tags: Machine Learning

Assignment 4

408410042 林靖紳

執行結果:

```
Training SVM with C = 100000, gamma = 100
Accuracy = 20% (20/100) (classification)
Training SVM with C = 100000, gamma = 1000
Accuracy = 13% (13/100) (classification)
Training SVM with C = 100000, gamma = 5000
Accuracy = 15% (15/100) (classification)
Training SVM with C = 150000, gamma = 100
Accuracy = 17% (17/100) (classification)
Training SVM with C = 150000, gamma = 1000
Accuracy = 26% (26/100) (classification)
Training SVM with C = 150000, gamma = 5000
Accuracy = 11% (11/100) (classification)
Training SVM with C = 200000, gamma = 100
Accuracy = 9% (9/100) (classification)
Training SVM with C = 200000, gamma = 1000
Accuracy = 4% (4/100) (classification)
Training SVM with C = 200000, gamma = 5000
Accuracy = 23% (23/100) (classification)
-----------------------------------------------------
Training SVM with best parameters: C = 150000, gamma = 1000, accuracy = 26.0%
-----------------------------------------------------
Evaluating SVM with best parameters and scaling
Training SVM with C = 150000, gamma = 1000
Accuracy = 7% (7/100) (classification)
With scaling, accuracy = 7.00%
-----------------------------------------------------
Evaluating SVM with best parameters without scaling
Training SVM with C = 150000, gamma = 1000
Accuracy = 0% (0/100) (classification)
Without scaling, accuracy = 0.00%
```

程式碼架構:

程式碼使用LIBSVM來訓練一個支援向量機(SVM)模型，並使用5-fold交叉驗證找到最佳的超參數C和gamma值，從而使得SVM模型最佳地適應數據。

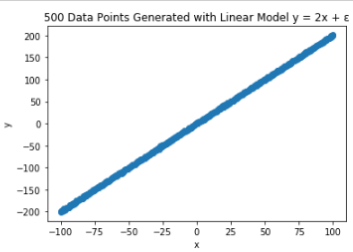
該程式碼中包含了以下功能：

1. 生成一個線性模型及對應的數據集，並將數據格式化為LIBSVM所需的格式。

In [2]:
```
# Define the linear model
def linear_model(x):
    return 2*x + np.random.normal(0, 1)

# Generate 500 data points with equal spacing in the range [-100, 100]
x = np.linspace(-100, 100, 500)
y = np.array([linear_model(xi) for xi in x])

# Plot the data
plt.scatter(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('500 Data Points Generated with Linear Model y = 2x + ε')
plt.show()
```



2. 設定交叉驗證的折數及每折的數據集大小，然後進行數據集的隨機切分。
3. 定義一個縮放數據的函數，用於將特徵值縮放到0到1之間。

In [4]:
```python
def split_data(data):
    np.random.shuffle(data)
    num_samples = len(data)
    num_train = int(num_samples * 0.8)
    train_data = data[:num_train]
    test_data = data[num_train:]
    return train_data, test_data


def scale_data(data, scaler):
    X = [d['features'] for d in data]
    X_scaled = scaler.fit_transform(X)
    scaled_data = []
    for i in range(len(data)):
        scaled_data.append({'label': data[i]['label'], 'features': X_scaled[i]})
    return scaled_data
```

In [5]:
```python
def train_and_evaluate_svm(data, c, gamma, scaling=True):
    # Split data into training and testing sets
    train_data, test_data = split_data(data)
```

4. 定義一個訓練和評估SVM模型的函數，其中訓練集和測試集由交叉驗證生成，模型訓練通過LIBSVM庫實現，模型的準確率通過測試集進行評估。

```python
# Train and evaluate SVM models for all combinations of C and gamma

print('Training SVM with C = {}, gamma = {}'.format(c, gamma))

# Write the training and testing data to files
with open('train.dat', 'w') as f:
    for d in train_data:
        f.write('{} {}\n'.format(d['label'], ' '.join('{}:{}'.format(i+1, x) for i, x in enumerate(d['features']))))
with open('test.dat', 'w') as f:
    for d in test_data:
        f.write('{} {}\n'.format(d['label'], ' '.join('{}:{}'.format(i+1, x) for i, x in enumerate(d['features']))))

# Load the training and testing data from the files
y_train, x_train = svm_read_problem('train.dat')
y_test, x_test = svm_read_problem('test.dat')

# Train the SVM model
model = svm_train(y_train, x_train, '-s 0 -t 2 -c {} -g {}'.format(c, gamma))
```

5. 定義了一個超參數C和gamma的參數網格，從而可以通過交叉驗證找到最佳的C和gamma參數值。
6. 最後，該程式碼使用最佳的C和gamma值來訓練SVM模型，並進行縮放和未縮放的模型評估，以比較模型的準確率。

```python
# Define the parameter grids for C and gamma
C_values = [100000, 150000, 200000]
gamma_values = [100, 1000, 5000]


# Find the best parameters using 5-fold cross-validation
best_accuracy = 0

for c in C_values:
    for gamma in gamma_values:
        accuracy = train_and_evaluate_svm(data, c, gamma)
        if accuracy > best_accuracy:
            best_c = c
            best_gamma = gamma
            best_accuracy = accuracy
            #best_params.append((best_c, best_gamma, best_accuracy))


print("-----------------------------------------------------")
# Train the SVM model with the best parameters
print('Training SVM with best parameters: C = {}, gamma = {}, accuracy = {}%'.format(best_c, best_gamma, best_accuracy))
#train_and_evaluate_svm(data, best_c, best_gamma)
print("-----------------------------------------------------")
# Evaluate the SVM with and without scaling
print('Evaluating SVM with best parameters and scaling')
accuracy_with_scaling = train_and_evaluate_svm(data, best_c, best_gamma)
print('With scaling, accuracy = {:.2f}%'.format(accuracy_with_scaling))
print("-----------------------------------------------------")
print('Evaluating SVM with best parameters without scaling')
accuracy_without_scaling = train_and_evaluate_svm(data, best_c, best_gamma, scaling=False)
print('Without scaling, accuracy = {:.2f}%'.format(accuracy_without_scaling))
```

結果討論

- 在支持向量機（SVM）中，C值和Gamma參數是用來調整模型的超參數。
    - C值是SVM的一個正則化參數，用於控制SVM對於錯誤樣本的容忍度。當C值較大時，SVM會儘可能地讓所有的樣本都分對，因此會出現過度擬合的情況；當C值較小時，SVM會對錯誤樣本的容忍度較高，因此可能會導致一些樣本分類錯誤。
    - Gamma參數是SVM中的一個核函數參數，用於控制SVM的決策邊界的彎曲度。當Gamma值較大時，SVM的決策邊界會比較複雜，可能會導致過度擬合的情況；當Gamma值較小時，SVM的決策邊界會比較簡單，可能會導致欠擬合的情況。

- 從結果可以觀察到經過 scaling 的資料比沒有 Scaling 的資料表現來的更好，這表示經過縮放後的資料可以更好的幫助分類器找到最佳的邊界。
    - 推測原因: 在沒有經過特徵縮放的情況下，不同特徵的取值範圍可能相差甚遠，這樣就會導致在不同特徵下計算距離時對結果影響的程度也不一樣。
    - 透過特徵縮放，可以將所有特徵的範圍固定在一個區間內，使每個特徵在計算距離時對於結果的影響程度相對均衡，提高模型的穩定性。