Learning Topological Constraints in Self-Organizing Map

Guénaël Cabanes, Younès Bennani

Learning topological constraints in Self-Organizing Map

Guénaël Cabanes and Younès Bennani

LIPN-CNRS, UMR 7030, Université de Paris 13
99, Avenue J-B. Clément
93430 Villetaneuse, France
cabanes@lipn.univ-paris13.fr

Abstract. The Self-Organizing Map (SOM) is a popular algorithm to analyze the structure of a dataset. However, some topological constraints of the SOM are fixed before the learning and may not be relevant regarding to the data structure. In this paper we propose to improve the SOM performance with a new algorithm which learn the topological constraints of the map using data structure information. Experiments on artificial and real databases show that algorithm achieve better results than SOM. This is not the case with trivial topological constraint relaxation because of the high increase of the Topological error.

1 Introduction

The Self-Organizing Map (SOM) [1] is a popular algorithm to analyze the structure of a dataset. A SOM consist in a set of artificial neurons that represent the data structure. Neurons are connected with topological (or neighborhood) connections to form a two dimensional grid. Two connected neurons should represent the same type of data, two distant neurons (according to the grid) should represent different data. These properties are insured during the learning process by using neighborhood information as topological constraints, i.e. each neuron is activated by data that are represented by this neuron, but each neuron also responds in a less degree to data represented by its neighbors.

However, in the SOM algorithm, the topological information is fixed before the learning process and may not be relevant regarding to the data structure. To solve this problem, some works have been done in order to adapt the number of neurons during the learning process, using informations from the database to analyze [2]. Results have shown that the quality of the model is improved when the number of neurons is learned from the data.

Despite of these results, there is very few works that address the problem of learning the topological constraints from the data structure. However, at the end of the learning process, some "neighbors" neurons may not represent the same data [3, 4]. In this paper we propose to improve the SOM performance with a new algorithm able to learn the topology of the map using data structure information: the Data-Driven Relaxation – SOM algorithm (DDR-SOM). The main idea is to associate to each topological connection of the map a value indicate how well the two connected neurons represent the same type of data, then to use this values to reduce some topological constraint between neurons that represent different data. These constraint relaxation are expected to improve

the quality of the SOM, especially by reducing the quantization error of the map and increasing the number of neurons that really participate to the data representation.

The remainder of the paper is organized as follow. Section 2 presents the Self-Organizing Map algorithm. Section 3 describes the DDR-SOM algorithm based on SOM. Section 4 shows experimental validations and discuss the obtained results. Finally, a conclusion is given in Section 5.

2 Self-Organizing Maps

A SOM consists in a two dimensional map of neurons which are connected to n inputs according to a set of prototypes vectors and to their neighbors with topological connection [5, 1].. The training set is used to organize these maps under topological constraints of the input space. Thus, a mapping between the input space and the network space is constructed ; two close observations in the input space would activate two close neurons of the SOM. When an observation is recognized, the activation of an output neuron inhibits the activation of other neurons and reinforce itself. It is said that it follows the so called "Winner Takes All" rule. The winner neuron updates its prototype vector, making it more sensitive for later presentation of that type of input. To achieve a topological mapping, the neighbors of the winner neuron can adjust their prototype vector towards the input vector as well, but at a lesser degree, depending on how far away they are from the winner.

The connectionist learning is often presented as a minimization of a cost function. In most case, it will be carried out by the minimization of the distance between the input samples and the map prototypes, weighted by the neighborhood function K_{ij}. The cost function to be minimized is defined by:

$$\tilde{R}(w) = \frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{M} K_{iu^*(x^{(k)})} \parallel x^{(k)} - w_i \parallel^2$$

N represents the number of data in the database, M the number of neurons in the map, $u^*(x^{(k)})$ is the neuron having the closest weight vector to the input data $x^{(k)}$. The relative importance of a neuron i compared to a neuron j is weighted by the value of the kernel function K_{ij} which can be defined as:

$$K_{ij} = \frac{1}{\lambda(t)} \times e^{-\frac{d_M^2(i,j)}{\lambda^2(t)}}$$

Where $\lambda(t)$ is the temperature function modeling the topological neighborhood extent. $d_M(i,j)$ is the Manhattan distance defined between two neurons i and j on the map grid, i.e. the minimal number of topological connection between i and j.

