Supervised Hybrid SOM-NG Algorithm

Mario J. Crespo-Ramos, Iván Machón-González, Hilario López-García
Área de Ingeniería de Sistemas y Automática (DIEECS)
Universidad de Oviedo
Gijón (Asturias), Spain
E-mail: crespomario@uniovi.es, machonivan@uniovi.es, hilario@uniovi.es

José Luis Calvo-Rolle
Escuela Universitaria Politécnica de Ferrol.
Departamento de Ingeniería Industrial
Universidad de A Coruña
A Coruña, Spain
E-mail: jlcalvo@cdf.udc.es

***Abstract*—The hybrid SOM-NG algorithm was formulated to improve the quantization precision in Self Organizing Maps by the means of combine both SOM and Neural Gas properties using a parameter γ to tune the topology preservation. A supervised learning algorithm is proposed to take advantage of the balanced hybrid algorithm. The proposed algorithm makes a linear approximation of the goal function for every Voronoi region. The algorithm gives good estimations and well balanced prototype positions combining the benefits of the original algorithms.**

***Index Terms*—hybrid algorithm, supervised learning, neural networks, self-organizing mapping, neural gas.**

I. INTRODUCTION

The Self-Organizing Map (SOM) [1] is a very popular algorithm because of its many features, specially the topological preservation between input and output spaces. Most of the uses of SOM were related with dimensionality reduction and unsupervised training, but SOM has been used in a supervised way in different applications. Supervised SOM was developed by different authors in different ways specially for classification purposes, but in this work numeric modeling will be studied.

Supervised modeling was approached in different ways. Additive composition of several supervised SOM networks [2] was developed from the theory that an additive composition of linear functions can be estimated with an additive composition of neural networks. Continuous Interpolation Self-Organizing Maps (CI-SOM) [3] are created using interpolation methods to approximate a continuous function using discrete prototype vectors. For time-series regression, special networks were created, Temporal Kohonen Map (TKM) [4] and Recurrent SOM [5] are examples of the use of leaky integrators for temporal sequence processing, they are compared in [6].

The main drawback of SOM training is the lack of quantization precision, specially if compared to Neural Gas (NG) [7] algorithm that does not take into account topological order aiming to minimize the quantization error. In [7], a regression method is also presented, using a reference value for the the prototype vector's position and a gradient vector for its Voronoi region.

Using these algorithms has a main advantage of using local approximations for each prototype vector, i.e., a model in every Voronoi region, so a closer model can be obtained if compared with using a single model for the whole data set. Using local linear models also require less computational resources, as they use simple mathematical techniques.

A hybrid algorithm was proposed based on both SOM and NG in [8]. This algorithm combines their properties and obtains a more balanced result achieving a trade-off between topology preservation and quantization error, i.e., between data projection and accuracy. This hybrid algorithm is briefly explained in Section II. Using this algorithm, a new estimation tool will be formulated in Section III and Section IV will explain the training procedure. The algorithm is tested in Section V and discussed in Section VI.

II. SOM-NG ALGORITHM

The SOM-NG algorithm is based on two different kernels:

$$h_{NG}(v, w_i) = exp\left(- \frac{k(v, w_i)}{\gamma^2 \cdot \sigma(t)} \right) \quad (1)$$

$$h_{SOM}(v, w_i) = exp\left(- \frac{s(r_{i*}, r_i)}{\sigma(t)} \right) \quad (2)$$

where $k(v, w_i)$ and $s(r_{i*}, r_i)$ are rank functions, γ is the topology preservation constant and $\sigma(t)$ is the neighborhood radius in training epoch t.

Each kernel determines the behaviour of each algorithm, i.e., h_{NG} is the influence over quantization error, as in NG, and h_{SOM} represents the topology preservation in a SOM-like way. For the sake of simplicity the parameters will be removed from expressions and h_{NG} and h_{SOM} will be used instead of $h_{NG}(v_j, w_i)$ and $h_{SOM}(v_j, w_i)$ in the whole paper.

The ranking function $k(v, w_i)$ is the position of prototype vector w_i in the distance ranking to data vector v in the input space. The best matching unit, i.e., the closest one, gets ranking 0, the next closer one gets 1 and continues until $m-1$, where m is the number of prototype vectors.

