CONTEXT-DEPENDENT TREE-STRUCTURED IMAGE CLASSIFICATION USING THE QDA DISTORTION MEASURE AND THE HIDDEN MARKOV MODEL

Kivanc M. Ozonat and SangHo Yoon

Information Systems Laboratory, Department of Electrical Engineering
Stanford University, Stanford, CA 94305, USA
{ozonat,holyoon}@stanford.edu

ABSTRACT

Vector quantization based on the Gauss mixture model (GMM) and the quadratic discriminant analysis (QDA) distortion measure has been shown to perform well in statistical image classification problems. Previous work in this area has concentrated on designing a separate GMM-based vector quantizer using the QDA distortion measure for each class using full search. We design a single vector quantizer for all classes using a tree-structured algorithm based on the (generalized) BFOS algorithm. This reduces the search complexity, while it increases the correct classification rate. Further, the pruning stage of our algorithm takes into account the dependencies between the image blocks assuming a hidden Markov model (HMM). During the test stage, our algorithm aims to iteratively maximize the joint probability of occurrence of all image blocks based on the HMM. Our simulation results indicate that our algorithm performs better (both in terms of computational complexity and classification rate) when compared to the previously published algorithms based on the GMM.

1. INTRODUCTION

Statistical image classification using vector quantization based on the Gauss mixture model (GMM) and the quadratic discriminant analysis (QDA) distortion measure has been shown to perform well in image and texture classification problems [1,2]. Previous work has concentrated on designing a separate vector quantizer for each class and training with full search using the Lloyd algorithm. The training stage iteratively finds the Gauss mixture component that minimizes the QDA distortion for each feature vector (extracted from each image block) using full search followed by the update of the parameters of the Gauss mixture components. During the test stage, for each test image block, the Gauss mixture component minimizing its QDA distortion is selected and the test block is assigned to the class to which the selected component belongs. Recent work has also incorporated context information, assuming a hidden Markov model (HMM) [3,4]. In particular, two of the feasible strategies have been using the path-constrained Viterbi algorithm in the test and training stages and using the Ising model in the test stage.

Our algorithm, tree-structured vector quantization based on the QDA distortion measure (TSVQ-QDA), is based on tree-growing followed by pruning using the BFOS algorithm. Tree-structured training reduces the computational complexity from $O(K^2)$ to $O(K)$, where K represents the number of mixture components. In addition, our algorithm leads to a better classification performance when compared to its full-search counterparts. In particular, its correct classification rate is higher than that obtained by both the classifier designed by the full-search Gauss mixture model vector quantization (GMM-VQ) and the classifier designed by first growing a codebook based on full search using the splitting measure used in TSVQ-QDA, followed by pruning. TSVQ-QDA's better performance over GMM-VQ is not a coincidence because GMM-VQ, as shown by our simulations, converges to a poor local minimum. This problem is avoided in TSVQ-QDA. Convergence to a poor local minimum does not occur when the full search quantizer is designed by first growing a codebook based on full-search followed by pruning; however, it is still inferior to TSVQ-QDA in terms of its correct classification rate. In addition to its high correct classification rate and low computational complexity, TSVQ-QDA does not require the designer to pre-select the number of Gauss mixture components since the number of components is determined by the algorithm in the pruning stage. On the other hand, in TSVQ-QDA, the level to which the tree needs to be grown (prior to pruning) needs be pre-selected, which we resolve by growing a set of trees each with a different depth and then pruning each using the BFOS algorithm to the point at which its QDA distortion is minimized. We then select the best one in terms of its classification rate on a validation set of vectors.

TSVQ-QDA incorporates context information in the pruning stage, where each pruned subtree of the fully-grown tree provides us with a joint likelihood of the image blocks based on the HMM. The optimum subtree selected by the BFOS algorithm is then the subtree that leads to the maximum joint likelihood. Training all classes together (instead of designing a separate quantizer for each) is a key factor that allows us to include the HMM in TSVQ-QDA.

During the test stage, the optimum sequence of hidden Markov states is determined by an iterative algorithm that tries to maximize the joint likelihood of the test image blocks using the optimal subtree and the HMM parameters obtained in the training stage. We observe that this algorithm leads to a better performance than the path-constrained Viterbi algorithm (assuming a causal model) with comparable complexity.

2. CLASSIFICATION USING TSVQ-QDA

2.1. The BFOS Algorithm in TSVQ-QDA

The BFOS algorithm requires each node of the tree to have two linear functionals such that one of them is monotonically increas-

This work was supported by the National Science Foundation under NSF Grants MIP-9706284-001 and CCR-0073050.

ing and the other is monotonically decreasing [5]. Toward this end, we view the QDA distortion of any subtree of the fully-grown tree as a sum of two tree functionals, u_1 and u_2, such that:

$$u_1 = \frac{1}{2}\sum_{s_i \in T} p_i \ln\left((2\pi)^d |\Sigma_i|\right) + \frac{1}{2M}\sum_{s_i \in T}\sum_{x_{k,l} \in s_i} g_i(x_{k,l}), \tag{1}$$

$$u_2 = -\sum_{s_i \in T} p_i \ln p_i, \tag{2}$$

where s_i is the i^{th} node with probability p_i, M is the number of training blocks, T is the set of the terminal tree nodes of the subtree, $x_{k,l}$ is the d-dimensional feature vector of the image block at location (k,l), μ_i and Σ_i are the mean vector and the covariance matrix of node s_i, and $g_i(x_{k,l})$ is defined as:

$$g_i(x_{k,l}) = (x_{k,l} - \mu_i)^t \Sigma_i^{-1} (x_{k,l} - \mu_i) \tag{3}$$

The functionals, u_1 and u_2, in (1) and (2) are linear as each can be represented as a linear sum of its components in each terminal node of the tree.

The monotonic decrease of u_1 is due to the minimization using the Lloyd algorithm described next, while the monotonic increase of u_2 follows from Jensen's inequality and convexity.

The splitting stage is followed by pruning based on the BFOS algorithm. By the linearity and monotonicity of the tree functionals, the optimal subtrees (to be pruned) are nested, and at each pruning iteration, the selected subtree is the one that minimizes:

$$r = -\frac{\Delta u_1}{\Delta u_2}, \tag{4}$$

where Δu_i, $i = 1, 2$, is the change of the tree functional u_i from the current subtree to the pruned subtree of the current subtree.

The magnitude of (4) increases at each iteration. This is a key point of our design as, then, pruning is terminated when the magnitude of (4) reaches 1, resulting in the subtree minimizing $u_1 + u_2$ [6].

2.2. Classifier Design

The classifier design is depicted in Fig. 1. The design starts with a single node tree, called T_1, out of which two child nodes are grown. The Lloyd algorithm is then applied between these two child nodes, minimizing (1), and this new tree is denoted as T_2. Each terminal node of T_2 is then split, two pairs of child nodes are obtained, and the Lloyd algorithm is applied between each pair, minimizing (1) to obtain T_3. This procedure of splitting a tree, T_i, to obtain T_{i+1} and running the Lloyd algorithm between pairs of the child nodes is repeated until $i = D$, where D is sufficiently large.

Following the tree-growing stage, we form a set of fully-grown trees. The set consists of the D trees, T_i, $1 \le i \le D$. Then, each of the D trees, T_i, $1 \le i \le D$, is pruned using the BFOS algorithm, and for each T_i, pruning is stopped when the magnitude of (4) reaches 1. Hence, for each T_i, $1 \le i \le D$, the subtree that minimizes the QDA distortion is obtained. We denote these best subtrees (in the sense of minimizing the QDA distortion) as P_i, $1 \le i \le D$. It should be noted that applying the BFOS algorithm D times does not increase the computational complexity of the algorithm significantly as the BFOS algorithm itself adds relatively small complexity to the algorithm.

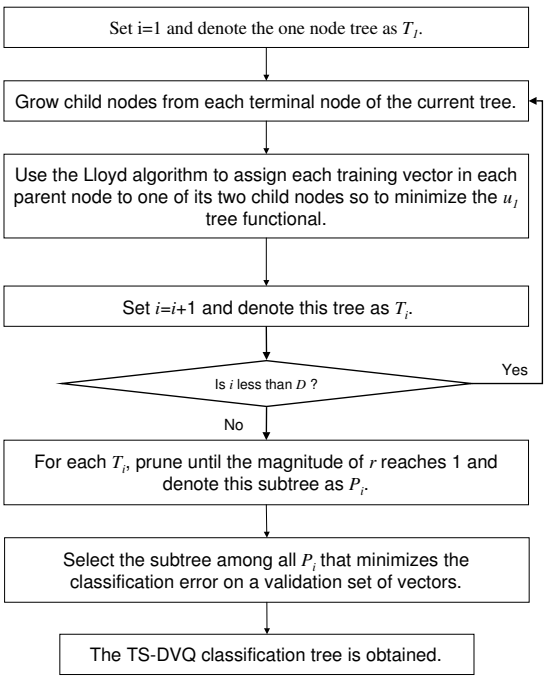

Fig. 1. Classifier Training Algorithm

Finally, the best subtree P^* is selected (to be the classification tree) as the subtree that minimizes the classification error on a (separate) validation set of vectors among all P_i, $1 \le i \le D$.

2.3. Classification of Test Vectors

A test vector, $x_{k,l}$, is assigned to the node s_{i^*} if:

$$i^* = \arg\min_i \left[-\ln p_i + \frac{1}{2}\ln\left((2\pi)^d |\Sigma_i|\right) + \frac{1}{2} g_i(x_{k,l})\right] \tag{5}$$

Since only a single quantizer is designed for all classes, each node, s_i, to which more than one class of training vectors is assigned, is labeled using the majority vote rule. In particular, the number of training vectors of each class, C_j, in node s_i is counted and denoted as $n_{i,j}$ and s_i is labeled as class C_{j^*}, where j^* is given by:

$$j^* = \arg\max_j n_{i,j} \tag{6}$$

During the test stage (also during the best subtree selection stage of classifier design), any vector assigned to node s_i is labeled as the class label of node s_i.

3. CONTEXT-DEPENDENT MODELING BASED ON THE HIDDEN MARKOV MODEL

3.1. Training Stage

According to our Markov model, the hidden Markov state of the image block at location (k,l) is independent of the states of the

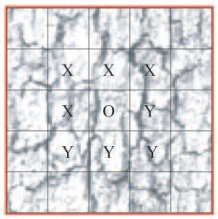

Fig. 2. X's represent N_1 and Y's represent N_2 with respect to O.

image blocks in $P_{k,l}$, where $P_{k,l} = \{(m,n) : m < k \text{ or } m = k, n < l\}$, given the state of the image blocks in the neighborhood $N_{1,k,l}$, where $N_{1,k,l} = \{(m,n) : (m,n) \in \{(k-1,l-1),(k-1,l),(k-1,l+1),(k,l-1)\}\}$. Denoting the hidden state (node), to which the training image block at location (k,l) is assigned in subtree t, as $v_{k,l,t}$ (hence each $v_{k,l,t}$ is one of the s_i's), the joint probability of all feature vectors and hidden states in the image is then given by:

$$\prod_{(k,l)\in R} a_{k,l,t} P(x_{k,l}|v_{k,l,t}), \tag{7}$$

where R is the set of the training image block locations, and $a_{k,l,t}$ and $P(x_{k,l}|v_{k,l,t})$ are defined as:

$$a_{k,l,t} = P(v_{k,l,t}|v_{m,n,t} : (m,n) \in N_{1,k,l}), \tag{8}$$

$$P(x_{k,l}|v_{k,l,t}) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}g_i(x_{k,l})}, \tag{9}$$

provided i is such that $s_i = v_{k,l,t}$ in (8).

Then, maximizing this joint probability is equivalent to minimizing:

$$u_1 + u_2, \tag{10}$$

where u_1 is as given in (1), and u_2 is given by:

$$u_2 = \frac{1}{M} \sum_{(k,l)\in R} -\ln a_{k,l,t} \tag{11}$$

A problem is that the linearity and the monotonicity of the tree functionals cannot be ensured when u_2 is as given in (10). To overcome this problem, we use the observation (from our simulations) that in regions with class changes, the hidden states of the image blocks in $N_{1,k,l}$ do not provide significant additional information on the state of the image block at (k,l), given the true classes of the image blocks in $N_{1,k,l}$. Then, we can re-define $a_{k,l,t}$ as:

$$a_{k,l,t} = P(v_{k,l,t}|C_{m,n} : (m,n) \in N_{1,k,l}), \tag{12}$$

where $C_{m,n}$ is the true class of the image block at (m,n).

The training is then done using the BFOS algorithm as in section 2 except that the u_2 in (2) is now replaced by the u_2 given in (11).

3.2. Test Stage

We modify our classification algorithm discussed in section 2.2 to incorporate context information. The context-dependent algorithm tries to maximize the joint likelihood of the entire image iteratively. In the initial iteration, the algorithm uses an HMM based only on the neighborhood model in section 3.1. For the remaining iterations, we extend the neighborhood region to include all of the eight adjacent neighbors.

