Exercise5

October 6, 2021

1 Exercise 5: Logistic Regression

In this exercise, you will use logistic regression to classify breast cancer as either malignant or benign. First run the code below to print and read the description of the data set.

[2]:
```
%matplotlib inline

from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
import numpy as np
import matplotlib.pyplot as plt

DataCancer=load_breast_cancer()
print(DataCancer.keys())
print(DataCancer.DESCR)

X_features=DataCancer.data
Y_targetClass=DataCancer.target
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename'])
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
    - area
    - smoothness (local variation in radius lengths)
    - compactness (perimeter^2 / area - 1.0)
    - concavity (severity of concave portions of the contour)
    - concave points (number of concave portions of the contour)
    - symmetry
    - fractal dimension ("coastline approximation" - 1)

    The mean, standard error, and "worst" or largest (mean of the three
    worst/largest values) of these features were computed for each image,
    resulting in 30 features.  For instance, field 0 is Mean Radius, field
    10 is Radius SE, field 20 is Worst Radius.

    - class:
            - WDBC-Malignant
            - WDBC-Benign

:Summary Statistics:

===================================== ====== ======
                                       Min    Max
===================================== ====== ======
radius (mean):                        6.981  28.11
texture (mean):                       9.71   39.28
perimeter (mean):                     43.79  188.5
area (mean):                          143.5  2501.0
smoothness (mean):                    0.053  0.163
compactness (mean):                   0.019  0.345
concavity (mean):                     0.0    0.427
concave points (mean):                0.0    0.201
symmetry (mean):                      0.106  0.304
fractal dimension (mean):             0.05   0.097
radius (standard error):              0.112  2.873
texture (standard error):             0.36   4.885
perimeter (standard error):           0.757  21.98
area (standard error):                6.802  542.2
smoothness (standard error):          0.002  0.031
compactness (standard error):         0.002  0.135
concavity (standard error):           0.0    0.396
concave points (standard error):      0.0    0.053
symmetry (standard error):            0.008  0.079
fractal dimension (standard error):   0.001  0.03
radius (worst):                       7.93   36.04
texture (worst):                      12.02  49.54
perimeter (worst):                    50.41  251.2
area (worst):                         185.2  4254.0
smoothness (worst):                   0.071  0.223
compactness (worst):                  0.027  1.058
```

```
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages
570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
```

1.0.1 Scale the features to have zero mean and unit variance. Use scaled features in all the following questions. Also, please use random_state = 0 in the train_test_split whenever used.

A) Use logistic regression, without regularization (penalty='none'). Find the accuracy of the model.

- You may need to change the solver (the optimization method) as well as the iterations (max_iteration) for results of optimization to converge. You can for example set, solver='lbfgs', max_iter=1000 in the logistic regression function. Learn more about input arguments of logistic regression function here: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[5]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X_features, Y_targetClass,␣
 ↪random_state=0)

scaler = StandardScaler().fit(X_train)

X_train_t = scaler.transform(X_train)
X_test_t = scaler.transform(X_test)

LogRegModel = LogisticRegression(solver='lbfgs', max_iter=1000, penalty='none')
LogRegModel.fit(X_train_t, Y_train)

print('Accuracy: %.2f' % LogRegModel.score(X_test_t, Y_test))
```

```
Accuracy: 0.94
```

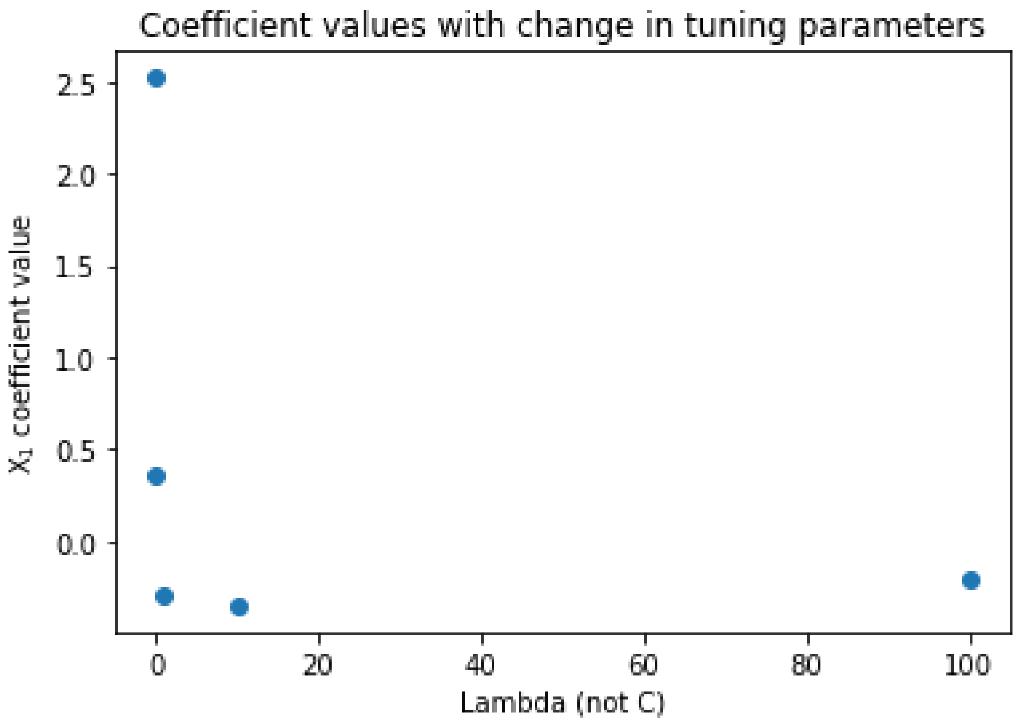
B) Apply Ridge regularization to Logistic regression. Try tuning parameters [0.01, 0.1, 1, 10, 100] and plot the coefficient of the first feature at each value of the regularization tuning parameter. What do you observe? Is your observation expected?

[8]:
```
coefs = []
params = [0.01, 0.1, 1, 10, 100]

for lambd in params:
    c = 1 / lambd
    LogRegModel = LogisticRegression(penalty='l2', C=c, solver='lbfgs',␣
↪max_iter=1000)
    LogRegModel.fit(X_train_t, Y_train)

    coefs.append(LogRegModel.coef_[0][0])

plt.scatter(params, coefs)
plt.xlabel('Lambda (not C)')
plt.ylabel('X$_1$ coefficient value')
plt.title('Coefficient values with change in tuning parameters')
plt.show()
```



(AP): This observation is consistent with the properties of ridge regression. A larger lambda (smaller c) means that there is more of a shrinkage penalty, meaning that the coefficients will tend towards 0. This is what is observed in the plots above.

C) Find the best tuning parameter of Ridge regularization in logistic regression using 5-fold cross validation. Here also try the tuning parameter values of [0.01, 0.1, 1, 10, 100]. Print the best tuning parameters and the accuracy when the best tuning parameters is selected.

[19]:
```
params = [0.01, 0.1, 1, 10, 100]

X_trainval = X_train
```

```
Y_trainval = Y_train
X_trainval_t = scaler.transform(X_trainval)

best_acc = 0
best_lambd = params[0]

for lambd in params:
    c = 1 / lambd
    LogRegModel = LogisticRegression(C=c, solver='lbfgs', max_iter=1000)
    scores = cross_val_score(LogRegModel, X_trainval_t, Y_trainval, cv=5)
    mean = scores.mean()

    if mean > best_acc:
        best_acc = mean
        best_lambd = lambd

print('Best tuning parameter is lambda = %.2f.' % best_lambd)
print('Trainval accuracy with this tuning parameter = %.2f.' % best_acc)

BestModel = LogisticRegression(C=1/best_lambd, solver='lbfgs', max_iter=1000)
BestModel.fit(X_trainval_t, Y_trainval)
print('Test accuracy with this tuning parameter = %.2f' % BestModel.
 ↪score(X_test_t, Y_test))
```

```
Best tuning parameter is lambda = 1.00.
Trainval accuracy with this tuning parameter = 0.98.
Test accuracy with this tuning parameter = 0.96
```

D) From the model in previous part with the best tuning parameter selected, use the Sigmoid function to find the probability that the first test example is predicted as class 1 (malignant).

[22]:
```
probabilities = BestModel.predict_proba(X_test_t)

print(f'Probability that first test example is malignant is␣
 ↪{probabilities[0][1]}.')
```

```
Probability that first test example is malignant is 0.0013613865587136708.
```