CT-475 Machine Learning & Data Mining

Assignment - 3

Categorical Classification Using Logistic Regression

Programming Language Used: Python

Date : 03rd December 2018

Team Members:

Name : Surya Balakrishnan Ramakrishnan
Student ID : 18231072
Class : 1 CSD 1 MSc Computer Science (Data Analytics)

Name : Sai Krishna Lakshminarayanan
Student ID : 18230229
Class : 1 CSD 1 MSc Computer Science (Data Analytics)

Overview:

The task in hand is to design implement and evaluate a machine learning algorithm from scratch without using any packages for the purpose of implementing core operations. We have selected **Logistic Regression Algorithm** using the **(One vs All)** approach using regularised cost function and gradient for the task. We have used the **Python** programming language and **Jupyter Notebook** to interpret the python code.

Description of Algorithm:

The logistic regression is a machine learning algorithm which works based on the principle of regression analysis, which predicts the probability of an outcome. The logistic regression is the extension of the linear regression, in which the dependent variable is categorical not continuous. The logistic regression can either be binomial or multinomial in other words it can either have two outcomes or more than two outcomes. The logistic regression evaluates the probability of each of the input to belong to a particular category. The baseline model with the Logistic Regression is to predict the most frequent outcome from the outcomes of all data points. Contrary to the Linear regression, where the baseline model predicts the average of all data points as the outcome.

Logistic Regression tries to perform the following:

- Models the probability of an occurring event depending on the values of the independent variable/variables.
- Estimate the probability of the occurrence of an event occurs vs the probability of the non-occurrence of an event.
- Predict the effect of a series of variables on the binary response variable
- Classify observations into different classes by estimating the probability of an observation to be present in a particular category or not.

Functions within the Logistic Regression:

1) **Sigmoid Function:**

   The sigmoid function is responsible for the prediction and classification of the given input. The sigmoid function is also called logistic function. The sigmoid function is defined as:

$$z = \Theta^T x$$
$$sigmoid(z) = \frac{1}{1 + e^{-(z)}}$$

   Where: e is the Euler's Number which is equal to 2.71828

Theta (Θ) is the weight.

2) Regularised Cost Function:

Regularisation in the logistic regression is important because of the reason that, it helps to improve the performance of the algorithm for un-seen data. The regularisation of the cost function ensures that proper weighting is added to the parameters. In order to accomplish this we add a new hyperparameter lambda (λ) to control the regularisation strength. The regularised cost function is given by the following formula:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2.$$

Where:

m is length of the target variable.
h_Θ is the sigmoid value obtained from the sigmoid function.
x is the matrix values of total data elements and the parameters.
y is the array of target variable.
Lambda (λ) is the hyperparameter.

3) Regularised Gradient Function:

The regularised gradient function is one of the optimisation strategies used in logistic regression. The gradient is used while the model is being trained, which tweaks the parameters iteratively to attain local minimum. This is done to minimise the cost function as much as possible. The gradient is the measure of change in the output of a function for change in input. In other words the regularised gradient measures the change in the weight of the parameter with respect to the error. For the model to learn in a proper way the slope of the gradient needs to be computed in a way that the value of the theta (Θ) is optimum. The regularised gradient function is given by the formula:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \quad \text{for } j \geq 1$$

Where:

m is length of the target variable.
h_Θ is the sigmoid value obtained from the sigmoid function.
x is the matrix values of total data elements and the parameters.
y is the array of target variable.
Lambda (λ) is the hyperparameter.

Implementation of the Algorithm:

In the implementation we have used the logistic regression for multi-class classification of the owls dataset based on the 4 parameters body_length, body_width, wing_length, wing_width. The algorithm classifies the input based on the given data which one of the following category of owls, Long Eared Owl, Snowy Owl, Barn Owl the input falls under. The classification of the class of owls is done by computing the probability of each input for each of the class, and the input is classified to the class with maximum probability.

Before the implementation of the algorithm we have visualised the data by plotting the dataset in the following ways:

1) Scatter plot of body_length, body_width with respect to the class of the owls.
2) Scatter plot of wing_length, wing_width with respect to the class of the owls.
3) Box plot of body_length with respect to the class of the owls.
4) Box plot of body_width with respect to the class of the owls.
5) Box plot of wing_length with respect to the class of the owls.
6) Box plot of wing_width with respect to the class of the owls.
7) Distribution of body_length
8) Distribution of body_width
9) Distribution of wing_length
10) Distribution of wing_width.

