Gurleen Singh Dhody 2000063241
Machine Learning Assignment 3 CSCI-B 555

[Note: Code compatible with Python 2.7.12, environment used Anaconda 4.0.0 while working on the code]

Question 1)

a) Naïve Bayes works by comparing the probability of occurrence of a given sample for all classes. Meaning the feature data/location of data determines in which class it belongs depending on the highest probability. The feature data hence directly impacts the probability of a sample being in a class. If a same feature or data is added to each sample, then that feature will contribute equally to probabilities of all classes. Before the same feature was added the samples were already labelled with their respective classes. Since addition of the same feature/data does not changes our hypothesis (nor weakening or strengthening it) this same feature addition is useless. [Naïve Bayes works by separating the classes of samples through the distribution of the data. If any feature in the data is the same the distribution stands at the same point and naïve Bayes won't be able to identify the class. As per the number of classes it clubs the various distributions of samples.]

   Thus, the addition of columns of 1 is useless as it only leads to more space-time complexity of calculations. Similarly, any other feature data that is same or appears to be same can be removed.

b) Answer in attached code file.

c) Answer in attached code file. The Neural Net has also a bias node in the hidden layer which gets updated with each iteration. The samples passed in each epoch are randomized.

d) **Naïve Bayes** works by separating the classes of samples through the distribution of the data. As per the number of classes it clubs the various distributions of samples as per their spread. The major assumption is based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

   Logistic Regression works by predicting on a classification problem. It may use any logistic function. It generally works by predicting a binary response for the classes.

   Neural Net is basically setting of neurons in the same fashion as the brain works. These neurons are activated by implementing activation functions (example: sigmoid, tanh). For their activation, they need to learn specific weights so that can activate with respect to given data accordingly.

The Neural Net (16 Hidden Network Layer Height) and Logistic Regression without regression performed similarly. Though the neural net took comparatively more time to train with respect to logistic. While Naïve Bayes was the fastest to train though its error larger but comparatively ok with the other two machine learning approaches. Below is the Standard Error and Average Error Reported for the 3 algorithms they ran 5 times each.

Sample Data: 1000; Test Data: 5000; 5 Runs

```
Best parameters for Random: {}
Average error for Random: 49.644 +- 0.48015119752
Best parameters for Naive Bayes: {'notusecolumnones': True}
Average error for Naive Bayes: 25.884 +- 0.542423758772
Best parameters for Logistic Regression: {'regwgt': 0.05, 'tolerance': 1e-06, 'step-size': 20, 'iteratio
rizer': 'None'}
Average error for Logistic Regression: 23.772 +- 0.339137775425
Best parameters for Naive Bayes Ones: {'notusecolumnones': False}
Average error for Naive Bayes Ones: 25.884 +- 0.542423758772
Best parameters for Neural Network: {'nh': 16, 'transfer': 'sigmoid', 'epochs': 100, 'stepsize': 0.01}
Average error for Neural Network: 23.752 +- 0.429929473658
```

Logistic Regression: 1000 iterations, tolerance: 1e-06, regularizer None

Neural Net: 100 epochs, alpha 0.01

All in all, except Naïve Bayes, both Neural Net and Logistic Regression could outperform Linear Regression.

```
Best parameters for Random: {}
Average error for Random: 49.2 +- 0.0
Best parameters for Naive Bayes: {'notusecolumnones': True}
Average error for Naive Bayes: 26.3 +- 0.0
Best parameters for Linear Regression: {'regwgt': 0.0}
Average error for Linear Regression: 23.38 +- 0.0
Best parameters for Logistic Regression: {'regwgt': 0.0, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000,
Average error for Logistic Regression: 20.66 +- 0.0
Best parameters for Neural Network: {'nh': 4, 'transfer': 'sigmoid', 'epochs': 100, 'stepsize': 0.01}
Average error for Neural Network: 21.52 +- 0.0
Best parameters for Naive Bayes Ones: {'notusecolumnones': False}
Average error for Naive Bayes Ones: 26.3 +- 0.0
```

The above-mentioned data is for the SUSY complete dataset that I ran out of curiosity. With more features and the same meta-parameters, the performance increases considerably, except the linear regression and Naïve Bayes.

