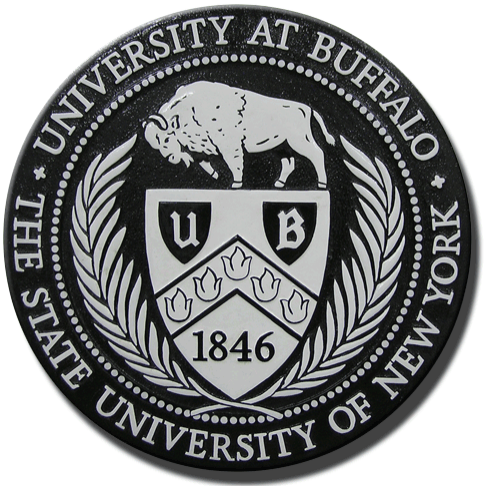
UNIVERSITY AT BUFFALO

The State University of New York



Introduction to Machine Learning

(CSE 574)

Report of Programming Assignment 3

Group #26, Members:

Ting Zhou

Xuan Han

May 3, 2017

Part 1 Logistic Regression

Training set Accuracy: 84.9%
Validation set Accuracy: 83.65%
Testing set Accuracy: 84.15%
For binary logistic regression, each time the algorithm optimizes the weights of one class. For example, after the first time of the optimization we can get the optimized weights in order to classify the digit "0". Then we assemble the weights that can identify "0" into the first column of the weight matrix. Following the same procedures, we can get the ten weight columns with respect to the ten digits. However, we need to do the optimization ten times in order to classify the ten digits, which takes much time. Furthermore, each time the gradient descent algorithm only finds the minimum point for one class. Thus the minimum point is not the global minimum for the whole map of the ten classes. That's why the classification accuracy is not ideal.

Part 2 Multi-class Logistic Regression

Training set Accuracy: 93.09%
Validation set Accuracy: 92.43%
Testing set Accuracy: 92.55%
Following the discussion in Part 1, multi-class logistic regression improves the performance of the binary logistic regression not only for the accuracy but also for the run time. The art behind the multi-class logistic regression is very easy. Firstly, the cost function can include all of the weights of the ten classes by using the softmax function. Thus this time we can find the global minimum of the whole map of the ten classes. That's why the accuracies are increased. Secondly, this time we only need to do the optimization one time for the ten classes. Thus the run time of this algorithm is only about one tenth of the run time of the binary logistic regression.

Part 3 Support Vector Machines

Linear Kernel (all other parameters are kept default)
Training set Accuracy: 97.286%
Validation set Accuracy: 93.64%
Testing set Accuracy: 93.78%
For linear kernel,

$$K(\boldsymbol{x}, \boldsymbol{x}') = < \boldsymbol{x}, \boldsymbol{x}' >$$

The feature space is still the input space. The accuracies are pretty good, which means the classes of the data are almost linear separable. There is no need to use the SVM in a higher order feature space to complete the classification.

RBF with value of gamma setting to 1 (all other parameters are kept default)
Training set Accuracy: 100.0%
Validation set Accuracy: 15.48%
Testing set Accuracy: 17.14%

For Radial Basis Function

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma|\boldsymbol{x} - \boldsymbol{x}'|^2\right)$$

The order of the feature space of rbf kernel could be infinite. The behavior of the model is very sensitive to the gamma parameter. From the results of this case, the gamma is too large. The radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting. Thus the training accuracy is 100%, but the prediction is very poor.

RBF with value of gamma setting to default (all other parameters are kept default)

Training set Accuracy: 94.294%

Validation set Accuracy: 94.02%

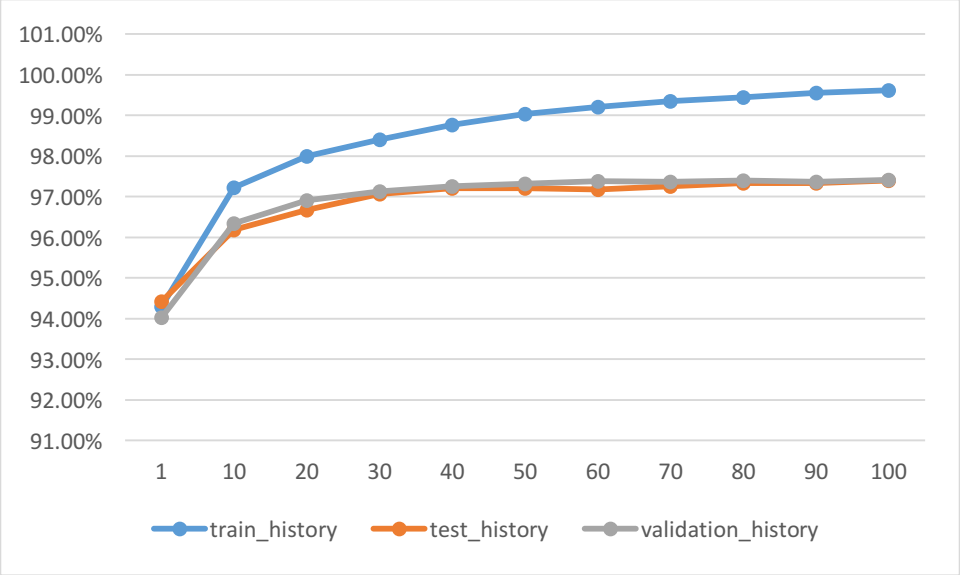
Testing set Accuracy: 94.42%

Following the last case, this time the gamma is not too large. Then the overfiitting disappears.

RBF with value of gamma setting to default and varying value of C (1; 10; 20; 30; …; 100)

The results of this case are shown in the table and the graph. The C parameter trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors [1]. Thus with the incease of C, the accuracy of training data is closer to 100%, but the prediction doesn't improve when C is higher than 30.

C	1	10	20	30	40	50	60	70	80	90	100
train_history	94.29%	97.22%	98.00%	98.40%	98.76%	99.03%	99.21%	99.35%	99.44%	99.56%	99.61%
test_history	94.42%	96.18%	96.67%	97.07%	97.20%	97.20%	97.17%	97.26%	97.33%	97.33%	97.40%
validation_history	94.02%	96.34%	96.91%	97.12%	97.25%	97.31%	97.38%	97.36%	97.40%	97.36%	97.41%



Reference:
[1] http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html