Iteration	Location	$q_{i,k,l}$
Initial Iteration	in S	$P(s_i)$
	not in S	$P(s_i \mid C'_{m,n} \in N_{1,k,l})$
Remaining Iterations	in S	$P(s_i \mid C'_{m,n} \in N_{1,k,l} \bigcup N_{2,k,l})$
	not in S	$P(s_i \mid C'_{m,n} \in N_{1,k,l} \bigcup N_{2,k,l})$

Fig. 3. The form of $q_{i,k,l}$ is dependent on the iteration and location.

During the initial iteration, first the hidden states of the image blocks on the upper and left edges of the test image are determined using the algorithm in section 2.3 without using context information. Then, the hidden states of the remaining image blocks are determined. In particular, a test block, $x_{k,l}$, is mapped to the state (node) s_i^* if:

$$i^* = \arg\min_i \left[-\ln q_{i,k,l} + \frac{1}{2}\ln\left((2\pi)^d|\Sigma_i|\right) + \frac{1}{2}g_i(x_{k,l}) \right], \tag{13}$$

where $q_{i,k,l}$ is given by $P(s_i|C'_{m,n} : (m,n) \in N_{1,k,l})$ and $C'_{m,n}$ is the class to which the test block at (m,n) is assigned by the current iteration of the test algorithm. $P(s_i|C'_{m,n} : (m,n) \in N_{1,k,l})$ is computed in the training stage using (12).

The test block, $x_{k,l}$, is then assigned to the class of the state s_{i^*}, using the majority rule discussed in section 2.3.

The initial iteration assigns each image block to a hidden state and a class and provides a reasonable initial estimate of the hidden states of the test blocks. We can now refine this initial estimate by defining an extended neighborhood N to include all of the eight adjacent neighbors as $N_{k,l} = \{N_{1,k,l} \bigcup N_{2,k,l}\}$, where $N_{1,k,l} = \{(m,n) : (m,n) \in \{(k-1,l-1),(k-1,l),(k-1,l+1),(k,l-1)\}$ and $N_{2,k,l} = \{(m,n) : (m,n) \in (k,l+1),(k+1,l-1),(k+1,l),(k+1,l+1)\}$. The remaining iterations assume the extended neighborhood model, which leads to a better classification result as the dependencies from all of the adjacent neighbors are used. The problem is that the training algorithm provides the $a_{k,l,t}$ parameters only for $N_{1,k,l}$, but not for $N_{2,k,l}$. The key to solving this problem is to assume that the statistics of the test image is invariant under a rotation of 180 degrees. We note that when the image is rotated by 180 degrees, $N_{1,k,l}$ and $N_{2,k,l}$ switch their locations. Then, using this symmetry between $N_{1,k,l}$ and $N_{2,k,l}$, we compute the parameters for $N_{k,l}$ from the training parameters computed for $N_{1,k,l}$.

Although the same distortion measure, as given in (13), is used throughout the test stage to assign the test blocks to the hidden nodes (states), $q_{i,k,l}$ varies depending on the iteration and the location of the test block. Denoting S as the set of images in the upper and left edges of the test image, the different forms of $q_{i,k,l}$ are summarized in Fig. 3. If a test block is located on an edge such that one or more of the neighbors don't exist, then $q_{i,k,l}$ is computed using only its existing neighbors. This is not taken into account in Fig. 3.

4. SIMULATION AND RESULTS

We used a set of six aerial images in our simulations. Each image is of size 512×512, and is divided into blocks of 8×8, with each block belonging to one of the two classes. In each experiment, we used 5 of the images for training and the remaining image for the test stage using cross-validation. The DCT coefficients matrix for each 8×8 image block is computed and the upper-left 4×4 DCT blocks from the DCT coefficients matrix are extracted to be used as feature vectors. We also computed the DCT coefficients of the 4 non-overlapping 4×4 image blocks within each 8×8 image block and included the 3 highest-energy DCT coefficients (except the DC coefficient) of these 4×4 image blocks in the feature vector. In each experiment, D, the number of levels discussed in section 2, is selected to be 8, although the results are not very sensitive to the exact value of D.

Table-1 compares the classification error of TSVQ-QDA with that of the recently published algorithms based on the GMM, using the same set of aerial images [2,3,4,7]. TSVQ-QDA with context information has a better performance than all of the algorithms. TSVQ-QDA with no context information is inferior to only the HMM-GMM, which is context dependent.

