Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 24 (2006) 649–655
image AND vision COMPUTING
www.elsevier.com/locate/imavis

A new technique for generalized learning vector quantization algorithm ☆

Zhou Shui-sheng[a,b,*], Wang Wei-wei[a], Zhou Li-hua[b]

[a] School of Science, Xidian University, No. 2, TaiBai Road, Xi'an 710071, China
[b] Multimedia Technology Institute, Xidian University, Xi'an 710071, China

Received 6 February 2004; received in revised form 5 December 2004; accepted 30 March 2005

Abstract

The disadvantage of the generalized learning vector quantization (GLVQ) and fuzzy generalization learning vector quantization (FGLVQ) algorithms is discussed in this paper. And a revised generalized learning vector quantization (RGLVQ) algorithm is proposed to overcome the disadvantage of GLVQ and FGLVQ. Furthermore, by introducing a stimulating coefficient in completing step, a new competing technique to improve the performance of the LVQ neural network is proposed also. The proposed algorithms are tested and evaluated using the IRIS data set. And the efficiency of the proposed algorithms is also illustrated by their use in codebook design for image compression based on vector quantization, and the training time for RGLVQ algorithm is reduced by 10% as compared with FGLVQ while the performance is similar. The new competing technique is also used to generate codebook and PSNR is improved in experiments.

Keywords: LVQ algorithm; Competitive network; Image compression; Codebook; Stimulating coefficient

The objective of *vector quantization* is to represent a set of vectors $\mathbf{x} \in \mathbf{X} \subset \mathbf{R}^n$ by a set of c *codevectors* $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\} \subset R^n$, also referred to as the codebook. Vector quantization is a very important technique for image compression [1–3]. Codebook design can be performed by clustering algorithms, which are typically developed by solving a constrained minimization problem using alternating optimization. These clustering techniques include the c-means algorithm, fuzzy c-means algorithm, etc. The very famous LBG algorithm [1,2] is a variation of the c-means algorithm that employs a splitting technique to compensate for the dependence of c-means algorithm on its initialization. Recent developments in neural network architecture resulted in learning vector quantization (LVQ) algorithms [4–10]. Learning vector quantization is the name used for unsupervised learning algorithms associated with competitive neural networks whose weight vectors represent the codevectors. Based on 'winner-take-all' competing strategy, T. Kohonen's [4,5] gave a learning vector quantization algorithm, named self-organizing feature map (SOFM) whose performance was greatly affected by the initial prototypes. Based on 'winner-take-most' completing strategy, Pal et al.[6] proposed a generalized learning vector quantization (GLVQ) algorithm,in which the initial prototypes have no effect on performance. The fuzzy generalized learning vector quantization (FGLVQ) algorithm based 'winner-take-most' strategy was proposed by Karayiannis etc [7–9]. Its performance is not affected by the initial prototypes also.

This paper discussed the advantage and disadvantage of GLVQ and FGLVQ in Section 1, and proposed a revised generalized learning vector quantization (RGLVQ) algorithm in Section 2. In Section 3 a new competing technique to improve the performance of the LVQ algorithms is proposed, Section 4 presents experiments testing the proposed algorithms on IRIS data set and also includes the evaluation of their performance in image compression based on vector quantization. Section 5 contains concluding remarks.

1. Analysis of LVQ algorithms

Consider the set of samples $\mathbf{X}$ from an n-dimensional Euclidean space $\mathbf{R}^n$ and let function $f(\mathbf{x})$ be the probability

☆ Supported by the Defense Pre-Research Project of the 'Tenth Five-Year-Plan' of China No. 413160501.
* Corresponding author. Address: School of Science, Xidian University, No. 2, TaiBai Road, Xi'an 710071, China. Tel.: +86 298 203 822.
E-mail address: sszhou@mail.xidian.edu.cn (Z. Shui-sheng).

0262-8856/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2005.03.005

density function of $\mathbf{x}\in\mathbf{X}\subset\mathbf{R}^n$ and let $\mathbf{V}=\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_c\}\subset R^n$ be the current codebook or clustering center, where c is clustering number. Define the loss function $L(\mathbf{x},\mathbf{V})$ as:

$$L(\mathbf{x},\mathbf{V})=\sum_{r=1}^{c}u_r\|\mathbf{x}-\mathbf{v}_r\|^2 \tag{1}$$

where $u_r=u_r(\mathbf{x})(r=1,2,\ldots,c)$is a set of weight and can also be interpreted as membership function which regulate the competition between the prototypes $\mathbf{v}_r$ for the input $\mathbf{x}$. For input $\mathbf{x}$, $\mathbf{v}_i$ satisfying $\|\mathbf{x}-\mathbf{v}_i\|^2\leq\|\mathbf{x}-\mathbf{v}_j\|^2$ $(j=1,2,\ldots,c,j\neq i)$ is the *winning prototype*. Without loss of generality, let the weight of the winning prototype be 1, and let other weights be non-negatives less than 1. The expectation of loss function $L(\mathbf{x},\mathbf{V})$ is $L(\mathbf{V})$ defined as

$$L(\mathbf{V})=\int_{\mathbf{R}^n}L(\mathbf{x},\mathbf{V})f(\mathbf{x})\mathrm{d}\mathbf{x} \tag{2}$$

