Improving Performance of Self-Organising Maps with Distance Metric Learning Method.

Piotr Płoński[1] and Krzysztof Zaremba[1]

[1]Institute of Radioelectronics, Warsaw University of Technology,
Nowowiejska 15/19,00-665 Warsaw, Poland,
{pplonski,zaremba}@ire.pw.edu.pl

Abstract. Self-Organising Maps (SOM) are Artificial Neural Networks used in Pattern Recognition tasks. Their major advantage over other architectures is human readability of a model. However, they often gain poorer accuracy. Mostly used metric in SOM is the Euclidean distance, which is not the best approach to some problems. In this paper, we study an impact of the metric change on the SOM's performance in classification problems. In order to change the metric of the SOM we applied a distance metric learning method, so-called 'Large Margin Nearest Neighbour'. It computes the Mahalanobis matrix, which assures small distance between nearest neighbour points from the same class and separation of points belonging to different classes by large margin. Results are presented on several real data sets, containing for example recognition of written digits, spoken letters or faces.

Keywords: Self-Organising Maps, Distance Metric Learning, LMNN, Mahalanobis distance, Classification

1 Introduction

Some real-world problems do not have an exact algorithmic solution. Currently, there is a vast number of Artificial Intelligence(AI) methods which can be used to solve them. One of the branches of AI are Artificial Neural Networks. They are mathematical models inspired by biology. In 1982 T.Kohonen presented architecture called Self-Organising Maps (SOM) [10], which provides a method of feature mapping from multi-dimensional space to usually a two-dimensional grid of neurons in an unspervised way. This way of data analysis was proved as an efficient tool in many applications, both in academic and industrial solutions [11]. For example in character recognition tasks, image recognition tasks, face recognition [2], analysis of words[12], grouping of documents [9], visualisation [5], and even bioinformatics (for example phylogenetic tree reconstruction [4]).

There exists a huge number of methods for improving SOM's performance. Some of them concentrated on finding an optimal size of a network[1], faster learning [14] or applying different neighbourhood functions [8]. In this paper we investigate two additional improvements. The first one [6], [2] uses Mahalanobis

metric instead of the Euclidean one. The second improvement [13] shows how to use SOM in a supervised manner.

In our approach, contrary to [6], [2], [3], instead of computing the Mahalanobis matrix as an inverse of covariance matrix, it is learned in a way assuring the smallest distance between points from the same class and large margin separation of points from different classes. Several algorithms exist for distance metric learning (DML) [17], [7]. In this paper we use so-called Large Margin Nearest Neighbour (LMNN) method [16]. It introduces the distance metric learning problem as a convex optimization, which assures that the global minimum can be efficiently computed. First, we shortly describe SOM model used in supervised manner and LMNN method. Then we show how we combine these two approaches into our SOM+DML model. Finally we present results on real data sets.

2 Methods

Let's denote data set as $D = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}$, where $\boldsymbol{x_i}$ is an attribute vector of i-th sample, and $\boldsymbol{y_i}$ is a class vector, where $\boldsymbol{y_{ij}} = 0$ for $j \neq class_i$ and $\boldsymbol{y_{ij}} = 1$ for $j = class_i$, where $class_i$ is class number for i-th sample.

2.1 SOM model

In this paper, we used the SOM architecture in a supervised manner so-called 'Learning Associations by Self-Organisation' (LASSO), first described in [13]. The main difference between this SOM architecture and the original Kohonen's SOM architecture [10] is that during the learning phase the LASSO method takes into consideration class vector, additionally to attributes. Herein, we used two-dimensional grid of neurons. Each neuron is represented by a weight vector W_{pq}, consisting of a vector corresponding to attributes A_{pq}, and to a class C_{pq} ($W_{pq} = [A_{pq}; C_{pq}]$), where (p, q) are indexes of the neuron in the grid. In a learning phase all samples are shown to the network in one epoch. For each sample we search for a neuron which is closest to the i-th sample. The distance is computed by:

$$Dist_{train}(D_i, W_{pq}) = (\boldsymbol{x_i} - A_{pq})^T(\boldsymbol{x_i} - A_{pq}) + (\boldsymbol{y_i} - C_{pq})^T(\boldsymbol{y_i} - C_{pq}). \quad (1)$$

