**Applied Machine Learning Final Project**

Sneha Lakshmi Nyayapati ([slnyayap@umail.iu.edu)](mailto:slnyayap@umail.iu.edu))

Nirad Ranjan Parhi ([nparhi@umail.iu.edu)](mailto:nparhi@umail.iu.edu))

December 9,2016

**Introduction**

In this project, we aim to write 2 classification algorithms

* + Logistic Regression
  + K Nearest Neighbors

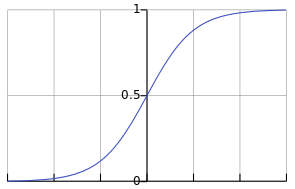
and apply them to two different datasets to evaluate their performance. We will first discuss about the Logistic Regression Algorithm and its performance on the Prima India Diabetes Dataset in classifying the patients who have diabetes from the other patients who do not have diabetes. Then we will discuss about the K Nearest Neighbors Algorithm implementation and its use in classifying the Iris dataset taken from UCI database. The Algorithm tries to classify the Iris dataset into 3 different types based on their features.

**Logistic Regression**

Logistic Regression is a very commonly used linear classification problem. It is mostly used for binary classification. Logistic Regression is named after the logistic function which is at the core of the working of logistic Regression. The logistic function is given as:

f(x) =

It can be plotted as:



The value of a logistic function always lies between 0 and 1 and we can use this property to do binary classification very easily. The output value of the logistic function less than 0.5 are rounded off to 0 and those greater than 0.5 are rounded off to 1.

We have also used the stochastic gradient descent method to find the optimal weights for each attribute so that the error in prediction is minimized. While performing the Stochastic Gradient Descent algorithm, each instance is shown to the model one at a time. The model makes a prediction on the instance and the error is calculated. The model is updated to reduce the error on the next prediction.

**Pima Indians Diabetes Data Set**

<https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes>

The Pima Indians Diabetes Data Set contains details about female patients, all 21 years old or above, of the Pima Indian Heritage. The task is to predict if a person will suffer from diabetes given her medical details.

There are 9 attributes in the data set. Attributes 1 to 8 are the attributes containing the various medical details. The 9th attribute is the class variable which denotes if a person has diabetes or not. Class variable 0 denotes that the person does not have diabetes and Class variable 1 denotes that a person has diabetes. All the attributes are numeric.

The details of the attributes, in sequence, are as follows:

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

**Results after running the algorithm on the dataset**

We ran the code by varying the parameters i.e. number of iterations and learning rate.

**For iterations =100 and rate = 0.1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 |
| Accuracy | 79.130 | 74.782 | 76.521 | 77.826 | 75.217 |
| True positive | 40 | 45 | 44 | 47 | 54 |
| True Negative | 142 | 127 | 132 | 132 | 119 |
| False Negative | 39 | 46 | 42 | 36 | 35 |
| False Positive | 9 | 12 | 12 | 15 | 22 |

**For iterations =200 and rate = 0.2**

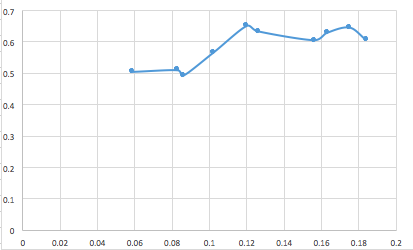
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 |
| Accuracy | 75.652 | 73.913 | 76.086 | 79.130 | 74.782 |
| True positive | 56 | 51 | 53 | 58 | 55 |
| True Negative | 118 | 119 | 122 | 124 | 117 |
| False Negative | 33 | 33 | 29 | 31 | 41 |
| False Positive | 23 | 27 | 26 | 17 | 17 |

**Average Accuracy = 76.30**

For the 10 runs the True positive Rate and False Positive Rate are summarized in the following table:

|  |  |
| --- | --- |
| False Positive Rate | True Positive Rate |
| 0.059 | 0.506 |
| 0.083 | 0.511 |
| 0.086 | 0.494 |
| 0.102 | 0.566 |
| 0.12 | 0.651 |
| 0.126 | 0.634 |
| 0.156 | 0.606 |
| 0.163 | 0.629 |
| 0.175 | 0.646 |
| 0.184 | 0.607 |

**ROC**

****

**Area Under the Curve = 0.073**

**Weka Output**

=== Run information ===

Scheme: weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4

Relation: Diabetes\_Weka

Instances: 767

Attributes: 9

1

95

60

18

58

23.9

0.26

22

No

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

Logistic Regression with ridge parameter of 1.0E-8

Coefficients...

Class

Variable No

====================

1 -0.1231

95 -0.0351

60 0.0133

18 -0.0006

58 0.0012

23.9 -0.0896

0.26 -0.9442

22 -0.0148

Intercept 8.3981

Odds Ratios...

