CSE474/574 - Introduction to Machine Learning Programming Assignment 3 Classification and Regression

TEAM MEMBERS:

SMUNAGAL@BUFFALO.EDU UB ID# 50168800

SHRAVYAT@BUFFALO.EDU UB ID# 50169587

CSE 574 GROUP # 9

INDEX.

Logistic Regression:

Observations:

In this Programming assignment, data given to us is a set of Images of Hand written digits which has to be classified into 10 classes. Since we cannot use a single Binary Logistic Regression Classifier for the given data, we have implemented the One-vs-All Strategy for the Logistic Regression. The classifier build learns 10 Binary classifiers (one for each of the output classes) so that each class can be distinguished from all other classes.

The function "blrObjFunction" learns a weight vector for every class i.e., it takes in the data set with the true labels and then tries to learn the weight vector so that the error shown as below is reduced. In this task for learning the weight vector for a particular class we use a Binary Logistic Regression where this class is considered as one class and rest of the classes are considered as one class.

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

$$\text{where } \theta_n = \sigma(\mathbf{w}^T\mathbf{x}_n)$$

For the task of minimizing the error function and to find the "w" which can gives us the best minimized error, we use the Gradient Descent function i.e., the "minimize" function, which takes the initial weights, the error value and the derivate of error function as inputs and gives a "w" which gives the best minimized error. The derivative or the gradient of the error function with respect to "w" is as below.

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

The function "blrObjFunction" is executed for 10 times since we have 10 output classes and it learns a weight vector for every class specifically. We then use this matrix of "w" which contains 10 weight vectors of 10 classes to do the prediction. For the task of prediction we have implemented the function "blrPredict". The formula used for prediction is as below.

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

The probability of the input data point belonging to one class is calculated for each individual class and the class with highest probability is taken as the label for the data point i.e., we need to compute the posterior probability $P(y = C_k|x)$ and the decision rule is to assign x to class C_k that maximizes $P(y = C_k|x)$.

The results obtained are as below: -

Data	Accuracy
Training Accuracy	86.14
Validation Accuracy	85.12
Testing Accuracy	85.38

Support Vector Machines:

Observations: A Support Vector Machine (SVM) is a discriminative classifier designed to learn a separating hyperplane i.e., given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples and also maximizes the margin of the training data. Maximizing the margin is nothing but the hyperplane is learnt in such a way that it is equally far from all the sets of data points belonging to different classes.
In the task of maximizing the margin, the classifier tries to learn a weight vector "w" such that it minimizes ||w|| since the margin is given as below.

$$M = 2/||w||$$

For the task of minimizing the "w", SVM can use different functions i.e., Kernel functions. In this programming assignment we use two of those functions: -

1. Linear Kernel - LinearSVC
2. Radial Basis Function – SVC with parameters "gamma" and "C"; where C is the Penalty parameter of the error term and gamma is the kernel co-efficient for "rbf".

The SVC is implemented using its functions SVC(), fit() and predict().
The fit() function tries to fit the SVM model according to the given training data i.e., takes in the training data and the true labels and tries to fit the data into a SVM. The predict() function performs classification on samples in training data and returns the class labels for them.
For the Linear Kernel we use the standard SVCLinear() function, and for the Radial Basis Kernel we use the standard SVC() function and defines the parameters as required.

Using linear kernel (all other parameters are kept default)The results obtained are as below: -

Data	Accuracy
Training Accuracy	92.82
Validation Accuracy	91.25
Testing Accuracy	91.68

Using radial basis function with value of gamma setting to 1 (all other parameters are kept default). The results obtained are as below: -

Data	Accuracy
Training Accuracy	100.0
Validation Accuracy	15.48
Testing Accuracy	15.76