Ranking function $s(r_{i*}, r_i)$ is a bit more complex because it is a modified ranking in distance between the best matching unit r_{i*} and unit r_i considering both of them on the output space. The modified ranking imposes that map units at same distance from a map unit must have same ranking value, so ranking values in a square lattice will be like 0,1,1,1,1,5,5, and so on. Calculating this ranking is a bit more complex than a monotonically increasing one, but it is constant during

 ISBN: 978-1-61208-172-4

the training, so it is calculated once before training starts. This modified ranking function obtains more robust results than a monotonous one because of the lack of random tie-breakers. The modified ranking is similar to a sigmoid function of distance, and it produces a slimmer Gaussian bell in the prototypes close to the *bmu* with a wider base in the farther ones.

The neighborhood radius $\sigma(t)$ was chosen to decrease exponentially to improve the algorithm's steadiness, the recommended expression is:

$$\sigma(t) = \sigma_{t0} \cdot \left(\frac{\sigma_{t_{max}}}{\sigma_{t0}} \right)^{t/t_{max}} \tag{3}$$

where t is the training epoch, t_{max} is the number of training epochs, σ_{t0} is the initial value and σ_{tmax} is the final value. Appropriate values for them are: $\sigma_0 = m/2$ and $\sigma_{t_{max}} = 0.001$, since it should tend to zero in order to minimize the quantization error.

In the unsupervised algorithm, the energy cost function optimizes quantization in a cooperative way. The energy cost function according to the squared Euclidean distance is:

$$E = \sum_{i,j} h_{NG} \cdot h_{SOM} \cdot \|v_j - w_i\|^2 \tag{4}$$

The batch version of the algorithm is obtained using Newton's method (5).

$$\Delta w_i = -J_E(w_i) \cdot H_E^{-1}(w_i) \tag{5}$$

where J_E is the Jacobian matrix of the energy cost function E and H_E is the Hessian matrix. Thus they are calculated from (4) and the following expressions are obtained:

$$J_E(w_i) = -2 \cdot \sum_j h_{SOM} \cdot h_{NG} \cdot \overrightarrow{d_{ji}} \tag{6}$$

where $\overrightarrow{d_{ji}} = v_j - w_i$ is the distance vector for data point v_j and prototype vector w_i.

$$H_E(w_i) = 2 \cdot \sum_j h_{SOM} \cdot h_{NG} \tag{7}$$

The increment for each prototype vector w_i in every epoch is calculated using (6) and (7) by substitution in (5):

$$\Delta w_i = \frac{\sum_j h_{SOM} \cdot h_{NG} \cdot \overrightarrow{d_{ji}}}{\sum_j h_{SOM} \cdot h_{NG}} \tag{8}$$

And the updating rule is:

$$w_i = \frac{\sum_j h_{SOM} \cdot h_{NG} \cdot v_j}{\sum_j h_{SOM} \cdot h_{NG}} \tag{9}$$

This updating rule is very similar to batch SOM and batch NG [9] ones. If the h_{SOM} term is deleted, i.e., replacing it by the unity, the updating rule is the NG one and the opposite is obtained canceling the h_{NG} term, it results the batch SOM one, with the difference of the modified ranking kernel. In fact, values of γ over 80 usually make the h_{NG} tend to the unity and that is the reason why the algorithm behaves in a SOM way for high values of γ.

This algorithm is tested and discussed in [8].

III. Supervised SOM-NG algorithm

Once the previously developed SOM-NG algorithm was presented, a supervised learning rule will be added. The aim is to approximate an unknown scalar field $f(v)$, defined in a multidimensional input space with the following expression:

$$\tilde{f}(v) = y_{i*} + a_{i*} \cdot (v - w_{i*}) \tag{10}$$

where y_{i*} is a reference value in the position of w_{i*} in the input space and a_{i*} is the gradient in the Voronoi region defined by w_{i*}, which is the best matching unit for input vector v.

The energy cost function for Supervised SOM-NG is:

$$E_S = \sum_{i,j} h_{SOM} \cdot h_{NG} \cdot \left(f(v_j) - \hat{f}_i(v_j) \right)^2 \tag{11}$$

where $\hat{f}_i(v_j) = y_i - a_i \cdot (v_j - w_i)$ is the approximation for data vector j using prototype vector i.