Gurleen Singh Dhody 2000063241
Machine Learning Assignment 3 CSCI-B 555

Question 2)

The customized logistic classification problem after implementing in the python code has the following performance

```
Best parameters for Random: {}
Average error for Random: 50.01 +- 0.659242146711
Best parameters for Logistic Alternative: {'alpha': 0.001, 'epoch': 200}
Average error for Logistic Alternative: 24.734 +- 0.446640953608
Best parameters for Linear Regression: {'regwgt': 0.0}
Average error for Linear Regression: 25.258 +- 0.541029589727
```

It is better than linear regression by a little margin. Algorithm was run 10 times.

It is implemented using batch gradient descent approach, with line search for optimum route.

[Note: In the gradient descent rule the gradient is added rather than subtracted because my implementation of the differentiated term of cost function has a common -ve value missing that I thought of later adding directly to the gradient rule instead of applying in the '_gradientFunction' in my code.]

The cost function and gradient descent rule with iterative approach derivation is present in an attached second pdf document named "**Question2Derivation.pdf**".

The function of P(y=1) is like sigmoid function. If its graph is plotted the function has probable values between ½ and 1 if w^Tx is +ve and between 0 and ½ if w^Tx is -ve. Similarly, this acts as an activation function for the classes. Where I describe my model as that if my sample has P(y=1) >=0.5 it has a better chance to labelled as class 1. Since P(y=1) = 1-P(y=0) so if P(y=1) is less than 0.5 which means P(y=0) is greater than 0.5. So, it is better to predict the sample belongs to class 0. All in all, it can be said that it is a replacement of the sigmoid function. Though it needs a good optimization route so it can select optimum solution. Using alternate stochastic gradient descent, I found it many times got stuck in local optimum, producing a bad result. It produces nearly the same result with respect to vanilla logistic regression using sigmoid function. Also, the custom function has a ugly gradient descent iterative formula compared to sigmoid. Since sigmoid has epsilon the log helps in contracting the term. [Check derivation]

Gurleen Singh Dhody 2000063241
Machine Learning Assignment 3 CSCI-B 555

Question 3)

```
Best parameters for Elastic Logistic Regression: {'regwgt': 0.05, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000,
Average error for Elastic Logistic Regression: 23.06 +- 0.0
Best parameters for L2 Logistic Regression: {'regwgt': 0.05, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'reg
Average error for L2 Logistic Regression: 22.94 +- 0.0
Best parameters for L1 Logistic Regression: {'regwgt': 0.05, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'reg
Average error for L1 Logistic Regression: 22.94 +- 0.0
Best parameters for Random: {}
Average error for Random: 48.94 +- 0.0
Best parameters for Logistic Regression: {'regwgt': 0.0, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regular
Average error for Logistic Regression: 23.12 +- 0.0
```

[1st Case] A detailed description of l1 l2 and elastic net(l1+l2) regularizer run on complete SUSY dataset. Train Data: 1000 and Test Data: 5000 regwgt: {0.01,0.05,.1}

```
Best parameters for Elastic Logistic Regression: {'regwgt': 0.1, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regularizer':
Average error for Elastic Logistic Regression: 22.23 +- 0.0
Best parameters for L2 Logistic Regression: {'regwgt': 0.1, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regularizer': 'l2'}
Average error for L2 Logistic Regression: 22.22 +- 0.0
Best parameters for L1 Logistic Regression: {'regwgt': 0.5, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regularizer': 'l1'}
Average error for L1 Logistic Regression: 22.19 +- 0.0
Best parameters for Random: {}
Average error for Random: 49.646 +- 0.0
Best parameters for Logistic Regression: {'regwgt': 0.1, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regularizer': 'None'}
Average error for Logistic Regression: 22.186 +- 0.0
```

