Adaptive Hierarchical Multi-class SVM Classifier for Texture-based Image Classification

Song Liu, Haoran Yi, Liang-Tien Chia, and Deepu Rajan
Center for Multimedia and Network Technology
School of Computer Engineering
Nanyang Technological University, Singapore 639798
{pg03988006, pg03763623, asltchia, asdrajan}@ntu.edu.sg

Abstract

In this paper, we present a new classification scheme based on Support Vector Machines (SVM) and a new texture feature, called texture correlogram, for high-level image classification. Originally, SVM classifier is designed for solving only binary classification problem. In order to deal with multiple classes, we present a new method to dynamically build up a hierarchical structure from the training dataset. The texture correlogram is designed to capture spatial distribution information. Experimental results demonstrate that the proposed classification scheme and texture feature are effective for high-level image classification task and the proposed classification scheme is more efficient than the other schemes while achieving almost the same classification accuracy. Another advantage of the proposed scheme is that the underlying hierarchical structure of the SVM classification tree manifests the interclass relationships among different classes.

1. Introduction

With the rapid growth in the number of images, there is an increasing demand for effective and efficient image indexing and retrieval mechanisms. For large image databases, successful image indexing will greatly improve the efficiency of content-based image retrieval (CBIR). One attempt to solve the image indexing problem is by using image classification to get high level concepts [8]. In such systems, an image is usually represented by various low-level features and high level concepts are learned from these features. Support Vector Machine (SVM) has recently attracted growing research interest due to its ability to learn with small samples and to classify high-dimensional data. However, SVM itself is capable of solving only binary classification problem [1]. In order to deal with multiple classes, several schemes have been proposed, e.g. (1) one-against-rest, (2) one-against-one and (3) decision directed acyclic graph SVM (DAGSVM) [3]. In the one-against-rest scheme, for a $n-$class classification problem, n classifiers $SVM_i, i = 1, \cdots, n$, which takes the samples of class i as positive samples and the rest as negative samples, are constructed during training. At the test stage, the test sample is evaluated against $SVM_i, i = 1, \cdots, n$ and the SVM_i which gives highest decision value as the predicted class is chosen. In the one-against-one scheme, one classifier $SVM_{i,j}$ is constructed for every pair of classes (i, j). There are $N(N-1)/2$ classifiers in all. During the test stage, the samples are evaluated against all the pairwise classifiers $SVM_{i,j}$. The final decision of the class for the test samples are determined by a voting scheme. In DAGSVM scheme, same as the one-against-one, $N(N-1)/2$ pairwise classifiers are constructed. In the test stage, a list with all class candidates is created. During each test with the SVM, the class candidate which is given negative label by the SVM is removed from the list.

Since the SVM classifier is a binary classifier, it is natural to organize the SVM classifiers in a binary tree structure. At each node, the classes are divided into two separate subsets. Therefore, we propose a new scheme, adaptive hierarchical SVM classification scheme, for multiple classes. This scheme is a binary SVM tree, where each node of the tree represents a SVM classifier. Both the training and testing phases of the classifier are carried out in a top-down manner. By utilizing this scheme in high-level image classification, it can improve the classification efficiency while achieving similar classification accuracy compared with other schemes. The structure of SVM classification tree also indicates interclass relationships among different classes, which provides useful information for measuring similarities of high-level concepts. We also propose a new texture feature for high-level image classification. The original texton [5] histogram approach has been extended to texture correlogram to provide texture description for a high-level concept.

The remainder of the paper is organized as follows. Section 2 describes the proposed hierarchical SVM tree classification scheme. The extraction of the texture correlogram is given in section 3. Section 4 presents the experimental re-

0-7803-9332-5/05/$20.00 ©2005 IEEE

sults of high-level image classification. Section 5 concludes the paper.

2 Adaptive Hierarchical Multi-class SVM Classifier

As described previously, SVM is a binary classifier. The proposed adaptive hierarchical multi-class SVM classification scheme is a binary SVM tree. We describe how to build the adaptive hierarchical multi-class SVM classifier (SVM tree) in the training stage and illustrate how to use the SVM tree to classify new input patterns during the test phrase.

2.1 Training

The training for the hierarchical SVM-tree classifier starts from the whole dataset. We recursively partition the current dataset into two non-overlapping data subsets. These two subsets are used as positive and negative samples to train an SVM classifier. The resulting SVM classifier is the current node classifier of the final SVM tree classifier. Since each classifier divides the data into two sets, $N-1$ such classifiers are needed to solve a N-class classification problem.