As it was done with w_i, Newton's method will be employed to calculate the learning rules for the estimation parameters:

$$\Delta y_i = -J_E(y_i) \cdot H_E^{-1}(y_i) \tag{12}$$

$$\Delta a_i = -J_E(a_i) \cdot H_E^{-1}(a_i) \tag{13}$$

Both Jacobian and Hessian matrices are calculated with respect of y_i:

$$J_E(y_i) = -2 \cdot \sum_j h_{SOM} \cdot h_{NG} \cdot \left(f(v_j) - \hat{f}_i(v_j) \right) \tag{14}$$

$$H_E(y_i) = 2 \cdot \sum_j h_{SOM} \cdot h_{NG} \tag{15}$$

As the variation term for the Voronoi region $a_i \cdot (v_j - w_i)$ is symmetrical around the region center w_i, the whole term can be despised and the increment becomes:

$$\Delta y_i = \frac{\sum_j h_{SOM} \cdot h_{NG} \cdot (f(v_j) - y_i)}{\sum_j h_{SOM} \cdot h_{NG}} \tag{16}$$

The following updating rule is obtained:

$$y_i = \frac{\sum_j h_{SOM} \cdot h_{NG} \cdot f(v_j)}{\sum_j h_{SOM} \cdot h_{NG}} \tag{17}$$

The same process is carried out with a_i:

$$J_E(a_i) = -2 \cdot h_{SOM} \cdot h_{NG} \cdot \overrightarrow{d_{ji}} \cdot \left(f(v_j) - \hat{f}_i(v_j) \right) \tag{18}$$

$$H_E(a_i) = -2 \cdot h_{SOM} \cdot h_{NG} \cdot \left(\overrightarrow{d_{ji}} \cdot \overrightarrow{d_{ji}} \right) \tag{19}$$

 ISBN: 978-1-61208-172-4

The increment for a_i is:

$$\Delta a_i = \frac{\sum_j h_{SOM} \cdot h_{NG} \cdot \overrightarrow{d_{ji}} \cdot \left(f(v_j) - \hat{f}_i(v_j) \right)}{h_{SOM} \cdot h_{NG} \cdot \left(\overrightarrow{d_{ji}} \cdot \overrightarrow{d_{ji}} \right)} \quad (20)$$

Since expression (20) cannot be simplified, an increment rule is obtained instead of an absolute updating rule.

IV. Procedure of the Algorithm

The prototype vectors and the estimation parameters are randomly initialized with small random values within the interval (0, 0.1).

begin initialize randomly prototype vectors w_{i} and estimation parameters y_{i} and a_{i}
determine the ranks of the map units $s(r_{i*}, r_i)$
calculate the neighborhood radius $\sigma(t)$ for every epoch according to (3)
do for each epoch
    determine the ranks of the prototypes $k(v_j, w_i)$
    calculate both kernels using (1) and (2)
    update the prototypes w_i using (9)
    update the estimation values y_i by means of (17)
until the maximum number of epochs
determine the ranks of the prototypes $k(v_j, w_i)$
do for each epoch
    calculate both kernels using (1) and (2)
    modify the estimation vectors a_i as in (20)
until the maximum number of epochs
end

During the first loop, prototype vectors w_i and estimation values y_i are distributed along the input space to make the best possible fit to the data. In the second loop, gradient vectors a_i are estimated keeping constant the previously calculated parameters w_i and y_i.

The first loop includes an inner loop that calculates the distance from every prototype to each data vector and sorts them into the distance ranking $k(v_j, w_i)$, and after it the distance ranking is calculated again. This result is constant as the prototype vectors w_i are not modified in the second loop. In the second loop the gradient vectors a_i are approximated using the corresponding neighborhood radius for each epoch, starting with σ_{t0} again.

The computational cost of the algorithm itself is approximately linear with the number of data vectors and the map size, but it also has to be taken in care the complexity of the operations, specially the matrix multiplication.

V. Experimental testing

The quality of the proposed algorithm was compared with the well known SOM and NG [10] algorithms in their batch versions. The comparison was done using different values of the topology preservation constant γ, each one of them with a representative range of square output maps. Experiments were repeated 20 times with a previously generated set of values as initialization to ensure that all the differences are produced by the algorithms and not by external causes.

