STAD68 Final Project

Johanthan Tsoi
1001502590

Data

Retrieved from: https://archive.ics.uci.edu/ml/datasets/spambase

In the dataset of spambase, it contains information that are contained in the e-mails individuals have received from sources ranging from advertisements to phishing websites. The dataset of the size 4601 has 56 attributes, where first 54 convey the percentage of a certain word or symbols that appears in the email and the remaining 3 contains the lengths of sequence of capital letters. And the last column or attribute contains the label vector that distinguishes if the email is indeed spam labeled as 1, or not labeled as 0. From a simple analysis of the data we found the total count of labels to be of ratio 1.5 to 1 with more non-spam or 0s. (Found in count plot graph).

With this binary classification dataset, we are going to predict if the email is spam or not by given only the content of the e-mail filling in each attributes mentioned above. The main task is to compare three algorithms and find the best algorithm/classifier that best suits this dataset. With the use of *Hoeffding's Confidence Interval*, we can have a standard to compare each algorithm on. We will use the confidence interval to compare the lengths of the interval and more importantly we will evaluate each interval by the empirical risk it produces, the lesser clearly the better. Whilst the main task is to minimize empirical risk, each algorithm also tackles problems such as achieving low approximation error but resulting in high estimation error, or known as overfitting. Since empirical risk or true error only accounts through the testing set, high estimation error would cause a huge influx of empirical risk creating a faulty algorithm. By the use of class weights and regularization parameters in our algorithm in order to minimize the chances of these problems.

In our dataset we will use read_data to parse through all the data that is fed through the algorithms. The parsing uses an ensemble from scikit-learn, train_test_split that uses random number generator that splits the whole data into 80% training data and 20% testing data. Where both sets of data are then split into their corresponding attribute set and labelling set for training and testing purposes. The training set is mostly used to feed through the learning algorithm and testing set is solely used for testing the accuracy of the function. The data is also transformed by standardization, hence will scale the data for a better result for algorithms that uses Euclidean distances such as SVM, logistic regression in our case.

Methods

In this binary classification problem, three different algorithms are implemented on the spambase dataset. Each learning algorithm is fed training data then used to predict test data which were isolated from the algorithm apart from when it was parsed through the system. The three learning algorithms performed in this experiment are Logistic Regression, Support Vector Machine and Random Forest Decision Tree.

Logistic regression is based on a sigmoid function that determines the classification task by finding the inner product of a parameter w, and data vector x. In this classification we are determining the probability of certain data vector belonging to either of the binary. The main idea of this algorithm is the use of a loss function that is monotonically with sigmoid function

$\frac{1}{1+\exp(y<w,b>)}$. And since we know that the sigmoid (log) function is monotonic therefore we can apply Empirical Risk Minimization (ERM) to the loss function by the convex characteristic. Convex functions have an advantage that can achieve ERM by the property of it's local minimum is also the global minimum which as a result the global minimum of a loss function is the ERM.

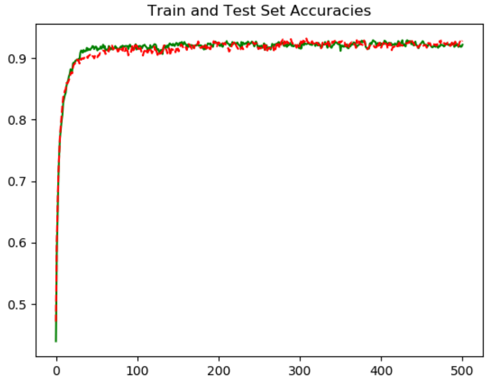
In our second algorithm, Support Vector Machine, similar to logistic regression, uses a halfspace (w,b) to separate the binary classification data. SVM finds a linearly separable halfspace (or radial basis…etc dependent on kernels) that best fits the dataset. In our case, instead of finding a hard line/halfspace that separates the data, we will use *Soft-SVM* where it gives the algorithm some relaxation on the separation. The general idea of SVM is it maximizes the norm from the half space to closest two opposite binary data point whilst maximizes the two points Euclidean distance. Furthermore, the risk minimization portion is done by Stochastic Gradient Descent methods by a learning rate of 0.01 in our training data.

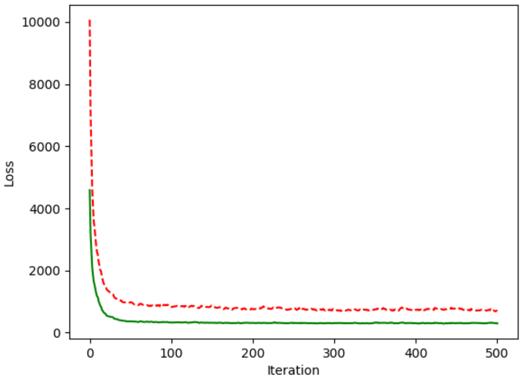
Lastly, the last algorithm to use is a Random Forest Decision Tree algorithm. It is namely a collection of multiple decision trees that iterates over multiple attributes and to determine the threshold of each feature for its labelling. And the final verdict of the labelling is determined by the majority vote of each prediction the individual tree from the collection in order to correctly label the data.

