CSE574: Introduction to Machine Learning
Programming Assignment 3

Handwritten Digits Classification using Logistic Regression and Support Vector Machine

Submitted by
Vidyadhar Reddy Annapureddy

The programming assignment extends the first programming assignment in classifying handwritten digits classification by implementing Logistic Regression and Support Vector Machine to classify hand-written digit images. We then compare the performance of these methods based on the accuracy on training, testing and validation datasets. For simplicity, the implementation uses the tool **sklearn.svm.SVM** for classification.

Logistic Regression

We build a multinomial logistic regression model using gradient descent to build 10 binary-classifiers, one for each class of digits 0-9. It generalizes logistic regression to multiclass problems and subsequently predicts the probabilities of the different possible outcomes of a categorically distributed dependent variable, given a set of independent variables (validation dataset).

The implementation can be broken down into 3 discrete steps:

1. *Preprocess*(): This preprocessing function is responsible for division of the original data set into training, validation and testing set with corresponding labels, conversion of original data set from integer to double by using *double(),*normalization of the data to [0, 1] and finally, feature selection. The pixels that are determined to be not useful for classification of digits are ignored for faster computations.

2. *blrobjFunction*(): This function takes in as input the weight vector, the data matrix, the binary label vector and returns a scalar value of error function of 2-class logistic regression and vector representing the gradient of error function. It computes 2-class Logistic Regression error function and its gradient.

3. *blrPredict*(): This function predicts the label of data given the data and parameter W of Logistic Regression by taking as input the weight matrix, the data matrix and returns a vector representing the predicted label of corresponding feature vector given in data matrix.

Logistic Regression	
Data	Accuracy (in %)
Training	92.328
Validation	91.46
Test	91.92

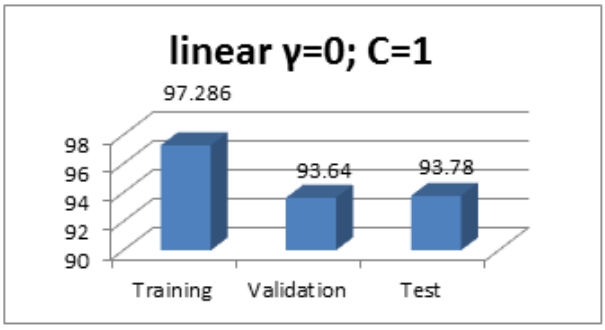
Support Vector Machine (SVM):

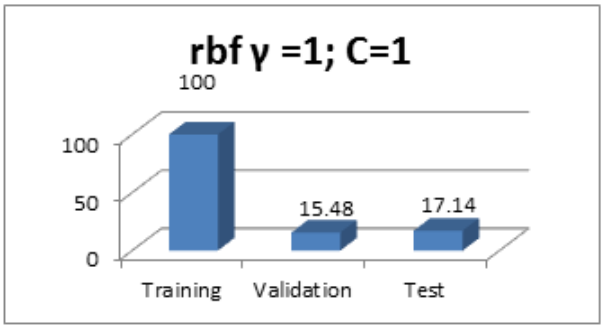
SVM constructs a hyper-plane in a high-dimensional space, used for classification or regression. A good separation is achieved for a hyper plane that has the largest distance to the nearest training data point of any class; since larger the margin, the lower the generalization error of the classifier. The implementation learns the SVM model and makes predictions with respective accuracies for training, validation and test data based on following parameters:

a. Using linear kernel (all other parameters are kept default)
b. Using radial basis function with gamma set to 1 and other parameters set to default.
c. Using radial basis function with all parameters set to default values.
d. Using radial basis function with gamma set to default and value of C varying uniformly from1 to 100.

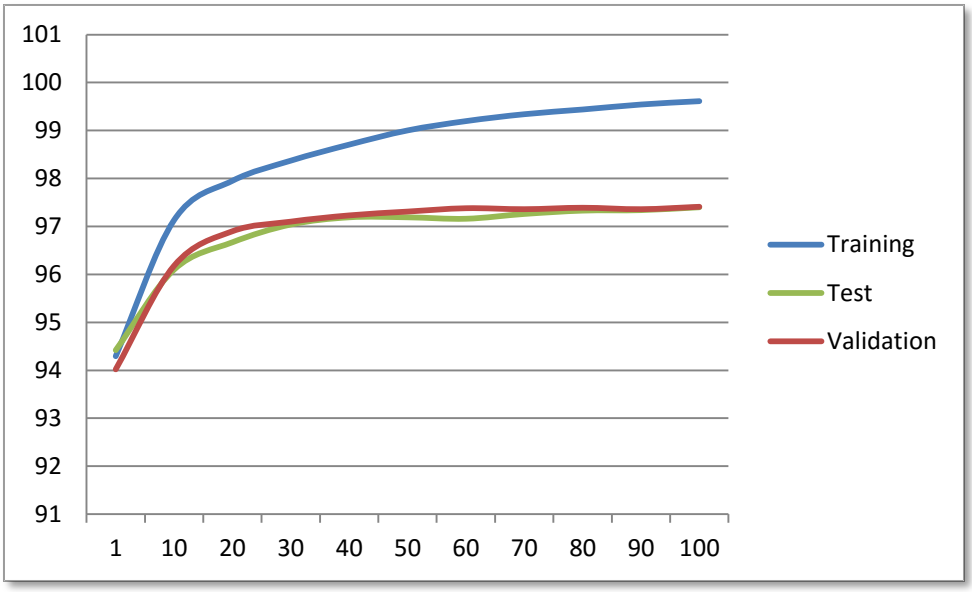
The accuracies for above analysis were observed as follows:

Accuracy (in %)	rbf	linear
	γ =1; C=1	γ=0; C=1
Training	100	97.286
Validation	15.48	93.64
Test	17.14	93.78





	rbf											
	γ =1; C=1	γ=0; C=1	γ=0; C=10	γ=0; C=20	γ=0; C=30	γ=0; C=40	γ=0; C=50	γ=0; C=60	γ=0; C=70	γ=0; C=80	γ=0; C=90	γ=0; C=100
Training	100	94.29	97.13	97.95	98.37	98.706	99.00	99.19	99.34	99.43	99.54	99.61
Validation	15.48	94.02	96.18	96.9	97.1	97.23	97.31	97.38	97.36	97.39	97.36	97.41
Test	17.14	94.42	96.1	96.67	97.04	97.19	97.19	97.16	97.26	97.33	97.34	97.4



Conclusion:

In Support Vector Machines, C is essentially a regularization parameter, which controls the trade-off between achieving a low error on the training data and minimizing the norm of the weights. The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyper plane if that hyper plane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyper plane, even if that hyper plane misclassifies more points. For low values of C, we expect to get misclassified examples, which coincide with the results.