CSE474 Machine Learning
Assignment 3 Report
Joseph Cascioli
Jobin Joseph
Benjamin Riexinger
Group 11

Logistic Regression

Logical Regression is a Discriminative and Probabilistic Classifier. Thus P(y|x) is modeled directly making the model learn faster. The classifier learns Linear Boundary, making it also a linear classifier. With multi-class, we can combine all the classes into one classifier that will sort each example based on its particular class. We can see this method will produce better results. Our accuracies went up by about 10% in each category. The multi-class also runs faster because it is only one classifier that is being run, instead of many as is the case with Binary.

Logistic Regression			
	Training set Accuracy	**Validation set Accuracy**	**Testing set Accuracy**
Binary	84.88%	83.67%	84.14%
Multi-Class	93.45%	92.48%	92.55%

Support Vector Machines

We use a hyperplane-based classifier when implementing Support Vectors. The hyperplane maximizes the distance from it and the nearest data point.

Linear kernel (default)
Training set Accuracy: 97.286%
Validation set Accuracy: 93.64%
Testing set Accuracy: 93.78%

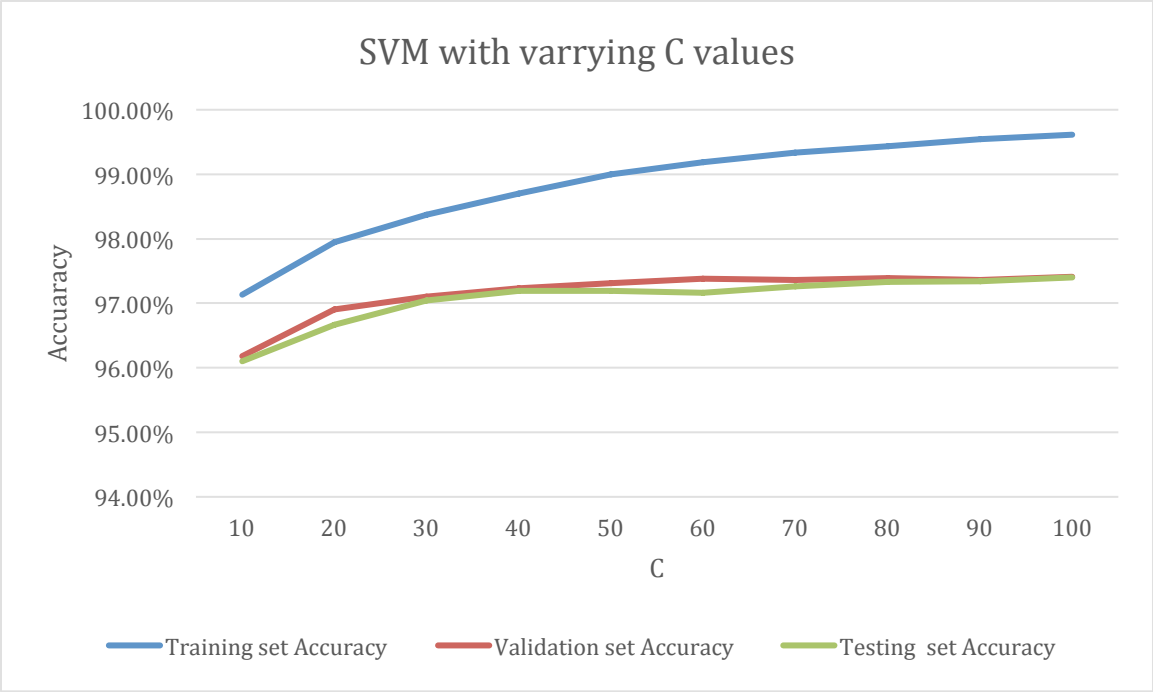
RBF kernel (Gamma = 1)
Training set Accuracy: 100.0%
Validation set Accuracy: 15.48%
Testing set Accuracy: 17.14%

RBF Kernel (Gamma = 0)
Training set Accuracy: 94.294%
Validation set Accuracy: 94.02%
Testing set Accuracy: 94.42%

RBF Kernel (Gamma = 0 C = 1)
Training set Accuracy: 94.294%
Validation set Accuracy: 94.02%
Testing set Accuracy: 94.42%

With default settings, we can already see a significant increase in accuracy with the SVM over Logistic Regression. Next we can see how the RBF Kernel will react with different parameter changes. With Gamma set equal to 1 we get a false positive on the potential accuracy when looking at the Training set Accuracy, being 100.0%. This false positive is due to overfitting caused by a high regularization parameter of the Gaussian radial basis function. Next, we ran Gamma set equal to 0 and here we received great accuracies and more importantly, over Gamma equal to 1, we get consistent results, due to a more correctly fit hyperplane. Finally we set our control run as all defaults, Gamma equal to 0 and C equal to 1. This yielded overall great results. Now we can successfully test the changes in performance when modifying the C value.

RBF Kernel (Gamma = 0)			
C	**Training set Accuracy**	**Validation set Accuracy**	**Testing set Accuracy**
10	97.13%	96.18%	96.10%
20	97.95%	96.90%	96.67%
30	98.37%	97.10%	97.04%
40	98.70%	97.23%	97.19%
50	99.00%	97.31%	97.19%
60	99.19%	97.38%	97.16%
70	99.34%	97.36%	97.26%
80	99.43%	97.39%	97.33%
90	99.54%	97.36%	97.34%
100	99.61%	97.41%	97.40%



As we can see from the above table, the higher the C value the better overall accuracy obtained across the board. Every increase of C gives us an increase in all categories. Diminishing returns hits quickly in terms of large increases in accuracy with each increase of C. This is all due because C is a parameter of the SVM which allows the function to either place examples on the wrong side of the hyperplane or ignore them all together. The higher the number choose for C the larger the penalty. This can have a negative impact of the hyperplane being over fitted to the Training Set. That is apparent when you look at the accuracies for all three categories and you notice the rather higher accuracy rate in comparison to the others.