Implementing “Visual Categorization with Bag of Keypoints”

CNS 148 Final Project

Rita Yun-Hsueh and Jason Yosinski

June 3, 2006

1 Introduction

In this project, we are supposed to redo our reference paper “Visual Categorization with Bag of Keypoints” but with a different approach. We still used the “bag-of-keypoints” method to get the features we want, but instead of using SVM, we used nearest-neighbor and fisher linear discriminant approaches. Also, we adopted a Naïve Bayes approach as a comparison.

2 SIFT Descriptors

First of all, we downloaded 8 directories from Caltech 101, and used “sift.m” as a function by David Lowe to get SIFT descriptors from each image. However, it took almost forever when we used 8 directories. Therefore, we ended up using only 3 directories: anchors, elephants, and helicopters. By using function ‘sift’ we gathered many descriptors for each image. Some of the images and their SIFT features are shown in Figure 1.

3 Cluster SIFT Descriptors

For each image, we had a set of descriptors. After we had all the descriptors, we use k-mean clustering to cluster our descriptors. In order to do this, we put all our descriptors from all our images into a big matrix with each descriptor as a row vector. Then we assign an arbitrary positive integer as k and put the big descriptor matrix as well as k into MATLAB’s kmeans function to cluster descriptors. We tried several values for k, particularly, we tried k = 1000 in order to show the result obtained in our reference paper Visual Categorization with Bag of Keypoint. However, our computer crashed when it ran out of memory. Therefore, we ended up letting k equal to 200. After k-mean we got a matrix C, in which each row is some specific cluster center in feature space, and the number of rows is the number of cluster centers. In other words, C records the position of cluster centers. We could also get a column vector IDX, in which each entry represents the group label for one specific descriptor. In other words, IDX records which center each descriptor is closest to. By doing so we finished clustering. Figure 2 shows the results of the clustering.
Figure 1: A small sample of our training images. The arrows show the location, scale, and direction of a SIFT descriptor.
Figure 2: SIFT descriptor clustering results. The histogram shows how many descriptors ended up in each cluster; we might worry if one cluster’s membership far exceeded all the others.
4 Keypoint Histogram

For each image, we created a keypoint histogram (in the form of a row vector), which records how many times a feature (cluster center) occurs in one specific image. Since we already label all our descriptors with specific group (cluster center), we can easily calculate the number of times each feature occurs by looking at its descriptors' labels. Put all these histogram (row vectors) together, we will have a big matrix where each row is a histogram for one image; whereas each column records, for one specific center, the number of its occurrence in images. In other words, each entry in this matrix is the number of times each feature (center) occurs in one image. The histograms for each image in Figure 1 are shown in Figure 3.
5 Training Set and Testing Set

After getting our histogram, we choose some portion of images (already in the form of histogram; in other words, we take out some row vectors from our histogram matrix) to be the training set, and leave the rest of the images (in histogram form) as testing set.

6 Classifier

6.1 Naïve Bayes

For each image from our testing set, we have its histogram. As well, we can also create a histogram for each class in our training set by summing all of the individual image histograms within each class. In our case, we ended up with three individual histograms for elephants, anchors, helicopters.

Calculate the probabilities of one specific feature (cluster center) appears in one specific directories. (say, calculate the probability of center number 1 appearing in elephant, anchor, or helicopters.) To do so, we look at the histograms for elephant, anchors, and helicopters and divide the height of center number 1 (number of occurrences) to the total area of the whole histogram (total number of occurrences of all the centers) separately. So, for center number 1, we may have \( p = 0.1 \) for elephant, \( p = 0.05 \) for anchors, and \( p = 0.0001 \) for helicopters, say. Then we make three assumptions: our testing image is elephant, anchor, helicopter. For each assumptions we make, we use the specific probabilities of occurrences of each center for each directories to calculate the probability that our testing image is belonged to this directories. Say, if for center number 1, from the histogram we know that it occurs for 10 times, so if we assume its an elephant, the probability should be \( 0.1^{10} \); if it’s assumed to be a anchor, the probability should be \( 0.05^{10} \); if it’s assumed to be a helicopter, the probability should be \( 0.0001^{10} \).

After calculating the probability for all the centers, we classified the image as the class which had the highest probability. To evaluate the performance, we use confusion matrix (Figure 4). The correctness rate is 54%.

Figure 4: Confusion matrix for our Bayes classifier.
6.2 Nearest Neighbor

For this classifier, we basically calculate histogram for *each* image from our training set. After that, we calculate the histogram for our testing image, and compare its histogram to all the histograms we have from training set. The label of the histogram from training set to which our testing image histogram is closest will tell us which directories our testing image belongs to. We use Euclidean distance to see how close our testing image is to other images from training images. By doing so, we classify our testing image. Same as above, we used confusion matrix to evaluate this classifier’s performance (Figure 5, and the correctness rate is 59%). We compare our testing image to histogram from *each* training image, whereas in Naïve Bayes we only compare our testing image to the sum of training images for one specific directory. Summing training images up may lose some important information, and it could be the reason why Nearest Neighbor performs better.

6.3 Fisher Linear Discriminant

Basically, we just use our histograms from our training images as data points, put them into a big matrix, and use function FisherLD, which was given in class, to find the transformation matrix LDs. After that, we multiplied our data matrix with LDs to project our data from higher-dimensioned space to lower-dimensioned space, and pick the most important two dimensions (two column vectors) from our product matrix and start classification. After that, we just use function ROC.m to evaluate its performance.

We did try a lot of ways to let this approach work. However, in the end even our code didn’t have any error, the ROC curve we got was totally blank. We had no idea what happened.