International Symposium on Signal Processing and its Applications (ISSPA), Kuala Lumpur, Malaysia, 13 – 16 August, 2001. Organized by the Dept. of Microelectronics and Computer Engineering, UTM, Malaysia and Signal Processing Research Centre, QUT, Australia

AN ONLINE LEARNING VECTOR QUANTIZATION ALGORITHM

Sunil Bharitkar Student Member, IEEE, *Dimitar Filev,* Senior Member, IEEE

ABSTRACT

We propose an online learning algorithm for the learning vector quantization (LVQ) approach in nonlinear supervised classification. The advantage of this approach is the ability of the LVQ to adjust its *codebook vectors* as new patterns become available, so as to accurately model the class representation of the patterns. Moreover this algorithm does not significantly increase the computational complexity over the original LVQ algorithm.

1. INTRODUCTION

Supervised classification approaches include, a) multilayer perceptrons (MLP) with gradient descent, and b) learning vector quantization (LVQ) methods. Applications of these methods are wide (e.g., controls, pattern recognition, fault classification signal/image processing, finance, etc.), and are well known (see [1], [2] for examples). Typically the MLP and LVQ algorithms are difficult to implement in an on-line industrial environment. This is because a new signal/pattern may not be adequately "represented" by these approaches. This necessitates the development and implementation of a simple but efficient online learning technique for these methods, which could be easily implemented without significant computational overheads.

In this paper, we develop an online learning method for the LVQ algorithm. The organization of this paper is as follows. In Section II, we propose the online LVQ approach for classification. In Section III we include an example in using the online approach for classifying two-dimensional patterns into two classes, and show that the classification rate approaches the Bayes limit. Section IV concludes the paper.

2. THE ONLINE LVQ ALGORITHM

Sunil Bharitkar is pursuing his Ph.D in the Signal and Image Processing Institute (Electrical Engineering), University of Southern California. Los Angeles. CA 90089-2564 (E-mail:bharitka@sipi.usc.edu)

Dr. D. Filev is with AMTD, Ford Motor Company, Michigan. USA. (E-mail:dfilev@ford.com)

2.1. Background

The conventional LVQ algorithm belongs to a class of signal approximation methods that model the probability density function $f(\mathbf{x})$, of some stochastic variable $\mathbf{x} \in \Re^n$, using a finite set of *codebook* vectors, $\mathbf{m}_i \in \Re^n$, $(i = 1, 2, ..., k)$ (where the subscript i could represent the hypothesis index in detection techniques). Once a set of codebook vectors are determined, the approximation of $\mathbf{x}$ implies finding the codebook $\mathbf{m}_c$ closest to $\mathbf{x}$ in the input space for a given distance metric (typically the L_p space, with $p = 1, 2, \infty$). The determination of c is achieved by the following decision process,

$$||\mathbf{x} - \mathbf{m}_c|| = min_i\{||\mathbf{x} - \mathbf{m}_i||\} \qquad (1)$$

i.e.,

$$c = arg \quad min_i\{||\mathbf{x} - \mathbf{m}_i||\} \qquad (2)$$

An optimal selection of $\mathbf{m}_i$ minimizes the average expected square of the quantization error, defined as

$$E = \int ||\mathbf{x} - \mathbf{m}_c||^2 f(\mathbf{x}) d\mathbf{x} \qquad (3)$$

Kohonen [1] discusses in detail the derivation of a recursive method to update these codebook vectors during the training phase by minimizing (3). This analytical equation has the form given below,

$$\mathbf{m}_i(k+1) = \mathbf{m}_i(k) + \alpha(k)\delta_{ci}[\mathbf{x}(k) - \mathbf{m}_i(k)] \quad (4)$$

where, δ_{ci} represents the Kronecker delta function $(\delta_{ci} = 1, c = i; 0, c \neq i)$.

A modification to (4) is the classic LVQ algorithm as a form of a reward-punishment scheme given by the following update (excitation-inhibition) of the appropriate codebook vectors,

$$\begin{aligned} \mathbf{m}_c(k+1) &= \mathbf{m}_c(k) + \alpha(k)[\mathbf{x}(k) - \mathbf{m}_c(k)] \\ & \qquad \mathbf{x}, \mathbf{m}_c \in S_c; \quad c \in [1, k] \\ \mathbf{m}_c(k+1) &= \mathbf{m}_c(k) - \alpha(k)[\mathbf{x}(k) - \mathbf{m}_c(k)] \\ & \qquad \mathbf{x} \in \overline{S_c}, \mathbf{m}_c \in S_c; \quad c \in [1, k] (5) \\ \mathbf{m}_i(k+1) &= \mathbf{m}_i(k) \qquad i \neq c \end{aligned}$$

where, $\alpha(k)$ is the learning rate, $\overline{S_c}$ is the complement set of S_c. In this paper the adjustment of the learning rate follows the Kohonen rule,

$$\alpha(k+1) = \frac{\alpha(k)}{1+\zeta\alpha(k)} \qquad (6)$$

where, $\zeta = 1$, if the classification is correct; and $\zeta = -1$, if the pattern is misclassified.

