**README**

This ReadMe document will describe in detail how to execute the program and also give a brief description of all the files and functions in the code.

**REQUIREMENTS**

1. Python 3.4 (This version will work with the cvxopt wheel attached with the code, for any other version of Python, a different one will be required)
2. PIL library (For image manipulations)
3. Numpy

Steps to install requirements:

* *$* sudo pip install –U numpy
* *$* sudo pip install –U cvxopt-1.1.7+openblas-cp34-none-win\_amd64.whl
* *$* sudo pip install –U numpy-1.11.2+mkl-cp34-cp34m-win\_amd64.whl
* *$* sudo pip install –U Pillow

**EXECUTION**

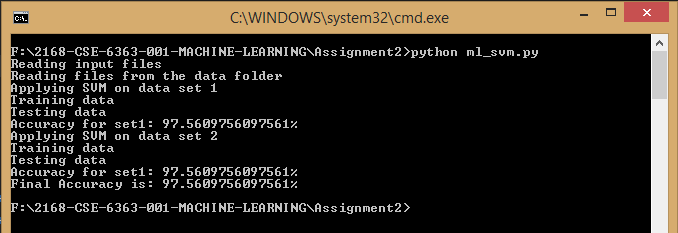
Once the dependencies are downloaded the program can be executed using the command:

*$* python ml\_svm.py

It takes around 4-5 minutes for all the files to be processed and final accuracy to be displayed.

**OUTPUT**

The output of the programs describes the basic steps being performed at a time, the accuracy at each stage and the final accuracy. Below is a screenshot of one execution:



Text version of output for better viewing:

Reading input files

Reading files from the data folder

Applying SVM on data set 1

Training data

Testing data

Accuracy for set1: 97.5609756097561%

Applying SVM on data set 2

Training data

Testing data

Accuracy for set1: 97.5609756097561%

**Final Accuracy is: 97.5609756097561%**

**CONTENTS OF THE ZIP**

The zip contains a data directory which contains all images used for training and testing the algorithm. cvxopt-1.1.7+openblas-cp34-none-win\_amd64.whl and numpy-1.11.2+mkl-cp34-cp34m-win\_amd64.whl which are used for quadratic programming (as mentioned above, these will work with Python 3.4 only). A ReadMe file which describes how to use the program and a python executable file called ml\_svm.py

**DESCRIPTION OF ml\_svm.py**

It contains the following functions:

* read\_pgm

This function uses the Python Image Library (PIL) to read the pgm file. The image matrix is then converted into a single array so it can be used as a single data point for the algorithm. Since each image is 92x112 in size, each array for an image is of length 10304.

* get\_data\_list

This function makes a python dictionary for the entire data, where the key is class name (s1 to s40) and the value is the list of images in each class. It returns a variable which holds the entire dataset for the algorithm.

* train

This function is used for training the data set it receives. The **one-vs-rest** strategy is used to train data where each class data is used to train against all the other classes present. Hence, multiple loops are used. It takes input as described above and uses it for calculating the values of alpha for each class using quadratic programming available through cvxopt library of python. A description of its input is given in a section below. Since all values of alpha are very low, the lower ones are ignored and only the bigger ones are used as support vectors. These values are then used to calculate b and w. The w array and b for each class will then be used to predict the class of testing files.

* predict

This function is used to predict the class of the testing image. For each image, 40 values are calculated for the 40 classes, and the list of these values is returned.

* svm

This function performs the training and testing of data.

* + It takes a parameter position which indicates which half of the data to be used for training, and the rest is used for testing. So when position is ‘top’, first 5 images of each class are used as training, and the other five for testing, and vice versa when the position is ‘bottom’.
  + After dividing the data, it calls the train function to get values of w and b for each class. This is stored in a variable.
  + The next step is prediction; each testing image calls the predict function, which provides the yi value for each class.
  + The class which has the maximum of these values is considered as the predicted class.
  + The actual and predicted classes are compared and accuracy is then calculated.
* main

The main function of the script, reads data, and calls the svm function using 2 different values of ‘position’ and the average accuracy is then calculated as the final accuracy, which is displayed.

**CVXOPT**

The cvxopt.solvers.qp function is used to solve quadratic programming. Its format is as follows:

cvxopt.solvers.qp(P, q, G, h, A, b)

The following values of the imputs are used for our SVM problem:

b is a matrix with a single value 0

A is a matrix of size 1x200 with the value of y. [200 is the number of samples in training set and y is the list of classes these samples]

h is a matrix of size 1x200 of all zeroes

G is a matrix of size 200x200, with all values as 1, except the diagonal values as -1

q is a matrix of size 1x200, with all values as -1

P is a matrix which contains the product of the dot products of x and x transpose and y and y transpose. [x is the input data set and y is the list of all classes for the dataset]