EE2211 Tutorial 7

Dr Feng LIN

Q1

This question explores the use of Pearson’s correlation as a feature selection metric. We are given the following training dataset:

	Datapoint 1	Datapoint 2	Datapoint 3	Datapoint 4	Datapoint 5
Feature 1	0.3510	2.1812	0.2415	-0.1096	0.1544
Feature 2	1.1796	2.1068	1.7753	1.2747	2.0851
Feature 3	-0.9852	1.3766	-1.3244	-0.6316	-0.8320
Target y	0.2758	1.4392	-0.4611	0.6154	1.0006

What are the top two features we should select if we use Pearson’s correlation as a feature selection metric?

Q1

Given N pairs of datapoints $\{(a_1, b_1), (a_2, b_2), \cdots, (a_N, b_N)\}$, the Pearson's correlation r is defined as

$$r = \frac{Cov(a,b)}{\sigma_a \sigma_b}$$

where

$$\sigma_a = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(a_i - \bar{a})^2} \qquad \sigma_b = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(b_i - \bar{b}\right)^2}$$

Empirical standard deviation of a and b

and

$$Cov(a,b) = \frac{1}{N}\sum_{n=1}^{N}(a_i - \bar{a})\left(b_i - \bar{b}\right)$$

Empirical covariance between a and b

where

$$\bar{a} = \sum_{n=1}^{N} a_n \qquad \bar{b} = \sum_{n=1}^{N} b_n$$

Empirical means of a and b respectively

Q1

Pearson's correlation $r = \frac{Cov(a,b)}{\sigma_a \sigma_b}$

$$\text{Mean of Feature 1} = \mu_1 = \frac{0.3510+2.1812+0.2415-0.1096+0.1544}{5} = 0.5637$$

$$\text{Mean of Feature 2} = \mu_2 = \frac{1.1796+2.1068+1.7753+1.2747+2.0851}{5} = 1.6843$$

$$\text{Mean of Feature 3} = \mu_3 = \frac{-0.9852+1.3766-1.3244-0.6316-0.8320}{5} = -0.4793$$

$$\text{Mean of Target y} = \mu_y = \frac{0.2758+1.4392-0.4611+0.6154+1.0006}{5} = 0.5740$$

$$\text{Feature 1 std} = \sigma_1 = \sqrt{\frac{(0.3510-\mu_1)^2+(2.1812-\mu_1)^2+(0.2415-\mu_1)^2+(-0.1096-\mu_1)^2+(0.1544-\mu_1)^2}{5}} = 0.8229$$

$$\text{Feature 2 std} = \sigma_2 = \sqrt{\frac{(1.1796-\mu_2)^2+(2.1068-\mu_2)^2+(1.7753-\mu_2)^2+(1.2747-\mu_2)^2+(2.0851-\mu_2)^2}{5}} = 0.3924$$

$$\text{Feature 3 std} = \sigma_3 = \sqrt{\frac{(-0.9852-\mu_3)^2+(1.3766-\mu_3)^2+(-1.3244-\mu_3)^2+(-0.6316-\mu_3)^2+(-0.8320-\mu_3)^2}{5}} = 0.9552$$

$$\text{Target y std} = \sigma_y = \sqrt{\frac{(0.2758-\mu_y)^2+(1.4392-\mu_y)^2+(-0.4611-\mu_y)^2+(0.6154-\mu_y)^2+(1.0006-\mu_y)^2}{5}} = 0.6469$$

Q1

- Pearson's correlation $r = \frac{Cov(a,b)}{\sigma_a \sigma_b}$

Cov(Feature 1, y) = $\frac{1}{5}[(0.3510-\mu_1)(0.2758-\mu_y)+(2.1812-\mu_1)(1.4392-\mu_y)+(0.2415-\mu_1)(-0.4611-\mu_y)+(-0.1096-\mu_1)(0.6154-\mu_y)+(0.1544-\mu_1)(1.0006-\mu_y)] = 0.3188$

Cov(Feature 2,y) = $\frac{1}{5}[(1.1796-\mu_2)(0.2758-\mu_y)+(2.1068-\mu_2)(1.4392-\mu_y)+(1.7753-\mu_2)(-0.4611-\mu_y)+(1.2747-\mu_2)(0.6154-\mu_y)+(2.0851-\mu_2)(1.0006-\mu_y)] = 0.1152$

