CHARACTER RECOGNITION -PRELIMINARY PROJECT REPORT

Group Members: Adarsh Barik and Mohith Murlidhar

As proposed in phase 1, we undertook the problem of identifying characters in images of natural scenes. Our focus is on recognizing characters in situations that would normally not be handled very well by traditional Optical Character Recognition (OCR) techniques. As indicated in our report from Phase 1, our experimental setup involved implementing the "Bag of Visual Words" methodology for the vector representation of the training image and training an SVM model using these vectors and given labels. We have implemented this and performed tests on the known samples by dividing the training data , where 60% was used for training , 20 % was used for validating the model and 20 % was used for testing the model [3]. Testing on unknown label samples will be done for the next phase.

The report consists of detailed explanation of the dataset, the experimental setup and results from our running our tests on the known label samples.

The Data Set

The Chars 74K data set consists of 3 categories of images- the English Data Set (natural images), the Font Data Set and the Hand- Printed Data Set. As per the Kaggle Competition, analysis was done on only the English Data Set. Its consists of 62 classes, formed by using upper and lower case characters, which were treated separately and including the digits from 0-9. The training and test data sets consists of images which were resized to 20X20 pixels (as provided). The training data set consists of 6283 images and corresponding to which are 6283 known labels. The unlabelled test data set consists of 6221 images. For this preliminary report, we don't use unlabelled test data set.

The Experiment Setup

The dataset (training and test) combined, consists of 12,503 images, which needed to be mapped for the object category recognition problem at hand. The methodology we used to do this is the Bag of Visual Words methodology. This method was first proposed in the text retrieval domain for text document analysis and was further adapted for computer vision applications. For image analysis, as applied in our case, we create a visual analogue of the word, formed by quantizing the continuous high dimensional space of image features to a manageable vocabulary of visual words. The detailed steps carried out to implement this methodology are as follows:

1. **Region detection and feature extraction**- this required us to perform image processing. For this project, we decided to proceed with the 6 different types of features proposed by the paper i.e Shape Context (SC), Geometric Blur(GB), Scale Invariant Feature Transform(SIFT), Spin Image, Maximum Response of filters (MR8), Patch

Descriptor (PC H). We used the SIFT features for our analysis right now, and we will work with the other descriptors, if time permits.

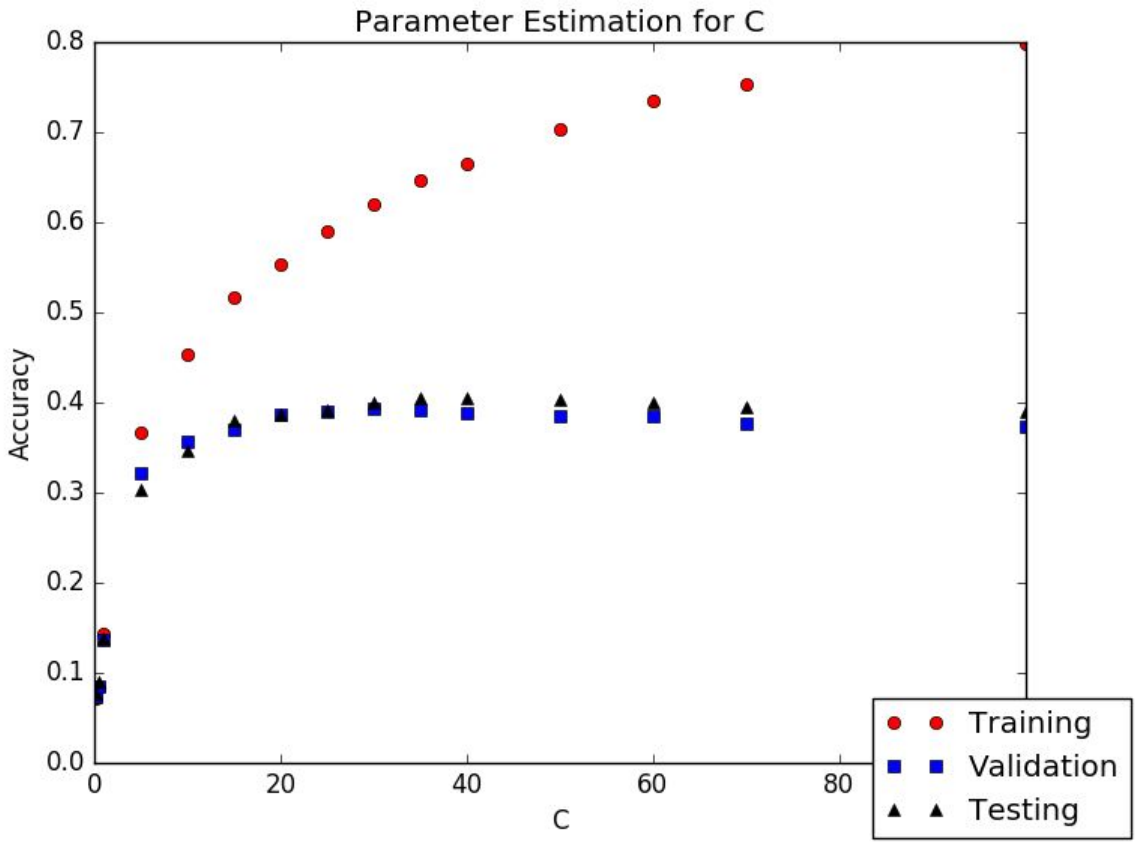
The SIFT features [1] are features extracted on regions located by the Harris Hessian-Laplace detector, which gives affine transform parameters. The feature histogram is computed as a set of orientation histograms on (4x4) pixel neighborhoods. The object histograms are relative to key-point orientation. The histograms contain 8 bins each, and each descriptor contains a 4X4 array of 16 histograms around the key-point. This leads to feature vectors of 128 elements.

We implemented this by using David Lowe's SIFT Descriptor Demo [4].

2. **Bag of words using K-means** - The low level features collected were grouped into a specified number of clusters using the K-means unsupervised algorithm. In our experiments, there are 5 visual words per class, leading to a vocabulary of 310.
3. **Vector Quantization** - Each feature extracted from an image was then mapped to the closest visual word determined by the clustering algorithm and each image was represented by a histogram over the vocabulary of visual words.

Model implementation and Analysis

1. **Model generation -** In this preliminary report, we use one-vs-one (all pairs) multiclass SVM as classification algorithm. In future, we'll be testing the code with other variants of SVM. The implementation is done using python module sklearn.svm.SVC from Scikit learn [5]. We generated our classification model by training over 3770 labelled training set. For now we have used rbf kernel in our configuration however we are planning to compare it against different kernels.
2. **Hyper-parameter estimation (Penalty parameter C) -** Once the model is generated we cross validated the model using 1257 labelled validation samples. We generated a plot between accuracy against training samples and validation samples for various values of penalty parameter C and chose the value which results in good validation accuracy. For current setup it comes about C=30 with 39.29% accuracy. Plot is shown below (we also report test accuracy over a test sample of 1256):



We'll estimate other parameters using cross validation in our final report.

3. **Testing -** For now testing has only be done against labelled data set. We'll test against unlabelled dataset during the next phase of project.

Future work

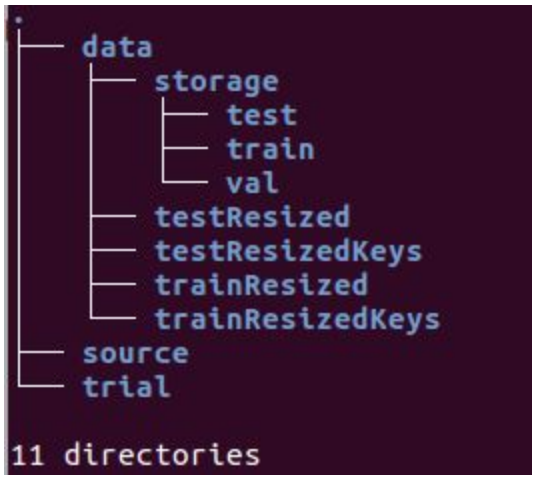
Below is a tentative blueprint of our future plans:

1. Generate and compare classification model for different types of SVMs such as one-vs-each
2. Study effect of changing kernels
3. More detailed hyper-parameter estimation for various kernels
4. Study effect of changing sample size on accuracy
5. Testing on unlabelled data set

How to run the code

1. The code is written using python in (and for) Linux machine. We have tested it with python 2.7+ and python 3.0+.
2. Code follows a particular directory structure to access and store relevant data. Please note that due to size limit we are not providing all the data but it can be accessed through https://github.com/Adarsh-Barik/CharRecognitionPy. Note: all the files sent via

Blackboard go inside "source" directory. Please make sure that below directory structure exists in your computer before running the code:



3. To run the code:
    a. Open terminal
    b. Go to "source" directory
    c. Run "python main.py" in terminal

Reference Papers

[1] David G. Lowe, **"Object recognition from local scale-invariant features,"** *International Conference on Computer Vision,* Corfu, Greece (September 1999), pp. 1150-1157

[2] T. E. de Campos, B. R. Babu and M. Varma, **"Character recognition in natural images"**, Proceedings of the *International Conference on Computer Vision Theory and Applications* (VISAPP), Lisbon, Portugal, February 2009.

[3] Andrew Ng, Coursera Lecture, Lecture 60 - Model Selection and Train/Validation/Test Sets, https://www.coursera.org/learn/machine-learning/lecture/QGKbr/model-selection-and-train-validation-test-sets

[4] David Lowe SIFT descriptor demo, http://people.cs.ubc.ca/~lowe/keypoints/

[5] Scikit Learn, http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html