Model collisions in the dissimilarity SOM

Fabrice Rossi

Projet AxIS, INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105
78153 Le Chesnay Cedex - France

Abstract. We investigate in this paper the problem of model collisions in the Dissimilarity Self Organizing Map (SOM). This extension of the SOM to dissimilarity data suffers from constraints imposed on the model representation, that lead to some strong map folding: several units share a common prototype. We propose in this paper an efficient way to address this problem via a branch and bound approach.

1 Introduction

The Dissimilarity Self Organizing Map (DSOM, also refered to as the Median SOM) is one of the adaptation of Kohonen's SOM, proposed in [6], to dissimilarity data, i.e., to data that are only described by the results of pairwise comparisons of all observations via a dissimilarity measure.

While the DSOM and some of its variants have been applied successfully to real world problems (see e.g., [7, 11]), the quality of obtained maps is sometimes impaired by a type of strong folding: several units share the same model. We propose in this article to modify the DSOM algorithm to forbid model collisions via a branch and bound approach. We first recall the DSOM algorithm in section 2 and then present the model collision problem together with the proposed solution in section 3. Section 4 show the improvements in topology preservation induced by our method on a real world dataset.

2 The DSOM algorithm

The DSOM algorithm is based on the batch version of the SOM. Let us consider N input data from an arbitrary input space $\mathcal{X}$, $\mathcal{D} = (\mathbf{x}_i)_{1 \leq i \leq N}$. We assume given all pairwise dissimilarities between the data, denoted $d(\mathbf{x}_i, \mathbf{x}_k)$. The dissimilarity is symmetric and positive (with $d(\mathbf{x}_i, \mathbf{x}_i) = 0$). As in the standard SOM, the DSOM maps input data from $\mathcal{X}$ to a low dimensional organized set of M units (or neurons) arranged via a prior structure. However, as we make no assumption on the structure of the data, we can no longer associate an arbitrary vector model to each unit. The main difference between the DSOM and the standard SOM lies in this limitation of the former. There are two main solutions to overcome this problem. One possibility is to completely avoid using reference models, as proposed in [5]. In this paper we focus on a simpler solution, proposed in [1, 6]: the reference model for each unit is chosen among the input data.

As in a standard DSOM, the prior structure of the units leads to the definition of a neighborhood function, h^l, such that $h^l(u, v)$ measures the influence of unit u on unit v (and *vice versa*). The DSOM is based on the batch paradigm: the

superscript l used in the neighborhood function corresponds to the epoch number in the batch training. For each epoch, the DSOM performs two operations. The first operation is the affectation of each input data to its best matching unit (BMU). The second operation is the model update. The elementary solution for the affectation is simply to minimize the dissimilarity between an input data and the models, i.e., to define the affectation function c^l (which maps at epoch l an input data index to a unit index) as follows

$$c^l(i) = \arg \min_{j \in \{1,\dots,M\}} d(\mathbf{x}_i, \mathbf{r}_j^{l-1}), \tag{1}$$

where $\mathbf{r}_j^{l-1}$ denotes the model of unit j at epoch $l-1$.

Model update is done through the principle of the generalized median exposed in [6], which is summarized in the following equation

$$\mathbf{r}_j^l = \arg \min_{\mathbf{r} \in \mathcal{D}} \sum_{i=1}^{N} h^l(c^l(i), j) d(\mathbf{x}_i, \mathbf{r}). \tag{2}$$

The new model for unit j, $\mathbf{r}_j^l$ is chosen such as to minimize the weighted sum of its dissimilarity to all the input data. Weight values reflect the prior structure of the map. In [6], variants of Equation 2 are considered: for instance, the search for the model $\mathbf{r}_j^l$ can be restricted to observations affected to units close to unit j in the map. We restrict ourselves to the presented version, but our analysis applies also to its variants.

It should be noted that a naive implementation of the optimization problem hidden in Equation 2 can lead to very high running times: a fast solution is proposed in [3] and is used as the implementation basis for the present work.

3 Model collisions

3.1 Statement of the problem

The fact that models are chosen in a discrete set has some negative consequences. Obviously, this restriction reduces the quality of the final quantization. However, the main problem is the effect of the restriction on the behavior of the algorithm.