Table-1) Comparison of Classification Algorithms

Classification Algorithm	Classification Error
TSVQ-QDA context	.138
HMM-GMM	.140
TSVQ-QDA no-context	.159
MHMM	.160
ARM	.178
Causal HMM	.188
GMVQ	.190
CART	.216
LVQ	.218

Table-2 compares the classification errors of TSVQ-QDA with those of a full-search GMM-VQ classifier and a classifier designed by first growing a codebook based on full search using the u_1 measure used in TSVQ-QDA, followed by pruning. We call this latter classifier a *pruning classifier*. All three classifiers use the same set of feature vectors and use context information as modeled in Section 3, and they all have a comparable number of mixture components (or nodes). Table-2 shows that TSVQ-QDA performs better than both of the full-search algorithms. We note that the classifier designed by the full-search GMM-VQ converges to a poor local minimum, i.e. its overall QDA distortion is higher than that designed by TSVQ-QDA. Our observations have shown that the inclusion of the $\ln p_i$ term in the codebook growing stage leads to the poor local optimum problem. The full-search classifier with pruning does not converge to a poor local minimum; however, in terms of classification performance, TSVQ-QDA is still better.

Table-2) Comparison with full search algorithms

Test Image	GMM-VQ	Pruning	TSVQ-QDA
1	.21	.17	.15
2	.11	.09	.09
3	.27	.25	.23
4	.16	.22	.17
5	.04	.05	.03
6	.19	.17	.16

We also compared the path-constrained Viterbi algorithm and our test algorithm in Section 3.2, both applied to the same test set and using the same set of parameters obtained from the training algorithm in Section 3.1. As our algorithm converged in 4-5 iterations, the number of paths allowed in the Viterbi algorithm was equal to 4 or 5 to have comparable complexity [3]. The path-constrained Viterbi algorithm led to a classification error of 0.142, which is worse than that obtained by the context-dependent TSVQ-QDA. Since the Viterbi algorithm can take into account only the causal dependence, our algorithm leads to a better classification performance.

5. SUMMARY AND CONCLUSIONS

We proposed TSVQ-QDA, tree-structured vector quantization with the QDA distortion measure, as an image classification algorithm. TSVQ-QDA performed better in our aerial image simulations than the full-search algorithms implemented previously. Further, TSVQ-QDA has low training complexity. Also, our test algorithm is not restricted to a causal HMM, instead it incorporates the dependencies from all of the adjacent image blocks. An important observation is that the overall QDA distortion obtained with TSVQ-QDA is lower than that obtained using the full search algorithm, which indicates that the classification performance of TSVQ-QDA is not a coincidence, and that the full-search algorithm converges to a poor local minimum.

6. ACKNOWLEDGEMENTS

We would like to thank Professor Robert M. Gray at Stanford University for helpful discussions and his suggestions.

7. REFERENCES

[1] R.M. Gray, "Gauss mixture vector quantization," *Proc. Int. Conf. Acoust., Speech and Signal Processing*, vol. 3, pp. 1769-1772, 2001.

[2] S. Yoon , C.S. Won, K. Pyun, and R.M. Gray, "Image classification using GMM with context information and reducing dimension for singular covariance," *Proc. Data Compression Conf.*, pp. 457-457, 2003.

[3] J. Li, A. Najmi, and R.M. Gray, "Image classification based on a multi-resolution two-dimensional hidden Markov model," *Proc. Int. Conf. Image Processing*, vol. 1, pp. 348-352, 1999.

[4] K. Pyun, C.S. Won, J. Lim, and R.M. Gray, "Robust image classification based on a non-causal hidden Markov Gauss mixture model," *Proc. Int. Conf. Image Processing*, vol. 3, pp. 785-788, 2002.

[5] P.A. Chou, T. Lookabaugh, and R.M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Transactions on Information Theory*, vol. 35, pp. 299-315, March 1989.

[6] K.M. Ozonat, "Image classification using tree-structured discriminant vector quantization," *Proc. Asilomar Conf. on Signals, Systems and Computers*, vol. 2, pp. 1610-1614, 2003.

[7] J. Li and R.M. Gray, "Image classification by a two-dimensional hidden Markov model," *Proc. Int. Conf. Acoust., Speech and Signal Processing*, vol. 6, pp. 3313-3316, 1999.

[8] A.K. Aiyer, "Robust image compression using Gauss mixture models," Ph.D. dissertation, Stanford University, Department of Electrical Engineering, 2001.