Sri Hari Malla
Indian Institute of Technology Hyderabad
Fundamentals of Machine Learning
CS19BTECH11039

Theory Questions

Sri Hari Malla - CS19BTECH11039

October 29, 2021

1 Neural Networks

1.1

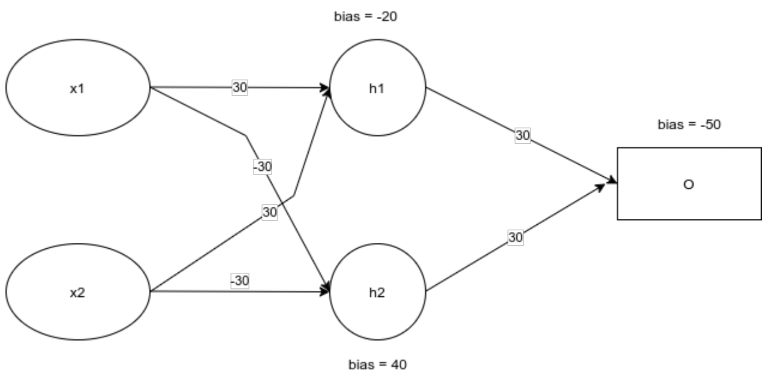


Figure 1: Network Representation

- **case 1** : x1=1 x2=1
  h1 = 30+30-20 = 40 >0 so h1=1
  h2 = -30-30+40 = -20 <0 so h2=0
  y1= 30-50 = -20 <0 so answer is 0.

- **case 2**: x1=1 x2=0
  h1 = 30-20 = 10 >0 so h1=1
  h2 = -30+40 = 10 <0 so h2=1
  y1= 30+30-50 = 10 >0 so answer is 1.

- **case 3**: x1=0 x2=1
  h1 = 30-20 = 10 >0 so h1=1
  h2 = -30+40 = 10 >0 so h2=1
  y1 = >0 so answer is 1.

- **case 4**: x1=0 x2=0
  h1 = -20 < 0 so h1=0
  h2 = 40 >0 so h2=1
  y1= -50 <0 so answer is 0.

Hence, a two layer perceptron with one hidden layer can solve the XOR problem

1.2

Given f, q neurons with functions:

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

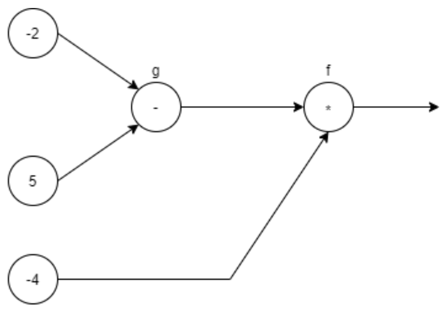


Figure 2: Graphical representation

Calculating the gradient of f with respect to x,y,z:

$$\frac{\partial f}{\partial x} = \frac{\partial (x-y)}{\partial x} * z = z = -4$$
$$\frac{\partial f}{\partial y} = \frac{\partial (x-y)}{\partial y} * z = -z = 4$$
$$\frac{\partial f}{\partial z} = \frac{\partial z}{\partial z} * (x-y) = x - y = -7$$

Sri Hari Malla
Indian Institute of Technology Hyderabad
Fundamentals of Machine Learning
CS19BTECH11039

2 Neural Networks

(2)

Given extension of cross-entropy error function for a multiclass problem is

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

Required to prove:

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

$$\frac{\partial E}{\partial a_k} = \frac{\partial}{\partial a_k}\left(-\sum_j t_j \ln(y_j)\right)$$

$$= -\sum_j t_j \frac{\partial}{\partial a_k}(\ln(y_j))$$

$$= -\sum_j t_j \cdot \frac{1}{y_j}\frac{\partial y_j}{\partial a_k}$$

$$\frac{\partial E}{\partial a_k} = -\frac{t_k}{y_k}\frac{\partial y_k}{\partial a_k} - \sum_{j!=k}^{K} t_j \frac{1}{y_j}\frac{\partial y_j}{\partial a_k} \longrightarrow (1)$$

