18CS10069_Q9

October 23, 2021

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import random
from math import sqrt
from tabulate import tabulate
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
np.random.seed(0)
```

[2]:
```
# prepare x and y
x = np.random.standard_normal(size=(500,2))
y = x[:, 0] * x[:, 1]
for i in range(len(y)):
    if y[i] >= 0:
        y[i] = 1
    else:
        y[i] = -1
```

(a), (b) With $\tilde{f}(x) = \theta_1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_1 x_2 + \theta_5 x_1^2 + \theta_6 x_2^2$ as classifier

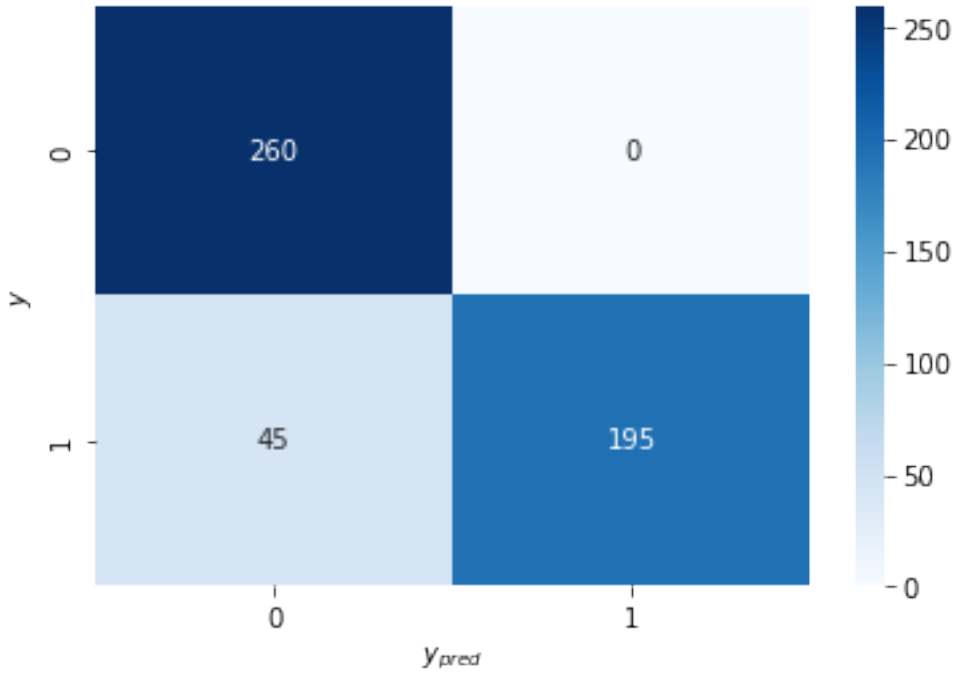
---

[3]:
```
# define feature matrix A; A . theta = y
A = np.zeros((500,6))
#set the A matrix for polynomial least squares classifier ,i.e. A.theta = y
for i in range(500):
    A[i] = np.array([1,x[i][0],x[i][1],x[i][0]*x[i][1],x[i][0]**2,x[i][1]**2])
```

[4]:
```
# find theta using pseudo inverse
theta = np.linalg.pinv(A).dot(y)

# predict the y similar to binary classification problem
y_int = np.dot(A,theta)
y_pred = np.array([1 if x>=0 else -1 for x in y_int])
```

[5]:
```
# confusion matrix
cm = confusion_matrix(y, y_pred, labels=[1,-1])
sns.heatmap(cm, annot=True, fmt='g', cmap="Blues");
plt.xlabel("$y_{pred}$");
```

