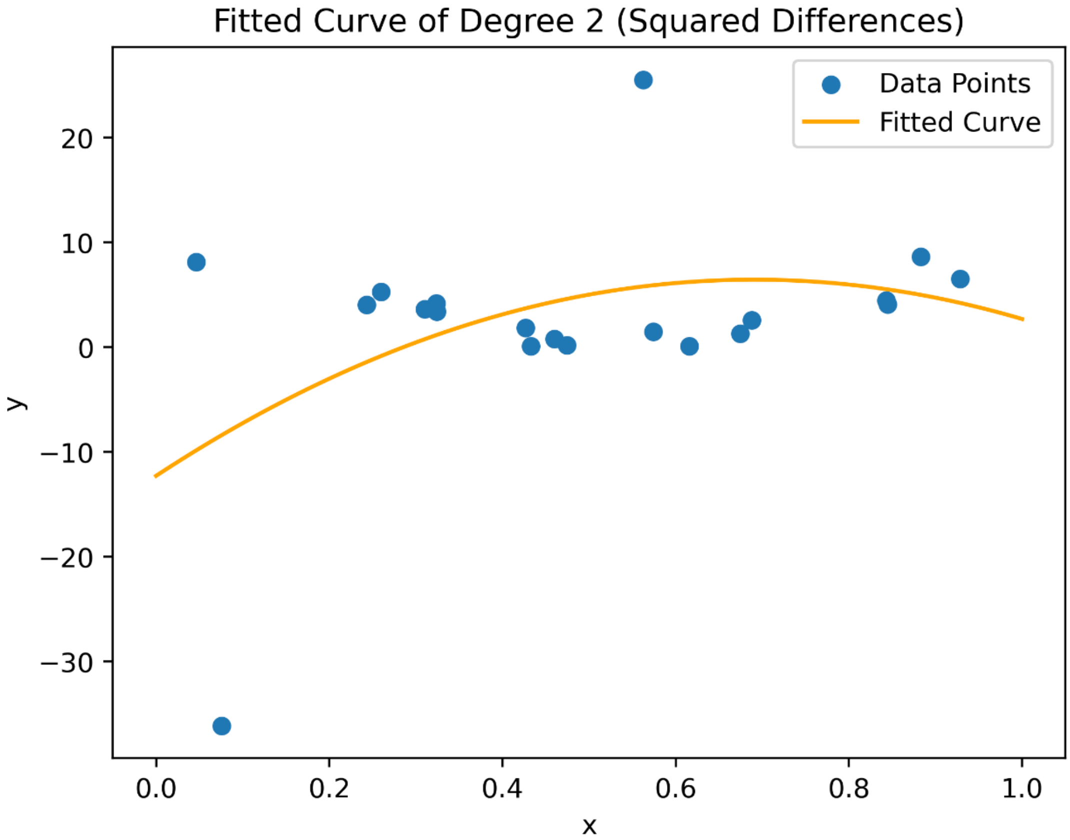
ENGN 2520 Pattern Recognition and Machine Learning

