Finding Clusters Within a Class to Improve Classification Accuracy

Yong Jae Lee
The University of Texas at Austin
EE381K Multidimensional Digital Signal Processing
Literature Survey Report
March 21st, 2008

Abstract

In machine learning, classification is defined as the task of taking an instance of the dataset and assigning it to a particular class. Classifiers are constructed using the training set, such that a novel instance is labeled with the "correct" class. Usually, there is the underlying assumption that instances within a class are similar and instances across classes are dissimilar. For example, given a dataset of images of cars and bicycles, the shapes and appearances of the two classes are different. While this is a valid assumption, in reality, there may also be differences within a particular class, although it may be less pronounced. For example, the car class may be composed of side-view, front-view, and rear-view images of cars. Therefore, variations can be found within each class and homogeneous groups can be formed for each variation. Then, each group can be considered to be a "sub-class" for training a classifier that can focus on the specific aspect of the class.

I. Introduction

Object recognition is a fundamental problem in computer vision that involves the tasks of detecting, categorizing, and identifying objects in images and videos. The goal is to allow a machine to understand and interpret an image or video the way humans do. There are many applications in various fields which can benefit from a successful object recognition system . For example, most of the returned images from search engines are irrelevant to the query term. This problem could be alleviated if the search were performed based on the content of the image rather than the content of the query text. In medical imaging, automatically identifying any irregularities or symptoms would increase early detection of diseases and ultimately chances of survival. In sports, controversial calls made by referees could be eliminated by automatically signaling out-of-bounds plays and other violations that require no human judgement. Vehicle accidents could entirely be eliminated with a vehicle that controls its speed based on road and weather conditions, proximity to other vehicles, etc.

The problem is challenging because of the variability in the position, scale and pose, shape of the object, imaging and lighting conditions, occlusions, and extreme clutter, where the object occupies a much smaller portion in the image than the background.

The standard procedure of measuring success of an object recognition system is to test the algorithm on benchmark datasets. A typical dataset contains a handful to hundreds of object categories. The underlying assumption is that images within the same object category are similar and that images from different object categories are dissimilar, with varying degrees depending on the category. While this is a valid assumption, it may be reasonable and even favorable to divide a category further into sub-categories. Each sub-category, or sub-class, can then be used to train the system, thereby focusing on specifics of the class that would otherwise be ignored with a globally trained system using the original class. This report will focus on the tools necessary for building such an object recognition system. The related works can be divided into four sections: image representation, pairwise image similarity computation, clustering, and classification.

II. Related Work

The "bag-of-features" approach to visual recognition has been quite successful. The basic idea is to represent an image as a bag of local features collected from salient regions. Patches are detected on salient points, e.g., corners, and a visual descriptor vector is evaluated for each patch. Thus, the image can be represented as a distribution of these features. Some examples of salient regions are corner-like regions [1] and "blob-like" regions [2]. The Scale Invariant Feature Transform [3] (SIFT) is another method for detecting and describing salient regions, where its main contribution is the descriptor which describes salient regions based on magnitude and direction of gradients.

To define a similarity measure between images, the distribution of local features can be compared. There are several options for computing similarity between images, including similarity in appearance and similarity in spatial layout. The Proximity Distribution Kernel [4] is a method which measures similarity between images based on the spatial layout of the local appearance features in the images.

Clustering is the partitioning of a data into subsets, such that the instances in each subset have proximity in terms of some defined distance measure. There are several methods for clustering, including k-means [5], agglomerative [6], and spectral clustering [7]. The Normalized Cuts [8] method is a type of spectral clustering algorithm which has been applied to image segmentation and data clustering.

Classification is a tool necessary for labeling novel instances. There are many possible classifiers, such as the Nearest Neighbor Classifier [9], Neural Networks [10], and Support Vector Machines [11]. Support Vector Machines (SVM) have been shown to produce very good results in the fields of text categorization and image categorization, among many others.

