Introduction to Machine Learning *CMSC 678, Fall 2018*

HW2

Please note that only PDF submissions are accepted. We encourage using LaTeX to produce your write-ups. 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 regularizer is a reasonable one.
   Ans: Regularizer is used to avoid overfitting. When the model tries to learn the noise in training dataset i.e. the data points that don't represent the true properties of the training dataset, there is high chance of overfitting. So, we need regularization to avoid the overfitting of model and increase accuracy.
   Regularization shrinks weight coefficients towards zero so as to avoid learning a complex or flexible model.
   L2 norm is a differenciable function. So, it is computationally easy to calculate minima. It is represented as tangent.

2. What values of λ in the regularized loss objective will lead to overfitting? What values will lead to under-fitting? Note that λ is a multiplier for the regularizer term.
   Ans: We use regularization to avoid over-fitting. So, if $\lambda = 0$, then there will be high risk of over-fitting as it would be equivalent to unregularized model. So, values of λ very closer to 0 will lead to over-fitting.
   On the other hand, higher values of λ will lead to under-fitting. It will over generalize the model.

3. Explain why the squared loss is not suitable for binary classification problems.
   Ans: Squared loss is generally not used for classification. It is used for Regression.
   Squared loss is dominated by outliers. It penalizes the outliers very extensively regardless of the sign of y_n and $\hat{y}_n$ which reduces accuracy.

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.
   Ans: We can modify squared loss function by adding a regularization term.
   Regularization shrinks weight coefficients towards zero thus alleviating the impact of outliers.

5. Perceptron algorithm:

   (a) Implement the perceptron algorithm for binary classification. 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. 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.
   Refer Figure 1

   (b) 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.
   Refer Figure 2 and 3

   (c) For each class, visualize the 20 best scoring and 20 worst scoring images that are classified as that class. The worst ones are close to the boundary and may be wrong.
   Refer figure 4,5,6,7

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

(e) 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? Explain why.
Ans: We see that with sorted data, model learns slowly. Accuracy of the model for lower iterations is lower around 50%. With increase in number of iterations, accuracy starts increasing. Refer figure 8

(f) Repeat (a) with 10 training examples per class and compare with the result of (a).
In (a), accuracy is around 99% for higher iterations with 1000 data points. Here, with only 20 data points (10 data points for each class), overall accuracy reduces. Acuuracy is 94% for higher iterations here compared to 99% in (a). Refer figure 9

6. SVM algorithm:

(a) Implement the gradient descent algorithm for binary SVM. Train and test it for classifying digits "1" and "6" in MNIST dataset. Plot the accuracy on the test set w.r.t. 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.
Refer figure 10

(b) 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. Choosing the learning rate may be a little tricky. One popular strategy is to reduce it proportional to $\frac{1}{t}$ where t is the iteration number.
Refer figure 11

(c) Implement a function for 1-vs-all multi-class SVM that calls the binary SVM function.
Refer figure 12

(d) 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.
Average Precision is: 0.96091

(e) Look at top mistakes and show images along with ground truth label and the predicted label to see if they make sense.
Refer figure 13

- Accuracy vs No of terations.png

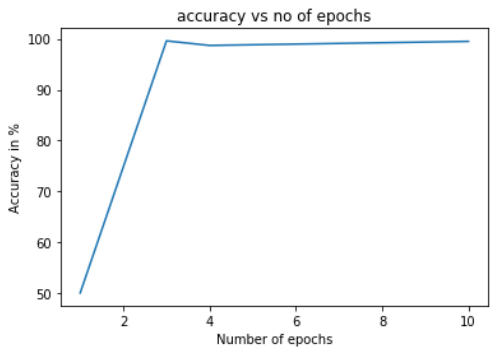

Figure 1:

- Weight Vector image for digit 1.png

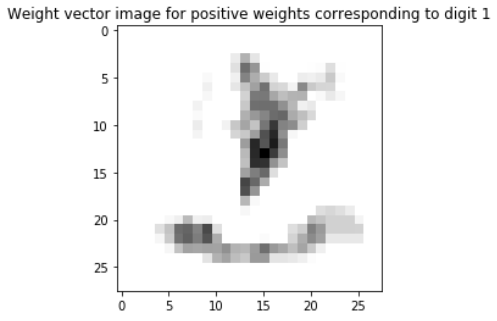

Figure 2:

- Weight Vector image for digit 6.png

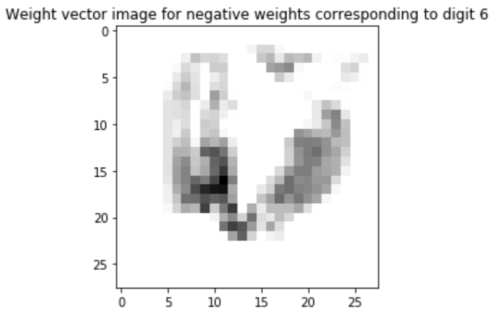

Figure 3:

- Positive best 20.png

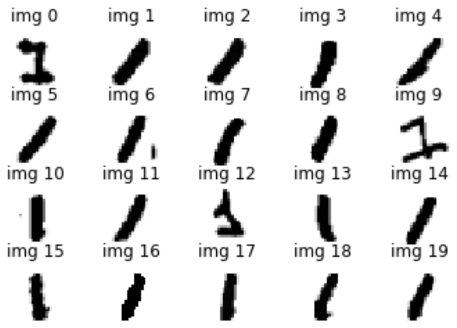

Figure 4:

- Positive worst 20.png

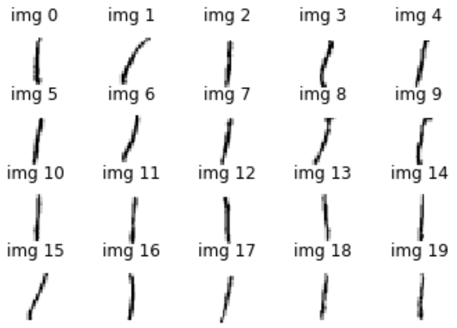

Figure 5:

- Negative best 20.png

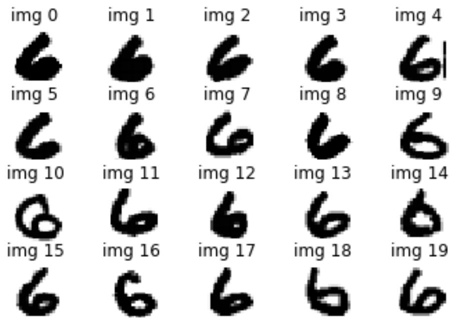

Figure 6:

- Negative worst 20.png

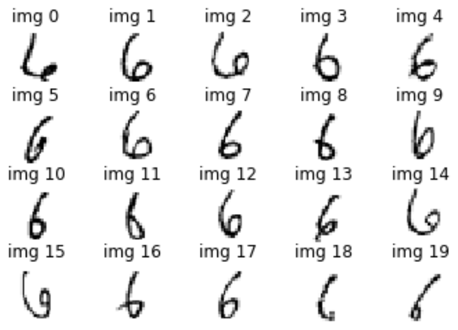

Figure 7:

- Accuracy vs No of terations for sorted data.png

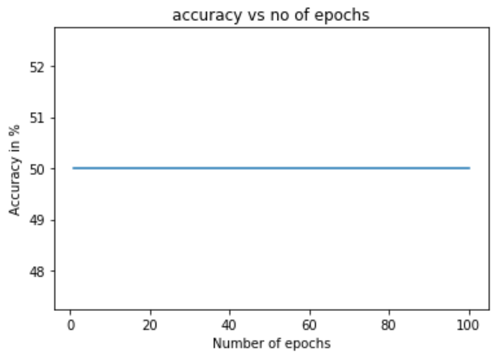

Figure 8:

- Accuracy vs No of terations for 20 data points.png

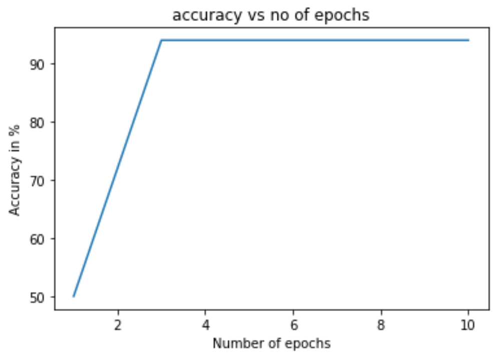

Figure 9:

- Accuracy vs No of terations.png

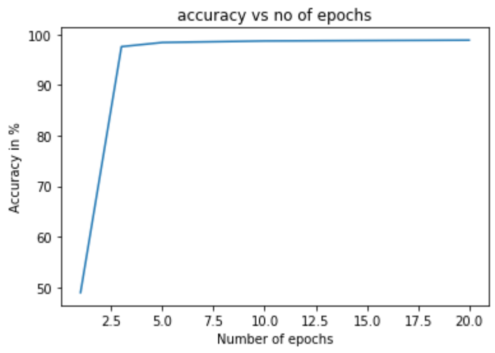

Figure 10:

- Accuracy vs Lambda.png

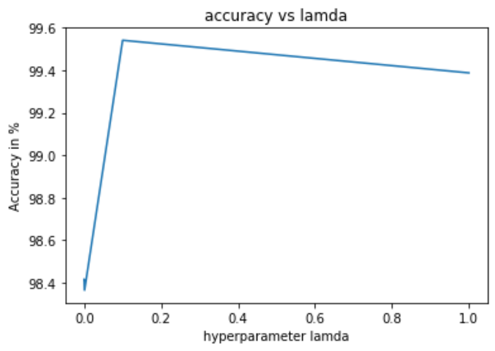

Figure 11:

- Accuracy vs No of terations for multiclass classification.png

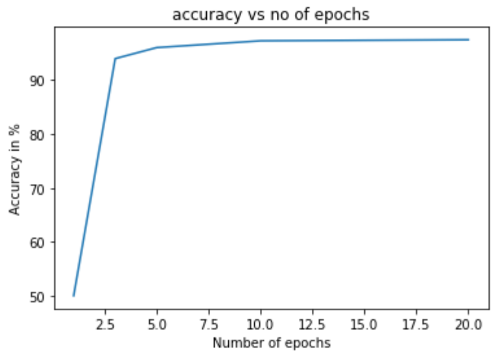

Figure 12:

- Top Mistakes.png

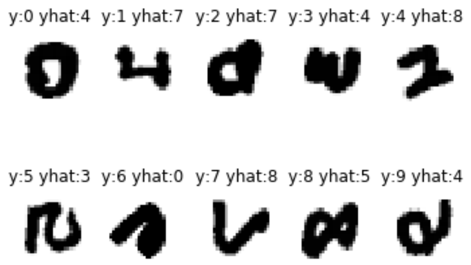

Figure 13: