E0 270 Machine Learning
Assignment - 2

Sruthi Gorantla
M. Tech. CSA
SR No. 15190

March 3, 2018

1. **(Support Vector Machines)**

   (a) (5 points) **Generating Synthetic Data**

   Solution:

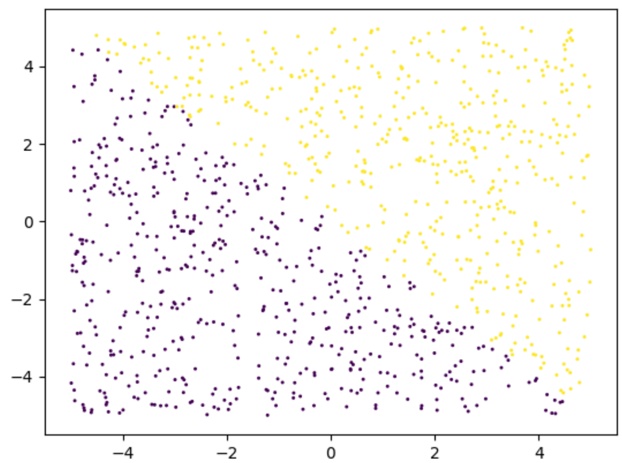


Figure 1: Learning Curve

The data is linearly separable as we can see a clear boundary between the two classes, which is the line $y = -x + 5$

   (b) (10 points) **Implementing Hard SVM**

   Solution:

   Accuracy of training data: 0.9987080103359173
   Accuracy of testing data: 0.995575221238938

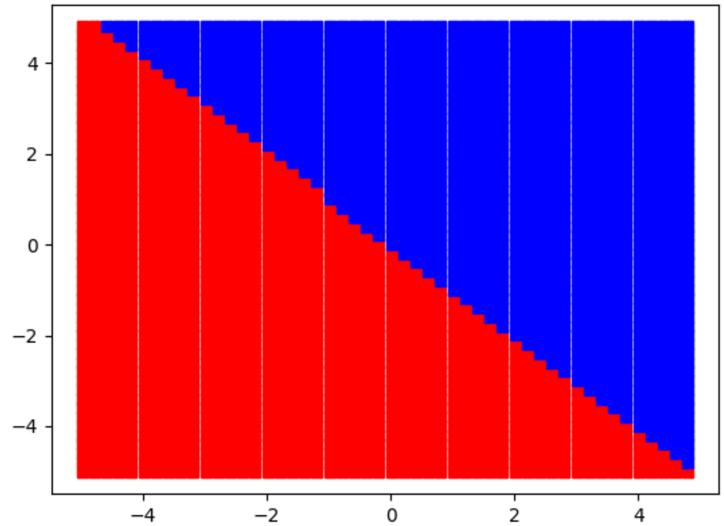


Figure 2: Decision Boundary

(c) (5 points) **Implementing Soft SVM**

Solution:

In the table below are the train and test accuracies of the linearly separable data generated in part (a)

	Train			Test		
C	0.1	1	10	0.1	1	10
p = 0	0.99	0.99	0.99	0.98	1	1
p = 0.1	0.97	0.98	0.96	0.97	0.97	0.96
p = 0.3	0.95	0.92	0.98	0.95	0.89	0.98

(d) (5 points) **Generating More Complex Synthetic Data**

Solution:

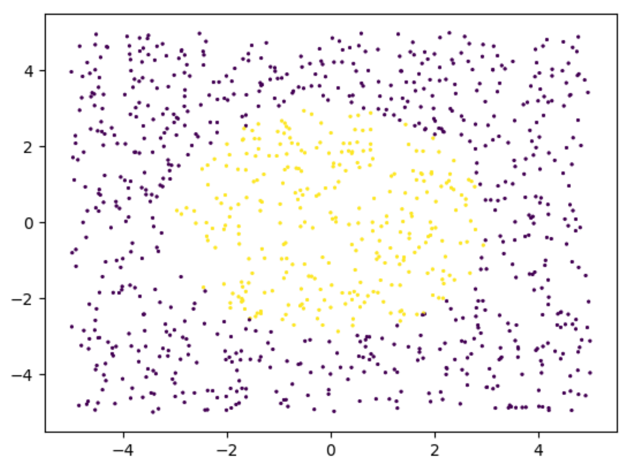


Figure 3: Non linearly separable data

Clearly the data is not linearly separable as we can't draw a line separating both the classes.

(e) (5 points) **Using Kernels**

Solution:

Accuracy of training data: 0.9655870445344129
Accuracy of testing data: 0.9545454545454546

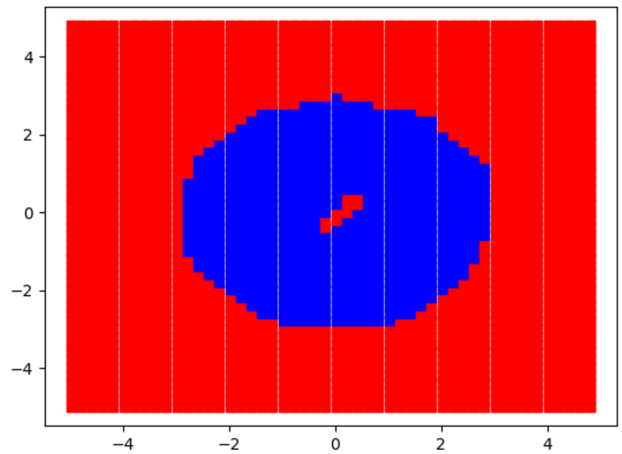


Figure 4: Decision Boundary

(f) (10 points) **Dealing with Real Data**

Solution:

C	Train	Test
0.01	0.89	0.80
0.1	0.89	0.80
1	0.89	0.81
10	0.97	0.78
100	0.99	0.79

This table shows the average train and test accuracy accross 5 folds of the data given.
Best C = 1
Accuracy of training data: 0.884
Accuracy of testing data: 0.8483107331647897

2. **(Neural Networks)**

(a) (0 points) **Forward Pass**

Solution:

Implemented.

(b) (0 points) **Backward Pass**

Solution:

Implemented.

(c) (20 points) **Putting it Together**

Solution:

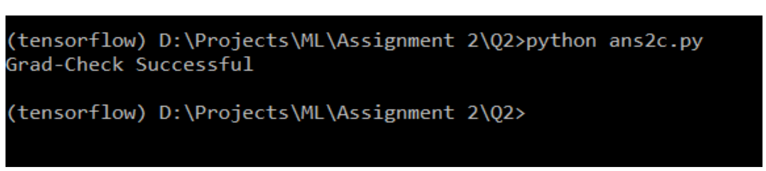


Figure 5: Screen shot of gradient check

In the __main__ function, at each layer, to compute the cost_pos, a small amount ϵ is added to all the weights/biases in that layer. Then a forward pass is done and the gradients are calculated in the backward pass. Similarly a small amount ϵ is subtracted from all the weights/biases in that layer and cost_neg is computed. Then the slope of (cost_pos,ϵ) and (cost_neg,$-\epsilon$) is calculated. This gives the approximation of the actual gradient at a the actual weights as shown below:

$$\frac{df(x)}{dx} \approx \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

Considering f is the cost function J and x representing the weights/biases, we change the weights by $\epsilon = h$ and then find the slope which is approximately equal to the slope we want to find at W/b. If slope calculated by this formula is not equal to the slope calculated by backpropogation by a small threshold, then we can say that the implementation of the backpropogation is wrong.

(d) (10 points) **Training on Real Data**

Solution:

Train accuracy: 0.976
Test accuracy: 0.89335784877

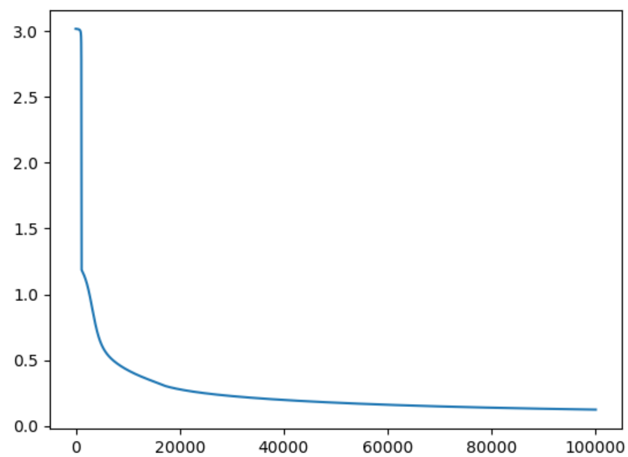


Figure 6: Cost incurred over 100,000 iterations

After around 75,000 iterations, the train loss kept decreasing whereas the test loss started increasing. This implies that the model has overfit to the train data. Hence we stop the training and compute the accuracy on the test data at around 73,000 iterations. This is called *early stopping*. This is performed in order to get better generalization.

(e) (10 points) **Using PyTorch**

Solution:

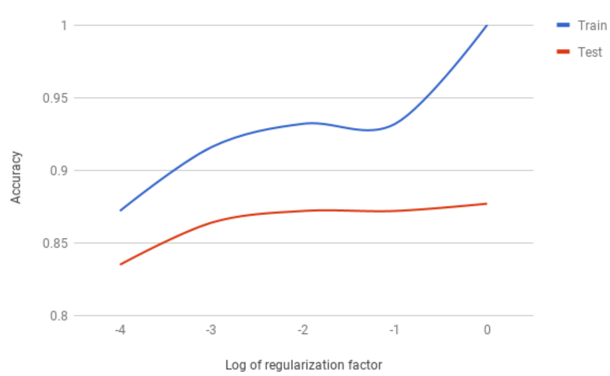


Figure 7: Accuracy plotted against log of regularization factor λ

NOTE: These results are highly sensitive to the initialization of the weight matrices.

