Association for Artificial Intelligence). MIT Press (1996), http://www.amazon.co.uk/dp/0262560976
6. Kohonen, T.: Self-organizing maps. Springer series in information sciences, 30, Springer, Berlin, 3rd edn. (December 2001), http://www.worldcat.org/isbn/3540679219
7. Villmann, T., Der, R., Herrmann, M., Martinetz, T.: Topology preservation in self-organizing feature maps: exact definition and measurement. Neural Networks, IEEE Transactions on 8(2), 256–266 (Mar 1997), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=557663
8. Ultsch, A., Siemon, H.P.: Kohonen's self organizing feature maps for exploratory data analysis. In: Widrow, B., Angeniol, B. (eds.) Proceedings of the International Neural Network Conference (INNC-90), Paris, France, July 913, 1990 1. Dordrecht, Netherlands. vol. 1, pp. 305–308. Kluwer Academic Press, Dordrecht, Netherlands (1990), http://www.uni-marburg.de/fb12/datenbionik/pdf/pubs/1990/UltschSiemon90
9. Saunders, R., Gero, J.S.: Artificial creativity: A synthetic approach to the study of creative behaviour. Computational and Cognitive Models of Creative Design V, Key Centre of Design Computing and Cognition, University of Sydney, Sydney pp. 113–139 (2001)
10. Martins, J.M., Miranda, E.R.: A connectionist architecture for the evolution of rhythms. In: Applications of Evolutionary Computing, pp. 696–706. Springer (2006)
11. Riga, T., Cangelosi, A., Greco, A.: Symbol grounding transfer with hybrid self-organizing/supervised neural networks. In: Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on. vol. 4, pp. 2865–2869. IEEE (2004)
12. Saunders, R., Gemeinboeck, P., Lombard, A., Bourke, D., Kocabali, B.: Curious whispers: an embodied artificial creative system. In: International conference on computational creativity. pp. 7–9 (2010)
13. de Buitléir, A., Russell, M., Daly, M.: Wains: A pattern-seeking artificial life species. Artificial Life 18(4), 399–423 (2012)
14. Salaja, R.T., Flynn, R., Russell, M.: Automatic speech recognition using artificial life. In: 25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communities Technologies (ISSC 2014/CIICT 2014). Institution of Engineering and Technology (IET) (2014), http://dx.doi.org/10.1049/cp.2014.0665
15. Salaja, R.T., Flynn, R., Russell, M.: Evaluation of wains as a classifier for automatic speech recognition. In: Signals and Systems Conference (ISSC), 2015 26th Irish. pp. 1–6 (June 2015), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7163770
16. Ritter, H.: Self-organizing maps on non-euclidean spaces. Kohonen maps 73, 97–110 (1999)
17. Alahakoon, D., Halgamuge, S.K., Srinivasan, B.: Dynamic self-organizing maps with controlled growth for knowledge discovery. Neural Networks, IEEE Trans-

actions on 11(3), 601–614 (May 2000), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=846732
18. Koikkalainen, P., Oja, E.: Self-organizing hierarchical feature maps. In: Neural Networks, 1990., 1990 IJCNN International Joint Conference on. vol. 2, pp. 279–284 (June 1990)
19. Kohonen, T.: The adaptive-subspace som (assom) and its use for the implementation of invariant feature detection. In: Proc. ICANN. vol. 95, pp. 3–10 (1995)
20. Batista, L.B., Gomes, H.M., Herbster, R.F.: Application of growing hierarchical self-organizing map in handwritten digit recognition. In: Proceedings of 16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI). pp. 1539–1545 (2003)
21. Cecotti, H., Belaïd, A.: Rejection strategy for convolutional neural network by adaptive topology applied to handwritten digits recognition. In: Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on. vol. 2, pp. 765–769 (Aug 2005), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1575648
22. Mohebi, E., Bagirov, A.: A convolutional recursive modified self organizing map for handwritten digits recognition. Neural Networks 60, 104 – 118 (2014), http://www.sciencedirect.com/science/article/pii/S0893608014001968
23. Ontrup, J., Ritter, H.: A hierarchically growing hyperbolic self-organizing map for rapid structuring of large data sets. In: Proceedings of the 5th Workshop on Self-Organizing Maps, Paris (France) (2005)
24. Pakkanen, J.: The evolving tree, a new kind of self-organizing neural network. In: proceedings of the Workshop on Self-Organizing Maps. vol. 3, pp. 311–316. Citeseer (2003)
25. Rauber, A., Merkl, D., Dittenbach, M.: The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. Neural Networks, IEEE Transactions on 13(6), 1331–1341 (Nov 2002)
26. Shah-Hosseini, H.: Binary tree time adaptive self-organizing map. Neurocomputing 74(11), 1823 – 1839 (2011), http://www.sciencedirect.com/science/article/pii/S0925231211000786, adaptive Incremental Learning in Neural NetworksLearning Algorithm and Mathematic Modelling Selected papers from the International Conference on Neural Information Processing 2009 (ICONIP 2009)ICONIP 2009
27. Zheng, H., Shen, W., Dai, Q., Hu, S., Lu, Z.M.: Learning nonlinear manifolds based on mixtures of localized linear manifolds under a self-organizing framework. Neurocomputing 72(1315), 3318 – 3330 (2009), http://www.sciencedirect.com/science/article/pii/S0925231209000605, hybrid Learning Machines (HAIS 2007) / Recent Developments in Natural Computation (ICNC 2007)
28. Van der Merwe, D., Engelbrecht, A.P.: Data clustering using particle swarm optimization. In: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on. vol. 1, pp. 215–220. IEEE (2003)
29. Brownlee, J.: Clever algorithms : nature-inspired programming recipes. Lulu, [Place of publication not identified] (2011), http://www.worldcat.org/search?qt=worldcat{_}org{_}all&q=9781446785065
30. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
31. de Buitléir, A.: Software release: som v9.0 (Jan 2016), http://dx.doi.org/10.5281/zenodo.45039, http://dx.doi.org/10.5281/zenodo.45039
32. de Buitléir, A.: exp-som-comparison v0.1.0.0 (Jan 2016), http://dx.doi.org/10.5281/zenodo.45040, http://dx.doi.org/10.5281/zenodo.45040