Training

In our logistic regression classifier, we use packages in scikit-learn again fit our classifier with our data. The function has a parameter of class weight, and with our count of labels there are approximately 1.5times more non spam than spam emails, therefore we give the more weight to the non-spam class. The training data is first fit into the logistic regression model with both attributes and labelling vectors. Then test data will feed the model only the attribute values to output a labelling function. Test data is only used to predict the labelling function. For a iteration we achieved a training error of 0.081 and testing error of 0.100, which only has a difference of ~002. In our classification report we have a table that lists the precision of non-spam and spam labelling which both averages a 90% accuracy over several iterations.

Secondly, in our SVM algorithm, it was implemented through tensorflow packages to train and test the data. Again equivalent to all algorithms, train and test data was again split with scikit-learn. We then train the model with the equation of $< w, x_i > + b$ and minimization equation $\min_{w,b}(\lambda\ \|w\|^2 + L_S^{hinge}\ (w,b))$, which is the process of minimizing both regularized norm (w) from the hyperplane to a data point and the hinge loss of the function simultaneously. The minimization equation is then optimized under tensorflow's *Stochastic Gradient Descent* (SGD) with step sizes 0.01 until the loss function achieves minimum. Ultimately when data is fed into the algorithm, it is fed in batches of 100 which represents the training algorithm only takes in 100 data points at a time and is iterated at a preset of 500 times. The training results achieves 90% accuracy and minimized loss fairly quickly after ~50 iterations.



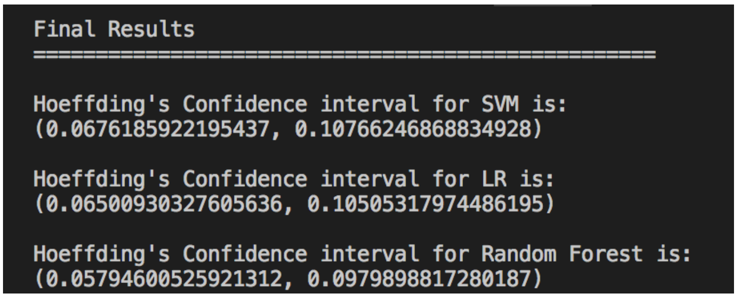


Lastly, in the random forest algorithm, it is fed the same training data as before. We use scikit-learn ensemble for random forest classifier. It takes in parameters such as the number of trees in the collection and class weights. We set the number of trees to 30 and class weights same as logistic regression due to the outnumbering of non-spam emails. After the initializing of the random forest trees, training data is then fit into the model. Training error is found to be very small after several iterations, averaging at 0.1%.

Evaluation

In all of our output of three algorithms, we can see a trend of training set out performing testing set. But that is only by a small margins and both sets do as well as one another reaching averages of 90% accuracy in most cases. Since there is a random factor in the cases, it is inevitable that there will be bad sets of training data that causes our learning algorithm to not perform as well. Secondly, our testing data are completely isolated from the training set as there are no bleeding of data from testing set used to train data.

Finally, we have our outputs of each algorithm to the corresponding *Hoeffding's* confidence interval:



```
Final Results
==================================================

Hoeffding's Confidence interval for SVM is:
(0.0676185922195437, 0.10766246868834928)

Hoeffding's Confidence interval for LR is:
(0.06500930327605636, 0.10505317974486195)

Hoeffding's Confidence interval for Random Forest is:
(0.05794600525921312, 0.0979898817280187)
```

These confidence intervals are the computed *Hoeffding's* confidence interval under the same set of data ran in three different algorithms with $\delta = 0.05$. All three confidence interval was able to achieve a lower bound of highest 6% empirical risk/testing error and and upper bound of approximately 10%. And in this case, random forest performs the best but in the testing phase of this project, random forest consistently out performs the other two algorithms in both the length of the confidence interval and also the lowest lower bound.

From the three algorithms we have found that random forest classifier is the best to classify our data. There are few challenges that surfaced during this project such as the inability to visualize the dataset because of the dimensionality and the chances of overfitting during training. As mentioned above the overfitting misconception might be just the chances of the randomness in drawing training data from the training set causing the testing set to under-perform as mentioned in the *No Free Lunch Theorem.* Moreover, the issues of testing accuracy being lower than training accuracy still persist but the margins of the two are very minimal and testing accuracy still achieves over 90%, therefore it could be disregarded for overfitting.