The first measure is the quantization error q_e defined as the mean distance from a data vector to its best matching unit according to

$$q_e = \frac{\sum_{j=1}^{N} \|v_j - w_{j*}\|}{N} \quad (21)$$

where N is the number of data vectors.

Topographic preservation is measured using the topographic error proposed in [11]. The topographic error is defined as the proportion of data whose two best matching units are not adjacent in the output map, mathematically defined as:

$$t_e = \frac{1}{N} \sum_{j=1}^{N} u(v_j) \quad (22)$$

where $u(v)$ is equal to 1 when the best and second best matching units are non-adjacent. Otherwise it is equal to zero. In this work 2-neighborhood measure is used, for rectangular maps it means each unit considers neighbors all the adjacent units, including the diagonal ones.

Finally, and most important, the estimation ability is measured by its root mean square error according to (23).

$$RMSE = \sqrt{\frac{1}{N} \sum_{j=1}^{N} \left(f(v_j) - \hat{f}(v_j) \right)^2} \quad (23)$$

All the non-numerical attributes and the missing values were removed before training. Data was normalized to zero mean and unitary standard deviation for every variable. For each data set a training data collection was created and the full collection was used to calculate the quality measures q_e, t_e and $RMSE$.

Estimation error for SOM was calculated using a normal training including all input and output variables together. The final value for the output variable is considered to be the estimation for any input vector in the Voronoi region.

For comparison purposes, new values will be calculated for errors. Quantization and estimation errors will be normalized with the expressions:

$$q'_e = \frac{q_e|_{SOM-NG}}{q_e|_{NG}} \quad (24)$$

$$RMSE' = \frac{RMSE|_{SOM-NG}}{RMSE|_{NG}} \quad (25)$$

where q'_e and $RMSE'$ are the relative quantization error and the relative root mean square error respectively and $error|_{algorithm}$ specifies which error is calculated and the algorithm that produces it. A value of 1 for any of them represents that the error is the same for the hybrid algorithm, i.e., the tested one, and NG, i.e., the reference algorithm.

Topographic comparative error is calculated using:

$$t'_e = t_e|_{SOM-NG} - t_e|_{SOM} \quad (26)$$

where t'_e is the comparative topographic error, that represents how worse the topographic preservation is in the hybrid algorithm compared to SOM results. Values of t'_e are withing the range $[-1, 1]$, where $t'_e > 0$ means the hybrid algorithm is not as ordered as SOM and $t'_e < 0$ means the opposite.

A. Concrete Passive Strength data set

This data set contains lab measurements of compressive strength for different concrete types used in [12] and it was obtained from UCI Machine Learning Repository [13]. This data set is used to measure the quality of the trained maps in non-projected data. In this experiment the dimensionality reduction and the map projection is not taken in care, but the influence of parameter γ over the quality measures is studied. Training data was created using a representative fraction of data vectors uniformly distributed along the whole data set.

Both q_e and t_e are represented in Figure 1 for three different map sizes. Values of γ lower than the unity have similar behaviour, with low quantization error and very high topographic error. On the opposite side, high values of γ have higher quantization errors and lower topographic ones. Intermediate values, like $\gamma = 10$, offer good equilibrium between both errors, being close as good in quantization as low values and also close to high values in topographic preservation.

The most interesting value is $\gamma = 10$ so it is going to be expanded to different map sizes. Results are shown in Figure 2.

This data set has a high dependence on initialization values because data points are one single cluster in the input space. This means that all the algorithms have a high deviation from their average value for such a big number of experiments. In Figure 2, both quantization and topographic errors have an acceptable balance as it was expected from that value of γ. It also can be seen that in average terms the estimation has a similar accuracy in comparison to the NG approach. Standard deviations were slightly lower in the proposed algorithm than in SOM for most of the calculated map sizes with $\gamma = 10$, so it can be considered robust enough.

In Figure 3, the estimation error is shown for three examples of map size and several values of γ. Low values of γ offer less accurate estimations and intermediate values reach good estimation capabilities while keeping a good trade-off with the other two quality measures.

