CSE 574 INTRODCUTION TO MACHINE LEARNING

PROGRAMMING ASSIGNMENT 3

CLASSIFICATION AND REGRESSION

TEAM MEMBERS

KEERTHANA KANNAN (50248873)

HIREN ARAVIND MURUGUDU GOVINDARAJAN (50248817)

PROBLEM 1: IMPLEMENTATION OF LOGISTIC REGRESSION

The given task was to implement logistic regression to classify handwritten digit images into their correct labels. The dataset consists of images from 0 to 9 and hence can have 10 possible values. But a binomial logistic regression classifier cannot directly classify multiple classes. So we built 10 binary classifiers to classify the digits.

Two functions were implemented. The first one being *blrObjFunction()*, in which the error and error_grad (gradient of the error function) were executed.

The error was given by,

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

And, the gradient of the error was given by,

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

In the second function *blrPredict()*, the posterior probability was computed using 2 name expressions (one for each class):

$$P(y = C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$$

and,

$$P(y = C_2|\mathbf{x}) = 1 - P(y = C_1|\mathbf{x})$$

The decision rule was used to assign 'x' to C_k that maximizes the posterior probability.

$$y_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \in C_1 \\ 0 & \text{if } \mathbf{x}_i \in C_2 \end{cases}$$

By doing this, the following results were obtained:

Training set Accuracy	90.744%
Validation set Accuracy	89.49%
Testing set Accuracy	89.96%

Total error in the training data	9.256%
Total error in the test data	10.04%

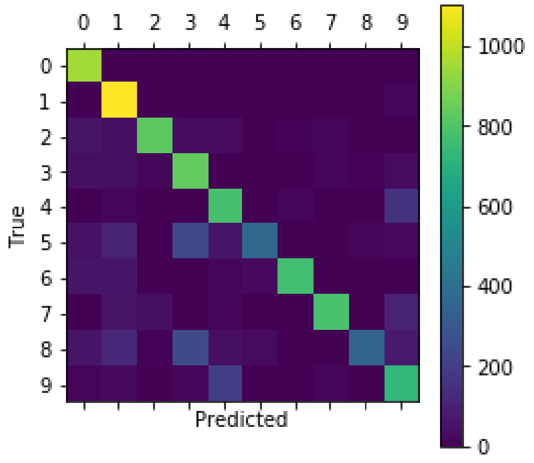
The confusion matrix for each class can be given as,

- Train data
  The values are mentioned in percentage.
  [[65 0 0 0 0 0 0 0 23 0]

```
[ 0 69  0  0  0  0  0  0 22  0]
[ 0  0 63  0  0  0  0  0 23  0]
[ 0  0  0 57  0  0  0  0 34  0]
[ 0  0  0  0 59  0  0  0 33  0]
[ 0  0  0  0  0 75  0  0 16  0]
[ 0  0  0  0  0  0 67  0 24  0]
[ 0  0  0  0  0  0  0 48 41  0]
[ 0  0  0  0  0  0  0  0 88  0]
[ 0  0  0  0  0  0  0  0 28 64]]
```

- Test data
  The values are mentioned as the number of predictions.

```
[[ 955    6    0   177    2    1    2    1    2    27]
 [    0 1207    0    8    0    0    3    0    0    17]
 [ 58    46 926   35   34    0   11   114    7    1]
 [ 44    40   17 842    4    2    1   18   10   32]
 [    2   16    2    0 987    0   13    2    0 160]
 [ 53 106    3 242   63 676    2    7   16   24]
 [ 66   63    6    2   18   29 872    0    0    2]
 [    3   60   46    4   19    0    0 888    2 106]
 [ 64 123   11 251   46   32    2    6 658   81]
 [ 14   25    3   15 205    1    0   14    1 831]]
```



We can see that the training error is less than that of test error. This is due to the fact that logistic regression considers all the points within a dataset to create a hyperplane that separates the points into their correct classes. Hence, it works well when there is less number of inputs. One of the possible reasons is that test set is larger than the training set, thus resulting in overfit.

PROBLEM 2: MUTI-CLASS LOGISTIC REGRESSION

In this, we build only one classifier that can classify all the 10 classes at the same time. Similarly, 2 functions were implemented. The first one is *mlrObjfunction()*, in which the error and error_grad (gradient of the error function) were executed.