III. Scale Invariant Feature Transform

SIFT is one of the most popular techniques for extracting local features in images. In the detection stage, the image is first convolved with Gaussian filters at multiple scales. Then the differences are taken between neighboring Gaussian-blurred images to produce Difference of Gaussians (DoG) images. Each pixel in the DoG images is compared to its immediate neighbors

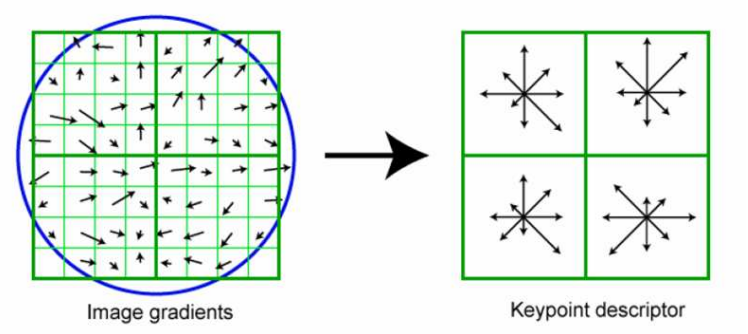

Fig. 1. Illustration of the SIFT [3] descriptor created by computing gradient magnitude and orientation at each sample point in a region around the interest point location.

both in the same scale (8 pixels) and the neighboring scales (9 pixels each), and candidate interest points are taken as the local maxima and minima. This step allows for invariance to scale. Since many of the candidate points are unstable, the algorithm discards points with low contrast and points that are poorly localized along an edge. This is done by performing a detailed fit to nearby data for each point, to determine accurately the location, scale, and ratio of principal curvatures.

Once the final set of keypoints are determined, each one is assigned an orientation. Pixels in a neighboring region around the interest points in the Gaussian-blurred image are assigned to a 36-bin orientation histogram based on their directions and Gaussian weighted gradient magnitudes. Each bin in the histogram covers 10 degrees, and peaks in the histogram correspond to the dominant orientations of the corresponding interest point. This step allows for invariance to rotation.

Now that scale and rotation invariant interest point regions are detected, highly distinctive descriptors are computed for each region. This is the feature descriptor step. Similar to computing the orientation of the interest points, a set of 8-bin orientation histograms are computed on 4 x 4 arrays (each array composed of 16 pixels) for each interest point region. The gradient magnitude of each pixel is weighted by a Gaussian, and added to a bin in the corresponding region histogram. This produces the SIFT descriptor, which is a 128 dimensional feature vector (4 x 4 x 8). Figure 1 shows an example of a (2 x 2 x 8) region descriptor. The descriptor achieves invariance to illumination conditions and minor viewpoint changes. Experiments in [3] show that high accuracy can be achieved for feature matching in different images, and have been shown

to outperform other local descriptors on many types of images.

IV. Proximity Distribution Kernel

In order to train discriminative classifiers that can distinguish one class from another, a similarity measure between all images in all classes must be made. The Proximity Distribution Kernel (PDK) [4] is a method that measures similarity between images based on the spatial layout of their local appearance features.

First, local features are extracted from salient regions in each image. The set of features from all images are vector quantized into R codewords which constitute the codebook of the dataset. Each feature in an image is replaced with the nearest R codeword in the codebook, and is associated with its x and y image coordinates. Then, for each image, a proximity distribution is measured: for each codeword pair v_i and v_j in the image, a distribution $H_r(i,j)$ of the r-spatially nearest codewords of type j to codewords of type i is stored in a 1-D vector of length r. The collection of these 1-D vectors for every combination of word pairs (all possible i and j pairs) produces the proximity distribution H_r. Finally, the proximity distributions between all images are compared to produce the PDK matrix, where the PDK between image I^1 and image I^2 is defined as:

$$PDK(I^1, I^2) = \sum_{i,j=1}^{V} \sum_{r=1}^{R} \min(H_r^1(i,j), H_r^2(i,j)) \tag{1}$$

A nice property of the PDK is that the relative spatial positions between the features in the image are considered, rather than their absolute distances to one another. Thus, the algorithm is invariant to scale, rotation, and translation of the object of interest. On datasets which have a lot of variability within each object category, the PDK has been shown in [4] to outperform methods that take only appearance into account. On datasets which do not have much variability, the PDK shows comparable accuracy to that of other methods.

