Introduction to Machine Learning *Sowmya Ramapatruni, BG80177*

HW3: SVM

Please note that only PDF submissions are accepted. We encourage using LaTeX to produce your writeups. You'll need *mydefs.sty* and *notes.sty* which can be downloaded from the course page.

Gradient descent algorithm:

1. Implement the gradient descent algorithm for binary SVM.

   $function[W, b] = grad\_svm\_binary(images_train, labels_train, images_test, labels_test, epoch, C)$
   This function takes train and test data as input and returns weight vector and bias . Function is placed in $part\_1.m$ file.

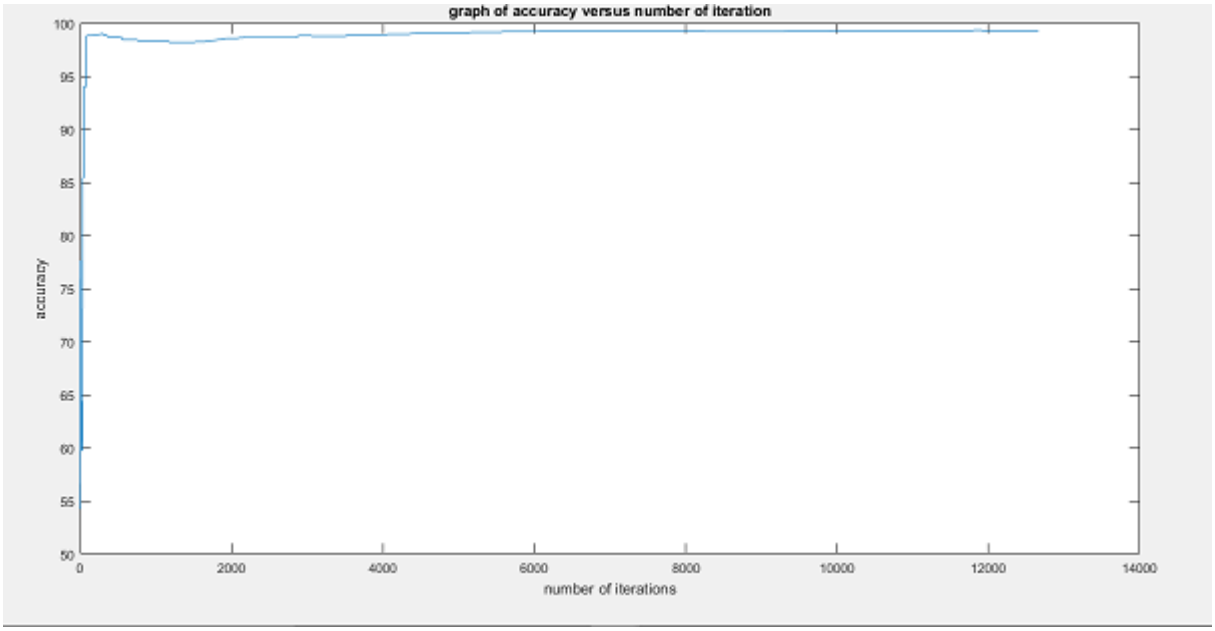
2. Similar to the previous homework, train and test it for classifying digits "1" and "6" in MNIST dataset.

   $function[W, b] = grad_svm_binary_plot(images_train, labels_train, images_test, labels_test, epoch, C)$
   $function[accuracy] = Predict(images_test, labels_test, W, b)$
   The above two functions are implemented for training and testing . The train function also plots the accuracy for each iteration. The predict function calculates the accuracy and returns it.
   The whole implementation is placed in $part\_2.m$