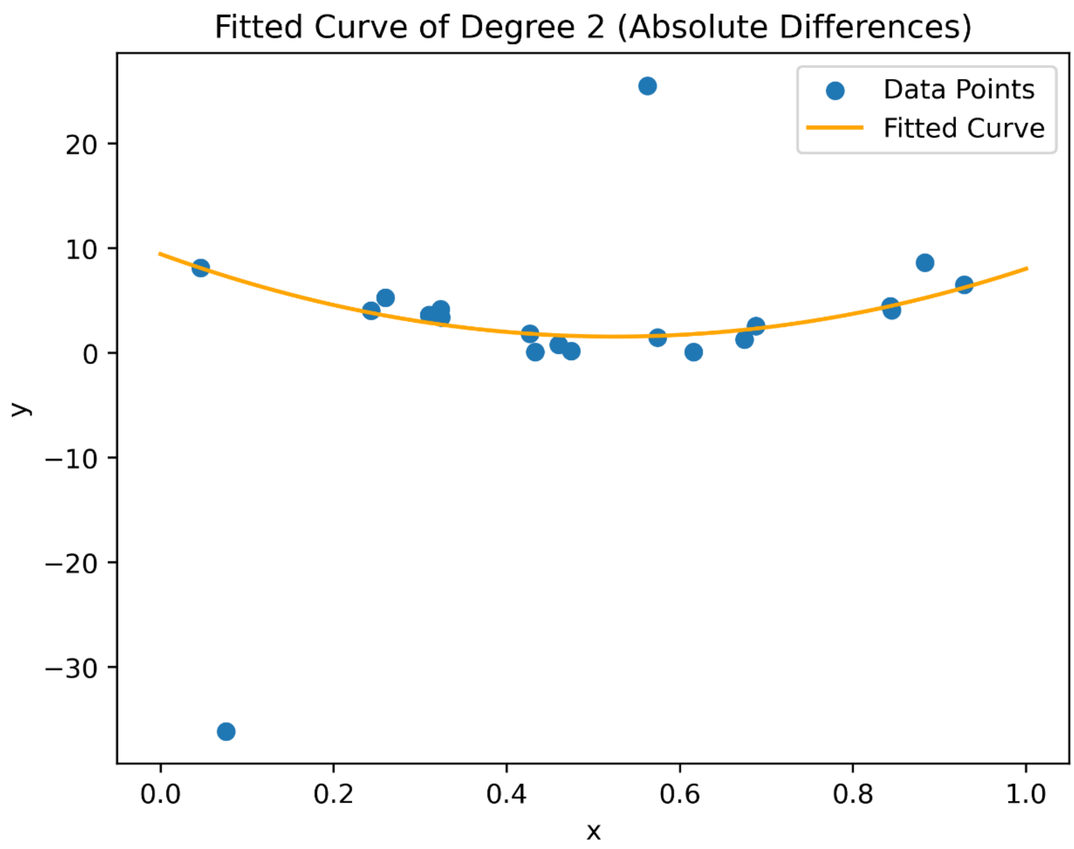
Homework 2

Zhuo Wang

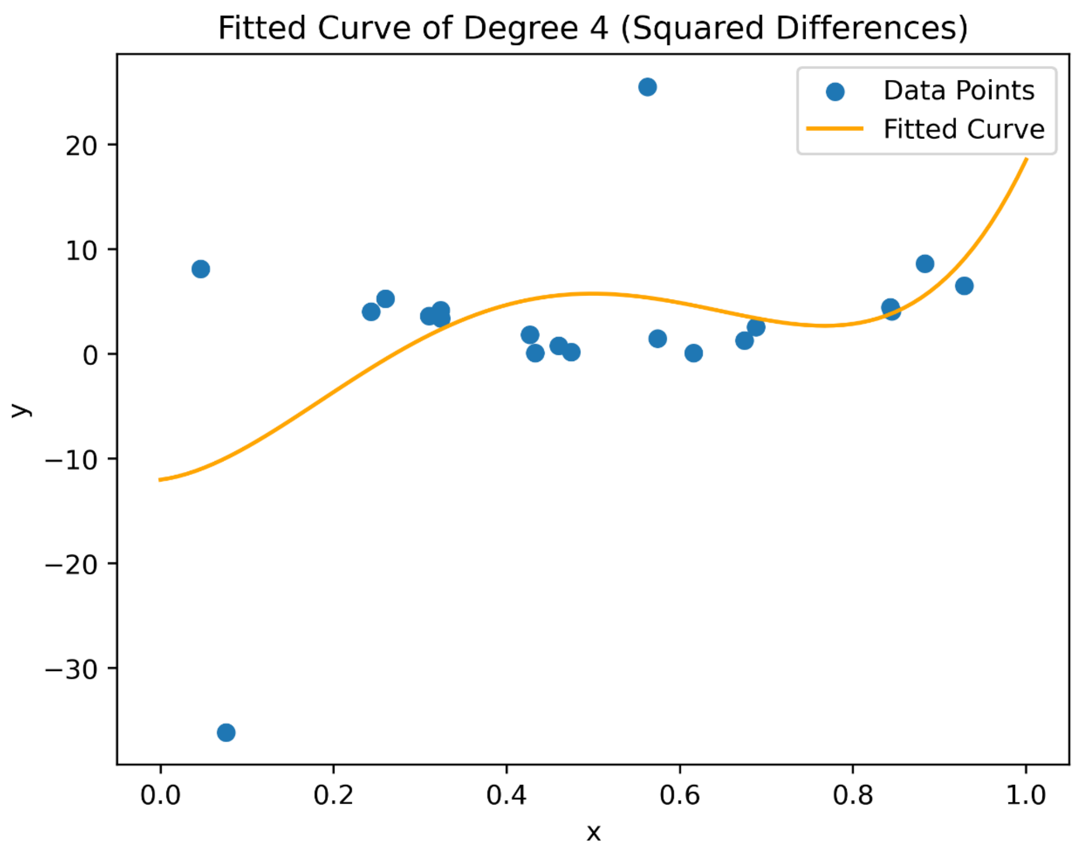
Problem 1

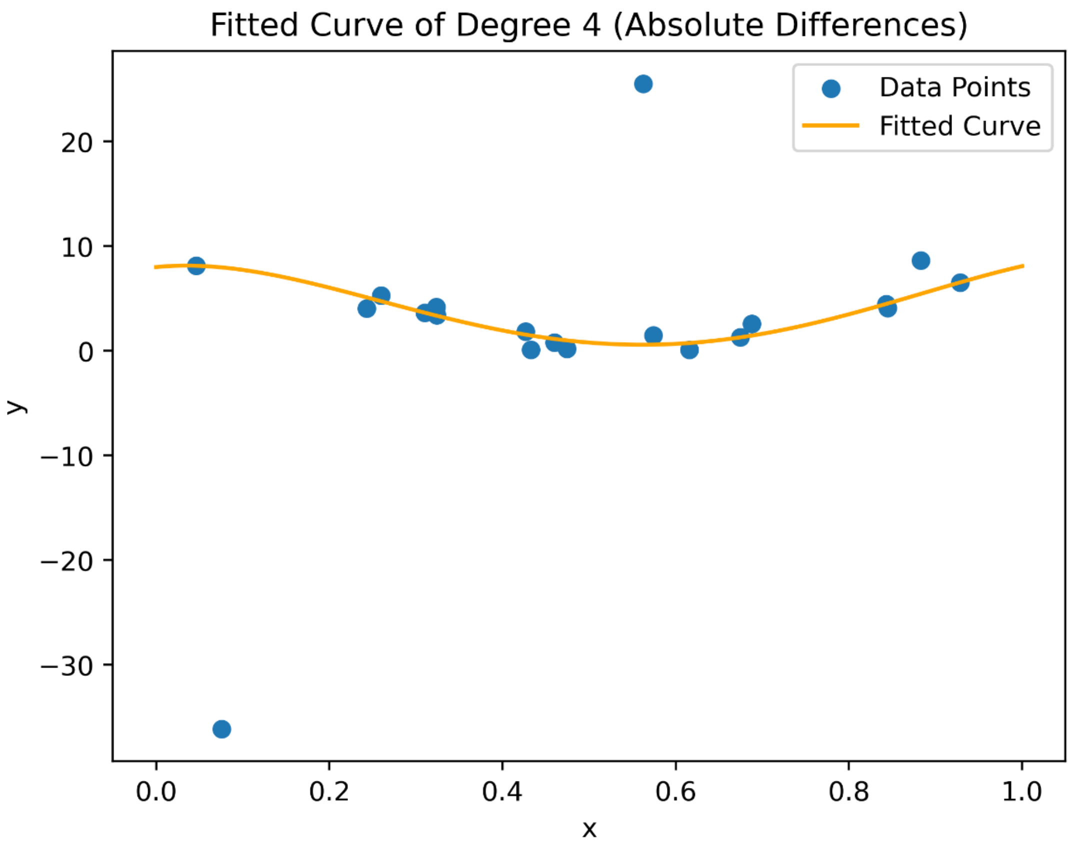
(a)




Fitted Curve of Degree 2 (Absolute Differences)
Data Points
Fitted Curve
20
10
0
−10
−20
−30
y
0.0
0.2
0.4
0.6
0.8
1.0
x

(b)


Fitted Curve of Degree 4 (Squared Differences)
Data Points
Fitted Curve
20
10
0
−10
−20
−30
y
0.0
0.2
0.4
0.6
0.8
1.0
x



(c)

It can be seen from the results that the effect of regression based on the sum of absolute differences is better than that based on the sum of squared differences. Because sum of squared differences regression tends to fit most data points, it is more sensitive to outliers. The sum of absolute difference is more robust for outliers because it weights the error evenly and does not depend on the size of the error.

When using degree 2 polynomials, regression based on the sum of squared differences can suffer from outliers in the training set, causing the polynomial curve to attempt to fit these points at the expense of fitting the overall dataset.

When using degree 4 polynomials, the model becomes more complex due to the increase in the order of the polynomials and is more susceptible to outliers in the training data.

Problem 2

$$\min \left(\sum_{i=1}^{N} |y_i - W^T \phi(x_i)| \right) + \lambda ||W||_1$$

$$U_i = \max(0,\ y_i - W^T \phi(x_i))$$

$$V_i = \max(0,\ -y_i + W^T \phi(x_i)) \qquad |W_j^2| = z_j$$

$$\min \sum_{i=1}^{N} (U_i + V_i) + \lambda \sum_{j=1}^{m} |W_j^2|$$

constraints

$$U_i - V_i = y_i - W^T \phi(x_i)$$

$$U_i, V_i \geq 0$$

$$z_j \geq W_j$$

$$z_j \geq -W_j$$

$$z_j \geq 0$$

Source Code

```
import numpy as np
from scipy.io import loadmat
from scipy.optimize import linprog
import matplotlib.pyplot as plt

def polynomial_features(x, degree):
    a = []

    for i in range(degree + 1):
        feature = []
        for xi in x:
            feature.append(xi ** i)

        a.append(feature)

    a = np.array(a).T

    a = np.squeeze(a)

    return a

def fit_polynomial_regression(x, y, degree):
    X = polynomial_features(x, degree)
    w = np.linalg.solve(X.T @ X, X.T @ y)

    return w


def calculate_loss(x, y, w):
    y_pred = np.dot(polynomial_features(x, len(w) - 1), w)
    loss = np.mean((y_pred - y) ** 2)
    return loss


def robust_regression(phi, X, y):
    M = phi.shape[1]
    N = len(X)

    c = np.concatenate((np.zeros(M), np.ones(N)))

    A_ub = []
    b_ub = []
```

```
    for i in range(N):
        a1 = np.concatenate((phi[i], np.zeros(N)))
        a1[M + i] = -1
        A_ub.append(a1)
        b_ub.append(y[i])

    for i in range(N):
        a2 = np.concatenate((-phi[i], np.zeros(N)))
        a2[M + i] = -1
        A_ub.append(a2)
        b_ub.append(-y[i])

    bounds = (None, None)


    result = linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=bounds,
method='highs')

    w = result.x[:M]

    return w




x_train = loadmat('Xtrain.mat')['Xtrain']
y_train = loadmat('Ytrain.mat')['Ytrain']


ws = fit_polynomial_regression(x_train, y_train, 2)
xp = np.linspace(0, 1, 100)
yp = np.dot(polynomial_features(xp, len(ws) - 1), ws)

plt.scatter(x_train, y_train, label='Data Points')
plt.plot(xp, yp,color='orange', label='Fitted Curve')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Fitted Curve of Degree 2 (Squared Differences)')
plt.legend()
plt.savefig('2sd.png', dpi=400, bbox_inches='tight')
plt.show()
```

```
wa = robust_regression(polynomial_features(x_train, 2), x_train,
y_train)

xp = np.linspace(0, 1, 100)
yp = np.dot(polynomial_features(xp, len(wa) - 1), wa)

plt.scatter(x_train, y_train, label='Data Points')
plt.plot(xp, yp,color='orange', label='Fitted Curve')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Fitted Curve of Degree 2 (Absolute Differences)')
plt.legend()
plt.savefig('2ad.png', dpi=400, bbox_inches='tight')
plt.show()


ws4 = fit_polynomial_regression(x_train, y_train, 4)
xp = np.linspace(0, 1, 100)
yp = np.dot(polynomial_features(xp, len(ws4) - 1), ws4)

plt.scatter(x_train, y_train, label='Data Points')
plt.plot(xp, yp,color='orange', label='Fitted Curve')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Fitted Curve of Degree 4 (Squared Differences)')
plt.legend()
plt.savefig('4sd.png', dpi=400, bbox_inches='tight')
plt.show()





wa4 = robust_regression(polynomial_features(x_train, 4), x_train,
y_train)

xp = np.linspace(0, 1, 100)
yp = np.dot(polynomial_features(xp, len(wa4) - 1), wa4)

plt.scatter(x_train, y_train, label='Data Points')
plt.plot(xp, yp,color='orange', label='Fitted Curve')
```

```
plt.xlabel('x')
plt.ylabel('y')
plt.title('Fitted Curve of Degree 4 (Absolute Differences)')
plt.legend()
plt.savefig('4ad.png', dpi=400, bbox_inches='tight')
plt.show()
```