B. Trigonometric function

To study the estimation capability of the proposed algorithm in topologically ordered data, a trigonometric function is going to be estimated with SOM, NG and hybrid SOM-NG algorithms. The selected function is:

$$z = f(x, y) = \sin(x + y) \cdot \cos(x \cdot y) \tag{27}$$

The training data set was created in $(x, y) \in [-4, 4] \times [-4, 4] \subset \mathbb{R}^2$ using a coarse mesh in the center of the region and a fine one close to the limits. The function is represented in Figure 4, just for illustrative purposes. The test data was

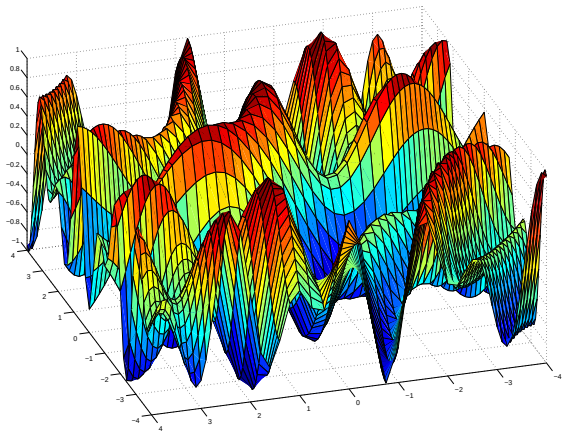

Figure 4. Trigonometric function to be estimated

uniformly arranged within the whole function domain, using a mesh with a step of 0.01 in both x and y.

Quantization error for this data set is different, as data is a discretized plane and SOM algorithms have better performance. If q_e is calculated for the training data NG still has the lowest quantization error, but for the test data set more ordered algorithms have better values, so SOM and high values of γ in the hybrid algorithm offer better quantization than NG.

Even in this circumstance, the hybrid algorithm has a good result, between the NG and SOM algorithms, as it can be seen in Figure 6. This special situation is good for the purpose of checking the algorithm, as it tends to get a good results because of the mixture of behaviours between SOM and NG.

In this data set, topographic preservation is similar to SOM, having values close to 0 in most of cases and bunch of outliers due to some different initializations that cause zonal disorders in one of the algorithms.

Estimation quality is good as it can be seen in the estimation error box-plot, looking for the quality of the best algorithm in estimation as it did in quantization. Estimation absolute results for three map sizes and different values of γ are shown in Fig 7. Best estimations are obtained for $\gamma = 0.2$, at the cost of having a very high topographic error, estimations obtained with values of γ greater than 5 are slightly worse, but keeping an acceptable topographic order.

VI. Conclusion

The hybrid SOM-NG algorithm has been tested with additional data sets to the previous work [8] and the results regarding data approximation and topographic preservation are confirmed.

The algorithm was improved with supervised learning using linear approximation for every Voronoi region. The value of γ has great influence over the final training.

Small values of γ, i.e., lower than 1, have a tendency towards NG behaviour, topographic preservation is very low and quantization is close to reach optimal results. The main

 ISBN: 978-1-61208-172-4

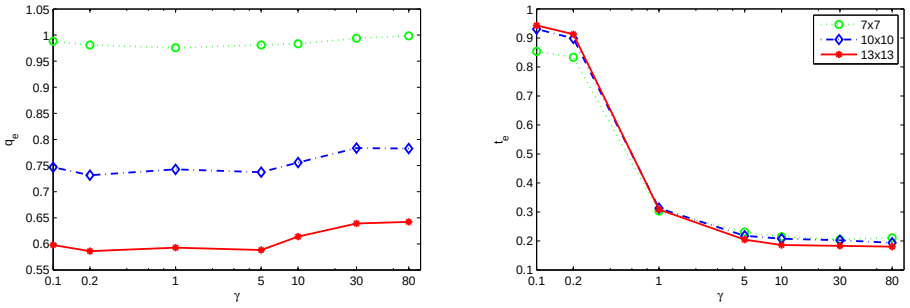

Figure 1. q_e and t_e versus γ for three different map sizes using Concrete data set

