Machine Learning

Homework - 3

Author: Ankur Sharma (2016225)

Q1.

a) *<also includes answer for c>*

NN with 1 hidden layer-

Sigmoid: lr = 0.9
Training Accuracy 99.09%
Test Accuracy 97.39%

Relu: lr = 0.1
Training Accuracy 97.89%
Test Accuracy 96.48%

NN with 3 hidden layers-

Sigmoid: lr = 0.9
Training Accuracy 99.78%
Test Accuracy 97.72%

Relu: lr = 0.001
Training Accuracy 92.44%
Test Accuracy 90.8%

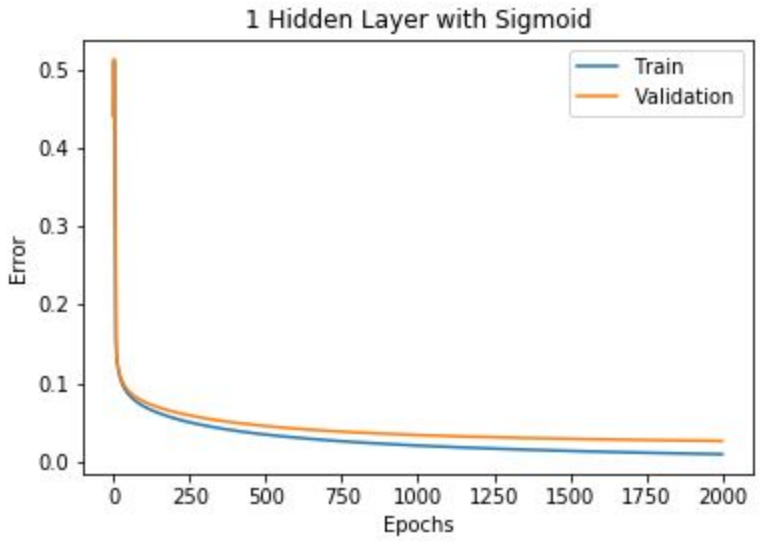
lr: *learning rate*

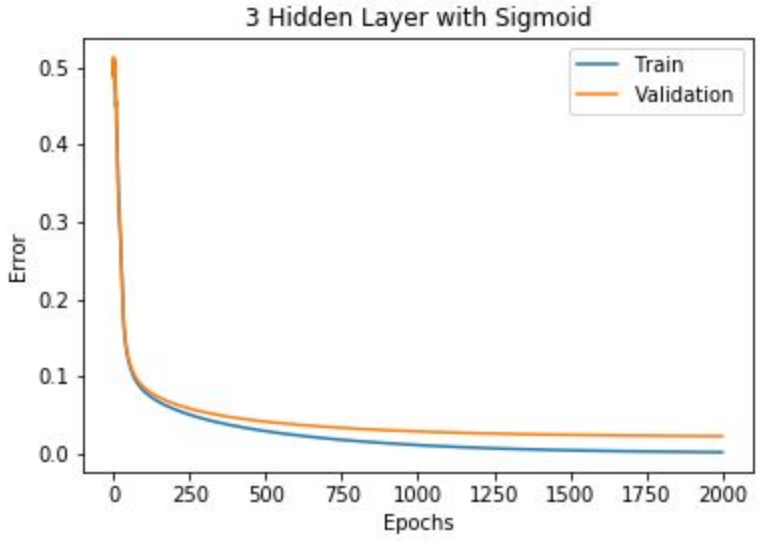
b) *<also includes answer for c>*

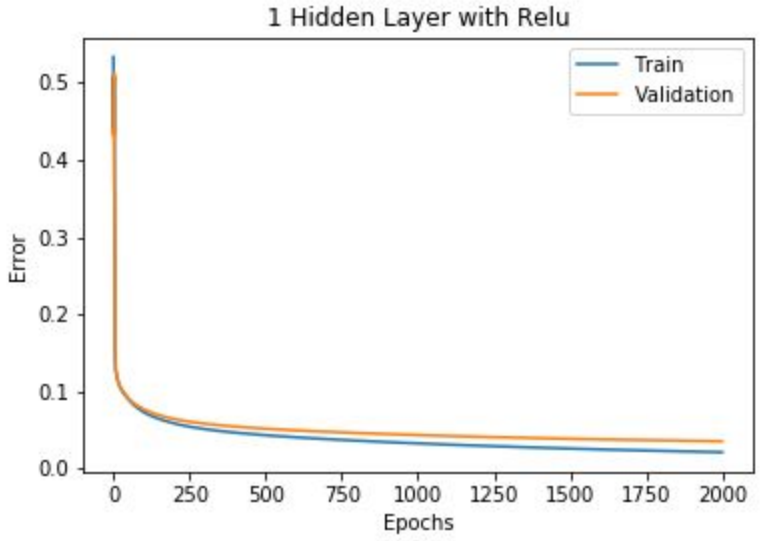
