Assignment 3

Dishank Jain
AI20BTECH11011

1 Neural Networks (2+2 marks)

(a) The XOR function returns true only when one of the arguments is true and another is false. Otherwise it returns false. Show that a two-layer perceptron can solve the XOR problem.

(b) x, y and z are inputs with values -2, 5 and -4 respectively. You have a neuron q and neuron f with functions:

$$q = x - y$$
$$f = q * z$$

Show the graphical representation and compute the gradient of f with respect to x, y and z.

1.1 Solution

1.1.1 (a)

Figure 1: Figure

If the two attributes are x and y, then in the proposed neural net, they give output 1 when they lie in the overlap of the blue and the green regions. Thus the XOR problem is solved.

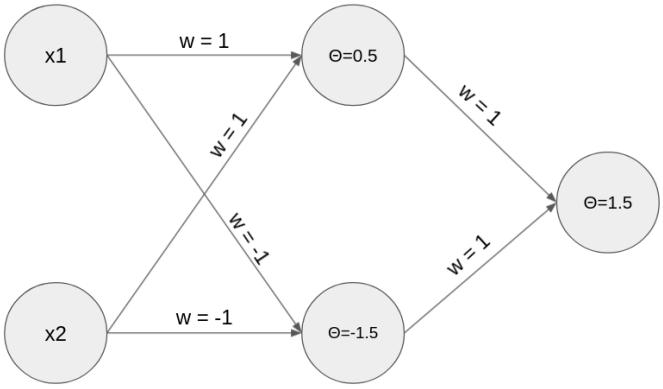

Figure 2: Neural Net diagram

1.1.2 (b)

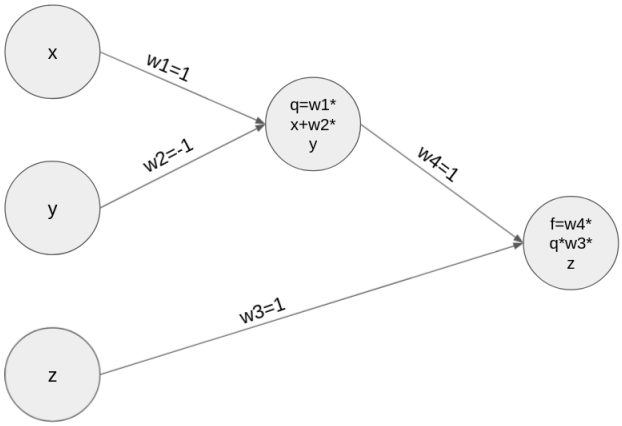

Figure 3: Neural Net diagram

$$\frac{\partial f}{\partial z} = z\frac{\partial q}{\partial z} + q\frac{\partial z}{\partial z} = q = x - y$$
$$\frac{\partial f}{\partial x} = z\frac{\partial q}{\partial x} + q\frac{\partial z}{\partial x} = z\frac{\partial (x-y)}{\partial x} = z$$

$$\frac{\partial f}{\partial y} = z\frac{\partial q}{\partial y} + q\frac{\partial z}{\partial y} = z\frac{\partial (x-y)}{\partial y} = -z$$

2 Neural Networks (4 marks)

The extension of the cross-entropy error function for a multi-class classification function is given by

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} ln y_k(\mathbf{x}_n, \mathbf{w})$$

where **K** is the number of classes, N is the number of data samples, and $\mathbf{t}_n$ is a one hot vector which designates the expected output for a data sample $\mathbf{x}_n$. The network outputs $y_k(\mathbf{x}_n, \mathbf{w}) = p(t_k = 1|\mathbf{x})$ are given by the softmax activation function:

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{exp(a_k(\mathbf{x}, \mathbf{w}))}{\sum_k exp(a_k(\mathbf{x}, \mathbf{w}))}$$

which satisfies $0 \leq y_k \leq 1$ and $\sum_k y_k = 1$, where a_k's are the softmax activations of the output layer neurons. Show that the derivative of the above error function with respect to the activation a_k for an output unit having a logistic sigmoid activation is given by:

$$\frac{\partial E}{\partial a_k} = y_k - t_k$$

2.1 Solution

$$\begin{aligned}
\frac{\partial E_n}{\partial a_i} &= -\sum_{k=1}^{K} \frac{\partial (t_{kn} ln y_k(\mathbf{x}_n, \mathbf{w}))}{\partial a_i} \\
&= -\sum_{k=1}^{K} \left(ln y_{kn} \frac{\partial t_{kn}}{\partial a_i} + t_{kn} \times \frac{1}{y_{kn}} \times \frac{\partial y_{kn}}{\partial a_i} \right) \\
&= -\sum_{k=1}^{K} t_{kn} \times \frac{1}{y_{kn}} \times \frac{\partial y_{kn}}{\partial a_i}
\end{aligned}$$

If $k = i$,

$$\frac{\partial y_{kn}}{\partial a_i} = y_{in} - y_{in}^2$$

else if $k \neq j$,

$$\frac{\partial y_{kn}}{\partial a_i} = -y_{kn} y_{in}$$

$$
\begin{aligned}
\implies \frac{\partial E_n}{\partial a_i} &= -\left(\sum_{k\neq i} t_{kn} \times \frac{1}{y_{kn}} \times (-y_{kn}y_{in}) + t_{in} \times \frac{1}{y_{in}} \times (y_{in} - y_{in}^2)\right) \\
&= -\left(y_{in}\left(-\sum_{k=1}^{K} t_{kn} + t_{in}\right) + t_{in}(1 - y_{in})\right) \\
&= -(y_{in}(t_{in} - 1) + t_{in}(1 - y_{in})) \\
&= y_{in} - t_{in}
\end{aligned}
$$

3 Ensemble methods (2+2 marks)

Consider a convex function $f(x) = x^2$. Show that the average expected sum-of-squares error $E_{AV} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_x[(y_m(x) - f(x))^2]$ of the members of an ensemble model and the expected error $E_{ENS} = \mathbb{E}_x[(\frac{1}{M}\sum_{m=1}^{M} y_m(x) - f(x))^2]$ of the ensemble satisfy:

$$E_{ENS} \leq E_{AV}$$

Show further that the above result holds for any error function $E(y)$, not just sum-of-squares, as long as it is convex in y.

3.1 Solution

$$
\begin{aligned}
E_{ENS} &= \mathbb{E}_x\left[(\frac{1}{M}\sum_{m=1}^{M} y_m(x) - f(x))^2\right] \\
&= \mathbb{E}_x\left[\left(\frac{1}{M}\sum_{m=1}^{M} y_m(x)\right)^2 + f^2(x) - \frac{2f(x)}{M}\sum_{m=1}^{M} y_m(x)\right] \\
&= \mathbb{E}_x\left[\left(\frac{1}{M}\sum_{m=1}^{M} y_m(x)\right)^2\right] + \mathbb{E}_x\left[f^2(x)\right] - \frac{2}{M}\sum_{m=1}^{M}\mathbb{E}_x[y_m(x)f(x)] \\
E_{AV} &= \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x[y_m^2(x) + f^2(x) - 2y_m f(x)] \\
&= \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x[y_m^2(x)] + \mathbb{E}_x[f^2(x)] - \frac{2}{M}\sum_{m=1}^{M}\mathbb{E}_x[y_m(x)f(x)]
\end{aligned}
$$

We can say that

$$\sum_{m=1}^{M}\left(y_m(x)-\frac{1}{M}\sum_{m=1}^{M}y_m(x)\right)^2\geq 0$$
$$\implies \sum_{m=1}^{M}y_m^2(x)+\sum_{m=1}^{M}\left(\sum_{m=1}^{M}\frac{1}{M}y_m(x)\right)^2-\sum_{m=1}^{M}y_m(x)\left(\sum_{m=1}^{M}\frac{1}{M}y_m(x)\right)\geq 0$$
$$\implies \frac{1}{M}\sum_{m=1}^{M}y_m^2(x)-\left(\frac{1}{M}\sum_{m=1}^{M}y_m(x)\right)^2\geq 0$$
$$\implies \frac{1}{M}\sum_{m=1}^{M}y_m^2(x)\geq\left(\frac{1}{M}\sum_{m=1}^{M}y_m(x)\right)^2$$
$$\implies \mathbb{E}_x\left[\frac{1}{M}\sum_{m=1}^{M}y_m^2(x)\right]\geq\mathbb{E}_x\left[\left(\frac{1}{M}\sum_{m=1}^{M}y_m(x)\right)^2\right]$$
$$\implies E_{ENS}\leq E_{AV}$$

We can further extend this result using Jensen's inequality. Jensen's inequality states that for any convex function $\psi(X)$, where X is a random variable, $\psi\left(\frac{\sum x_i}{n}\right)\leq\frac{\sum\psi(x_i)}{n}$. If we take the error function as $E(y)$, which is convex, then we have

$$E_{AV}=\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x[E(y_m(x))]$$
$$E_{ENS}=\mathbb{E}_x\left[E\left(\frac{1}{M}\sum_{m=1}^{M}y_m(x)\right)\right]$$