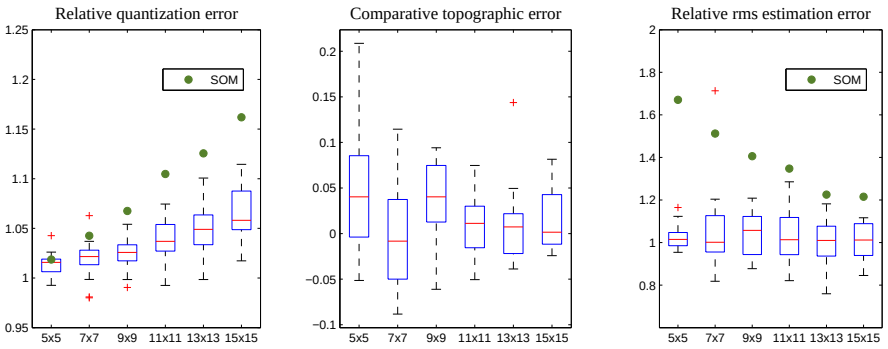

Figure 2. Concrete Passive Strength relative errors versus map size for $\gamma = 10$

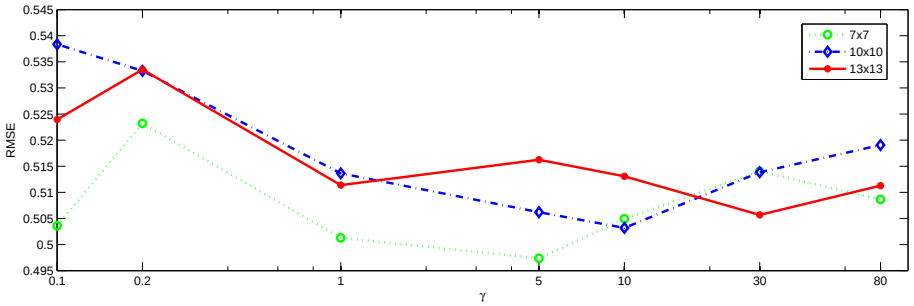

Figure 3. RMSE versus γ for three different map sizes for Concrete data set

drawback of using this learning condition is the lack of robustness. Low values of γ reduce the cooperation between neurons, as if the neighborhood radius had a smaller value, and initialization has a greater influence on the final results. If topographic preservation is not necessary, NG is a simpler and better choice.

The result is similar to a SOM training for high values of γ, having very small differences for any value over 80. Topographic preservation is very good, with some differences with SOM because of the ranking based kernel. The estimation done using the network as function approximator is satisfactory in comparison with the NG approach.

During the experiments the neighborhood function based on the modified ranking $s(r_{i*}, r_i)$ has different influence over the rest of the units than the usual squared distance. It is important to realize that the neighborhood radius for the squared euclidean distance measures *how far* does the *bmu* affect other units in the cooperative phase, while in the ranking it means *how many* units does it affect.

The most interesting feature of this algorithm is the use of intermediate γ values where a good trade-off solution is obtained. Relative measures, like the ones presented in Section V, demonstrate that close to optimum results can be achieved with a single algorithm.

REFERENCES

[1] T. Kohonen, *Self Organizing Maps*. Springer, 2001.

[2] J. L. Buessler, J. P. Urban, and J. Gresser, "Additive composition of supervised self-organizing maps," *Neural Processing Letters*, vol. 15, no. 1, pp. 9–20, 2002.

 ISBN: 978-1-61208-172-4

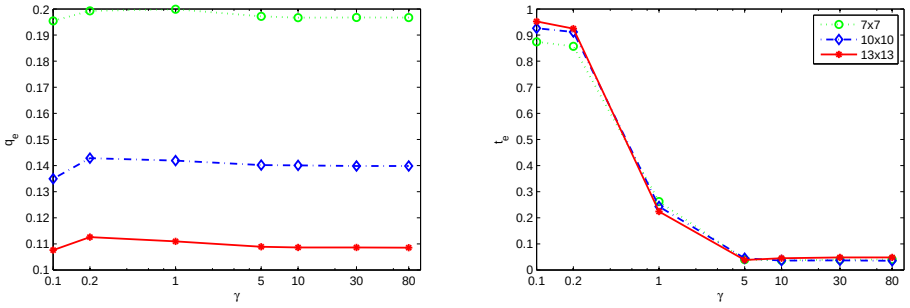

Figure 5. q_e and t_e versus γ for three different map sizes training with the trigonometric function

