Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 20, Sayı 3, 414-420, 2016

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 20, Issue 3, 414-420, 2016

DOI: 10.19113/sdufbed.22419

A Geometrical Modification of Learning Vector Quantization Method for Solving Classification Problems

Korhan GÜNEL *[1], **Rıfat AŞLIYAN** [1], **İclal GÖR** [1]

[1]Adnan Menderes University, Faculty of Arts and Sciences, Department of Mathematics, 09010, Aydın

(Alınış / Received: 15.06.2016, Kabul / Accepted: 23.08.2016, Online Yayınlanma / Published Online: 08.09.2016)

Keywords
Machine learning,
Learning vector quantization,
Geometrical learning approach

Abstract: In this paper, a geometrical scheme is presented to show how to overcome an encountered problem arising from the use of generalized delta learning rule within competitive learning model. It is introduced a theoretical methodology for describing the quantization of data via rotating prototype vectors on hyper-spheres.
The proposed learning algorithm is tested and verified on different multidimensional datasets including a binary class dataset and two multiclass datasets from the UCI repository, and a multiclass dataset constructed by us. The proposed method is compared with some baseline learning vector quantization variants in literature for all domains. Large number of experiments verify the performance of our proposed algorithm with acceptable accuracy and macro f_1 scores.

Sınıflandırma Problemlerinin Çözümü için Destekleyici Öğrenmeli Vektör Nicemleme Metodunun Geometrik Modifikasyonu

Anahtar Kelimeler
Makine öğrenmesi,
Destekleyici öğrenmeli vektör nicemleme,
Geometrik öğrenme yaklaşımı

Özet: Bu çalışmada yarışmacı öğrenme modelinde kullanılan genelleştirilmiş delta öğrenme kuralı ile ortaya çıkan problemin üstesinden gelmek için geometrik bir yaklaşım önerilmiştir. Veri nicemlemesinin izah edilebilmesi için prototip vektörlerinin hiper-küreler üzerinde döndürülmesi esasına dayalı teorik bir metodoloji geliştirilmiştir. Önerilen öğrenme algoritması UCI veri havuzundan alınmış bir tanesi ikili sınıflandırma veri seti ve iki tanesi çok sınıflı olan veri setleri ile bir tanesi tarafımızdan hazırlanan çoklu sınıf veriseti üzerinde test edilmiş ve geçerliliği denetlenmiştir. Önerilen metot, literatürde referans alınan bazı destekleyici öğrenmeli vektör nicemleme ağ varyasyonları ile farklı alanlarda karşılaştırılmıştır. Çok sayıdaki deneysel çalışmalar, makul doğruluk değerleri ve makro f_1 skorları ile önerilen algoritmanın performansını doğrulamaktadır.

1. Introduction

For solving classification problems, numerous techniques using a geometrical approach are presented in statistical learning theory [1–3]. Cabrelli et al. (2000) constructed Convex Recursive Deletion Regions (CoRD) that are able to be classified by two-layer perceptron networks [4]. Wang and Chaudhari (2004) designed Fast Covering Learning Algorithm (FCLA) for Boolean Neural Networks which separate input space by using the intersection of a reference hyper-sphere and three concentric hyper-spheres based on three different radii [5].
Shoujue and Jiangliang (2005) suggested a geometrical learning algorithm using descriptive geometry. In this algorithm, hyper-sausage units were presented as simple geometrical units for learning samples. Geometrical features of high dimensional constructions of samples were investigated by way of projections from high dimension to lower dimension [6].
Bayro-Corrochano and Anana-Daniel introduce a new algorithm based on clifford geometric algebra. This method is the generalization of the real and complex valued Support Vector Machines called the Clifford Support Vector Machines (CSVM) [7].
Zhang et al. (2005) use the multinomial manifold for text classification using Euclidean geometry [8].
Delogue et al. (2008) presented a new approach to classify patterns defined in a real domain by using linear programming for determining Polyhedron of Minimal Volume. The authors used a constructive algorithm in order to achieve a reduced computation complexity within the multi-layer perceptron networks [9].
Liu et al. (2009) define a geometric method called scaled convex hull (SCH) based on a theoretical approach [10].
Nova and Estéves (2014) emphasize some advantages of Learning Vector Quantization (LVQ) algorithms compared with Multi Layer Perceptron (MLP) and Support Vector Machine (SVM) [11]. The construction of LVQ topology is simple because of fixed number of free param-

* *Corresponding author: kgunel@adu.edu.tr*

eters as number of prototype vectors. Also, LVQ needs fewer computational workload in comparison with MLP and SVM. In LVQ, the prototype vectors are used to represent the classes in learning domain. However, the generalized delta learning rule of LVQ may cause the prototype vectors move far away from the center of the class, which is referenced by the prototype vectors, after some iterations.

In this study, we have developed a rigorous and quite general framework via some reference hyper-spheres for classification problems to overcome the mentioned problem with the generalized delta learning rule. The proposed approach follows quite naturally by changing the viewpoint on the learning rule of well-known LVQ methods. The main contributions of this study are partially to handle the bottleneck of generalized delta learning rule, and to guarantee the learning. In addition, it is shown that while the prototype vectors are converged or diverged to the input vector, they can be moved on a curve instead of moving linearly.

In the sequel, firstly, the Learning Vector Quantization is introduced as a preliminary, and some variants of LVQs are presented briefly. Then, the proposed method based on geometrically heuristic approach to separate the input space has been mentioned in detail. Finally, the experimental results and some important issues about the proposed learning approach have been discussed according to the existing knowledge.

1.1. Some variants of Learning Vector Quantization

LVQ algorithm is firstly introduced by Kohonen as a heuristic classification method [13], and other variants of LVQs in the literature can be categorized according to their learning rules [11, 12]. Generally, LVQ methods are decomposed into 3 branches as heuristic methods (Kohonen LVQs), marjin maximization and likelihood ratio maximization methods [11].

Learning Vector Quantization (LVQ) is used to generate the Kohonen codebook vectors that characterize the classes in an input space. The network topology of LVQ consists of three layers as an input layer, a Kohonen layer and an output layer. Each neuron in the output layer indicates a class of input space, and each class is referenced by some vectors in the Kohonen layer. The main idea of LVQ is that the class of an input vector $\mathbf{x}$ is specified by the nearest Kohonen codebook vector.

Initially, each prototype vector is predefined or randomly generated. First, given an input vector $\mathbf{x}$ belonging to the training data set, the winning neuron is determined via calculating Euclidean distance. The winning neuron is the nearest prototype vector to the input vector, $\mathbf{x}$. If the topology of LVQ has totally M vectors in the Kohonen layer, the winning vector, or in other words, the weight of winning neuron is updated by the Kohonen learning rule given in the Eq. (1.1).

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) - \lambda(t)(\mathbf{x} - \mathbf{w}_k(t)) \tag{1.1}$$

where $\mathbf{w}_k$ is the nearest prototype vector such that $k = \arg\min_{1\le i\le M}\{\|\mathbf{x}-\mathbf{w}_i\|\}$ [13]. In Eq. (1.1), λ is a constant or monotonically decreased function defined as the learning rate such that $0 < \lambda < 1$ and t denotes the iteration value or time. The version of LVQ is known as LVQ1.

In LVQ2 algorithm, the local and global winning neurons as prototype vectors are determined, firstly [14]. While the local winning vector, $\mathbf{w}_l$ is the nearest vector to the input vector $\mathbf{x}$, which belongs to the same class with $\mathbf{x}$, the global winning vector, $\mathbf{w}_g$ is the nearest vector to the input among all prototype vectors. In this approach, the local winner, $\mathbf{w}_l$ is moved to closer to the input $\mathbf{x}$ using the Eq. (1.2) when the condition $\min\left(\frac{d_l}{d_g}, \frac{d_g}{d_l}\right) > s$ is satisfied such that $d_l = |\mathbf{x}-\mathbf{w}_l|$, $d_g = |\mathbf{x}-\mathbf{w}_g|$ and s is arbitrary selected relative window width. The idea is to fall the input $\mathbf{x}$ into the relative window defined around the midplane of $\mathbf{w}_l$ and $\mathbf{w}_g$. Kohonen (1996) suggest that a relative window width, s can be selected as a value between 0.2 and 0.3 [15]. Furthermore, the global winner, $\mathbf{w}_g$ is moved away from the input, $\mathbf{x}$ with the Eq. (1.3) under the same conditions in the improved version of LVQ2 called as LVQ2.1. However, this process can not guaranty the convergence.

$$\begin{aligned} \mathbf{w}_l(t+1) &= \mathbf{w}_l(t) - \lambda(t)(\mathbf{x}-\mathbf{w}_l(t)), \qquad (1.2)\\ \mathbf{w}_g(t+1) &= \mathbf{w}_g(t) + \lambda(t)(\mathbf{x}-\mathbf{w}_g(t)) \qquad (1.3)\end{aligned}$$

where $l = \arg\min_{i_1\le i\le i_m}\{\|\mathbf{x}-\mathbf{w}_i\|\}$ such that the Kohonen neurons indexed between i_1 and i_m are in the same class with $\mathbf{x}$ for $m < n$, and $g = \arg\min_{\substack{1\le j\le n,\\ j\notin\{i_1,\ldots,i_m\}}}\{\|\mathbf{x}-\mathbf{w}_j\|\}$.

Kohonen (1990) proposed the third version of LVQ which has the same process as in LVQ2.1 except that $g = \arg\min_{1\le j\le M}\{\|\mathbf{x}-\mathbf{w}_j\|\}$ [13]. That means, the $\mathbf{w}_l$ and $\mathbf{w}_g$ can fall in the same class. As a result, an extra weight updating is applied using the Eq. (1.4) for $k\in\{l,g\}$.

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \varepsilon\lambda(t)(\mathbf{x}(t)-\mathbf{w}_k(t)), \tag{1.4}$$

where $\varepsilon\in(0,1)$ is a stabilizing constant factor. The value of ε should reflect the width of the adjustment window around the border between the classes of $\mathbf{w}_l$ and $\mathbf{w}_g$. However, the problem of LVQ2.1 is still remained for LVQ3 algorithm [16].

