A Source Coding Approach to Classification by Vector Quantization and the Principle of Minimum Description Length

Jia Li
Department of Statistics
The Pennsylvania State University
University Park, PA 16802
Email: jiali@stat.psu.edu

Abstract

An algorithm for supervised classification using vector quantization and entropy coding is presented. The classification rule is formed from a set of training data $\{(X_i, Y_i)\}_{i=1}^n$, which are independent samples from a joint distribution P_{XY}. Based on the principle of Minimum Description Length (MDL), a statistical model that approximates the distribution P_{XY} ought to enable efficient coding of X and Y. On the other hand, we expect a system that encodes (X, Y) efficiently to provide ample information on the distribution P_{XY}. This information can then be used to classify X, i.e., to predict the corresponding Y based on X. To encode both X and Y, a two-stage vector quantizer is applied to X and a Huffman code is formed for Y conditioned on each quantized value of X. The optimization of the encoder is equivalent to the design of a vector quantizer with an objective function reflecting the joint penalty of quantization error and misclassification rate. This vector quantizer provides an estimation of the conditional distribution of Y given X, which in turn yields an approximation to the Bayes classification rule. This algorithm, namely Discriminant Vector Quantization (DVQ), is compared with Learning Vector Quantization (LVQ) and CART^R on a number of data sets. DVQ outperforms the other two on several data sets. The relation between DVQ, density estimation, and regression is also discussed.

I Introduction

The goal of supervised classification is to form decision rules based on a set of training data so that misclassification penalty is minimized. Assume the training set is $\mathcal{L} = \{(x_i, y_i); i = 1, 2, ..., n\}$, where (x_i, y_i)'s are independent samples from a joint distribution P_{XY}, $X \in \mathcal{X}$, $Y \in \mathcal{Y}$. Typically X is a continuous random variable described by a pdf f_X on the d dimensional Euclidean space R^d; and Y is a discrete random variable with finitely many possible values. Let $\mathcal{Y} = \{1, 2, ..., M\}$ without loss of generality. The random variable X is normally referred to as the feature vector and Y is the class. Based on the training data, the aim is to develop classification rule $\kappa(x)$ that predicts class identities from the feature vectors of new test observations. The performance of $\kappa(x)$ is measured by the *Bayes risk*. Suppose the cost of labeling X as class m when the true class is j is $C_{j,m}$. The Bayes risk is defined as

$$B(\kappa) = \sum_{j=1}^{M} \sum_{m=1}^{M} C_{j,m} P(\kappa(X) = m \text{ and } Y = j) \ .$$

To minimize the Bayes risk, note that $B(\kappa) = E_X \sum_{j=1}^{M} P(Y = j \mid X) C_{j,\kappa(X)}$. It is thus sufficient to minimize the conditional risk $E_{Y|X=x} C_{Y,\kappa(x)}$ for each x. The optimal classification rule, referred to as the Bayes classifier, is given by

$$\kappa(x) = \min_{m \in \mathcal{Y}}{}^{-1} \sum_{j=1}^{M} P(Y = j \mid X = x) C_{j,m} \ .$$

Most frequently, $C_{j,m} = 1$, if $j \neq m$ and 0 otherwise. The resulting Bayes risk is the probability of misclassification. Correspondingly, the Bayes classifier is $\kappa(x) = \max_m{}^{-1} P(Y = m \mid X = x)$, i.e., the rule of majority vote. In this paper, Bayes risk is restricted to the probability of misclassification.

Vector quantization (VQ) has yielded efficient algorithms in statistical classification. VQ based algorithms are often referred to as prototype methods [8] in statistics because data are represented by a set of points (quantized vectors), or prototypes. Complying to the terminology of data compression, I also refer to a prototype as a codeword. Normally, a class is assigned to each prototype by majority vote on the associated class distribution of the prototype; and a test feature vector is identified as the class of its closest prototype.

The well-known k-means algorithm for classification [2, 7] is exactly the Lloyd algorithm. Based on the intuition that feature vectors in one class tend to cluster, k-means aims at minimizing the mean squared error between a feature vector and the centroid of the cluster it lies in, a goal coinciding with that of the Lloyd algorithm for data compression. K-means partitions the feature space into cells. Each cell represents one cluster with its class determined by the majority class of data it contains. The nearest neighbor classification algorithm [5, 4] can be viewed as a special case of k-means, in which the number of clusters is the number of data.

K-means was originally proposed for unsupervised classification, or clustering. Hence, prototypes are identified based solely on the intuition that data in each class tend to cluster. For supervised classification, k-means is usually inferior to other VQ based algorithms that exploit more adequately class identities provided in training. Kohonen *et al.* [10, 9] modified k-means and proposed a variety of Learning Vector Quantization (LVQ) algorithms. The basic idea is to adjust centroids obtained by k-means using class identities so that decision boundaries between classes approximate Bayes boundaries better. Starting from an initial set of centroids, LVQ predicts the class of each feature vector x_i in the training set to be the majority class of its closest cluster. If the prediction is the real class y_i, the cluster centroid is moved slightly towards x_i; otherwise, away from x_i. Implementations of various LVQ algorithms are available in the LVQ_PAK software package [11].

Another approach to incorporating class information in the design of prototypes is the Bayes Vector Quantization (BVQ) algorithm proposed by Oehler, Gray, Perlmutter, Olshen *et al.* [14, 15]. BVQ aims at achieving good compression and classification simultaneously. The objective function to minimize is a weighted summation of the mean squared error between a feature vector and its codeword and the penalty of misclassification, normally chosen to be the probability of misclassification. Although BVQ has been developed for joint compression and classification, it is of interest purely in the sense of classification. It has been found that the two factors, compression and classification, are not always in conflict; a vector quantizer that minimizes the sum of misclassification penalty and a small portion of compression distortion often results in better classification than a single purpose classifier, and vice versa [15, 13].

In this paper, vector quantization is used to estimate the joint distribution of the feature vector and the class. The algorithm, namely Discriminant Vector Quantization (DVQ), is inspired by the principle of minimum description length (MDL). MDL is proposed by Rissanen [16, 17] for model selection. When models with different complexity, or more particularly, different number of parameters are used to describe a source of data, the issue of selecting the most appropriate model cannot be solved by usual statistical principles, such as maximum likelihood and minimum prediction error. The reason is that the fitness of a model measured by likelihood or prediction error is guaranteed to improve when more parameters are contained. The MDL principle chooses the model that yields the shortest code length to reveal the data. The model itself has to be described as well, which costs bits. Although a more complex model fits data better, it is penalized on the other hand by requiring longer code length to describe. Since MDL suggests to select a model that reveals data with the shortest description length, the DVQ algorithm designs an encoding system composed of vector quantizers and entropy encoders to minimize the code length of training data. The joint distribution of the feature vector and class is inferred from the optimal encoding system, which is then used to form the Bayes classification rule.

II Estimate P_{XY} by Vector Quantization

To approximate the joint distribution P_{XY}, assume the following model:

$$P_{XY}(x,y) = \sum_{k=1}^{K} a_k \phi_k(x) b_k(y) \qquad x \in R^d, y \in \{1, 2, ..., M\} \ , \tag{1}$$

where $\sum_{k=1}^{K} a_k = 1$; $\phi_k(x)$ is a Gaussian distribution; and $b_k(y)$ is a probability mass function (pmf) of classes. The Gaussian distribution $\phi_k(x)$ is specified by mean μ_k and a common covariance matrix Σ. For simplicity, it is also assumed that Σ is diagonal, i.e., the components of a vector are independent.

The above mixture model is equivalent to a source with hidden states $S \in \{1, 2, ..., K\}$. The marginal pmf of S is given by a_k, $1 \leq k \leq K$. Conditioned on $S = k$, X and Y are independent with distributions $\phi_k(x)$ and $b_k(y)$ respectively. Constraining to X, (1) is in the form of a Gaussian mixture model (GMM). We can also view Model (1) as an extension of kernel density estimation [19] in the sense that the positions of kernel functions are to be estimated rather than fixed at the training data. Because of the very large number of mixture components usually involved in the model and the reduced complexity of the covariance structure of the components, Model (1) bears more similarity to kernel density estimation than to GMM, which will be discussed in Section V.

