Iowa State University

EE 526X

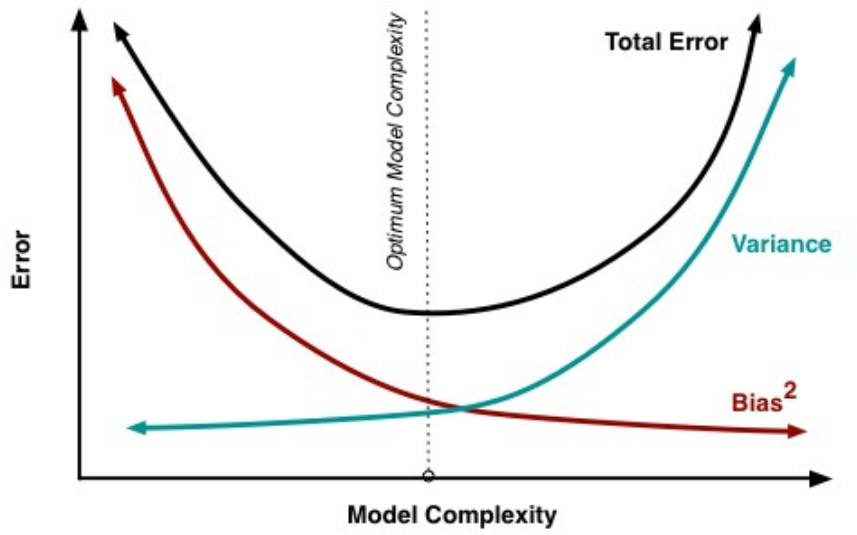
Assignment 1

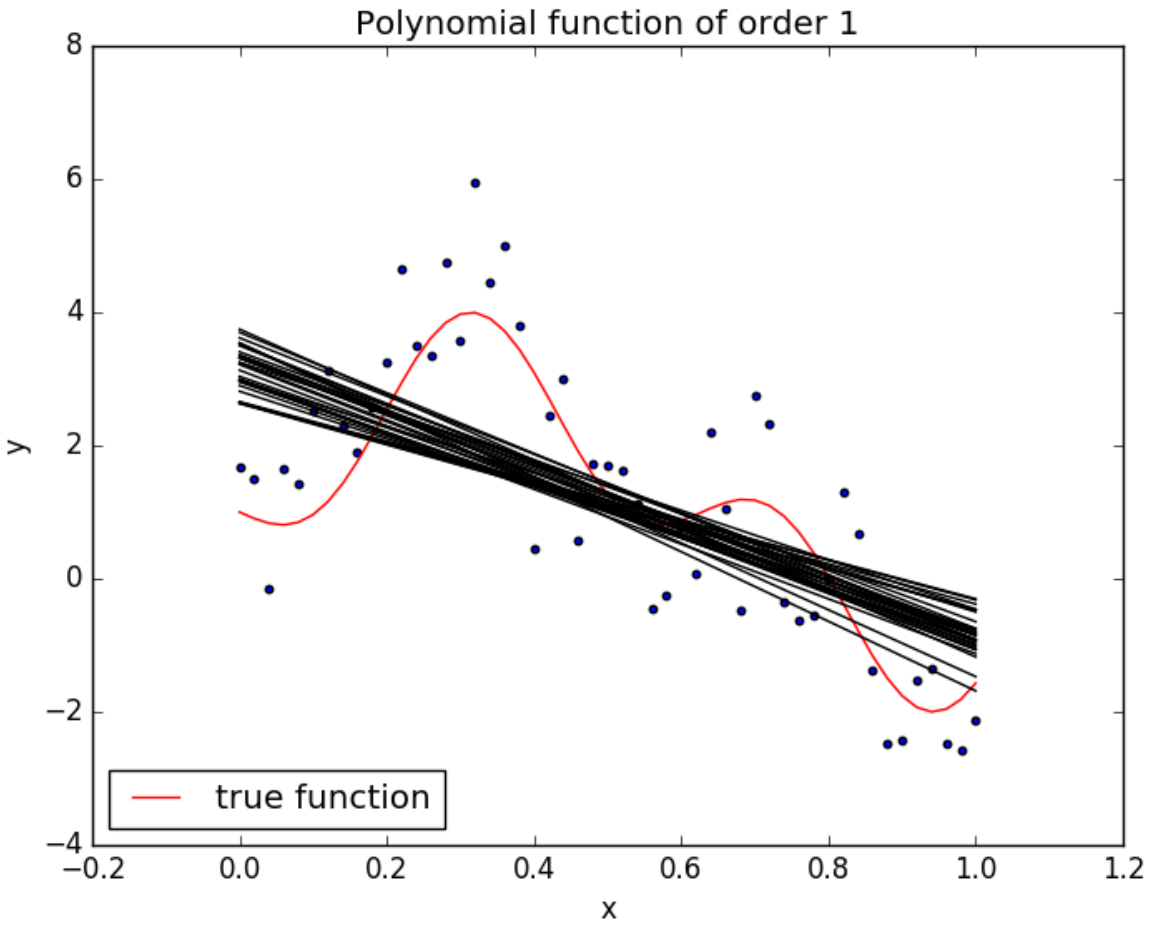
Eliska Kloberdanz

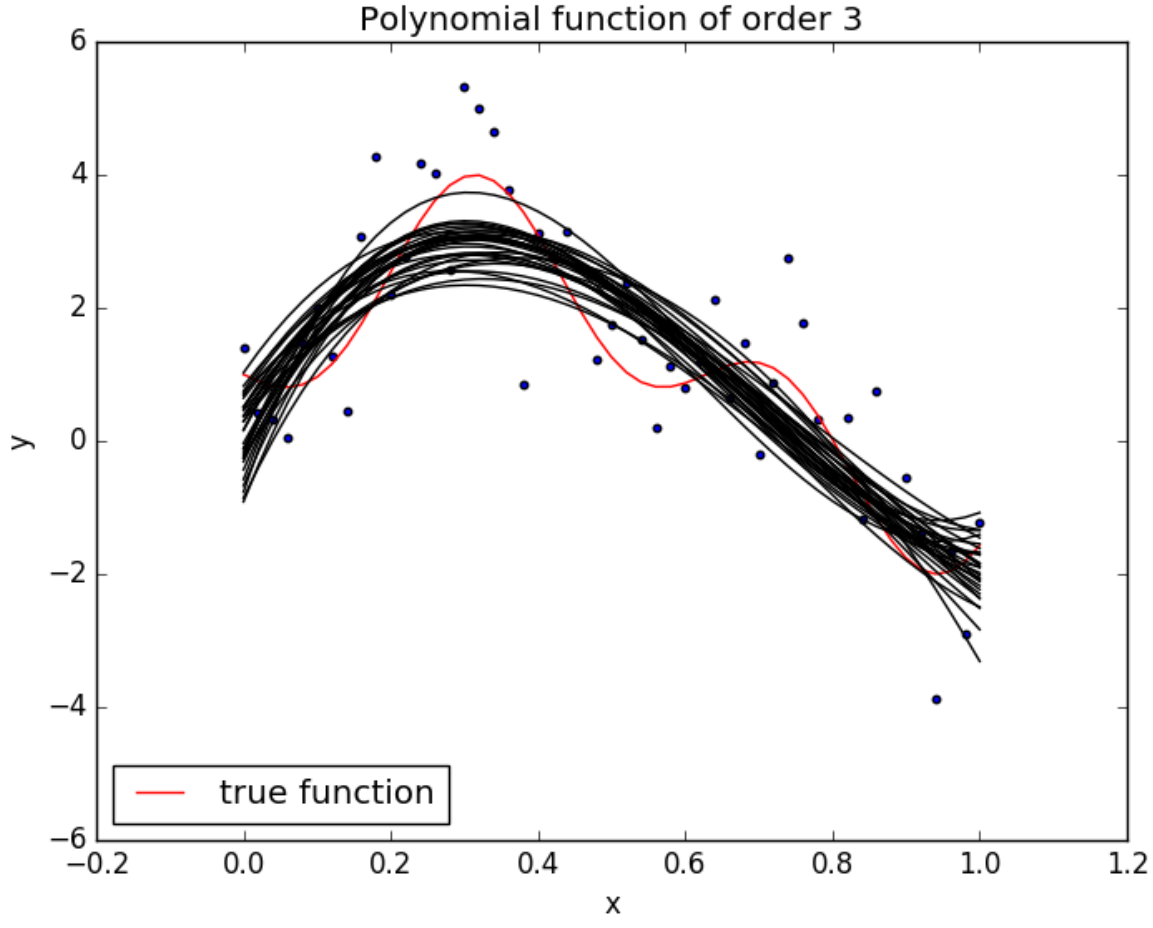
September 23, 2019

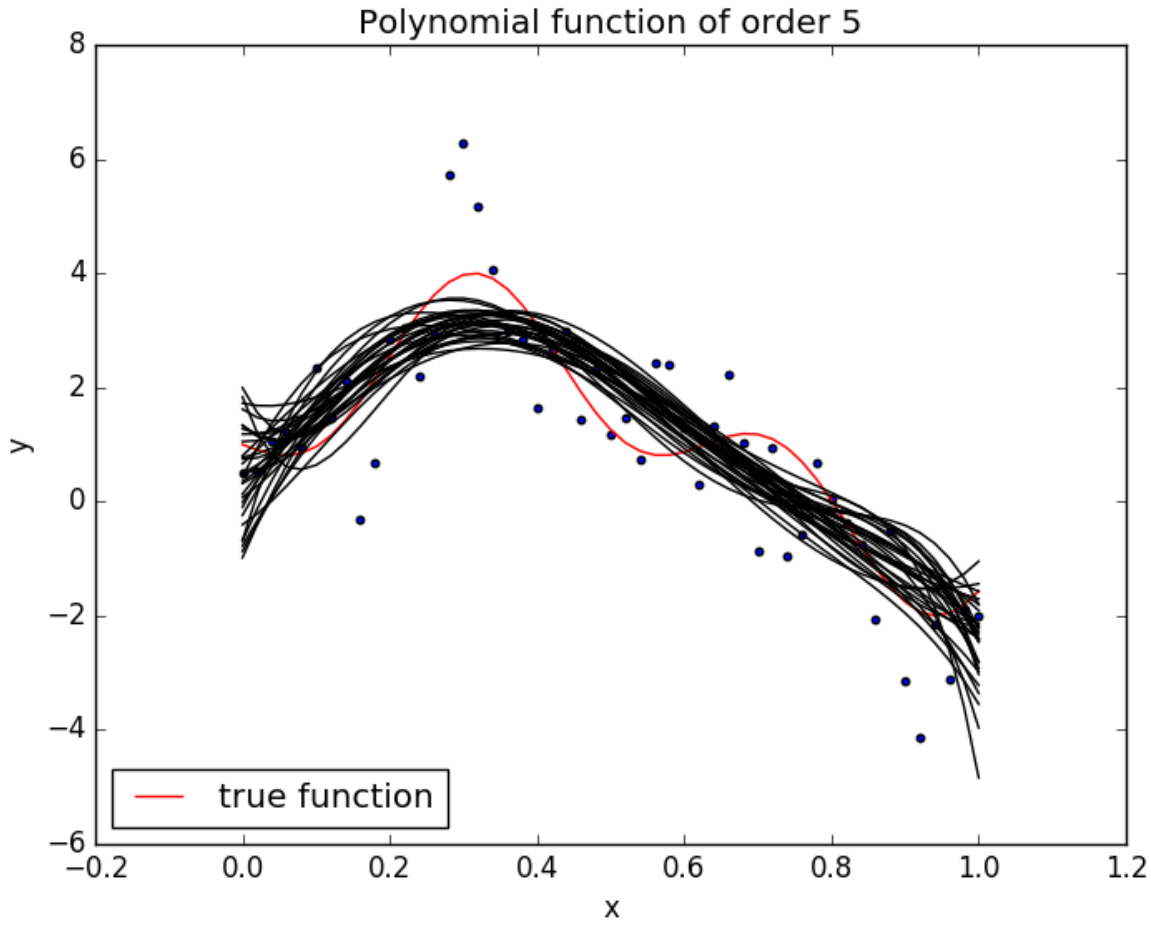
Problem 1

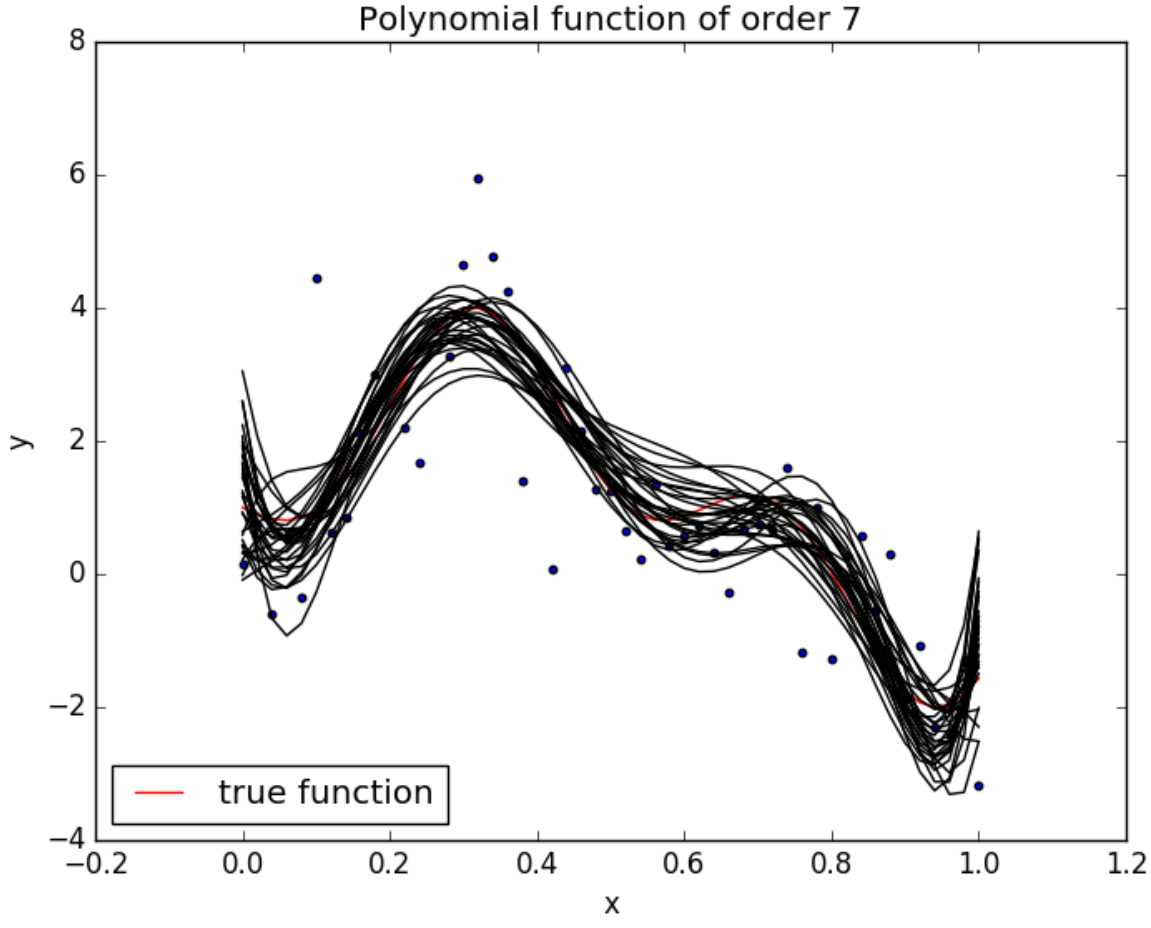
Please see my polynomial fitting function implementation by running: python3 assignment1p1.py. In that script I generated 51 equally spaced x values between 0 and 1 and also 51 y values such that $y_i = f(x_i) + e_i$ for i=0, 1, …, 50, where e_i iid standard Gaussian random variables. Then I fit a polynomial of order k = 1, 3, 5, 7, 9, 11 by minimizing the residual sums of squares. To do that, I created the Beta matrix $\beta' = (X^TX)^{-1}X^Ty$, the X matrix, and predicted y by $y = X\beta + e$. I repeated this experiment (generating y values and fitting a polynomial) 30 times for each k. This experiment illustrates the trade-off between bias and variance. As the polynomial order increases, the bias decreases – we get a closer fit to the ground truth function. However, while the bias decreases with increasing model capacity, the variance increases. This means that while the shape of the fitted polynomials is more similar to the ground truth function, the fitted polynomials vary more (are further away) from the their average. Based on the graphs below, polynomial of order 7 seems to achieve the smallest total error – the best trade-off between bias and variance.

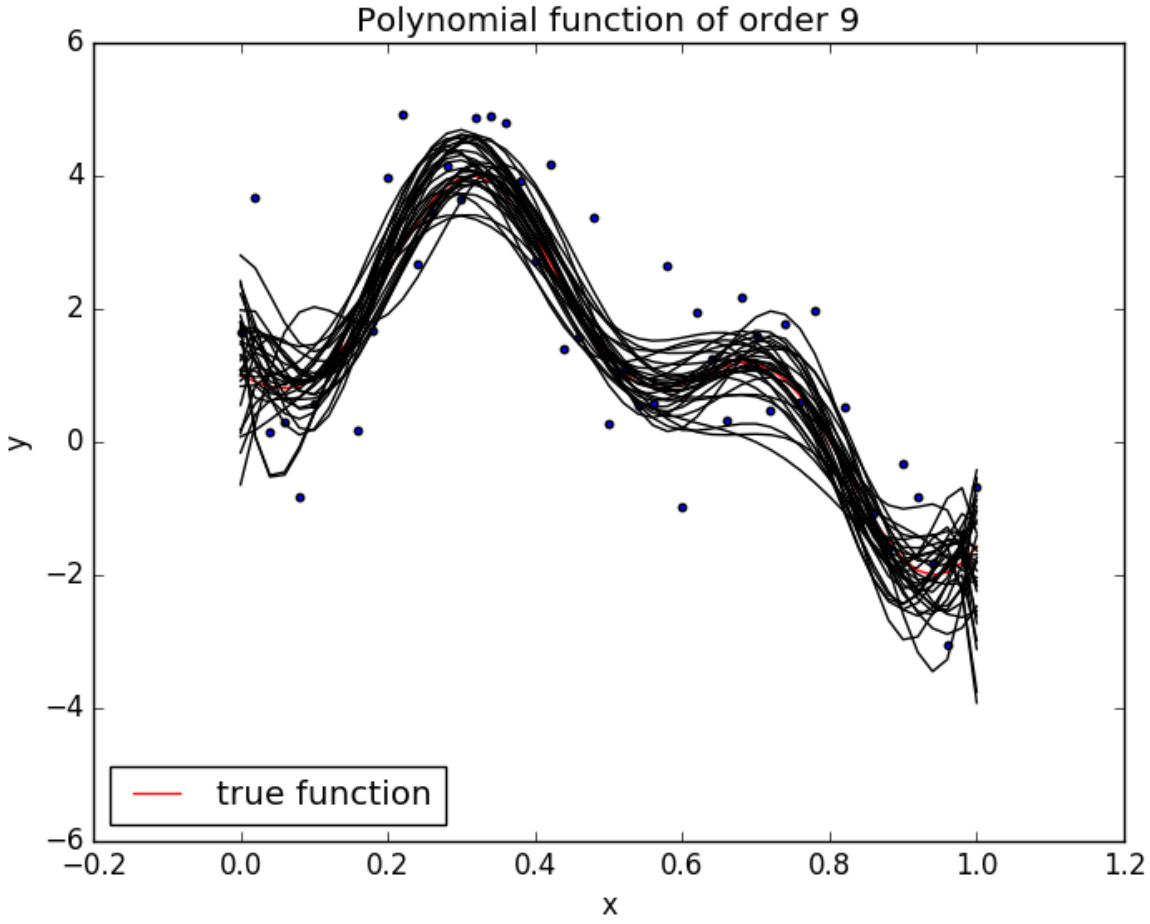


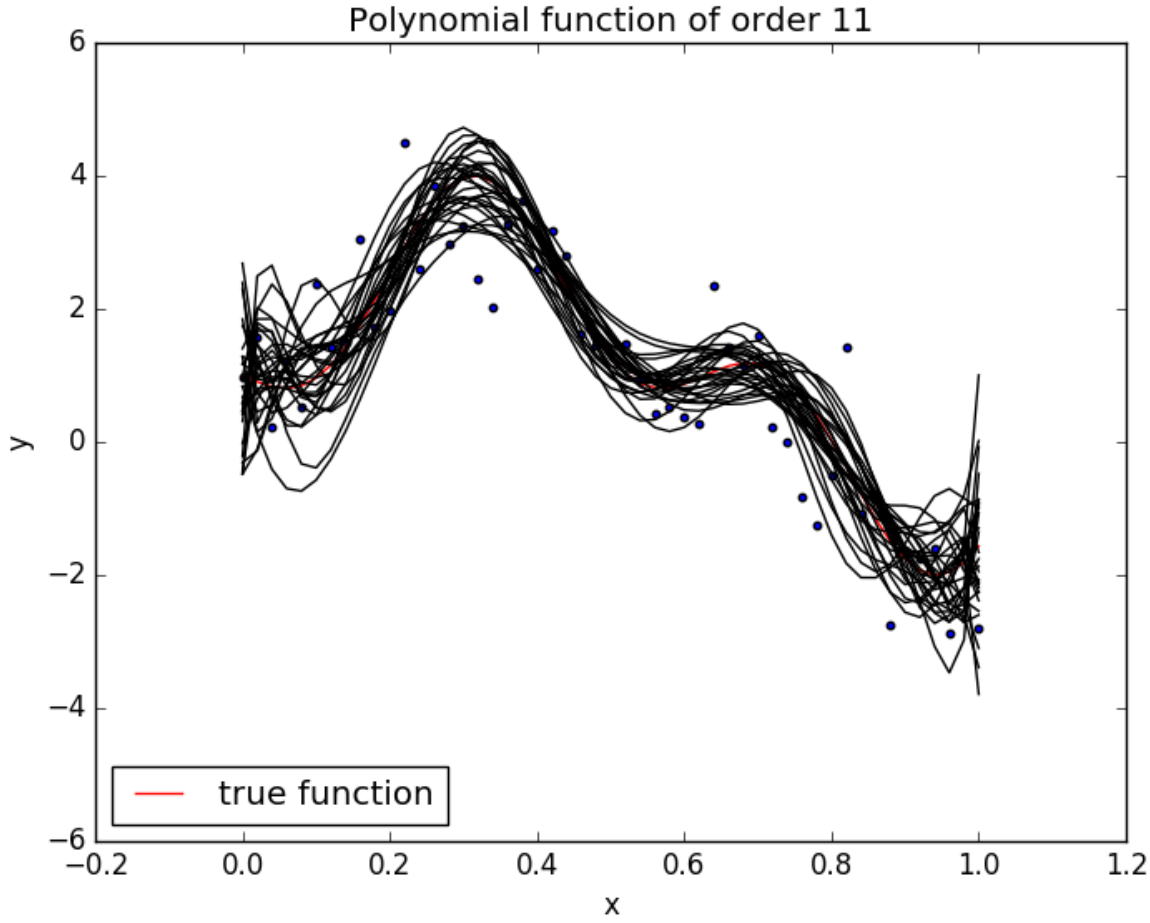

Polynomial function of order 1
8
6
4
2
0
−2
−4
y
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
x
true function


Polynomial function of order 3
6
4
2
0
−2
−4
−6
y
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
x
true function

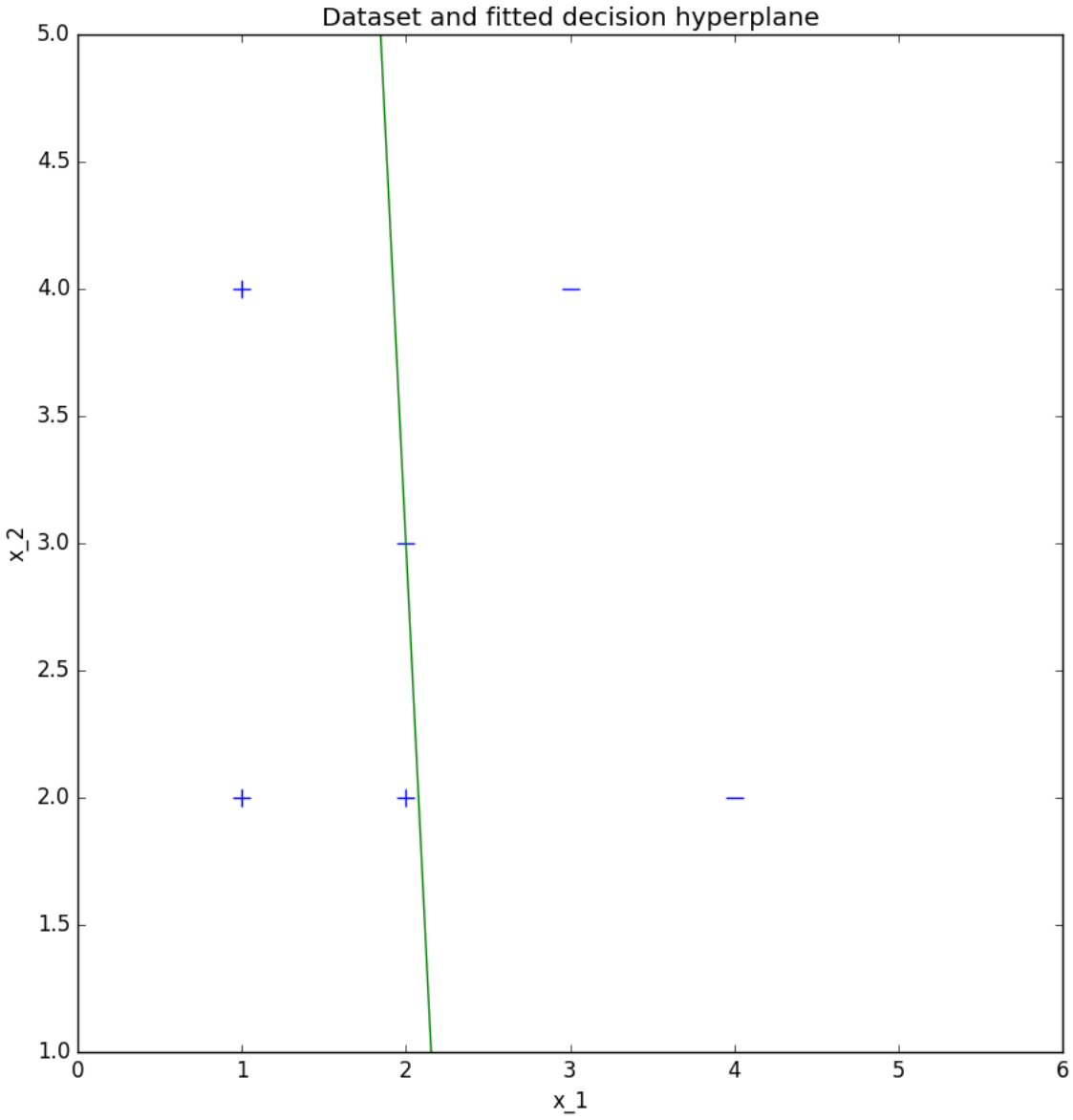

Polynomial function of order 5
8
6
4
2
0
−2
−4
−6
y
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
x
true function


Polynomial function of order 7
8
6
4
2
0
−2
−4
y
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
x
true function


Polynomial function of order 9
6
4
2
0
−2
−4
−6
y
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
x
true function


Polynomial function of order 11
6
4
2
0
−2
−4
−6
y
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
x
true function

Problem 2



Please see my perceptron implementation by running: python3 assignment1p2.py. In problem 2 I implemented a perceptron and tested it on a dataset consisting of six points. I predicted the label (1 or -1) of these 6 points as follows:

- if $\theta^T x >= 0$ then output that y=1, where θ = [bias, $weight_1$, $weight_2$] and x=[1, x_1, x_2]
- else output y=-1

I implemented learning θ as follows:

$\theta^{(t+1)} = \theta^{(t)} + \eta_t (y^{(i)} - \hat{y}^{(i)}) x^{(I)}$, where η_t = learning rate, $\hat{y}^{(i)}$ = predicted label

Therefore, the algorithm updates θ only when it predicts an incorrect label. I initialized θ to [0, 0, 0] and kept updating it until the algorithm converged.