LVQ algorithms are based on minimization of the expectation function $L(\mathbf{V})$ to update the clustering center $\mathbf{V}$. Minimization of (2) using gradient descent is difficult because the probability density function $f(\mathbf{x})$ is not known explicitly. Pal et al.[6] suggested the use of the gradient of the instantaneous loss function (1) to sequentially update the prototypes $\mathbf{V}$ with respect to the input vectors $\mathbf{x}\in\mathbf{X}$. 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 η_0, the iteration number T and the initial codebook $\mathbf{V}=\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_c\}$; let $t=0$;
(2) Calculate learning rate $\eta=\eta_0(1-t/T)$;
(3) For each input vector $\mathbf{x}$ and current $\mathbf{V}$:
   (a) Find the winning prototype $\mathbf{v}_i$ satisfying $\|\mathbf{x}-\mathbf{v}_i\|^2\leq\|\mathbf{x}-\mathbf{v}_j\|^2$, $\forall j\neq i$;
   (b) Calculate the updating coefficients ω_r by the gradient of $L(\mathbf{x},\mathbf{V})$, $r=1,2,\ldots c$;
   (c) Update $\mathbf{v}_r$ as $\mathbf{v}_r=\mathbf{v}_r+\eta\omega_r(\mathbf{x}-\mathbf{v}_r)$, $r=1,2,\ldots c$;
(4) If $t<T$, $t=t+1$, go to step 2;
(5) Output codebook $\mathbf{V}=\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_c\}$.

The difference between LVQ algorithms is in different weight u_r selected in Eq. (1) for non-winning prototypes to minimize the loss function.

In Kohonen's[4,5] SOFM let $u_r=u_r(\mathbf{x})=0$, $r\neq i$. The updating coefficients are $\omega_i=1$ and $\omega_r=0(r\neq i)$ and the performance is greatly affected by the initial prototypes used. While in GLVQ algorithm, proposed by Pal etc [6], $u_r=u_r(\mathbf{x})=1/D$, $r\neq i$, where $D=\sum_j\|\mathbf{x}-\mathbf{v}_j\|^2$. The updating coefficients ω_r are

$$\omega_r=\begin{cases}1-\dfrac{1}{D}+\dfrac{\|\mathbf{x}-\mathbf{v}_i\|^2}{D^2}, & r=i\\ \dfrac{\|\mathbf{x}-\mathbf{v}_i\|^2}{D^2}, & \end{cases} \tag{3}$$

Although the performance of GLVQ is not affected by the initial prototypes used, GLVQ has an obvious disadvantage: From Eq. (3) we have $\omega_i<\omega_r$ $(r\neq i)$ and even $\omega_i<0<\omega_r$ when $D<1$. It means that in iterating process the winning prototype will be moving towards input vector $\mathbf{x}$ with little step than the non-winning prototypes do, and even will be moving away from the input vector $\mathbf{x}$ while the non-winning prototypes will be always moving towards the input vector $\mathbf{x}$. This is unreasonable and it is the 'scale' problem pointed out by Gonzales et al. [10]. It means that the algorithm is very sensitive to simple scaling of the data set.

In contrast with above two, FGLVQ algorithms, proposed by Karayiannis[7–9], let $u_r=u_r(\mathbf{x})=d_{ir}p(d_{ir})$ $r\neq i$, where $d_{ir}=\|\mathbf{x}-\mathbf{v}_i\|^2/\|\mathbf{x}-\mathbf{v}_r\|^2$, and $p(z)$ is a monotonically decreasing function satisfying $0<p(z)<1/z$, $\forall_z\in(0,1)$. FGLVQ are based on the 'winner-take-most' competing strategy also, and let the weights of the non-winning prototypes $\mathbf{v}_r$ $(r\neq i)$ be different with respect to the distance between $\mathbf{v}_r$ for input vector $\mathbf{x}$. The performance of FGLVQ is also not affected by the initial prototypes used. The updating coefficients ω_r are

$$\omega_r=\begin{cases}1+\sum_{r\neq i}^{c}w(d_{ir}), & r=i\\ n(d_{ir}), & r\neq i\end{cases} \tag{4}$$

where $w(z)=p(z)+zp'(z)$, $n(z)=z^2p'(z)$ and $p'(z)$ is the derivative of $p(z)$. In Karayiannis etc [7,8], FGLVQ1, FGLVQ2 and FGLVQ3 are proposed by choosing by choosing $p(z)=(1+\alpha z)^{-1}(0<a<\infty)$, $p(z)=\exp(-\beta z)$ $(0<\beta<\infty)$or $p(z)=(1-\gamma z)(0<\gamma\leq 1)$, respectively.

According to the different $p(z)$ selected, the updating coefficient of winning prototype ω_i is affected by the term $1+\sum_{j\neq i}w(d_{ij})$, which depends on the number of the prototypes c. This dependence could make the performance of FGLVQ algorithms sensitive to the initial learning rate η_0. The algorithms will not converge when η_0 is too big and the result will hover around the initial prototyped if η_0 is too small. We must adjust η_0 to ensure the algorithms to converge properly. The experimental results in [7] indicated that: For the clustering IRIS data with 3 prototypes, the initial learning rate η_0 with best performance was near 0.5, while the vector quantization for image compression with 512 prototypes, the best initial rate was near 0.005.

From analysis above, We can conclude that the disadvantage of GLVQ and FGLVQ is because the updating coefficients are not bounded properly.