The neuron (p, q) with the smallest distance to i-th sample is called the Best Matching Unit (BMU), we note its indexes as (B_i, B_j). Once the BMU is found, the weight update step is executed. The weights of each neuron are updated with following formulas:

$$A_{pq} = A_{pq} + \eta(A_{pq} - \boldsymbol{x_i}), \quad (2)$$

$$C_{pq} = C_{pq} + \eta(C_{pq} - \boldsymbol{y_i}), \quad (3)$$

where η is a learning coefficient, consisting of a learning step size parameter μ and a neighbourhood function τ, so $\eta = \mu\tau$. Learning speed parameter is decreased

between consecutive epochs, so that network's ability to remember patterns is improved. It is described by $\mu = \mu_0 exp(-t\lambda)$, where μ_0 is a starting value of the learning speed, t is the current epoch and λ is responsible for regulating the speed of the decrease. Neighbourhood function controls changing the weights with respect to the distance to the BMU. It is noted as $\tau = exp(-\alpha S(B_i, B_j, p, q))$, where α describes the neighbourhood function width and $S(B_i, B_j, p, q)$ is the distance in the grid between the neuron and the BMU, computed by the following formula:

$$S(B_i, B_j, p, q) = (B_i - p)^2 + (B_j - q)^2. \tag{4}$$

We assumed a cost function as a sum of distances between samples and corresponding BMUs:

$$F = \sum_l Dist_{train}(D_l, W_{B_i,B_j}). \tag{5}$$

We train network till the cost function stops decreasing or a selected number of learning procedure iterations is exceed.

The exploitation phase is performed after the learning phase. New samples, which do not take part in the training, are shown to the network in order to designate their class. It should be noted that only the part with attributes is presented to the network. The BMU is found by computing a distance between an attribute input vector and an attribute part of the weights using the following formula:

$$Dist_{test}(D_i, W_{pq}) = (\boldsymbol{x_i} - A_{pq})^T(\boldsymbol{x_i} - A_{pq}). \tag{6}$$

For the tested sample, the designated class corresponds to position of maximum value in the part which code class information C_{pq} in BMU weights.

2.2 LMNN method

In many cases the mostly used metric is an Euclidean one. It often gives poor accuracy, because it takes all dimensions with equal contribution and assumes no correlations between the dimensions. Mahalanobis distance seems a better metric choice, because it is scale-invariant and takes into account input dimensions correlations. It is defined by:

$$Dist_M(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i} - \boldsymbol{x_j})^T M(\boldsymbol{x_i} - \boldsymbol{x_j}), \tag{7}$$

where M is usually an inverse of a covariance matrix. In case where M is an identity matrix, the distance (7) is equal to the Euclidean distance.

In this paper, we learned Mahalanobis matrix using the method described in [16]. Matrix coefficients are computed in a way assuring large margin separation of points from different classes and a small distance between the points of the same class. Before starting the matrix learning for each point, k nearest neighbours with the same class are found. They are called target neighbours, denoted by $\theta_{ij} = \{0, 1\}$, where $\theta_{i,j} = 1$ means that x_j is the target neighbour of x_i, and $\theta_{i,j} = 0$ means otherwise. With no prior knowledge, the Euclidean distance can be used to point the target neighbours. Target neighbours are unchanged during

the whole learning. Let's add a variable to indicate when the two samples have the same class, denoted as $y_{il} = \{0, 1\}$, where $y_{il} = 1$ when i-th and l-th samples are from the same class, zero when they are from different classes.

Finding an optimal matrix M can be expressed as a semidefinite programming (SDP) optimization problem with the following cost function (8) and constrains (9), (10), (11):

$$minimize \sum_{ij} \theta_{i,j} Dist_M(\boldsymbol{x_i}, \boldsymbol{x_j}) + c \sum_{i,j,l} \theta_{i,j}(1 - y_{il})\xi_{ijl}, \tag{8}$$

$$Dist_M(\boldsymbol{x_i}, \boldsymbol{x_l}) - Dist_M(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq 1 - \xi_{ijl}, \tag{9}$$

$$\xi_{ijl} \geq 0, \tag{10}$$

$$M \succeq 0. \tag{11}$$

The first term in the minimization function penalizes a large distance between the samples and their target neighbours. The second term penalizes a small distance between the samples from different classes - it is expressed as slack variables ξ_{ijl}. The c parameter balances the influence between these two terms. In this paper it is set to 0.5, which gives equal strength to each term. The constraint given in (11) requires the M matrix to be positive semidefinite - all its eigenvalues should be nonnegative. Semidefinite programming is a convex optimization problem, so a global minimum can be efficiently computed. SPD can be solved using general purpose solvers, however in our approach we used Matalb implementation code[1] described in [16], which is finely tuned to efficiently solve this kind of problems. Here most of slack variables are not used because samples are well separated.

