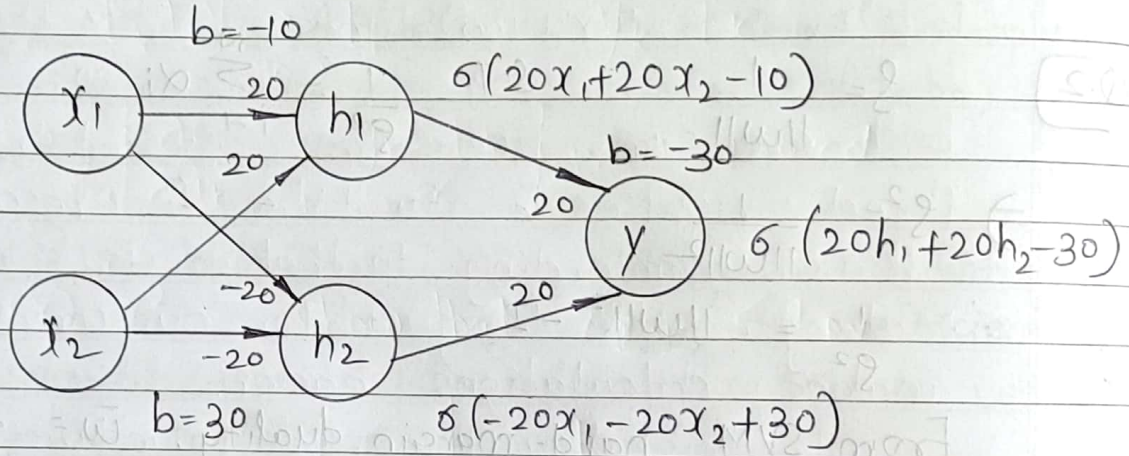
Foundations of Machine Learning
Assignment - 3

Q.1] a) Neural Networks
XOR problem with one hidden layer



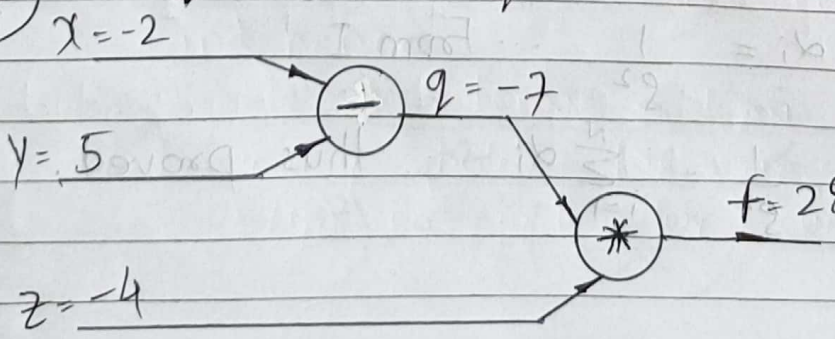
i) For (0,0)
$\sigma(20\times0+20\times0-10)\approx0$, $\sigma(-20\times0-20\times0+30)\approx1$,
$\sigma(20\times0+20\times1-30)\approx0$

ii) For (0,1)
$\sigma(20\times0+20\times1-10)\approx1$, $\sigma(-20\times0-20\times1+30)\approx1$,
$\sigma(20\times1+20\times1-30)\approx1$

iii) For (1,0)
$\sigma(20\times1+20\times0-10)\approx1$, $\sigma(-20\times1-20\times0+30)\approx1$,
$\sigma(20\times1+20\times1-30)\approx1$

iv) For (1,1)
$\sigma(20\times1+20\times1-10)\approx1$, $\sigma(-20\times1-20\times1+30)\approx0$,
$\sigma(20\times1+20\times0-30)\approx0$

b) $q=x-y$, $f=q*z$



1) $\frac{\partial f}{\partial z} = -7$ 2) $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial x} = -4 \cdot 1 = -4$

3) $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y} = -4 \cdot (-1) = 4$

$\therefore$ Ans: $\frac{\partial f}{\partial x} = -4, \frac{\partial f}{\partial y} = 4, \frac{\partial f}{\partial z} = -7.$

Q.2) Neural networks

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

Softmax activation function: $y_k(x,w) = \frac{\exp(a_k(x,w))}{\sum_j \exp(a_j(x,w))}$

$\sum_k y_k = 1$ ... given

$\Rightarrow$ Derivative of softmax functn is given by.

If $k=j$: $\frac{\partial y_k}{\partial z_k} = \frac{\partial \frac{e^{a_k}}{\sum_j a_j}}{\partial z_k} = \frac{e^{a_k}\sum_j - e^{a_k} e}{\sum_j^2 a_j} = y_k(1-y_k)$ -I

If $k \neq j$, $\frac{\partial y_k}{\partial z_j} = \frac{\partial \frac{e^{a_k}}{\sum_j a_j}}{\partial z_j} = -y_k y_j$ -II

Derivative of error function with respect to activation function., sigmoid

$$\frac{\partial E}{\partial a_k} = \frac{-\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \log(y_k)}{\partial a_k} = -\sum_{n=1}^{N} t_{kn} \frac{\partial \log(y_n)}{\partial a_k}$$

$$= -\sum_{n=1}^{N} t_n \frac{1}{y_n} \frac{\partial y_n}{\partial a_k} = \frac{-t_k}{y_k} \frac{\partial y_k}{\partial a_k} - \sum_{n \neq k}^{N} \frac{t_n}{y_n} \frac{\partial y_n}{\partial a_k}$$

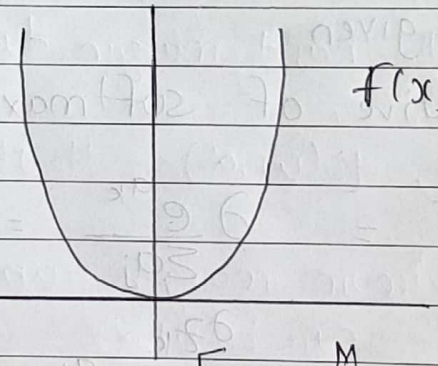
$$= \frac{-t_k}{y_k} y_k(1-y_k) - \sum_{n \neq k}^{N} \frac{t_k}{y_k}(-y_n y_k)$$

$$= -t_k + t_k y_k + \sum_{n \neq k}^{N} t_n y_k \Rightarrow -t_k + \sum_{n=1}^{N} t_n y_k$$

$$= -t_k + y_k \sum_{n=1}^{N} t_n = y_k - t_k$$

$$\therefore \boxed{\frac{\partial E}{\partial a_k} = y_k - t_k}$$ is proved

Q.3] $f(x) = x^2$ Show that $E_{ENS} \leq E_{AV}$



We know,

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

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

Rearranging the both sides.

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

Here, all terms of E_{ENS} contained in E_{AN} and hence

$E_{ENS} \leq E_{AN}$ is proved.

It basically holds for any error function $E(y)$. not just for sum of squares.

X	X^2	$P(X=x)$
1	1	1/6
2	4	1/6
3	9	1/6
4	16	1/6
5	25	1/6
6	36	1/6

$g: X \to X^2$

$E(X) = \sum_x x.P(x) = \frac{1}{6}(1+2+\ldots 6)$

$= 3.5$

$E(X^2) = 12.25$

$\therefore E(X^2) > E(X)$

Basically Jensen's inequality is used to bound the complicated expression E_{AV} by simpler expression E_{ENS}

* Jensen's inequality for finite M by induction on the number of elements of M. Suppose M contains k elements and assume it holds for distribution on k-1 points. From induction hypothesis,

$E_{AV} = p_1 f(x_1) + p_2 f(x_2)$

$\geq f(p_1 x_1 + p_2 x_2)$

$\geq E_{ENS}$ For $M=2$

$\therefore$ $E_{AV} \geq E_{ENS}$ proved for convex functions

Programming Questions

Q.4) Random Forests

Ans.

a) In this question, Random Forest algorithm is implemented from scratch. Original dataset is divided into 70% in training and 30% in testing. Following table shows trees created using random forest algorithm, accuracy and time required to run the algorithm using scratch code and Scikitlearn's built-in random forest classifier.

No. of Trees	Accuracy		Time Taken	
	using Random Forest algorithm (in %)	using Scikitlearn's built-in function (in %)	with Random Forest algorithm (in secs)	with Scikitlearn's built-in function (in secs)
5	85.952	93.773	4.54	0.03
8	89.066	94.424	6.76	0.04
13	90.297	95.293	11.9	0.07
22	91.311	95.51	20.89	0.12
37	90.587	95.293	35.51	0.17
45	92.397	95.366	41.99	0.22
50	91.745	95.583	46.84	0.26
67	91.673	95.221	62.16	0.31
80	91.528	95.583	75.77	0.37
100	92.397	95.51	95.06	0.59

b) For exploring the sensitivity with respect to number of features used for best split, accuracy with respect to parameter 'm' is calculated. At start, when we increase the value of m, accuracy increases i.e., the model is very sensitive and then the accuracy becomes almost constant.

'm'	Accuracy (in %)
1	82.187
2	84.432
4	86.097
6	88.776
10	86.097
14	88.052
18	87.835
24	86.314
28	86.676
36	87.328
42	87.835
48	87.473
50	87.4

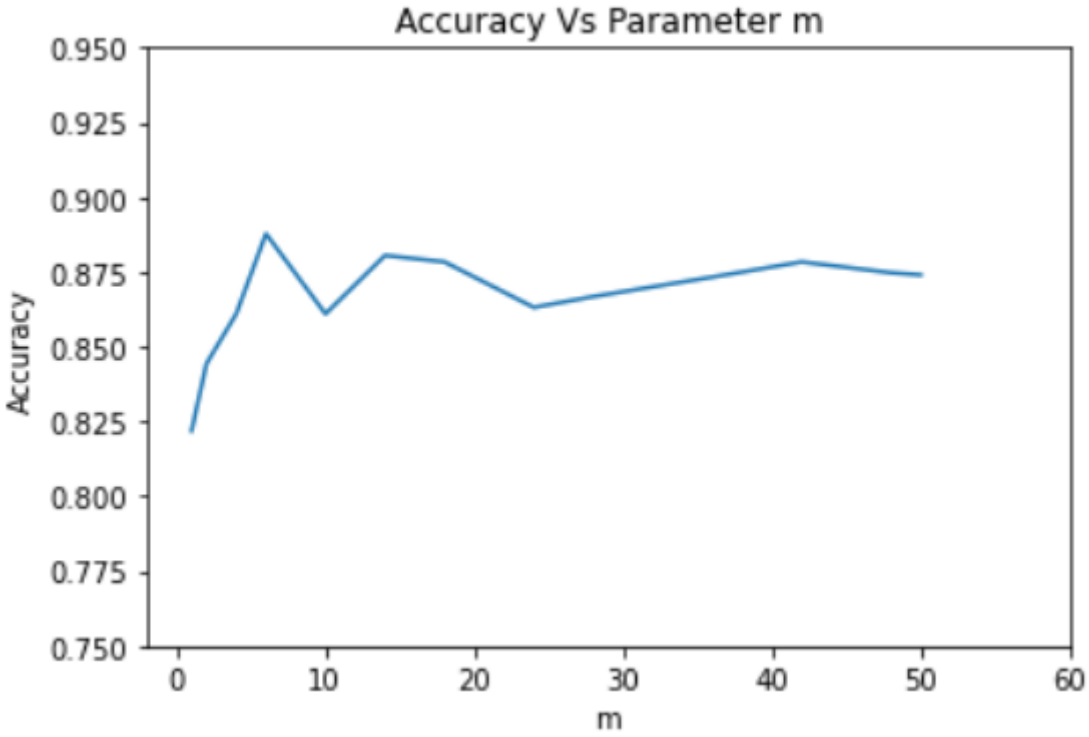


Fig.1: Plot of Accuracy Vs Parameter m.

c) OOB (out-of-bag) error and testing error is calculated against the values of m.

'm'	OOB Error	Testing Error
1	0.11007	0.113067
2	0.10282	0.12496
4	0.09413	0.09779
6	0.08834	0.11125
10	0.10065	0.10834
14	0.09558	0.1103
18	0.09558	0.1109
24	0.09848	0.11161
28	0.10427	0.1036
36	0.10355	0.11508
42	0.09993	0.11189
48	0.10138	0.10807
50	0.09848	0.10163
56	0.10644	0.11601

