CSCI-B 657

Computer Vision

A3

Spring 2016

Indiana University Bloomington

Sumit Dey | skdey@umail.iu.edu
Rajeev Meda | rmeda@iu.edu
Anand Sharma | asaurabh@indiana.edu

March 27, 2016

Running the code:

1. Go to the project directory and build the project using ‘make’.
2. To get the output for assignment questions run the following commands:

Part 1:

Question 3: **Train**: ./a3 train svm **Test**: SVM: ./a3 test svm

Part2:

Question 1- **Train**: ./a3 train svm P **Test**: ./a3 test svm P

Question 2- **Train**: ./a3 train svm H **Test**: ./a3 test svm H

Question 3-

Train: ./a2_3 train bow **Test**: ./a2_3 test bow

Part 3:

Note**:** This part requires the OverFeat package and due to github restrictions we were not able to push the package to repository. Hence to run this part, please ensure that OverFeat packing is present in the directory of project. The packing should be inside a folder named ‘overfeat’.

Train: ./a3 train cnn **Test:** ./a3 test cnn

Part 1-

1. SVM The accuracy of colored images was higher than the gray scale images. The possible for this behavior is that colored images contain more variations along the different channels, gray scale causes loss of information.

The image is resized to 40 * 40 pixel image, this is done to ensure that there are equal features for all images. Features are then converted to a 1600 dimensional vector. The train features are written to a file called "train_features.dat". The format of the file is as follows:

Every line of the image corresponds to features of on image. Format of a line is follows:

<target> <feature>:<value> <feature>:<value> ... <feature>:<value>

Here, <target> is the corresponding class of the image, <feature> is the number assigned to the feature starting from 1 and with an increment of 1, <value> this is the value of the feature.

Accuracy of colored images is: 19% whereas accuracy of gray scale images is: 10%

Part 2:

1.

We have used the **sklearn** for Eigen value decomposition. The process we followed was to first compute the PCA for all training examples, flatten them, subsample to different dimensions, for example for 900 PCA component we tried the 30X30 as subsample size. [Components are provided in the code for the same how to make changes to these parameters.]. It will generate an svm model : *svm_model_3*

We did the same preprocessing for the test data. The Eigen value decomposition convergence was depended upon subsample size as well as the fit method used.

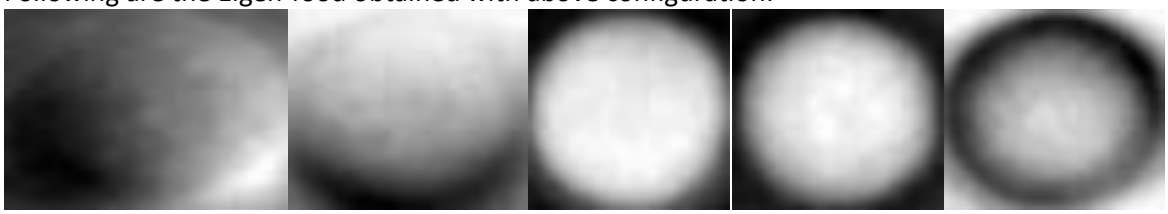
For sample run: number of components 900 and subsample size 30X30.

Performance:
Final epsilon on KKT-Conditions: 0.45547
Upper bound on duality gap: 0.04677
Dual objective value: dval=1.38211
Primal objective value: pval=1.42888

Total number of constraints in final working set: 179 (of 232)
Number of iterations: 233
Number of calls to 'find_most_violated_constraint': 46250
Number of SV: 173
Norm of weight vector: |w|=1.66333
Value of slack variable (on working set): xi=0.11755
Value of slack variable (global): xi=0.45547
Norm of longest difference vector: ||Psi(x,y)-Psi(x,ybar)||=8285.70860
Runtime in cpu-seconds: 84.50
Final number of constraints in cache: 6250

Accuracy: Classifier accuracy: 17 of 250 = 6.8% (versus random guessing accuracy of 4%)

Following are the Eigen-food obtained with above configuration:



Following table captures the details about the accuracy:

	N=900	N=400	N=100
C=0.01, e = 0.5	6.8%	9%	11%
C=0.1, e=0.5	6.8%	8%	11%
C=1, e= 0.5	4.5%	5.6%	6.4%

Command To Run: train:: ./a3 train svm P test:: ./a3 test svm P

We can improve the accuracy by doing the following:

1. Zero normalize the image before computing the PCA.
2. Subsample with higher size.
3. Apply convolution, or Gaussian noise elimination (wrap around at boundary).