Following graphs show that while training validation and training error both decreases. There is not much difference in train and validation error which implies that the NN generalizes the data very well. If my model overfit the data then there

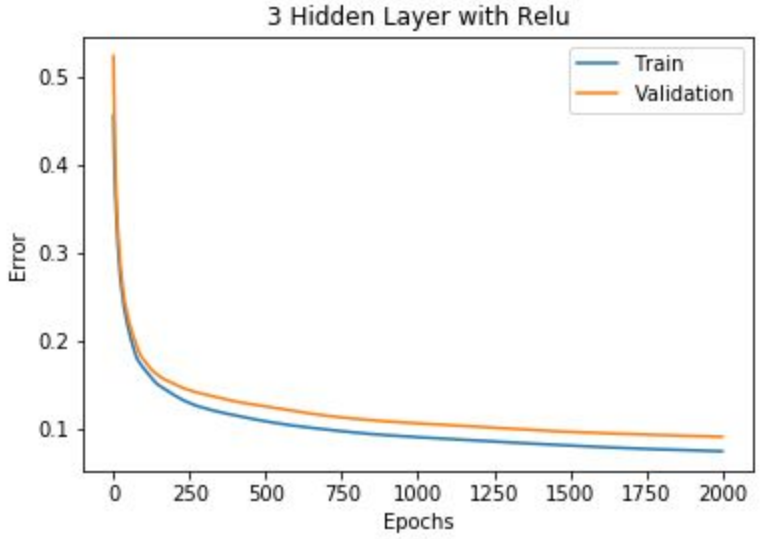
would be high difference in validation and train error which is clearly not the case here. In case of underfit, there would be high error in training data as well.

I didn't face any such issues while implementing the NN because I've done it before and I am familiar with its implementation. In case of relu, I faced 'divide by zero' exception because I was using very high learning weight.