```
plt.ylabel("$y$");
```

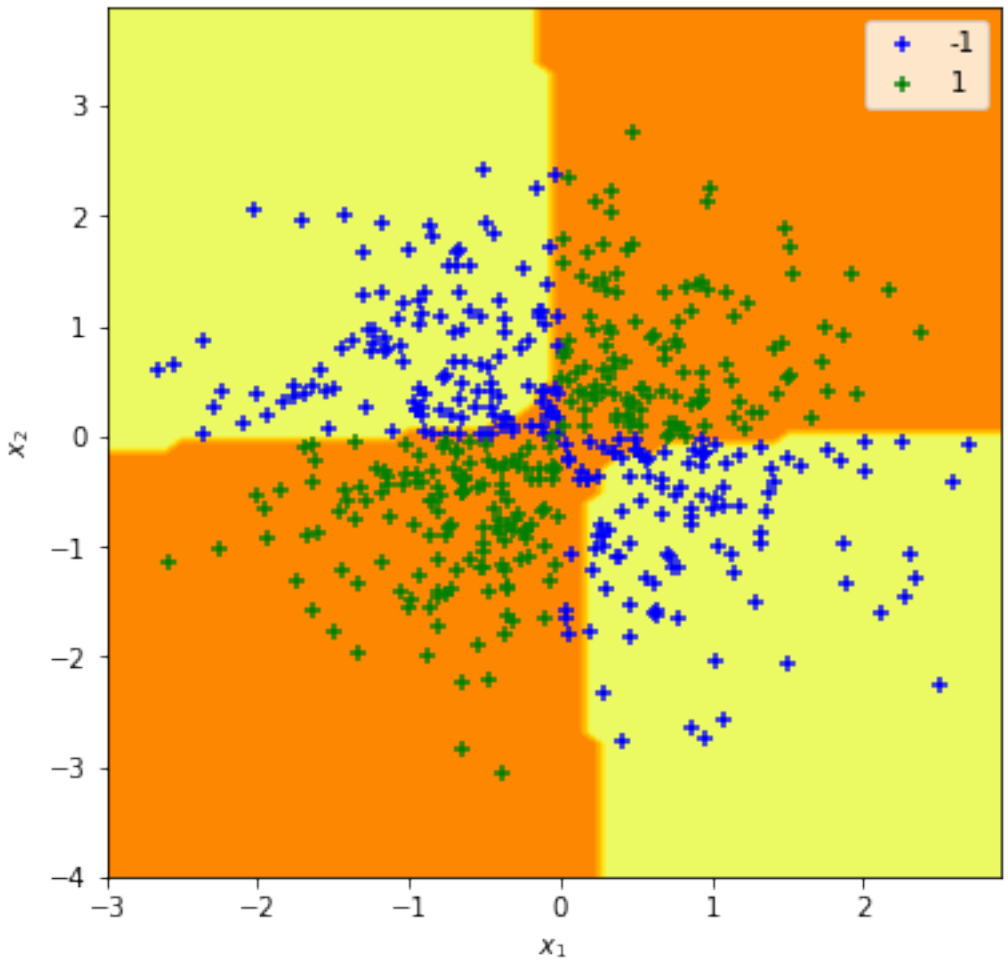


[6]:
```
error_rate = (cm[0][1]+cm[1][0])/500
print(f'Error rate :{error_rate}')
```

```
Error rate :0.09
```

[7]:
```
x1grid = np.arange(-3, 3, 0.1)
x2grid = np.arange(-4, 4, 0.1)

xx, yy = np.meshgrid(x1grid, x2grid)
r1, r2 = xx.flatten(), yy.flatten()
r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
X = np.hstack((r1,r2))

grid = np.zeros((X.shape[0], 6))
for i in range(grid.shape[0]):
    grid[i] = np.array([1, X[i][0], X[i][1], X[i][0]*X[i][1], X[i][0]**2,␣
 ↪X[i][1]**2])


y_int = np.dot(grid,theta)
y_pred = np.array([1 if x>=0 else -1 for x in y_int])
zz = y_pred.reshape(xx.shape)
```

```
plt.figure(figsize=(6,6))
plt.contourf(xx, yy, zz, cmap='Wistia')

col = {-1:'b', 1:'g'}
for class_value in [-1,+1]:
        row_ix = np.where(y == class_value)
        plt.scatter(x[row_ix, 0], x[row_ix, 1], marker='+',␣
 ↪c=col[class_value],label=class_value)

plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.legend(loc='best')
plt.show();
```

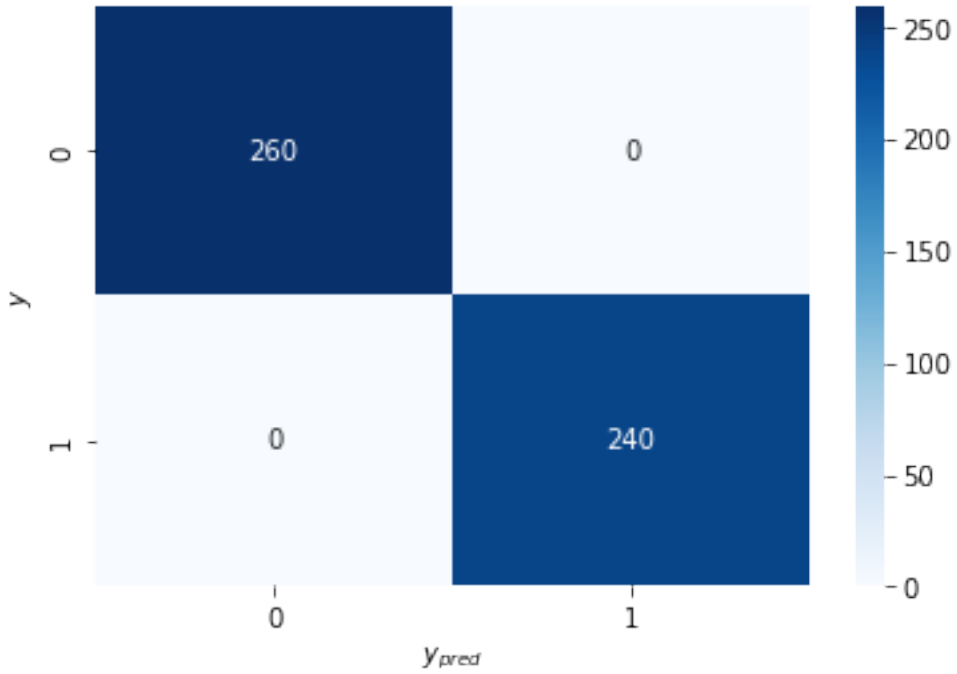


(c) With $g = x_1 x_2$ as classifier

---

[8]:
```
def gfunc(x):
    return 1 if x[0]*x[1] >=0 else -1