Cov(Feature 3,y) = $\frac{1}{5}[(-0.9852-\mu_3)(0.2758-\mu_y)+(1.3766-\mu_3)(1.4392-\mu_y)+(-1.3244-\mu_3)(-0.4611-\mu_y)+(-0.6316-\mu_3)(0.6154-\mu_y)+(-0.8320-\mu_3)(1.0006-\mu_y)] = 0.4949$

Correlation of Feature 1 & y = $\frac{Cov(Feature\ 1,y)}{\sigma_1 \sigma_y} = \frac{0.3188}{0.8229 \times 0.6469} = \boxed{0.5988}$

Correlation of Feature 2 & y = $\frac{Cov(Feature\ 2,y)}{\sigma_2 \sigma_y} = \frac{0.1152}{0.3924 \times 0.6469} = 0.4537$

Correlation of Feature 3 & y = $\frac{Cov(Feature\ 3,y)}{\sigma_3 \sigma_y} = \frac{0.4949}{0.9552 \times 0.6469} = \boxed{0.8009}$

Therefore, the top 2 features are Feature 1 and Feature 3.

Q2

This question further explores linear regression and ridge regression. The following data pairs are used for training and testing:

Training:

$\{x = -10\} \rightarrow \{y = 4.18\}$
$\{x = -8\} \rightarrow \{y = 2.42\}$
$\{x = -3\} \rightarrow \{y = 0.22\}$
$\{x = -1\} \rightarrow \{y = 0.12\}$
$\{x = 2\} \rightarrow \{y = 0.25\}$
$\{x = 7\} \rightarrow \{y = 3.09\}$

Testing:

$\{x = -9\} \rightarrow \{y = 3\}$
$\{x = -7\} \rightarrow \{y = 1.81\}$
$\{x = -5\} \rightarrow \{y = 0.80\}$
$\{x = -4\} \rightarrow \{y = 0.25\}$
$\{x = -2\} \rightarrow \{y = -0.19\}$
$\{x = 1\} \rightarrow \{y = 0.4\}$
$\{x = 4\} \rightarrow \{y = 1.24\}$
$\{x = 5\} \rightarrow \{y = 1.68\}$
$\{x = 6\} \rightarrow \{y = 2.32\}$
$\{x = 9\} \rightarrow \{y = 5.05\}$

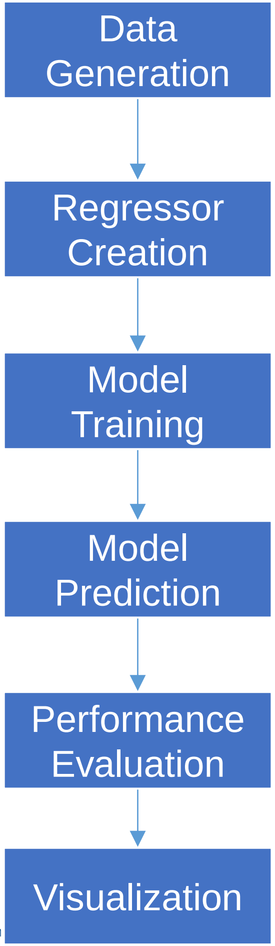
Q2

This question further explores linear regression and ridge regression. The following data pairs are used for training and testing:

a) Use the polynomial model from orders 1 to 6 to train and test the data without regularization. Plot the Mean Squared Errors (MSE) over orders from 1 to 6 for both the training and the test sets. Which model order provides the best MSE in the training and test sets? Why?

b) Use regularization (ridge regression) $\lambda = 1$ for all orders and repeat the same analyses. Compare the plots of (a) and (b). What do you see?

Main Function: Tut_Q2



- Training Data: Input features (x) and labels (y) for training the model.
- Test Data: Input features (xt) and labels (yt) for testing the model.

CreateRegressors(): A function that generates regressor matrices based on the input features and the maximum polynomial order.

EstimateRegressionCoefficients(): A function that estimates the coefficients of the regression model, with an option for regularization.

