ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 21-26

Dynamic Neural Clustering

Katarzyna Mościńska
Department of Electronic Engineering
Silesian Technical University

Abstract. A novel algorithm for Neural Networks Clustering is proposed. The number of network nodes (i.e. the number of reference vectors) has not been fixed a priori, it may be increased as well as decreased during the training process. The algorithm is fast, robust against the outliers of the input vectors, and against the nonstationary data. The performance of the algorithm is demonstrated by experiments. The extraction of clusters by Dynamic Neural Clustering does not depend upon the initialization of the network.

1. Introduction

Clustering algorithms perform the partition of the input space according to the hidden structure of the k-dimensional input data. They attempt to group similar input vectors into clusters such that the elements of each cluster are more similar to each other than to any member from other clusters. Representation of input vectors by the cluster centroids is referred to as vector quantization.

Let us denote by $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$ an input vector, $i = 1 \ldots n$, and by $\boldsymbol{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jk})$, $j = 1 \ldots N$, a representation vector, i.e. the centroid of the cluster C_j. The overall quantization error (distortion) d is defined as:

$$d = \sum_{i=1}^{n} ||\boldsymbol{x}_i - \boldsymbol{w}_{win}|| \tag{1}$$

where $\boldsymbol{w}_{win}$ is the representation vector that has a minimum distance to the input vector:

$$||\boldsymbol{x}_i - \boldsymbol{w}_{win}|| = \min_{1<j<N} ||\boldsymbol{x}_i - \boldsymbol{w}_j|| \tag{2}$$

An N-level k-dimensional quantizer is said to be optimal, if it minimizes distortion for all quantizers that perform the partition of the input space into N subspaces, or clusters. Optimal quantizers cannot be determined as long as we do not know the probability density function $f(\boldsymbol{x})$ of the input vector. In general, $f(\boldsymbol{x})$ may be well estimated if the number of clusters increases. However, there is always a tradeoff between the overall distortion and the compression obtained by the quantization process.

A number of "classical" algorithms for clustering have been proposed [8], the most popular being the k-means or LBG algorithm [6]. Classical vector quantizers have been shown to converge to a local optimum, but global optimality is hard to achieve, whereas computational time is too long [7].

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 21-26

2. Neural Networks for Clustering

Artificial neural networks (ANN) possess many desirable features: they are robust, fault tolerant, and often can be efficiently applied when classical methods fail. One of the well examined areas of the ANN applications is clustering.

A large variety of ANN topologies as well as learning algorithms have been proposed for clustering; among the most popular is the Kohonen's Self-Organizing Feature Map (SOM) [1]. Kohonen has also initiated study of the Learning Vector Quantization algorithms (LVQ), which neglect topological structure of the input data but preserve the SOM's idea of the codebook design [1, 5]. In general, these algorithms suffer from the following drawbacks:

- they are not robust against the codebook vectors initialization,
- they do not have robustness against outliers in the input data,
- they cannot neatly adapt if the input data is nonstationary.

Recently some other algorithms for clustering have been proposed. Pal and Bezdek [5] derived modified learning rules to optimize an objective function whose goal is to produce proper clusters. This algorithm, however, requires much more computation than the Kohonen's-like approach. Pitas and Kotropoulos [3] proposed vector quantizer based on multivariate order statistic. Although this approach allows to overcome the previously mentioned drawbacks, it requires a lot of computational resources, especially memory. Choi and Park [4] proposed the algorithm which creates new network nodes if the best matching code vectors are much too far from the input patterns.

This paper presents the novel Dynamic Neural Clustering algorithm (DNC) in which some of the above mentioned concepts were exploited.

3. Dynamic Neural Clustering

The main idea of SCONN algorithm, developed by Choi and Park [4], is to start with a very few network nodes and to expand the network with time. Let us modify this approach so that the network might be expanded as well as shrunk during the generation of the representation vectors. Therefore we prevent representation vectors both redundant as well as not responding to any input patterns. If there are more network nodes activated by an input vector, they become labeled as redundant. All nodes in the network have predefined priority coefficients; the node with the highest priority among the redundant nodes becomes the winner whereas the other ones in the redundant nodes set are moved away from the winner. When the number of redundancy labels exceeds some predefined limit, the redundant nodes are removed from the network. At the same time nodes may be added to the network so as not to destroy previously established centroids.

The proposed algorithm can be summarized as follows:

1. Initialize number of nodes N_0 i.e. initial number of centroids along with their weights w_j, learning parameter α_0, redundancy identification factor r, antiredundancy spreading factor m, redundancy counters R_j and redundancy counter

limit R_{max}, choice counters C_j and choice counter limit C_{min}, node generation threshold l_0 and node generation factor q, iterations stop condition parameter ϵ_d. Perform the iterative calculations as defined below, with $t = 1, \ldots, t_{max}$ being the iteration counter.

