polynomial_classifier

October 21, 2021

[260]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[261]:
```python
N = 500
def load_data():
    x_train = np.random.randn(N, 2)
    y_train = (x_train[:, 0] * x_train[:, 1] >= 0) * 2 - 1
    return x_train, y_train
```

[262]:
```python
def least_squares(A, b, factor=1.0):
    return np.linalg.inv(A.T @ A) @ (A.T @ b)
```

[263]:
```python
# calculate the matrix A
def find_coeff_matrix(x_train):
    A = np.empty((x_train.shape[0], 6))
    A[:,0] = 1
    A[:,1:3] = x_train
    A[:,3] = x_train[:,0] * x_train[:,1]
    A[:,4:6] = x_train**2
    return A
```

[264]:
```python
def confusion_matrix(y_true, y_pred, labels):
    matrix = np.zeros((len(labels), len(labels)), dtype=int)
    for i in range(len(y_pred)):
        x = labels.index(y_true[i])
        y = labels.index(y_pred[i])
        matrix[x, y] += 1
    return matrix
```

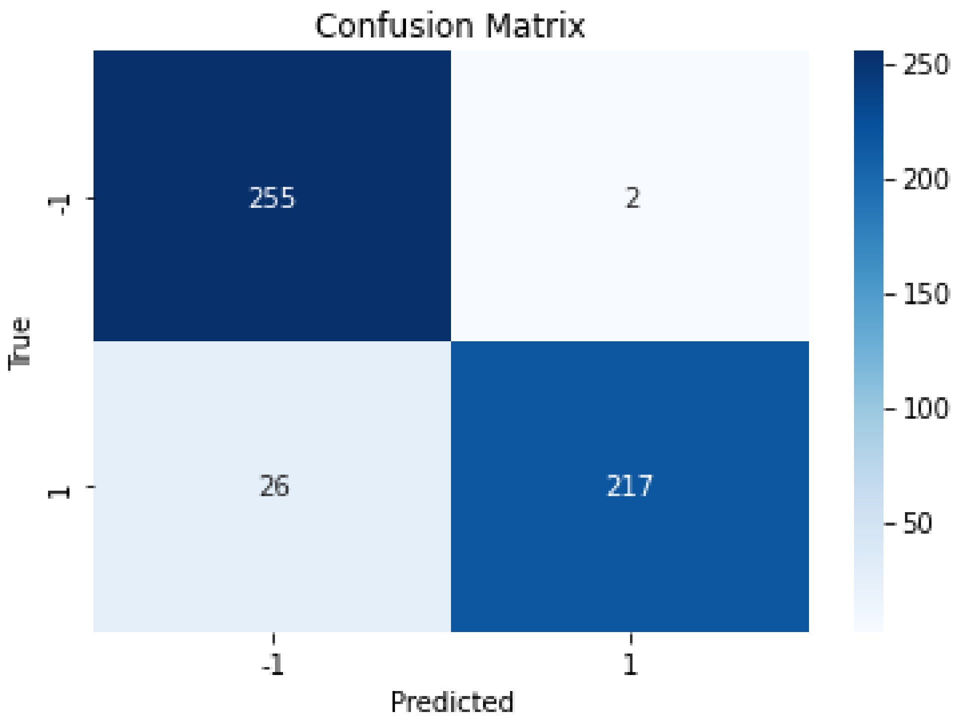
[265]:
```python
x_train, y_train = load_data()
A = find_coeff_matrix(x_train)
x_hat = least_squares(A, y_train)
y_pred = A @ x_hat
y_pred = np.sign(y_pred).astype(np.int32)

err_rate = np.mean(y_pred != y_train)
```

```
print(f'Error Rate: {err_rate * 100:.4f}%\n')
```