Although Model (1) represents a density estimation approach to classification, there is a key difference from normal density estimation used in classification. Usually, the distribution of X is estimated separately within each class. Hence, GMM or kernel density estimation is applied only to X instead of X and Y together. The joint distribution of X and Y is formed by the within-class distributions of X and a prior distribution of Y. In particular, for GMM, given the hidden state S, Y becomes deterministic and equals the class in which the hidden state is. Model (1), however, assumes that each component in the mixture generates classes according to a fixed distribution. When the samples of X in training are sparse, which often occurs especially with high dimensional X, density estimation of X within each class tends to be inaccurate and results in poor classification. By explicitly estimating the distribution of Y together with X, Model (1) has a potential advantage by forcing a classifier to use information about Y more directly. On the other hand, it retains the modeling flexibility possessed by kernel density estimation. In addition, Model (1) leads naturally to a scheme for incorporating source coding techniques in classification, which will become clear in the discussion.

By the Bayes rule, the optimal prediction of Y based on $X = x$ is

$$\max_{j \in \mathcal{Y}}{}^{-1} P\{Y = j \mid X = x\} = \max_{j \in \mathcal{Y}}{}^{-1} P\{Y = j, X = x\} = \max_{j \in \mathcal{Y}}{}^{-1} \sum_{k=1}^{K} a_k \phi_k(x) b_k(j) \ . \quad (2)$$

Equation (2) is used to classify X based on estimated a_k, ϕ_k, and b_k.

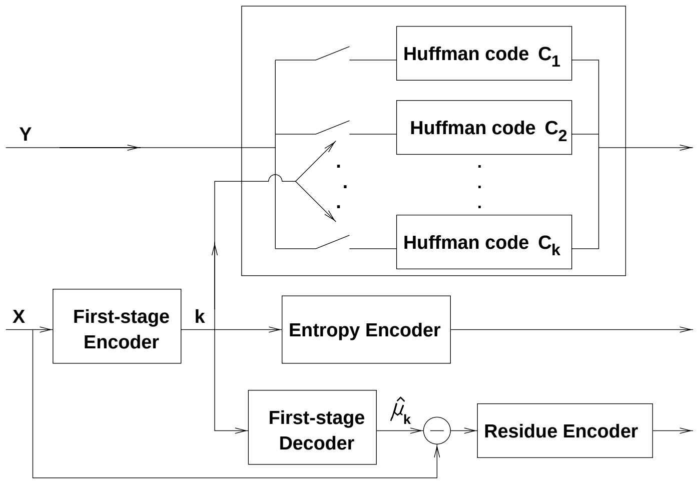

Figure 1: The encoding system for X and Y

A natural way to encode X and Y based on Model (1) is a two-stage vector quantizer for X and a classified lossless encoder for Y. In particular, the first stage VQ quantizes X to the estimated mean of each mixture component $\hat{\mu}_k$. The second stage VQ quantizes the residual between X and $\hat{\mu}_k$. To encode Y, a Huffman code is designed for each mixture component. Denote the first stage encoder and decoder of X by α and β correspondingly, the residual encoder by $\tilde{\alpha}$, the entropy encoder of indices output by α by ζ, and the lossless encoder of Y by κ. Figure 1 shows the block diagram of the encoding system.

Inspired by the principle of MDL, I propose to estimate the parameters by optimizing the encoding system, i.e., minimizing the number of bits (or nats) needed to quantize X to a certain precision and to encode Y losslessly. Instead of applying MDL to assess models that are estimated by other means, the principle is used directly to estimate a model. The approach of using vector quantization to estimate Gaussian mixture models has been investigated by Gray, Young, and Aiyer [6] from an information theoretic perspective of minimum discrimination distortion.