2.3 SOM+DML model

We are interested in a such linear transformation of sample attributes that will assure that the Euclidean distance computed on the transformed attributes will be equal to the Mahalanobis distance computed on the original attributes. Mahalanobis matrix M can be written as $M = L^T L$, where L is the searched transformation. Lets denote $\boldsymbol{u_i}$ as the transformed attributes of i-th sample and $\boldsymbol{u_j}$ as the transformed attributes of j-th sample:

$$\boldsymbol{u_i} = L\boldsymbol{x_i}, \tag{12}$$

$$\boldsymbol{u_j} = L\boldsymbol{x_j}. \tag{13}$$

The distance between the transformed attributes in Euclidean distance should be equal to Mahalanobis distance between the original attributes:

$$Dist_M(\boldsymbol{x_i}, \boldsymbol{x_j}) = Dist_E(\boldsymbol{u_i}, \boldsymbol{u_j}). \tag{14}$$

[1] Matalb impementation of LMNN algorithm available from `http://www.cse.wustl.edu/~kilian/code/code.html`

We will now search for L. Now for matrix M we will find eigenvectors Φ and a matrix with eigenvalues on diagonal Λ:

$$M\Phi = \Phi\Lambda. \tag{15}$$

Matrix Λ can be expressed as:

$$\Phi^T M\Phi = \Lambda. \tag{16}$$

Since matrix M found by the LMNN algorithm is positive semidefine, diagonal elements in matrix Λ are nonnegative. Thus, we can write:

$$\Lambda^{-\frac{1}{2}}\Lambda\Lambda^{-\frac{1}{2}} = I. \tag{17}$$

Substituting (16) into (17), we obtain:

$$\Lambda^{-\frac{1}{2}}\Phi^T M\Phi\Lambda^{-\frac{1}{2}} = I. \tag{18}$$

Now we can note L as:

$$L = \Lambda^{-\frac{1}{2}}\Phi^T. \tag{19}$$

Using (19) we can write (18) as:

$$LML^T = I. \tag{20}$$

We see that $M = L^T L$ and hence we can write:

$$(\boldsymbol{x_i} - \boldsymbol{x_j})^T M(\boldsymbol{x_i} - \boldsymbol{x_j}) = (L\boldsymbol{x_i} - L\boldsymbol{x_j})^T(L\boldsymbol{x_i} - L\boldsymbol{x_j}) \tag{21}$$

From (21) we see that (14) is true. This transformation of input attributes is also known as *whitening transform*. It is worth mentioning that, in a transformation phase, only attributes $\boldsymbol{x}$ are transformed, the class part of input vector $\boldsymbol{y}$ is unchanged. Therefore, even though the data were pre-processed using the L transformation, we still can use the original SOM algorithm.

3 Results

Performance of the SOM+DML method was compared to the SOM model on six real data sets. As an accuracy measure we take the percentage of incorrect classifications. It is worth mentioning that getting the highest number of correct classifications is not the goal of this paper. Data sets are described in Table 1. Sets 'Wine', 'Ionosphere', 'Iris', 'Isolet', 'Digits' are sets from the 'UCI Machine Learing Repository' [2], set 'Faces' are from the 'The ORL Database of Faces'[3].

Now we briefly introduce the origin of the sets. Data sets 'Wine', 'Ionosphere', 'Iris' are classic benchmark sets, often used in testing newly developed classification algorithms. 'Isolet' data set represents a spoken letter recognition task.

