Machine Learning Final Project
Support Vector Machines for Face Recognition

By Michael Cheng, CWID: 10820067

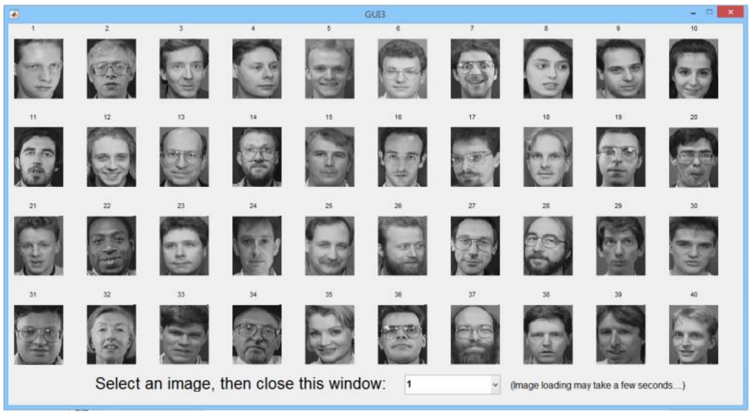
Method:

In order to apply SVM to images, I have to figure out the right features to extract from the images. Features are needed so we can plot feature elements on the feature plane/space and treat them as “data”. These features also have to be able to be applied SVM robustly and easily.

I remembered from my Computer Vision course this semester we learned about SIFT (Scale-Invariant Feature Transform) features. Obviously, SIFT detects and describes local features on images, but I will not go over the details of the concepts of SIFT. The notable thing is, I will be using the open source computer vision package VLFeat for SIFT feature related procedures.

The following are the stages of my SVM face recognition program:

1. GUI Setup:
   The program starts with a GUI window with all the test images displayed and numbered. The user can pick one test image or choose to test all 40 test images at once and check the performance.



Select an image, then close this window: 1 (Image loading may take a few seconds....)

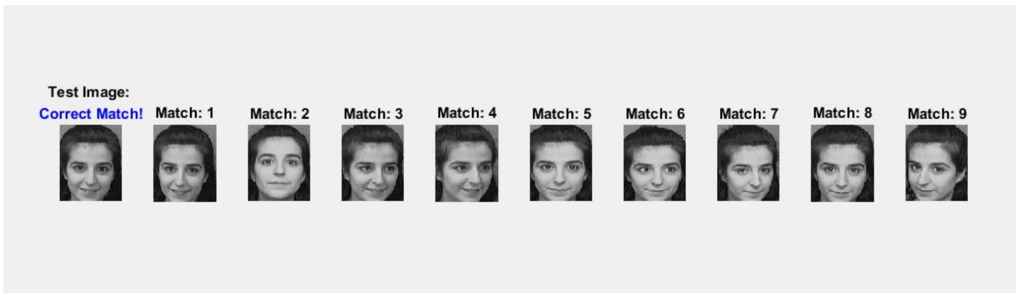
2. Store Face Image:
   Store all the image in the data base to an array, labeling the person and the number of the image of that person.
3. Generate “Words” and “Vocabulary”:
   This stage utilizes the VLFeat package. First, use the "dense SIFT" function to get the descriptors features across the whole image uniformly (Normal SIFT will pick the feature it likes, we don't want that). Store all the descriptors of all the images, then apply K-means clustering, setting K = 1000. Each cluster can be viewed as a “visual word”, and these 1000 words make up the “vocabulary”. With the vocabulary and the decision tree of each word, we can go back to each image and observe which words from

the vocabulary were used in each image.

4. Histogram:
   The histogram of an image is basically a word count of the image. It store how many times each word out of the 1000 word vocabulary is used in this image. Histogram is a great feature format to apply SVM. Also, generate the histogram for the test images.
5. SVM Data Training:
   Rather than using complex methods like SMO to find the alpha values to generate the weights, the method I use for SVM data training is to use the gradients of the Hinge loss function, accompanied with the basic law/constraint of the classifying function to "move" the weights gradually to the right direction. I make the algorithm run for 5 iterations with a learning rate value of 0.005.
6. Brute Force Method:
   As the name sounds, this is utilizing fast computers at its best. I run SVM data training for each possible pair of people out of the 40. Every pair of SVM training will generate a set of weights for test image classification. This means we will have 40x40 weight vectors generated. The dot product of the weights and the histogram of the test image will show which person of the pair the image is more similar to. Record all the 40x40 outcome results and sum up the amount of time the image has been classified to each person, the person with the most classification count is the match.
7. Display Results:
   Show if the match is correct.

Results:

The results were pretty great. Since there are a couple of variables that are chosen randomly throughout the code, the vocabulary and words of each image will be different every time, causing really small chances of different outcome. For most cases, the program can generate results that are 100% correct. Sometimes it will have misclassifications, but no more than one or two out of the forty at most.



```
Number of correct matches: 40
Number of wrong matches: 0
```