d) **SVM**

Training Accuracy 96.28%

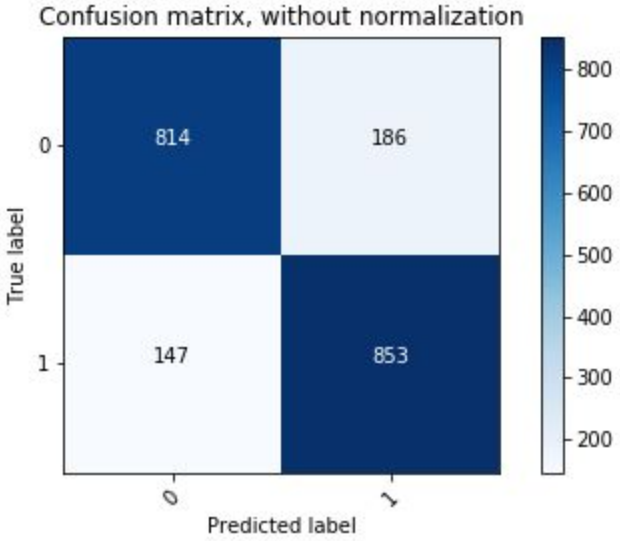
Test Accuracy 95.42%

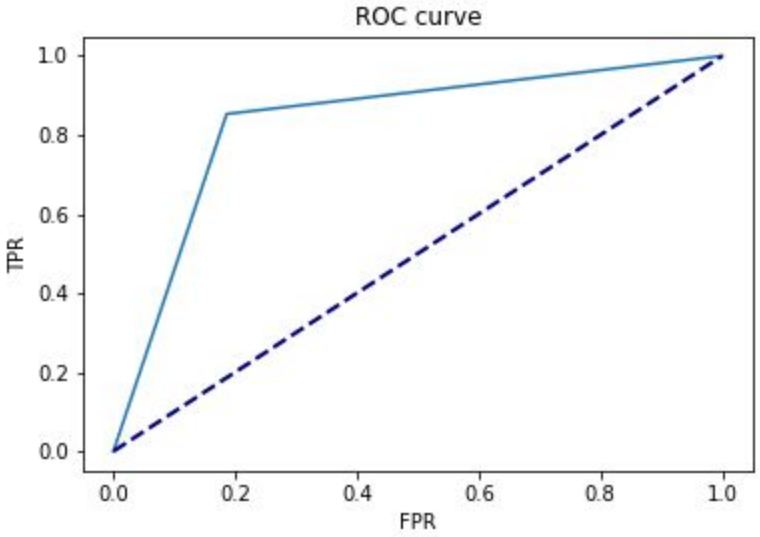
I don't notice very high difference in the performance of any NN and SVM.

Q2.

Training Accuracy: 89.52%

Test Accuracy: 83.35%


Confusion matrix, without normalization
814
186
147
853
0
1
True label
0
1
Predicted label
800
700
600
500
400
300
200


ROC curve
TPR
FPR
0.0
0.2
0.4
0.6
0.8
1.0

Q4 C.E. cross entropy : $-(y \log(p) + (1-y)\log(1-p))$
(for binary)

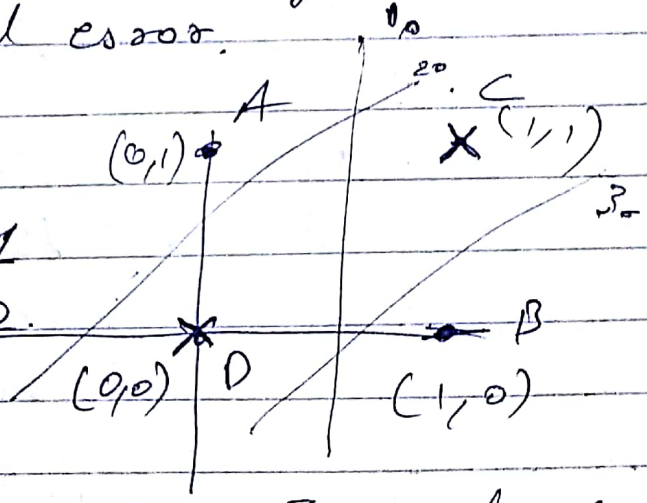
$y \in \{0, 1\}$

Case I: $y = 1$, p is near 1
loss will be very low.

Case II: $y = 1$, p is near 0
loss will be very high

Similar result can be seen when $y = 0$ but in the opposite direction. Hence, taking sign of loss in account which results in Jeffrey convergence as compared to squared error.

Q5 No.
A, B belongs to class 1
while C, D " " class 2.



You can see none of the line (1, 2, 3) can separate the data. It is clear that the data is not linearly separable and hence, one can't classify XOR function using linear activation functions in the NN.

3- As the input ranges from 500 to 1000, at such big values of input, derivative of sigmoid function drop has a very small norm which would result in slow learning. Moreover, derivative of sigmoid is at max 0.25 which leads to very slow convergence when compared to relu which has a constant big value of gradient norm. Hence, Relu is better than sigmoid. Min-max scaling or normalization can be done to remedy this problem.

Yes one should initialize weights randomly. The best practice is to initialize them b/w -1 and +1 randomly. Initializing weights with 0 will be a problem and all the neurons in the layer will give same output and would not contribute.

Cross entropy is used in case of classification as MSE is very sensitive to outliers and leads to slow learning. while cross entropy converges faster.