[2] http://archive.ics.uci.edu/ml/
[3] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Its samples correspond to 26 letters of the alphabet. The original number of attributes (617) was projected by PCA to 100 principal components, which covers 93% of its variance. 'Digits' data set represents a handwritten digit recognition problem. The samples are from 10 classes obtained from 43 people. Original images were 32x32 bitmaps downsampled to 64 attributes by creators. The face recognition task is presented by data set 'Faces'. It contains images of faces obtained from 40 people (40 classes). For each person 10 images were taken in different times, varying lightening, changing facial expressions and details. The original data - 92x112 pixels images in 256 gray levels was projected by PCA to 50 leading components (83% of variance), the so-called egienfaces method[15]. Each data set, if not originally divided to train/test subsets, was randomly divided by us - 70% of data to training subset and 30% to testing subset.

	Wine	Ionosphere	Iris	Isolet	Digits	Faces
Train examples	126	246	105	6238	3823	280
Test examples	52	105	45	1559	1797	120
Attributes	13	34	4	100*	64	50*
Classes	3	2	3	26	10	40
Runs	100	100	100	20	20	20
Net size	4x4	6x6	6x6	20x20	20x20	20x20
k in LMNN	2	5	1	4	3	3

Table 1. Description of data sets used to test performance of the LASSO+DML method and parameters of networks. The number of nearest neighbour in the LMNN was set by cross validation. (*) In 'Isolet' and 'Faces' data sets, the number of attributes was reduced with PCA.

For each data set, we arbitrarily chose the network size (selecting optimal network size is not in the scope of this paper). The network size for the SOM and the SOM+DML models was equal. For all data sets, the following values of the learning parameters were used: $\mu_0 = 0.01$, $\lambda = 0.005$, $\alpha = 0.1$. For each data set a number of k target neighbours in the LMNN algorithm was selected using cross validation with ten times repetition. Fig.1 presents results of a selecting the target neighbours parameter (k) for all sets. Resulting k values are shown in Table 1. SOM weights were initialized with random numbers drawn from a normal distribution with mean 0 and standard deviation 0.5. Several runs were performed for each data set (see Table 1), so that local minimums were avoided. The final result is a mean over all runs. The SOM and the SOM+DML models were trained with identical number of iterations.

The comparison of results obtained by the SOM and the SOM+DML method is presented in Table 2. With one exception the SOM+DML method achieves lower error rates in both training and testing subsets. On the 'Iris' data set the LMNN algorithm seems to cause the overfitting effect. It is clearly visible in Fig.1 during search for the k parameter. The greatest improvement was achieved on

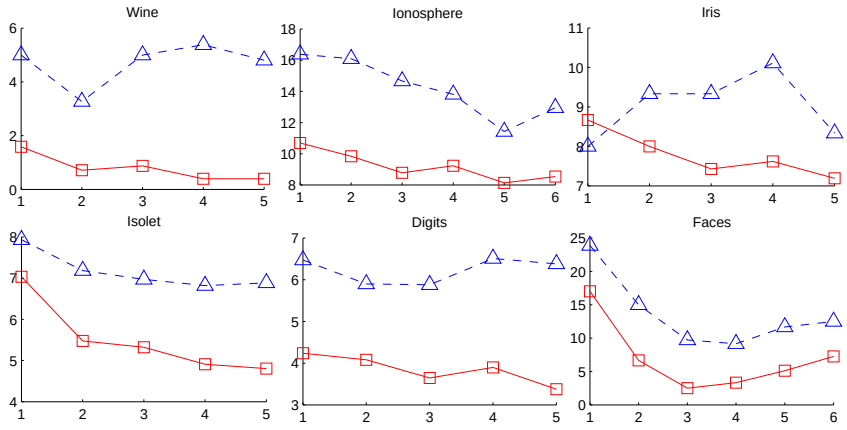

Fig. 1. Searching for optimal number of target neighbours in the LMNN method for all data set. On the x-axis there is k number of target neighbours, on the y-axis is percent of incorrect classification. Triangles with dashed lines represent test error and squares with solid lines illustrate a train error.

the 'Faces' set (20.87% over the SOM model). The comparison of example face pictures classified as belonging to the same class by the SOM method and the SOM+DML method is presented in the Fig.2. For the SOM+DML method, when using the 'Faces' set we observed a significant difference between the training error and the testing error. It is a small set, therefore the metric was well matched to the training set, giving small error. On 'Wine' and 'Isolet' data sets the improvement was 1.75% and 1.89% respectively. On the 'Digits' set there was a 2.68% improvement on the testing subset, which corresponds to roughly 50 digits. For the 'Ionosphere' data set the improvement was 6.67%. It is worth mentioning that for this set the largest k value was used.