One issue for the training algorithm of the SVM tree classifier is how to divide the classes into two separate subsets. The optimal partition is the one that the gap between these two subsets after partition is the largest. But in order to find the largest gap of the optimal partition, C_n^2 comparisons are needed. To alleviate the heavy computation cost of direction comparison method, we view the division of the class sets into two subsets as a clustering problem. We wish to group the classes into two non-overlapping subsets.

We represent each class by its corresponding mean vector $\mu_{\mathbf{k}} = \frac{1}{n_k}\sum_{\mathbf{x_i}\in C_k}\mathbf{x_i}$, where n_k is the number of samples in class C_k and X_i is the data vector. Then we cluster the N μ_ks $(k = 1, \cdots, n)$ into two clusters by k-means algorithm ($k = 2$). The summary of the training algorithm is described in Algorithm 1.

Algorithm 1 Adaptive Hierarchical SVM Training Algorithm

Input: Whole dataset.

Output: SVM Tree classifier.

1: Partition the dataset into two non overlapping subsets A and B using the k-means partition algorithm described above.
2: Train a binary classifier with the datasets A and B as positive and negative samples, respectively.
3: Repeat step 1 and 2 on datasets A and B, respectively until they only contain data from a single class.

Figure 1 illustrates an example of the training algorithm for SVM tree classifier. The dataset contains 5 classes. After training, SVM tree classifier contains 4 node SVM classifiers. At the top level, the dataset $\{1, \cdots, 5\}$ is divided into to set $\{1, 4, 5\}$ and $\{2, 3\}$ by SVM_1. At the second level, dataset $\{2, 3\}$ is divided into $\{2\}$ and $\{3\}$ respectively by SVM_2; dataset $\{1, 4, 5\}$ is divided into $\{1, 4\}$ and $\{5\}$ by SVM_3. Finally, the dataset $\{1, 4\}$ is divided in to $\{1\}$ and $\{4\}$ by SVM_4. Figure 1(b) shows an example SVM tree classifier.

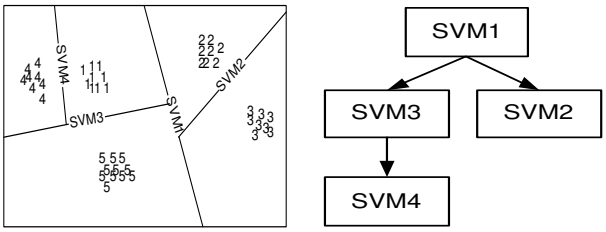


Figure 1: (a) adaptive training of multi-class SVM classifier. (b) SVM tree.

2.2 Testing

The training algorithm described previously yields a binary SVM tree classifier. The SVM tree is very similar to the decision tree classifier [7]. The only difference from the decision tree is that the decision function for the node is a SVM classifier. Therefore, the classification of new pattern is carried out in a top-to-bottom manner. The test pattern start from the root nodes. At every node, based on the classification result of the current node SVM (positive/negative), the test pattern is passed to the left/right branch of the tree. The final classification decision of the test pattern is given by the classification result of the final leaf node SVM classifier.

The smallest computation of classifying the test pattern is just one SVM evaluation when we can make the decision at the top node. The worst case is $N-1$ SVM evaluations when we have to traverse all $N-1$ SVM nodes classifiers before we get the classification decision. The test phase for one pattern by one-against-one, one-against-rest and DAGSVM approaches require $N(N-1)/2$, N and $N-1$ SVM evaluations respectively [3]. Compared with those approaches, the proposed SVM tree classifier is more efficient in the test phase. The efficiency gained in testing phase is very important for many practical applications since the classification stage in many practical applications are required to be online and requires fast response, while training can be done offline.

3. The Texture Correlogram

It is a challenge to select appropriate low-level features for high level image classification. The extracted features should be discriminating enough for differentiating semantically meaningful image categories. A number of image

features based on color and texture attributes have been investigated for image retrieval and classification tasks. Among them, color correlogram has been proved to be an efficient and effective feature [4]. Compared to color histogram, correlogram contains not only pixel-based color distribution but also the spatial color distribution information. Histogram-based texture feature has been studied by Malik and Belongie in [5], the authors used histogram of texton for image retrieval and classification. We extend their idea by extracting correlogram from texton as the feature.