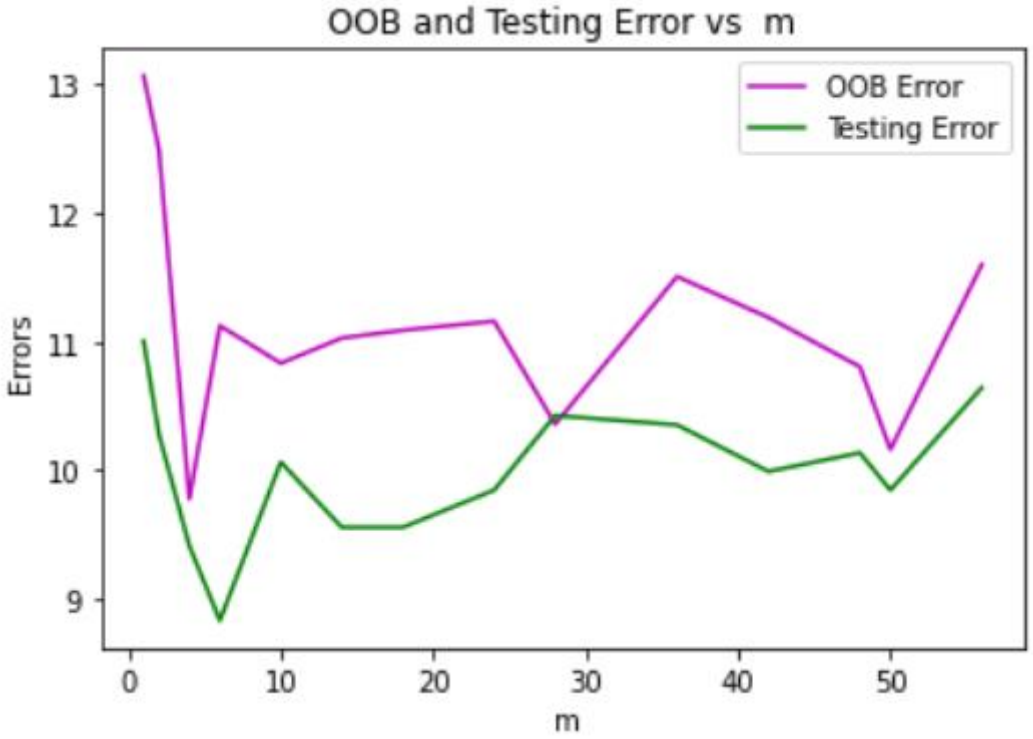


Fig.2: OOB and Testing Error is plotted against the value of m.

Q.5) Gradient Boosting

Ans.

a) In this question, initially training data is loaded. Then specific columns in training data are selected, others are dropped. All rows with **'Current'** are dropped from **loan_status**. Checked, if any column contains missing values or not. Observed, that only one column contains **40** missing values. These missing values have replaced with '**Mode**' of the attribute. Converted categorical columns into binary i.e., into numerical features. Same pre-processing steps are applied on testing data. Then, normalization has done on both the data.

b) Applied Gradient Boosting Algorithm and created four model using different hyperparameters.
Following table shows respective model, its specifications, accuracy, precision and recall values.

Model No.	Specifications	Accuracy	Precision	Recall
1(default)	loss='deviance', learning_rate=0.1, n_estimators=100, max_depth=3, max_features=None, random_state=None	96.014%	0.95726	0.99761
2	loss='exponential', learning_rate=1, n_estimators=10, max_depth=3, max_features=2, random_state=None	94.859%	0.94353	0.99926
3	loss='deviance', learning_rate=10, n_estimators=200, max_depth=5, max_features=2, random_state=None	73.172%	0.93846	0.73208
4	loss='deviance', learning_rate=0.01, n_estimators=500, max_depth=1, max_features=None, random_state=None	96.084%	0.95592	1

The best accuracy calculated over entire test set was 96.084% and the contributed hyperparameters are deviance loss, 0.01 learning rate, 500 estimators, etc.

A decision tree model is built using Scikit-learns inbuilt decision tree function. Accuracy calculated is 91.72%. Value of Decision Tree Precision and Recall calculated are 0.95959 and 0.94218 respectively.

Comparison of best model performance against the decision tree function.

The accuracy of gradient boosting function is 96.084, is larger than decision tree function, is 91.72%. Precision of decision tree function is larger than gradient boosting function while recall is lower. The contributed hyperparameters in decision tree function are gini criterion, best splitter, etc. while in gradient boosting functions are loss, learning rate, estimators, etc.