Data Analysis and Machine-Learning

Chapter 3.1:

Generalized Linear Model (1) Classifications



1. Introduction

In line with the linear regression models covered in chapter 2, generalized linear models are based on the assumption that there is a linear relationship between input (features) X and output (target) Y. The central difference between the two is that while linear regression models assume "continuous" relationship between x and y (e.g., the increase in the value of Y depending on the increase in the value of X), generalized linear models are based on the concept of "probabilities". For example, when the output Y consists of classification data (e.g., 0, 1), the beta coefficients depict the probabilities of y being either 1 or 0.

To illustrate, let us assume that the beta coefficient for x1 is estimated as follows:

Sample interpretation of this function in a generalized linear model would be: as x1 increases by 1, the probability of y being 1 increase by 0.82. As such, under the presumption that the function depicts probabilities, the type of linear models covered in chapter 2 can be expanded and applied similarly on classification models.

1. Sample Illustrations

#Import required libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import datasets

from sklearn.linear\_model import LogisticRegression

#Generate classification samples using sklearn datasets

x, y = datasets.make\_classification(

    n\_samples=8, n\_features=2, n\_classes=2, random\_state=123,

    n\_informative=1, n\_redundant=0, n\_clusters\_per\_class=1

)

np.c\_[y, x]

output:

array([[ 1. , -0.11166034, 2.20593008],

[ 1. , 1.75652211, -0.43435128],

[ 1. , 0.80351538, 2.18678609],

[ 0. , -1.0887545 , 1.49138963],

[ 0. , -0.52755975, -0.638902 ],

[ 0. , -0.65949851, -0.67888615],

[ 1. , 0.73494356, -0.44398196],

[ 0. , -1.31281138, -0.09470897]])

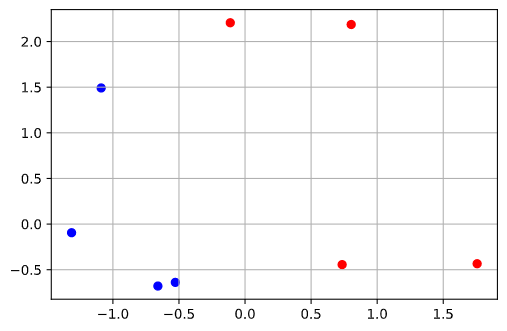
#Visualization

plt.scatter(x[:,0], x[:,1], c=y, cmap='bwr')

plt.grid()

plt.show()

Output:



#Classification (1)

#Classification (1)

xx = np.linspace(-1.5,1.5, 100)

f = -1.8\*xx-1.4

plt.figure(figsize=(10,5))

plt.subplot(1,2,1)

plt.scatter(x[:,0], x[:,1], c=y, cmap='bwr')

plt.plot(xx, f)

plt.grid()

f2 = -4\*xx+0.5

plt.subplot(1,2,2)

plt.scatter(x[:,0], x[:,1], c=y, cmap='bwr')

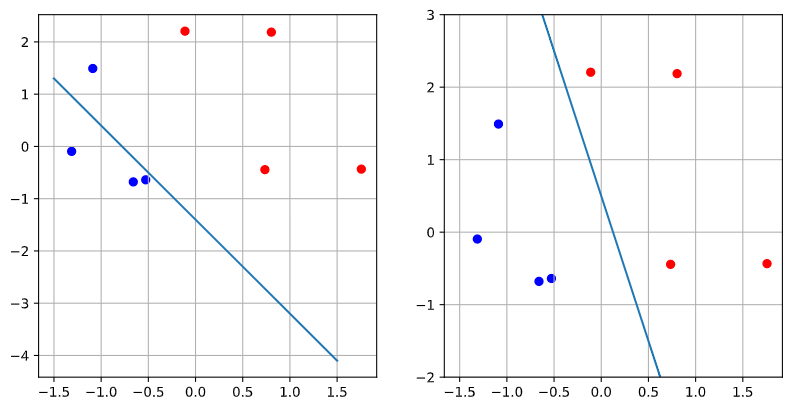
plt.plot(xx, f2)

plt.grid()

plt.ylim([-2, 3])

plt.show()

#output:



In this case, it is relatively easier to decide which model is better; it can be stated that ‘f1’ is a better model than ‘f’ because ‘f’ has one (blue dot) misclassification.

