International Journal of Fuzzy Logic and Intelligent Systems, vol. 11, no. 2, June 2011, pp. 84−88
DOI : 10.5391/IJFIS.2011.11.2.084

Fuzzy Learning Vector Quantization based on Fuzzy k-Nearest Neighbor Prototypes

Seok-Beom Roh, Ji-Won Jeong and Tae-Chon Ahn*

Department of Electronic & Control Engineering Wonkwang University
***Research Institute of Engineering Technology Development Wonkwang University**

Abstract

In this paper, a new competition strategy for learning vector quantization is proposed. The simple competitive strategy used for learning vector quantization moves the winning prototype which is the closest to the newly given data pattern. We propose a new learning strategy based on k-nearest neighbor prototypes as the winning prototypes. The selection of several prototypes as the winning prototypes guarantees that the updating process occurs more frequently. The design is illustrated with the aid of numeric examples that provide a detailed insight into the performance of the proposed learning strategy.

Key Words : learning vector quantization, competition strategy, fuzzy k-nearest neighbor approach, classification

1. Introduction

Vector quantization is a very important technique for image compression [1, 2]. Recent developments in neural network architecture resulted in learning vector quantization (LVQ) algorithms [3-7].

Learning vector quantization is the name used for unsupervised learning algorithms associated with competitive neural networks whose weight vectors represent the code vectors. Based on 'winner-take-all' competing strategy, T. Kohonen's [3, 4] gave a learning vector quantization algorithm, named self-organizing feature map (SOFM) whose performance was greatly affected by the initial prototypes.

The simple competitive learning strategy limits the learning opportunity of the prototypes which are almost near to the given data pattern though they are not the nearest prototype to the given data pattern.

In this paper, to lessen the limitation of learning opportunity of the prototypes, we modify the selection method to choose the winning prototype. In simple competitive strategy, the nearest prototype to the given data pattern is selected as the winning prototype.

In case of the proposed competitive strategy, the k-nearest neighbor prototypes [8-10] are selected as the winners of the competition. Each prototype is assigned its own priority (i.e. weighting value). Each prototype participates the learning or updating process according to its priority.

To evaluate the proposed model for classification problem, we made several experiments using 2-dimensional synthetic datasets and various machine learning datasets.

Manuscript received Apr. 11, 2011; revised Jun. 7, 2011.
This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2006-353-D00021), in which main calculations were performed by using the supercomputing resource of the Korea Institute of Science and Technology Information (KISTI).

2. Learning Vector Quantization Algorithms

2.1 Selecting a Template (Heading 2)

Consider the set of samples $\boldsymbol{X}$ from an n-dimensional Euclidean space $\boldsymbol{R}^n$ and let function $f(\boldsymbol{x})$ be the probability density function of $\boldsymbol{x} \in \boldsymbol{X} \subset \boldsymbol{R}^n$ and let $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_c\} \subset \boldsymbol{R}^n$ be the current codebook or clustering center, where c is clustering number. Define the loss function $L(\boldsymbol{x}, \boldsymbol{V})$ as:

$$L(\boldsymbol{x}, V) = \sum_{i=1}^{c} u_i \cdot \left\| \boldsymbol{x} - \boldsymbol{v}_i \right\|^2 \tag{1}$$

$$. \; u_i(\boldsymbol{x}) = \begin{cases} 1, & if \; i = \min\limits_j \left\| \boldsymbol{x} - \boldsymbol{v}_i \right\|^2 \\ 0, & otherwise \end{cases} . \tag{2}$$

where $u_i = u_i(\boldsymbol{x}), i = 1, 2, \cdots, c$ is a set of weight and can also be interpreted as membership function which regulate the competition between the prototypes $\boldsymbol{v}_i$ for the input $\boldsymbol{x}$. For input $\boldsymbol{x}$, $\boldsymbol{v}_i$ satisfying $\left\| \boldsymbol{x} - \boldsymbol{v}_i \right\|^2 \le \left\| \boldsymbol{x} - \boldsymbol{v}_j \right\|^2$, $(i \neq \forall j)$ is the winning prototype.