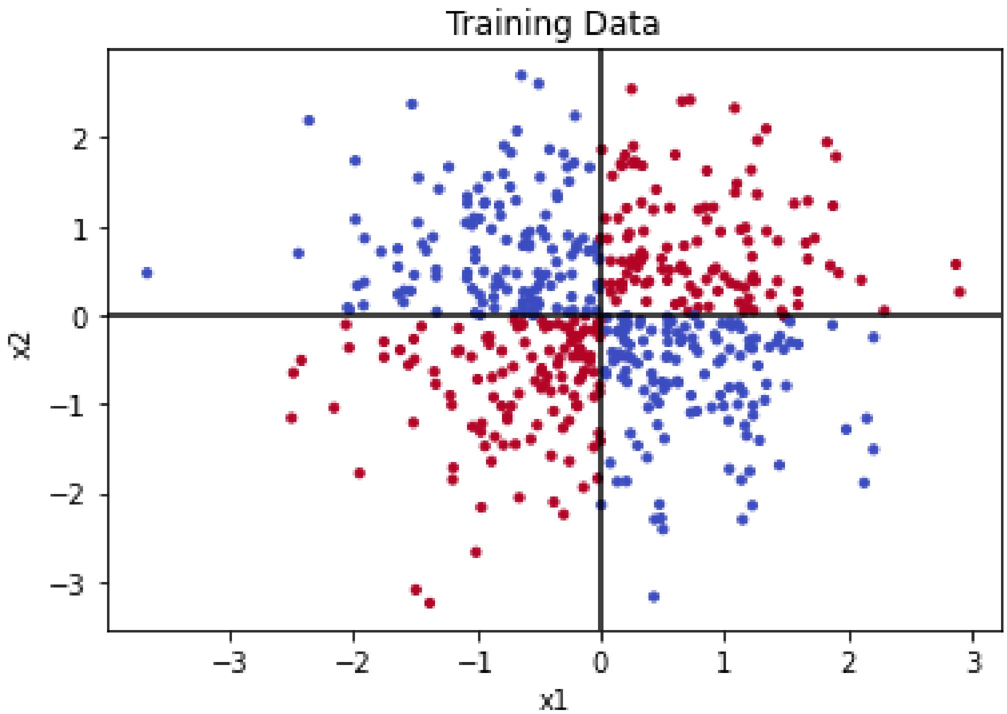
```
Error Rate: 5.6000%
```

[266]:
```
cnf_mat = confusion_matrix(y_train, y_pred, labels=[-1, 1])
sns.heatmap(cnf_mat, xticklabels=
        ['-1', '1'], yticklabels=['-1', '1'], annot=True, cmap = 'Blues',␣
 ↪fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

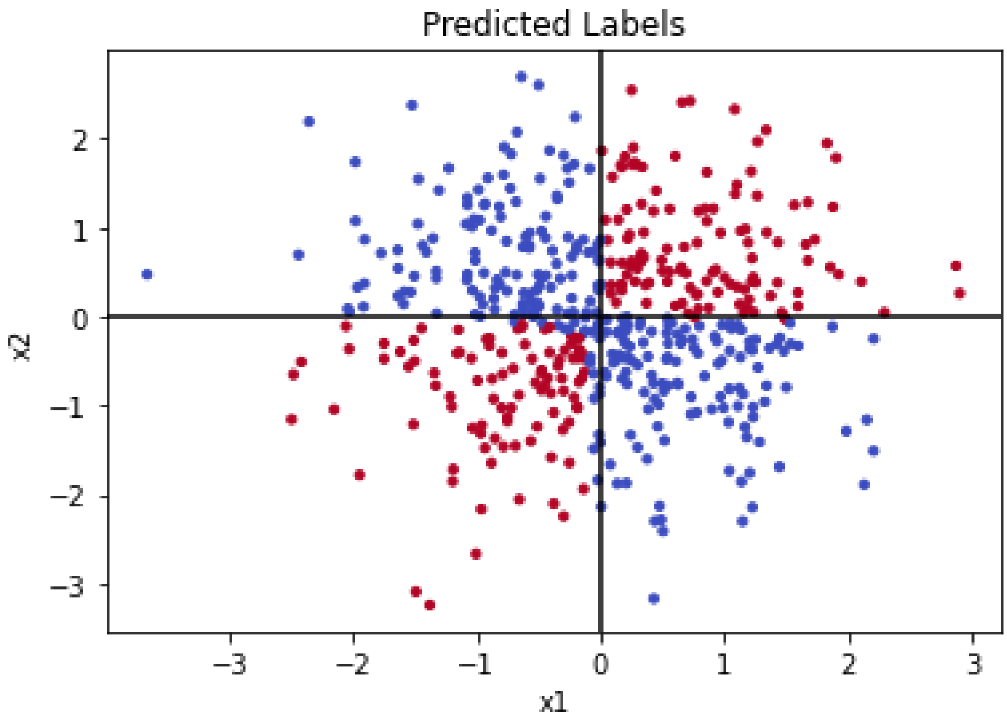


[267]:
```
def display_scatter_plot(x, y, title):
    fig, ax = plt.subplots()
    ax.scatter(x[:,0], x[:,1], c=y, s=10, cmap='coolwarm')
    ax.axhline(y=0, color='k')
    ax.axvline(x=0, color='k')
    ax.set_title(title)
    ax.set_xlabel('x1')
    ax.set_ylabel('x2')
    plt.show()
```

[268]:
```
# plotting the training data as it is
display_scatter_plot(x_train, y_train, 'Training Data')
```

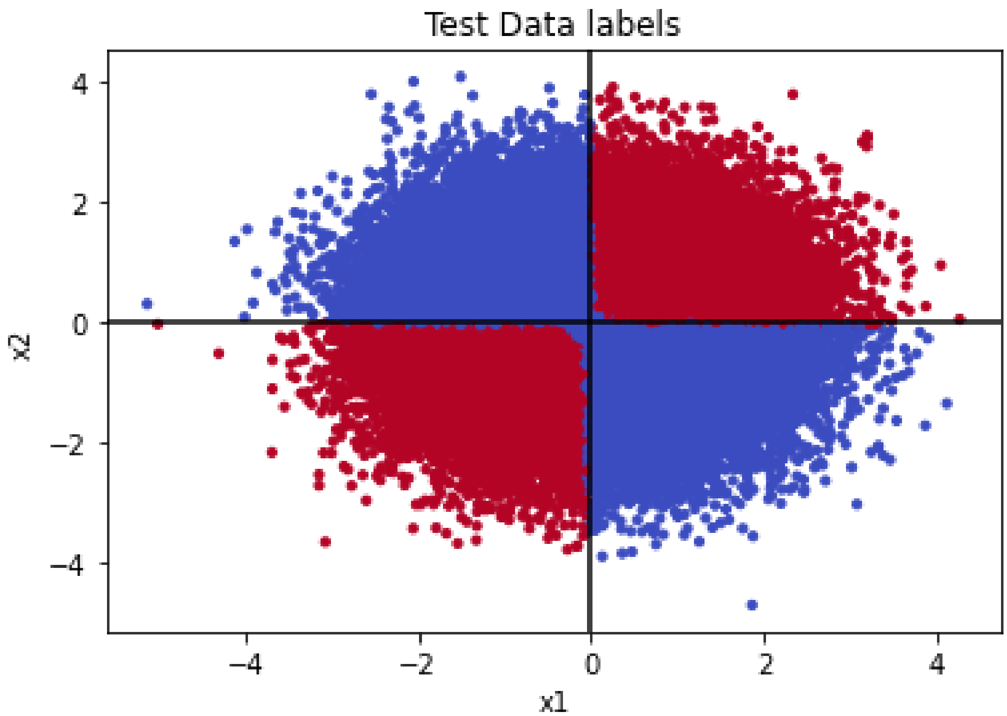


[269]:
```
# plotting the predicted labels
display_scatter_plot(x_train, y_pred, 'Predicted Labels')
```



[270]:
```
# Create a test set and test the model on it
test_pts = 100000
x_test = np.random.randn(test_pts, 2)
y_test = (x_test[:, 0] * x_test[:, 1] >= 0) * 2 - 1
A_test = find_coeff_matrix(x_test)
y_hat = A_test @ x_hat
y_hat = np.sign(y_hat).astype(np.int32)
```

[271]:
```
# plotting the test set labels
display_scatter_plot(x_test, y_hat, 'Test Data labels')