V. Spectral Clustering with Normalized Cuts

Once we have a measure of similarity between all images, we can use this information to classify novel images. We can also use it to find clusters within each object category. If we

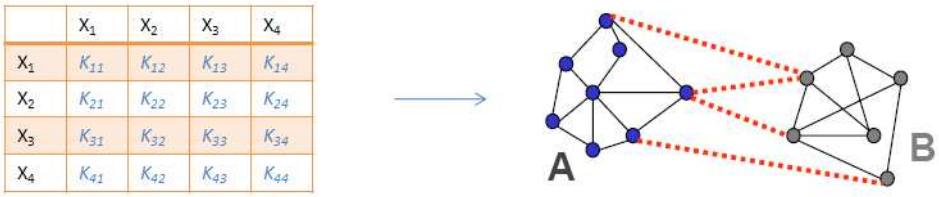

Fig. 2. Using graph theory, images can be considered as nodes and pair-wise similarities can be considered as non-directed edges between nodes.

view each image as a node in a graph, and the similarity values to be a non-directive weighted edge between the images, then we can use graph theoretic methods to partition the graph into clusters. This is the idea of the normalized cuts spectral clustering technique proposed in [8], and is shown in Figure 2. The objective is to partition the graph in such a way that the edges between different groups have low weights while edges within a group have high weights. Each class will be clustered in this way, with the number of clusters for each class being a user-defined parameter.

Specifically, the problem is formulated as maximizing the objective function, $\mathbf{w}_n^T A \mathbf{w}_n$, where A is the affinity (similarity) matrix and $\mathbf{w}_n$ is the vector of weights linking the data points to the n^{th} cluster. Since scaling the weight vector by λ scales the objective function by λ^2 (and hence is not significant), the objective function is maximized subject to $\mathbf{w}_n^T \mathbf{w}_n = 1$. This can be formulated as an eigenvalue problem: $A\mathbf{w}_n = \lambda \mathbf{w}_n$. To find k clusters, the eigenvectors associated with the k largest eigenvalues is computed. By thresholding the components of the eigenvectors (which are the associated weights to the data points), the data points can be clustered according to their weights to each other. The authors show that this is not enough to produce good clusters when the data has outliers. In such data, the optimal cut in the graph will be chosen to leave the outlier by itself. Hence, the problem is adjusted to maximize the within cluster similarity relative to the across cluster difference. The exact solution is found to be NP-hard, but the authors show that an approximate solution can be found.

The paper applies the algorithm to image segmentation where the Normalized Cuts algorithm does quite well. Due to the subjectiveness of image segmentation, quantitative evaluations are not made. However, many algorithms have used the Normalized Cuts method for data clustering

and image segmentation, and in practice have shown very good results.

VI. Support Vector Machines

Support vector machines (SVM) are supervised learning methods used for classification [11]. The objective is to find an $n-1$ dimensional hyperplane that achieves maximum separation between two classes of points in an n-dimensional feature space. The nearest points on either side of the hyperplane are called *support vectors*, and the hyperplane that maximizes the distance to the support vectors is shown to be the unique optimal boundary between the two classes.

The m training points in the feature space can be considered as $\{(\mathbf{x}_1,y_1), (\mathbf{x}_2,y_2), \ldots, (\mathbf{x}_m,y_m)\}$ where $\mathbf{x}_i$ is an n-dimensional feature vector and $y_i \in \{+1,-1\}$ denotes its class label. The points are said to be linearly separable if there exists a vector $\mathbf{w}$ and a scalar b such that: $\mathbf{w}\cdot\mathbf{x}_i + b \geq 1$ if $y_i = 1$ and $\mathbf{w}\cdot\mathbf{x}_i + b \leq -1$ if $y_i = -1$ is valid for all training points, as shown in Figure 3. This can be re-written as

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \quad i = 1,\ldots,m \tag{2}$$

whereby the optimal hyperplane can be found by the optimization problem of minimizing $\|\mathbf{w}\|^2/2$ subject to (2). A test instance $\mathbf{z}$ is classified with $f(\mathbf{z}) = sign(\mathbf{w}\cdot\mathbf{x}_i + b)$. If $f(\mathbf{z})$ = 1, the test instance is classified as the positive class; otherwise, it is classified as the negative class.