2. At the beginning of each iteration adjust (decrease) the learning parameter α_t and node generation threshold l_t. For all the input vectors $\boldsymbol{x}_i$ presented randomly

   For all the network nodes wj

   Find the best responding node $\boldsymbol{w}_{win}$ according to the rule:

$$d_{win} = \|\boldsymbol{x}_i - \boldsymbol{w}_j\| = \min_{1<j<N_t} \|\boldsymbol{x}_i - \boldsymbol{w}_j\| \tag{3}$$

   If there are other nodes $\boldsymbol{w}_j$ responding sufficiently strong to the input vectors, i.e.

$$\|\boldsymbol{w}_j - \boldsymbol{x}_i\| < (1 + r) d_{win} \tag{4}$$

   then there exist redundant nodes in the network. In this case increase the redundancy counters R_j for the nodes involved and spread them away according to the following formula

$$\boldsymbol{w}_j = \boldsymbol{w}_{win} - m\,(\boldsymbol{x}_i - \boldsymbol{w}_{win}) \tag{5}$$

   with m being the adjustable redundancy spread factor.

   If for the best responding node $d_{win} > l_t$, create a new node using the following recipe:

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{win} + q\,(\boldsymbol{x}_i - \boldsymbol{w}_{win}) \tag{6}$$

   and increase the number of nodes N_t; otherwise, modify the weights of the winner node:

$$\boldsymbol{w}_{win} = \boldsymbol{w}_{win} + \alpha_t\,(\boldsymbol{x}_i - \boldsymbol{w}_{win}) \tag{7}$$

3. Remove nodes with excessive values of redundancy counters $R_j > R_{max}$; remove inactive (degenerated) nodes identified through too small values of choice counters $C_j < C_{min}$.

4. If $\frac{|d_{t-1} - d_t|}{d_{t-1}} \leq \epsilon_d$ break, else continue iterations by going to (2).

The proposed algorithm possesses the ability to find the hidden structure of the data, i.e. it extracts the number of clusters independent from the initial number of centroids.

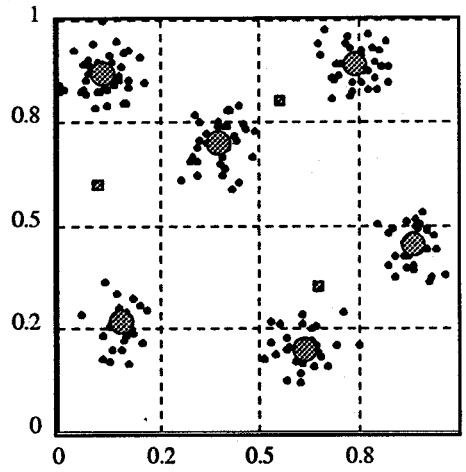

Fig. 1. Results of Experiment No 1, $d = 0.00243$.

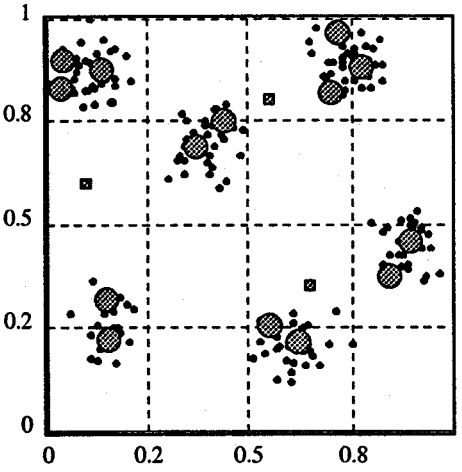

Fig. 2. Results of Experiment No 2, $d = 0.00152$.

4. Experimental Results

Clustering properties of the proposed algorithm can be illustrated by the following experiment. Let input vector $x_i = \{x_1, x_2\}$; $i = 1 \ldots 500$, form $K = 6$ clusters of the normally distributed variables, with mean values as defined by the set $\{0.15, 0.25\}$, $\{0.6, 0.2\}$, $\{0.9, 0.45\}$, $\{0.75, 0.9\}$, $\{0.1, 0.85\}$, $\{0.4, 0.7\}$; the standard deviation was equal to 0.05. We assumed that the starting number of network nodes N_0 was equal to 3, 6 and 12 for consecutive experiments, .i.e. it was too small, exact and too large, respectively, when compared with the number of clusters in the training data set. The nodes initial weights were chosen in various manner, close to the data cluster centres as well as far away from them; the initial nodes positions are presented in the figures with small boxes. Learning parameters were as follows: learning parameter $\alpha_0 = 1.0$, redundancy identification factor $r = 0.1$, antiredundancy spreading factor $m = 0.7$, node generation threshold $l_0 = 0.3$ and node generation factor $q = 1.0$, iterations stop condition parameter $\epsilon_d = 0.001$.

