ML HW3

Abhimanshu Mishra — amishr11@binghamton.edu

April 15, 2020

1 Models

Model	Modification	Test Set Accuracy
Perceptron	-	93.515%
Perceptron	No Stopwords	92.887%
Naive Bayes	-	94.561%
Naive Bayes	Stemming	93.933%
Naive Bayes	No Stopwords	94.561%
Naive Bayes	No Stopwords and stemming	93.933%

Table 1: Accuracy of different models on test set

The first perceptron model reported in Table 1 was trained with learning rate 0.8 for 100 epochs. The second perceptron model reported in Table 1 was trained with learning rate 1 for 100 epochs.

It looks like naive bayes performs comparably to perceptron for this task and dataset.

I swept learning rates values from 0.01 to 1. I swept epochs from 100 to 200.

By looking at Table 2 and Table 3 together, we see that stopword removal during training does not have that big an effect on accuracy on a test set. As learning rate increases, so does performance. An increase in number of epochs does not bring about a significant change since the model converges pretty early. Additionally, from Table 2 and Table 3 individually, it is clear that a learning rate of 0.8 is the best and the model reaches convergence at 100 epochs. However, any learning rate between 0.8 and 1 should be optimal as seen by relatively similar performance is achieved in this range.

NOTE: During training time, I shuffle the training set before training. This results in a slight variance every time. So, the numbers are not exactly reproducible.

Learning Rate	Epoch	Test Set Accuracy
0.01	100	85.565%
0.02	100	85.565%
0.03	100	88.075%
0.04	100	89.121%
0.05	100	88.703
0.1	100	90.377%
0.5	100	91.004%
0.8	100	**93.515%**
1	100	92.05%
00.01	150	86.611%
0.02	150	87.238%
0.03	150	87.238%
0.04	150	88.285%
0.05	150	87.866%
0.1	150	92.259%
0.5	150	91.632%
0.8	150	92.05%
1	150	93.096%
0.01	200	85.774%
0.02	200	87.448%
0.03	200	87.657%
0.04	200	89.331%
0.05	200	88.703%
0.1	200	90.167%
0.5	200	93.096%
0.8	200	91.632%
1	200	93.096%

Table 2: Test set accuracy correlated with learning rate and epochs, without stopword removal

2 XOR

The given problem is not linearly separable. So, there is a need for non-linearity to solve it. This is provided by neural networks.

Table 4 shows the relation between the inputs and projected output of the desired neural network. The required neural network for this truth table (Table 4) is drawn in Figure 1. This network has two hidden units in one hidden layer. There is only one node in the output layer since there are only two classes possible - $\times$ and $\diamond$.

If these 4 data points are given, the neural network will learn just this pattern and will be able to correctly classify these examples after training i.e., zero training error.

Learning Rate	Epoch	Test Set Accuracy
0.01	100	85.983%
0.02	100	85.356%
0.03	100	87.029%
0.04	100	89.331%
0.05	100	90.586%
0.1	100	88.912%
0.5	100	91.841%
0.8	100	92.259%
1	100	**92.887%**
0.01	150	85.356%
0.02	150	87.448%
0.03	150	88.494%
0.04	150	89.121%
0.05	150	88.075
0.1	150	92.259%
0.5	150	92.259%
0.8	150	91.423%
1	150	91.841%
0.01	200	85.774%
0.02	200	86.192%
0.03	200	89.54%
0.04	200	89.121%
0.05	200	89.749
0.1	200	91.423%
0.5	200	92.259%
0.8	200	91.213%
1	200	91.841%

Table 3: Test set accuracy correlated with learning rate and epochs when stopwords have been removed in training

X	Y	Shape
-1	-1	$\times$
-1	1	$\diamond$
1	1	$\times$
1	-1	$\diamond$

Table 4: Truth Table for required network

A hidden unit is activated only is the sum of product of weights and inputs

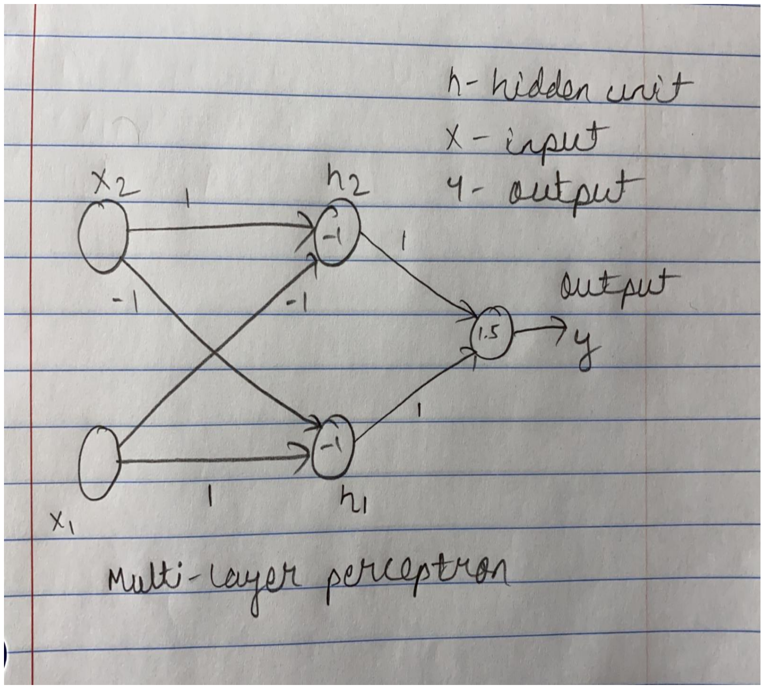

Figure 1: Required Neural Network

incoming into it is greater than a threshold.

$$y = \begin{cases} 1, & \text{for } \phi(\sum_i w_i x_i) \geq \theta \\ 0, & \text{for } \phi(\sum_i w_i x_i) < \theta \end{cases} \tag{1}$$

This is an example of a step function acting as activation function, where ϕ is the identity function in this case.

In the figure, the value inside the nodes is the threshold for that unit while the values over the edges are the weights. When the input to the network is 1 and 1, the output value will be 0 i.e., $\times$. When the input to the network is -1 and -1, the output value will be 0 i.e., be $\times$. When the input to the network is -1 and 1, the output value will be 1 i.e., be $\diamond$. When the input to the network is 1 and -1, the output value will be 1 i.e., be $\diamond$.