Logistic Regression

Logistic regression is a basic clissifer. It is a binary-classification method (to separate two classes) that can be also modified to classify any multi-class system. Similar to the linear regression, the machine tries to determine the parametes of a model (it can be a curve or a hyper surface) by minimising the cost function using a traning set (X). In contrast to the linear regression, the determined curve is used to separate the different classes (y) in the feature (hyper) surface.

Important : You should complete the code where ever there is a "Task" or a "Question" and give some comments where it is needed.

In [ ]:
```
# Task : open mixedBallsLarge2.csv as a Pandas dataframe
import pandas as pd
df_balls =
```

In [ ]:
```
# Task : Print the dataframe :
```

The table contains the data of some balls distributed on the earth. There are two ball-classes : blue and red. The features x1 and x2 are the cordinates of balls on the earth in meter.

In [ ]:
```
# We get prepared to plot our classes in the feature space.
# Task : What are df_blue and df_red ?
import numpy as np
df_blue = df_balls.iloc[ np.where(df_balls['colour']=='blue')]
df_red = df_balls.iloc[ np.where(df_balls['colour']=='red')]
```

In [ ]:
```
# Question : What does np.where do ?
```

In [ ]:
```
# We plot the distribution of balls on the earth :

from matplotlib import pylab as plt

plt.scatter( df_blue["x1"], df_blue["x2"], c='b' )
plt.scatter( df_red["x1"], df_red["x2"], c='r', marker='.' )
plt.xlabel("x1 (m)")
plt.ylabel("x2 (m)")
plt.show()
```

The balls are more or less separated in x1-x2 space with mixing in some regions.

In [ ]:
```
# Task : construct the feature matrix and the label vector from df_bal
# Attention : X and y should be numpy arrays.
X =
y =
```

In [ ]:
```
# Task : convert the classes from text to binary values by using Label
y =
```

In [ ]:
```
# Task : construct the training and the test sets by splitting the dat
#        use 70% of the data for training. Put the random state to 0.
X_train, X_test, y_train, y_test =
```

In [ ]:
```
# We make an instance of the LogisticRegression class :
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression(solver='liblinear')
```

In [ ]:
```
# Task : similar to the linear regression, fit logiftic regression noc
```

In [ ]:
```
# Task : print the classification accuracy (score) for both training a
```

In [ ]:
```
# Computing the confusion matrix :
from sklearn.metrics import confusion_matrix
cm = confusion_matrix( y_test, reg.predict(X_test) )

#Task : print the confusion matrix :
```

In [ ]:
```
# Task : compute the following quantities from the confusion matrix :
TN =
TP =
FP =
FN =
```

In [ ]:
```
# Task : compute manually the accuracy from the confusion matrix and c
#        the score value for the test set you computed three cells abc
```

In [ ]:
```
# Task : compute the precision from the confusion matrix
```

In [ ]:
```
# Task : make a prediction for test set :
y_test_pred =

# Recalcul the precision by using sklearn :
from sklearn.metrics import precision_score
precision_score(y_test, y_test_pred)
```

In [ ]:
```
# We have fitted a linear multi-variable model to find the seoaration
# The parameters of the model can be calculated :
theta = reg.coef_
theta_0 = reg.intercept_

# Task : print the parameters :
```

In [ ]:
```
# Plotting the data and the determined border
x_1 = np.linspace(-5,25,1000)
x_2 = -(theta_0[0] + theta[0][0]*x_1)/theta[0][1]
plt.plot( x_1, x_2, c='black' )

plt.scatter( df_blue["x1"], df_blue["x2"], c='b' )
plt.scatter( df_red["x1"], df_red["x2"], c='r', marker='.' )
plt.xlabel("x1 (m)")
plt.ylabel("x2 (m)")
plt.show()
```

Polynomial Border

As we saw in the last plot most classes can be seprated in the feature space by a straight line. However a linear model is not always the best classifier. We will compare the accuracy of the linear and nonliniear fits in this section.

In [ ]:
```
# Task : open the file mixedBallsHomogen2 as a Pandas dataframe.

df_hb =
```

In [ ]:
```
# Task : make dataframes for blue and red balls and plot them in x1-x2
df_hb_blue =
df_hb_red =
```

As you notice the border between two classes is a curve rather than a line.

