Introduction to Machine Learning *Sushant Chaudhari*

HW2

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.

1. Explain in what case and why we need a regularizer and why l_2 norm regualzrizer is a reasonable one.

   When we do not have the number of required training samples or the model complexity is large, so we use regularizer to avoid over-fitting. The need to regularize a model will tend to be less and less as you increase the number of samples that you want to train the model with or your reduce the model's complexity. However, the number of examples needed to train a model without (or with a very very small regularization effect) increases [super]exponentially with the number of parameters and possibly some other factors inherit in a model. L2 regularizer penalizes large weights so that tends to improve generalization, because it means that no input dimension can have a very large influence on the scores all by itself.

2. What values of λ in the regularized loss objective will lead to overfitting? What values will lead to underfitting? Note that λ is a multiplier for the regularizer term.

   When the value of λ is too small i.e. zero for instance which is the unregularized case, it will lead to overfitting also known as high variance. Whereas if the value is too large it will lead to underfiiting or high bias, as the weights are forced to be very small in this limit.

3. Explain why the squared loss is not suitable for binary classification problems.

   The squared loss penalizes large residuals a lot more. If we use squared loss, it is not robust to outliers. We cannot move the farthest away point arbitrarily farther out without affecting the median. Hence it is not suitable for binary classification. An alternative to the squared loss is the absolute deviation loss, which simply takes the absolute value of the residual.

4. One disadvantage of the squared loss is that it has a tendency to be dominated by outliers – the overall loss $\sum_n (y_n - \hat{y}_n)^2$, is influenced too much by points that have high $|y_n - \hat{y}_n|$. Suggest a modification to the squared loss that remedies this.

   One possible solution is to modify the squared loss to Huber loss function. The Huber loss function is convex in a uniform neighborhood of its minimum a=0, at the boundary of this uniform neighborhood, the Huber loss function has a differentiable extension to an affine function at points a=-delta and a=delta . These properties allow it to combine much of the sensitivity of the mean-unbiased, minimum-variance estimator of the mean (using the quadratic loss function) and the robustness of the median-unbiased estimator (using the absolute value function).

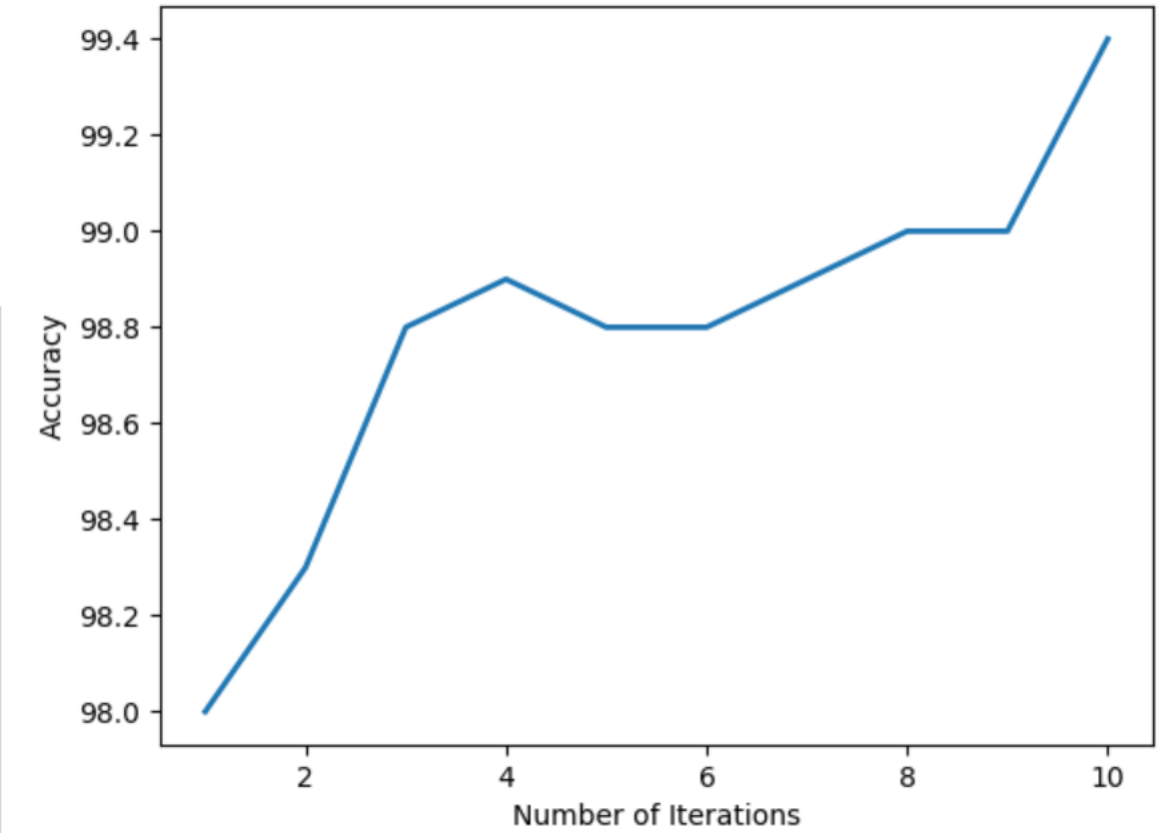
5. Perceptron algorithm:

   (a) Implement the perceptron algorithm for binary classification.
   Please refer the submitted code for this part.

   (b) Train and test it for classifying digits "1" and "6" in MNIST dataset. Note that we don't need the data of other digits in this part. Please use 1000 training examples and 1000 testing examples (500 for each class). report the accuracy after a reasonable number of iteration when the accuracy does not change or starts going down.

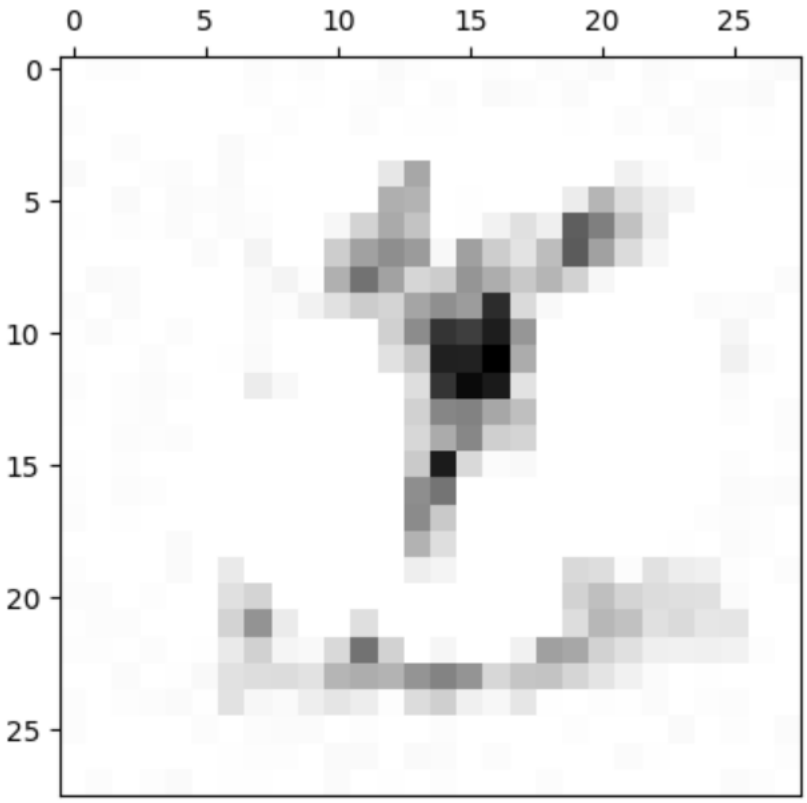
   I increased the number of iterations while testing and achieved an accuracy of 99.4 percent when the number of iteration reaches to 10. After this when I increased the number of iterations the accuracy started to drop.

   (c) Plot the accuracy on the test set w.r.t. the number of iterations. Here, processing each data-point is considered one iteration, so 1000 iterations means one pass over all training data. For this, you need to evaluate the model very often (once for each data point).