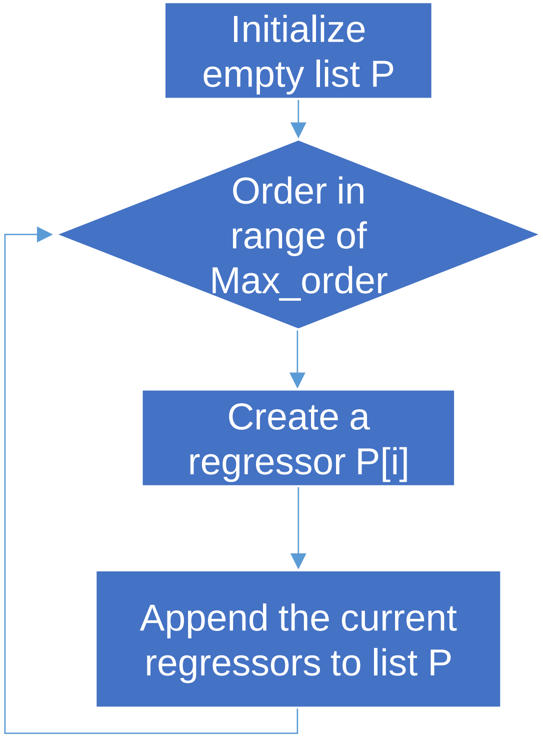
PerformPrediction(): A function that uses the estimated coefficients and regressor matrices to make predictions.

Mean Squared Error (MSE): Calculation of MSE for both training and test sets to evaluate the model's performance.

Plotting of Predicted Curves: Using matplotlib to plot the actual and predicted values.
Plotting of MSE: Visual representation of the MSE for different polynomial orders.

Function: CreateRegressors()

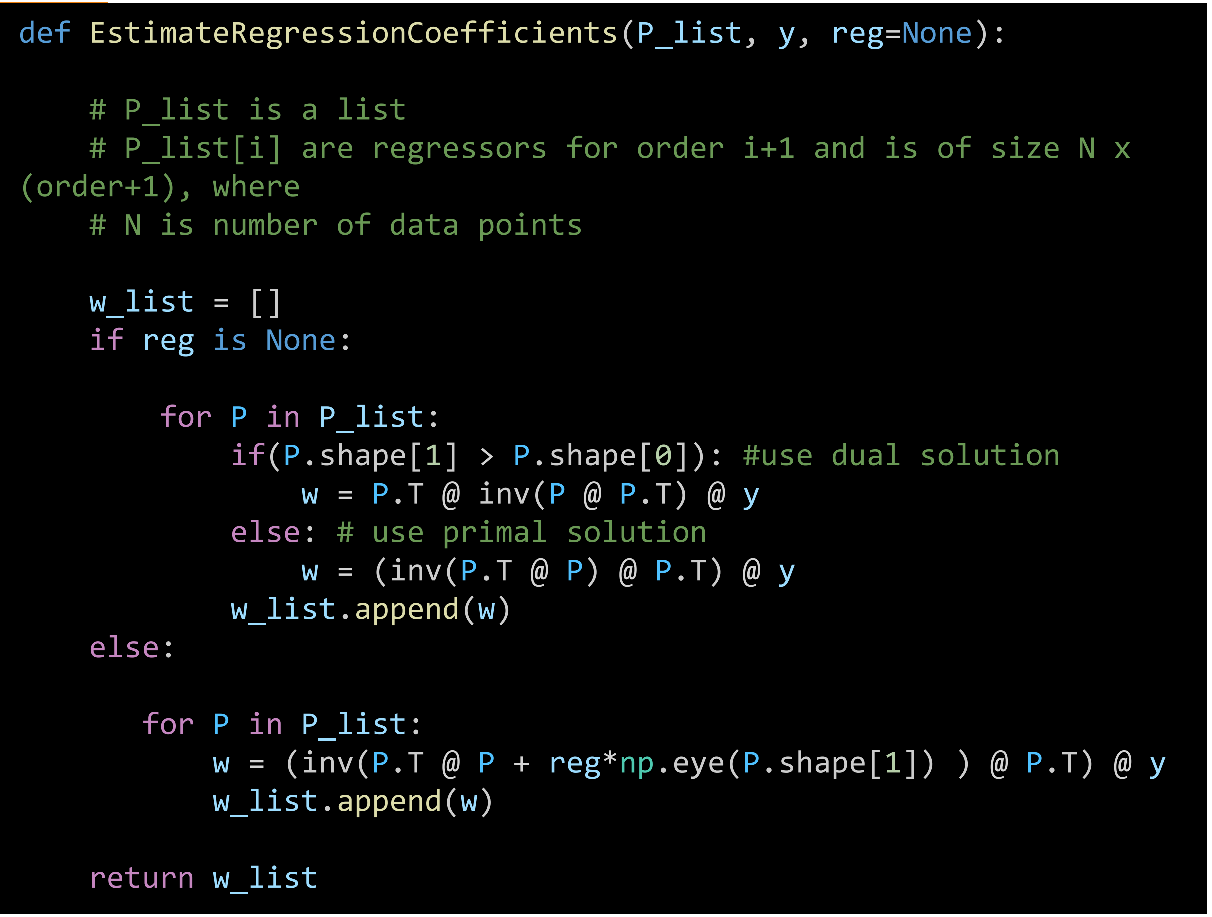
```
def CreateRegressors(x, max_order):

    # x is assumed to be array of length N
    # return P = list of regressors based on max_order
    # P[i] are regressors for order i+1 and is of size N x (order+1), where
    # N is number of data points

    P = [] #initialize empty list
    for order in range(1, max_order+1):
        current_regressors = np.zeros([len(x), order+1])
        current_regressors[:,0] = np.ones(len(x))
        for i in range(1, order+1):
            current_regressors[:,i] = np.power(x, i)
        P.append(current_regressors)

    return P
```


