# Assignment 4: Supervised learning in computer vision

October 7, 2009

In this assignment, you will develop a number of classifiers for handwritten digits. You will use the supervised learning paradigm, in which you are given some examples of two different classes: handwritten 3's and handwritten 5's.

You will be expected to develop classifiers, using the "training samples", that classify the "test samples" as well as possible. **Important:** You will *not* be graded on how well your classifier works on the test data, but rather, on whether you implement the classifiers as described.

1. Download the file digits.mat from web page. Use the command 'load digits.mat' to load it into matlab. Type 'whos' to see the variables that are defined in it. You should see four variables named train_threes, train_fives, test_threes, and test_fives. They are each a group of 50 images stored in a three-dimensional array. Try plotting a few of the images using imagesc to make sure they look good.

2. Write a function Euclid_dist.m which calculates the Euclidean distance between two images that are the same size. As long as they are the same size, it should work for arbitrary sized images.

3. Now write a function nearest_neighbor.m which takes four sets of images (for example, the ones you loaded from digits.mat). The first two arguments should be training images for class A and class B. The last two arguments should be test images which are members of class A and class B. The function should return how many of the test images were classified correctly using the nearest neighbor method on the training images for each class. For example, suppose that the first test image (which is actually in class A) was classified by the nearest neighbor classifier as class B. This would be an error, i.e., the image was not classified correctly.

4. Write another function knn.m exactly like the previous one, but that uses k-nearest neighbors instead of nearest neighbor. It should take the value of k as an additional argument.

5. Next, you will be writing a classifier based on maximum a posteriori classification. To start with you will be developing a feature function called left-right-weight.m. This function should take an image as its only argument and return a

number called the left-right-weight. The left-right-weight is defined as follows. First find the leftmost and rightmost "on" pixels in the image. Then find the x-position which is half way between the x-value of the rightmost pixel and the leftmost pixel. Finally, calculate the *percentage* of pixels in the image which are strictly to the right of the midline, defined by the halfway point between the leftmost and rightmost pixels. If the midline falls between two pixels, then find the percentage of pixels to the right of that midline. If the midline falls on a pixel column, then do not include that column in the calculation of the pixels on the right. This percentage is the left-right-weight.

6. The left-right-weight is a percentage, so it can take on any continuous value, depending upon the image size. Write a function called estimateLikelihood.m which takes a set of training images, and a number of bins for the left-right-weight, and returns a vector of probabilities corresponding to the frequency with which the left-right-weight values fell into each bin. For example, for a set of 50 training images, if the number of bins was 2, then estimateLikelihood should return the frequency with which the left-right-value was less than 50% and greater than 50%. Remember that these frequencies should add to 1.

7. Write a function called BayesRule.m which takes the likelihoods for each class (as a matrix) and the prior probabilities for each class (as a vector) and returns a matrix of posterior probabilities.

8. Finally, use all of these tools to write a function MAP.m (this stands for maximum a posteriori classifier) which uses the training data to estimate likelihoods, assumes that the prior for both classes is even (.5 and .5), and classifies the test examples. You should run this for 3 different values of the "bin" parameter in the estimateLikelihood function. Use values of 2, 10, and 100 for the bin parameter.

For the write-up, report the accuracy of the nearest neighbor classifier, and the k-nearest neighbor classifier for at least two different values of k. Also report the accuracy of the MAP classifier for each different value of the bin parameter. Did any of the classifiers perform worse than you expected? Better?

Also, turn in all the matlab functions you wrote.