3 Data-Driven Relaxation in Self-Organizing Map

3.1 Principle

In the DDR-SOM algorithm, we propose to associate each neighborhood connection a real value v which indicates the relevance of the connected neurons. Given the organization constraint of the SOM, both closest neurons of each data must be connected by

a neighborhood connection. A pair of neighbor neurons that are together a good representative of a set of data should be strongly connected, whereas a pair of neighbor neurons that don't represent the same type of data should be weakly connected. Each neighborhood connection is then associated to a value varying from 1 (strong connection) to 2 (week connection) using a logistic function depending on the number of data well represented by both the two connected neurons.

These values are used to estimate a weighted Manhattan distance $d_{WM}(i,j)$ between two neurons i and j. This distance is the minimal number of connections between i and j weighted by the value associated to each connection (see fig. 1 for an example).

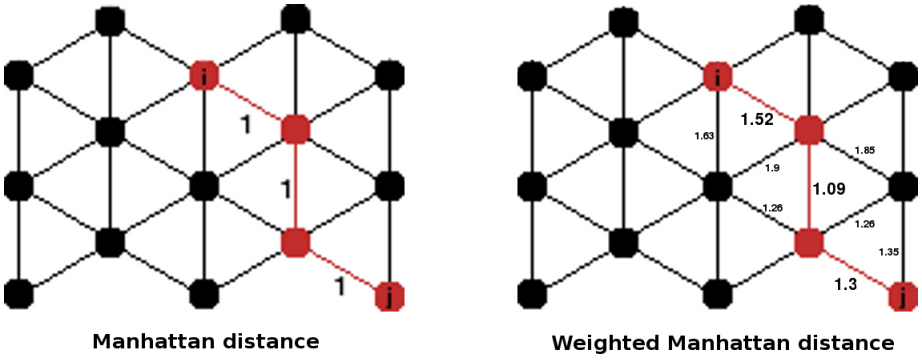

Fig. 1. Comparison of Manhattan distance and Weighted Manhattan distance between neurons i and j for hexagonal topology. Here $d_M(i,j) = 3$, it's the minimal number of topological connection between i and j, whereas $d_{WM}(i,j) = 1.52 + 1.09 + 1.3 = 3.91$, it's the minimal path between i and j according to the connection values v.

For that purpose we use Johnson's algorithm [6] to find:

- the shortest paths along neighborhood connection between all pair of neurons.
- the length of this path according to the values v associated to each connection.

In this way the distance between two neurons is expected to reflect the "true" neighborhood of these neurons.

Connection values and distances between neurons are updated during the learning of the map. Thus, the final quality of the SOM should be improved.

3.2 Algorithm

The DDR-SOM algorithm proceeds in tree steps :

1. **Initialization step** :
   - Define the topology of the SOM.
   - Initialize all prototypes vectors of the map w.
   - Initialize all neighborhood connections values v to zero.

2. **Competition step** :
   - Find the first BMU u^* and the second BMU u^{**} for each input data $x^{(k)}$:

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

   and

$$u^{**}(x^{(k)}) = \underset{1 \leq i \leq M,\ i \neq u^*}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

   - Update the neighborhood connections values v according to the following rule:

$$v_{i,j} = \frac{1 + 2e^{\frac{Nth - N(i,j)}{\sigma Nth}}}{1 + e^{\frac{Nth - N(i,j)}{\sigma Nth}}}$$

   $v_{i,j}$ is a logistic function that grown from 1 (when i and j represent the same data) to 2 (when i and j represent different data). $N(i,j)$ is the number of data (x) that have i and j in $\{u^*(x), u^{**}(x)\}$. Nth is the theoretical value for $N(i,j)$ under homogeneous hypothesis, i.e. Nth is the mean of $N(i,j)$ over all neighborhood connection.