2:

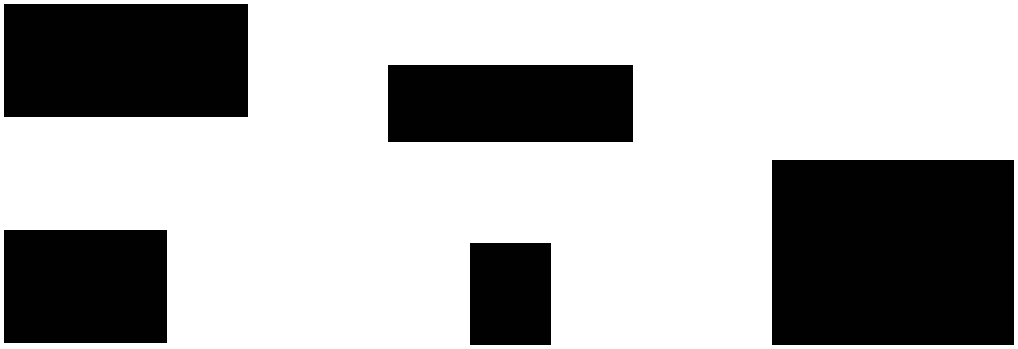
For this exercise we have implemented the following process:

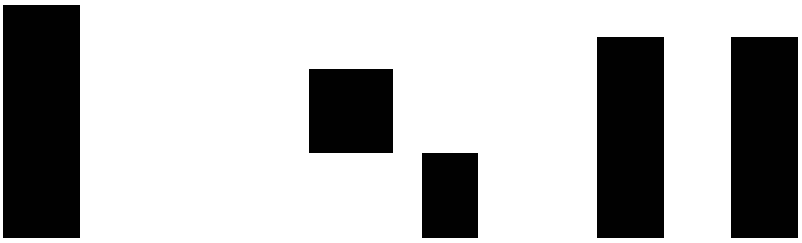
1. Gray scale the image.
2. Z-normalized the image.
3. Compute integral.
4. Resize the image subsample.

5. Extract the Harley features using different Harley patterns. Using the 15 different patterns.

We have achieved the following accuracy of up to 10% to 12%, using different parameters like: C=0.01, e=0.5 C=1,e=0.5.

Following are the different Harley features we have used:





3

Bag of Words Model

Build a visual vocabulary

1. First we resize all the images into either 40*40 or 100*100.
2. We first get SIFT descriptors for all each images in the train folder.
3. Then cluster all the 128D SIFT descriptors into 500 clusters using K-Means algorithm. We used kmeans function from opencv for this operation.
4. We save the centers of these 500 clusters. These represent the visual vocabulary for our model.

Represent each image as a histogram of the visual vocabulary.

1. First we calculate the SIFT descriptors for the image.
2. Then look for the nearest visual word for each descriptor and generate a histogram for each image. We represent histogram as a vector of integers.

Use SVM to train the vectors containing histograms of each image. For this we used SVM Multiclass.

Then Use the trained model to classify any new image given.

Some things to note before running the code for this part.

1. The K-Means clustering method takes upto 20 minutes for 40*40 resized image and over an hour for 100*100 resized images. Hence we included centers_40.yml in the folder which contains the centers calculated using K-Means. **The code for part 2.3 is in the file a2_3.cpp and kmeans part is commented out initially. So please make sure you place centers_40.yml in the folder before training SVM.**
2. The file for trained SVM model is **bow_svm_model.** If you want to test without training, please place it in the same directory.

3. We are using open CV for KMeans clustering and the directory path of OpenCV library is hard coded. If you are not running it on Silo/Tank please ensure that correct path is set up of the library.

The accuracy for this model is 4%.

Part 3:

In this part we extract images using the OverFeat package. As in part 1, we resize the image 231 * 231 pixel image, this is done to ensure that there are equal features for all images. The accuracy of this model is 69%.