CSE 591: Statistical Learning and Pattern Recognition
Option 5 Project Report

Juan Guzman, Isaac Jones, and Michael McGrath

Introduction:

For Option 5, we were asked to implement the algorithm for object detection in images described in “Discriminative Decorrelation for Clustering and Classification” by Bharath Harihan, Jitendra Malik, and Deva Ramanan, hereafter “the paper.” Since the technical details of the implementation can be found in the paper, this report will serve primarily as a guide to using our code in order to repeat the effects of the paper instead of a full technical report. In addition, a comparison of our results to the results specified in the paper is given as a measure of the quality of implementation. Lastly, a discussion of the contribution of each group member is included.

Running the Included Source Code:

In order to duplicate the procedure described in the paper, a number of scripts must be run in the MATLAB environment. There are five relevant scripts that must be run. The first script is somewhat whimsically named “bossHoG.m.” This script performs performs the conversion from the image space to the histograms of oriented gradients (HoG) space and writes those histograms to a series of files. However, it does not preserve the histograms in MATLAB’s memory, so it can be difficult to put the histograms to proper use. Using the xml annotation files a list of histogram subsections to load is built with the “buildJobList.m” script. While doing that it compares the size of the original image to its HoG representation and converts the objects bounding box to HoG space. The next script “buildLDAInputData.m” concerned with loading the list of histograms of oriented gradients generated by the previous script into MATLAB’s memory so that further computations can be done with them. While it is somewhat pointless to have both of these functions if the results are all obtained on the same machine, this allows for portability of the HoG results across machines. This allows for the assembly of all of the LDA components into one object in MATLAB’s memory, making passing the LDA input to the classifier much easier on the programmer. Lastly, “test.m” performs actual LDA classification. Unlike the paper, we chose to use a 10-fold cross-validation training/test model, since unlike the original VOC Challenge situation, a larger dataset of both the given training and given test data was available. This allows us to get a better idea of the performance of the algorithm in real-world situations. Because of this deviation from the experimental procedure of the original paper, we expect to see a slight difference from the results of the original paper, but this difference should not be very large.

As an alternative to the procedure listed above, a single file, “run.m” can be used for the entirety of the project. Simply running this file with the appropriate input will perform all of the steps performed in the above, longer procedure.

Results:

There were a few errors in our methods that caused us not to get correct results. Stemming from a misunderstanding of the paper to be re-implemented, our process could not train an LDA classifier. As implied by the description of our method above, we attempted to train an LDA classifier with all of the Histograms of Oriented Gradients data instead of following the techniques of the paper to reduce the dimensionality of the dataset to make it feasible for training. in the original paper, the dimensionality of the data was reduced through a series of pre-processing steps, which we failed to do before attempting to train an LDA classifier. In particular, dimensionality reduction was achieved through 3 major steps, with some additional processing.

The first dimensionality reduction step taken by the authors is a creation of a composite background sample in on order to reduce the size of the negative class to a small number of samples instead of an extremely large number. In order to accomplish this, all of the images are assumed to be background images with no objects of interest. Unlike the SVM formulation that had been used in the past, the LDA formulation used in the original paper allows for the estimation of a single set of parameters for the background class, instead of differentiating the background class from all of the individual class from scratch for every object class.Thus, for the LDA model, we can estimate a single μ and Σ for the entire background class and reuse them for the background for every model. This significantly reduces the amount of computation time needed to train the LDA models, as only the positive training examples and information similar to that computed for the background from the negative examples must be specified.

In addition to reducing the dimensionality of the background, preprocessing was done on the data to reduce the dimensionality of the objects of interest. First, the aspect ratio for each of the positive training examples is computed, and a k-means clustering is done on the aspect ratios. The number of clusters is determined by inspecting a scatter plot of the aspect ratios for each class. The number of quasi-linear components in the graph gives the number of means for k-means clustering to use. After this clustering, N-cuts is used to subdivide the clusters of aspect ratios. Lastly, in order to refine the HOG features into more semantically meaningful features, a whitening is applied to the HOG features, which removes some of the background information from the data.

Lastly, the third technique that the authors used to reduce the dimension of the LDA training and classification problem is to use only the medoid of the positive training example clusters against the medoids of the other trainings examples and the background in a many-to-all fashion. This reduces the computational complexity significantly. In the original paper, the final number of clusters used was 77. This is vastly reduced from the 10,000 original images that we attempted to feed directly into an LDA classifier to train the model.

As of time of writing, we had not changed the above system to properly match the model described in the paper. As such, we only have partial results to demonstrate the effectiveness of the model. Figure 1 shows a histogram of the aspect ratios of the training samples. This data is

used to generate clusters of training samples to feed into the eventual LDA classifier. However, since the variance in aspect ratios is rather large, with one aspect ratio at a big over 13, this view alone is not good enough to determine where eligible clusters of aspects ratios may be.