Class

Variable No

====================

1 0.8842

95 0.9655

60 1.0134

18 0.9994

58 1.0012

23.9 0.9143

0.26 0.389

22 0.9853

Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 594 77.4446 %

Incorrectly Classified Instances 173 22.5554 %

Kappa statistic 0.4773

Mean absolute error 0.3105

Root mean squared error 0.3966

Relative absolute error 68.2638 %

Root relative squared error 83.1913 %

Total Number of Instances 767

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.886 0.433 0.792 0.886 0.836 0.485 0.829 0.889 No

0.567 0.114 0.727 0.567 0.637 0.485 0.829 0.710 Yes

Weighted Avg. 0.774 0.322 0.769 0.774 0.767 0.485 0.829 0.827

=== Confusion Matrix ===

a b <-- classified as

442 57 | a = No

116 152 | b = Yes

**About the code**

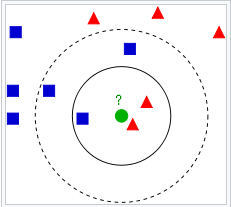
We have implemented Logistic Regression along with Stochastic Batch Gradient Descent optimization. The weights assignedto theattributes are updated according to an equation so that the error in prediction on the training set is minimized after each iteration. A learning rate parameter has been used in the update equation to control the updates of the weights. Normalization of the attributes has also been done in the code. In the initial runs without normalization of the data set, the accuracy was coming to be very low. After normalizing the data (each attribute between 0 to 1), the accuracy of the predictions has increased sufficiently.

**Improvements in the code**

The logic for cross validation by splitting the dataset into a number of folds will be added. Currently, cross validation is being done manually by altering the input data set.

The logic for Batch Gradient Descent needs to be added in the code base. This way we can check which optimization method works better on a particular data set – stochastic or batch.

**K Nearest Neighbors**

****

(Is the circle similar to the triangles or the square? KNN will tell.)

The K Nearest Neighbors algorithm is mostly used for classification of data (although it can be used for regression also). This algorithm is very use to interpret and depends on the similarity between two points to make a prediction. The similarity is usually found out by calculating the distance between two instances.

The K Nearest Neighbors algorithm will use the total training set as its model. When a new example is provided to it, it will scan through the entire training set to find the most similar instances and assign the majority of the class variable that is most prevalent in those instances to the new unseen instance.

Distance between points is used as the scale to find out similarity between the instances. In our case, we are using Euclidian distance as the measure. KNN is a lazy algorithm as it does not build a model till a prediction is required. As it uses the entire training dataset as its model, it can be expensive on resources also.

**Iris Data set**

<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>

This is a dataset containing the details of the flower of the Iris plant. The data set contains 150 instances. There are 3 classes in the dataset. Each instance has 4 attributes regarding the measurement of the Iris flower. The attribute details are as follows:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
   1. Iris Setosa
   2. Iris Versicolour
   3. Iris Virginica

All the attributes are numeric in nature. The dataset is not skewed. All the 3 classes are evenly distributed.

**Results after running the algorithm on the dataset**

The following table shows the Accuracy of the algorithm in 10 test runs:

|  |  |
| --- | --- |
| Run | Accuracy |
| 1 | 62.22 |
| 2 | 75.55 |
| 3 | 75.55 |
| 4 | 88.88 |
| 5 | 95.55 |
| 6 | 88.88 |
| 7 | 97.77 |
| 8 | 86.66 |
| 9 | 95.55 |
| 10 | 86.66 |

**Average Accuracy = 85.327**

**About the code**

The code implementation depends on finding the distance between a new unseen instance from all other instances in the training dataset. We use Euclidian distance as the measure in our algorithm. After the distance is found, we pick up the ‘K’ instances in the training data set which are closer to the unseen instance. Finally, the majority of the class value of the nearest instances is assigned to the unseen instance as its class value.

**Improvements in the code**

The code needs to be further modified to allow for n-fold cross validation. Currently, the same is being done by randomly sampling the dataset to choose random test set and training set

Data normalization needs to be included into the algorithm. Normalization might increase the accuracy of the model.

**Conclusion**

In this project, we have used two different classification algorithms running on two different data sets. The Logistic Regression algorithm does the job of binary linear classification whereas K Nearest Neighbor Algorithm performs multiclass classification algorithm with ease. In our experiments, the KNN algorithm performs better than the Logistic regression, but this is the case when they are run on 2 different data sets. The KNN algorithm is sensitive towards the training data set. Accuracy changes significantly if the training data set changes.

**References**

1. <https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>
2. <https://en.wikipedia.org/wiki/Logistic_function>
3. <https://en.wikipedia.org/wiki/Receiver_operating_characteristic>
4. Programming collective intelligence by Toby Segaran