The first step of building texture correlogram is texture feature extraction. We extract Gabor texture feature [6] from a $w \times w$ pixel sliding window and shift this window along one direction by s pixels one time until the whole image is covered by the window. For a $M \times M$ image, $((M-w)/s)^2$ set of Gabor texture features are extracted. The next step is building up the texton from the extracted Gabor texture features. Since Gabor texture features are high-dimensional vectors, it is very hard to apply normal quantization methods on them. To solve this problem, we just use k-means clustering algorithm to generate the codebook from Gabor texture features. Subsequently, for each feature vector, we select the index value of the nearest codebook vector measured by city-block distance to represent its texton. Thus, the cluster number defined in k-means algorithm also decides the numbers of different textons.

After we get the texton matrix for an image, we calculate the texture correlogram as follows. A correlogram C^k is a matrix whose element (i,j) is the probability of finding a texton with index value j at a distance k from a texton with index value i. For an $N \times N$ texton matrix, the correlogram element at (i,j) for distance k is given by

$$C^k(i,j) = P(I(t_2) = j | I(t_1) = i, d(t_1,t_2) = k) \quad (1)$$

where $I(t_2)$ $(I(t_1))$ is the index value of texton t_2 (t_1). We use the L_∞ norm as the distance measure $d(t_1,t_2)$, i.e., $d(t_1,t_2) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$, where texton $t_1 = (x_1, y_1)$ and $t_2 = (x_2, y_2)$. The computation of the correlogram involves the counting of the value pairs (i,j), which is very similar to the coocurrence matrix defined in [2] for texture analysis, i.e.,

$$A(i,j) = \#\{I(t_1) = i, I(t_2) = j, d(t_1,t_2) = k\} \quad (2)$$

and then correlogram can be computed by

$$C^k(i,j) = \frac{A(i,j)}{n_i \cdot 8k} \quad (3)$$

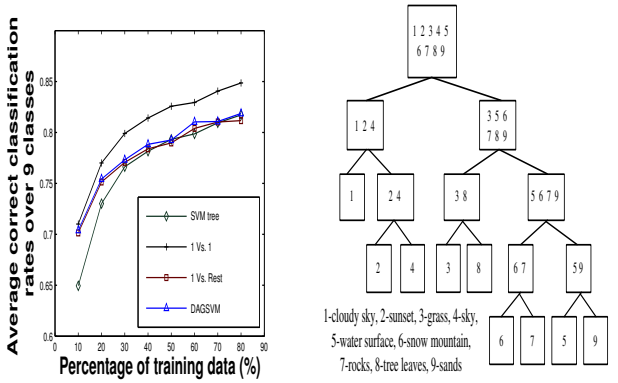
where n_i is the number of textons with index value i in I and $8k$ is the number of textons in the neighborhood at distance k from t_1. A set of correlograms is found for each distance in the set $s \in \{1,\ldots,k\}$. The computational complexity of finding a correlogram for an $N \times N$ texton matrix and distance k is $O(N^2k)$. To compute an entire set of correlograms, one for every distance in $s \in \{1,\ldots,k\}$, the computational complexity is $O(N^2q^2)$, where q is the cardinality of s. This can be reduced to $O(N^2q)$ using dynamic programming [4]. However, since we do not propose to use every possible distance in s, and moreover, since $q \ll N$, the computational burden is not a major concern.

A subset of the correlogram C^k is the autocorrelogram which contains only the diagonal elements of C^k. It gives the probability of finding the same texton index at a particular distance. Since the computations can be further simplified by considering only autocorrelograms, we choose to use autocorrelogram to describe image contents on the basis of texture attribute.

4. Experimental Result

The image database in the experiment consists of 900 256×256 images-blocks which are cropped from 900 different images in Corel image gallery. These image-blocks are classified into nine semantically meaningful categories manually. Each category contains 100 image-blocks. The nine categories are "sky", "cloudy sky", "sunset", "grass", "tree leaves", "sands", "rocks", "snow mountain" and "water surface".

To evaluate the performance of proposed classification scheme, we compare the average correct classification rates over nine classes of our scheme with the other schemes using the proposed texture correlogram feature. Figure 2 (a) shows the classification results. In the figure, the x-axis indicates how much percentage of data is used as the training data. In this experiment, we randomly select fixed amount of data as training data and calculate the mean of correct classification rates over nine classes. To ensure the correct classification rates is not affected by the different choices of training and test data, we perform the training/classifiction



(a) (b)

Figure 2: (a) Performance comparison of adaptive hierarchical multi-class SVM classification scheme and other schemes. (b) resulting class hierarchy.