In the next step we perform the mean normalisation of the dataset. We have used the sklern library to split the given data into training and test dataset. The test dataset has been split into $(1/3)^{rd}$ of the entire dataset. The training dataset has been split to $(2/3)^{rd}$ of the entire dataset.

In the next step the implementation of the logistic regression is done. The first step is to calculate the sigmoid value of the dataset. Once the sigmoid is calculated the next step is to calculate the regularised cost and the regularised gradient functions. Now with the help of the regularised cost function and regularised gradient function we find the optimal value of Theta (Θ), which is then used to train the model.

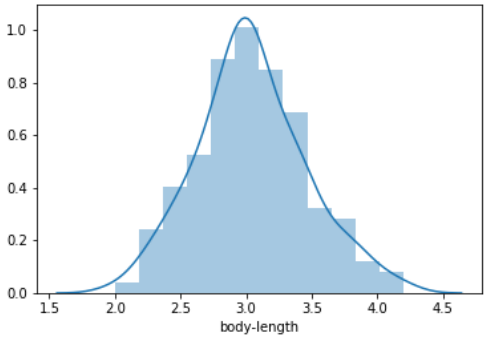
The next step is to train our model in the training dataset. The training of the model is done using the one versus all methodology. In the one versus all methodology the multiple categories of the target variable is converted into n binary classification problem, where n is the number of categories within the target variable. In this case we have 3 different types of owls in the dataset so the algorithm converts it into 3 binary classification problems. For each of the type of owls we have the binary classification. Once we have the 3 binary classification results the algorithm merges these binary classification results into one and the model is trained.

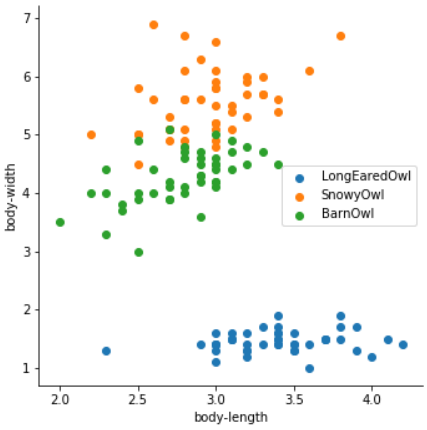
Then in the next step, we run the prediction and print the accuracy of the model. We consider the fact that any value for the accuracy above 80% is the optimum value which the algorithm

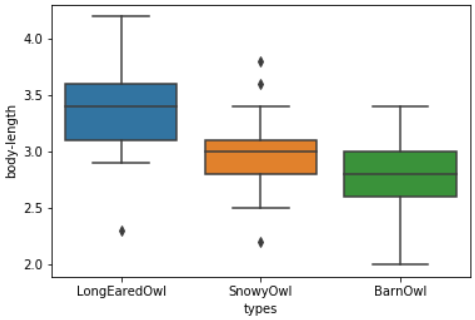
should return. Then we print the confusion matrix. All the plots and confusion matrix is exported in png format. Also we have also split the data set into test and training dataset for 10 iterations and for each individual iteration we have computed the accuracy and the accuracy has been exported and saved as a csv file. Then we have computed the final accuracy by obtaining the average of all the 10 accuracies obtained in each iteration.

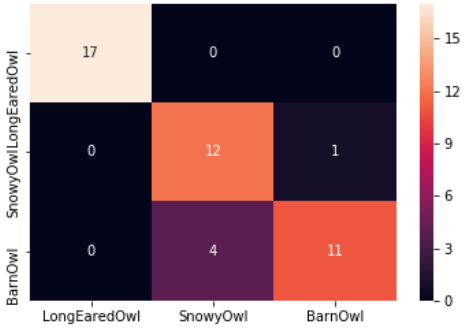
Observations:

Sample plots please see the output folder for entire plots.









- From the normal distributions we see a perfect normal distribution for body length and somewhat of a normal distribution for wing length and a non-normal distribution for body width and wing width.
- From the scatter plots we can see that while plotting body length, body width with respect to the class of own and similarly for wing length, wing width we observe that the long eared owl is the smallest type of owl and snowy owl is the largest.
- Some of the barn owls have similar structure to the snowy owls.
- The box plot median values supports the second observation.
- We obtain an accuracy of approximately. 84%
- From the confusion matrix we can see the True Positive, True Negative, False Positive, False Negative values for each of the classes of owls whose total results to 45 which is the count of the test dataset.