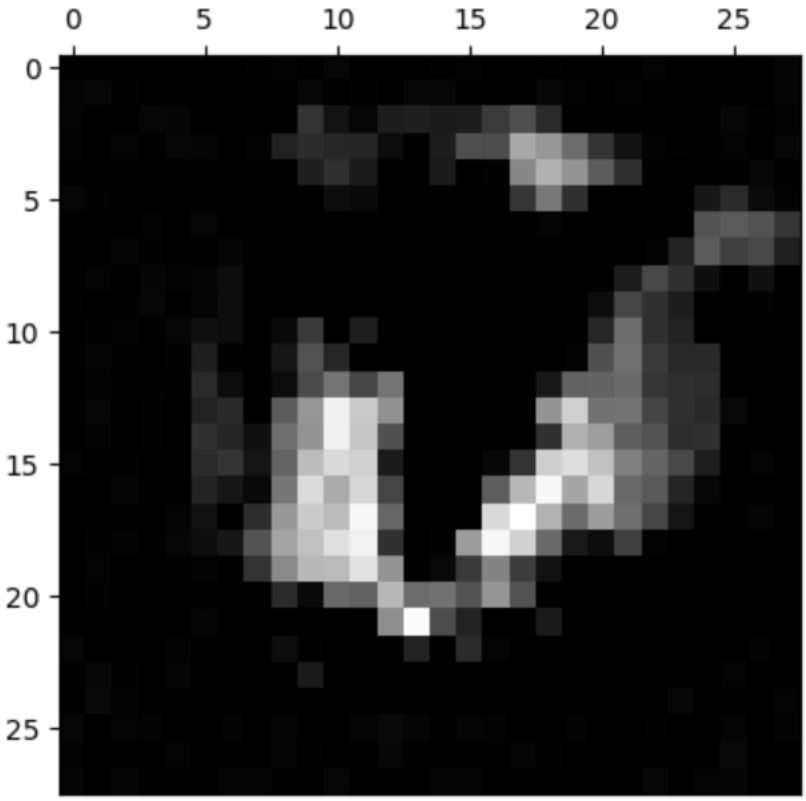


The X axis shows the number of iterations and Y axis shows the accuracy. As seen in the graph the accuracy goes on increasing as the number of iterations are increased with maximum accuracy achieved for 10th iteration after which it drops.

(d) Visualize the learned model in the image form to see if it makes sense. Note that the weight vector has both positive and negative values, so you should plot those on two separate planes. Note that you should use exactly the same testing data to be able to compare the results.



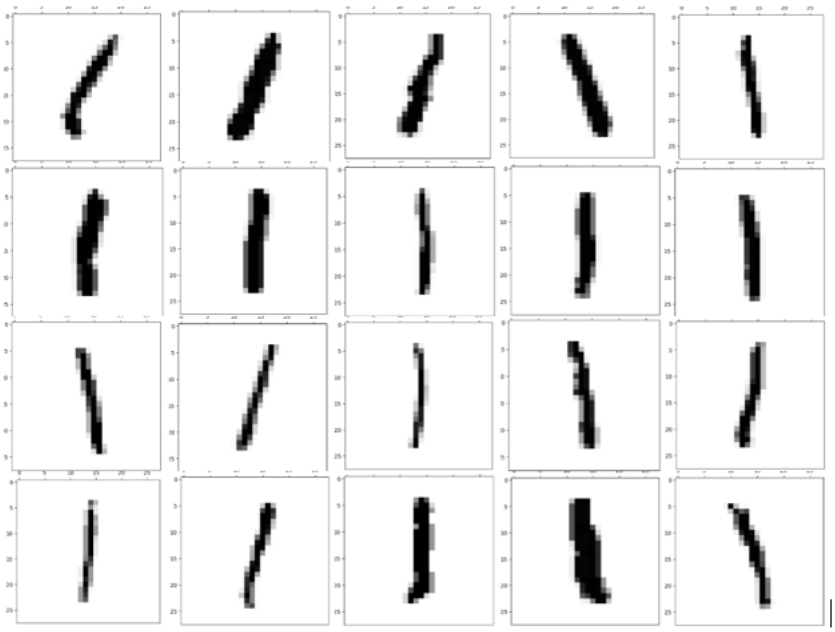
The above image shows the weight matrix plotted with all the positive values. As seen the image looks like 1 with all the possible combinations of 1.