#Classification (2)

xx = np.linspace(-1.5,1.5, 100)

f = -1.8\*xx+.2

plt.figure(figsize=(10,5))

plt.subplot(1,2,1)

plt.scatter(x[:,0], x[:,1], c=y, cmap='bwr')

plt.plot(xx, f)

plt.grid()

f2 = -4\*xx+0.5

plt.subplot(1,2,2)

plt.scatter(x[:,0], x[:,1], c=y, cmap='bwr')

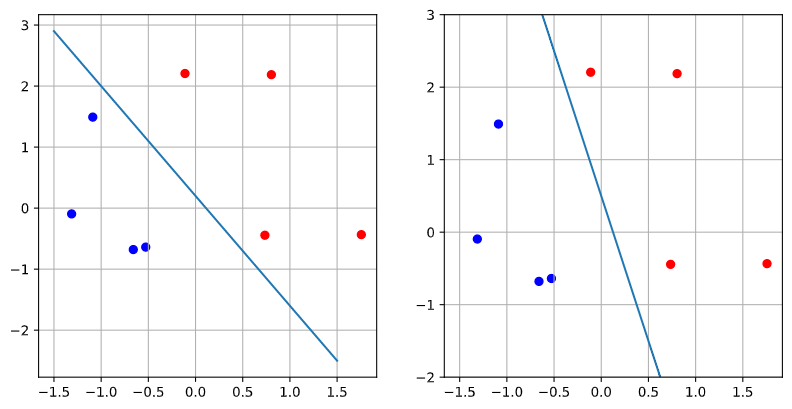
plt.plot(xx, f2)

plt.grid()

plt.ylim([-2, 3])

plt.show()

#output:



In this case, however, it is difficult to decide which model is better. Such ambiguity calls for a “continuous” evaluation method with the concept of “probability.”

1. MLE (Maximum Likelihood Estimation) and Cross-Entropy

MLE is an estimation method aiming to maximize the probability of getting the right estimation value. Assume that the probabilities of having 1 and 0 are respectively:

#Define probabilities (1, 0)

P = np.array([

    [0.85, 0.15],

    [0.25, 0.75],

    [0.77, 0.23],

    [0.12, 0.88],

    [0.25, 0.75],

    [0.87, 0.13]

])

With probabilities defined as such, the error function (cross-entropy) for this classification model can be organized as:

As implied from the function, the cross-entropy function splits the case for having the probability of getting one of the two classifications (e.g., 1) versus the case of getting the probability of the other remaining classification (e.g., 0). In this context, the cross-entropy method has benefits with regards to the prediction accuracy since it assigns cross-penalties (e.g., for each case of probabilities ‘p’ and ‘q’, respectively) to wrong predictions.

Understanding the concept of cross-entropy and respective probabilities for classifications, let us consider the probability estimations made with logistic regression model, and visualize the result for better understanding of the concept:

model = LogisticRegression().fit(x, y)

model.predict\_proba(x)

Output:

array([[0.22432162, 0.77567838],

[0.26542773, 0.73457227],

[0.16038373, 0.83961627],

[0.86194643, 0.13805357],

[0.81900014, 0.18099986],

[0.66892211, 0.33107789]])

#Visualization

xy1, xy2 = np.meshgrid(

    np.arange(x1.min() - 0.2, x1.max() + 0.2, 0.1),

    np.arange(x2.min() - 0.2, x2.max() + 0.2, 0.1)

)

xx = np.c\_[xy1.flatten(), xy2.flatten()]

yhat = model.predict\_proba(xx)

plt.scatter(x1, x2, c=y, cmap='bwr')

plt.contourf(xy1, xy2, yhat[:,1].reshape(xy1.shape), cmap='bwr', alpha=0.4)

plt.show()