The expectation of loss function $L(\boldsymbol{x}, \boldsymbol{V})$ is $L(\boldsymbol{V})$ defined as

$$L(\boldsymbol{V}) = \int_{\boldsymbol{R}^n} L(\boldsymbol{x}, \boldsymbol{V}) f(\boldsymbol{x}) d\boldsymbol{x} \tag{3}$$

LVQ algorithms are based on minimization of the expectation function $L(\boldsymbol{V})$ to update the clustering center V. Minimization of (3) using gradient descent is difficult because the probability density function $f(\boldsymbol{x})$ is not known explicitly. Pal et al. [5] suggested the use of the gradient of the instantaneous loss function (1) to sequentially update the prototypes $\boldsymbol{V}$ with respect to the input vectors $\boldsymbol{x} \in \boldsymbol{X}$. This

approach is frequently used in the development of learning algorithm.

The process of the LVQ algorithms can be summarized as follows.

1. Initialization: select c, fix learning rate h_0, the iteration number T and the initial codebook $\boldsymbol{V}=\{\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_c\}$ let $t=0$
2. Calculate learning rate: $\eta=\eta_0(1-t/T)$;
3. For each input vector x and current $\boldsymbol{V}=\{\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_c\}$

3-1. Find the winning prototype $\boldsymbol{v}_i$ satisfying $\|\boldsymbol{x}-\boldsymbol{v}_i\|^2 \le \|\boldsymbol{x}-\boldsymbol{v}_j\|^2, i\neq\forall j$

3-2. Calculate the updating coefficients ω_i by the gradient of $L(\boldsymbol{x},\boldsymbol{V})$

3-3. Update $\boldsymbol{v}_i$ as $\boldsymbol{v}_i=\boldsymbol{v}_i+\eta\omega_i(\boldsymbol{x}-\boldsymbol{v}_i)$

4. If $t<T, t=t+1$, go to step 2;
5. Output codebook $\boldsymbol{V}=\{\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_c\}$ First, confirm that you have the correct template for your paper size. This template has been tailored for output on the US-letter paper size.

3. Fuzzy K-Nearest Neighbors Approach

The key assumption behind the *k*NN approach is that the closest instances to the query point have similar target values one could predict for the query. The neighborhood is defined by the k-nearest neighbors of query point $\boldsymbol{q}$ where the closeness is articulated through some distance function. Quite commonly, the distance can be treated as the Euclidean one or its generalized weighted version. The set of indexes for the selected neighborhood comes in the form

$$A=\{j: \boldsymbol{x}_j \text{ is one of the k nearest instances to query q}\} \quad (4)$$

After defining the neighborhood of query point $\boldsymbol{q}$, we determine the target value at this query. The expressions commonly encountered in classification problems read as follows

$$\hat{t}(\boldsymbol{q})=\arg\max_c \sum_{j=1}^{k} f\left(t\left(\boldsymbol{x}_{L\{j\}}\right),c\right) \quad (5\text{-}1)$$

$$f\left(t\left(\boldsymbol{x}_{L\{j\}}\right),c\right)=\begin{cases}1, & if\ t\left(\boldsymbol{x}_{L\{j\}}\right)=c\\ 0, & if\ t\left(\boldsymbol{x}_{L\{j\}}\right)\neq c\end{cases} \quad (5\text{-}2)$$

Here, $t(\boldsymbol{x})$ denotes the known target value of pattern $\boldsymbol{x}$, while $\hat{t}(\boldsymbol{q})$ stands for the estimated target value at the query point $\boldsymbol{q}$.