Using Jensen's inequality,

$$E\left(\frac{1}{M}\sum_{m=1}^{M}y_m(x)\right)\leq\frac{\sum_{m=1}^{M}E(y_m(x))}{M}$$
$$\implies \mathbb{E}_x\left[E\left(\frac{1}{M}\sum_{m=1}^{M}y_m(x)\right)\right]\leq\mathbb{E}_x\left[\frac{\sum_{m=1}^{M}E(y_m(x))}{M}\right]$$
$$\implies \mathbb{E}_x\left[E\left(\frac{1}{M}\sum_{m=1}^{M}y_m(x)\right)\right]\leq\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x\left[E(y_m(x))\right]$$
$$\implies E_{ENS}\leq E_{AV}$$

4 Random Forests (10 marks)

4.1 Solution

4.1.1 (5 marks)

The accuracy of my Random forest for one randomly chosen run is 0.9355. It's run time is 28.7 seconds. The accuracy of the inbuilt Random forest was 0.9500. It's run time was 1.65 seconds.

4.1.2 (2.5 marks)

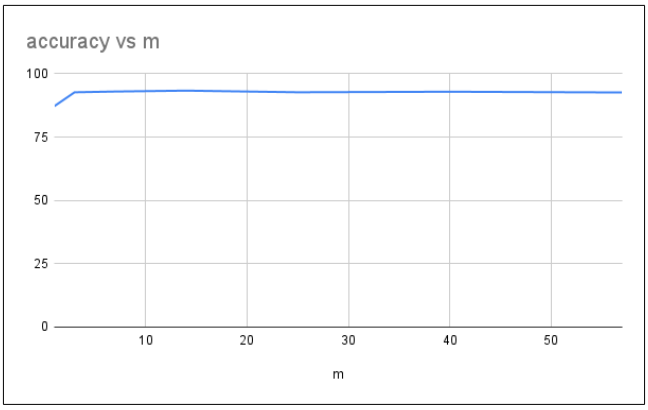

Figure 4: Caption

4.1.3 (2.5 marks)

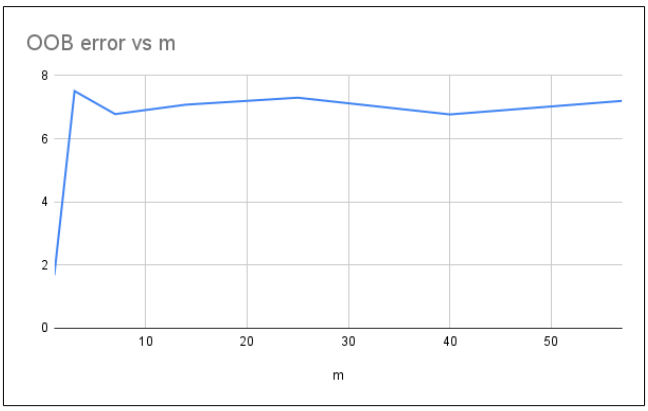

Figure 5: Caption

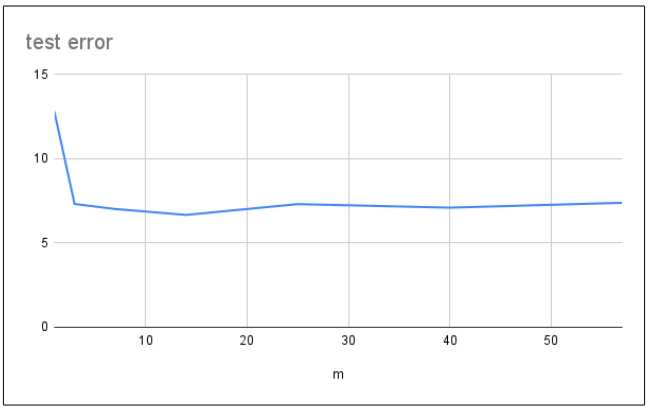

Figure 6: Caption

5 Gradient Boosting (8 marks)

5.1 Solution

5.1.1 (a)

The data had to undergo quite extensive pre-processing. First, the columns with all NaN values were removed. Then all records with loan status as 'Current' were removed. This left us with 57 attributes. Each attribute was individually dealt with. A lot of these attributes were dropped because of various reasons. Like some of them had a single value for the attribute. 2 of the attributes were more than 50% NaNs, so they had to be dropped too. All the dates were dropped because I felt they were irrelevant to the task at hand. Quite a few irrelevant data attributes were dropped as well, like id, member id, url, etc. Unfortunately, I also had to drop the description column since the description was in natural language.

Next, we had to transform categorical data into numerical data. Home ownership, term and verification status were converted to binary by clubbing categories in two when needed. Purpose was converted to ordinal data. Note that the interest was also dropped because interest was amply conveyed by grade as LC decides interest directly based on the grade. Grade was converted to ordinal data and sub-grade was scaled and added to it in a way that harmony of the data was maintained. State was also converted to ordinal data. It was thought that state would be relevant because in US, the laws and incomes can vary significantly from state to state. A few state values were tweaked to maintain consistency in mapping. The records for which public record of bankruptcies was not available were dropped.