y_pred = np.array([gfunc(xi) for xi in x])
```

[9]:
```
# confusion matrix
cm = confusion_matrix(y, y_pred, labels=[1,-1])
sns.heatmap(cm, annot=True, fmt='g', cmap="Blues");
plt.xlabel("$y_{pred}$");
plt.ylabel("$y$");
```



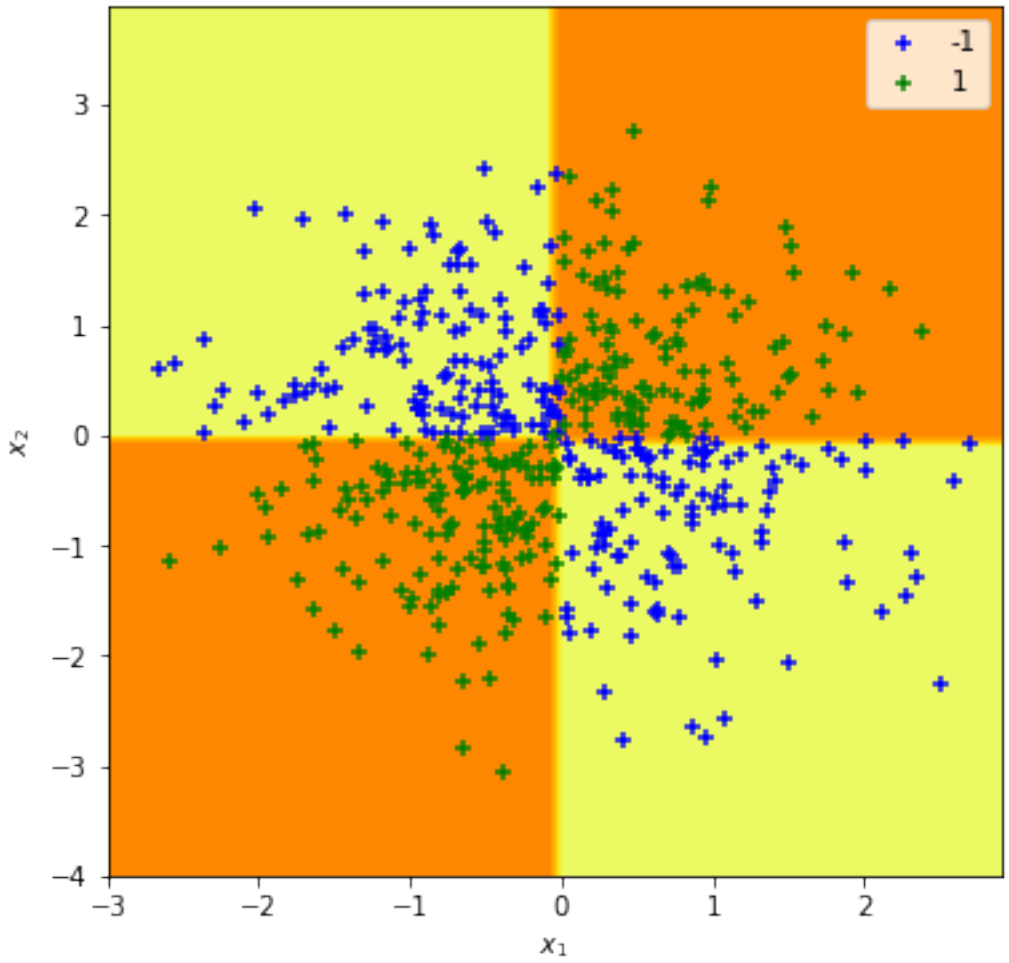
[10]:
```
error_rate = (cm[0][1]+cm[1][0])/500
print(f'Error rate :{error_rate}')
```

```
Error rate :0.0
```

Yes the error rate is 0.0

[11]:
```
x1grid = np.arange(-3, 3, 0.1)
x2grid = np.arange(-4, 4, 0.1)

xx, yy = np.meshgrid(x1grid, x2grid)
r1, r2 = xx.flatten(), yy.flatten()
r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
grid = np.hstack((r1,r2))
```

```
y_pred = np.array([gfunc(x) for x in grid])
zz = y_pred.reshape(xx.shape)

plt.figure(figsize=(6,6))
plt.contourf(xx, yy, zz, cmap='Wistia')

col = {-1:'b', 1:'g'}
for class_value in [-1,+1]:
        row_ix = np.where(y == class_value)
        plt.scatter(x[row_ix, 0], x[row_ix, 1], marker='+', c=col[class_value],␣
 ↪label=class_value)

plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.legend(loc='best')
plt.show();
```



[18]:
```
# comparing f and g parameters
print('We use similar features for both the above functions, then\n')
print(f"For f-tilde: {theta}")
print(f"For g: {[0,0,0,1,0,0]}")
print(f"Difference Norm (theta_f_tilde -  theta_g): {np.linalg.norm(theta - np.
 ↪array([0,0,0,1,0,0])):.4f}")
```

```
We use similar features for both the above functions, then

For f-tilde: [ 0.04429711  0.01248993 -0.04008282  0.67963288 -0.02858354
0.02853257]
For g: [0, 0, 0, 1, 0, 0]
Difference Norm (theta_f_tilde -  theta_g): 0.3286
```