Assume $\alpha(x) = k$, if $x \in \mathcal{P}_k$, $k \in \{1, 2, ..., K\}$, $\bigcup_{k=1}^{K} \mathcal{P}_k = \mathcal{X}$, and $\beta(k) = \hat{\mu}_k$. Denote $P\{X \in \mathcal{P}_k\}$ by $q(k)$. Given α, the optimal entropy encoder for its output $\alpha(X)$ is the Huffman code assigning a code length $\log \frac{1}{q(k)}$ to k. The fact $\log \frac{1}{q(k)}$ may not be an integer is ignored. In the sequel, nats is used. Denote $P\{Y = m \mid X \in \mathcal{P}_k\}$ by $p(k, m)$. The optimal entropy encoder for Y given $X \in \mathcal{P}_k$ is a Huffman code assigning code length $\log 1/p(k, m)$ to class m. As shown in Figure 1, κ, the encoder for Y, contains multiple Huffman codes and selects one to encode Y according to $\alpha(X)$. To encode the residual of the first-stage quantizer, $X - \beta(\alpha(X))$ is assumed to be governed by a multivariable Gaussian distribution with independent dimensions. The variance of each dimension, denoted by D_j, $j = 1, 2, ..., d$, is the mean squared error (MSE) in that dimension achieved by the first-stage quantizer $\beta \cdot \alpha$. According to the rate distortion function of a univariable Gaussian distribution [3], on average, the number of nats needed to encode the jth dimension of the residual to a precision with MSE σ_j^2 is $\frac{1}{2} \log D_j / \sigma_j^2$. Since a Gaussian source represents the worst scenario for quantization given variance [12], $\frac{1}{2} \log D_j / \sigma_j^2$ is an upper bound on the average code length needed for the residual. To sum up, the average code length to encode $\{(X_i, Y_i)\}_{i=1}^{n}$ so that the jth dimension of X is quantized with distortion σ_j^2 and Y is encoded losslessly is,

$$nH(q) + \frac{n}{2} \sum_{j=1}^{d} \log \frac{D_j}{\sigma_j^2} + n \sum_{k=1}^{K} q(k) H(p(k)) \ ,$$

where $H(q)$ is the entropy of the pmf specified by $q(k)$, $k = 1, 2, ..., K$; and $H(p(k))$ is the entropy of the pmf specified by $p(k, m)$, $m = 1, 2, ..., M$, k fixed. The code length of the training sequence $\{(x_i, y_i)\}_{i=1}^{n}$ is

$$\sum_{i=1}^{n} \log \frac{1}{q(\alpha(x_i))} + \frac{n}{2} \sum_{j=1}^{d} \log D_j - \frac{n}{2} \sum_{j=1}^{d} \log \sigma_j^2 + \sum_{i=1}^{n} \log \frac{1}{p(\alpha(x_i), y_i)} \ , \qquad (3)$$

where $D_j = \frac{1}{n} \sum_{i=1}^{n} (x_i^{(j)} - \hat{x}_i^{(j)})^2$, $x^{(j)}$ denotes the jth dimension in the vector x; and $\hat{x} = \beta(\alpha(x))$ is the quantized value of x by the first-stage encoder and decoder.

The goal is to design α, β, κ, and ζ to minimize the objective function

$$L(\alpha, \beta, \kappa, \zeta) = \sum_{i=1}^{n} \log \frac{1}{q(\alpha(x_i))} + \frac{n}{2} \sum_{j=1}^{d} \log D_j + \sum_{i=1}^{n} \log \frac{1}{p(\alpha(x_i), y_i)} \ , \qquad (4)$$

obtained from (3) by omitting the third term, which depends only on the preselected precision of X. The third term in (4), with expected value $n \sum_{k=1}^{K} q(k) H(p(k))$, reflects the penalty on misclassification. If $H(p(k))$ is low, the quantization cell $\mathcal{P}_k$ is "pure" in the sense that a certain class dominates. Hence, the probability of misclassification is low. The second term in (4) represents the cost resulted from compression. The first term in (4) depends on the entropy of indices output by α. It represents to a certain extent the complexity of the quantized values. In summary, MDL leads to the design of a vector quantizer with an objective function incorporating compression distortion, the penalty of misclassification, and the complexity of quantized feature vectors.

After the encoding system is designed, the parameters in Model (1) are estimated by $\hat{a}_k = q(k)$, $\hat{b}_k(m) = p(k,m)$, $\hat{\mu}_k = \beta(k)$, and $\hat{\Sigma} = diag(D_1, D_2, ..., D_l)$. To classify a new test sample with only X specified, Equation (2) is used with the estimated parameters.