The error was given by:

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\ln P(\mathbf{Y}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk} \ln \theta_{nk}$$

and, the error gradient was given by:

$$\frac{\partial E(\mathbf{w}_1, \cdots, \mathbf{w}_K)}{\partial \mathbf{w}_k} = \sum_{n=1}^{N}(\theta_{nk} - y_{nk})\mathbf{x}_n$$

Similarly, the posterior probabilities were given by softmax transformation of linear functions of the feature variables.

$$P(y = C_k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T\mathbf{x})}$$

The results were obtained as

Training set Accuracy	93.138%
Validation set Accuracy	92.54%
Testing set Accuracy	92.53%

Total error in the training data	6.862%
Total error in the test data	7.47%

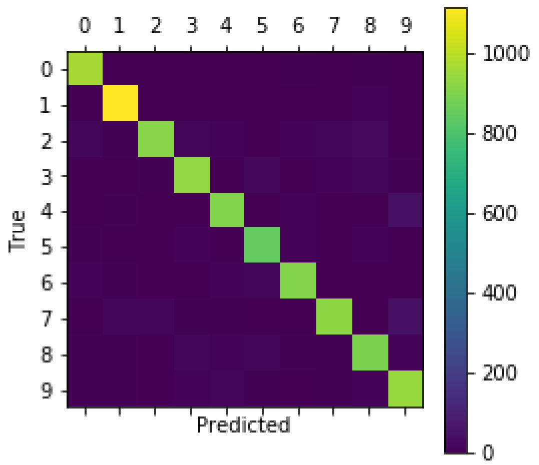
The confusion matrix for each class can be given as,

- Train data
  The values are mentioned in percentage.
```
[[86  0  0  0  0  0  1  0  0  0]
 [ 0 82  0  0  0  0  0  0  3  6]
 [ 1  0 85  0  0  0  0  0  0  0]
 [ 0  0  0 78  0  4  0  1  8  0]
 [ 1  0  0  0 87  0  0  0  0  4]
 [ 0  0  0  0  0 88  1  0  0  2]
 [ 0  1  0  0  0  0 90  0  0  0]
 [ 0  1  0  0  1  2  0 84  0  0]
 [ 0  3  0  1  0  1  0  1 81  1]
 [ 1  0  0  1  0  3  0  1  0 86]]
```

- Test data

  The values are mentioned as the number of predictions.

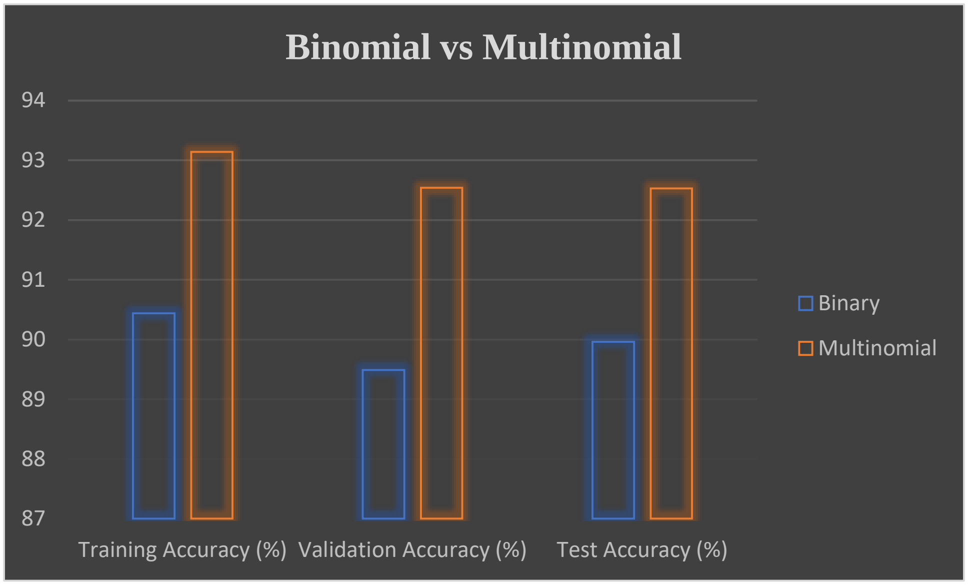
```
[[ 963    0    1    0    0    3    6    1    6    0]
 [   0 1115    1    3    0    1    4    0   11    0]
 [  14    6  916   20   13    0   11   19   30    3]
 [   3    2    5  934    0   26    1   12   19    8]
 [   1    5    3    1  904    1   12    1    4   50]
 [   7    3    0   11    1  847    9    1   10    3]
 [  12    5    2    1   11   18  906    0    3    0]
 [   1   14   21    6    5    2    0  925    2   52]
 [   7    6    2   14   10   19    5    7  893   11]
 [   8    7    2    9   15    8    0    5   10  945]]
```



Here, the training error is less than that of test error. One of the possible reasons is that test set is larger than the training set, thus resulting in overfit.