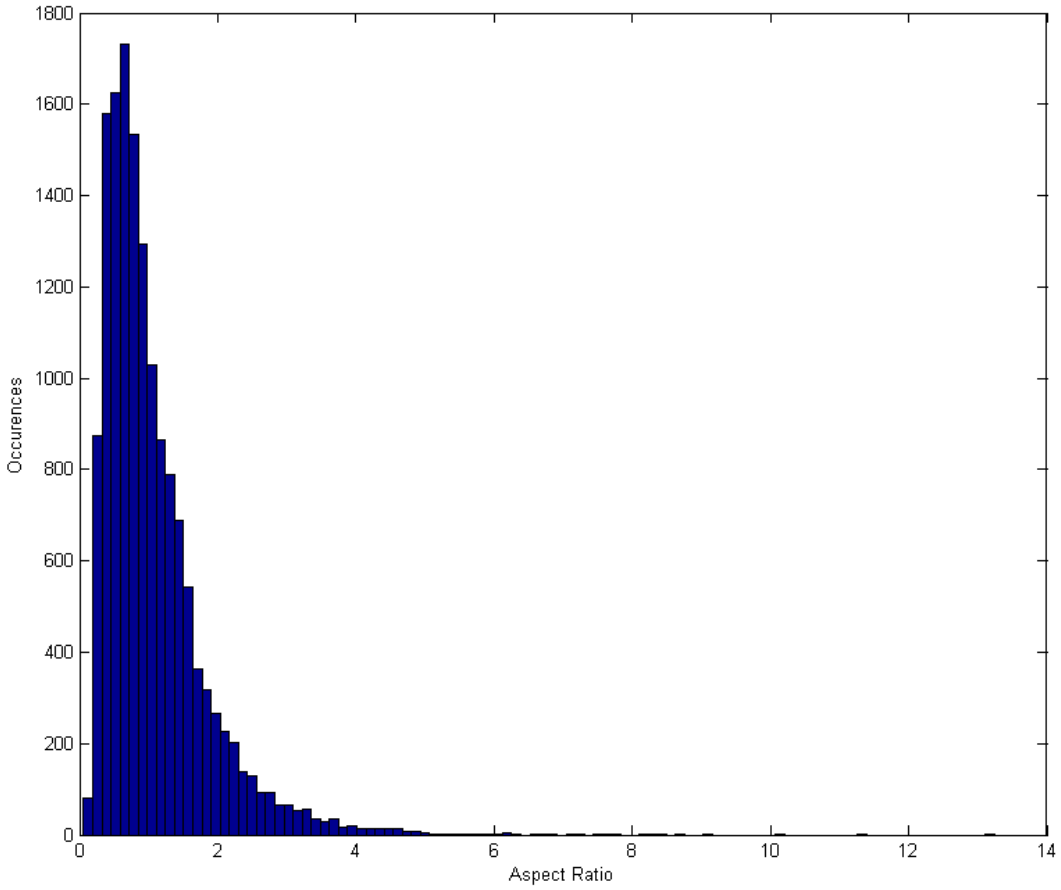

Figure 1: Histogram of All Positive Training Sample Aspect Ratios

Figure 2 demonstrates a closer view of the histogram of Figure 1 for aspect ratios between 0 and 2. Here, we can start to see some clustering in the aspects ratios. There is clearly a peak in the ratios at approximately 0.3, another one at approximately 0.5, a smaller one at just over 1.0, and so on. The visualization of the data allows for the selection of the number of clusters in the clustering portion of the algorithm so that the positive training examples for each class can be clustered into an appropriate number of clusters.

