Lab Sheet 4

Logistic Regression for Classification (15/07/2019)

PART A : Prerequisites for Logistic Regression implementation

1. Plot the attached dataset training_data using scatter plot. There is a target feature with discrete values 0,1. If the target feature is 1, the samples should be shown as red circle. If the target feature is 0, the samples should be shown as green x.

2. Plot a line y=(2x+3)

3. Define a function sigmoid(z) that takes one parameter z and computes 1/(1+e^-z). Create a vector V with 10 values randomly in the range [-1000,1000]. Transform V to V' that consists of respective sigmoid values using the defined function. Observe the range of output values in V'.

4. Define a function hypothesis(theta, X) that takes two vectors as parameters, theta and X. If sigmoid(theta.X) >= 0.5, output 0 else output 1.

5. Define a function cost(theta,X,y) to compute the error

$$\text{Error} = 1/m*\Sigma\text{-yilog}(h_\theta(\text{xi})) - (1\text{-yi})\log(1\text{-}h_\theta(\text{xi}))$$

   Where xi is the ith sample and yi is the ith label, h_θ(xi) is the hypothesis(theta,xi)

PART B : Implementation of logistic regression

6. Implement gradient descent algorithm for logistic regression.

```
def logistic_reg()
    # read training dataset

    # convert dataset to a feature matrix X

    # normalizing feature matrix X

    # stack columns with all ones in feature matrix
    # target feature to be taken in a separate vector
    y = dataset[:, -1]

    # initial theta values
    theta = np.matrix(np.zeros(X.shape[1]))

    # theta values after running gradient descent
    beta, num_iter = grad_desc(X, y, theta)

    # estimated theta values and number of iterations
    print("Estimated regression coefficients:", beta)
    print("No. of iterations:", num_iter)

    # predicted labels
    y_pred = predict(theta, X)

    # number of correctly predicted labels
    print("Correctly predicted labels:", np.sum(y == y_pred))

    # plotting regression line
    Plot line as per theta coefficients
```

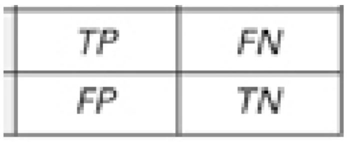
7. Use sklearn built in function to find the model

   https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

```
from sklearn.linear_model import LogisticRegression

X, y = “Load data”

clf = LogisticRegression(random_state=0, solver='lbfgs',

                         multi_class='multinomial').fit(X, y)

clf.predict(X[:2, :])

clf.predict_proba(X[:2, :])

clf.score(X, y)
```

PART C: Performance Evaluation of the classifier

8. Plot the confusion matrix. Sample code as follows



TP	FN
FP	TN

```
from sklearn.metrics import confusion_matrix
```

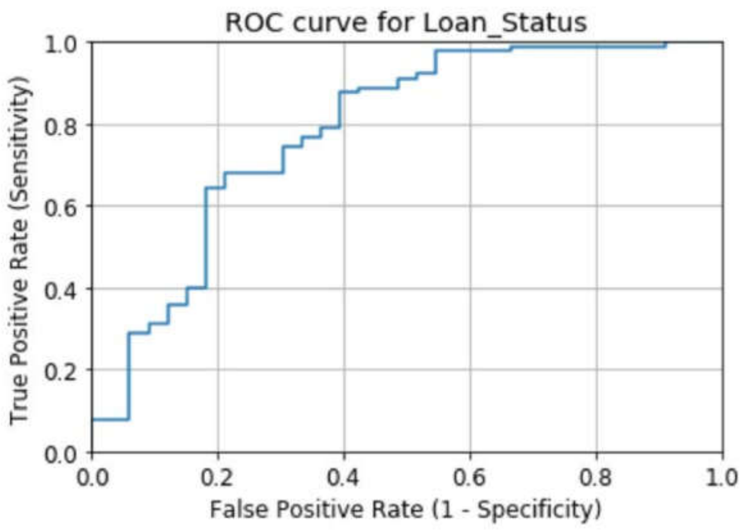
```
y_true = [2, 0, 2, 2, 0, 1]
y_pred = [0, 0, 2, 2, 0, 2]
confusion_matrix(y_true, y_pred)
```

9. Compute the accuracy score.
10. Print a classification report using the following sklearn function

```
from sklearn.metrics import classification_report
y_true = [0, 1, 2, 2, 2]
y_pred = [0, 0, 2, 2, 1]
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true, y_pred, target_names=target_names))
```

11. Plot ROC curve for loan status

    https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html



```
import numpy as np
from sklearn import metrics
y = np.array([1, 1, 2, 2])
scores = np.array([0.1, 0.4, 0.35, 0.8])
fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
```

12. Compare the performance of classifiers obtained in 6 and 7

PART D: Extra credit

13. Extend the logistic regression algorithm for three class data set.