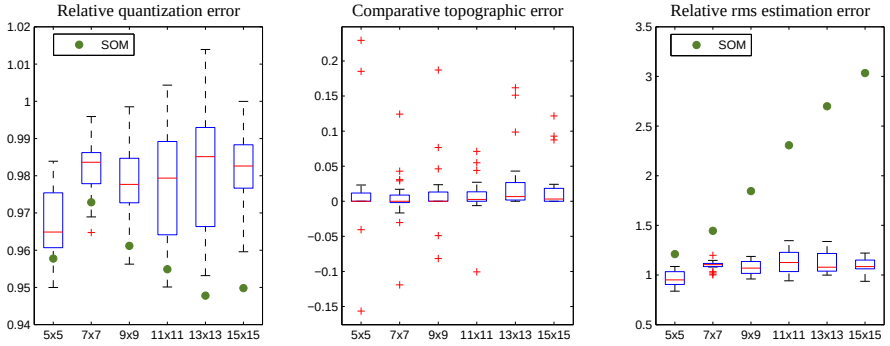

Figure 6. Trigonometric function relative errors versus map size with $\gamma = 10$

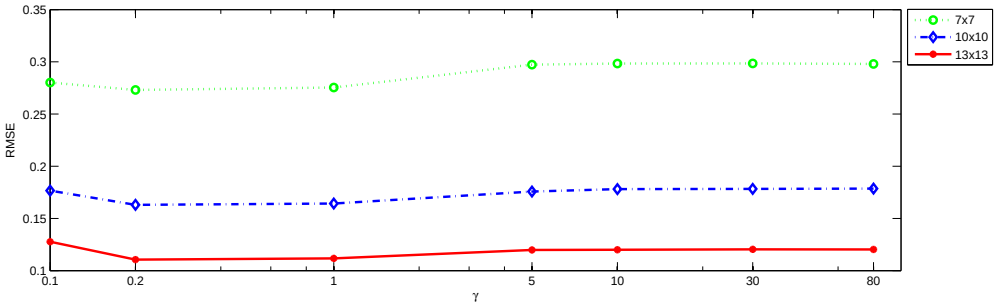

Figure 7. RMSE versus γ for three different map sizes approximating the trigonometric function

[3] J. Göppert and W. Rosenstiel, "The continuous interpolating self-organizing map," *Neural Processing Letters*, vol. 5, no. 3, pp. 185–192, 1997.

[4] G. J. Chappell and J. G. Taylor, "The temporal kohonen map," *Neural Networks*, vol. 6, no. 3, pp. 441 – 445, 1993.

[5] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski, "Recurrent SOM with local linear models in time series prediction," in *In 6th European Symposium on Artificial Neural Networks*, pp. 167–172, D-facto Publications, 1998.

[6] M. Varsta, J. Heikkonen, J. Lampinen, and J. Millán, "Temporal Kohonen map and the recurrent self-organizing map: analytical and experimental comparison," *Neural processing letters*, vol. 13, no. 3, pp. 237–251, 2001.

[7] T. Martinetz, S. Berkovich, and K. Schulten, "'neural-gas' network for vector quantization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, pp. 558 –569, July 1993.

[8] I. Machón-González, H. López-García, and J. Calvo-Rolle, "A hybrid batch SOM-NG algorithm," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 198–202, 2010.

[9] M. Cottrell, B. Hammer, A. Hasenfuß, and T. Villmann, "Batch neural gas," in *International Workshop on Self-Organizing Maps (WSOM 2005)*, pp. 275–282, 2005.

[10] M. Cottrell, B. Hammer, A. Hasenfufl, and T. Villmann, "Batch and median neural gas," *Neural Networks*, vol. 19, no. 6-7, pp. 762–771, 2006.

[11] K. Kiviluoto, "Topology preservation in self-organizing maps," in *IEEE International Conference on Neural Networks, 1996*, vol. 1, pp. 294 –299 vol.1, 1996.

[12] I.-C. Yeh, "Modeling of strength of high performance concrete using artificial neural networks," *Cement and Concrete Research*, vol. 28, no. 2, pp. 1797–1808, 1998.

[13] A. Frank and A. Asuncion, "UCI machine learning repository." http://archive.ics.uci.edu/ml, Sept. 2010.

 ISBN: 978-1-61208-172-4