At the beginning of the algorithm, neighborhood functions are very broad (i.e., $h^l(u, v)$ is very slowly decreasing when the distance between u and v increases in the prior structure), in order to favor global organization. As a consequence, the optimization problems of Equation 2 are quite similar for close units in the map. In conjunction with the discrete nature of the optimization space, this can lead to model collisions: several distinct units share the same model.

While this particular problem tends to reduce epoch after epoch (as neighborhood functions become sharper and sharper), another source of model collisions is permanent. Some dissimilarities (e.g., the Levenshtein distance for strings [9]) have a low resolution, in the sense that they take only a few discrete values on a given input set. This can lead to model collisions (and to ambiguity in the BMU

determination). In theory, model collisions must be avoided as they corresponds to a very strong type of map folding. Moreover, they impair convergence because of the uncertainty they introduce in the BMU determination.

3.2 Proposed solutions for BMU determination

Several solutions have been proposed to tackle the BMU determination problem induced by prototype collisions. The naive approach consists in choosing randomly the affectation unit among BMUs. In [7], Kohonen and Somervuo use a variable size neighborhood to calculate a generalized dissimilarity between an input data and a set of units. If the models of several units have the same minimal dissimilarity with an input data, the comparison is extended to the models of neighbor units (via the sum[1] of the dissimilarities between the input data and the corresponding models). The radius of the neighborhood is increased until there is only one winner.

In [4], the authors calculate the BMU as follows

$$c^l(i) = \arg \min_{j \in \{1,\dots,M\}} \sum_{k=1}^{N} h^l(j,k) d(\mathbf{x}_i, \mathbf{r}_k^{l-1}). \tag{3}$$

However, this criterion doesn't favor organization as efficiently as the one of [7]. It is therefore combined with a simple trick that completely prevents collision: models are chosen according to Equation 2 in increasing order of j; once a data point is used to be $\mathbf{r}_j^l$, it can't be chosen anymore for units $k > j$.

In both cases, the remaining ties between candidate BMUs are solved via a random selection.

3.3 Explicite preventions of the collision

We propose in this paper to study the interest of an optimal prevention of model collision. The theoretical solution consists in looking for a set of *distinct* models that minimizes the following combined criterion

$$\sum_{j \in \{1,\dots,M\}} \sum_{i=1}^{N} h^l(c^l(i), j) d(\mathbf{x}_i, \mathbf{r}_j^l). \tag{4}$$

However, this problem belongs to the class of combinatorial optimization problems and there is no efficient and exact algorithm to solve it (this was the motivation for the trick used in [4]). Nevertheless, branch and bound techniques [8] can be used to provide quickly a good approximation of the result.

In theory, the search space consists in all size M lists of distinct elements of $\mathcal{D}$ (there are $N!/(N-M)!$ such lists). However, Equation 4 is obtained by summing independent positive values (optimized independently in Equation 2):

$$\sum_{i=1}^{N} h^l(c^l(i), j) d(\mathbf{x}_i, \mathbf{r}). \tag{5}$$

[1] In our implementation, the sum is weighted via the neighborhood functions.

This allows one to narrow the search down to at most M^M lists: for each unit j, we calculate B_j, the set of M data points that give the least M values to Equation 5. The minimal value of Equation 4 is reached on a list from $\prod_{j=1}^{M} B_j$.

It should be noted that the computation of B_j does not introduce a huge penalty to the DSOM algorithm. As shown in [3], solving Equation 2 can be done in $O(NM)$. Maintaining a list of M best candidates increases this cost to $O(NM \log M)$. In practice, the real cost is far lower because of the early stopping strategy used in [3].