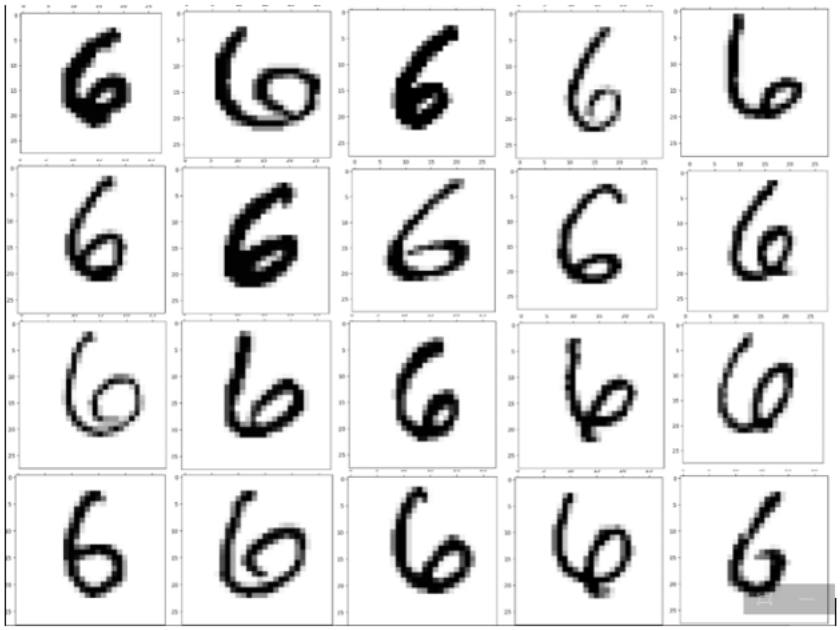
The above image shows the weight matrix plotted with all the negative values. As seen the image looks like 6 with all the possible combinations of 6.

(e) Visualize the 20 best scoring and 20 worst scoring images from each class.