In [ ]:
```
# Task : construct the feature matrix and the label vector from df_bal
X =
y =
```

In [ ]:
```
# Task : transform the lable vector to binaries :
```

In [ ]:
```
# Task : construct the training and the test sets by splitting the dat
#        use 70% of the data for training. Put the random state to 0.
```

In [ ]:
```
# Task : fit the linear model by making an instance of LogisticRegress
```

In [ ]:
```
# Task : compute the fit score (accuracy) for the test set
```

In [ ]:
```
# Task : compute and print the confusion matrix for the test sample
cm2 =
```

In [ ]:
```
# Task : compute the model parameters and plot the fitted line and the
#        This is completely similar to the last section.
theta =
theta_0 =
```

The line does not efficiently separate the classes. We try to fit a polynomial instead of a line :

In [ ]:
```
# Task : use the PolynomialFeatures class and make a degree=2 feature
pof =
Xp =
```

In [ ]:
```
# Task : make the training and test sets from Xp and y (random_state=2
Xp_train, Xp_test, yp_train, yp_test =
```

In [ ]:
```
# Task : make an instance of LogisticRegression and fit it to the trai
reg_pod =
```

In [ ]:
```
# Task : compute the score for the test set and compare it
#        to the score of the linear fit :
```

In [ ]:
```
# Task : compute and print the confusion matrix for the test sample
cm3 =
```

Question : compare cm3 with cm2 what do you conclude ?

Question : Fit again the date by making Xp with degree=3 and 4. How does the score chabges ?

In [ ]:
```
# Run this cell to plot the whole data and the determined boundry

a, b = np.meshgrid( np.linspace(0,10,1000), np.linspace(0,10,1000) )
Z = np.array( [ a.ravel(), b.ravel() ] ).T
X_p = pof.fit_transform(X)
Z_p = pof.fit_transform(Z)

Xp_train, Xp_test, yp_train, yp_test = \
        train_test_split(X_p, y, test_size=.3, random_state=2)

reg_pod = LogisticRegression(solver="liblinear")
reg_pod.fit(Xp_train, yp_train)

plt.contour(a, b, reg_pod.predict(Z_p).reshape(a.shape), colors='yellc
plt.scatter( df_hb_blue["x1"], df_hb_blue["x2"], c='b' )
plt.scatter( df_hb_red["x1"], df_hb_red["x2"], c='r' )
```

Convergence Diagram

In [ ]:
```
# Explaine the following code :

def computeScores(tests, X, y) :
    Xp_train, Xp_test, yp_train, yp_test = \
        train_test_split(X, y, test_size=tests, random_state=2)
    reg_pod = LogisticRegression(solver="liblinear")
    reg_pod.fit(Xp_train, yp_train)
    return reg_pod.score(Xp_train, yp_train), reg_pod.score(Xp_test, y

from sklearn.preprocessing import PolynomialFeatures
pof = PolynomialFeatures(degree=3)
X_p = pof.fit_transform(X)

n_p = 10
s_train, s_test= np.zeros(n_p), np.zeros(n_p)
ratio = np.linspace(.1, .95, n_p)

for i in range(n_p) :
    s_train[i], s_test[i] = computeScores( ratio[i], X_p, y )
```

In [ ]:
```
# Plotting the convergence curve
plt.plot( 1-ratio, s_train, c='b' )
plt.plot( 1-ratio, s_test, c='r' )
plt.xlim( left=-.01, right=1.01 )

plt.ylim( top=1.001)
plt.xlabel("Train_set / whole data")
plt.ylabel("Accuracy")
plt.title(" Blue: training set, Red: Test set")
plt.show()
```

Exercise 1

Find the balls that are mis-classified for the mixedBallsLarge2.csv. Plot the data and the border. Mark the misclassified calss with green colour.

Exercise 2

Consider the file Social_Network_Ads.csv. It contains the online shopping data of some internet users. The Purchased column is either 0 or 1 that shows if a user has bought the article or not. Take the Age and the EstimatedSalary as the features and classify the data by using simple linear logistic regression. Compute the score, precision and confusion matrix for the test sample. Plot the whole data and the separation border together.

Attention : the Age and the EstimatedSalary do not span the same value ranges. You should rescale them by using the StandardScaler class.

In [ ]:
```

```