	Wine		Ionosphere		Iris		Isolet		Digits		Faces	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
SOM	4.66	5.31	16.26	18.10	9.14	**7.78**	7.90	9.20	6.89	8.84	26.84	32.75
SOM+DML	**1.05**	**3.56**	**8.13**	**11.43**	**8.76**	8.44	**5.32**	**7.31**	**3.86**	**6.16**	**4.52**	**11.88**

Table 2. Percent of incorrect classification on training and testing subsets for the SOM and the SOM+DML method. Results are means over all runs.

4 Conclusions

A method of improving performance of the Self-Organising Maps in classification tasks was described. Linear transformation of data was performed be-

Fig. 2. Example of pictures from the 'Faces' data set classified as the same person by (a) the SOM model and by (b) the SOM+DML model.

fore SOM training phase. Matrix for the transformation has been obtained from the LMNN algorithm, which computes Mahalanobis matrix while assuring large margin separation between the points of different classes. We called our method SOM+DML. Testing of the method was demonstrated on several data sets, focused on recognition of: faces, handwritten digits and spoken letters. Test results confirm that the distance metric learning method improves the performance of the SOM network. Finding the optimal matrix for linear transformation plays a crucial role in obtaining improved results. Matlab implementation of the SOM+DML model is available from `http://home.elka.pw.edu.pl/~pplonski/som_dml`.

References

1. Alahakoon, D., Halgamuge, S. K. and Sirinivasan, B. Dynamic Self Organizing Maps With Controlled Growth for Knowledge Discovery. IEEE Transactions on Neural Networks, vol. 11, pp 601-614, (2000)
2. Aly, S., Tsuruta, N., Taniguchi, R.: Face Recognition under Varying Illumination Using Mahalanobis Self-organizing Map. Artificial Life and Robotics, vol.13, no. 1, pp 298-301 (2008)
3. Bobrowski, L., Topczewska, M.: Improving the K-NN Classification with the Euclidean Distance Through Linear Data Transformations. In Industrial Conference on Data Mining(2004), pp 23-32
4. Dopazo, J., Carazo, J.M., Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree. Journal of Molecular Evolution, vol.44(2), pp 226-33, (1997)
5. Duch, W., Naud, A.: On Global Self-Organizing Maps. In ESANN (1996), pp 91-96
6. Fessant, F., Aknin, P., Oukhellou, L., Midenet, S.: Comparison of Supervised Self-Organizing Maps Using Euclidian or Mahalanobis Distance in Classification Context. In IWANN(2001) pp 637-644
7. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In NIPS(2005), pp 513–520

8. Jiang, F., Berry, H., Schoenauer, M.: The impact of network topology on self-organizing maps. In GEC Summit(2009), pp 247-254
9. Kłopotek, M. A., Pachecki,T.: Create Self-organizing Maps of Documents in a Distributed System. Intelligent Information Systems, Siedlce, pp 315-320, (2010)
10. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics, vol.43 pp 59-69 (1982)
11. Kohonen, T., Oja, E., Simula, O., Visa, A., J. Kangas, Engineering applications of the self-organizing map. Proceedings of the IEEE(2002), vol. 84, no. 10, pp. 1358-1384.
12. Kohonen, T., Xing, H.: Contextually Self-Organized Maps of Chinese Words. In WSOM(2011) pp 16-29
13. Midenet, S., Grumbach, A.: Learning Associations by Self-Organization: The LASSO model. Neurocomputing vol.6(3) pp 343-361 (1994)
14. Rauber, A., Tomsich, P., Merkl, D.: parSOM: A Parallel Implementation of the Self-Organizing Map Exploiting Cache Effects: Making the SOM Fit for Interactive High-Performance Data Analysis. International Joint Conference on Neural Networks(2000), pp 177-182
15. Turk, M., Pentland, A.: Eigenfaces for recognition. Journal of Cognitive Neuroscience, vol.3, no.(1), pp 71–86, (1991)
16. Weinberger, K. Q., Blitzer, J., Saul, L. K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. In NIPS(2006), pp 1473-1480
17. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.J.: Distance Metric Learning with Application to Clustering with Side-Information. In NIPS(2002), pp 505-512