5.1.2 (b)

- 1. Model 1: All default values. Precision = 0.9940, Recall = 0.9999, Accuracy = 0.9948
  2. Model 2: Exponential loss function. Precision = 0.9922, Recall = 0.9998, Accuracy = 0.9932
  3. Model 3: Learning rate = 1. Precision = 0.9957, Recall = 0.9529, Accuracy = 0.9565
  4. Model 4: Learning rate = 0.05. Precision = 0.9902, Recall = 1.0, Accuracy = 0.9916
  5. Model 5: Number of trees = 50. Precision = 0.9911, Recall = 0.9999, Accuracy = 0.9923
  6. Model 6: Number of trees = 150. Precision = 0.9951, Recall = 0.9997, Accuracy = 0.9956
  7. Model 7: Number of trees = 200. Precision = 0.9958, Recall = 0.9997, Accuracy = 0.9962
  8. Model 8: Number of trees = 250. Precision = 0.9964, Recall = 0.9998, Accuracy = 0.9968
  9. Model 9: Number of trees = 300. Precision = 0.9966, Recall = 0.9998, Accuracy = 0.9969
  10. Model 10: Number of trees = 500. Precision = 0.9969, Recall = 0.9997, Accuracy = 0.9971
  11. Model 11: Number of trees = 1000. Precision = 0.9975, Recall = 0.9999, Accuracy = 0.9979
  12. Model 12: Number of trees = 5000. Precision = 0.9989, Recall = 0.9995, Accuracy = 0.9986
  13. Model 13: Sub sample = 0.8. Precision = 0.9943, Recall = 0.9998, Accuracy = 0.9950
  14. Model 14: Sub sample = 0.7. Precision = 0.9944, Recall = 0.9998, Accuracy = 0.9950
  15. Model 15: Sub sample = 0.6. Precision = 0.9936, Recall = 0.9993, Accuracy = 0.9940
  16. Model 16: Sub sample = 0.6, Number of trees = 200. Precision = 0.9950, Recall = 0.9997, Accuracy = 0.9954
  17. Model 17: Sub sample = 0.7, Number of trees = 500. Precision = 0.9962, Recall = 0.9997, Accuracy = 0.9965
  18. Model 18: Sub sample = 0.8, Number of trees = 500. Precision = 0.9968, Recall = 0.9996, Accuracy = 0.9969
  19. Model 19: max depth of tree = 5. Precision = 0.9968, Recall = 0.9998, Accuracy = 0.9971

20. Model 20: max depth of tree = 7. Precision = 0.9974, Recall = 1.0, Accuracy = 0.9978
21. Model 21: max depth of tree = 10. Precision = 0.9979, Recall = 0.9997, Accuracy = 0.998
22. Model 22: max depth of tree = 10, number of trees = 200. Precision = 0.998, Recall = 0.9999, Accuracy = 0.9983
23. Model 23: max depth of tree = 10, number of trees = 500. Precision = 0.9982, Recall = 0.9999, Accuracy = 0.9984
24. Model 24: max depth of tree = 7, number of trees = 500. Precision = 0.9982, Recall = 0.9999, Accuracy = 0.9984
25. Model 25: max depth of tree = 7, number of trees = 5000. Precision = 0.9989, Recall = 0.9986, Accuracy = 0.9977

- The figure below visualizes the accuracy vs the number of trees with other parameters taking their default values.

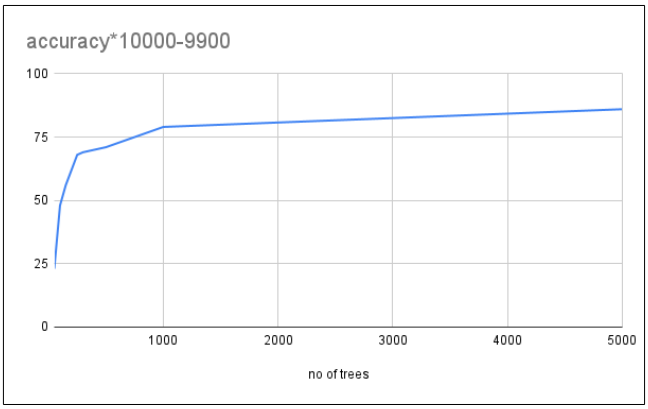

Figure 7: Caption

- For the decision tree, Precision = 0.9974, Recall = 0.9978, Accuracy = 0.9959. The best Precision that I could get was 0.9989 (when 5000 trees were used). The best Recall that I could get was 1.0 (when learning rate was 0.05). The best Accuracy that I could get was 0.9986 (when 5000 trees were used).