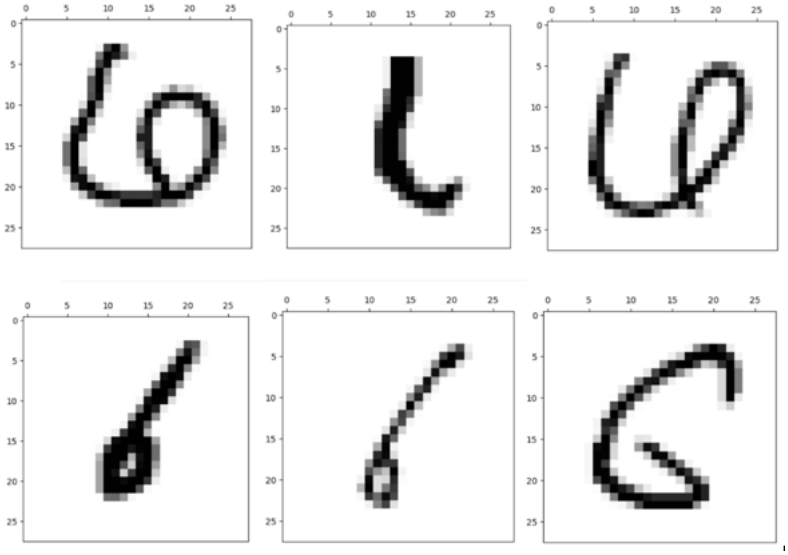
20 best scoring images for 1



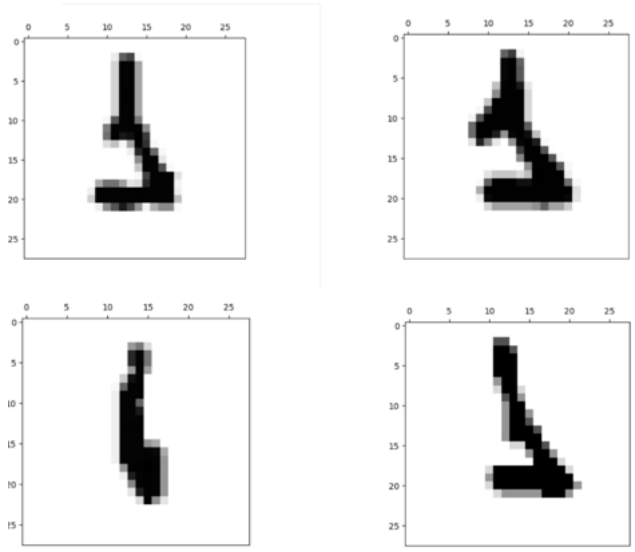
20 best scoring images for 6



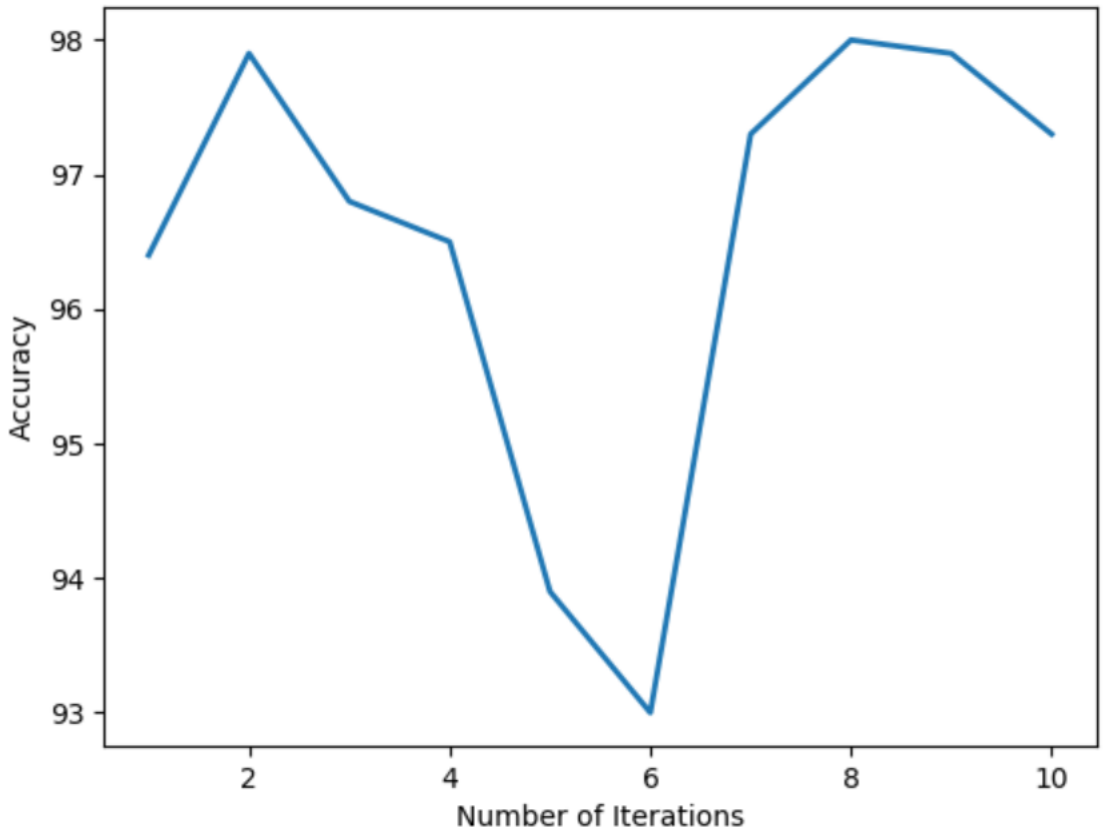
worst scoring images for 6. My code could only return few worst images as the accuracy was high.



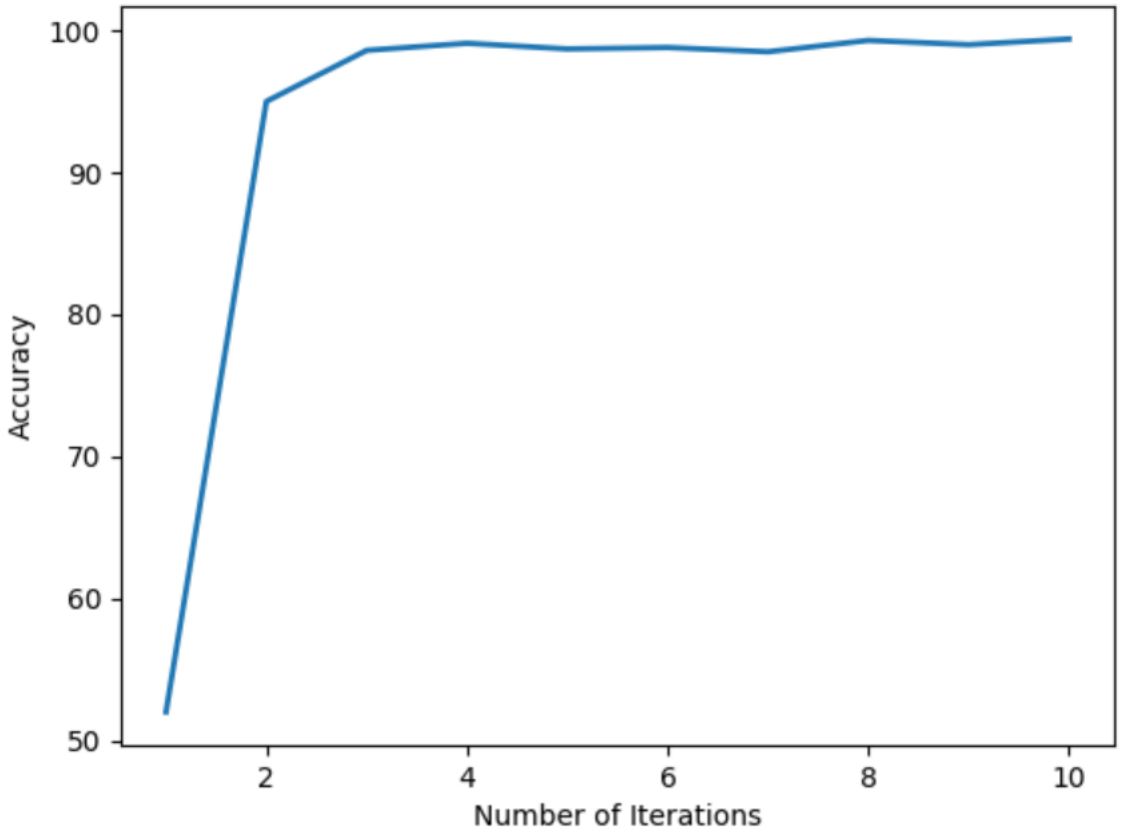
worst scoring images for 1. My code could only return few worst images as the accuracy was high.