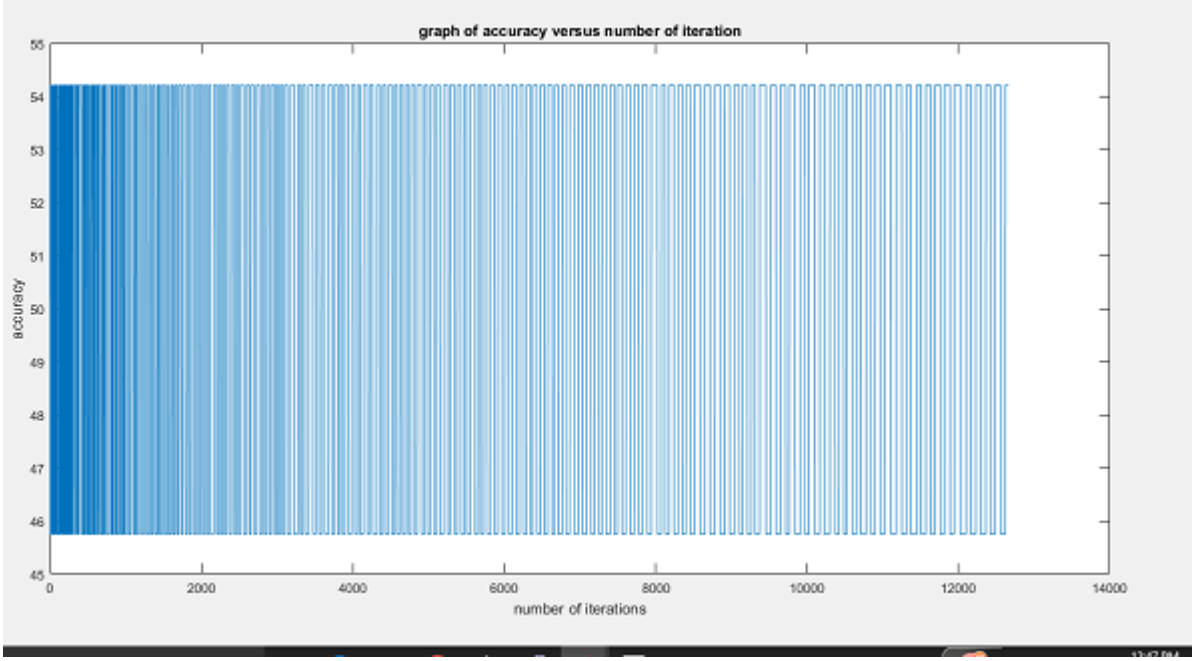
3. Plot the accuracy on the test set wrt. the number of iterations. Here, processing each data-point is considered one iteration. You don't need to use all training data if it takes a long time to run.

   (a) Play with the hyper-parameter C to see its effect in the accuracy and overfitting. For instance, try very large and very small values for it.

   plot of accuracy wrt iterations is placed in $plot\_3$. It is also pasted below. C value is taken as 0.01. and the learning rate is reduced promotional to iteration number.



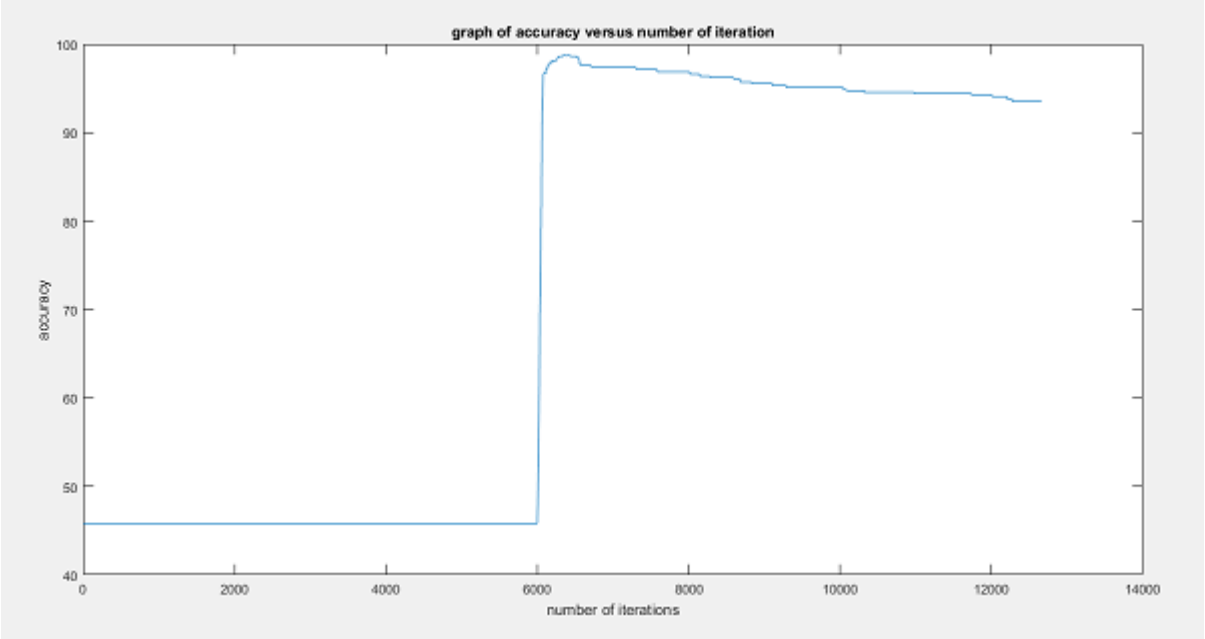
plot for larger C(100)



(b) Choosing the learning rate may be a little tricky. One popular strategy is to reduce it promotional to $\frac{1}{t}$ where t is the iteration number.

4. Sort the data before training so that all "1"s appear before "6"s and plot the accuracy wrt iterations. Is this faster or slower in training?

The implementation is placed in $part\_4.m$. And the plot generated is named as $plot\_4$.
The model took more iterations to generate a stable accuracy. After seeing the first few positive examples (maybe five), it would likely decide that every example is positive, and would stop learning anything. It would do well for a while (next 495 examples), until it hit the batch of negative examples. Then it would take a while (maybe ten examples) before it would start predicting everything as negative. By the end of one pass through the data, it would really only have learned from a handful of examples (fifteen in this case).
So one thing you need to avoid is presenting the examples in some fixed order.



5. Implement a function for 1-vs-all multi-class SVM that calls the binary SVM function.

> $function[W, B] = one\_vs\_all(images_train, labels_train, images_test, labels_test, epoch, C)$
> The above one_vs_all function iterates through all the available classes(0-9) and calls binary svm for each of them . This function returns a weight vector of size 10 * 784 and bias of size 10.
> The source code is placed in $part\_5.m$

6. Train and test it on all 10 digits in MNIST and report the confusion matrix as well as the average precision. $conf(i, j)$ is the number of images that are from category i and are classified into category j. You should normalize this so that all rows sum to one and then average accuracy is the average of its diagonal values.

> $[W, B] = one_vs_all(images_train, labels_train, images_test, labels_test, 1, 0.01);$
> $[accuracy, con_matrix] = predict_all(images_test, labels_test, W, B);$
> The above two functions are implemented. The predict functions calculates the confusion matrix and accuracy. Contents of confusion matrix for C = 0.01 is pasted below. The average accuracy obtained is around 86 .

con_matrix

10x10 double

	1	2	3	4	5	6	7	8	9	10	11
1	0.9015	9.4697e-04	0.0180	0	0.0095	0.0142	0.0123	0.0028	0.0256	0.0152	
2	0	0.9820	0	9.0009e-04	0.0018	0.0018	0.0018	0.0036	0.0036	0.0045	
3	0.0021	0.0072	0.9176	0.0196	0.0021	0.0041	0.0103	0.0093	0.0237	0.0041	
4	0.0039	0.0071	0.0212	0.7398	0.0063	0.0956	0.0016	0.0063	0.0933	0.0251	
5	0	0	0.0073	0.0021	0.8774	0.0147	0.0105	0.0052	0.0346	0.0482	
6	0.0068	0.0027	0.0027	0.0150	0.0027	0.8840	0.0314	0.0014	0.0409	0.0123	
7	0.0031	0.0051	0.0134	0	0.0093	0.0247	0.9176	0.0010	0.0257	0	
8	0.0060	0.0026	0.0299	0.0154	0.0068	0.0257	0.0034	0.8186	0.0248	0.0667	
9	0.0044	0.0206	0.0279	0.0015	0.0073	0.0235	0.0015	0.0015	0.9104	0.0015	
10	0.0028	0.0028	0.0176	0.0130	0.0918	0.0158	0.0019	0.0362	0.0594	0.7588	
11											
12											
13											

7. Look at top mistakes and show images along with ground truth label and the predicted label to see if they make sense.

The implementation is placed in $part\_7.m$. The plot of top 10 mistakes are plotted and placed in $plot\_10.m$. Sample figure is pasted below.