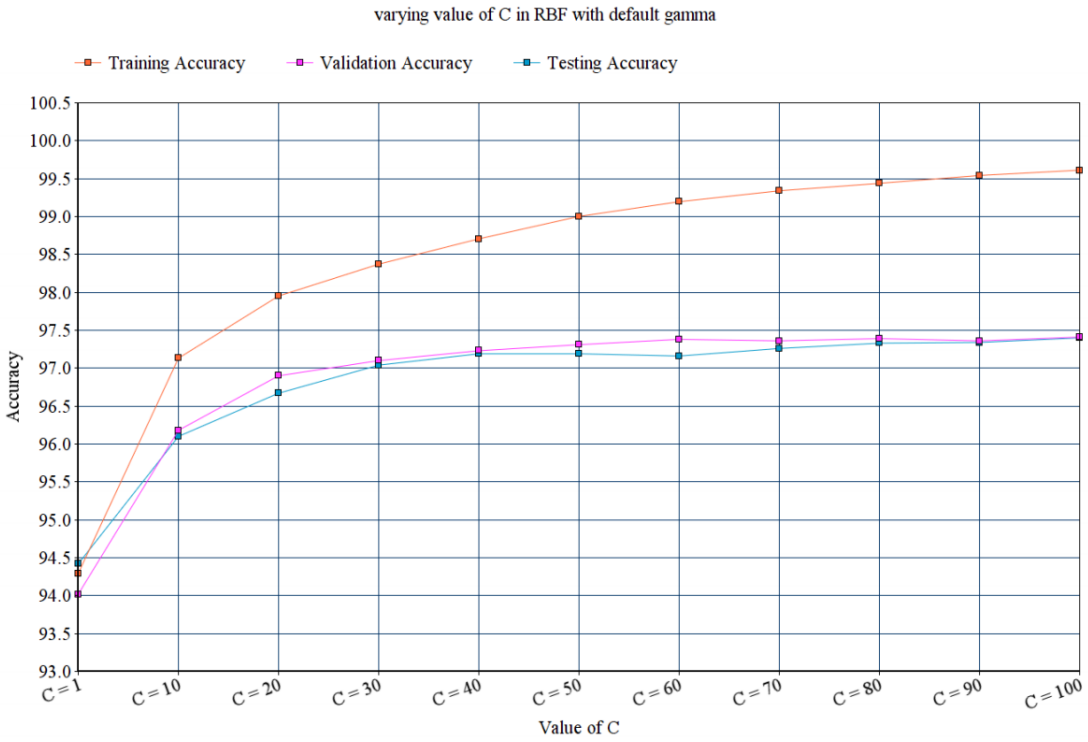
Using radial basis function with value of gamma setting to default (all other parameters are kept default). The results obtained are as below: -

Data	Accuracy
Training Accuracy	94.294
Validation Accuracy	94.02
Testing Accuracy	94.42

Using RBF gamma default and varying value of C (1, 10, 20, 30, ⋯, 100)

C Value	Training accuracy	Validation accuracy	Testing 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

Graph:



Multi-Class Logistic Regression

Observations: Instead of using the One-vs-All model and learning 10 Binary Logistic Classifies, we have implemented the Multiclass Logistic Regression an extension of the Binary Logistic Classifies which learns the weight matrix "w" with 10 different weight vectors for the 10 different classes in just one classifier and we then use it for predicting the training data.
For the multi-class Logistic Regression, the posterior probabilities are given by a "softmax transformation" of linear functions of the feature variables given as below.

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

Where "w_k" is the weight vector for each specific class. The error function, which is known as the cross-entropy error function and the gradient of the error function with respect to "w" are given as below.

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

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

For the learning task, we do the same thing as we have done in the case of the Binary Logistic Regression i.e., we use Gradient Descent method for learning a weight matrix of 10 weight vectors each for each of the 10 classes such that it gives us a minimum error.

The function mlrPredict() is implemented to learn the weight vectors for the 10 different classes. The function calculates the Error value and its gradient and passes to the "minimize" function which in turn updates the weight vector as below and sends it back to the mlrPredict() function.

$$\mathbf{w}_k^{new} \leftarrow \mathbf{w}_k^{old} - \eta\frac{\partial E(\mathbf{w}_1, \cdots, \mathbf{w}_K)}{\partial \mathbf{w}_k}$$

For the prediction, we have implemented the function "mlrPredict" which does the prediction similar to that what we have done in the function "blrPredict" i.e., calculate the posterior probabilities of the data point with respect to each class and takes the class which gives the highest probability and label it as the class for that data point. The posterior probability of a class given a data point is show in the above equations.

The results obtained are as below: -

Data	Accuracy
Training Accuracy	93.39
Validation Accuracy	92.43
Testing Accuracy	92.67