Finally, I plotted the dataset and the final hyperplane, which I calculated using the learned θ:

hyperplane = slope * x + intercept, where slope = $-\text{weight}_1 / \text{weight}_2$ and intercept = -bias / weight_2

Learned theta: [bias, w_1, w_2] = [5.8 -2.6 -0.2]

Predicted labels: [1, 1, 1, -1, -1, -1]

Actual labels: [1 1 1 -1 -1 -1]

Problem 3

Please see my logistic regression implementation by running: python3 assignment1p3.py. In problem 3 I implemented logistic regression and predicted y ={spam, not spam} as follows:

probability of spam = $\sigma(\theta^T x)$, where σ is a logistic function $\sigma(z) = 1/1+e^{-z}$

I decided to predict y as follows:

if probability of spam > 0.5:

    y = 1 # spam

else:

    y = 0 # not spam

I learned the θ using gradient descent as follows: $\theta^{(t+1)} = \theta^{(t)} -$ learning rate * $\nabla_\theta J$

(a) Un-normalized data

The accuracy on test data of logistic regression model with learning rate = 0.01 and number of epochs = 100 is: 39.7001303781 %

The accuracy on test data of logistic regression model with learning rate = 0.1 and number of epochs = 100 is: 39.7653194263 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 100 is: 62.9074315515 %

The accuracy on test data of logistic regression model with learning rate = 0.3 and number of epochs = 100 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.5 and number of epochs = 100 is: 39.7001303781 %

The accuracy on test data of logistic regression model with learning rate = 0.8 and number of epochs = 100 is: 60.7561929596 %

The accuracy on test data of logistic regression model with learning rate = 0.9 and number of epochs = 100 is: 62.9074315515 %

The accuracy on test data of logistic regression model with learning rate = 0.99 and number of epochs = 100 is: 39.7001303781 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 0 is: 39.3741851369 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 10 is: 39.3741851369 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 20 is: 60.0391134289 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 50 is: 47.001303781 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 80 is: 39.7001303781 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 100 is: 62.9074315515 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 150 is: 60.6910039113 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 200 is: 40.221642764 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 300 is: 40.482398957 %

(b) Normalized data

The accuracy on test data of logistic regression model with learning rate = 0.01 and number of epochs = 100 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.1 and number of epochs = 100 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 100 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.3 and number of epochs = 100 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.5 and number of epochs = 100 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.8 and number of epochs = 100 is: 60.6910039113 %

The accuracy on test data of logistic regression model with learning rate = 0.9 and number of epochs = 100 is: 60.7561929596 %

The accuracy on test data of logistic regression model with learning rate = 0.99 and number of epochs = 100 is: 60.9517601043 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 0 is: 39.3741851369 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 10 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 20 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 50 is: 60.6258148631 %

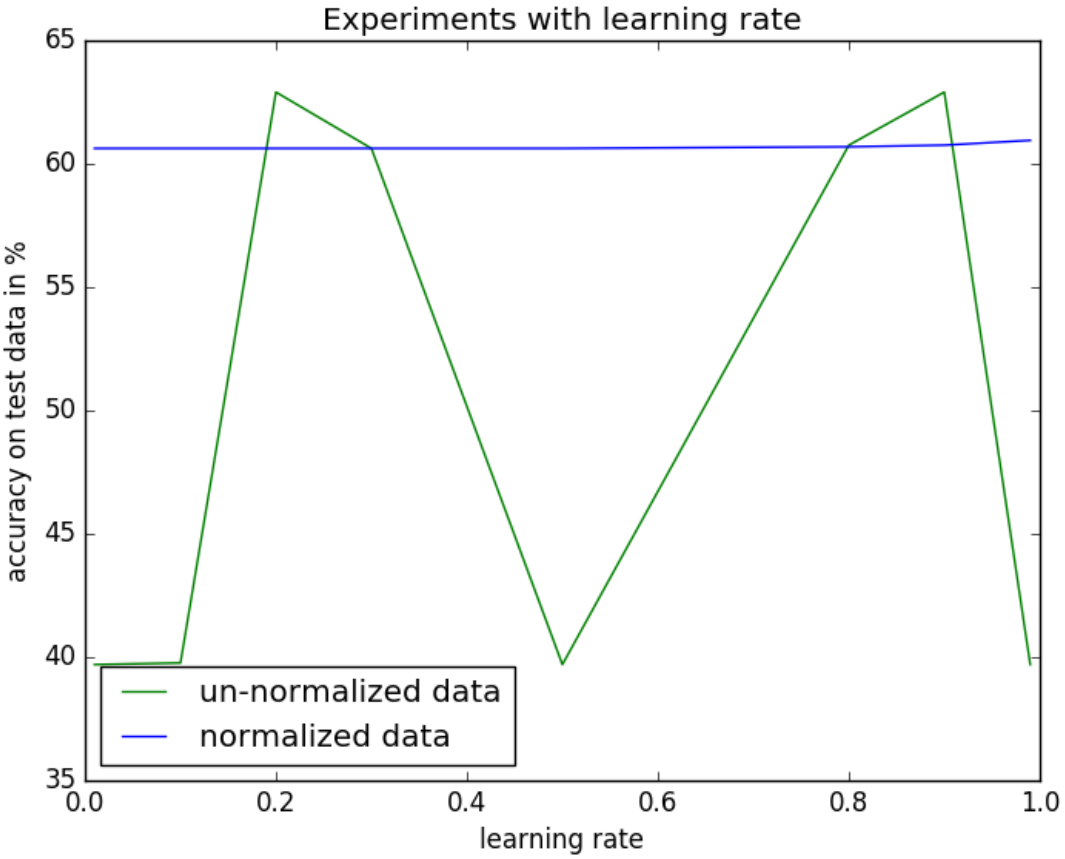
The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 80 is: 60.6258148631 %