3. **Adaptation phase** :
   - Update the distance $d_{WM}(i,j)$ for all pair of neurons i and j according to the neighborhood connections values v.
   - Update the kernel function K according to the temperature $\lambda(t)$ and d_{WM}.
   - Update prototypes vectors w_i of each neuron i in order to minimize the cost function:

$$w_i = \frac{\sum_{k=1}^{N} K_{iu^*(x^{(k)})}.x^{(k)}}{\sum_{k=1}^{N} K_{iu^*(x^{(k)})}}$$

4. **Repeat steps 2 and 3** until $t = t_{max}$

4 Experimental results

4.1 Databases description

In order to test the validity of the new algorithm we used 10 artificial and real databases with different number of data and features.

Databases “Target”, “TwoDiamonds”, “Tetra” and “Hepta” come from Fundamental Clustering Problem Suite (FCPS) [7]. They are artificial data in low dimensional space with well known structure. They are often used as benchmark for clustering algorithms [3, 4, 7]. Databases “Housing”, “Harot”, “Iris” and “Wine” are well known databases in various dimensional spaces from the UCI repository [8]. Finally, “Cockroach” and “Chromato” are very noisy real databases from biological experiments.

These databases are expected to reflect the diversity of the modeling problems that are encountered by SOM’s users.

4.2 Estimation of the quality of the SOM

We use the following three usual quality indexes to evaluate the training performance of SOM-based algorithms:

Quantization Error Qe:
This measures the average distance between each data vector and its BMU [1]. The smaller is the value of Qe, the better is the algorithm.

$$Qe = \frac{1}{N} \sum_{k=1}^{N} \parallel x^{(k)} - w_{u^*(x^{(k)})} \parallel^2$$

Topographic Error Te:
Te describes how well the SOM preserves the topology of the studied data set [9]. It's the proportion of all data vectors for which first and second BMUs are not adjacent neurons (i.e. are not connected with a topological connection). A small value of Te is more desirable. Unlike the quantization error, it considers the structure of the map.

Neuron Utilization Ne:
Ne measures the percentage of neurons that are not BMU of any data in the database [10]. A good SOM should have a small Ne, i.e. all neurons must be used to represent the data.

In all following experiments, all indexes are normalized in order to compare efficiently results on different databases. To represent the gain or the loss in comparisons to SOM, each error index is divided to the value obtained with the SOM algorithm (the SOM's error is then alway equal to 1). For each experiments in this Section we used the SOM-Toolbox [11] package, all parameters of the SOM have been set to default values (in particular we use hexagonal grid as initial topology of the map).

4.3 Topological relaxation in SOM

Basically, the main principle of the new algorithm is to decrease the topological constraint of the SOM by increasing the distance between neurons. These modifications are data-driven to optimize the final quality of the SOM.

The first step in our experiment is to analyze how behave a SOM with a trivial relaxation of the topological constraint. We expect that smaller constraint leads to a better modeling (i.e. smaller Quantification and Neuron Utilization errors), but also leads to worst topological error, as the topological constraint's function is to reduce the Topological error of the SOM.

We calculate the Quantification, Neuron Utilization and Topological error for each databases from results of different versions of the SOM algorithm where each distance between two neurons is multiplied by a constant value (see Fig. 2). The higher is this value, the weaker is the topological constraints. For example, SOM(α) is similar to the SOM algorithm, but $d(i,j) = \alpha \times d_M(i,j)$. We tested different values for this constant from SOM(1), similar to SOM, to SOM(10), where neurons are almost independent (in that case the algorithm behavior is similar to a K-means algorithm).

As the gain in Ne and Qe is associated to a loss in Te, we propose to define a General error that reflect the trade-off between Ne, Qe and Te:

$$Ge = Te^2 \times Ne \times Qe$$

Ge is the product of two trade-off: Ne vs. Te and Qe vs. Te. The value of Ge is smaller when the gain in Ne and Qe is higher than the loss of Te in comparison to the SOM algorithm. Ge is bigger in the other case.

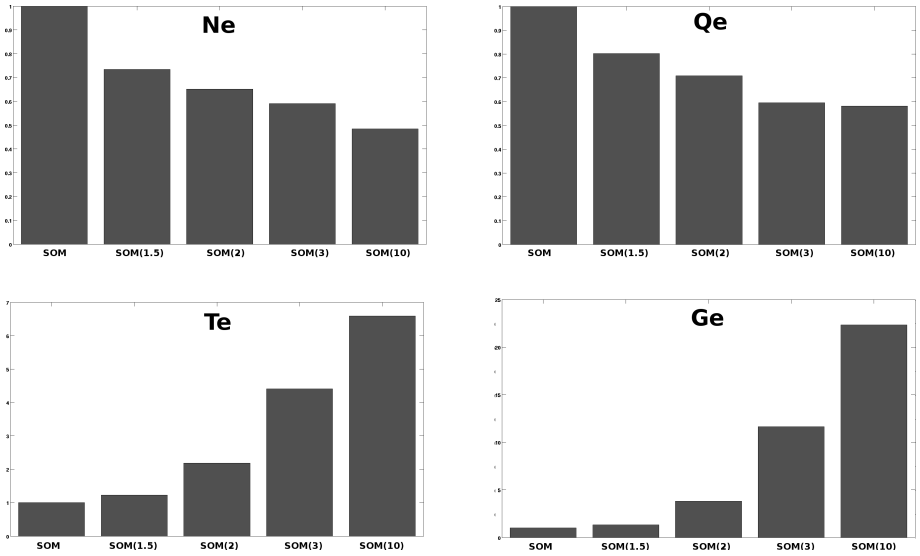

Fig. 2. Visualization of the mean value of Qe, Ne, Te and Ge over all databases.

Result are summarized in Fig. 2, here we show the mean value of Qe, Ne, Te and Ge over all databases for different values of α. As expected, Ne and Qe decrease when topological constraint decrease, whereas Te highly increase. But the value of Ge shows that under relaxation of the topological constraint the gain of Ne and Qe don't overcome the loss of Te. Thus, the best trade-off is to use the classical SOM algorithm!

Now the question is: can we use data to find some topological constraint relaxations which are a better trade-off than the SOM.

4.4 Evaluation of DDR-SOM

To evaluate the quality of DDR-SOM, we compare SOM and different versions of DDR-SOM with different values of the parameter σ. Fig. 3 shows means values of Qe, Ne, Te and Ge over all databases. Table 1 show the Ge value for each databases and each algorithms.

We can note the DDR-SOM mean topological constraint is similar to SOM(1.5). But as we can see, the DDR-SOM algorithm performance is much better than SOM(1.5) and

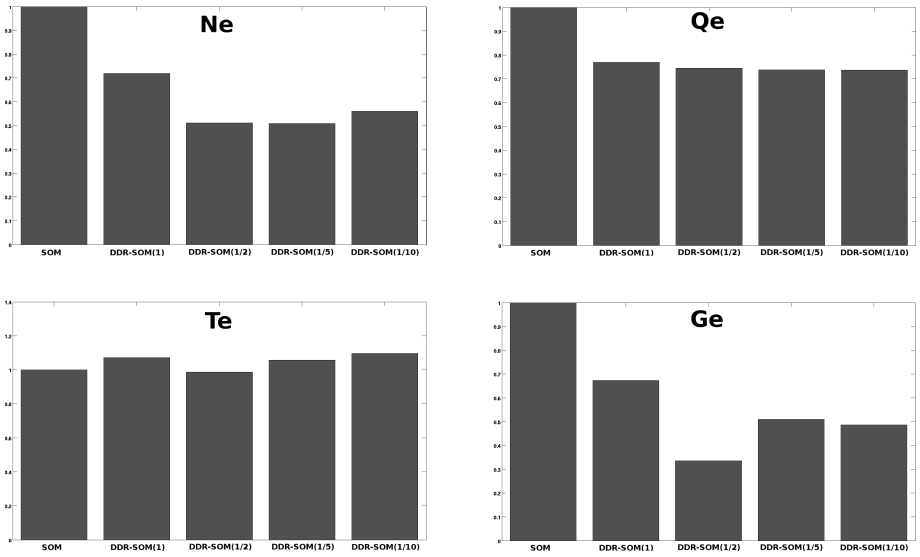

Fig. 3. Visualization of the mean value of Qe, Ne, Te and Ge over all databases for different values of σ in DDR-SOM.

SOM, i.e. the gain in Ne and Qe is higher than the loss in Te. Actually, with DDR-SOM Qe is similar that with SOM(2) and Ne is very low, similar that with SOM(10), whereas Te is similar that with the classical SOM algorithm.

Table 1. Ge value for each database and each algorithm.

	DDR(1)	DDR(1/2)	DDR(1/5)	DDR(1/10)
Target	0,57	0,39	0,87	0,89
TwoDiamonds	0,22	0,27	0,22	0,11
Hepta	1,82	0,62	0,97	0,57
Tetra	1,17	0,53	1,61	1,12
Iris	0,09	0,21	0,29	0,28
Harot	0,79	0,17	0,06	0,38
Housing	0,83	0,54	0,35	0,55
Wine	0,51	0,14	0,13	0,33
Cockroach	0,62	0,42	0,52	0,54
Chromato	0,12	0,08	0,09	0,09

These results lead to two remarks:

1. The DDR-SOM quality is better than SOM for all values of σ, although a value of $\sigma = 1/2$ seems to give better results for those databases.
2. The gain in Ge in comparison to SOM tend to be higher for databases in high dimensional space (e.g. “Chromato”, “Wine”, etc . . .).

5 Conclusion

In this paper we propose a new algorithm adapted from SOM, in order to improve the quality of the model, using a data-driven relaxation of the topological constraints. We defined a Global error that represent the trade-off between Topological error, Quantification error and Neural Utilization error.

Experiments on artificial and real databases show that DDR-SOM model achieve better results than the SOM algorithm. We also showed that this improvement is not obtained with trivial topological constraint relaxation because of the high increase of the Topological error. Data-driven relaxation seems to be a good solution to improve the $NeQe/Te$ trade-off of the SOM

6 Acknowledgments

This work was supported in part by the $CADI$ project (N^o ANR-07 TLOG 003) financed by the ANR (Agence Nationale de la Recherche).

References

1. Kohonen, T.: Self-Organizing Maps. Springer-Verlag, Berlin (2001)
2. Fritzke, B.: Growing grid - a self-organizing network with constant neighborhood range and adaptation strength. Neural Processing Letters **2**(5) (1995) 9–13
3. Cabanes, G., Bennani, Y.: A local density-based simultaneous two-level algorithm for topographic clustering. In: Proceeding of the International Joint Conference on Neural Networks. (2008) 1176–1182
4. Matsushita, H., Nishio, Y.: Self-Organizing Map with False-Neighbor Degree between Neurons for Effective Self-Organization. IEICE Transactions on Fundamentals **E91-A**(6) (2008) 1463–1469
5. Kohonen, T.: Self-Organization and Associative Memory. Springer-Verlag, Berlin (1984)
6. Johnson, D.: Efficient algorithms for shortest paths in sparse networks. Journal of the ACM **24**(1) (1977) 1–13
7. Ultsch, A.: Clustering with SOM: U*C. In: Procedings of the Workshop on Self-Organizing Maps, paris, france (2005) 75–82
8. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
9. Kiviluoto, K.: Topology Preservation in Self-Organizing Maps. International Conference on Neural Networks (1996) 294–299
10. Cheung, Y., Law, L.: Rival-Model Penalized Self-Organizing Map. IEEE Trans. Neural Networks **18**(1) (2007) 289–295
11. Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J.: Self-Organizing Map in Matlab: the SOM Toolbox. Proceedings of the Matlab DSP Conference (1999) 35–40