Multi-Class Vs One-Vs-All Strategies

Regression Type	Training Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
Binary	90.744	89.49	89.96
Multinomial	93.138	92.54	92.53



It is seen that the multinomial regression results in better accuracy than binomial regression. In the binary strategy for an unseen instance, a class with maximum confidence is chosen. Whereas, multinomial classifier learns all the classes directly. Therefore, the parameters of each class are estimated interdependently and the model is more robust against the outliers. This gives a better decision boundary than the binary strategy.

PROBLEM 3: SUPPORT VECTOR MACHINES

We used the Support Vector Machine tool in sklearn.svm.SVM to classify our dataset.

	Training set accuracy (%)	Validation set accuracy (%)	Test set accuracy (%)
Linear kernel	97.286	93.64	93.78
Radial basis Gamma=1	100	15.48	17.14
Radial basis gamma='auto'	94.294	94.02	94.42

In the above table, the accuracies of SVM based models are greater than the logistic regression based models. This is because, SVM learns about a hyperplane that has maximum margin, whereas logistic regression learns about any hyperplane that is capable of separating data. This difference induces a performance difference between the two.

Considering the number of input features, SVM is better as it works with more number of inputs while logistic regression deals with fewer input features.

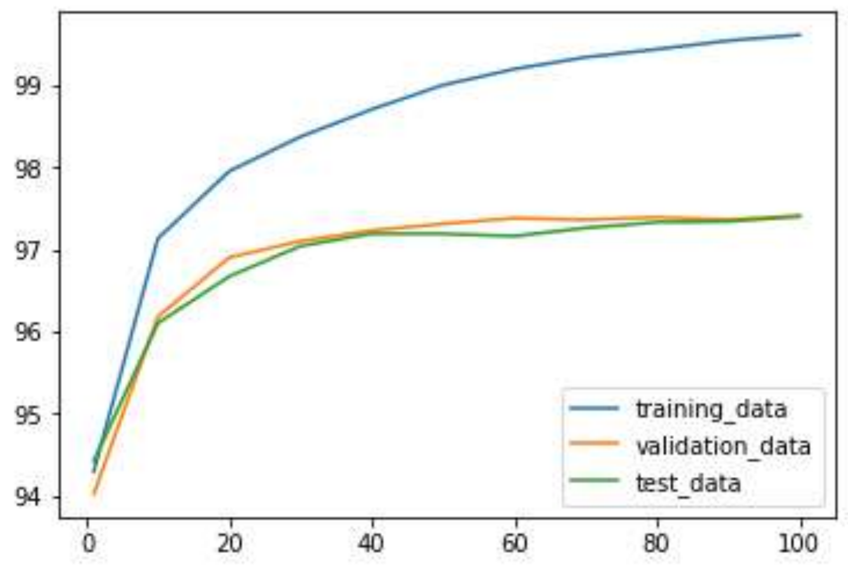
Generally, logistic regression is effective when there is no indirect information about the output. But in this case, SVM is a better choice because the inputs are pixels and they are able to determine the response directly.

Working with SVM results in high accuracy when using with a Gaussian kernel. This is because, it can map the inputs into a dimensional space that is infinite. It is observed in Table 1 that accuracy is low on both test data and validation data and overfitting is also observed. If gamma value is chosen closely to 0, then an optimal result can be expected and validation and test data, despite achieving maximum accuracy in the training data.

It is also observed that lower value of parameter C results in a large marginal hyperplane and vice-versa. Thus, the errors in the training samples are impacted by C value.

Radial Basis Function with Varying ‘C’ Values:

C	**Training set accuracy (%)**	**Validation set accuracy (%)**	**Test set accuracy (%)**
1	94.294	94.02	94.42
10	97.132	96.18	96.1
20	97.952	96.9	96.67
30	98.372	97.1	97.04
40	98.706	97.23	97.19
50	99.002	97.31	97.19
60	99.196	97.38	97.16
70	99.34	97.36	97.26
80	99.438	97.39	97.33
90	99.542	97.36	97.34
100	99.612	97.41	97.4



The above graph is a plot between training, test and validation data based on different values of C. It is observed that, for high values of parameter C, the accuracy is also high. Thus a larger value of C is usually taken for the training phase. As mentioned earlier, this also results in a hyperplane with a large margin at the cost of samples getting misclassified.

On the other hand, when the value of C is increased, values with low errors are accepted as weight of the error terms tend to increase. This results in generating hyperplane with smaller margin but this time the number of misclassified samples are relatively less. Thus, there is increase in accuracy.

Overfitting can result while choosing the values of C which is not apparent in the plot as accuracy is increasing for all the data sets. Thus increasing the complexity of the hyperplane can end in overfitting.

CONCLUSION

The accuracy of the SVM model is greater than that of logistic regression models for which the reason has also been discussed. The error for each class in the binomial and multinomial logistic regression was portrayed in a confusion matrix.