Sato and Yamada (1995) suggested the Generalized Learning Vector Quantization (GLVQ) algorithm to minimize the cost function, E to ensure the prototype vectors continue approximating the class distribution via the relative distance difference defined in the Eq. (1.5)

$$\mu(\mathbf{x}) = \frac{d_l - d_g}{d_l + d_g}, \tag{1.5}$$

where d_l and d_g are the distances between the input vector, $\mathbf{x}$ and the prototype vectors $\mathbf{w}_l$ and $\mathbf{w}_g$ respectively, and $\mu(\mathbf{x})$ is the classifier function [16]. To minimize the cost function $E = \frac{1}{2}\sum_{k=1}^{N} f(\mu(\mathbf{x}_k))$, the weights of the winning vectors $\mathbf{w}_l$ and $\mathbf{w}_g$ are updated using the Eq. (1.6) based on the steepest descent method such that N is the total number of samples in the training data set.

$$\mathbf{w}_l = \mathbf{w}_l + \lambda\frac{\partial f}{\partial\mu}\frac{d_g}{(d_l+d_g)^2}(\mathbf{x}-\mathbf{w}_l), \tag{1.6}$$

$$\mathbf{w}_g = \mathbf{w}_g - \lambda \frac{\partial f}{\partial \mu} \frac{d_l}{(d_l+d_g)^2}(\mathbf{x}-\mathbf{w}_g) \tag{1.7}$$

where $d_l = |\mathbf{x}-\mathbf{w}_l|^2$ and $d_g = |\mathbf{x}-\mathbf{w}_g|^2$. In GLVQ, the convergence property of LVQ depends on the definition of the cost function, and the function $f(\mu,t)$ can be selected as the sigmoid function of $1/(1+\exp(-\mu t))$.

In the optimized versions of LVQ1 and LVQ3, the process are the same as LVQ1 and LVQ3 respectively, except that each prototype vector has its own learning rate, λ_i for $1 \le i \le M$. The optimal values of λ_i are determined by a recursion function given in the Eq. (1.8).

$$\lambda_i(t) = \frac{\lambda_i(t-1)}{1+s.\lambda_i(t-1)} \tag{1.8}$$

where $s = 1$ if $\mathbf{w}_i$ and $\mathbf{x}$ are in the same class, namely the classification is correct, otherwise $s = -1$. Kohonen et al. (1996) claim that the optimized version of LVQs provides fast convergence, however it cannot be applied to LVQ2.1 because of divergency [15].

Hammer and Villmann (2002) presented Generalized Relevance Learning Vector Quantization (GRLVQ) for enlarging GLVQ [17]. In GRLVQ, the similarity measure is an adaptive metric which uses the weighted counterpart with relevance weights of the vectors as seen in the Eq. (1.9).

$$d_r(\mathbf{x},\mathbf{w}) = \sum_i r_i(x_i - w_i)^2 \tag{1.9}$$

where r_i is relevance factor such that $r_i < 0$ initially and $\sum_i r_i = 1$. It is obvious that the usage of relevance factors allows the scaling of the input dimensions. Learning process of GLVQ is done by stochastic gradient descent with respect to the cost function E with the Eq. (1.10), and the relevance learning is performed by adaptation of the relevance weights again by gradient descent with the Eq. (1.11) where ∂_S denotes the stochastic gradient.

$$\frac{\partial_S E}{\partial \mathbf{w}_l} = \frac{\partial_S E}{\partial d_l}\frac{\partial d_l}{\partial \mathbf{w}_l}, \qquad \frac{\partial_S E}{\partial \mathbf{w}_g} = \frac{\partial_S E}{\partial d_g}\frac{\partial d_g}{\partial \mathbf{w}_g} \tag{1.10}$$

where

$$\frac{\partial_S E}{\partial d_l} = \frac{2d_g.f'(\mu(\mathbf{w}))}{(d_l+d_g)^2}, \qquad \frac{\partial_S E}{\partial d_g} = -\frac{2d_l.f'(\mu(\mathbf{w}))}{(d_l+d_g)^2}.$$

$$r_i = r_i - \varepsilon_r \frac{\partial_S E}{\partial r_i} \tag{1.11}$$

where $0 < \varepsilon_r < 1$, and $\frac{\partial E_S}{\partial r_i} = \frac{2d_g.f'(\mu(\mathbf{w}))}{(d_l+d_g)^2}\frac{\partial d_l}{\partial r_i} - \frac{2d_l.f'(\mu(\mathbf{w}))}{(d_l+d_g)^2}\frac{\partial d_g}{\partial r_i}$.

Seo and Obermayer (2003) derived a learning rule applying gradient descent on a cost function based on a likelihood ratio in Soft Learning Vector Quantization (SLVQ) [18]. In the improved version of SLVQ called Robust Soft Learning Vector Quantization, the cost function is optimized without the window rule. Biehl et al. (2007) examined the generalization ability of LVQ Algorithms with a theoretical perspective [19].

In Generalized Matrix Learning Vector Quantization (GMLVQ) algorithm, a parametric quadratic form of the distance given in the Eq. (1.12) is used in the improved version of GRLVQ [20].

$$d_\Lambda(\mathbf{x},\mathbf{w}) = (\mathbf{x}-\mathbf{w})^T \Lambda (\mathbf{x}-\mathbf{w}), \tag{1.12}$$

where Λ is a quadratic, positive, semi-definite matrix.

Kaestner et al. (2012) suggested functional relevance in learning vector quantization to reduce the free parameters for updating relevance factor [21]. In Generalized Functional Relevance Learning Vector Quantization (GFRLVQ), the relevance factor, r_i is interpreted as a function of $r_i = r(t_i)$ and it is defined as a linear combination of some basis functions such as Gaussian or Lorentzian type functions that have only single peak.

$$r(t_i) = \sum_j \beta_j \kappa_j(\omega_j, t_j) \tag{1.13}$$

where the basis functions κ_j have few parameters $\omega_j = (\omega_{j,1}, \omega_{j,2}, \ldots, \omega_{j,p})^T$ and $\beta_j > 0$ and $\sum_j \beta_j = 1$.

Hammer et al. (2014) extend the properties of LVQ by applying the learning algorithm to similarity and dissimilarity data [22].

Hofmann et al. (2015) propose kernel robust soft LVQ (RSLVQ) which is capable of classifying complex data sets [23]. They describe a general gram matrix including low rank approximations and how the models could be implemented in this approach.

Bohnsack et al. (2016) improve LVQ method in order to get the classification of matrix data based on matrix norms [24]. In general learning algorithms are based on vectorial approach, the contribution of the article is to work for matrix norms.

The advances in the literature give us an idea to solve to the mentioned problem of LVQ with evading the unjustifiable workload. The idea is presented in the next section in detail.

2. Learning Algorithm based on Rotating Kohonen Codebook Vectors on Hyper-spheres

The geometric interpretation of the Kohonen rule is to decrease or increase the distance between the prototype vector, $\mathbf{w}_i$ and the input vector, $\mathbf{p}$ according to the learning rate, λ shown in Fig. 1. An unsolved problem is occurred because of using the Kohonen learning rule, which causes the divergence of Kohonen codebook vectors from the boundary of the class as seen in the Fig. 1. Namely, assuming that $\mathbf{p}$ belongs to the class, A, and $\mathbf{w}$ is to be a prototype vector for the class, C. If $\mathbf{w}$ is the global winner, then it is moved away from the vector, $\mathbf{p}$ according to generalized delta learning rule. Although it is not expected to move away from the surrounding of the class C, when $\mathbf{w}$ is diverged from $\mathbf{p}$, it is also moved away from the class, C, and it enters the domain of the class, B. To handle this problem, we propose a new learning rule without relative window parameter for learning vector quantization.

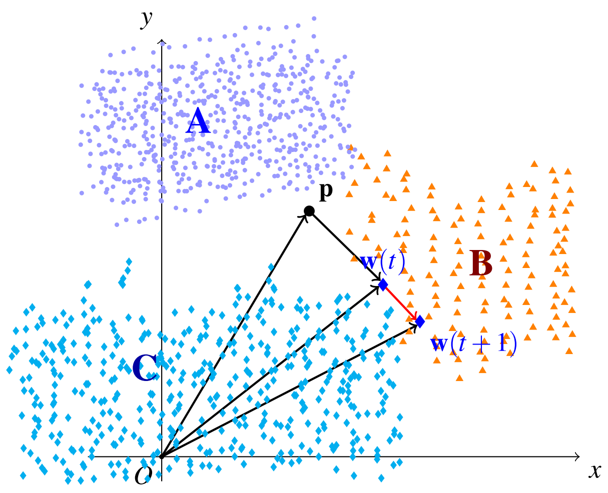

Figure 1. The main bottleneck of delta learning rule for adaptation of codebook vectors in LVQ.

In machine learning methods, unadjusted free parameters without optimization cause unexpected test results, and the optimization of free parameters is also a difficult task in neural networks. In other words, less number of free parameters means less workload. Buchala et al. (2005) emphasize that the using of large number of free parameters lead to unsuccessful test results [25]. They conclude that the bad classification performance has occurred when presenting irrelevant information by using considerable number of free parameters. Furthermore, Maeda and Ishii (2009) mentioned about the phenomena known as overfitting arise from too many parameters [26]. In vector quantization to avoid overfitting, reducing the effectness of many free parameters is important. Denil et al. (2013) propose a general purpose technique for decreasing a number of free parameters in neural networks [27]. With this manner, Seo and Obermayer (2003) developed Robust Soft Learning Vector Quantization method using a cost function without the window rule for optimising the vector quantization [18].

In this way, we developed a heuristic learning vector quantization method removing the window width parameter, which is initialized as a fixed and an arbitrary free parameter. The principal idea is to rotate the prototype vector, $\mathbf{w}$ onto a hypersphere whose center is the centroid of the class represented by the winning prototype vector. The radius of the hypersphere is the distance between the centroid of the class and the winning prototype vector. This hypersphere is not a free parameter, it is calculated dynamically. We called the proposed method as LVQRKV (Learning Vector Quantization with Rotating Kohonen codebook Vectors).

Given a set of points $(x_1, x_2, \ldots, x_n)$ in n-dimensional Euclidean space, a hypersphere, S_{n-1} is represented by $\sum_{i=1}^{n}(x_i - c_i)^2 = r^2$ where $\mathbf{x}$ is the input vector, $\mathbf{c}$ is the center and r is the radius of S_{n-1}.

In this paper, initially, we construct two different reference hyperspheres. One of the hyperspheres has the input vector $\mathbf{p}$ as the center point and $|\mathbf{pw}|$ as the radius where $\mathbf{w}$ is the winning prototype vector. The other hypersphere is formed by the midpoint between the prototype vector $\mathbf{w}$ and a centroid of a class, ie. $\mathbf{m}_A$ as a center and $|\mathbf{wm}_A|$ as the diameter. In training stage, the hyperspheres are constructed dynamically at each iteration. In other words, these hyperspheres are generated for each input vector.

On account of the fact that an n-dimensional Euclidean space is analogous to the spherical coordinate system, the coordinates of $\mathbf{w}$ consist of a radial coordinate $\rho = |\mathbf{w}|$, and $n-1$ angular coordinates $\alpha_1, \alpha_2, \ldots, \alpha_{n-1}$ where α_{n-1} ranges over $[0, 2\pi)$ radians and the other angles range over $[0, \pi]$ radians. Then, the coordinates of the vector $\mathbf{w}$ are formulated as

$$\left.\begin{array}{rcl} w_1 &=& \rho\cos(\alpha_1), \\ w_2 &=& \rho\sin(\alpha_1)\cos(\alpha_2), \\ &\vdots& \\ w_{n-1} &=& \rho\sin(\alpha_1)\sin(\alpha_2)\cdots\sin(\alpha_{n-2})\cos(\alpha_{n-1}) \end{array}\right\} \tag{2.1}$$

and

$$w_n = \rho\sin(\alpha_1)\sin(\alpha_2)\cdots\sin(\alpha_{n-2})\sin(\alpha_{n-1}). \tag{2.2}$$

Now, if the vector $\mathbf{w}$ is rotated by $\Delta\alpha$ radian around the α_1-axis, $\mathbf{w}$ is moved to a new point $\mathbf{w}(t+1)$ with the angular coordinate $\alpha_1 = \alpha_1 - s\Delta\alpha$ where s specifies the direction of the rotation and t is the iteration value. If $s = 1$, $\mathbf{w}$ is rotated about α_1-axis in a clockwise direction, and if $s = -1$ then the direction is counterclockwise.

If the inverse transforms are applied, the angular coordinates of $\mathbf{w}$ can be written as seen in Eq. (2.3) and Eq. (2.4).

$$\alpha_i = \operatorname{arccot}\left(\frac{w_i}{\sqrt{\sum_{j=i+1}^{n} w_j^2}}\right) \tag{2.3}$$

for $1 \le i < n-1$, and

$$\alpha_{n-1} = 2\operatorname{arccot}\left(\frac{\sqrt{w_{n-1}^2 + w_n^2} + w_{n-1}}{w_n}\right). \tag{2.4}$$

Euclidean coordinate of $\mathbf{w}(t+1)$ is recalculated again by the Eq. (2.1) and Eq. (2.2). This process guarantees that $\mathbf{w}$ converges both of the centroid of the class and the input. In Fig. 2, $\mathbf{w}(t+1)$ is an interior point of the hypersphere, so $|\mathbf{pw}(t+1)| < |\mathbf{pw}(t)|$. On the other hand, $|\mathbf{m}_A\mathbf{w}(t+1)| < |\mathbf{m}_A\mathbf{w}(t)|$ because the longest chord is passed through the center of the hypersphere. Also, it is obvious that $\mathbf{w}(t+1)$ is still on the hypersphere whose center is $\mathbf{c}$ because the length of diameter is not changed during the process.

Moreover, if $\mathbf{w}$ is the global winner, then it is diverged from the input vector $\mathbf{p}$ belonging to the class, A, and it is converged to the centroid of the other class, B in the proposed algorithm. Therefore, the global winner vector, $\mathbf{w}$ is rotated onto hypersphere whose center is the midpoint between the prototype vector, $\mathbf{w}$ and the centroid of the class, B and radius is $|\mathbf{cw}|$ as seen in Fig. 3. In this case, $\mathbf{w}(t+1)$ must be outside of the hypersphere whose center is the input vector, $\mathbf{x}$ and radius is the $|\mathbf{pw}|$ after rotation. The learning is guaranteed again, because $|\mathbf{m}_B\mathbf{w}(t+1)| < |\mathbf{m}_B\mathbf{w}(t)|$ and $|\mathbf{pw}(t+1)| > |\mathbf{pw}(t)|$.

Then, the gradient descent algorithm is applied for updating α_1 angular parameter with respect to the cost function

E using generalized delta learning rule as in Eq. (2.5).

$$\alpha_1 = \alpha_1 - \lambda \frac{\partial E}{\partial \alpha_1} \tag{2.5}$$

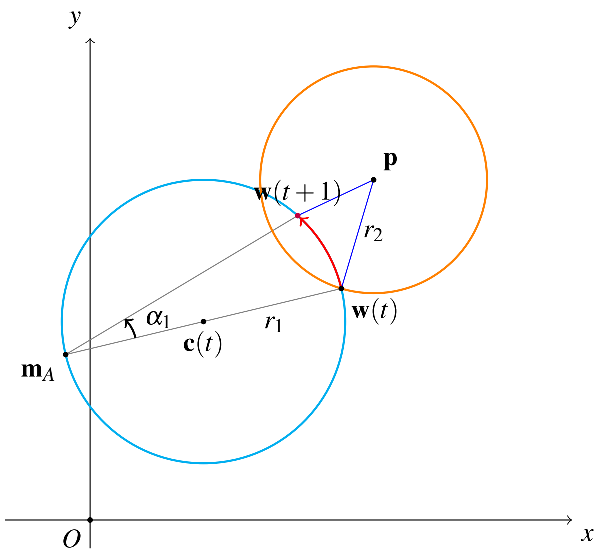

Figure 2. If $\mathbf{w}(t+1)$ is inside of the hyper sphere whose center is $\mathbf{p}$ and radius is $|\mathbf{pw}|$, the rotating process guarantees the learning.

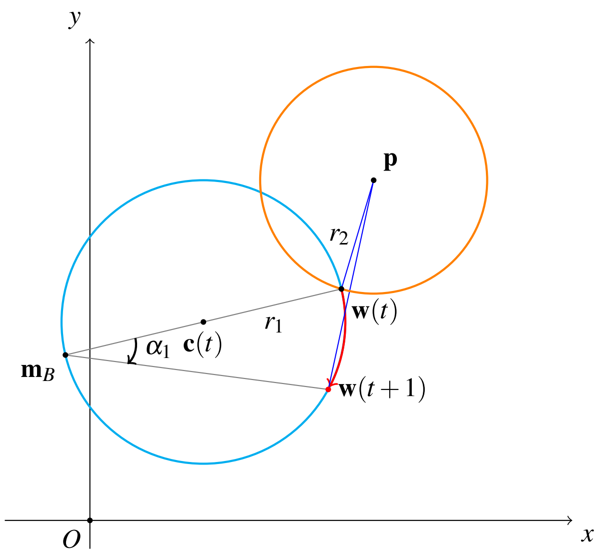

Figure 3. If $\mathbf{w}(t+1)$ is outside of the hyper sphere whose center is $\mathbf{p}$ and radius is $|\mathbf{pw}|$. The rotating process guarantees the learning.

The cost function, E in Eq. 2.5 is essentially defined as similar with the cost function in the GLVQ [16]. The only difference occurs in the definition of the classifier function $\mu(\mathbf{p})$ as seen in Eq. 2.6.

$$\mu(\mathbf{p}) = \frac{\alpha_{1l} - \alpha_{1g}}{\alpha_{1l} + \alpha_{1g}}, \tag{2.6}$$

where α_{1l} and α_{1g} are the angles between the input vector $\mathbf{p}$ and the prototype vectors $\mathbf{w}_l$ and $\mathbf{w}_g$ respectively. In this case, the learning rule given in Eq. 2.5 turns into the Eq. 2.7 and 2.8 such that f is the sigmoid function for local and global winner respectively.

$$\alpha_{1l} = \alpha_{1l} - 2\lambda \frac{\partial f}{\partial \mu} \frac{\alpha_{1g}}{(\alpha_{1l} + \alpha_{1g})^2}, \tag{2.7}$$

$$\alpha_{1g} = \alpha_{1g} + 2\lambda \frac{\partial f}{\partial \mu} \frac{\alpha_{1l}}{(\alpha_{1l} + \alpha_{1g})^2} \tag{2.8}$$

Eq. 2.7 reduces the angle between the input vector $\mathbf{p}$ and the local winning prototype vector $\mathbf{w}_l$, while Eq 2.8 increases the angle between the input vector $\mathbf{p}$ and the global winning prototype vector $\mathbf{w}_g$.
It is easily appreciable that it is enough to change the angular coordinate α_1 only by means of rotation. Because, while rotating $\mathbf{w}(t)$ onto the hyperspheres, $\mathbf{w}(t+1)$ must be inside or outside of the hyper sphere.

3. Results

In this study, we apply the proposed method to four different data sets, which three of them were quoted from UCI repository[28, 32]. These datasets are Iris Flowers, Breast Cancer Diagnosis and Optical Digit Recognition. In the last experience, we have created a corpus constructed by documents quoted from Turkish news.
The corpus consists of four categories as “economy", “arts and culture", “health and fitness", and “sports" as overlapped classes. So, the corpora have totally 300 Turkish documents for training and testing operations. In preprocessing, all unnecessary characters such as punctuation marks are removed from the documents. All capital letters are converted to lower case and only one space character is allowed between two consecutive words. After the frequencies of tokens as word monogram in all documents have been computed in feature extraction stage, the documents have been represented as column vectors which contain frequencies using bag-of-words model. Kohonen layer of the network for all methods consists of 50 neurons for each class. Text categorization task is the most sophisticated experiment in our study, on the account of having the feature vectors with 400 attributes for each documents. The feature vectors are constructed using the frequencies of 100 most common high frequency words in each class. Then, we apply our method on the corpus for observing the performance of the proposed method on text categorization.
In training stage, the class cluster mean initialization is made to obtain the initial values of the weights of prototype vectors in competitive Kohonen layer. In this way, all methods start to train with the same weights for making more accurate and reliable comparisons among the methods. All methods have the same topology. Namely, the neurons in the competitive Kohonen layer are divided into N_C class having the same number of neurons, and correspond to an output layer neuron for each method. Furthermore, all networks are trained using the identical samples in same order, and also the monotonically decreasing learning rate is selected as 0.5 at the beginning of training stage to compare LVQRKV with the baseline models LVQ and other variants of LVQ. The constant stabilizing

factor ε is chosen as 10^{-3} in the methods using the stabilizer. The maximum number of epochs is selected as 300 in all experiments. The relative window width is selected as 0.2 in the methods requiring the generation of a window for updating weights.

Table 1. Performance analysis on different four datasets with the learning rate λ as 0.5 according to 10-fold cross validation.

Datasets	Method	Mean Accuracy	Mean F-measure
Iris Flower	LVQ1	0.9391 ± 0.0561	0.9070 ± 0.0884
	LVQ2	0.9342 ± 0.0572	0.8997 ± 0.0896
	LVQ2.1	0.9342 ± 0.0572	0.8997 ± 0.0896
	LVQ3	0.9350 ± 0.0559	0.9007 ± 0.0880
	OLVQ1	0.9366 ± 0.0558	0.9034 ± 0.0875
	OLVQ3	0.9333 ± 0.0564	0.8985 ± 0.0884
	GLVQ	0.9342 ± 0.0554	0.8998 ± 0.0867
	RLVQ	0.9416 ± 0.0560	0.9110 ± 0.0876
	GRLVQ	0.9235 ± 0.0772	0.8825 ± 0.1227
	LVQRKV	0.9333 ± 0.0570	0.8986 ± 0.0890
Breast Cancer	LVQ1	0.9437 ± 0.0131	0.9437 ± 0.0133
	LVQ2	0.9505 ± 0.0145	0.9504 ± 0.0145
	LVQ2.1	0.9505 ± 0.0145	0.9504 ± 0.0145
	LVQ3	0.9449 ± 0.0171	0.9448 ± 0.0173
	OLVQ1	0.9479 ± 0.0089	0.9479 ± 0.0090
	OLVQ3	0.9468 ± 0.0122	0.9467 ± 0.0123
	GLVQ	0.9491 ± 0.0131	0.9490 ± 0.0132
	RLVQ	0.9280 ± 0.0197	0.9279 ± 0.0202
	GRLVQ	0.9377 ± 0.0248	0.9376 ± 0.0254
	LVQRKV	0.9468 ± 0.0169	0.9467 ± 0.0171
Optical Digit Recognition	LVQ1	0.9840 ± 0.0103	0.9190 ± 0.0545
	LVQ2	0.9824 ± 0.0103	0.9115 ± 0.0532
	LVQ2.1	0.9824 ± 0.0103	0.9115 ± 0.0532
	LVQ3	0.9837 ± 0.0101	0.9179 ± 0.0532
	OLVQ1	0.9811 ± 0.0113	0.9059 ± 0.0546
	OLVQ3	0.9808 ± 0.0118	0.9045 ± 0.0563
	GLVQ	0.9801 ± 0.0122	0.9011 ± 0.0584
	RLVQ	0.8700 ± 0.1313	0.3101 ± 0.3111
	GRLVQ	0.9619 ± 0.0197	0.8065 ± 0.1065
	LVQRKV	0.9821 ± 0.0112	0.9103 ± 0.0560
Text Categorization	LVQ1	0.8072 ± 0.0644	0.6061 ± 0.1476
	LVQ2	0.8081 ± 0.0587	0.6061 ± 0.1293
	LVQ2.1	0.8081 ± 0.0587	0.6061 ± 0.1293
	LVQ3	0.8118 ± 0.0536	0.6151 ± 0.1216
	OLVQ1	0.7810 ± 0.1076	0.5420 ± 0.2063
	OLVQ3	0.8090 ± 0.0580	0.6102 ± 0.1262
	GLVQ	0.7917 ± 0.0682	0.5758 ± 0.1436
	RLVQ	0.8125 ± 0.0735	0.6241 ± 0.1390
	GRLVQ	0.7653 ± 0.0653	0.5240 ± 0.1329
	LVQRKV	0.8088 ± 0.0548	0.6084 ± 0.1254

Eventually, we split the original sample sets randomly into 10 equal size subsamples for applying 10-fold cross validation. In all experiments, one of single subsample set is retained as the validation data for testing, and other remaining instances are used for training the model. This process is carried out 10 times to obtain more robust result in testing. Finally, the average of accuracy and f-measure scores of all tests with standard deviation are calculated.
In this work, the datasets have been used for comparing the performance of the proposed method with some variants of learning vector quantization methods as seen in Table 1.

4. Discussion and Conclusion

In machine learning, the methods for solving classification problem vary widely. The question is which method is suitable for a specific domain. Actually, each method has own pros and cons. Recent researches focus on developing new methods including less free parameter to reach the solution of the classification problem faster and faster. In our study, we suggest a new algorithm based on geometrical approach called as LVQRKV that handles out the bottleneck of the Kohonen's generalized delta learning rule. We apply the proposed method to four different datasets, and the proposed method is compared with the baseline learning vector quantization methods on various experiments as classification of iris flower, breast cancer recognition, optical recognition of handwritten digits and text categorization. The experiments show that LVQRKV is suitable for both multidimensional and overlapped classification problems.
The obtained results from the large number of experiments verify the performance of our proposed algorithm.

Acknowledgment

We would like to acknowledge support for this project from the Council of Higher Education in Turkey (YÖK), Coordination of Academic Member Training Program (ÖYP) in Adnan Menderes University, under Grant no. ADÜ-ÖYP-12006.

References

[1] Watanabe, S. 2009. Algebraic Geometry and Statistical Learning Theory. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press.

[2] John, G. 1993. Geometry-Based Learning Algorithms.

[3] Kim, J.H., and Park, S. 1995. The geometrical learning of binary neural networks. IEEE Transactions on Neural Networks, 6(1), 237-247.

[4] Cabrelli, C., Molter, U. and Shonkwiler, R. 2000. A constructive algorithm to solve "convex recursive deletion" (CoRD) classification problems via two-layer perceptron networks. EEE Transactions on Neural Networks Learning Systems, 11(3), 811-816.