30 times and average the classification rates. From figure 2 (a), The one-against-one scheme has the best performance for the classification task but it requires high computation. Our proposed scheme is close to one-against-one scheme in terms of classification performance while requires less computation. Figure 2 (b) shows the resulting class hierarchy generated by utilizing texture correlogram feature. We can see that this hierarchy exactly represents the interclass relationships among classes. Therefore, our proposed method can be utilized to automatically give the inferences about the semantic relationships among the high-level categories.

In the second experiment, we compare the performance of texture correlogram with the cooccurance matrix and Gabor texture feature. As stated earlier, the autocorrelogram is used as the feature index. The distance set to compute the correlogram is $\{1, 3, 5, 7\}$. To reduce the computational complexity, the size of Gabor filter dictionary is set to 12. The number of different textons are set to 64. For extracting original Gabor texture feature, we use same parameters set in [6]. For extracting cooccurance matrix feature, we calculate the cooccurance matrix using three different distances: [0,1], [1,1] and [1,0], and then the energy, entropy and contrast of each cooccurance matrix are concatenated as a feature vector. Figure 3 shows the experimental results when utilizing texture correlogram and Gabor filter as features for high-level image classification. Compared to the vector size of Gabor texture feature, which is 30×2 component feature vectors, the proposed texture correlogram consumes more storage space because we defined 64 different textons and thus final feature vector has 256 components. However, from figure 3, we can conclude the proposed texture correlogram outperforms the original Gabor texture feaure when we apply these two kinds of features for image classification. Moreover, the above comparisons proved that the spatial distribution information is a very useful cue for high-level texture image classification.

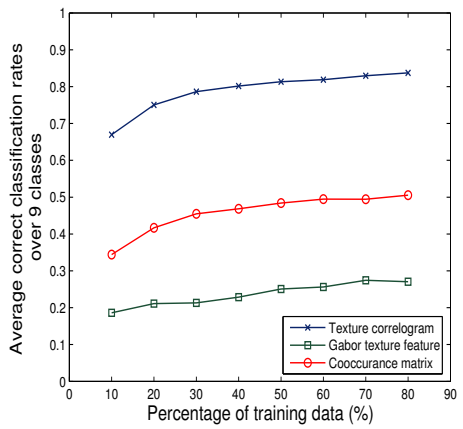


Figure 3: Performance comparison of texture correlogram, cooccurance matrix and Gabor texture feature.

In this experiment, we also evaluate the effect of different choices of sliding-window sizes on the performance of texture correlogram. We tried three different window sizes, 8×8, 16×16 and 32×32. Experimental results shows that the smaller the window size the better the performance of texture correlogram. There are two explanations: 1) Since the small image-block contains simpler texture pattern than the big one, it is easy to quantize those small blocks into textons with limited numbers. 2) The smaller window size gives higher resolution of the texture. This is helpful for accurately estimating the spatial distribution for high-level image contents.

5. Conclusions

The proposed adaptive hierarchical multi-class SVM classification scheme and texture correlogram is proved to be an effective way to solve multi-class high-level classification problem. Compared with the other multi-class classification schemes, our method is more efficient in the test phrase. The experimental results on image classification demonstrate that our classification schemes achieve similar classification accuracy and the texture correlogram outperforms the original Gabor texture feature. Moreover, the hierarchical tree structure generated by our approach indicates the interclass relationships among different classes and dataset. This will help us to further analyze the relationships between high-level concepts.

References

[1] C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, (20):273–297, 1995.

[2] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of IEEE*, 67(5):786–804, 1979.

[3] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

[4] J. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proc. IEEE Comp. Soc. Conf. Comp. Vis. and Patt. Rec.*, pages 762–768, 1997.

[5] J. Malik, S. Belongie, J. Shi, and T. K. Leung. Textons, contours and regions: Cue integration in image segmentation. In *ICCV (2)*, pages 918–925, 1999.

[6] B. Manjunath and W. Ma. Texture features for browsing and retrieval of image data. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

[7] R.O.Duda, P.E.Hart, and D.G.Stork. *Pattern Classification*. John Wiley & Sons, INC, 2001.

[8] A. Vailaya, A. Jain, and H. J. Zhang. On image classification: city images vs.landscapes. *Pattern Recognition*, 31:1921–1936, December 1998.