Given $y_k = \dfrac{\exp(a_k(x,w))}{\sum \exp(a_k(x,w))}$

$$\frac{\partial y_k}{\partial a_k} = \frac{\partial}{\partial a_k}\left(\frac{e^{a_k}}{e^{a_k} + \sum_{j!=k} e^{a_j}}\right)$$

$$= \frac{\left(e^{a_k} + \sum_{j!=k} e^{a_j}\right)e^{a_k} - e^{a_k}e^{a_k}}{\left(\sum_j e^{a_j}\right)^2}$$

$$= \frac{\left(\sum_j e^{a_j}\right)e^{a_k} - \left(e^{a_k}\right)^2}{\left(\sum_j e^{a_j}\right)^2} = \frac{e^{a_k}}{\left(\sum_j e^{a_j}\right)} - \frac{\left(e^{a_k}\right)^2}{\left(\sum_j e^{a_j}\right)^2}$$

$$\frac{\partial y_k}{\partial a_k} = y_k - y_k^2 \longrightarrow (2)$$

Figure 3: page1

Sri Hari Malla
Indian Institute of Technology Hyderabad
Fundamentals of Machine Learning
CS19BTECH11039

$$\frac{\partial y_j}{\partial a_k} = \frac{\partial}{\partial a_k}\left(\frac{e^{a_j}}{e^{a_k}+\sum_{j\neq k} e^{a_j}}\right)$$

$$= \frac{0 - e^{a_j}\cdot e^{a_k}}{\left(\sum_j e^{a_j}\right)^2}$$

$$\frac{\partial y_j}{\partial a_k} = -y_j \cdot y_k \longrightarrow (3)$$

Substituting (2) and (3) in (1)

$$\frac{\partial E}{\partial a_k} = -\frac{t_k}{y_k}\left(y_k - y_k^2\right) - \sum_{j\neq k}^{K} t_j \frac{1}{y_j}\left(-y_j y_k\right)$$

$$= -t_k(1-y_k) + \sum_{j\neq k}^{K} t_j y_k$$

$$= -t_k + t_k y_k + \sum_{j\neq k}^{K} t_j y_k$$

$$\frac{\partial E}{\partial a_k} = -t_k + y_k\left(\sum_j t_j\right)$$

$$\sum_j t_j = 1 \Rightarrow \frac{\partial E}{\partial a_k} = y_k - t_k$$

Hence shown

Figure 4: page2

3 Jensen's Inequality

For a real convex function ϕ numbers $x_1, x_2 \ldots\ldots x_n$ in it's domain and positive weights a_i, jensen's equality is written as:

$$\phi\left(\frac{\sum a_i x_i}{\sum a_i}\right) \leq \left(\frac{a_i \phi\left(x_i\right)}{\sum a_i}\right)$$

- Lets assume all weights as 1 so that summation of the weights is M.
- Given a convex function f(x)=$(x)^2$ and a convex function y.
- let's assume $x_i = [(y_m(x) - f(x))^2]$ and the real convex function ϕ is E_x.

$$E_{AV} = \frac{1}{M} \sum_{m=1}^{M} E_x \left[y_m(x) - f(x)^2\right]$$
$$E_{ENS} = E_x \left[\frac{1}{M} \sum_{m=1}^{M} y_m(x) - f(x)^2\right]$$

According to the Jensen's inequality we can state that $E_{ENS} \leq E_{AV}$ irrespective of E(y).

4 Random Forests

4.1 Own Implementation of Random Forest:

Observations from Code:

```
Accuracy score with own random forest :  0.9029688631426502
--- Time taken --- 37.53448295593262 seconds ---

Accuracy score with inbuilt random forest :  0.9312092686459088
--- Time taken --- 0.052702903747558594 seconds ---
```

- Used Test train split to split the Data into training and test sets.
- The implementation part of the Code can be found in the zip folder.
- The Accuracy obtained with Developed Random forest is close to the accuracy obtained with Inbuilt Random forest.
- Time taken for developed random forest is far greater than that of Inbuilt Random forest Classifier.
- The Inbuilt Random forest classifier is more optimised in all aspects than our code.
- These are the conclusions obtained after running the code for several times.

