ENGN 2520 Pattern Recognition and Machine Learning

Homework 1

Zhuo Wang

Problem 1

(a)

$$p(T_A, T_B|w) = \prod_{i=1}^{N}{}_{(x,y)\in T_A} \frac{1}{\sqrt{2\pi}\sigma_A} \exp -\frac{(y_i - w^T\phi(x_i))^2}{2\sigma_A^2} \times$$

$$\prod_{i=1}^{N}{}_{(x,y)\in T_B} \frac{1}{\sqrt{2\pi}\sigma_B} \exp -\frac{(y_i - w^T\phi(x_i))^2}{2\sigma_B^2}$$

$$-\log(p(T_A, T_B|w) = \frac{1}{2\sigma_A^2}\left(\sum_{i=1}^{N}{}_{(x,y)\in T_A}(w^T\phi(x_i) - y_i)^2\right) + \log\sqrt{2\pi}\sigma_A +$$

$$\frac{1}{2\sigma_B^2}\left(\sum_{i=1}^{N}{}_{(x,y)\in T_B}(w^T\phi(x_i) - y_i)^2\right) + \log\sqrt{2\pi}\sigma_B$$

$$\arg\min_w \left\{ \sum_{(x,y)\in T_A} \frac{(w^T\phi(x_i) - y_i)^2}{2\sigma_A^2} + \sum_{(x,y)\in T_B} \frac{(w^T\phi(x_i) - y_i)^2}{2\sigma_B^2} \right\}$$

(b)

$$\nabla_w \left(\sum_{(x,y)\in T_A} \frac{(w\phi(x_i) - y_i)^2}{2\sigma_A^2} + \sum_{(x,y)\in T_B} \frac{(w\phi(x_i) - y_i)^2}{2\sigma_B^2} \right) = 0$$

(c)

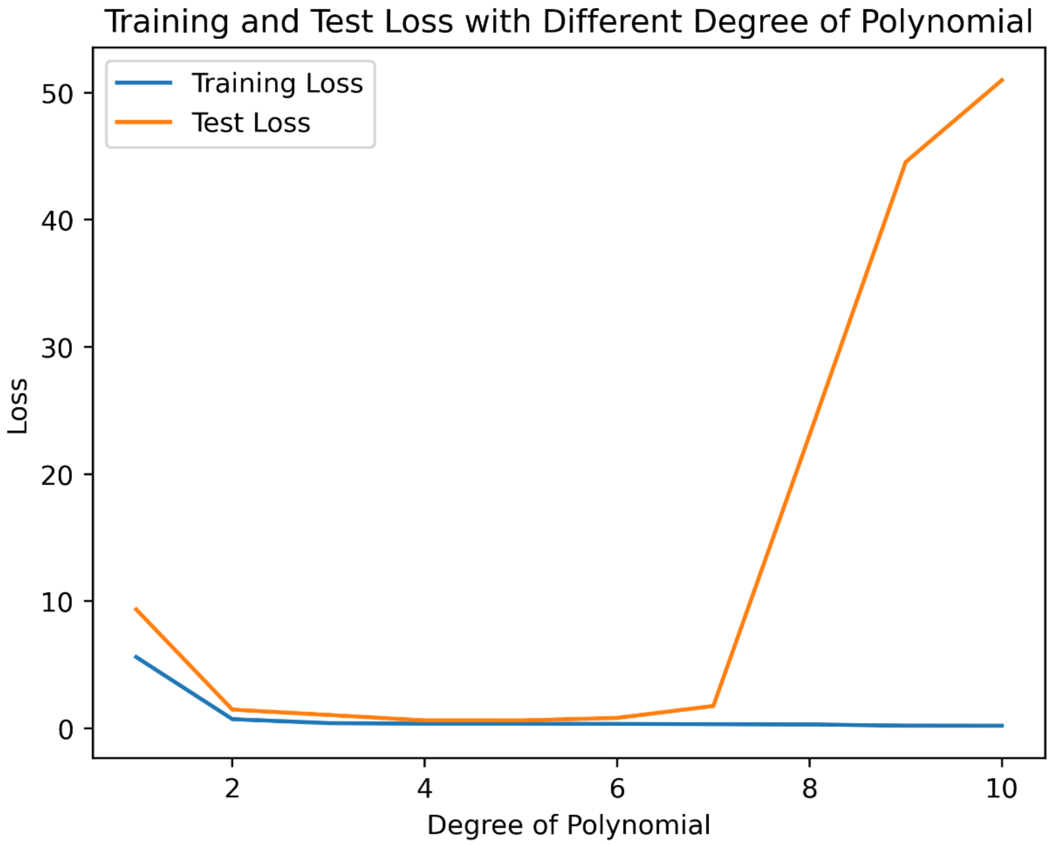
According to this mathematical model, it makes sense to combine the two sets of data

TA and TB, but the weights given to each set of measurements are changed in accordance with the variance of the errors σ_A^2 and σ_B^2. In comparison to the data in TA, the higher variance in TB suggests fewer accurate measurements, which means they have a smaller impact on the calculation of ω.

(d)
We can estimate σ_A and σ_B from the data if they are unknown. Estimating σ_A^2 and σ_B^2 using the sample variance of the residual. So we can calculate ω after estimating these variances.

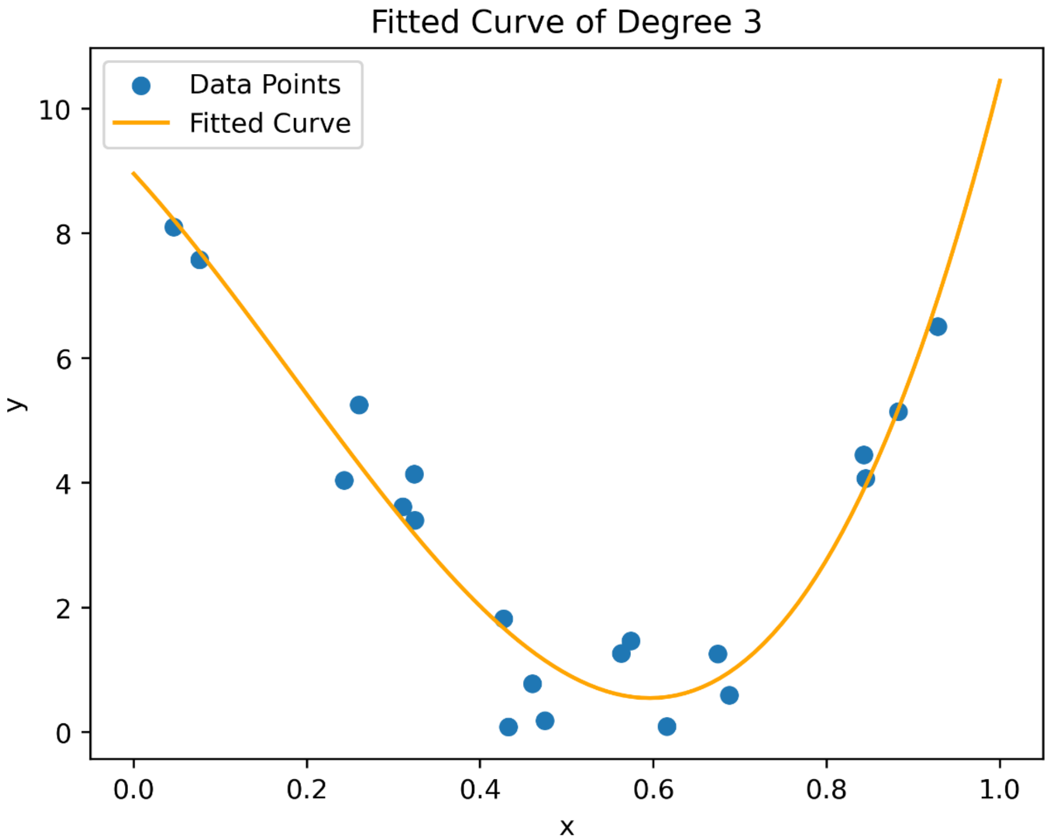
Problem 2

(a)

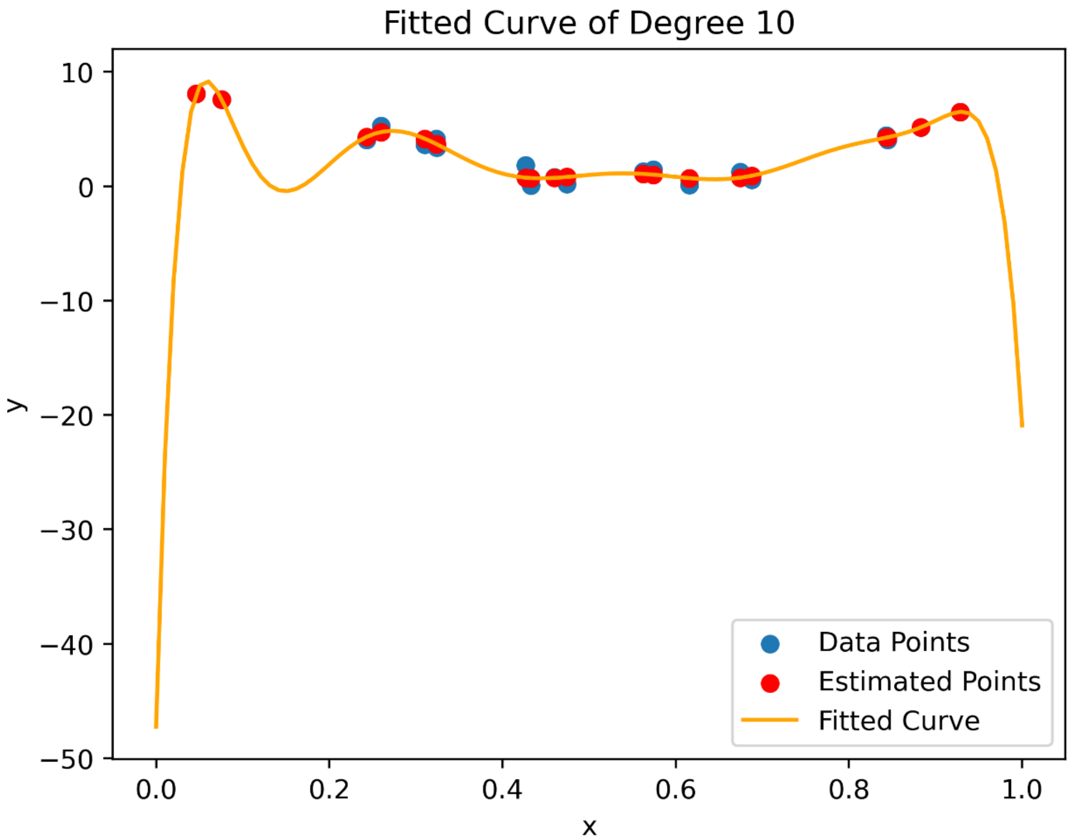


Significant overfitting can be observed from the degree 7 to the degree 10.

(b)


Fitted Curve of Degree 3
Data Points
Fitted Curve
10
8
6
4
2
0
y
0.0
0.2
0.4
0.6
0.8
1.0
x

(c)


Fitted Curve of Degree 10
10
0
−10
−20
−30
−40
−50
y
Data Points
Estimated Points
Fitted Curve
0.0
0.2
0.4
0.6
0.8
1.0
x

(d)

```python
import numpy as np
from scipy.io import loadmat
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

x_train = loadmat('Xtrain.mat')['Xtrain']
y_train = loadmat('Ytrain.mat')['Ytrain']
x_test = loadmat('Xtest.mat')['Xtest']
y_test = loadmat('Ytest.mat')['Ytest']

plt.scatter(x_train, y_train)
plt.show()

degrees = range(1, 11)
train_loss = []
test_loss = []

for degree in degrees:
    w = fit_polynomial_regression(x_train, y_train, degree)
    train_loss.append(calculate_loss(x_train, y_train, w))
    test_loss.append(calculate_loss(x_test, y_test, w))

plt.plot(degrees, train_loss, label='Training Loss')
plt.plot(degrees, test_loss, label='Test Loss')
plt.xlabel('Degree of Polynomial')
plt.ylabel('Loss')
plt.title('Training and Test Loss with Different Degree of Polynomial')
plt.legend()
plt.savefig('a.png', dpi=400, bbox_inches='tight')
plt.show()

wb = fit_polynomial_regression(x_train, y_train, 3)
xp = np.linspace(0, 1, 100)
yp = np.dot(polynomial_features(xp, len(wb) - 1), wb)

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 3')
plt.legend()
plt.savefig('b.png', dpi=400, bbox_inches='tight')
plt.show()
```

```
wc = fit_polynomial_regression(x_train, y_train, 10)
xpc = np.linspace(0, 1, 100)
ypc = np.dot(polynomial_features(xpc, len(wc) - 1), wc)

yd = np.dot(polynomial_features(x_train, len(wc) - 1), wc)

plt.scatter(x_train, y_train, label='Data Points')
plt.scatter(x_train, yd, color='red', label='Estimated Points')
plt.plot(xpc, ypc, color='orange', label='Fitted Curve')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Fitted Curve of Degree 10')
plt.legend()
plt.savefig('c.png', dpi=400, bbox_inches='tight')
plt.show()
```