E1 213 – Pattern Recognition and Neural Networks

Assignment # 2

Harshit Samani **M.Tech Artificial Intelligence** **SR No: 17862**

Problem 1

Problem 1 is a 2-class classification problem. We have to compare linear least squares and logistic regression methods for learning a linear classifier. The problem has three subproblems. In each case we have 2000 training samples and 1000 test samples.

Sub problem A

For subproblem a, the data is two dimensional and the class conditional densities are given to be Gamma distributed. The number of data samples used for training the regressions is varied as 10, 20, 50, 100, 250, 500, 1000, 1500, 2000 (random sampling from training data). Using this data, linear least squares regression and logistic regression were implemented. The performance of both the classifiers were observed by testing against the full test set (which is tabulated below).

N (training data size)	Linear regression accuracy	Logistic regression accuracy
10	0.77	0.87
20	0.83	0.89
50	0.83	0.91
100	0.87	0.93
250	0.88	0.92
500	0.87	0.93
1000	0.88	0.94
1500	0.88	0.94
2000	0.87	0.94

Th efficiency of both the classifiers increases (neglecting initial spikes and dips due to lesser number of data samples) with the number of samples used for training the regression. This is quite obvious from the fact that a greater number of samples leads to better estimation of parameters. We can also see that in general logistic regression outperforms linear regression, which is shown to be almost always the case, empirically (for classification). This turns out to be particularly true when the number of samples is quite large, which contributes in finding better estimates for parameters. The low accuracy of the classifiers (compared to other subproblems in this problem) maybe due to the similar distributions from which samples are drawn from.

Sub problem B

** GitHub repository access is private as of now. Access can be made available, if necessary.*

For subproblem b, the data is two dimensional and the class conditional densities are given to be Uniform distributed. The number of data samples used for training the regressions is varied as 10, 20, 50, 100, 250, 500, 1000, 1500, 2000 (random sampling from training data). Using this data, linear least squares regression and logistic regression were implemented. The performance of both the classifiers were observed by testing against the full test set (which is tabulated below).

N (training data size)	**Linear regression accuracy**	**Logistic regression accuracy**
10	0.86	0.95
20	0.92	0.96
50	0.92	0.97
100	0.92	0.99
250	0.93	0.98
500	0.92	0.99
1000	0.93	0.99
1500	0.93	0.99
2000	0.93	0.99

The accuracy of both the classifiers increases (neglecting rare dips) with the number of samples owing to the reasons explained in subproblem a. The logistic regression classifier performs exceptionally well in classification of test data for which one of the reasons could be relatively lesser overlap between the class conditional densities – one class being uniform over [0.5, 6.0] x [0.5, 6.0] and other class being uniform over [0,1] x [0,1] with means at [3.25, 3.25] and [0.5, 0.5] respectively.

Sub problem C

For subproblem c, the data is ten dimensional and the class conditional densities are given to be Gaussian Normal distributed. The number of data samples used for training the regressions is varied as 10, 20, 50, 100, 250, 500, 1000, 1500, 2000 (random sampling from training data). Using this data, linear least squares regression and logistic regression were implemented. The performance of both the classifiers were observed by testing against the full test set (which is tabulated below).

N (training data size)	**Linear regression accuracy**	**Logistic regression accuracy**
10	0.19	0.90
20	0.83	0.94
50	0.88	0.93
100	0.92	0.92
250	0.93	0.93
500	0.94	0.95
1000	0.94	0.94
1500	0.94	0.94
2000	0.94	0.94

The accuracy of both the classifier increases (neglecting rare dips) with the number of samples owing to the reasons explained in subproblem a. Interestingly, the accuracy for both linear and logistic regression are approximately same for N >= 100. I tried to find whether that's always true for large dimensional data, but failed to find any evidence supporting that.

Problem 2

Problem 2 is a 3-class classification problem with Iris dataset. The features represent the measurements of few attributes of a flower. There are 150 input samples in total, which we need to split for training and testing.

The standard ML practice is to split the dataset into 70% training set, 15% development set and 15% testing set. However, we are not performing any industry level training and hence the development set can be avoided. Further due to the extremely low number of data samples compared to typical ML datasets, I decided to split the dataset into 80% training set and 20% testing set.