Initialize empty list P
Order in range of Max_order
Create a regressor P[i]
Append the current regressors to list P

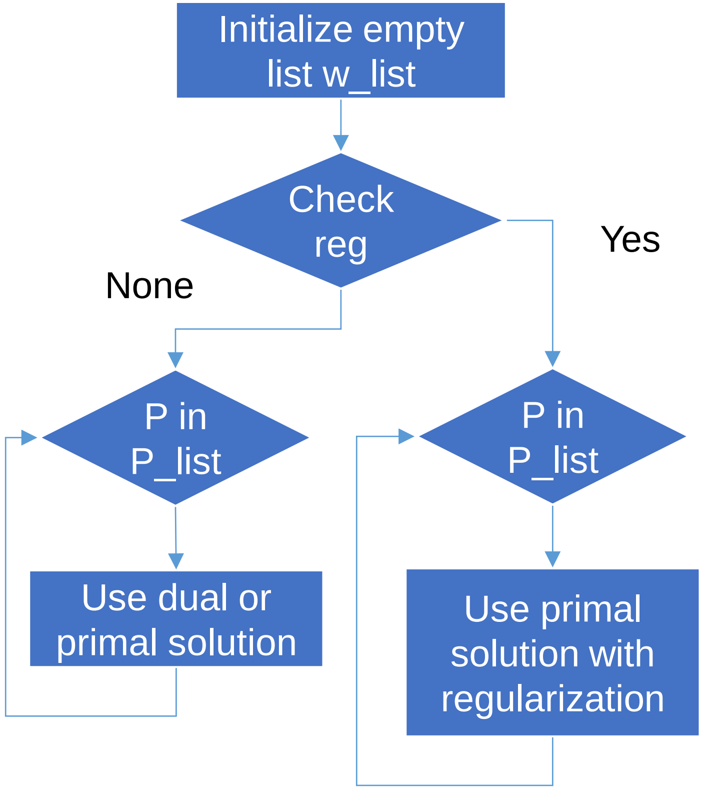
Create a regressor P[i]

- Set the first column of the matrix to ones. This represents the constant term in the polynomial.
- Set column i to x to the power of i: Calculate the power of x raised to the current index i and set it as the value for the current column in the matrix.

Function: EstimateRegressionCoefficients()