Conclusion:

- The misclassification is observed in case of Barn Owl and Snowy Owl
- This misclassification could be because of the overlapping of Barn Owl, Snowy Owl obtained in the scatterplots.
- Since we used one versus all approach it is time consuming plus the dataset is small which could also account for the misclassification.
- Some other approach could be used to improve the accuracy which is suitable for a small dataset and a dataset with imbalances.

Work Split-up for the Documentation:
Surya Balakrishnan Ramakrishnan (18231072) hereby referred as Surya
Sai Krishna Lakshminarayanan (18230229) hereby referred as Sai Krishna

- Overview, Description of Algorithm, Sigmoid Function :- Surya
- Regularised Cost and Gradient Function :- Sai Krishna
- Implementation of Algorithm paragraph 1-3 :- Sai Krishna
- Implementation of Algorithm paragraph 4-6 :- Surya
- Observation Points 1-3 :- Surya
- Observation Points 4-6 :- Sai Krishna
- Conclusion Points 1-2 :- Surya
- Conclusion Points 3.4 :- Sai Krishna

Work Split-up for the Code:
Surya Balakrishnan Ramakrishnan (18231072) hereby referred as Surya
Sai Krishna Lakshminarayanan (18230229) hereby referred as Sai Krishna

Detailed Split-up is provided in the code.

- Distribution plots 2 each
- Box Plot 2 each
- Scatter Plot 1 each
- Pre-processing, Splitting of data and sigmoid function :- Sai
- Logical Regression (Regularised cost, Regularised Gradient, optimal theta), one vs all, training, prediction. :- Surya
- Accuracy, error, Confusion matrix (actual vs predicted) , final table output. :- Sai

Appendix:

Document Reference: https://en.wikipedia.org/wiki/Logistic_regression

Code:

```
#This portion of the coding till the end of second distribution plot is done by Sai Krishna Lakshminarayanan (18230229)Data Analytics
# Importing the packages required for executing the algorithm
#install the missing packages if any is present
import statistics
import random
import numpy as num
from subprocess import check_output
import pandas as panda
import matplotlib.pyplot as plot
from scipy import optimize as op
from sklearn.metrics import confusion_matrix
import csv
import seaborn as sea
from sklearn.cross_validation import train_test_split
#make sure owls.csv is present in the working directory
owls = panda.read_csv('owls.csv')# obtaining the dataset from the local disk

#Data visualisation of the owl data to get some understanding
#Distribution plot of body length
sea.distplot(owls["body-length"])
plot.savefig('body-lengthdistplot.png')#storing the output locally as a png file

#Distribution plot of body width
sea.distplot(owls["body-width"])
plot.savefig('body-widthdistplot.png')

#This portion of the coding till the end of fourth distribution plot is done by Surya Balakrishnan Ramakrishnan (18231072)Data Analytics
#Distribution plot of wing length
sea.distplot(owls["wing-length"])
plot.savefig('wing-lengthdistplot.png')
```

```
#Distribution plot of wing width
sea.distplot(owls["wing-width"])
plot.savefig('wing-widthdistplot.png')

#The first two box plots are done by Sai Krishna
#Box plot of the owl types with respect towards the body length
sea.boxplot(x="types", y="body-length", data=owls)
plot.savefig('body-lengthboxplot.png')

#Box plot of the owl types with respect towards the wing length
sea.boxplot(x="types", y="wing-length", data=owls)
plot.savefig('wing-lengthboxplot.png')

#The last two box plots are done by Surya
#Box plot of the owl types with respect towards the body width
sea.boxplot(x="types", y="body-width", data=owls)
plot.savefig('body-widthboxplot.png')

#Box plot of the owl types with respect towards the wing width
sea.boxplot(x="types", y="wing-width", data=owls)
plot.savefig('wing-widthboxplot.png')

#first scatter plot is done by Sai Krishna
#scatter plot of the owl types with respect to body length and width
sea.FacetGrid(owls, hue="types", height=5).map(plot.scatter, "body-length", "body-width")
plot.legend(loc='center right');
plot.savefig('body-lengthVsbody-width.png')

#second scatter plot is done by Surya
#scatter plot of the owl types with respect to wing length and width
sea.FacetGrid(owls, hue="types",height=5).map(plot.scatter, "wing-length", "wing-width")
plot.legend(loc='lower right');
plot.savefig('wing-lengthVswing-width.png')

#This part of the code till the start of regularised cost function is done by Sai Krishna
#Data preprocessing to perform the regression later
types = ['LongEaredOwl', 'SnowyOwl', 'BarnOwl']#the three different types of owls present
e = owls.shape[0] #Total count of the owl data set
f = 4 #No of features it has other than the target variable
g = 3#Number of classes present in the target variable types which denotes the three types of
owl

x = num.ones((e,f + 1))
y = num.array((e,1))
x[:,1] = owls['body-length'].values# giving the body length values
x[:,2] = owls['wing-length'].values # giving the wing length values
x[:,3] = owls['body-width'].values# giving the body width values
x[:,4] = owls['wing-width'].values# giving the wing width values
y = owls['types'].values #labels which for which the classification is to be performed
```

```
a=10
sum=0
acc=[]
err=[]
for b in range(a): # to perform 10 iteration of the prediction and accuracy
    for j in range(f):
        X[:, j] = (X[:, j] - X[:,j].mean())

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state =
b*7)#randomly splitting the data
    def sigmoid(z):# to perform the sigmoid operation
        return 1.0 / (1 + num.exp(-z))

#from this till the start of accuracy is done by Surya
    def regularisedcostfunction(t, X, y, l = 0.1): #Regularized cost function
        e = len(y)
        h = sigmoid(X.dot(t))
        reg = (l/(2 * e)) * num.sum(t**2)
        #regularises the cost function by adding proper weights inorder to avoid over or less
values
        return (1 / e) * (-y.T.dot(num.log(h)) - (1 - y).T.dot(num.log(1 - h))) + reg

    def regularisedgradientfunction(t, X, y, l = 0.1): #Regularized gradient function
        e, f = X.shape
        t = t.reshape((f, 1))
        y = y.reshape((e, 1))
        h = sigmoid(X.dot(t))
        reg = l * t /e
        #obtaining the local minimum and minimizing the cost function
        return ((1 / e) * X.T.dot(h - y)) + reg

    def logreg(X, y, t):
        result = op.minimize(fun = regularisedcostfunction, x0 = t, args = (X, y),
                             method = 'TNC', jac = regularisedgradientfunction)

        return result.x

    all_t = num.zeros((g, f + 1))

    #One vs all
    i = 0
    for owl in types:
        #set the labels in 0 and 1
        tmp_y = num.array(y_train == owl, dtype = int)
        optTheta = logreg(X_train, tmp_y, num.zeros((f + 1,1)))
        all_t[i] = optTheta
        i = i+1
    #Predictions
    P = sigmoid(X_test.dot(all_t.T)) #probability for each owl
    p = [types[num.argmax(P[i, :])] for i in range(X_test.shape[0])]
```

```
#From this part till the end coding is done by Sai Krishna
    count = len(["ok" for ix, label in enumerate(y_test) if label == p[ix]])
    c=(float(count) / len(y_test))*100#accuracy value
    acc.append(c)
    error=1-(c/100)#error value
    err.append(error)
    print("Test Accuracy of",b+1,"scenario is", c, '%',"\n")# Test accuracy in each iteration
    print("Error possibility of",b+1,"scenario is",error,"\n")# erro…

 confusionmatrix = confusion_matrix(y_test, p, labels = types) #Regularized gradient
function
sea.heatmap(confusionmatrix, annot = True, xticklabels = types, yticklabels =
types);#generating heat map along for the matrix
plot.savefig('confusionmatrix.png')

iteration=[]
testtotal=[]
for i in range(a):
    iteration.append(i+1)#assigning the iteration values
    testtotal.append(y_test.shape[0])#assigning the test total value
final_table = panda.DataFrame(# creating a data set to store the values of iteration,test
total,accuracy and error
    {
        'Number of Iteration ':iteration,
        'Test Data':testtotal,
        'Accuracy %': acc,
     'Error=(1-accuracy)': err,
    })
print(final_table)#desired output
final_table.to_csv('finaltable.csv', encoding='utf-8', index=False)#storing the end result as a
csv
```