3. (10 points) **(Cost Sensitive Binary Classification)**

Solution:

Given
D: Joint probability distribution on $\mathbf{X} \times \{+1, -1\}$
μ: marginal distribution on $\mathbf{X}$
$\eta(\mathbf{x})$: $p(y = 1 \mid x)$
We know that expectation of loss is Risk. Hence, we have to find h^* that gives the infimum of the loss calculated with all the available h's using Risk formulation done in class (NOTE: Not all such h mappings are practically available to us. Hence, we use infimum rather than minimum). As formulated in class, for Naive Bayes' Classifier, the decision boundary is calculated as follows:
Choose $y = 1$ if:

$$\frac{P(x \mid y = 1)}{P(x \mid y = -1)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{22}} \frac{P(y = -1)}{P(y = 1)}$$

where λ_{ij} is the cost of misclassifying class j as class i. Given $\lambda_{11} = 0; \lambda_{12} = 1 - c; \lambda_{21} = c; \lambda_{22} = 0$. Using Bayes' rule we get,

$$\frac{P(y = 1 \mid x)}{P(y = -1 \mid x)} > \frac{1 - c}{c}$$

Given $\eta(x) = P(y = 1 \mid x) \implies 1 - \eta(x) = P(y = -1 \mid x)$. Hence,

$$\frac{\eta(x)}{1 - \eta(x)} > \frac{1 - c}{c}$$

Therefore, the Bayes classifier says:
Predict $y = 1$ if $\eta(x) > 1 - c$

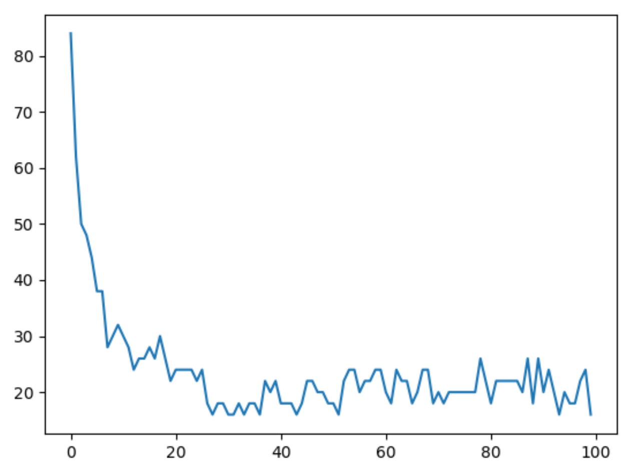
In the second setting $\lambda_{12} = b$ and $\lambda_{21} = a$, the rest remaining the same. Hence, the classifier will be:
Predict $y = 1$ if:

$$\frac{\eta(x)}{1 - \eta(x)} > \frac{b}{a}$$

Therefore, the Bayes classifier says:
Predict $y = 1$ if $\eta(x) > \frac{b}{a+b}$

4. (10 points) **(Perceptron)**

Solution:



(a)

Figure 8: Number of misclassifications plotted against number of iterations for shuffled data

(b) Training Accuracy: 0.952
Testing Accuracy: 0.8848540565387267

(c)

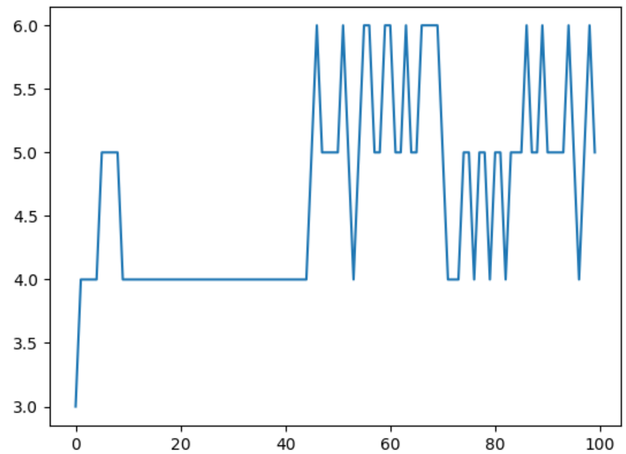


Figure 9: Number of misclassifications plotted against number of iterations for un-shuffled data

The accuracies shown in part (b) are performed on the shuffled data. When the data is not shuffled, the perceptron tries to update weights by a large factor to cope with the change in label seen after succeeding in predicting the same label correctly continuously over a large set of examples. Another observation is, the number of miscalssifications in the first iteration are only 3. Because the first half of the data belongs to one class and the next class belongs to another class. The perceptron only makes mistake during the first the middle and one more after first or middle example. In the second iterations, the perceptron again makes mistake because it tends to remember most the last few updates as they are from the same class and forgets the previous information. Hence, there is an increase in the number of misclassifications after particular iterations and the number of misclassifications increase after few iterations unlike shuffled data. The graph of number of misclassifications is as shown above.

Shuffling the features in a fixed order over all the examples wouldn't hurt the classification becuase each feature represents a dimension in space and these dimensions are independent of each other. Hence, even after reordering the features, all the information remains same.

(d) Not all the weights are comparable. The top 5 and bottom 5 features ordered according to the weights assigned to them by the perceptron are:
Top 5: ['char_freq_!', 'word_freq_remove', 'char_freq_$', 'word_freq_over', 'word_freq_money']
Bottom 5: ['word_freq_address', 'word_freq_edu', 'word_freq_direct', 'word_freq_technology', 'word_freq_project']