Introduction to Machine Learning *Mohit Khatwani,AJ75499*

HW2

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

- Regularization is a way to avoid overfitting of data. While training the model when unnecessary parameters are considered it tends to over-fit due to training of noisy data. In order to avoid this.
- It shrinks the model by reducing unnecessary parameters. L2 Norm basically is known as least square error. When using L2 norm it is easy to calculate the minima of error as it represented as tangent.
- A regularization term is added to loss function to penalize it so that loss function is reduced $\min_f \sum_{i=1}^{n} V(f(x_i), y_i) + \lambda R(f)$

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.

- Regularized loss objective will lead to overfitting when $\lambda = 0$. This means that if $\lambda = 0$ then loss function will act as unregularized, which will create a problem of overfitting.
- Regularized loss objective function will lead to underfitting when λ is very large. It will not align along the given points and thus over generalize.

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

- Squared loss is more commonly used in regression.
- Squared loss is represented as $V(f(\boldsymbol{x}), y) = (1 - yf(\boldsymbol{x}))^2$
- Squared loss is not suitable for binary classification problems because it penalizes the outlier data very extensively.
- Since value of $yf(\boldsymbol{x})$ is very large, regardless the sign of y and $f(\boldsymbol{x})$ it will penalize very large.

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.

- Squared loss can be modified to avoid inaccuracy by outlier data by adding weight normalization.
- A regularization function can be added to the loss function so that the model created can be shrinked and effect of outlier data is reduced.

5. Perceptron algorithm:

   (a) Implement the perceptron algorithm for binary classification.

   (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.
   Accuracy for classifying digits 1 and 6 is: 99.2 perc

   (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).

   Figure 1 shows Accuracy for classifying digits 1 and 6 with respect to number of iterations.

   (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.

   Learned model in image form for digits 1 and 6 is shown in figure 2 and figure 3.

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

   Figure 4 shows 20 best scoring images for number 1
   Figure 5 shows 20 worst scoring images for number 1
   Figure 6 shows 20 best scoring images for second number (6)
   Figure 7 shows 20 worst scoring images for second number (6)

   (f) Randomly flip the label (add labeling error) for 10% of the training data and repeat (b) and (c).

   Accuracy for classifying digits 1 and 6 with 10 percent random flip 93.89
   Figure 8 shows accuracy plot with label for 10 percent of data changed

   (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.

   Figure 9 shows accuracy plot with number of iterations of sorted training data.
   It is slower in training. We can infer from the graph that in sorted training data the model becomes its best at about 4000 iterations.
   While in shuffled training data the model gets learned at about 1000 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.

   Accuracy for digits 2 and 8 is 91.9 percent
   Figure 10 shows accuracy vs iteration plot for number 2 and 8
   Figure 11 shows weight vector visualization for first number: 2
   Figure 12 shows weight vector visualization for second number: 8.

   It is slower in training for numbers 2 and 8 because I think training number 8 is difficult to train for our perceptron in only 500 samples.
   Final Accuracy is also comparatively less than numbers 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.

   Accuracy with 10 training examples 90.0 percent.
   Figure 13 shows accuracy vs iteration plot for numbers 1 and 6 with 10 training samples

Accuracy with 12000 training examples 99.522 percent
Figure 14 shows accuracy vs iteration plot numbers 1 and 6 with almost 12000 training samples

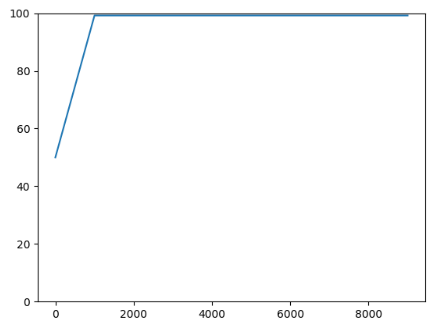

Figure 1: Plot for accuracy with respect to number of iterations. For number 1 and 6

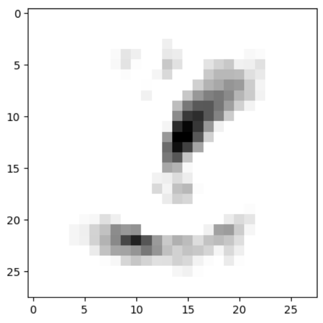

Figure 2: Learned model for 1

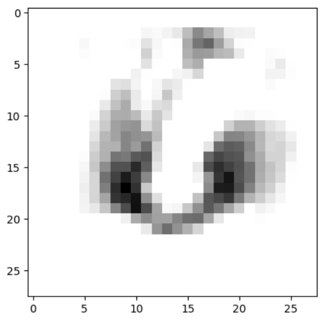

Figure 3: Learned model for 6

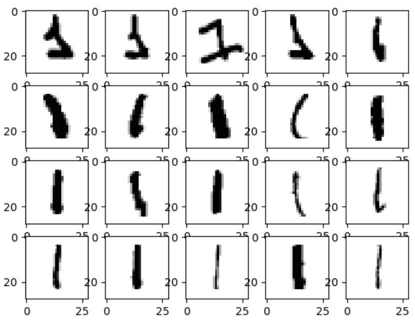

Figure 4: 20 Best scoring images for 1

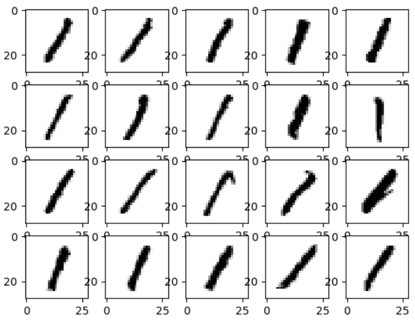

Figure 5: 20 Worst scoring images for 1

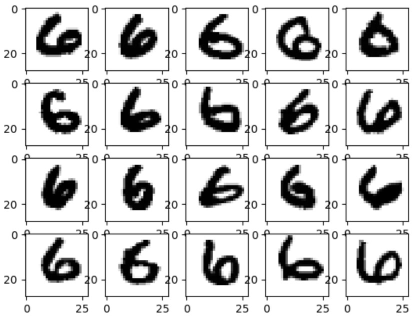

Figure 6: 20 Best scoring images for 6

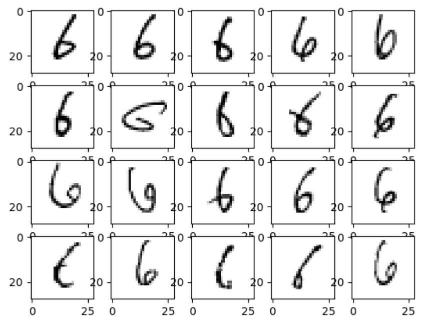

Figure 7: 20 Worst scoring images for 6

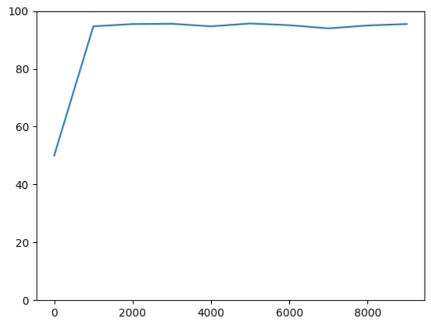

Figure 8: Accuracy plot with random flip of 10 percent of data

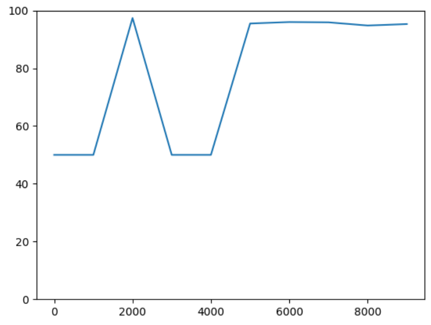

Figure 9: Accuracy plot with respect to number of iterations and sorted training data

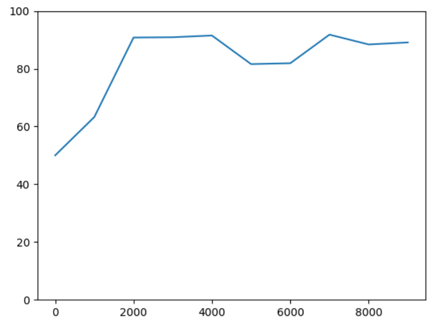

Figure 10: Accuracy plot with respect to iteration for numbers 2 and 8

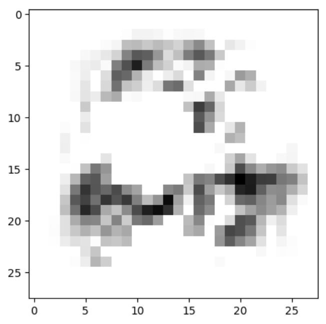

Figure 11: Weight vector visualization for first number: 2

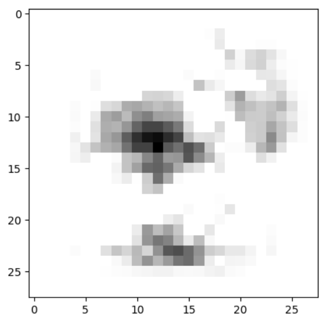

Figure 12: Weight vector visualization for second number: 8

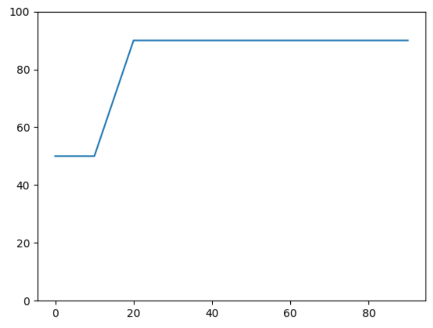

Figure 13: Accuracy vs iteration plot with 10 training samples

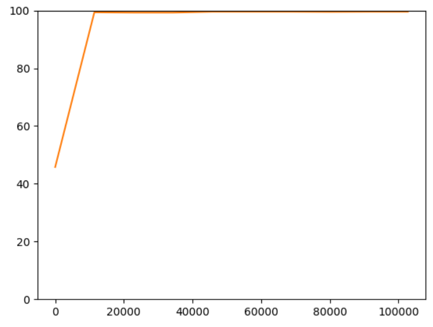

Figure 14: Accuracy vs iteration plot with 12000 training samples