BAYES RISK WEIGHTED VECTOR QUANTIZATION WITH CART ESTIMATED CLASS POSTERIORS

Keren O. Perlmutter *Robert M. Gray* *Richard A. Olshen*† *Sharon M. Perlmutter*

Information Systems Lab, Dept. of Electrical Engineering
Stanford University, Stanford, CA 94305-4055
†Division of Biostatistics, Dept. of Health Research and Policy
Stanford University, Stanford, CA 94305-5092

ABSTRACT

A Bayes risk weighted vector quantizer (Bayes VQ) combines compression and low-level classification of images by incorporating a Bayes risk component into the distortion measure used to design the code. The class posterior probabilities required for the Bayes risk computation can be estimated based on a labeled training sequence. We here introduce two new methods for estimating these posteriors. In particular, two types of tree-structured estimators are constructed by applying the classification and regression tree algorithm CART^{TM} to eight features of the training sequence. We apply the resulting Bayes VQ systems to aerial photographs where the goal is to compress the images and classify man-made and natural regions. These systems provide classification superior to that of previous work with Bayes VQ while maintaining similar compression performance. The systems also provide moderate to substantial improvement in classification with only a small loss in compression to performance obtained with a modified version of Kohonen's "learning vector quantizer" and with an independent design of quantizer and classifier.

1. INTRODUCTION

Compression and classification play important roles in communicating and interpreting digital images. With multispectral or aerial imagery, for example, compression is useful for storage and transmission, whereas classification can enable the simultaneous or subsequent segmentation of these images into different categories of terrain.

The general setup for the problem of compression and classification consists of a joint random process $\{X(n), Y(n) : n = 0, 1, \ldots\}$, where the $X(n)$ are k-dimensional real-valued vectors and the $Y(n)$ designate membership in a class and take values in a set $\mathcal{H} = \{0, 1, \cdots, M-1\}$. A VQ that provides both compression and classification operates on the observed sequence X and consists of three functions: an encoder α that views only X and outputs a binary index $i = \alpha(x)$, a decoder β that maps the indices into the reproduction vectors $\beta(i) = \hat{X}_i$, and a classifier δ that associates a class label $\delta(i) \in \mathcal{H}$ for every encoder output index $i = 1, \ldots, N$. Because the index i can be used simultaneously to decompress and to classify the vector, the classification is implicit in the compression, and hence does not cost additional computation or bits once the image is compressed.

The quality of the reproduction $\hat{X} = \beta(\alpha(X))$ for an input X is measured by a nonnegative distortion, $d(X, \hat{X})$. For simplicity, we here consider the squared error distortion $d(X, \hat{X}) = ||X - \hat{X}||^2$. The average distortion

$$D(\alpha, \beta) = E[d(X, \beta(\alpha(X)))] \tag{1}$$

is then the mean squared error (MSE). The quality of the classifier is measured by the Bayes risk,

$$\begin{aligned} B(\alpha, \delta) &= \sum_{k=0}^{M-1} \sum_{j=0}^{M-1} C_{jk} P(\delta(\alpha(X)) = k \text{ and } Y = j) \\ &= \sum_{i=0}^{N-1} P(\alpha(X) = i) \sum_{k=0}^{M-1} 1(\delta(i) = k) \\ &\quad \times \sum_{j=0}^{M-1} C_{jk} P(Y = j | \alpha(X) = i), \end{aligned} \tag{2}$$

where the indicator function 1 (expression) is 1 if the expression is true and 0 otherwise. The cost C_{jk} represents the cost incurred when a class j vector is classified as class k. We assume $C_{jk} = 0$ when $j = k$.

A number of vector quantization algorithms have been developed to address the compression or classification problem. Few of these methods, however, jointly optimize these two signal processing techniques. For example, nearest neighbor schemes, including the full search "learning vector quantizer" (LVQ) [1], implicitly design the quantizer for classification rather than compression. Other techniques focus on the separate and independent design of quantizer and classifier [2, 3]. An independent design might first involve the design of the quantizer using the generalized Lloyd algorithm, a descent algorithm that alternately optimizes the encoder and decoder. A Bayes classifier, defined by $\arg\min_k \sum_{j=0}^{M-1} C_{jk} \times \Pr(Y = j|\alpha(x))$, would then be designed for the VQ output $\alpha(X)$.