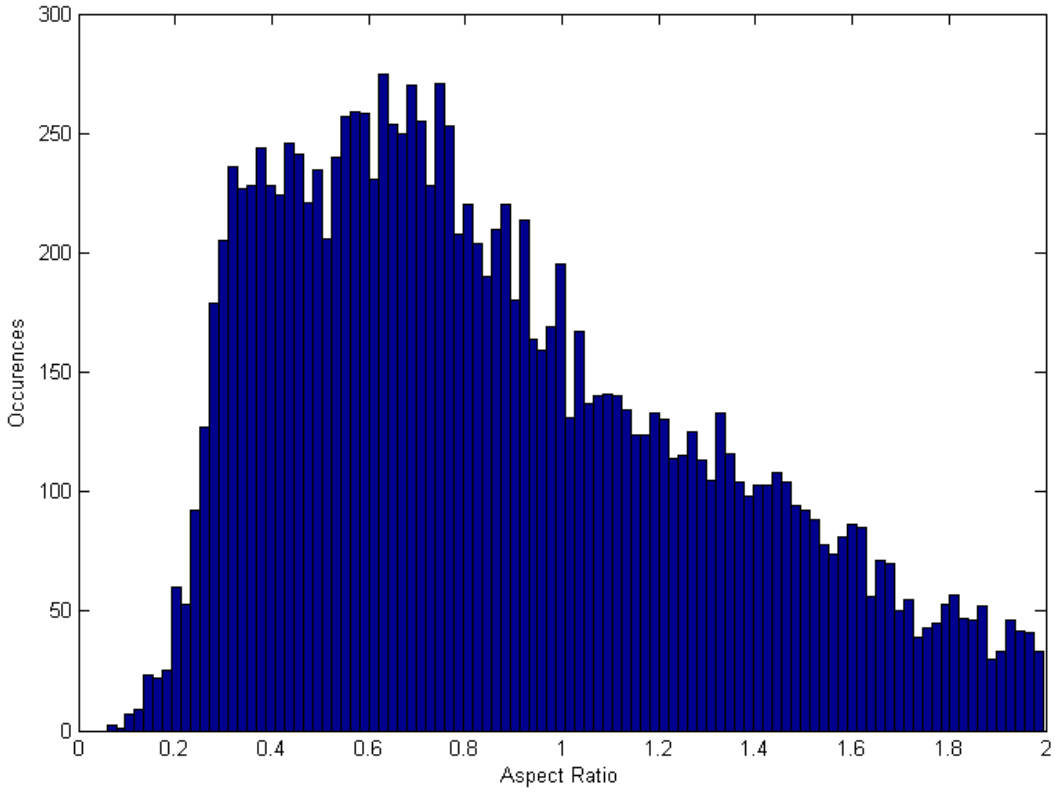

Figure 2: Histogram of Aspect Ratios Between 0 and 2

Figure 3 shows an alternate visualization of the aspect ratio data. In this case, the data is displayed as an X Y scatter plot in the image size. Note that here, images with similar aspect ratios will appear as points on a line which intersects the origin. The image shown depicts only the data for the “planes” class, since plotting all positive training samples would be impractical. This visualization allows for an intuitive way to view the data and select the appropriate *k* for clustering. In Figure 3, we can see reasonably clearly that there are two aspect ratios that dominate the “planes” class.

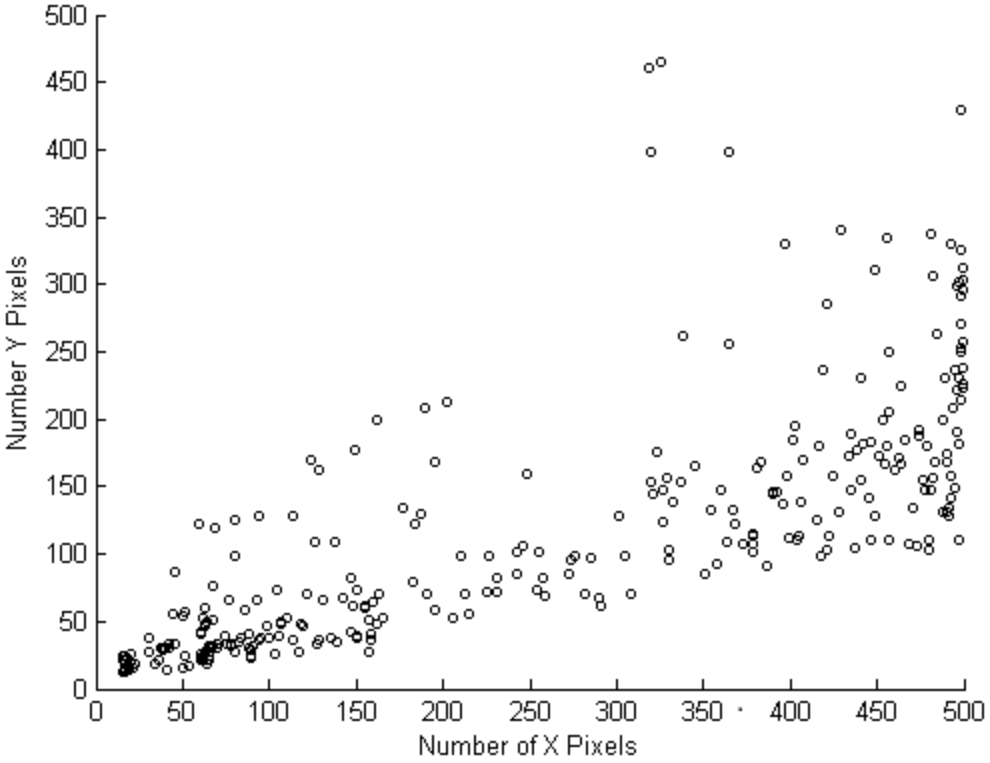

Figure 3: Scatter Plot of Positive Example Sizes for “Planes” Class

As of time of writing, no further results from the code have been generated.

Contributions:

Juan Guzman: Presentation / Poster Session Garbage.
Input parsing and transformation.
Isaac Jones: Project Written Report.
HoG processing.
Michael McGrath: LDA Coding.
Code integration.