In general, the proposed algorithm proved to be perfectly well suited to the neural networks clustering tasks when it is difficult to predict in advance the number of clusters. In all experiments the proposed algorithm could easily and quickly adjust the number of network nodes to 6, i.e. equal to the number of data clusters in the training data set, independent from the initial number of network nodes and its topology. Thus the proposed algorithm was found to be effective in adding the needed nodes as well as in removing the excessive ones, thus enabling the necessary trade-off between the clustering accuracy, convergence speed and calculation speed. Fig.1 presents the localization of clusters as well as network nodes at the beginning (small boxes) and at the end of the training process (medium disks), with initial number of network nodes having been equal to 3. As one can see there were 3 additional nodes added, the mean overall distorsion was $d = 0.00243$. As mentioned above, almost identical final results have been obtained for different numbers and configurations of initial nodes. Fig.2

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 21-26

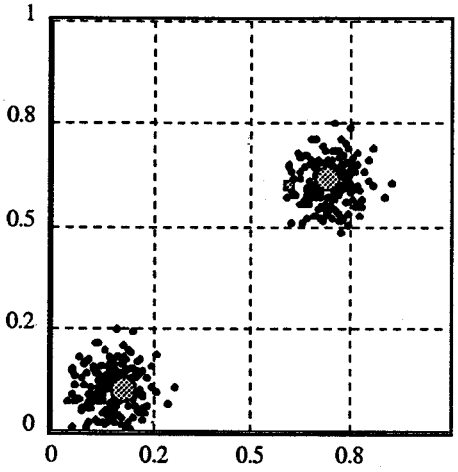

Fig. 3. Results of Experiment No 3, $d = 0.00267$.

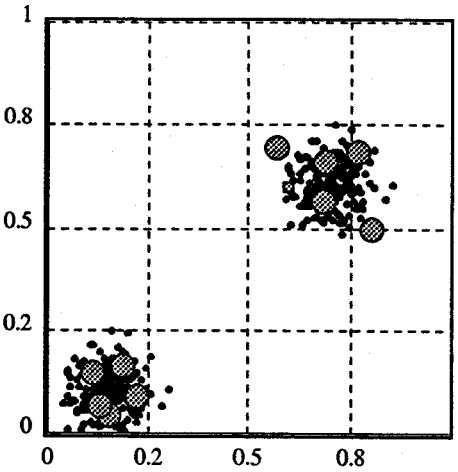

Fig. 4. Results of Experiment No 4, $d = 0.00103$.

presents the results of similar experiment with node generation threshold l_0 this time equal to 0.1 meaning that one is more willing to allow for the generation of new nodes. The results presented are perfectly coherent with the above prediction: there are 14 nodes for 6 data clusters--the price paid for much better value of the overall distorsion which was equal to 0.00152. Also for this set of experiments the proposed algorithm has been shown to adjust quickly the number of network nodes to the clustering accuracy demands irrespective from the initial network nodes topology.

In the next 2 experiments the first 200 samples of the input vector x_i were random variables normally distributed around the $\{0.15, 0.1\}$ mean value with 0.05 standard deviation, whereas the next 200 data were normally distributed around the $\{0.7, 0.6\}$ mean value with the same standard deviation. Fig.3 presents the results of the proposed algorithm run with the values of learning parameters the same as for the Fig.1 corresponding experiment, i.e. among others with the value of node generation threshold $l_0 = 0.3$; l_0 was changed to 0.1 for the experiment visualized in Fig.4. Similarly to the first two experiments described above, there were different numbers and weights of initial network nodes chosen, with practically negligible influence on the final network structure and parameters. For $l_0 = 0.3$ the resulting network included 2 nodes for 2 data clusters with overall distorsion $d = 0.00267$, whereas for $l_0 = 0.01$ the network nodes number was expanded to 10 with the overall distorsion correspondingly decreased to the value $d = 0.00103$. In both cases the network has easily adapted to the presented nonstationary data set.

5. Conclusions

A new algorithm for neural network clustering has been presented. The proposed algorithm is fast, independent from the initialization of the network nodes and is robust against the nonstationary data. The algorithm can be applied for a number of tasks,

including image data compression.

The proposed algorithm can be modified in various manners; for instance, the way of creating the new network nodes and the criterion for removing redundant nodes may be the subject of the further study. The idea of simultaneous expanding and shrinking of the network may be also implemented into clustering algorithms with different algorithms for moving centroids.

References

[1] Kohonen T. *Self-Organization and Associative Memory.* Berlin, Springer-Verlag, 1989, 3rd ed.

[2] Kohonen T. *Learning Vector Quantization.* Neural Networks, Vol.1, suppl.1, pp.303-310.

[3] Pitas I. *Kotropoulos C.A.* A Class of Order Statistic Learning Vector Quantization, Proc. of IEEE International Symposium on Circuits and Systems, London 1994, pp. 387-390.

[4] Choi D., Park S. *Self-Creating and Organizing Neural Networks.* IEEE Transactions on Neural Networks, Vol.5, No 4, July 1994, pp.561-575.

[5] Pal N.R., Bezdek J.C.. *Generalized Clustering Networks and Kohonen's Self-Organizing Scheme.* IEEE Transactions on Neural Networks, Vol.4, No 4, July 1993, pp. 549-557.

[6] Linde Y., Buzo A., Gray R.M. *An Algorithm for Vector Quantizer Design.* IEEE Trans. on Communications, Vol. 28, No 1., Jan 1980, pp.84-95.

[7] Makhoul J., Roucos S., Gish H. *Vector Quantization in Speech Coding.* Proceedings of the IEEE, Vol. 73, No 11, Nov.1985, pp. 1551-1588.

[8] Hartigan J. *Clustering Algorithms.* New York, Wiley, 1975.