The authors show that if the data is not linearly separable, then slack variables can be introduced where some points from the opposite class can belong on the otherside of the separating

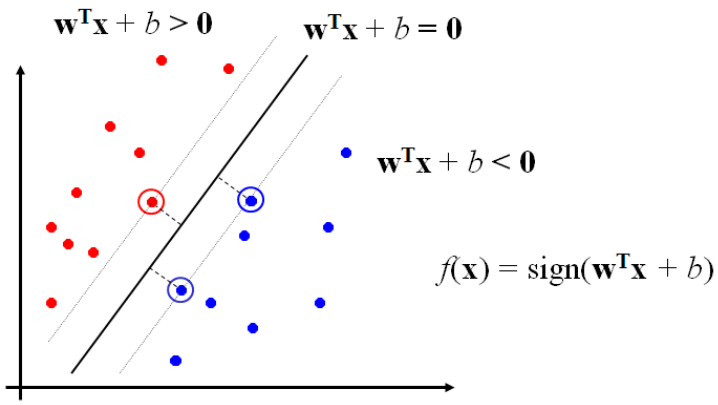

Fig. 3. Decision boundary and margin of SVM. The circled points are the support vectors.

hyperplane. Another method they propose is to map the points into a higher dimensional space in which the data is linearly separable. It is shown that any set of datapoints can be mapped to higher dimensional spaces in which they are separable via the "kernel trick". A kernel function is a function that is equivalent to an inner product in feature space which implicitly maps data to a high-dimensional space. As long as the set of points can be written as an inner product, it can be mapped to a space where it can be separated. This is proven by Mercer's Theorem [12], and is shown that every semi-positive definite symmetric function is a kernel function.

Since the SVM is a binary classifier, there are several options to extend it to the multi-class setting. The simplest is a vote-based method, where all possible pairwise binary classifications between all classes are made. The test instance is labeled with the class that receives the most votes. This is called 1-vs-1 SVM classification.

VII. Conclusion

Object recognition is a challenging problem, but in recent years many techniques have achieved some success. There are many components to the problem, including image representation, learning, clustering, and classification. While a dataset representing an object category should be homogeneous in the sense that each image should contain the object of interest, there is usually enough variation within each object category to form homogeneous sub-categories. These sub-categories can then be treated as sub-classes to train classifiers with the ultimate goal of increasing performance compared to a global classifier which trains on the original class images. The techniques to do so have been reviewed in this paper.

VIII. Future Work

The dataset I will use to test the method will be the PASCAL Visual Object Classes Challenge 2005 dataset [13]. It is composed of 4 object categories: cars, bicycle, people, and motorbikes. It fits nicely with the proposed algorithm because there are many variations in viewpoints, scale, and pose of the objects in the images. The code for extracting SIFT features is available on the webpage of Robotics Research Group at the University of Oxford. LIBSVM [14] has an implementation of the SVM classifier in Matlab. Normalized Cuts clustering code is also

available. I have coded the PDK algorithm in Matlab. Because the algorithm is set up as a combination of modules, I can easily substitute other methods for each module to make improvements.

REFERENCES

[1] K. Mikolajczyk and C. Schmid, "Scale & Affine Invariant Interest Point Detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[2] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *British Machine Vision Conference*, 2002.

[3] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] H. Ling and S. Soatto, "Proximity Distribution Kernels for Geometric Context in Category Recognition," *IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.

[5] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 281-297, p. 14, 1967.

[6] N. Jardine and R. Sibson, *Mathematical Taxonomy*, 1971.

[7] A. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 Conference*, 2002.

[8] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[10] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1995.

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[12] J. Mercer, "Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 209, pp. 415–446, 1909.

[13] M. Everingham, A. Zisserman, C. Williams, L. Van Gool, M. Allan, C. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorko *et al.*, "The 2005 PASCAL visual object classes challenge," *First PASCAL Challenge Workshop*, 2005.

[14] C. Chang and C. Lin, *LIBSVM: a library for SVMs*, 2001.