Assignment: Greedy feature selection by a Bound on Information Gain

September 17, 2013

In this assignment, you will compare a greedy (more efficient) method of selecting k features from a set of possible features to an exhaustive search for the best k features.

You will use the same training and testing sets of 3's and 5's from the previous assignment.

1. Download the file digits.mat from the course web page. Put the file in your current working directory. 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 appear as you expect.

2. Write a function miPixelClass() to *estimate* the mutual information between the brightness value at a particular pixel location (either 0.5 or 255.5) and the class label. It should take the pixel position as a 2x1 vector of coordinates. It will be an estimate because it will be based on estimates of the joint probability distributions of pixel values and class labels. Use this functino to find the pixel with the highest mutual information with the class label.

3. Now write a function mikPixelsClass() that takes a set of pixel locations in the form of a 2xk matrix, and finds the mutual information between all of the pixels jointly, considered as a vector random variable, and the class label.

4. Use this function to find the best set of k features with the highest mutual information with the class label, for values of k equal to 2 and 3. If your program is too slow to run for $k = 3$, you may want to avoid choosing pixel positions that are always on or off, since they can't contribute to your mutual information. This should significantly speed up your code. Also, try to find ways to avoid using for loops, as they will significantly slow down your code.

5. Next, write a function informationGain() which takes two pixel locations, and returns the information gain about class from the first feature given the second feature.

6. Now, write a function nextBestFeature() which takes a list of features that have already been chosen, and finds the next best feature by maximizing the minimum information gain over the previous features, as explained in lecture.

7. Use the nextBestFeature() function to find the best k features for $k = 2, 3, 4, 5, 6$. Show an image of the average 3's with the 6 best pixel positions plotted as the numbers 1-6. Show the same features on an image of the average 5's.

8. Finally, report the accuracy of all your classifiers: using the single best, 2 best, 3 best, and greedy 2 best, greedy 3 best,..., greedy 6 best.

9. Turn in the results of your classifier and all your source code in a .pdf file. **When you zip your results to email them to Rae, please name the zip file with your name, so she will know who sent it when she saves your file.**