Let us denote $(p_{ij})_{1 \leq j \leq M}$ the M best candidates for unit j and $(s_{ij})_{1 \leq j \leq M}$ the corresponding values of Equation 5. An exhaustive search in $\prod_{j=1}^{M} B_j$ would still have a very high cost. We use branch and bound to avoid it. Branching is done at the unit level: the exploration consists in choosing a model p_{1j_1} for unit 1 in B_1, then a model p_{2j_2} for unit 2 in B_2, etc. The corresponding value of Equation 4 is $\sum_{i=1}^{M} s_{ij_i}$. As the s_{ij} are positive, $\sum_{i=1}^{k} s_{ij_i} \leq \sum_{i=1}^{k+1} s_{ij_i}$. This gives a natural way to cut a branch: when the accumulated sum $\sum_{i=1}^{k} s_{ij_i}$ becomes larger than the best value obtained so far, the exploration of the subtree $(p_{1j_1}, \ldots, p_{kj_k}, *, \ldots, *)$ is aborted.

To initialize the process with a good starting point, we use the trick proposed in [4] with an additional randomization: the first branch is explored in a greedy way, each time choosing the best prototype among the still available ones. However, the order of the units is randomly chosen at each epoch in order to avoid favoring one part of the map over the others.

The main drawback of branch and bound is that the worst case situation corresponds to a prohibitive exhaustive search. We therefore terminate the search after N branches. As the cost of evaluating one full branch is $O(M)$, this gives a total cost for one unit of $O(NM)$ which is lower than the time needed to collect the M best candidates for this unit. This limit could be safely increased to $N \log M$ or chosen freely by the user. In the experiments reported in the next section, the computation time is increased by approximately 70 %.

4 Experiments

To evaluate the proposed algorithm on real world data, we have chosen a simple benchmark: clustering of a small English word list. We use the SCOWL word lists [2]. The smallest list in this collection contains 4 946 very common English words. After removing plural and possessive forms, the word list reduces to 3 200 words. A stemming procedure [10] applied to this list reduces it to 2 220 stemmed "words". Due to space constraints, we report our results only for the shortest list and for a 10×10 hexagonal map ($M = 100$). Similar results were obtained for the large word list and for different size of the maps.

The quality of the obtained DSOM is calculated via Venna and Kaski's trustworthiness (M1) and continuity (M2) measures [12] as they quantify the topology preservation. Those measures are defined for each possible value of a parameter k, the number of nearest neighbor considered in the calculation. A value of M1 close to 1 indicates that observations affected to close units in the DSOM are

neighbors according to the dissimilarities. Symmetrically a value of M2 close to 1 indicates that neighbors according to the dissimilarities are affected to close units in the DSOM. We compare two algorithms via the mean over $[1, k_{max}]$ of percentage of improvement of the best algorithm over the other (for each measure). k_{max} is chosen as the expected number of data points contained in the neighborhood of radius two in the hexagonal map (as there are 19 units in such a neighborhood, including the reference unit, $k_{max} = 19 \times N/M$).

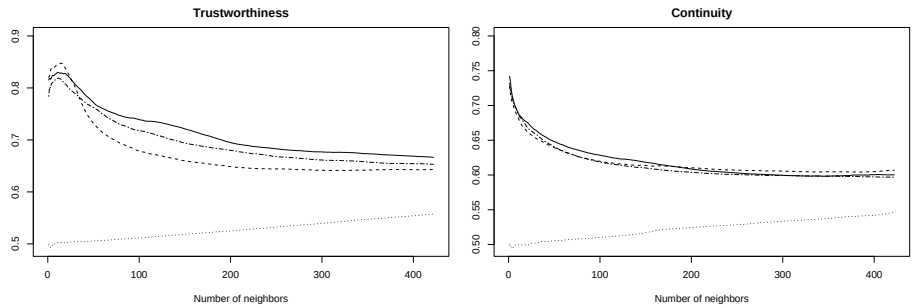

Fig. 1: Trustworthiness and continuity (solid line: KS-DSOM, dot-dash: ECGR-DSOM, dash: R-DSOM, dot: reference)

We compare our approach to the following algorithms: the DSOM with the tie breaking rule from [7] (KS-DSOM), the DSOM with the affectation criterion and prototype collision avoidance from [4] (ECGR-DSOM) and a simple DSOM based on Equation 1 with random tie breaking (R-DSOM). Results are summarized in Figure 1, which displays M1 and M2 for a large range of the neighbor number. The Figure includes reference curves corresponding to an untrained DSOM with random models. KS-DSOM appears to be the best overall solution, but is beaten by R-DSOM on M1 for a low number of neighbors and on M2 for a large number. ECGR-DSOM obtains intermediate results.