[2nd Case] A detailed description of l1 l2 and elastic net(l1+l2) regularizer run on complete SUSY dataset. Train Data: 10000 and Test Data: 50000 regwgt: {0.1,0.5,1}

```
Best parameters for Elastic Logistic Regression: {'regwgt': 0.5, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regularizer':
Average error for Elastic Logistic Regression: 22.22 +- 0.0
Best parameters for L2 Logistic Regression: {'regwgt': 0.5, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regularizer': 'l2'}
Average error for L2 Logistic Regression: 22.21 +- 0.0
Best parameters for L1 Logistic Regression: {'regwgt': 0.5, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regularizer': 'l1'}
Average error for L1 Logistic Regression: 22.19 +- 0.0
Best parameters for Random: {}
Average error for Random: 49.42 +- 0.0
Best parameters for Logistic Regression: {'regwgt': 2.5, 'tolerance': 1e-06, 'step-size': 20, 'iterations': 1000, 'regularizer': 'None'}
Average error for Logistic Regression: 22.14 +- 0.0
```

[3nd Case] A detailed description of l1 l2 and elastic net(l1+l2) regularizer run on complete SUSY dataset. Train Data: 50000 and Test Data: 10000 regwgt: {.5,2.5,5}

Interestingly with increase in samples there is no effect of regularizaton. The model initself can learn from the data and in terms of bias and overfitting is moving towards an optimal trade-off solution, without need of regularization to tune it.

The regularization weights have been increased in the 2nd & 3rd case, because in my implementation my regularized parameters are divided by number of samples to tune down the regularization with increase in samples. However, to show the comparison of vanilla logistic regression to regularizers I have increased the regularizer weights so that overall regularization values remains same in the 2 runs that I have made over the dataset.

[Note: The high step size is acting as a counter balance to counter the small lambda values with respect to data size]

Gurleen Singh Dhody 2000063241
Machine Learning Assignment 3 CSCI-B 555

a) L2 and L1 parameter $|w|_2$ and $|w|_1$ would be added respectively to the cost function. When gradient descent approach would be used the differentiation of the cost function along with the differentiation of l1 and l2 would be added to each gradient descent step. The iterative approach has been implemented in the code. The utilities python file contained the l1 and l2 functions and their differentiation. These were used to implement the regularization.

b) Elastic Net a combination of L1 and L2 being applied together is used. L2 and L1 are applied together with correspondence to weights of the features. The iterative update rule is provided in the file '**Question3Elastic.pdf**'

c) **L1** regularizer using soft-thresholding penalizes the weights in a manner that it tries to reduce the weight of features that are not-important (i.e do not contribute much in prediction of y). This it does by reducing the weight of such features to 0. In a way it is able to do feature selection. Hence preventing overfitting and reducing space-time complexity of our model.

   L2 regularizer is used since it prevents any important feature to gain complete control over the model for prediction. Hence preventing bias. By keeping a sanity check it forces weights to have small values for each feature. Preventing any feature to dominate over another.

   Elastic Net is a combination of L2 and L1 and brings the best of both. Elastic Net covers grouping effect which is a drawback for the lasso. In Lasso If there is a group of variables among which the pairwise correlations are very high, then the lasso tends to select only one variable from the group and does not care which one is selected.

REFERENCES

- https://web.stanford.edu/~hastie/Papers/B67.2%20(2005)%20301-320%20Zou%20&%20Hastie.pdf
- https://www.wolframalpha.com/
- https://theclevermachine.wordpress.com/2014/09/06/derivation-error-backpropagation-gradient-descent-for-neural-networks/
- https://theclevermachine.wordpress.com/2014/09/11/a-gentle-introduction-to-artificial-neural-networks/
- https://en.wikipedia.org/wiki/Elastic_net_regularization
- http://stats.stackexchange.com/questions/184029/what-is-elastic-net-regularization-and-how-does-it-solve-the-drawbacks-of-ridge
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier