FIS 630 Homework 4

Ketan Kokane

April 4, 2019

Problem 1 (20 points)

(a): Draw Decision tree for XOR function.
Solution

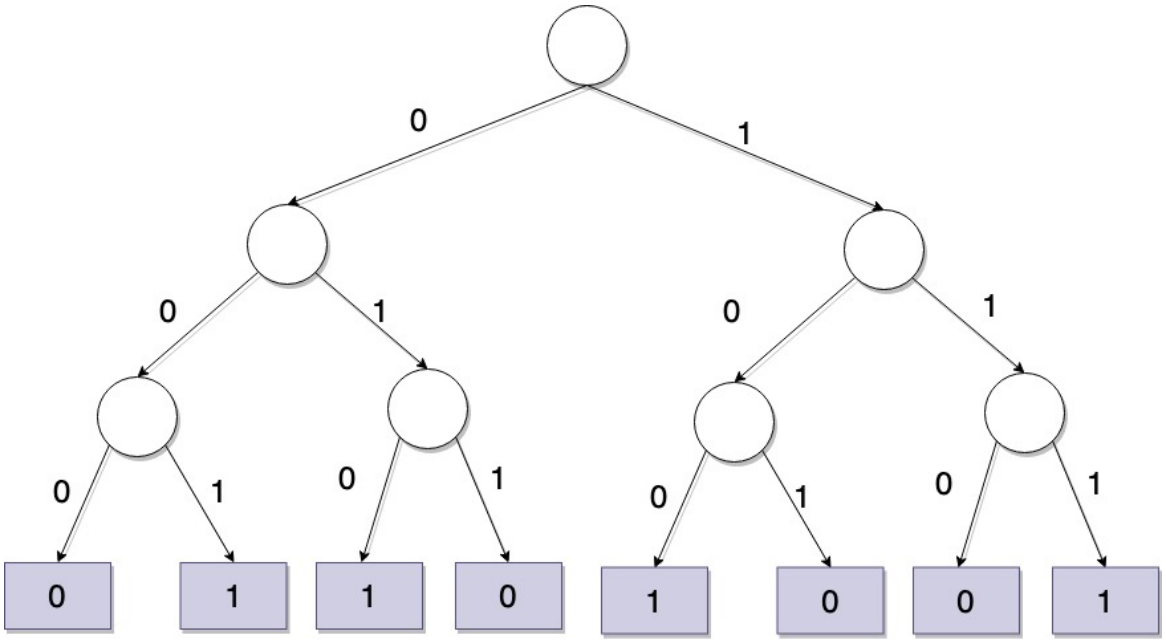

Figure 1: The Decision tree for $XORfunctionwiththreeinputs$

(b) Decision graph

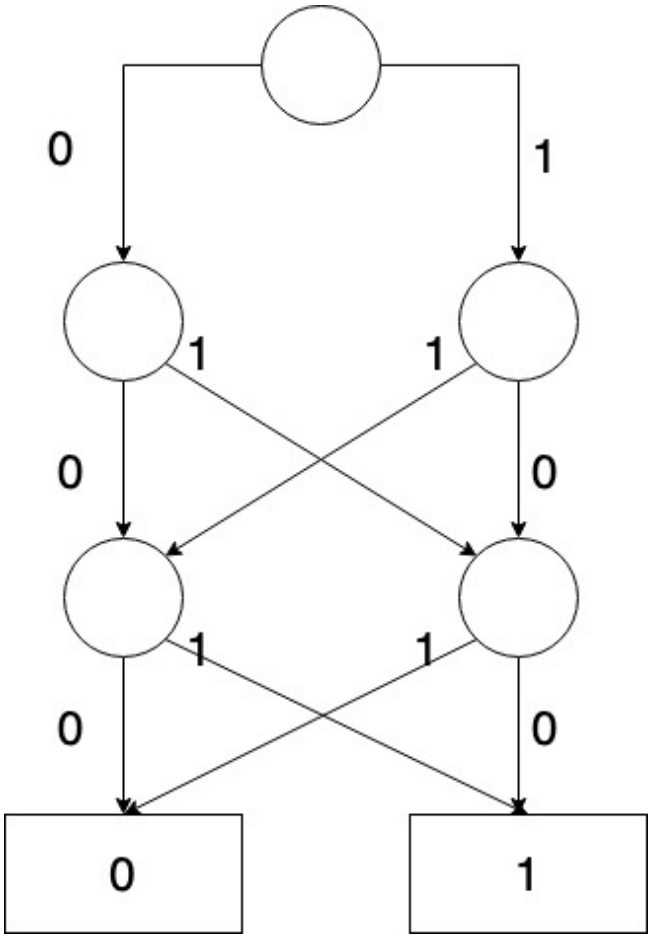

Figure 2: The Decision graph for $XORfunctionwiththreeinputs$ based on the tree in Figure 1

Problem 2 (20 points)

Use only basic data libraries, like pandas.
Suppose a 7-nearest-neighbors regression search returns $7, 6, 8, 4, 7, 11, 100$ as the 7 nearest y values for a given x value. What is the value of $\hat{y}$ that minimizes the $L1$ loss function on this data? There is a common name in statistics for this value as a function of the y values; what is it? Answer the same two questions for the $L2$ loss function.

Solution
L_1 Loss function to minimize the error which is the sum of absolute difference between $\hat{y}$ and y_i

$$L_1 = \sum_{i=1}^{n} |\hat{y} - y_i|$$

The name for the L_1 function in statistics is **Sum of Absolute Error**.
$L_1 = |\hat{y} - 7| + |\hat{y} - 6| + |\hat{y} - 8| + |\hat{y} - 4| + |\hat{y} - 7| + |\hat{y} - 11| + |\hat{y} - 100|$
Value of L_1 will be smallest for value 7 i.ie = 102
And the value $\hat{y} = 7$ is the median of the given values.
L_2 Loss function to minimize the error which is the sum of all the squared difference between $\hat{y}$ and y_i

$$L_2 = \sum_{i=1}^{n} (\hat{y} - y_i)^2$$

The name for the L_2 function in statistics is **Sum of Squared Errors**.

$L_2 = (\hat{y}-7)^2 + (\hat{y}-6)^2 + (\hat{y}-8)^2 + (\hat{y}-4)^2 + (\hat{y}-7)^2 + (\hat{y}-11)^2 + (\hat{y}-100)^2$
Taking the derivative WRT to will result in minimum value of $\hat{y}$ when the derivative = 0
$\frac{\partial L_2}{\partial \hat{y}} = 2(\hat{y}-7) + 2(\hat{y}-6) + 2(\hat{y}-8) + 2(\hat{y}-4) + 2(\hat{y}-7) + 2(\hat{y}-11) + 2(\hat{y}-100)$
$0 = 2(\hat{y}-7) + 2(\hat{y}-6) + 2(\hat{y}-8) + 2(\hat{y}-4) + 2(\hat{y}-7) + 2(\hat{y}-11) + 2(\hat{y}-100)$
$\hat{y} = 20.42$
which is the mean of the given values
The value in statistics for function L_2 is called the mean of the given values.

Problem 3 (20 points)

Consider an ensemble learning algorithm that uses simple majority voting among K learned hypotheses. Suppose that each hypothesis has error and that the errors made by each hypothesis are independent of the others. Calculate a formula for the error of the ensemble algorithm in terms of K and ϵ, and evaluate it for the cases where K = 5, 10, and 20 and = 0.1, 0.2, and 0.4. If the independence assumption is removed, is it possible for the ensemble error to be worse than ϵ?

Solution
Example, An Ensemble decision tree having K trees, using majority voting to decide the final classification of the ensemble. $\epsilon = 0.1$ meaning that the probability of 1 out 10 instances is mis-classified. So based on the majority voting if at-least 3 out of 5 trees mis-classify then the entire ensemble mis-classifies the instance.
This means that for the ensemble to give error. At least 3 of any 5 tress should have produced the error or any 4 of them should have produced the error or all of them should have produced the error.

Thus the formula for calculating the error of the ensemble is
(5 choose 3) * $error^3$ * $success^2$ + (5 choose 4) $error^4$ * $success$ + 5 choose 5 $error^5$
thus formula

$$= \sum_{i=(K+1/2)}^{K} \binom{K}{i} \epsilon^i * (1-\epsilon)^{K-i}$$

($K = 5$, $\epsilon = 0.1$)

$$\sum_{i=(3)}^{5} \binom{5}{i} (0.1)^i * (1-(0.1))^{5-i}$$

$$= 0.008560000000000002$$

($K = 10$, $\epsilon = 0.1$)

$$\sum_{i=6}^{10} \binom{10}{i} (0.1)^i * (1-(0.1))^{10-i}$$

$$= 0.00014690260000000007$$

($K = 20$, $\epsilon = 0.1$)

$$\sum_{i=11}^{20} \binom{20}{i} (0.1)^i * (1-(0.1))^{20-i}$$

$$= 7.088606331722206e-07$$

($K = 5$, $\epsilon = 0.2$)

$$\sum_{i=(3)}^{5} \binom{5}{i} (0.2)^i * (1-(0.2))^{5-i}$$

$$= 0.05792000000000003$$

$(K = 10,\ \epsilon = 0.2)$

$$\sum_{i=6}^{10} \binom{10}{i} (0.2)^i * (1 - (0.2))^{10-i}$$
$$= 0.006369382400000004$$

$(K = 20,\ \epsilon = 0.2)$

$$\sum_{i=11}^{20} \binom{20}{i} (0.2)^i * (1 - (0.2))^{20-i}$$
$$= 0.0005634136976601912$$

$(K = 5,\ \epsilon = 0.4)$

$$\sum_{i=(3)}^{5} \binom{5}{i} (0.4)^i * (1 - (0.4))^{5-i}$$
$$= 0.3174400000000001$$

$(K = 10,\ \epsilon = 0.4)$

$$\sum_{i=6}^{10} \binom{10}{i} (0.4)^i * (1 - (0.4))^{10-i}$$
$$= 0.16623861760000005$$

$(K = 20,\ \epsilon = 0.4)$

$$\sum_{i=11}^{20} \binom{20}{i} (0.4)^i * (1 - (0.4))^{20-i}$$
$$= 0.12752124614721672$$

```
In [26]: import math

         def choose(K, i):
             return int(math.factorial(K)/(math.factorial(i) * math.factorial(K-i)))

         def getError(K, error):
             result = 0
             majority = math.ceil((K+1)/2)
             for i in range(majority, K + 1):
                 result += choose(K, i ) * math.pow(error,i) * math.pow(1-error, K-i)
             return result

         getError(5,0.1)

Out[26]: 0.008560000000000002
```

Figure 3: Above code snippet was used to calculate the required probabilities

(if the Independence assumption is removed)
Yes, the error of the ensemble would be worsened if the assumption that the every tree in the ensemble is independent of each other is removed. for Eg take an ensemble of K = 3 (K1, k2 and k3)and $\epsilon = 2/5$, meaning any tree can make 2 out 5 predictions wrong. so lets take 5 examples and run through the ensemble
for example 1 : K1 and K2 makes error thus ensemble makes error
for example 2 : K1 and K3 makes error thus ensemble makes error
for example 3 : K3 and K2 makes error thus ensemble makes error
for example 4 and 5 no one tree any error.
Thus for 5 examples the ensemble made an error of 3/5 which is more than individual error made by tree.

Problem 4 (20 points)

Construct by hand a neural network that computes NOR. (The best was is to draw a graph and label the edges w/ the weights. You may also want to give the nodes of the graph meaningful labels)
Solution

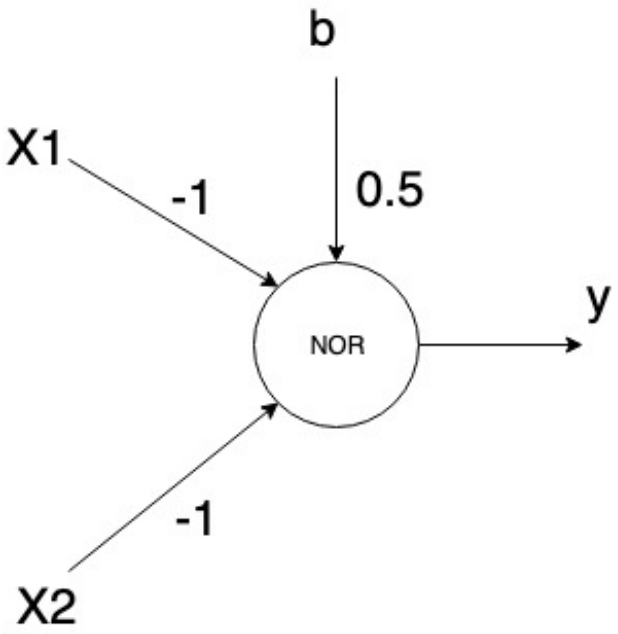

Figure 4: A single perceptron for NOR function with two inputs

X1	X2	-X1	-X2	b	-X1 -X2 + b	y
0	0	0	0	0.5	0.5	1
0	1	0	-1	0.5	-0.5	0
1	0	-1	0	0.5	-0.5	0
1	1	-1	-1	0.5	-1.5	0

if $-X1 - X2 + b > 0$ then $y = 1$ Else $y = 0$

Problem 5 (20 points)