In the case of function approximation, the fuzzy *k*-nearest neighbor predicts the target value by using a weighted average applied to points located in a varying neighborhood [8]. After obtaining the neighbors of the given query point $\boldsymbol{q}$, we calculate the similarity between the nearest neighbor and the query point $\boldsymbol{q}$. The similarity between these two points reads as

$$S(\boldsymbol{x}_{L\{i\}},\boldsymbol{q})=\begin{cases}1, & if\quad \boldsymbol{x}_{L\{i\}}==\boldsymbol{q}\\ \dfrac{\left(\dfrac{1}{\|\boldsymbol{x}_{L\{i\}}-\boldsymbol{q}\|}\right)^{\frac{2}{p-1}}}{\sum_{j=1}^{k}\left(\dfrac{1}{\|\boldsymbol{x}_{L\{j\}}-\boldsymbol{q}\|}\right)^{\frac{2}{p-1}}}, & if\quad \boldsymbol{x}_{L\{i\}}\neq\boldsymbol{q}\end{cases} \quad (6)$$

Here, $L\{i\}$ stands for the ith element of the set L defined as in (1), $\boldsymbol{x}_{L\{i\}}$ is an instance among the neighborhood of query point $\boldsymbol{q}$, and p is the fuzzification coefficient. We note that $0<S(\boldsymbol{x}_{L\{i\}},\boldsymbol{q})\le 1$.

4. Fuzzy Learning Vector Quantization Algorithm

As shown in section II, the learning vector quantization algorithms are based on the simple competitive learning. In other words, learning vector quantization is based on the winner-take-all strategy. In this paper, we modify and expand learning vector quantization using the concept of *k*-nearest neighbors and fuzzy set theory.

For the conventional learning vector quantization with the simple competitive learning, there is only one winning prototype which is the nearest prototype to the given data point. However, it is possible for this strategy not to use the information perfectly. Although there is the opportunity to learn the other prototypes which are not winner under the simple competitive strategy, the prototypes except the winning prototype are not learned. To overcome the above mentioned problem, we modify (2) into (7) using fuzzy *k*-nearest neighbor approach.

$$u_i(\boldsymbol{x})=\begin{cases}\dfrac{\left(\dfrac{1}{\|\boldsymbol{x}-\boldsymbol{v}_i\|}\right)^{\frac{2}{p-1}}}{\sum_{j=1}^{k}\left(\dfrac{1}{\|\boldsymbol{x}-\boldsymbol{v}_{L\{j\}}\|}\right)^{\frac{2}{p-1}}}, & if\ i\in L\\ 0, & if\ i\notin L\end{cases} \quad (7)$$

Here, L is the set of indexes for the *k* nearest prototypes to the given data point and k means the predefined number of the nearest neighbor prototypes. For (7), the number of prototypes which can be learned is “k” not 1. The modification of the loss function results in the modification of updating rule of the prototypes. The conventional updating rule is defined as (8)

$$\boldsymbol{v}_i(t+1)=\begin{cases}\boldsymbol{v}_i(t)+\eta(\boldsymbol{x}-\boldsymbol{v}_i(t)) & Label(\boldsymbol{x})=Label(\boldsymbol{v}_i)\\ \boldsymbol{v}_i(t)-\eta(\boldsymbol{x}-\boldsymbol{v}_i(t)) & Label(\boldsymbol{x})\neq Label(\boldsymbol{v}_i)\end{cases} \quad (8)$$

Here, $Label(\boldsymbol{x})$ and $Label(\boldsymbol{v}_i)$ mean the class label of the given data point $\boldsymbol{x}$ and the prototype $\boldsymbol{v}_i$ respectively.

The modified updating rule is defined as (9).

$$\boldsymbol{v}_i(t+1) = \begin{cases} \boldsymbol{v}_i(t) + \eta \cdot u_i \cdot (\boldsymbol{x} - \boldsymbol{v}_i(t)) & Label(\boldsymbol{x}) = Label(\boldsymbol{v}_i) \\ \boldsymbol{v}_i(t) - \eta \cdot u_i \cdot (\boldsymbol{x} - \boldsymbol{v}_i(t)) & Label(\boldsymbol{x}) \neq Label(\boldsymbol{v}_i) \end{cases} \quad \forall i \in L \quad . \tag{9}$$

Figure 1 and 2 show the updating process of the conventional learning vector quantization based on the simple competitive learning and the updating process of the proposed learning vector quantization respectively.