```



[272]:
```
print(x_hat)
```

```
[-0.01742263  0.0294122   0.01539744  0.6397916   0.00439876 -0.0037246 ]
```

Ashutosh Kumar Singh . 19CS30008

9. (a) The error rate and confusion matrix are shown above.

Error rate = 5.6000 %

(b) The regions are also shown above.
The regions are separated by a hyperbolic boundary.

(c) Yes, the second degree polynomial $g = x_1 x_2$ classifies the generated points with zero error.

The function $\hat{y} = \begin{cases} +1 & g(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$

classifies all points correctly.

The polynomial considered by us is:-

$$\tilde{f}(x) = \theta_1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_1 x_2 + \theta_5 x_1^2 + \theta_6 x_2^2$$

The parameter values obtained are:-

$\theta_1 = -0.0174$

$\theta_2 = 0.0294$

$\theta_3 = 0.0153$

$\boxed{\theta_4 = 0.6397}$

$\theta_5 = 0.0043$

$\theta_6 = -0.0037$

Thus, we can see that here too, the coefficient for $x_1 x_2 = \theta_4$ is significantly greater than all the other coefficients.

In fact, all the other coefficients are nearly zero, or very small, though not exactly zero.

Thus, on comparison with $g = x_1 x_2$, we can say that even for f, the most significant part in classification is indeed played by the term $x_1 x_2$.