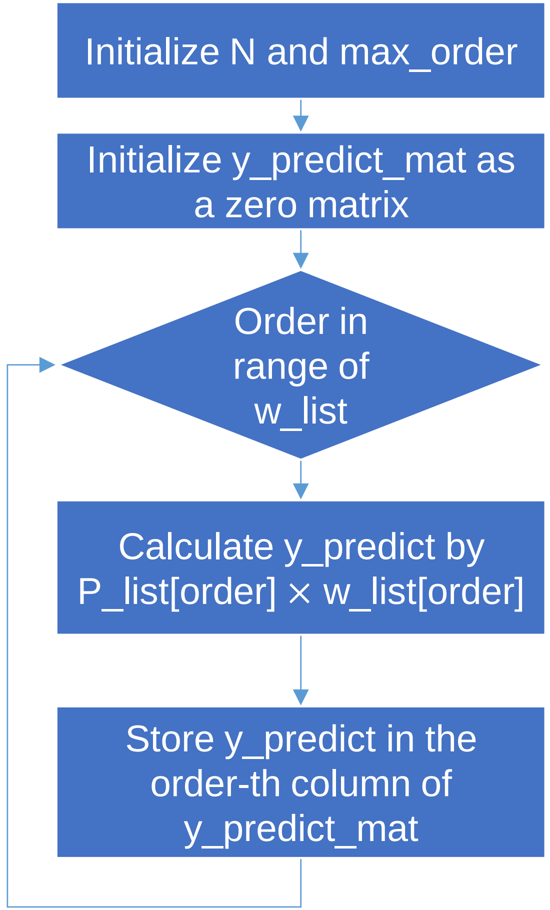


```
def EstimateRegressionCoefficients(P_list, y, reg=None):

    # P_list is a list
    # P_list[i] are regressors for order i+1 and is of size N x
(order+1), where
    # N is number of data points

    w_list = []
    if reg is None:

        for P in P_list:
            if(P.shape[1] > P.shape[0]): #use dual solution
                w = P.T @ inv(P @ P.T) @ y
            else: # use primal solution
                w = (inv(P.T @ P) @ P.T) @ y
            w_list.append(w)
    else:

        for P in P_list:
            w = (inv(P.T @ P + reg*np.eye(P.shape[1]) ) @ P.T) @ y
            w_list.append(w)

    return w_list
```


Initialize empty list w_list
Check reg
None
Yes
P in P_list
P in P_list
Use dual or primal solution
Use primal solution with regularization

Function: `PerformPrediction()`

```python
def PerformPrediction(P_list, w_list):

    # P_list is list of regressors
    # w_list is list of coefficients
    # Output is y_predict_mat which N x max_order, where N is 
the number of samples

    N = P_list[0].shape[0]
    max_order = len(P_list)
    y_predict_mat = np.zeros([N, max_order])
    for order in range(len(w_list)):
        y_predict = np.matmul(P_list[order], w_list[order])
        y_predict_mat[:,order] = y_predict

    return y_predict_mat
```



Initialize N and max_order
Initialize y_predict_mat as a zero matrix
Order in range of w_list
Calculate y_predict by P_list[order] × w_list[order]
Store y_predict in the order-th column of y_predict_mat

Q2

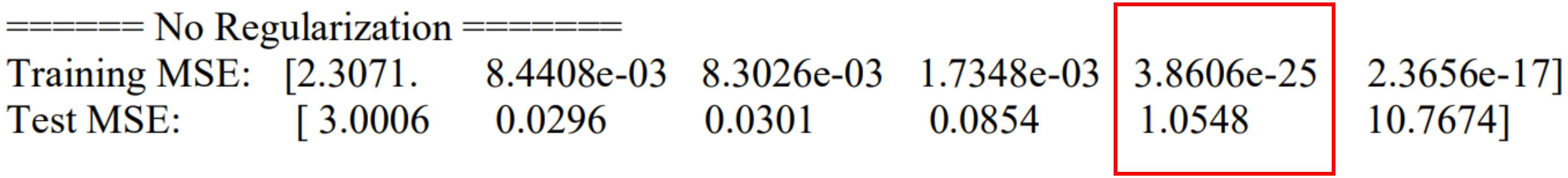
This question further explores linear regression and ridge regression. The following data pairs are used for training and testing:

a) Use the polynomial model from orders 1 to 6 to train and test the data without regularization. Plot the Mean Squared Errors (MSE) over orders from 1 to 6 for both the training and the test sets. Which model order provides the best MSE in the training and test sets? Why?