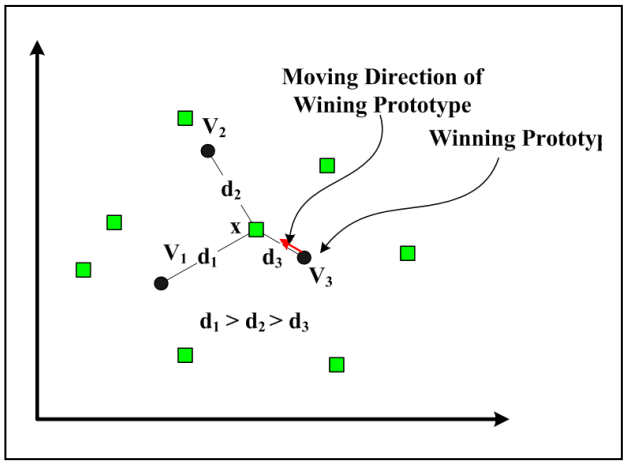

Fig. 1. Learning Vector Quantization based on Simple Competitive Learning

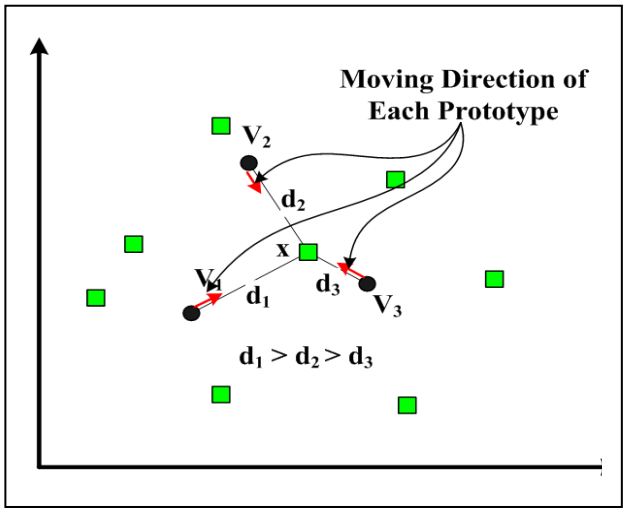

Fig. 2. The Proposed Learning Vector Quantization

5. Simulation Studies

We perform experiments in order to evaluate and quantify the effectiveness of the proposed fuzzy learning vector quantization classifier. The proposed classification method is experimented making use of a series of numeric data such as some synthetic datasets and several machine learning datasets (http://www.ics.uci.edu/~mlearn/MLRepository.html). In the assessment of the performance of the classifiers, we use the error rate of the classifier. In the simulations, K-fold cross-validation has been applied to evaluate the classification result. The whole data set is divided into K blocks using K-1 blocks as a training set, and the remaining block as test set. Here the number of blocks is set to 5. We investigate and report the results of each simulation in terms of the mean and the standard deviation of the performance index.

The classification performance of classifiers is quantified in terms of the Error Rate being expressed as

$$Error\ Rate\ (\%) = \frac{\sum_{i=1}^{n} f\left(Label(\boldsymbol{x}_i), \hat{t}(\boldsymbol{x}_i)\right)}{n} \cdot 100 \tag{10-1}$$

$$f(a, b) = \begin{cases} 1, & a \neq b \\ 0, & a = b \end{cases} \tag{10-2}$$

We consider some predefined values of the parameters whose values are summarized in Table 1. The choice of these particular numeric values has been motivated by the need to come up with a possibility to investigate of the performance of the model in a fairly comprehensive range of scenarios.

Table 1. Selected numeric values of the parameters of the proposed technique

Parameter	Value
Size of codebook in each class (C)	10, 20, 30, 40
Number of nearest neighbor prototypes (k)	2, 5, 10, and whole prototypes
Fuzzification Coefficient (p)	2

5.1 Synthetic Datasets

The two-dimensional synthetic examples are suitable for illustrating the boundary area and understanding the characteristic of the proposed prototype generating technique. We use some normally distributed data set composed of sub-groups described by their mean vector and covariance matrix as shown in Figure 3. Figure 3 shows 2sub-groups governed by the corresponding Gaussian distributions with mean vector $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Phi}$. Each class is composed of two clusters. The mean vectors $\boldsymbol{m}_i$ and covariance matrix $\boldsymbol{\Phi}$ (being the same for all groups) are the following

$$\boldsymbol{m}_1 = \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}, \quad \boldsymbol{m}_2 = \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}, \quad \boldsymbol{m}_3 = \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}, \quad \boldsymbol{m}_4 = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix},$$

$$\Phi = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}.$$

Each sub-group consists of 200 data.

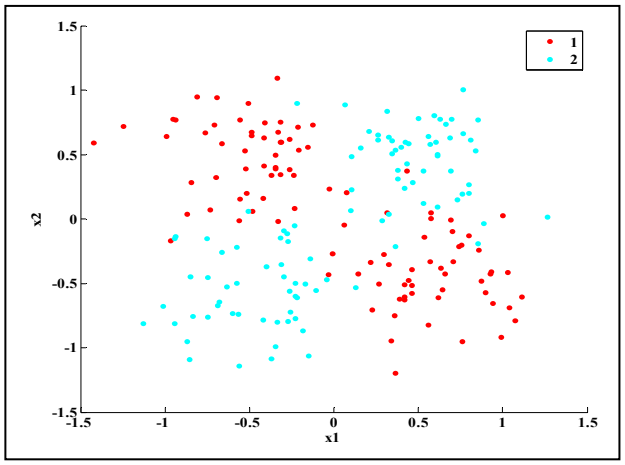

Fig. 3. Two-class synthetic dataset based on a mixture of Gaussian distributions

Table 2 summarizes the classification performance (%) of the Nearest Neighbor (NN) classifier with the generated prototypes. From the comparative analysis we can say that the proposed learning strategy shows the better performance than the generic learning strategy does.

Table 2. Classification Result of comparative analysis between the proposed classifier and generic LVQ classifier

Classifier	C	k	Error Rate (%) (mean±STD)
Proposed LVQ Classifier	10	2	13.1±1.78
		5	13.0±2.42
		10	13.4±2.53
		20	13.2±2.64
	20	2	12.9±2.71
		5	12.1±2.53
		10	12.7±0.91
		40	12.0±1.32
	30	2	14.7±2.89
		5	14.5±2.15
		10	14.3±3.23
		60	14.7±3.01
	40	2	12.5±2.09
		5	12.8±2.20
		10	12.3±2.11
		80	10.4±2.51
Learning Vector Quantization	10		13.3±0.97
	20		14.2±1.60
	30		13.2±4.04
	40		13.0±3.34

5.2 Machine Learning Datasets

In what follows, we make several experiments with some machine learning data sets concerning classification problems which are provided at the machine learning repository (http://www.ics.uci.edu/~mlearn/MLRepository.html). Table 3 summarizes the pertinent details of the data such as the number of features and number of patterns.

Table 3. Machine Learning datasets used in the experiments

Data sets	Number of features	Number of patterns (data)	Number of Class
Australian	42	690	2
Balance	4	625	3
Liver	6	347	2
Glass	9	214	6
Iris	4	150	3

Table 4 summarizes the classification performance (%) of the proposed classifier *vis-à-vis* the generic LVQ classifier for 5 data sets, respectively.

The comparative analysis demonstrates that the proposed learning strategy is better than LVQ classifier with the simple competitive learning strategy in terms of the classification performance. Especially, for Balance data set, the NN classifier with the proposed learning strategy is very superior to the LVQ classifier with simple competitive learning strategy.

