PROJECT – 3 REPORT

TEAM MEMBERS: -
SHIVANGI TRIPATHI
PRATIK SANGHVI

Course Number: -
CSE574

In this assignment, we implement Logistic Regression on MNIST data to classify the handwritten digits and implement Support Vector Machine from sklearn to perform classification.

PROBLEM 1.1:

In order to implement the Logistic Regression, we need to implement 3 functions, namely: preprocess(), blrObjFunction() and blrPredict() function.

Preprocess() Function:

We load the MNIST data, divide the train data into train and validation set. Finally, we have a train set, validation set and test set with their respective labels. We perform feature selection and remove features that don't provide any useful information in classification.

The output of this function is Train data, Train labels, Validation data, Validation labels, Test data, Test labels.

The labels associated with digits can take 10 possible values, so this is a multiclass classification problem. We need to employ a one-vs-all strategy and build 10 binary classifiers.

For this, we create a matrix Y, with binary labels. For a given class, each column in Y can have either a 1 or representing if the feature belongs to the class or not.

For each class (column in Y) we call the blrObjfunction to train the model and use gradient descent to give optimum weight vector.

blrObjFunction():

The input to this function:

1. Matrix X with data. It is a NxD matrix, where N is the number of data points and D being number of features. We add the bias term at the start for each feature.
2. A weight vector w, with a size of ((d+1) x 1)
3. A column vector y with labels, size Nx1

We compute the sigmoid function and then compute the error and the error gradient given by the formulae:

$$\boxed{E(\mathbf{w}) = -\frac{1}{N}\ln p(\mathbf{y}|\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\{y_n \ln \theta_n + (1-y_n)\ln(1-\theta_n)\}}$$

$$\boxed{\nabla E(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\theta_n - y_n)\mathbf{x}_n}$$

This are the 2 outputs from the blrObjfunction.

blrPredict():

Input to this function:

1. The data that we want to use for prediction
2. A matrix of weights, each column is a weight vector for the classifiers

The function takes in the inputs, computes the sigmoid and then outputs the class with maximum probability.

Following are the accuracy values:

Training Set Accuracy	92.68%
Validation Set Accuracy	91.49%
Test Set Accuracy	91.92%

A Confusion Matrix helps in understanding the precision and recall of our classifier.

Precision helps in understanding out of all the times the model predicted a class, how many times it was correct.

Recall helps in understanding out of all the times a class should have been predicted, how many were correctly predicted as that class.

Precision is given by the formula:

(True Positives for A) / (True Positives for A + False Negatives for A)

Recall is given by the formula:

True Positives for A / (True Positives for A + False Positives for A)

	TRAINING SET										PRECISION
	4819	1	10	7	8	18	24	6	28	2	0.978875
	1	5622	27	11	3	17	3	11	40	7	0.979101
	33	40	4514	66	52	18	51	62	108	14	0.910448
	17	21	124	4604	8	141	21	42	103	50	0.897291
	8	21	24	7	4544	9	26	12	46	145	0.938455
	43	18	33	129	41	3902	79	19	106	51	0.882606
	23	12	30	3	20	63	4739	4	22	2	0.963603
	12	21	51	12	44	10	4	4965	12	134	0.94302
	40	108	59	117	27	127	37	20	4228	88	0.871573
	25	19	15	84	167	34	1	157	44	4403	0.889675
RECALL	0.959769	0.955635	0.923675	0.913492	0.924705	0.899286	0.950652	0.937146	0.892548	0.899306	

	TEST SET										PRECISION
	961	0	1	2	1	4	6	3	1	1	0.980612
	0	1115	3	1	0	1	4	1	10	0	0.982379
	7	8	919	19	11	5	13	12	33	5	0.890504
	4	1	19	919	2	20	4	14	17	10	0.909901
	1	3	6	2	914	0	10	2	4	40	0.930754
	10	3	1	41	11	761	18	8	30	9	0.853139
	9	4	7	2	4	20	907	1	4	0	0.946764
	2	9	21	5	8	2	1	952	2	26	0.92607
	13	13	8	20	14	33	7	10	844	12	0.86653

	8	8	1	12	32	11	1	24	12	900	0.891972
RECALL	0.946798	0.957904	0.932049	0.898338	0.91675	0.887981	0.934089	0.926972	0.881923	0.897308	

From the train, validation and test accuracies above, we see that the validation and test accuracies are almost the same. The train accuracy is slightly higher than the test accuracy, which means train error is slightly lower than the test error. This would mainly be because of model overfitting. The model would have over fit on the training data and hence would not have generalized well and hence its performance on test data would have reduced a little.

PROBLEM 1.2

In this problem, we need to implement a multi class Logistic Regression, but this time instead of using multiple binary classifications, we build 1 classifier with 10 classes.

The functions mlrObjFunction and mlrPredict have similar definitions to the function in previous problem, the only difference, instead of computing the sigmoid, we compute the softmax function.

Following are the accuracy values:

Training Set Accuracy	93.45%
Validation Set Accuracy	92.47%
Test Set Accuracy	92.55%

To look at the error for each category, we look at a confusion matrix and the precision and recall for each class.