Data from the respective classes are fed sequentially to this algorithm (known as the training phase) repeatedly until some criteria is satisfied. The initial choice of the codebook vectors may be important depending on the application. The LVQ method works very well in devising the class representations by codebook vectors for a given set of patterns. These patterns form the set of codebook vectors, whereby these codebook vectors may then be considered as sufficient information for their corresponding classes.

Unfortunately, in industrial applications, large quantities of data are typically very difficult to obtain so as to train the codebook vectors, which necessitates an online methodology that will evolve the codebook vectors as more data becomes available. In the following subsection, we propose this simple, but yet effective method for online training of the LVQ.

2.2. Online LVQ Algorithm

We shall assume that a limited training set is available to form the codebook vectors. The breakdown of data availability in an industrial setting may be in two phases: (a) development phase (offline learning), where training data is available in a batch, and (b) industrial phase (online evolution), where for some *limited* duration an expert (e.g., an operator in a plant) guides the algorithm to classify the data. After this phase, the algorithm may be considered to have developed to a state where it works in an unsupervised fashion to classify an incoming pattern.

In the development phase, we train the LVQ (5) by the conventional approach where we update the codebook vectors sequentially with the training patterns, until a set number of epochs is satisfied (an epoch is the duration during which all training data is applied to the LVQ once, we shall use the words iteration and epoch interchangibly). The initial choice of the codebook vectors is governed as follows,

$$\mathbf{m}_i(0) = \mathbf{x}_i, \qquad \mathbf{x}_i \in S_i; i \in [1,k] \qquad (7)$$

Assuming that we train the LVQ for N epochs with the limited training data, we proceed to the online phase to train the LVQ. In the online implementation,whenever we recieve a pattern we add it to the existing set of training vectors and retrain the LVQ to obtain a modified set of codebook vectors . In other words we have an expanding set of training vectors which we can use for the LVQ. Unfortunately, we cannot forever expand this training set for training the LVQ due to the following reasons, a) by increasing the training set size it becomes computationally prohibitive to run the LVQ; b) this would always need the presence of an expert to apply the new pattern correctly to the LVQ; and c) by increasing the training set forever, we would end up training the LVQ by patterns that were also used for training in the development phase, thereby leading to problems associated with overtraining (poor generalization).

So as a solution to the three problems mentioned above, we anneal the number of epochs (i.e., whenever a new pattern becomes available the number of epochs for training the LVQ is reduced). However, also modify the LVQ (5) with a *gradual* update rule during the epoch due to the presence of a new pattern in the training set (i.e., we would prefer not to change the update vectors significantly to the presence of a new pattern in the training set). Eventually this pattern is retained in the training set for further training whenever a different pattern becomes available. The modified LVQ (5) update is given below,

$$\begin{aligned}
\mathbf{m}_c(k+1) &= \mathbf{m}_c(k) + \\
&\quad \alpha(k)\frac{d(\mathbf{x}(k),\mathbf{m}_c(k))}{\sum_i d(\mathbf{x}(k),\mathbf{m}_i(k))}[\mathbf{x}(k)-\mathbf{m}_c(k)]; \\
&\quad \mathbf{x}(k),\mathbf{m}_c(k) \in S_c; c \in [1,k] \\
\mathbf{m}_c(k+1) &= \mathbf{m}_c(k) - \\
&\quad \alpha(k)\frac{d(\mathbf{x}(k),\mathbf{m}_c(k))}{\sum_i d(\mathbf{x}(k),\mathbf{m}_i(k))}[\mathbf{x}(k)-\mathbf{m}_c(k)]; \\
&\quad \mathbf{x}(k) \in \overline{S_c}, \mathbf{m}_c(k) \in S_c; c \in [1,k] \\
\mathbf{m}_i(k+1) &= \mathbf{m}_i(k) \qquad i \neq c \qquad (8)
\end{aligned}$$

where $d(a,b)$ is an L_p distance metric between two points a, b. These two small changes (annealing the learning rate and modifying the codebook update) have a positive impact during online adaptation as can be seen from the results in the next section.

3. RESULTS

We shall apply the LVQ (3) and (8) for online classification of two-dimensional patterns drawn equally likely from two normal distributions (two classes, C_1, and C_2), with their individual densities described by,

$$\begin{aligned}
f(\mathbf{x}|C_1) &= \frac{1}{2\pi\sigma_1^2}\exp(-\frac{1}{2\sigma_1^2}||\mathbf{x}-\boldsymbol{\mu}_1||^2) \quad (9) \\
\boldsymbol{\mu}_1 &= (0,0)^T \quad ; \quad \sigma_1^2 = 1 \\
f(\mathbf{x}|C_2) &= \frac{1}{2\pi\sigma_2^2}\exp(-\frac{1}{2\sigma_2^2}||\mathbf{x}-\boldsymbol{\mu}_2||^2) \quad (10)
\end{aligned}$$

$$\mu_2 = (2,0)^T \quad ; \quad \sigma_2^2 = 4$$

Since the patterns are drawn equally likely from the two classes we have $P(C_1) = P(C_2) = 0.5$.

Using Bayes citeria, the optimal decision boundary between the two classes can be determined by applying the *likelihood ratio test* [3],

$$\Lambda(\boldsymbol{x}) = \frac{f(\mathbf{x}|C_2)}{f(\mathbf{x}|C_1)} \lessgtr_{C_2}^{C_1} \lambda \tag{11}$$

where, $\lambda = P(C_1)/P(C_2) = 1$. In our case, with the log-likelihood of (11), we have

$$\Lambda(\boldsymbol{x}) = \frac{\sigma_1^2}{\sigma_2^2} \exp(\frac{1}{2\sigma_1^2}||\mathbf{x}-\boldsymbol{\mu}_1||^2 - \frac{1}{2\sigma_2^2}||\mathbf{x}-\boldsymbol{\mu}_2||^2) \lessgtr_{C_2}^{C_1} 0 \tag{12}$$

On simplifying, the decision region is characterized by,

$$\begin{aligned} \mathbf{x}^T\mathbf{x} + \frac{2}{3}\mathbf{x}^T\boldsymbol{\mu}_2 &\lessgtr_{C_2}^{C_1} \frac{4}{3}(T+1) \\ T &= 4\ln\frac{\sigma_2}{\sigma_1} \end{aligned} \tag{13}$$

The misclassification of patterns can be determined by computing the average probability of error, P_e,

$$P_e = P(e|C_1)P(C_1) + P(e|C_2)P(C_2) \tag{14}$$

where,

$$\begin{aligned} P(e|C_1) &= P(\mathbf{x}^T\mathbf{x} + \frac{2}{3}\mathbf{x}^T\boldsymbol{\mu}_2 > \frac{4}{3}(T+1)| \\ & \quad \mathbf{x} = \boldsymbol{\mu}_1 + \mathbf{n}_1(u)) \approx 0.1 \\ \mathbf{n}_1(u) &\sim N_2(\mathbf{0};\sigma_1^2\mathbf{I}) \end{aligned} \tag{15}$$

and

$$\begin{aligned} P(e|C_2) &= P(\mathbf{x}^T\mathbf{x} + \frac{2}{3}\mathbf{x}^T\boldsymbol{\mu}_2 < \frac{4}{3}(T+1)| \\ & \quad \mathbf{x} = \boldsymbol{\mu}_2 + \mathbf{n}_2(u)) \approx 0.25 \\ \mathbf{n}_2(u) &\sim N_2(\mathbf{0};\sigma_2^2\mathbf{I}) \end{aligned} \tag{16}$$

We have shown the computation for $P(e|C_1)$ in the appendix. A similar method can be employed to evaluate $P(e|C_2)$. Thus

$$P(e) \approx 0.17 \tag{17}$$

The objective of this analysis is to demonstrate the the online adaptive codebook method (8) converges towards the Bayesian limit. To demonstrate this, the experimental setup was as follows,

a) Generate 100 patterns (50 from C_1, 50 from C_2), i.e., $P(C_1) = P(C_2) = 0.5$)

b) Train the LVQ (5) offline, with 10 patterns drawn from C_1 and 10 patterns from C_2. The two codebook vectors would be considered as the final class representation if we were not going to use the online LVQ .(8).

c) We draw a pattern randomly from this set of 100 patterns for a given seed, and use this pattern as *new* information to update the LVQ using (8) and anneal the epoch, so that when the next pattern is drawn the LVQ trains for a lesser duration.

d) After the LVQ codebook vectors are updated (8), we run a testing algorithm, containing the 100 patterns as test data, and compute the misclassification. This result is shown in Fig. (1), and Fig. (2). On repeated application of *new* patterns we see that the misclassification tends to the Bayesian limit.

4. CONCLUSION

In this paper we proposed an important algorithm for online pattern classification. This algorithm is particularly suitable when data is difficult to obtain (e.g., industrial and medical applications). We demonstrated the convergence of the misclassification percentage towards the Bayesian limit by updating the LVQ codebook vectors (using (8), and annealing the epoch). By adjusting the epcoh anneal rate, we may determine as to how soon we would prefer to have a final steady set of codebook vectors (which would not be updated further).