(f) Randomly flip the label (add labeling error) for 10% of the training data and repeat (b) and (c). As seen below the accuracy drops when we introduce the labeling error and building a good model becomes slow.

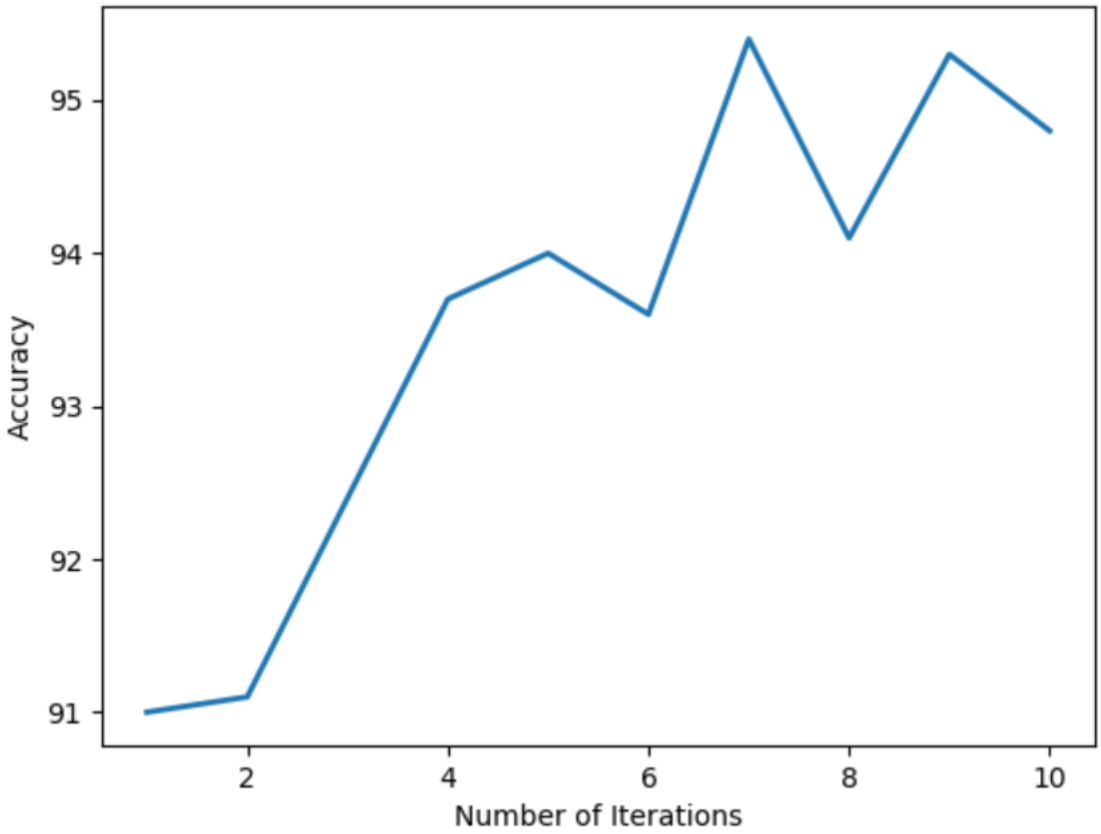


(g) Sort the data before training so that all "1"s appear before "6"s and plot the accuracy w.r.t. the number of iterations. Is this faster or slower in training? By faster, we mean the number of iterations needed to get a good model. Explain why.