Bayes risk weighted vector quantization [3] jointly optimizes for compression and classification by incorporating

This work was supported by the National Science Foundation under grant NSF MIP-9016974.

a Bayes risk component into the distortion measure used to design the quantizer, thereby enabling simultaneous optimization of a code with respect to both compression and classification. Bayes VQ with posterior estimation [4, 5] consists of a VQ preceded by a posterior estimator. The VQ (either full search or tree-structured) uses a modified distortion measure in the design of the compression code that allows simultaneous optimization for both compression (using squared error for general appearance) and Bayes risk (for classification accuracy) by combining the two terms with a Lagrangian importance weighting. The modified distortion measure is

$$\rho_{\lambda,\hat{P}}(x,\hat{x},l) = ||x-\hat{x}||^2 + \lambda \sum_{j=0}^{M-1} C_{j,l}\hat{P}(Y=l|x).$$

The fidelity criterion is thus $J_{\lambda,\hat{P}}(\alpha,\beta,\delta) = E[\rho_{\lambda,\hat{P}}] = D(\alpha,\beta) + \lambda B(\alpha,\delta)$, where D and B are defined in (1) and (2), respectively. This weighted combination allows trade-offs between priorities for compression and classification. The encoder selects the nearest neighbor with respect to the modified distortion measure to determine the best codeword representative. As with the generalized Lloyd algorithm, we use a descent algorithm that in turn optimizes the encoder, decoder, and classifier. In the tree-structured design we consider in this study, the trees are grown by choosing the node that yields the largest ratio of decrease in average (modified) distortion to increase in bit rate, and then are pruned [6] in order to obtain optimal subtrees. We note that the independent design of quantizer and classifier can be considered as the special case of the Bayes VQ design with $\lambda = 0$.

Both the design and implementation of the Bayes VQ require knowledge of the posterior probabilities. However, these posteriors are not required by the decoder; and thus no additional bits need be sent to transmit the information. In the nonparametric case we must obtain an estimate $\hat{P}$ based on the empirical class distribution of the learning set $P_{\mathcal{L}}(l|x)$. We want to use an estimator that is computationally simple, and thus we do not consider kernel estimation or projection pursuit techniques here. In [4, 5, 7], a computationally simple posterior estimator was described and implemented into the Bayes VQ. The probabilities were provided by a tree-structured vector quantizer (TSVQ) that was designed on the raw pixel intensity vectors and that was grown by splitting nodes that contributed most to the relative entropy distortion. We will refer to the tree-structured version of that system as Bayes TSVQ with relative entropy based posterior estimation (BTSVQ with RE p.e.). That Bayes VQ system provided good performance compared to that of the independent design and to that of a modified version of LVQ [5], but improved posterior estimation holds promise for better classification [7]. We here present two improved posterior estimation techniques that use eight features extracted from the learning set. In particular, we use the classification and regression tree algorithm CART [8] to construct both a class probability tree and a classification tree.

2. POSTERIOR ESTIMATION

We design two different estimators using feature vectors that are extracted from the learning set to form the posterior estimates used during the Bayes VQ design and encoding. One estimator is a class probability tree and the other is a classification tree. These two estimators differ only in the pruning methods used. The trees allow a number of candidate features and the selection of the best among these upon which to split the data.