III Algorithm

It is obvious to see that for a given α, the optimal decoder sets $\beta(k)$ as the centroid of all x_i's such that $\alpha(x_i) = k$. The optimal entropy encoder ζ uses the Huffman code determined by the pmf $q(k)$, $k = 1, 2, ..., K$. The classified entropy encoder κ contains multiple Huffman codes determined by the pmfs $p(k, m)$, $m = 1, 2, ..., M$. For fixed β, ζ, and κ, α is not a nearest neighbor encoder, which is the main difficulty in the design. A recursive descending algorithm is used to iteratively update the four components in the system. The procedure to update α with fixed β, ζ, and κ is presented first. The update of β, ζ, and κ with fixed α is straightforward. As ζ and κ contain Huffman codes derived from pmfs $q(\cdot)$ and $p(\cdot,\cdot)$, the formula for updating $q(\cdot)$ and $p(\cdot,\cdot)$ are provided instead.

As the training data set is finite, searching for the optimal α is equivalent to determining n integer values $\alpha(x_i)$, $i = 1, ..., n$, where n is the size of the training data set and $\alpha(x_i) \in \{1, ..., K\}$. Denote α at the beginning of iteration t by $\alpha^{(t)}$. Let $\hat{x}_i = \beta(\alpha^{(t)}(x_i))$. The algorithm updating $\alpha^{(t)}(x_i)$ to $\alpha^{(t+1)}(x_i)$ is as follows:

1. $1 \rightarrow i$; $\quad \frac{1}{n}\sum_{l=1}^{n}(x_l^{(j)} - \hat{x}_l^{(j)})^2 \rightarrow D_j$, $j = 1, 2, ..., d$.

2. $\min_k{}^{-1}\left[\log\frac{1}{q(k)} + \frac{n}{2}\sum_{j=1}^{d}\log(D_j + \frac{(x_i^{(j)} - \beta(k)^{(j)})^2 - (x_i^{(j)} - \hat{x}_i^{(j)})^2}{n}) + \log\frac{1}{p(k,y_i)}\right]$

   $\rightarrow \alpha^{(t+1)}(x_i)$

3. $D_j + \frac{(x_i^{(j)} - \beta(\alpha^{(t+1)}(x_i))^{(j)})^2 - (x_i^{(j)} - \hat{x}_i^{(j)})^2}{n} \rightarrow D_j$, $j = 1, 2, ..., d$.

4. $\beta(\alpha^{(t+1)}(x_i)) \rightarrow \hat{x}_i$

5. $i + 1 \rightarrow i$. If $i > n$, stop; otherwise, go back to step 3.

After $\alpha^{(t+1)}$ is obtained, β, $q(\cdot)$, and $p(\cdot,\cdot)$ are updated by the following formula

$$\beta(k) = \frac{\sum_{i=1}^{n} x_i I(\alpha^{(t+1)}(x_i) = k)}{\sum_{i=1}^{n} I(\alpha^{(t+1)}(x_i) = k)}\ , \qquad q(k) = \sum_{i=1}^{n} I(\alpha^{(t+1)}(x_i) = k)/n\ ,$$

$$p(k,m) = \frac{\sum_{i=1}^{n} I(\alpha^{(t+1)}(x_i) = k) I(y_i = m) + 1}{\sum_{i=1}^{n} I(\alpha^{(t+1)}(x_i) = k) + M} ,$$

where $k = 1, 2, ..., K$ and $m = 1, 2, ..., M$. As usual $I(\cdot)$ is the indicator function that equals 1 if the argument is true and 0 otherwise. Note that to avoid classes with zero frequency, $p(k,m)$ is computed by a modified version of the normal frequency estimation.

IV Experiment

To test DVQ, 7 data sets listed below were examined. These data sets were taken from the UCI machine learning database repository [1] at:
`http://www.ics.uci.edu/~mlearn/Machine-Learning.html`.

1. Liver Disorder: This data set is used for detecting the presence of liver disorder based on 6 features, five of which are blood tests and one is the quantity of alcoholic beverages consumed per day. There are 345 cases in this data set.