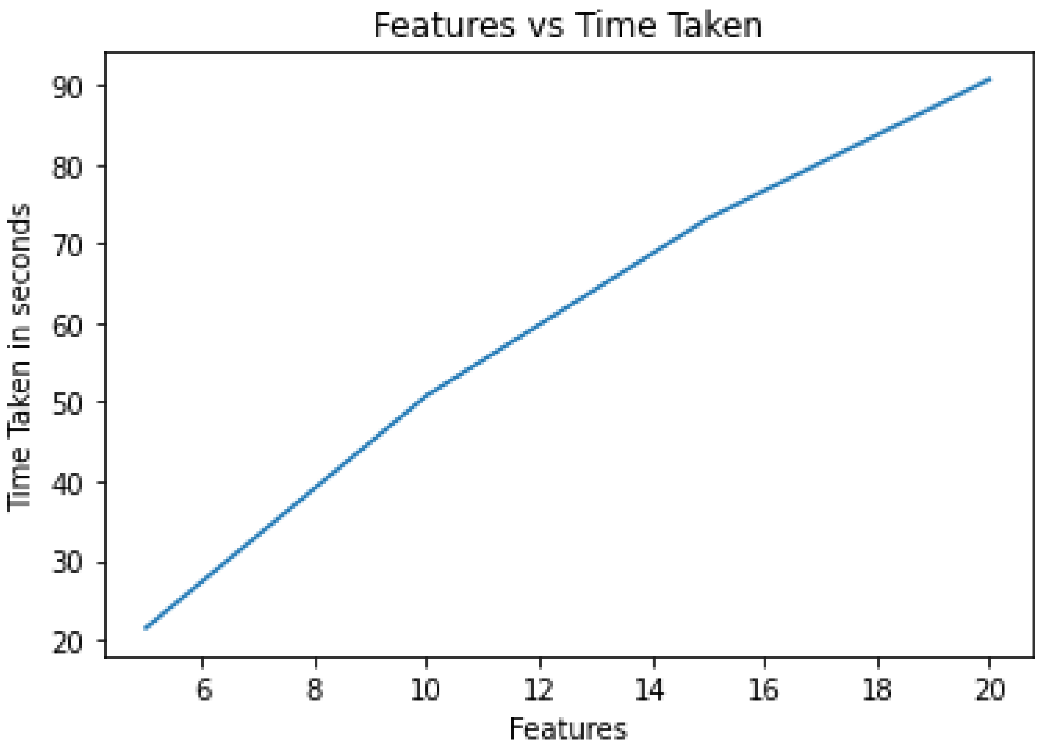
4.2 Variation of sensitivity with Number of Features

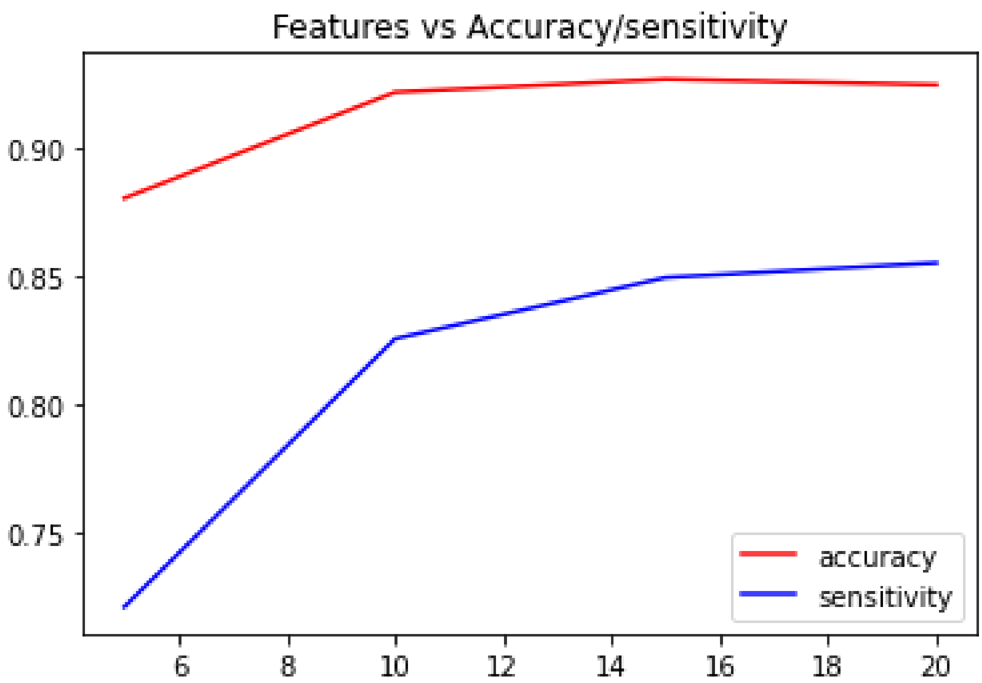
- Please see the code in the zip folder.

