DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY AT BUFFALO

CSE 574 Programming Assignment 3
Classification and Regression

Group 100
Luigi Di Tacchio Zakieh Hashemifar Alexander Simeonov
April 30, 2015

1 COMPARISON BETWEEN LOGISTIC REGRESSION AND SVM

Algorithm		Accuracy		
		Training data	Validation data	Test data
Logistic Regression		92.354%	91.39%	91.9%
SVM	Linear Kernel	97.286%	93.64%	93.78%
	RBF, gamma=1	100.0%	15.48%	17.14%
	RBF, gamma=0	94.294%	94.02%	94.42%
	RBF, C=1	94.294%	94.02%	94.42%
	RBF, C=50	99.002%	97.31%	97.19%
	RBF, C=100	99.612%	97.41%	97.4%

Table 1.1: Accuracy for logistic regression and SVM

Table 1 shows a comparison between logistic regression and SVM.

Generally speaking, logistic regression considers all the points within a dataset for finding a separating hyperplane; on the other hand, SVM just works with the samples which are close to each other and close to the margin (which are called support vectors) and tries to provide the maximum possible distance between them and the hyperplane.

As we can see from table 1, logistic regression performs worse than SVM on any data set.

This is due to the following reasons:

- In general, logistic regression learns any hyperplane that separates the data, while SVM learns the best one (with the maximum margin): this justifies the performance difference in the first instance.
- Logistic regression works well with a low number of input features, while SVM performs better when the number of input dimensions is high, like in this case (716 pixels).
- Logistic regression works well when the input data does not give direct information on the output. In this case, the predictors are pixels and directly determine the response. Therefore, SVM performs better in this case.

SVM gives higher accuracy when using a Gaussian kernel, that maps the input features into an infinite dimensional space. In most cases, a non-linear kernel works better than a linear one, despite a longer training time.

Gamma controls the influence of each training example on the learned hyperplane: table 1 shows that when gamma is too high, we have overfitting and very low accuracy on both validation and test set, despite getting 100% of accuracy on the training set. When gamma is 0 (actually close to 0 - SVC sets it to 1 / number of features), we have much better results on validation and test set.

The parameter C determines the impact of the error on the training examples, and as a consequence, controls the complexity of the learned curve: a lower C gives a smoother curve, while higher C value give a more complex boundary. More details on the effect of C will be given in section 2.

2 Accuracy of SVM with Gaussian kernel for different values of C

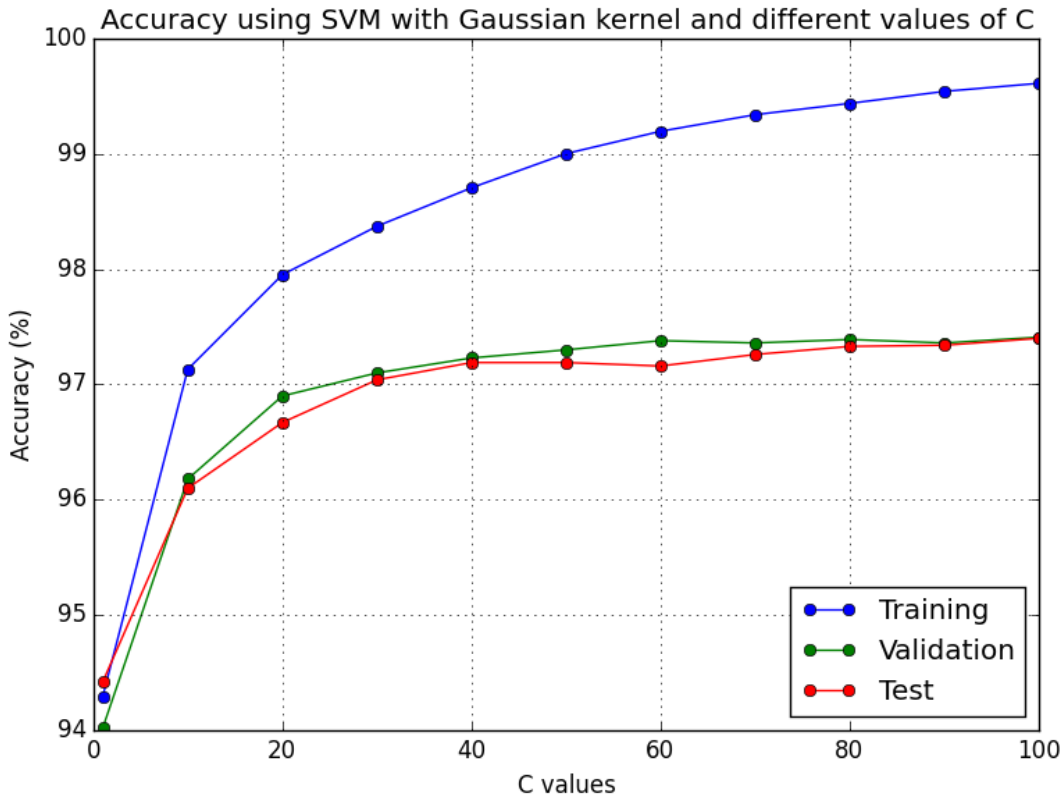

Figure 2 plots the accuracy on training, validation and test data for different values of C, using SVM and a Gaussian kernel.

As it can be seen from the plot, we have a higher accuracy for higher values of C. That is because C controls the penalty for the error term on each training example.

When C is low, the weight of each error term is low as well: therefore, a larger error value is accepted during the training phase. Consequently, a larger margin hyperplane is created in the expense of having more samples misclassified.

Conversely, when C increases, the weight of each error term increases as well, so lower error values will be accepted. As a result, a smaller margin hyperplane will be built, but less number of points will be misclassified, so the accuracy of data classification increases.

Based on this explanation, there is a high risk of overfitting for larger values of C. It is not evident from the plot, that shows increasing accuracy for all 3 data sets. However, we can see how the accuracy on the training set increases considerably when C grows, while the accuracy on validation and test set saturates pretty fast (from 40 to 60 it actually decreases for the test set) and this is a clear sign that increasing the complexity of the curve may give overfitting.