Using the data, two classifiers were implemented and trained –

i. three linear 2-class classifiers using 'one vs rest' strategy learnt through linear least squares
ii. a 3-class linear classifier (by taking the target variable as a 3-dimensional one-hot vector) learnt through linear least squares

The trained 'one vs rest' classifier attained an accuracy of 80% and '3-class one hot target vector' classifier attained an accuracy of 80% on test set. I also experimented by varying the train-test split ratio to study the effect of the split value (results are tabulated below). A dataset split of 80% training and 20% testing does perform well compared to the rest.

Train data ratio	**One vs Rest classifier Accuracy**	**3-class one hot classifier accuracy**
0.9	0.67	0.67
0.8	0.80	0.80
0.7	0.80	0.80
0.6	0.77	0.77
0.5	0.77	0.77
0.4	0.81	0.81
0.3	0.81	0.81
0.2	0.83	0.83
0.1	0.82	0.82

One thing we could note from the above is that both 'one vs rest' classifier and '3-class one hot target vector' classifier performs almost similarly for all cases. In a sense, we can think of both these methods to be the same. Both these methods treat a particular class as positive class and treat the other two classes together as a negative class and trains on that.

Problem 3

Problem 3 is a 2-class classification problem with twenty four-dimensional feature vectors. The features represent various financial attributes of a person and the class labels denote whether a person is 'good' or 'bad' for extending credit. Using the data, linear least squares regression and logistic regression was implemented and trained. There are 1000 input samples in total, which we need to split for training and testing.

As for the reasons explained in Problem 2, I decided to split the dataset into 80% training set and 20% testing set. The trained linear regression classifier attained an accuracy of 76% and logistic regression classifier attained an accuracy of 76% on test set. I also experimented by varying the train-test split ratio to study the effect of the split value (results are tabulated below). A dataset split of 80% training and 20% testing does perform well compared to the rest.

Train data ratio	**Linear regression accuracy**	**Logistic regression accuracy**
0.9	0.71	0.70
0.8	0.76	0.77
0.7	0.77	0.77
0.6	0.76	0.77
0.5	0.76	0.75
0.4	0.73	0.75
0.3	0.74	0.75
0.2	0.74	0.74
0.1	0.73	0.71

For this dataset, both linear regression and logistic regression model performs more or less the same. The (relatively) low performance of the classifiers maybe due to the improper feature selections used. Few models available in the web attains up to 86% test accuracy by dropping few of the less important/contributing columns such as age, nationality etc.

Problem 4

Problem 4 is a 1D regression problem. The file contains (x_i, y_i) pairs. The x-y relationship is given by the polynomial $y = f(x) = 0.25x^3 + 1.25x^2 - 3x - 3$. There are 100 input samples in total, which we need to split for training and testing.

As for the reasons explained in Problem 2, I decided to split the dataset into 80% training set and 20% testing set. Since the problem at hand is a continuous value regression problem, I have chosen Root Mean Squared Error (RMSE) and R2 Score as evaluation metrics. The following table summarises the observations with degree of polynomial used for fitting, the RMSE and R2 score for the fit.

Degree	**RMSE**	**R2 Score**
1	14.057	0.193
2	7.415	0.775
3	2.243	0.979
4	2.203	0.980
5	2.242	0.979
6	2.219	0.980
7	2.319	0.978
8	2.371	0.977
9	2.294	0.979
10	2.224	0.980
20	2.156	0.981
40	5.915	0.857
60	9.576	0.625
100	12.342	0.378
150	16.274	-0.082
200	18.220	-0.356
249	26.159	-1.795

We can observe a sharp decrease in RMSE and increase in R2 score till we move till to the third degree fit, after which the values does not change much. We can infer that we have obtained our best fit polynomial, which was $y = 0.246x^3 + 1.243x^2 – 2.861x - 2.261$. If we further increase the degree, till a certain degree, the evaluation metrics doesn't vary much – the coefficients of higher order terms tend to be close to 0 and the lower order terms with that of above obtained best fit polynomial. If we further increase the degree to more than hundred, the model starts to overfit leading to decaying performance – with high RMSE and low (even negative) R2 Score.