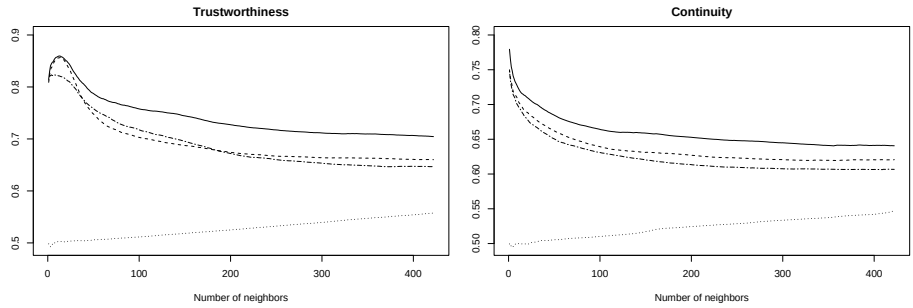

Fig. 2: Trustworthiness and continuity **without model collision** (solid line: KS-DSOM, dot-dash: ECGR-DSOM, dash: R-DSOM, dot: reference)

Those three methods are compared to variants in which model collisions are

prevented using the algorithm proposed in section 3.3. As shown by Figure 2 results are improved for KS-DSOM (4.1% gain for M1, 6.6% for M2) and for R-DSOM (3.2% and 2.9%), but remain stable for ECGR-DSOM. ECGR-DSOM appears therefore to be limited by ambiguity in BMU determination, in addition to model collisions. KS-DSOM becomes the best method overall especially in terms of broad organisation (large number of neighbors). KS-DSOM (and R-DSOM) can be used with the simple collision prevention method used in ECGR-DSOM, but this leads to lesser improvements (for instance 2.4% and 4.7% for the KS-DSOM), especially for large neighborhoods.

5 Conclusion

Overall, model collision prevention has a positive impact on topology preservation. The proposed solution, based on branch and bound, leads to the domination of the KS-DSOM over other solutions for all values of the neighborhood size. As the computational load increase remains reasonable and can be put under user control, model collisions should be always used in the DSOM.

References

[1] C. Ambroise and G. Govaert. Analyzing dissimilarity matrices via Kohonen maps. In *Proceedings of 5th Conference of the International Federation of Classification Societies (IFCS 1996)*, volume 2, pages 96–99, Kobe (Japan), March 1996.

[2] K. Atkinson. Spell checking oriented word lists (SCOWL). Available at URL `http://wordlist.sourceforge.net/`, August 2004. Revision 6.

[3] B. Conan-Guez, F. Rossi, and A. El Golli. Fast algorithm and implementation of dissimilarity self-organizing maps. *Neural Networks*, 19(6–7):855–863, July–August 2006.

[4] A. El Golli, B. Conan-Guez, and F. Rossi. A self organizing map for dissimilarity data. In D. Banks, L. House, F. R. McMorris, P. Arabie, and W. Gaul, editors, *Classification, Clustering, and Data Mining Applications (Proceedings of IFCS 2004)*, pages 61–68, Chicago, Illinois (USA), July 2004. IFCS, Springer.

[5] T. Graepel, M. Burger, and K. Obermayer. Self-organizing maps: Generalizations and new optimization techniques. *Neurocomputing*, 21:173–190, November 1998.

[6] T. Kohonen and P. J. Somervuo. Self-organizing maps of symbol strings. *Neurocomputing*, 21:19–30, 1998.

[7] T. Kohonen and P. J. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15(8):945–952, 2002.

[8] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[9] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.*, 6:707–710, 1966.

[10] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[11] F. Rossi, A. El Golli, and Y. Lechevallier. Usage guided clustering of web pages with the median self organizing map. In *Proceedings of XIIIth European Symposium on Artificial Neural Networks (ESANN 2005)*, pages 351–356, Bruges (Belgium), April 2005.

[12] J. Venna and S. Kaski. Neighborhood preservation in nonlinear projection methods: An experimental study. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2001)*, pages 485–491, Berlin, 2001. Springer.