	TRAINING SET										**PRECISION**
	4786	1	12	7	11	33	30	7	32	4	0.972171
	1	5592	26	17	6	19	2	13	58	8	0.973877
	23	45	4503	72	58	24	59	53	108	13	0.908229
	14	18	95	4654	4	148	15	39	105	39	0.907036
	8	20	21	7	4576	6	42	13	24	125	0.945064
	39	13	36	117	34	3963	68	18	102	31	0.896404

	23	11	29	1	24	52	4758	2	16	2	0.967466
	8	16	49	18	34	9	4	4989	14	124	0.947578
	22	75	51	103	16	113	23	16	4387	45	0.90435
	17	18	9	55	126	30	2	134	42	4516	0.912508
RECALL	0.96863	0.962644	0.932105	0.921402	0.935979	0.901296	0.951029	0.944171	0.897504	0.920318	

	TEST SET										**PRECISION**
	960	0	0	3	0	6	6	4	1	0	0.979592
	0	1110	3	2	0	2	4	2	12	0	0.977974
	6	8	924	16	10	3	14	8	39	4	0.895349
	4	1	20	914	0	25	3	10	26	7	0.90495
	1	1	6	2	921	0	9	4	9	29	0.937882
	10	2	2	37	10	773	15	6	30	7	0.866592
	9	3	4	2	7	15	914	3	1	0	0.954071
	1	9	19	6	6	2	0	952	2	31	0.92607
	9	8	6	26	9	23	10	8	868	7	0.89117
	11	8	0	10	28	5	0	20	8	919	0.910803
RECALL	0.949555	0.965217	0.939024	0.897839	0.929364	0.905152	0.937436	0.936087	0.871486	0.915339	

From the train, validation and test accuracies above, we see that the validation and test accuracies are almost the same. The train accuracy is slightly higher than the test accuracy, which means train error is slightly lower than the test error. This would mainly be because of model overfitting. The model would have over fit on the training data and hence would not have generalized well and hence its performance on test data would have reduced a little.

Compare the performance difference between multi-class strategy with one-vs-all strategy:

Looking at the accuracies from both the methods we see that multi-class strategy has higher accuracies. Multi-class uses only a single classifier, as compared to binary classifiers that uses n number of binary classifiers and hence its performance would vary with the number of binary classifiers that we would use. Hence a single classifier that used in a Multi class approach gives higher accuracies.

Accuracy is sometimes misleading, Precision and Recall helps in understanding how the model is performing on individual classes. Here we see that precision and recall values of Multi class classifier are greater than the binary classifiers.

PROBLEM 1.3:

In this problem, we need to perform classification on our dataset using Support Vector Machine.

Initially, we randomly sample 10000 samples to train the SVM model.

Now, we calculate the accuracy of the prediction with respect to training data, validation data and test data for different models.

SVM with Linear Kernel:

Training set Accuracy:	99.71%
Validation set Accuracy:	91.49000000000001%
Testing set Accuracy:	85.99%

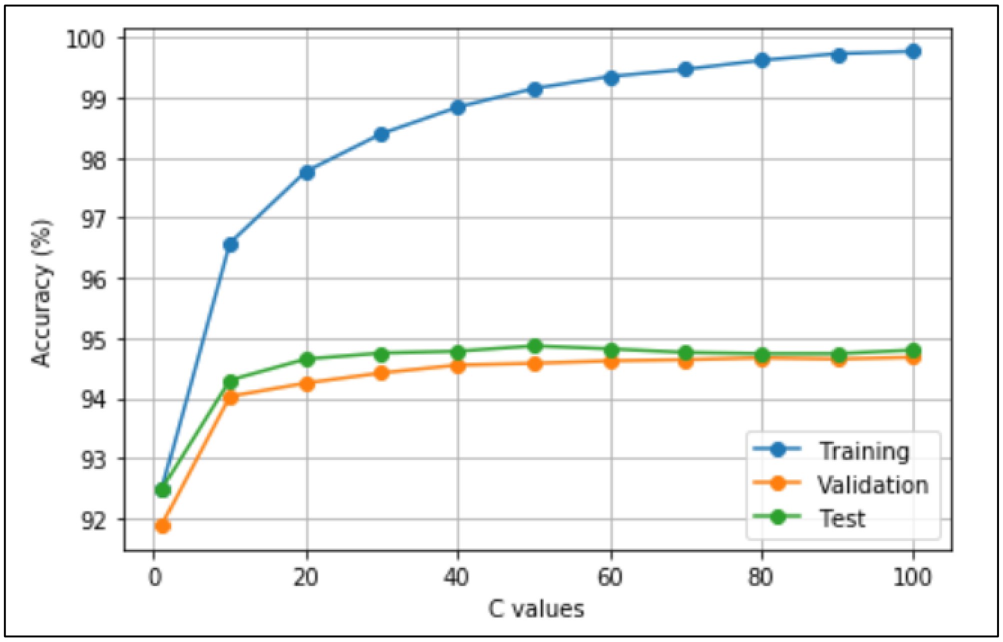
SVM with Radial Basis Function with gamma = 1:

Training set Accuracy:	100.00%
Validation set Accuracy:	15.52%
Testing set Accuracy:	17.29%

SVM with Radial Basis Function with gamma = 0:

Training set Accuracy:	92.89%
Validation set Accuracy:	92.00%
Testing set Accuracy:	92.43%

SVM with Radial Basis Function with varying values of C:



Now, looking at the accuracies of all the SVM models, model with C value = 10 has the highest test data accuracy hence we select the model where the C value is 10.

At C=10, the difference between train and test accuracies is the minimum. Also, beyond this point, though the training accuracy increases there is not much increase in validation and test accuracies. Thus the model is then over fitting.

Now, using this model we train using the entire dataset.

Training set Accuracy:	97.13199999999999%
Validation set Accuracy:	96.17999999999999%
Testing set Accuracy:	96.1%