Assignment 3: Classification and Regression

CSE 574 Introduction to Machine Learning

Isabela Lago, Nathan Margaglio, Timothy Schuler

August 16th, 2018

1 Logistic Regression

1.1 Binary Logistic Regression

For the first part of this assignment, we implement Logistic Regression to classify hand-written digit images into correct corresponding labels. Similar to the first assignment, we have a dataset consisting of a number of 28x28, black and white images of hand-written digits (from 0 through 9) and it's corresponding label as a number.

The task has us implement a binary classifier, which cannot classify our digits (since there are 10 of them). In order to accommadate for this, we build 10 classifiers, each of which corresponding to a single digit. Each classifier attempts to determine if the image at hand is it's corresponding digit or not. We then assume the classifier with the greatest positive certainty is the correct label (a *one-vs-all* strategy).

In order to accomplish this task, we write the functions *blrObjFunction()* and *blrPredict()*, which performs the training and predicting, respectfully. The following are the results of training the classifiers and testing their accuracy (percentage of correct classifications) on the training, validation, and testing data sets.

```
Training Set Accuracy   = 92.72%
Validation Set Accuracy = 91.46%
Testing Set Accuracy    = 92.00%
```

We see that the results are relatively good (compared to the same accuracies from the first project's neural network). The fact that the training set accuracy is fairly close to the validation set accuracy and the testing set accuracy indicates that overfitting is not a problem for this strategy, and implementing a binary logistic classifier fairs well for this kind of task.

1.2 Multi-Class Logistic Regression

For the second part of this project, we use the Support Vector Machine tool from *sklearn.svm.SMC* (http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html) to perform classification on our data set. We experiment with different sets of parameters for the function in order to compare the different accuracies.

First, we vary the kernels. We test the same data sets from the previous section using a linear kernel with all other default parameters, a radial basis with all other default parameters, and a radial basis with value of γ set to 1 and everything else default.

The linear kernel took about 10 minutes to train on my machine, while the radial kernels took almost 20 minutes each to train, meaning that, at least in the case, the training time for the linear kernel is twice as fast. The linear kernel, however, can't produce as complex a model as the radial kernels, meaning that the radial kernels have a better chance of finding a model that better fits the data. The value of γ, which, we will test at both $1/[n\_features]$ (the default) and 1, will provide a trade-off of bias and variance.

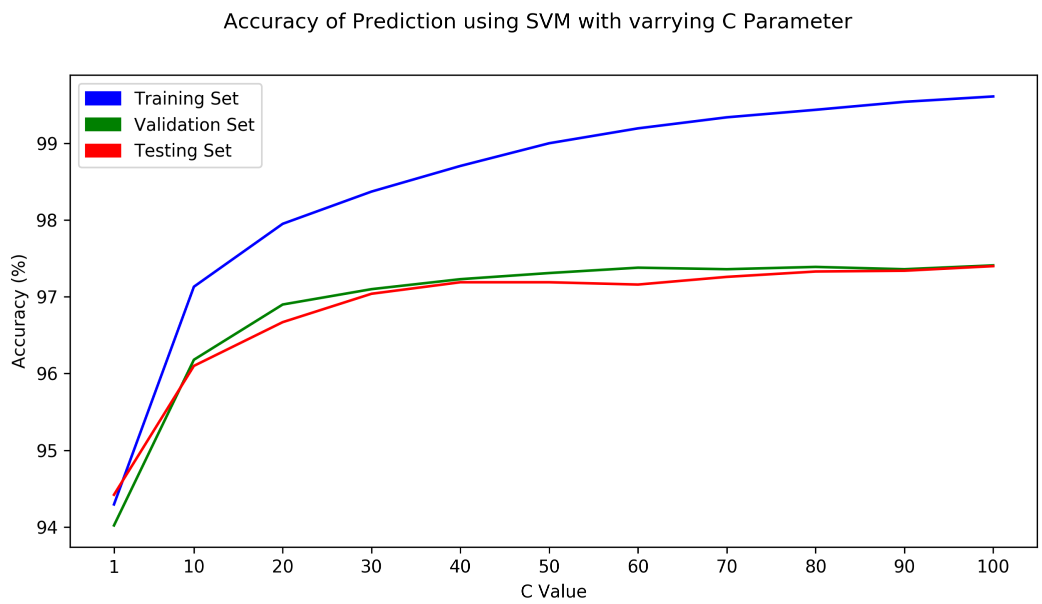
The results for the tests are as follows:

```
Linear Kernel (default parameters):
Training Set Accuracy   = 97.28%
Validation Set Accuracy = 93.64%
Testing Set Accuracy    = 93.78%

Radial Kernel (default parameters):
Training Set Accuracy   = 94.29%
Validation Set Accuracy = 94.02%
Testing Set Accuracy    = 94.42%

Radial Kernel (Gamma = 1):
Training Set Accuracy   = 100.00%
Validation Set Accuracy = 15.48%
Testing Set Accuracy    = 17.14%
```

The results show that, among the three variations, the radial kernel with default paramters perform the best with the validation and testing sets. Both the linear kernel and the radial with default kernel performed better than the binary logistic classifier on every set. The radial kernel with $\gamma = 1$ had a training set accuracy of 100%, highest of all the models and indicating extreme overfitting. It's validation accuracy being 15.48% and it's testing accuracy being 17.14% verify this claim.

Next, we test the accuracy of a SVM with a radial kernel while varying the C parameter. We perform the same tests as above while varying C through the values $1, 10, 20, 30, ..., 100]$ for 11 tests of each set (33 total). The value of C will determine how stable the model is by adjusting the margin for close examples. A low C means a narrow margin and less error, but at the cost of stability versus a higher C.

Below are the plots for the accuracies of the tests:



We immedietly see that a higher C performs better in this situation, with the highest accuracies at $C = 100$. The accuracy seems to be logarithmic, indicating that increasing C beyond this value wouldn't lead to much more increases in accuracy. When $C = 100$, the results are

```
Radial Kernel (default parameters; C = 1):
Training Set Accuracy   = 99.54%
Validation Set Accuracy = 97.41%
Testing Set Accuracy    = 97.40%
```

We see that this model performed significantly better than the other models we trained in the previous sections. These values indicate that, for our dataset, a support vector machine with a radial kernel, $\gamma = 1/[n\_features]$, and $C = 100$ is the best choice among the models we have tested.

1.3 Multi-Class Logistic Regression

For the third part of this project, we implement multi-class logistic regression to classify the handwritten digit dataset from the previous sections. We extend the binary classifier from the first section to instead classify an example over 10 classes (one for each possible digit), versus the binary classifer which was a set of 10 classifiers each determining whether a digit is "their" digit or not. We implement the code in a similar fashion to the first section, except now we use a *softmax* activation function instead of the *sigmoid* activation. This gives us the correct posterior probabilities over the 10 classes. We use the cross-entropy error function so that we optimize on correct classifications versus output values. The results are as follows:

```
Training Set Accuracy   = 93.45%
Validation Set Accuracy = 92.48%
Testing Set Accuracy    = 92.55%
```

So, our multi-class logisitc regression model performs better than the linear regression model, but not as well as the support vector machine.

1.4 Results

Through the experimentation of these various classifiers, we've found, for the task of classifying images of handwritten digits, that a Support Vector Machine with a radial kernel and parameters $\gamma = 1/[n\_features]$ and $C = 100$ performs the best in terms of accuracy. Although different data sets will require different models for the best performance, we can argue that using Support Vector Machines with similar parameters may be a good place to start when finding a classifier for a similar task.