You were given the following data on two different but closely related subspecies of alligators, one from Florida, the other from Texas. Write (in python) a single layer perceptron that can detect the origin of alligator, given weight and length. Use only basic data libraries, like pandas. In particular, you should write your own back propagation algorithm and clearly indicate in your comments where it is. Submit source code as a section of your pdf submission. (You do not have to report accuracy, but you might want to note it).
Solution

Weights(lgs)	*Length(in) —*	*origin?*
11	70	T
35	11	F
21	45	T
60	80	F
37	32	F
26	64	T
44	30	F
12	60	T

Table 1: Dataset $\mathcal{D}$

After plotting the dataset, it can be seen that the a linear classifier would be enough to classify the data into two classes. Thus the designed the below single layer perceptron for the required classification.

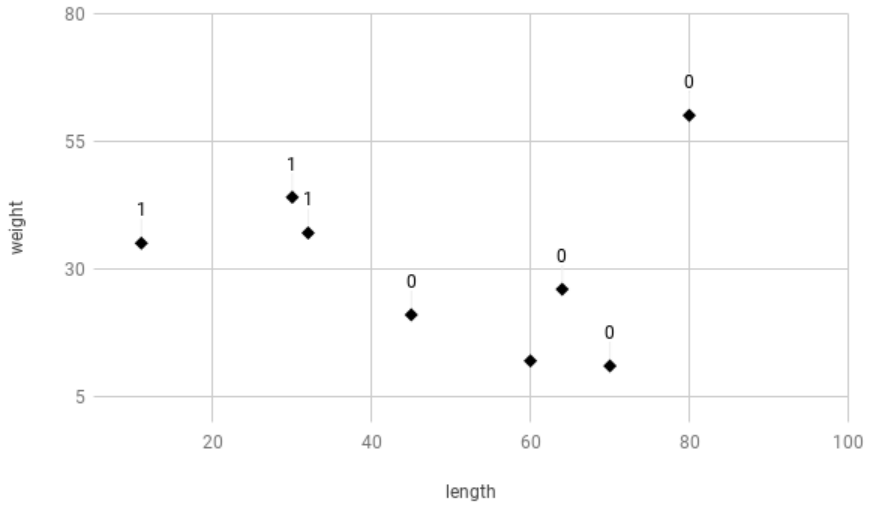

Figure 5: Above shows an abstract visualization of the dataset, and used for analysis purposes and may not match exactly to given dataset

Output of the perceptron:

$$z = weight * w1 + length * w2 + b$$

Output = $threshold(z)$
the threshold function returns the output as 1 if the $z > 0$, else returns 0
Error = TrueOutput - PredictedValue
Weight/bias update rule on prcoess of i^{th} instance of given data

$$w1 = w1 + \alpha * error * weight_i$$
$$w2 = w2 + \alpha * error * length_i$$
$$b = b + \alpha * error * 1$$

After training the perceptron on the entire dataset, it can be seen that the perceptron converges after 6 training examples.

After training the examples on 6 data points the last two data points are used for testing. As they both belong to separate classes. The accuracy given by the classifier is 100%.

Attached code shows the weight updates matrix and the classification of two data points which the network hasn't seen before. The Complete code is in classifer.py file.
To execute the file

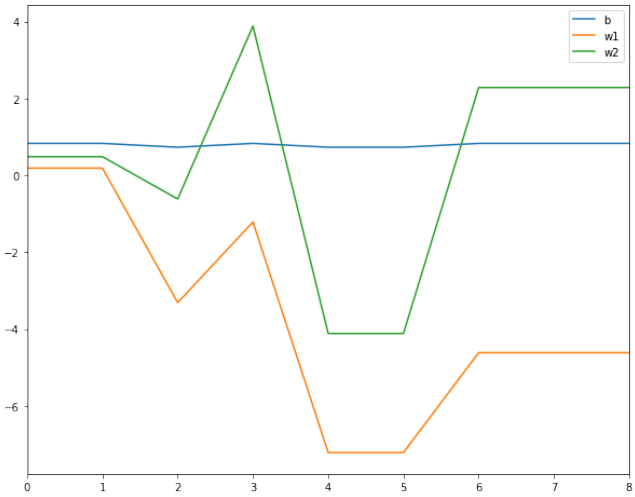

Figure 6: The trend of weights of the neuron

```
python classfier.py
```