Assume a tree T has the set of terminal nodes $\tilde{T}$, so that $|\tilde{T}|$ denotes the number of terminal nodes in T. Let L denote the training set for a J-class problem. We associate with each terminal node t the estimates $p(j|t)$, $j = 1, ..., J$, for the conditional probability of being in class j given node t. Then for any vector $x \in t$, the estimator d would provide $d(x) = (p(1|t), \ldots, p(J|t))$. We also divide the training set L into two sets L_1 and L_2, and use L_1 to construct the estimator d (or equivalently T), and use L_2 to obtain the test sample estimate $R^{ts}(T)$ of the risk of this estimator. The test sample estimate $R^{ts}(T)$ for the class probability tree is formed as $R^{ts}(T) = \sum_{t\in\tilde{T}}\sum_i (z_{n,i} - d(i|x_n))^2 \times p(t)$, where for each intensity vector/class label pair (x_n, j_n), we define J values $\{z_{n,i}\}$ as $z_{n,i} = 1$ if $j_n = i$, 0 otherwise. It is shown in [8] that this reduces to $R^{ts}(T) = \sum_{t\in\tilde{T}}(1 - \sum_j p^2(j|t)) \times p(t)$, where $1 - \sum_j p^2(j|t)$ is defined as the Gini index of diversity. Thus the class probability tree is grown by splitting the node that most reduces the Gini index, as this continually minimizes the estimate $R^{ts}(T)$ for the mean squared error $\sum_i (z_{n,i} - d(i|x_n))^2$. The class probability tree is then pruned upward using the criterion $R^{ts}(T) + \alpha|\tilde{T}|$, where α is a complexity parameter, and where $R^{ts}(T) = \sum_{t\in\tilde{T}} r^{ts}(t)p(t)$ and $r^{ts}(t)$ is the within node Gini index.

The second estimator we design is a classification tree that is also grown by splitting the node that most reduces the Gini index. However, this tree is pruned upward using $r^{ts}(t)$ as the within-node misclassification cost. The estimates $p(j|t), j = 1, \ldots J$, for a given vector x are obtained in a manner analogous to that used for the class probability tree.

The trees are constructed using vectors consisting of eight features that are extracted from the vectors in the spatial domain. Given a vector $X = (X_1, \ldots, X_k)^T$ with class membership $Y \in \{1, 2\}$ (two-class problem), define $\hat{u_i}$ as the mean vector and Σ_i as the covariance matrix for the class i features. F_X is the 7×1 vector of features that are computed from X and that are listed as 1 through 7 below. We consider the following eight features:

1. $\min_X = \min_i X_i, i = 1, \ldots, k$
2. $\max_X = \max_i X_i, i = 1, \ldots, k$
3. $m_X = \frac{1}{k}\sum_{i=1}^{k} X_i$
4. $\sigma_X = \sqrt{\sum_{i=1}^{k}(X_i - m_X)^2}$
5. $S_X = m_X/\sigma_X$
6. $R_X = \max_X - \min_X$
7. $\text{med}_X = \text{median}(X)$
8. $LD_X = F_X^T(\frac{n_1\Sigma_1 + n_2\Sigma_2}{n_1+n_2-2})^{-1}(\hat{u_1} - \hat{u_2})$, i.e. we compute a two-sample linear discriminant (LD) score for use as a predictor using the previously described seven features as components of the LD (even though these features are not Gaussian) in addition to the predictors from which it was computed.