As seen in the above graph the model is trained faster when the data is sorted. From the 4th iteration itself we achieve an accuracy of 99.00 and above. Te reason behind this is that when the data is sorted the weight vector is trained in less time and hence the accuracy is achieved higher with less number of iterations.

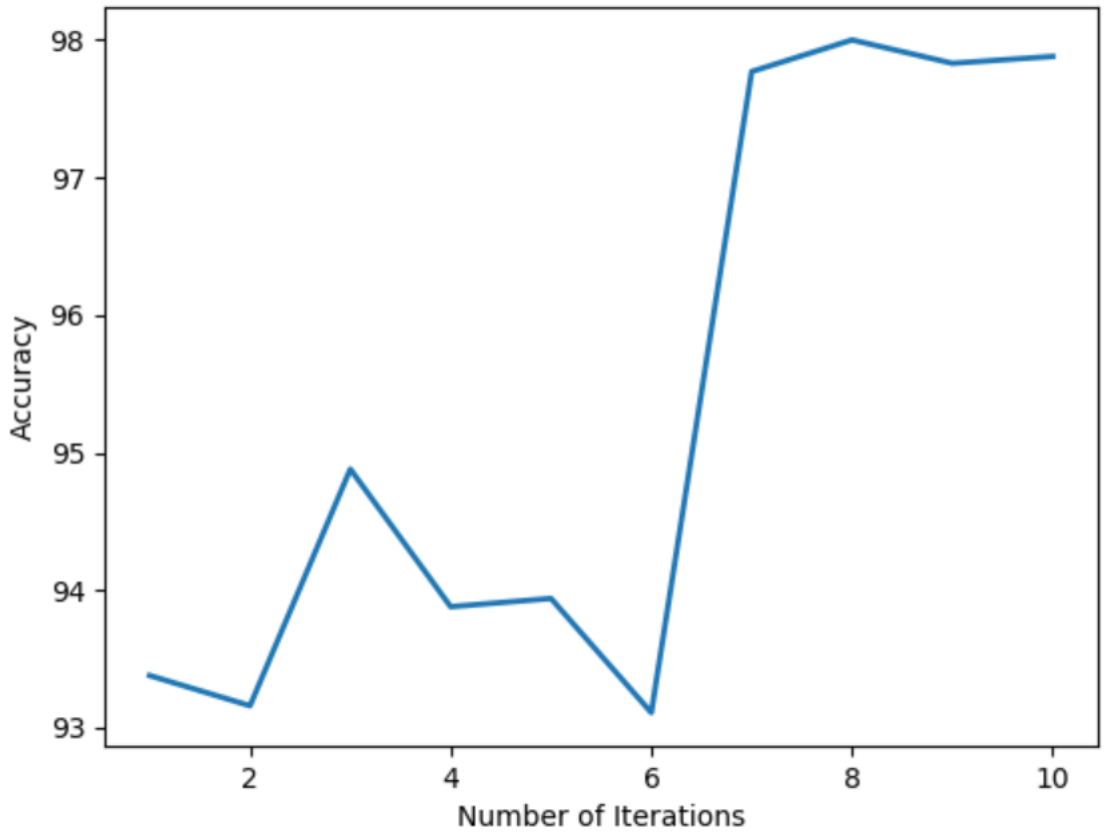
(h) Repeat (b), (c), and (d) for classifying "2" and "8". Is this faster or slower in training? Again, by faster, we mean the number of iterations needed to get a good model. Explain why.



[wi

As seen in the above graph the training is slower when the digits are 2 and 8. The reason being 2 and 8 almost have similar values in the weight vector and hence it requires more number of iterations to achieve higher accuracy. Highest accuracy achieved ranges around 95.4 which is less than the accuracy achieved for 1 and 6.

(i) Repeat (b) and (c) once with 10 training examples (5 for each class) and then all almost 12,000 training examples. Use the same test set that you used before.



As seen above it takes more number of iterations to achieve higher accuracy. Highest accuracy achieved is 98.00. So the model is trained slower in this case.