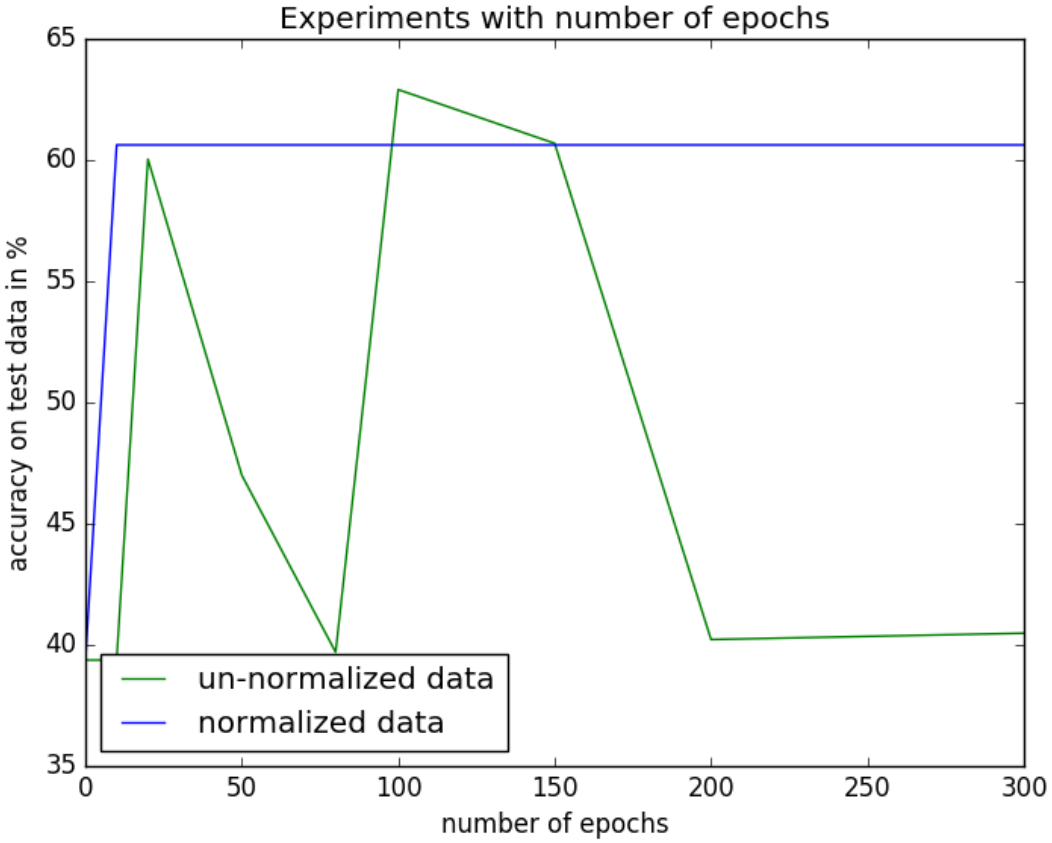
The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 100 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 150 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 200 is: 60.6258148631 %

The accuracy on test data of logistic regression model with learning rate = 0.2 and number of epochs = 300 is: 60.6258148631 %


Experiments with learning rate
accuracy on test data in %
learning rate
un-normalized data
normalized data
65
60
55
50
45
40
35
0.0
0.2
0.4
0.6
0.8
1.0


Experiments with number of epochs
accuracy on test data in %
number of epochs
un-normalized data
normalized data
65
60
55
50
45
40
35
0
50
100
150
200
250
300

Based on these results, we can see that normalization of data makes a great difference. The accuracy of un-normalized data is very volatile across different learning rates and different number of epochs. The accuracy on test data of the normalized data is above 60% across different learning rates and number of epochs.

These are the learning rates and number of epochs I experimented with:

learning_rates = [0.01, 0.1, 0.2, 0.3, 0.5, 0.8, 0.9, 0.99]

number_of_epochs = [0, 10, 20, 50, 80, 100, 150, 200, 300]