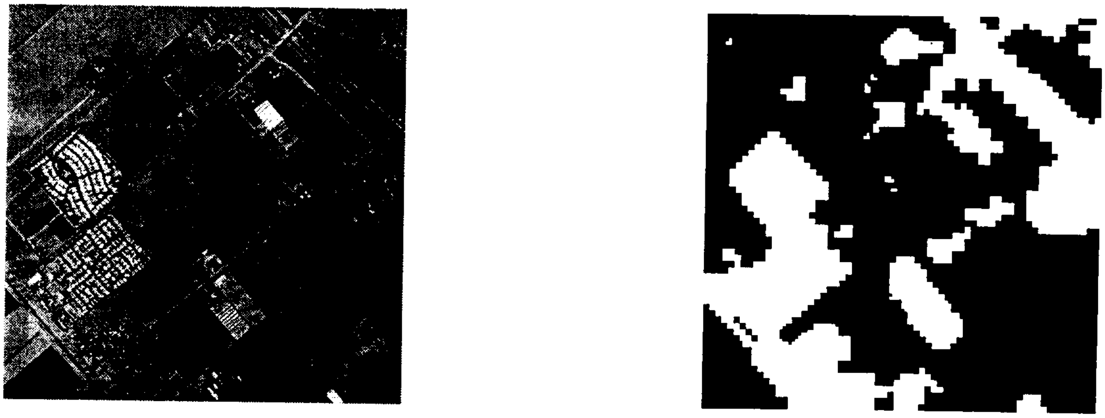

Figure 1: (a) Original 8 bpp aerial image and (b) original class labeled image

3. RESULTS

The goal was to compress an 8 bit per pixel (bpp) 512×512 aerial image and to identify regions as either man-made or natural. Figure 1 presents a typical test image and the hand-labeled classification for this image. In the classified image, man-made regions are indicated in white whereas natural regions are indicated in black.

The codebook design and image encoding were performed using 4×4 pixel blocks. Simulations were performed using six-fold cross-validation [8]. We use PSNR as the measure for compression error, where PSNR is defined as $10\log_{10}(255^2/\mathrm{MSE})$. The quality of the classifier is measured by the empirical Bayes risk given by (2). Equal costs were assigned to the misclassification errors. The Bayes risk thus signifies the fraction of the total vectors that are misclassified. On average, the images consisted of $43.6 \pm 24.6\%$ natural vectors and $56.4 \pm 24.6\%$ man-made vectors.

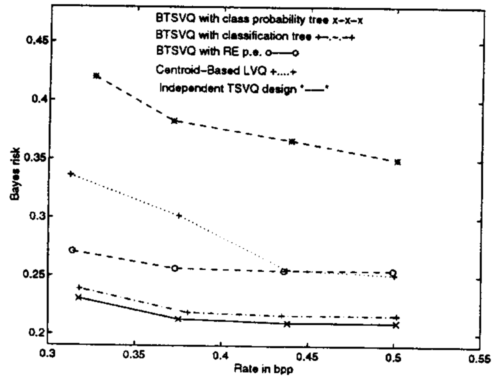

Figure 2: Effect of bit rate on Bayes risk

We compare the proposed Bayes VQ systems with the Bayes VQ with RE p.e. system of [5], an independent TSVQ design, and a modified version of LVQ. The Bayes VQ designs used $\lambda = 10^6$. Both the class probability trees and classification trees were considerably less complex than the corresponding relative entropy based posterior estimating TSVQ. For the LVQ design, the codebook was initialized using the LVQ_PAK *eveninit* algorithm and then designed using the optimized learning rate LVQ1 method, *olvq1* [9]. A modification of the resulting codebook (as described in [5]) was imposed to improve the compression performance. We will denote this modified version of LVQ as centroid-based LVQ. In simulations, the number of iterations used in the algorithm was equal to five times the number of training vectors.

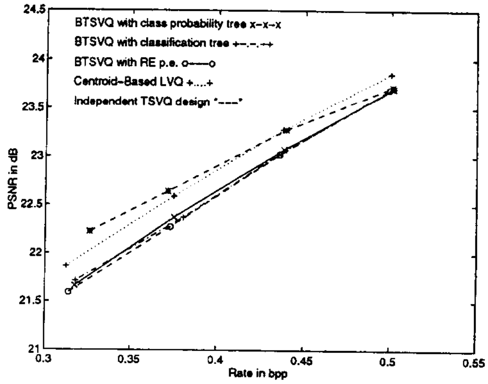

Figure 3: Effect of bit rate on PSNR

Figures 2 and 3 illustrate the classification and compression performance obtained with the BTSVQ with class probability tree system, the BTSVQ with classification tree system, the BTSVQ with RE p.e. system of [5], the independent TSVQ design, and centroid-based LVQ. Figure 4(a) illustrates a test image compressed using the BTSVQ with class probability tree system, and Figure 4(b) gives the corresponding classification obtained with the system. Both the BTSVQ with class probability tree and the BTSVQ with classification tree systems outperform the other three methods in classification at all bit rates. The proposed systems provide up to 18% relative gain (that is, with respect to the percentage of possible improvement) over the BTSVQ with RE p.e. system. The system using the class probability tree slightly outperformed the system using the classification tree. This may be an artifact of the particular training sets we consider or may be something more sig-