Observations from Code:

```
Number of features :  5 , Accuracy :  0.8805213613323678 ,
Time taken :  21.47656774520874 s, Recall Score :  0.7213114754098361

Number of features :  10 , Accuracy :  0.9217958001448225 ,
Time taken :  50.81852197647095 s, Recall Score :  0.8257575757575758

Number of features :  15 , Accuracy :  0.9268645908761767 ,
Time taken :  73.12491822242737 s, Recall Score :  0.8495726495726496

Number of features :  20 , Accuracy :  0.9246922519913107 ,
Time taken :  90.66081619262695 s, Recall Score :  0.8553113553113553
```

Visualization:


Features vs Time Taken
Time Taken in seconds
Features
90
80
70
60
50
40
30
20
6
8
10
12
14
16
18
20


Features vs Accuracy/sensitivity
0.90
0.85
0.80
0.75
6
8
10
12
14
16
18
20
accuracy
sensitivity

Observations:

- With increase in features, Time taken is increasing linearly.
- With increase in features, Accuracy observed is increasing.
- With increase in features, Sensitivity is increasing.
- sensitivity is Calculated using `recall_score`
- Accuracy is always higher than sensitivity as observed from the graph.
- So with increase in features, Accuracy, sensitivity and time taken are increasing. To get a reasonable time, accuracy, sensitivity, the average of start and end is a viable option.

4.3 Exploring OOB error and Test error

Observations from code:

```
Number of Features :  6
OOB error :  0.09924550203134064
Test error :  0.07458363504706733

Number of Features :  7
OOB error :  0.09994222992489887
Test error :  0.07965242577842147

Number of Features :  8
OOB error :  0.10136336692353287
Test error :  0.07965242577842147

Number of Features :  10
OOB error :  0.11601884570082455
Test error :  0.09341057204923964

Number of Features :  12
OOB error :  0.09723040659988214
Test error :  0.07965242577842147
```

Visualization:

Observations:

- It is clear from the graph that test error decreases as features increasing.
- From the graph, OOB error also seems to be decreasing as features are increasing.
- At any point of time and with any number of features, OOB error is always more than the test error.
- The graph is observed for many times and every time new skewed values are obtained.

5 Pre-Processing and Gradient Boost Classifier

5.1 Pre-processing

- Removed columns having NaN values
- Removed Unique Columns in the data set
- Identify the columns which has a value which is predominant from the rest.
- Drop the identified columns

- Identify the columns which has data types other than python float
- Label encode the columns, (One hot key encoder also works but yielding almost same values)
- Split the data into test data and train data.
- Now the data is pre processed and ready for further evaluation.

5.2 Gradient Boosting Classifier

5.2.1 Best Accuracy,precision,recall :

Many models were built, best recorded values are:

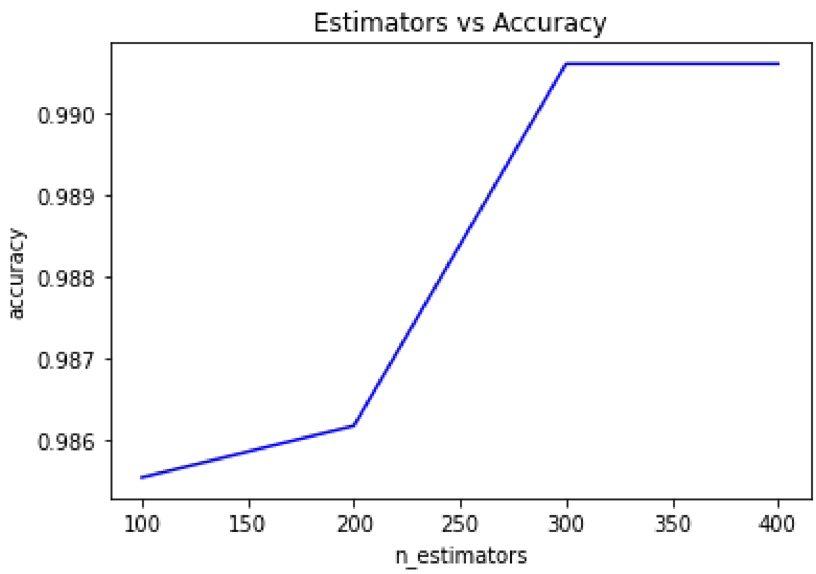
```
Best Recorded Accuracy : 0.9931256238101282
Best Precision :  0.9902622310276867
Best Recall :  0.9712843880195519
HyperParameters : N_estimators = 500, Learning_rate = 0.6, max_depth = 10
```

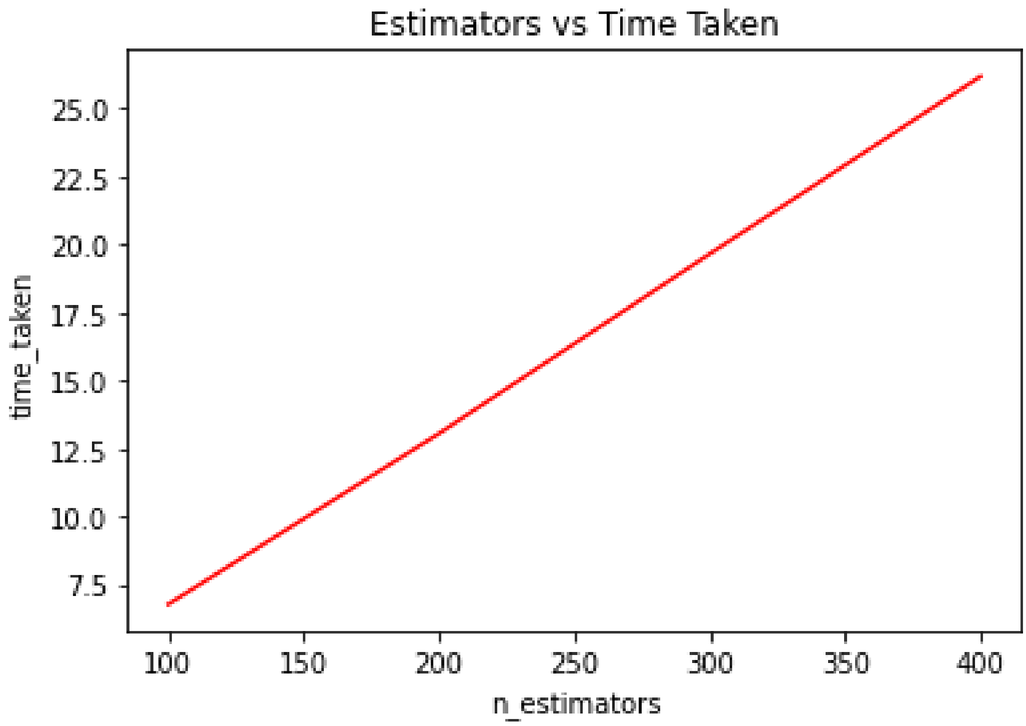
5.2.2 Change in Accuracy/Time vs Estimators

Observed values:

```
Estimators = [100,200,300,400]
Accuracy :  [0.9855311587764944, 0.986165757075771,
    0.9906079451707069, 0.9906079451707069]
Time Taken :  [6.787179231643677, 13.06709623336792,
    19.666242599487305, 26.19341540336609]
```

Visualisation:





5.2.3 Decision Tree Comparision

Accuracy, precision, recall using Decision Tree:

```
Accuracy :  0.9854042391166392
Precision :  0.9714219913751554
Recall :  0.9689849424269265
```

Comparision:
Clearly, gradient boost classifier is out performing Accuracy, precision and recall when compared to Single decision tree.

LaTeX generated document

THE END