[5] Wang, D. and Chaudhari, N.S. 2004. An approach for construction of Boolean neural networks based on geometrical expansion. Neurocomputing, 57, 455-461.

[6] Shoujue, W. and Jiangliang, L. 2005. Geometrical learning, descriptive geometry, and biomimetic pattern recognition. Neurocomputing, 67, 9-28.

[7] Bayro-Corrochano, E. and Anana-Daniel, N. 2005. MIMO SVMs for classification and regression using the geometric algebra framework. Proceedings of the International Joint Conference on Neural Networks, 895–900.

[8] Zhang, D., Chan, X. and Lee, WS. 2005. Text classificitaion with kernels on the multinomial manifold. Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, 266-273

[9] Delogu, R., Fanni, A. and Montisci, A. 2008. Geometrical synthesis of MLP neural networks. Neurocomputing, 71 (4-6), 919-930.

[10] Liu, Z., Liu, J.G., Pan, C. and Wang, G. 2009. A novel geometric approach to binary classification based on scaled convex hulls. IEEE Transactions on Neural Networks, 20(7), 1215-1220.

[11] Nova, D. and Estévez, P.A. 2014. A review of learning vector quantization classifiers. Neural Computing and Applications, 25(3-4), 511-524.

[12] Kaden, M., Lange M., Nebel, D., Riedel, M. , Geweniger, T. and Villmann, T. 2014. Aspects in Classification Learning - Review of Recent Developments in Learning Vector Quantization. Foundations of Computing and Decision Sciences, 39(2), 79-105.

[13] Kohonen T. 1990. The self-organizing map. Proceedings of the IEEE, 78(9), 1464-1480.

[14] Kohonen, T.1990. Improved versions of learning vector quantization. Proceedings of the International Joint Conference on Neural Networks, 1, 545-550.

[15] Kohonen T., Hynninen, J., Kangas, J., Laaksonen, J. and Torkkola, K. 1996. LVQ_PAK: The learning vector quantization program package. Helsinki University of Technology, Laboratory of Computer and Information Science, Finland.

[16] Sato, A.S. and Yamada, K. 1995. Generalized learning vector quantization. In G. Tesauro, D. Touretzky and T. Leen (Eds.): Advances in Neural Information Processing Systems. MIT Press, 423-429.

[17] Hammer, B. and Villmann, T.2002. Generalized relevance learning vector quantization. Neural Networks - New developments in self-organizing maps, 15(8-9), 1059-1068.

[18] Seo, S. and Obermayer, K. 2003. Soft learning vector quantization. Neural Computation, 15(7), 1589-1604.

[19] Biehl, M., Ghosh,A., Hammer,B. and Bengio,Y. 2006. Dynamics and Generalization Ability of LVQ Algorithms. The Journal of Machine Learning Research, 8, 323–360.

[20] Schneider, P., Hammer,B. and Biehl, M. 2009. Adaptive relevance matrices in learning vector quantization. Neural Computation, 21, 3532-3561.

[21] Kaestner, M., Hammer, B., Biehl, M. and Villmann, T. 2012. Functional relevance learning in generalized learning vector quantization. Neurocomputing, 90, 79-105.

[22] Hammer, B., Hofmann, D., Schleif, F.M. and Zhu, X. 2014. Learning vector quantization for (dis-)similarities. Neurocomputing, 131, 43-51.

[23] Bohnsack, A., Domaschke, K., Kaden, M., Lange, M. and Villmann, T. Learning matrix quantization and relevance learning based on Schatten-p-norms. Neurocomputing, 192, 104-114.

[24] Bohnsack, A., Domaschke, K., Kaden, M., Lange, M. and Villmann, T. 2016. Learning matrix quantization and relevance learning based on Schatten-p-norms. Neurocomputing, 192, 104-114.

[25] Buchala, S., Davey, N., Gale, T.M. and Frank, R.J. 2005. Analysis of linear and nonlinear dimensionality reduction methods for gender classification of face images. International Journal of Systems Science, 36(14), 931-942.

[26] Maeda, S. and Ishii, S. 2009. Learning a multidimensional companding function for lossy source coding. Neural Networks, 22(7), 998-1010.

[27] Denil, M., Shakibi, B., Dinh, L., Ranzato, M. and Freitas, N. 2013. Predicting Parameters in Deep Learning. In CJC. Burges, L. Bottou , M. Welling , Z. Ghahramani and KQ. Weinberger (Eds.): Advances in Neural Information Processing Systems, 26, 2148–2156.

[28] Bache, K. and Lichman, M. 2014. UCI Repository of Machine Learning Databases. Irvine CA: University of California. School of Information and Computer Science. Available on:

[29] Fisher, R.A. 1936. The use of multiple measurements in taxonomic problems. Annals of Eugenics, 2, 179-188.

[30] Wolberg, W.H., Street, W.N., Heisey, D.M. and Mangasarian, O.L. 1995. Computer-derived nuclear features distinguish malignant from benign breast cytology. Human Pathology, 26, 792-796.

[31] Alpaydin, E. and Kaynak, C. 1998. Cascaded Classifiers. Kybernetika.

[32] Kaynak, C. 1995. Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition. MSc Thesis. Institute of Graduate Studies in Science and Engineering. Bogazici University.