Figure 4: (a) Compressed image and (b) classified image using Bayes TSVQ with class probability tree at 0.5 bpp

nificant. In regards to the latter, it may be related to the implicit bias-variance tradeoffs of the two techniques (as the classification tree approach produces a greater bias in estimating densities than the class probability tree approach). BTSVQ with RE p.e. provides the next best classification after the two proposed techniques, followed by centroid-based LVQ and then the independent TSVQ design. Recall that LVQ uses full search while all the competitors use tree-structured searches. The two approaches with BTSVQ and CART estimated posteriors also provide compression similar to that of BTSVQ with RE p.e. Although the independent design outperforms the Bayes TSVQ methods in compression at the lower rates, the former's classification performance is quite poor. For instance, the BTSVQ with class probability tree system offers up to 19% absolute gain (or 45% relative gain) in classification performance over the independent design for a penalization of less than 0.5 dB in compression. In addition, the two BTSVQ techniques with CART estimated posteriors outperformed centroid-based LVQ in classification (substantially so at the lower rates) for only a slight loss in compression. For example, at the lower bit rates, the BTSVQ with class probability tree provides up to 11% absolute gain (or 33% relative gain) in classification performance over centroid-based LVQ for a penalization of 0.2 dB. Without the centroid-based modification, LVQ provides slightly less compression performance than the BTSVQ systems with CART estimated posteriors at the lower bit rates. In addition, the Bayes VQ systems have the ability to trade off some of the substantial classification improvement for some improvement in compression.

4. CONCLUSIONS

We proposed two techniques to estimate the posteriors required during Bayes VQ design and encoding. Each method consisted of a tree-structured estimator that was constructed by applying CART to eight features extracted from the training set. The resulting systems outperformed the Bayes VQ of [5] in classification, while maintaining similar compression. For only a small loss in compression, the proposed systems provided considerable improvement in classification to centroid-based LVQ and to the independent design.

5. REFERENCES

[1] T. Kohonen, "An introduction to neural computing," *Neural Networks*, vol. 1, pp. 3–16, 1988.

[2] E. E. Hilbert, "Cluster compression algorithm: a joint clustering/data compression concept," Publication 77-43, Jet Propulsion Lab, Pasadena, CA, Dec. 1977.

[3] K. L. Oehler, *Image Compression and Classification using Vector Quantization.* Ph.D. Dissertation, Stanford University Electrical Engineering Department, September 1993.

[4] K. O. Perlmutter, R. M. Gray, K. L. Oehler, and R. A. Olshen, "Bayes risk weighted tree-structured vector quantization with posterior estimation," in *Proceedings of the 1994 IEEE Data Compression Conference (DCC)* (J. A. Storer and M. Cohn, eds.), (Snowbird, Utah), pp. 274–283, IEEE Computer Society Press, March 1994.

[5] K. O. Perlmutter, C. L. Nash, and R. M. Gray, "A comparison of bayes risk weighted vector quantization with posterior estimation with other VQ-based classifiers," in *Proceedings of the International Conference on Image Processing*, vol. 2, pp. 217–221, Nov. 1994.

[6] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. Inform. Theory*, vol. IT-35, pp. 299–315, Mar. 1989.

[7] 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 Process.*, 1994. Submitted for possible publication.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.* Belmont, CA: Wadsworth, 1984.

[9] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola, "LVQ_PAK: The learning vector quantization program package, version 2.1," tech. rep., Helsinki University of Technology, Laboratory of Computer and Information Science, Finland, Oct 1992. Available via anonymous ftp to cochlea.hut.fi (130.233.168.48).