2. Heart Disease: This data set concerns heart disease diagnosis. There are two classes: presence of heart disease versus absence of heart disease. The number of features is 13. The original data set contains 303 cases, 6 of which have missing data. These 6 cases are excluded in the study here.

3. Diabetes: This data set concerns whether a patient shows signs of diabetes according to the World Health Organization criteria. The number of features is 8. There are 768 cases in the data set.

4. Glass: This data set concerns the classification of 6 types of glasses based on 9 features. There are 214 cases in the data set.

5. Iris: This data set is used to distinguish 3 types of iris plants. The number of features is 4. There are 150 cases in the data set.

6. Ionosphere: This data set concerns the classification of radar returns from the ionosphere. There are 2 classes and the number of features is 32. The number of cases in the data set is 351.

7. Sonar: This data set is for distinguishing mines and rocks based on sonar signals. For each case, there are 60 numbers that range from 0.0 to 1.0 recorded. The data set contains 208 cases.

Table 1 presents the misclassification rates achieved by CART, LVQ, and DVQ. All the misclassification rates are obtained by 10-fold cross-validation. The numbers of mixture components, or codewords, used by both DVQ and LVQ are selected by 10-fold cross-validation using only the training data in each fold. The LVQ_PAK software was used to perform LVQ. Codebook initialization in LVQ was completed

Algorithm	Liver	Heart	Diabetes	Glass	Iris	Ionosphere	Sonar
CART	35.7%	20.9%	28.1%	33.5%	2.7%	6.9%	24.1%
LVQ	33.0%	36.0%	26.7%	32.3%	2.7%	13.3%	19.7%
DVQ	32.1%	15.5%	25.7%	44.1%	2.7%	12.8%	15.8%

Table 1: Comparison of the probability of misclassification.

by the programs *eveninit* and *balance*. Codebooks were trained using the optimized-learning-rate LVQ1, implemented in program *olvq1*. This version of LVQ is suggested as the most robust of all the LVQ algorithms [11]. For each application, the number of iterations is set to be 50 times the size of the training data set, which is sufficient as discussed in [11].

From Table 1, it is interesting to note that both DVQ and CART perform significantly better than the others on certain data sets. For instance, DVQ yields substantially lower error rates with the Sonar and Heart data sets. CART outperforms greatly the other two with the Ionosphere data set. The large difference in error rates with different data sets indicates the algorithms are complementary to each other to a certain extent. The performance of LVQ is between that of CART and DVQ for 5 out of the 7 data sets. For the other 2 data sets, it is the worst. DVQ performs much worse than the others with the Glass data set. However, it achieves the lowest error rate for 5 data sets.

V Discussion

In Section II, the mixture model $P_{XY}(x, y) = \sum_{k=1}^{K} a_k \phi_k(x) b_k(y)$, $x \in R^d$, $y \in \{1, 2, ..., M\}$, is proposed. The role of the Gaussian components in the mixture is more of serving as kernels than as "modes" in a normal Gaussian mixture model. The number of components is often very large, comparable to the number of training data. For example, the sonar data set has about 187 training data in each fold of cross-validation; and the average number of mixture components chosen by cross-validation is 150. During the design of the encoding system, a component is deleted automatically if it yields an empty quantization cell. For the sonar data set, the average number of components after the design is about 84. The focus of DVQ is to determine the positions of the Gaussian components. As in kernel density estimation [19], the covariance matrices of the Gaussian distributions are assumed to be of a common and simple structure, in particular, independent dimensions.

In kernel density estimation, a distribution is estimated as a summation of kernels centered at each training data point. In DVQ, the positions of the Gaussian kernels and the kernel widths are derived by optimizing an encoding system. It is unnecessary to constrain to Gaussian kernels in Model (1). In fact, if the domain of X is bounded, the residual $X - \beta(\alpha(X))$ ought to follow a distribution with finite support. Suppose a different kernel function is used in (1), we may design the encoding system in the same manner and fit the kernel function to the residual if the kernel cannot be determined completely by its variance and expectation. We can view DVQ as an application of MDL to nonparametric density estimation. Estimating a large number of parameters by exploiting a certain coding scheme under the spirit of MDL has not become a widespread methodology. An early work in this direction is an algorithm on the optimization of histogram estimators with variable bin widths developed by Rissanen, Speed, and Yu [18].