5. REFERENCES

[1] T. Kohonen , *Self-Organization and Associative Memory*, Springer-Verlag, New York. 1988

[2] S. Haykin, *Neural Networks:A Comprehesive Foundation*, Macmillan Press. 1994

[3] H. L. Van Trees, *Detection, Estimation, and Modulation Theory*, Part I, John Wiley, New York. 1968

A. APPENDIX

We shall derive (15) by using the gaussian moment factoring theorem. Now

$$P(e|C_1) = P(\mathbf{x}^T\mathbf{x} + \frac{2}{3}\mathbf{x}^T\boldsymbol{\mu}_2 > \frac{4}{3}(T+1)|\mathbf{x} = \mathbf{0}+\mathbf{n}(u)) \tag{18}$$

Let $z = \mathbf{n}^T\mathbf{n} + \frac{2}{3}\mathbf{n}^T\boldsymbol{\mu}_2 = \sum_{i=1}^{2}(n_i^2 + \frac{2}{3}n_i\mu_2(i))$. Then,

$$\begin{aligned} P(e|C_1) &= Q(\frac{4(T+1) - 3m_z}{3\sigma_z}) \\ m_z &= E\{z\} \quad \sigma_z = \sqrt{E\{(z-m_z)^2\}} \end{aligned} \tag{19}$$

$$Q(p) \;=\; \frac{1}{\sqrt{2\pi}}\int_{p}^{\infty} e^{-\frac{t^2}{2}}dt, p \geq 0$$

Now,

$$m_z = E\{\sum_{i=1}^{2} n_i^2\} + \frac{2}{3}E\{\sum_{i=1}^{2}\mu_2(i)n_i\} = 2\sigma_1^2 \quad (2$$

and

$$\begin{aligned}\sigma_z^2 &= E\{z^2\} - m_z^2 = E\{(\mathbf{n}^t\mathbf{n} + \frac{2}{3}\mathbf{n}^t\boldsymbol{\mu}_2)^T \\ &\quad (\mathbf{n}^t\mathbf{n} + \frac{2}{3}\mathbf{n}^t\boldsymbol{\mu}_2)\} - 4\sigma_1^4 \\ &= E\{(\sum_{i=1}^{2} n_i^4 + \sum_{i=1,i\neq j}^{2}\sum_{j=1}^{2} n_i^2 n_j^2)\} + \\ &\quad \frac{4}{3}E\{\sum_{i=1}^{2} n_i^2 \sum_{j=1}^{2} n_j\mu_2(j)\} \qquad (\\ &\quad + \frac{4}{9}||\boldsymbol{\mu}_2||^2\sigma_1^2 - 4\sigma_1^4 = 4\sigma_1^4 + \frac{4}{9}\sigma_1^2||\boldsymbol{\mu}_2|\end{aligned}$$

since, from the gaussian moment factoring theor we have

$$\begin{aligned}E\{\sum_{i=1}^{2} n_i^4\} &= 6\sigma_1^4 \\ E\{\sum_{i=1,i\neq j}^{2}\sum_{j=1}^{2} n_i^2 n_j^2)\} &= 2\sigma_1^4 \qquad (\\ E\{\sum_{i=1}^{2} n_i^2 \sum_{j=1}^{2} n_j\mu_2(j)\} &= 0; since, E\{n_i^3\} =\end{aligned}$$

Substituting (21) and (20) in (19) (with $\sigma_1^2 =$ and $\boldsymbol{\mu}_2 = (2,0)^T$), we obtain.

$$P(e|C_1) = 0.1037 \qquad ($$

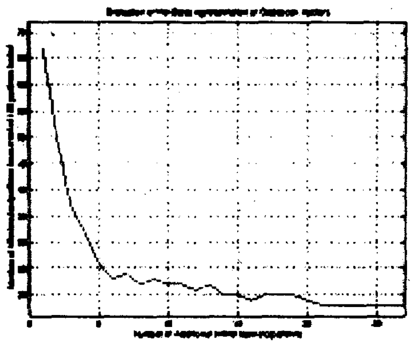

Fig 1 Online LVQ classification performance

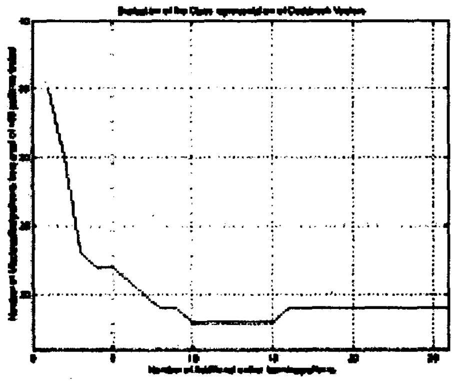

Fig. 2: Online LVQ classification performance approaches Bayes limit