Table 4. Classification performance (%) of the proposed classifier *vis-à-vis* the generic LVQ classifier

Data Sets		Proposed Classifier	LVQ Classifier
Australian	C	40	30
	k	5	N/A
	Error Rate(%) (mean±STD)	31.91±0.93	32.84±0.80
Balance	C	20	10
	k	60	N/A
	Error Rate(%) (mean±STD)	10.34±0.09	22.18±0.46
Liver	C	10	10
	k	2	N/A
	Error Rate(%) (mean±STD)	32.58±1.54	34.90±2.51
Glass	C	40	30
	k	10	N/A
	Error Rate(%) (mean±STD)	29.16±1.13	31.68±1.42
Iris	C	20	20
	k	10	N/A
	Error Rate(%) (mean±STD)	2.93±0.60	3.87±0.56

6. Conclusion

In this paper, we proposed the new learning vector quantization technique using the fuzzy k-nearest neighbors approach. The conventional learning vector quantization method is based on the simple competitive learning which chooses the only one winning prototype. The remainders of prototypes except the winning prototype are not updated. In the proposed technique, the number of winning prototypes is more than 1 and the winning prototypes are selected by k-nearest neighbor prototypes to the given data pattern. The weighting values are assigned to the selected k-nearest neighbor prototypes according to the distance between the prototype and the given data pattern. The bigger the weighting value is, the smaller the distance between the prototype and the given data pattern. The weighting values are applied to the updating rule of the prototypes. To evaluate the proposed model for classification problem, we made several experiments using 2-dimensional synthetic datasets and various machine learning datasets. From the experiments, we can see that the proposed fuzzy learning vector quantization can improve the classification performance.

References

[1] M.N. Nasser and A.K. Robert, "Image coding using vector quantization: a review," *IEEE Trans. Commun*., vol. 36, no. 8, pp. 957–971, 1988.

[2] Gersho, R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, MA, 1992.

[3] T. Kohonen, "The self-organization map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464-1480, 1990.

[4] T, Kohonen, "Improved versions of learning vector quantization," in: *Proceedings of the International Joint Conference Neural Networks*, vol. I, pp. 545–550, San Diego, CA, 1990.

[5] N.R. Pal, J.C. Bezdek, and E.C.-K. Tsao, "Generalized clustering network and Kohonen's self-organizing schemes," *IEEE Trans. Neural Network*, vol. 4, no. 4, pp 549-557, 1993.

[6] N.B. Karayiannis and P.I. Pai, "Fuzzy algorithms for learning vector quantization," *IEEE Trans. Neural Network*, vol. 7, no. 5, pp. 1196-1211, 1996.

[7] N.B. Karayiannis, "A methodology for constructing fuzzy algorithms for learning vector quantization," *IEEE Trans. Neural Network*, vol. 8, no. 3, pp 505-518, 1997.

[8] J. M. Keller, M. R. Gray and J. A. Givens Jr., "A Fuzzy K-Nearest NeighborAlgorithm," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-15, no. 4, pp. 580-585, July 1985.

[9] T. M. Cover and P. E. Hart, "Nearest Neighbor pattern Classification," *IEEE Trans. Inform. Theory*, vol. IT-13, no. 1, pp. 21-27, Jan. 1967.

[10] T. M. Cover, "Estimation by the Nearest Neighbor Rule," *IEEE Trans. Inform. Theory*, vol. IT-14, no. 1, pp. 50-55, Jan. 1968.

Seok-Beom Roh received the B.S., M.S., and PhD. degrees in control and instrumentation engineering from the Wonkwang University, South Korea in 1994, 1996, and 2006 respectively. He is currently a research professor at the Wonkwang University. His research interests include fuzzy set, neural networks, genetic algorithms, statistical learning and computational intelligence.

Ji-Won Jeong received the B.S. degree in Department of Electrical Electronic and Information Engineering from the Wonkwang University, South Korea in 2010. He is currently working towards the Master of Science degree in Control and Instrumentation Engineering from the Wonkwang University. His research interests include circuit simulation and computational intelligence.

Tae-Chon Ahn is a professor at Wonkwang University in South Korea. He was an Editor-in-Chief in Institute of Control, Robotics and Systems in South Korea. He received BS, MS and PhD in Electrical engineering from Yonsei University, South Korea, in 1978, 1980, and 1986, respectively. His research interests include system modeling and knowledge discovery, machine learning, bio-inspired optimizations, computational intelligence and pattern recognition.