Although DVQ is proposed for classification, it is easily extendible to regression. If Y is a continuous random variable rather than a discrete class label, we can replace the pmf $p_k(y)$ in Model (1) by a kernel function and consider a two-stage encoder for Y. The objective function in the design of the encoding system will reflect the tradeoff between distortion and rate, or equivalently, prediction error and model complexity from a statistical point of view.

VI Conclusion

In this paper, a novel classification algorithm, namely Discriminant Vector Quantization, has been developed. Based on the principle of Minimum Description Length, this algorithm estimates the joint distribution of the feature vector and the class by optimizing an encoding system. The DVQ algorithm has been tested and compared with CART and LVQ on several data sets. Experiments demonstrated promising results. The relation between DVQ and kernel density estimation, and the possible extension of DVQ to regression have also been discussed.

Acknowledgment

The author would like to thank the reviewers for their very helpful comments.

References

[1] C. Blake, E. Keogh, and C. J. Merz, "UCI repository of machine learning databases," `http://www.ics.uci.edu/~mlearn/MLRepository.html`, Department of Information and Computer Science, University of California, Irvine, 1998.

[2] R. K. Blashfield and M. S. Aldenderfer, "The literature on cluster analysis," *Multivariate Behavioral Research*, vol. 13, pp. 271-295, 1978.

[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., 1991.

[4] B. Dasarathy, *Nearest Neighbor Pattern Classification Techniques*, IEEE Computer Society Press, 1991.

[5] E. Fix and J. Hodges, "Discriminatory analysis—nonparametric discrimination: Consistency properties," *Technical Report 21-49-004, 4*, US Air Force, School of Aviation Medicine, Randolph Field, TX.

[6] R. M. Gray, J. C. Young, and A. K. Aiyer, “Minimum discrimination information clustering: modeling and quantization with Gauss mixtures,” *Proc. Int. Conf. Image Processing (ICIP)*, Thessaloniki, Greece, October 2001.

[7] J. A. Hartigan and M. A. Wong, “Algorithm AS136: a k-means clustering algorithm,” *Applied Statistics*, vol. 28, pp. 100-108, 1979.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning—Data Mining, Inference, and Prediction*, Springer, 2001.

[9] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, 1989.

[10] T. Kohonen, G. Barna, and R. Chrisley, “Statistical pattern recognition with Neural Networks: benchmarking studies,” *IEEE Int. Conf. Neural Networks*, pp. I-61-68, July 1988.

[11] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola, “LVQ_PAK: The learning vector quantization program package (version 3.1),” Technical Report, Helsinki University of Technology, Laboratory of Computer and Information Science, Finland, April, 1995. Available via anonymous ftp to cochlea.hut.fi.

[12] A. Lapidoth, ”On the role of mismatch in rate distortion theory,” *IEEE Trans. Inform. Theory*, vol. 43, pp. 38-47, Jan. 1997.

[13] J. Li, R. M. Gray, and R. A. Olshen, “Joint image compression and classification with vector quantization and a two dimensional hidden Markov model,” *Data Compression Conference (DCC)*, pp. 23-32, Snowbird, Utah, March 1999.

[14] K. L. Oehler and R. M. Gray, “Combining image compression and classification using vector quantization,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 461-473, May 1995.

[15] K. O. Perlmutter, S. M. Perlmutter, R. M. Gray, R. A. Olshen, and K. L. Oehler, “Bayes risk weighted vector quantization with posterior estimation for image compression and classification,” *IEEE Trans. Image Processing*, vol. 5, no. 2, pp. 347-360, Feb. 1996.

[16] J. Rissanen, “Modeling by shortest data description,” *Automatica*, vol. 14, pp. 465-471, 1978.

[17] J. Rissanen, “Stochastic complexity and modeling,” *The Annals of Statistics*, vol. 14, no. 3, pp. 1080-1100, 1986.

[18] J. Rissanen, T. P. Speed, and B. Yu, “Density estimation by stochastic complexity,” *IEEE Trans. Inform. Theory*, vol. 38, no.2, pp. 315-323, 1992.

[19] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, 1986.