Q2(a)

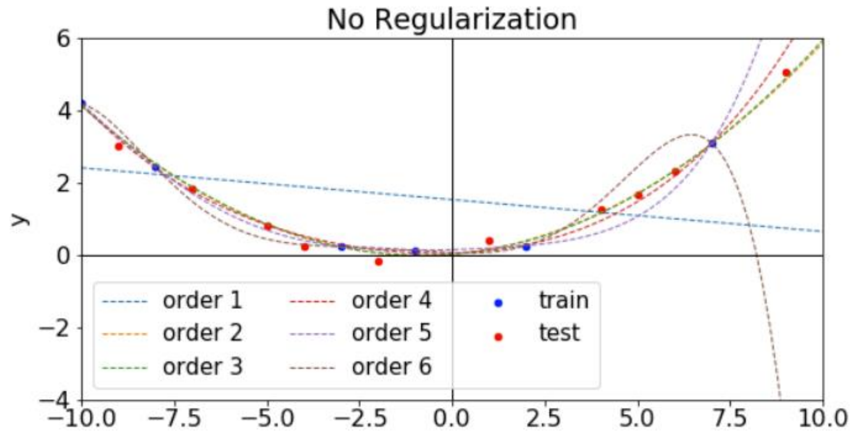
There are 6 training data points.

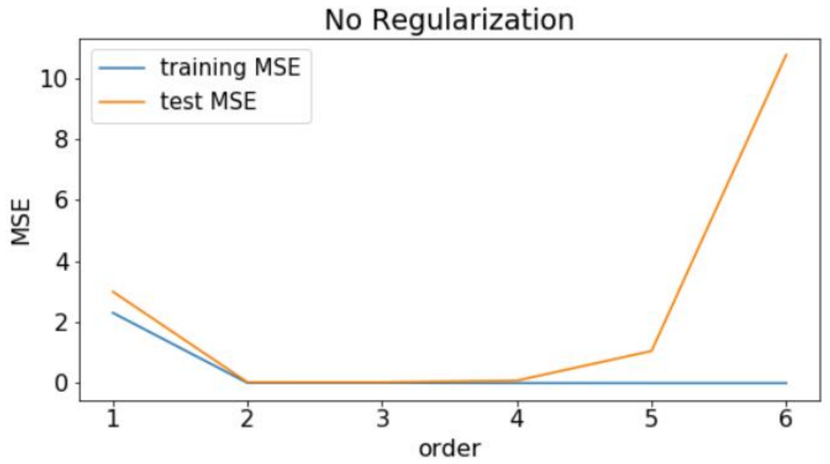
- For polynomial orders 1 to 5, we can use the Primal solution: $\hat{w} = (P^T P)^{-1} P^T y$
- For order 6, there are 7 unknowns, so the system is under-determined, so we use the Dual solution: $\hat{w} = P^T (P P^T)^{-1} y$



```
====== No Regularization =======
Training MSE:   [2.3071.      8.4408e-03   8.3026e-03   1.7348e-03   3.8606e-25   2.3656e-17]
Test MSE:        [ 3.0006      0.0296       0.0301       0.0854       1.0548       10.7674]
```

Q2(a)





Observe that the estimated polynomial curves for orders 5 and 6 pass through the training samples exactly. This results in training MSE of virtually 0, but high test MSE => overfitting

Note that even though the true underlying data came from a quadratic model (order = 2), estimated polynomial curves for orders 2, 3 and 4 have relatively low training and test MSE

Polynomial curve of order 1 (linear curves) have high training and test MSE => underfitting

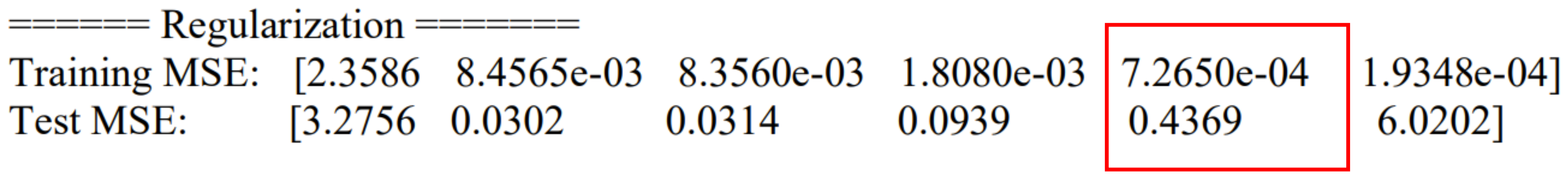
Q2

This question further explores linear regression and ridge regression. The following data pairs are used for training and testing:

b) Use regularization (ridge regression) $\lambda = 1$ for all orders and repeat the same analyses. Compare the plots of (a) and (b). What do you see?

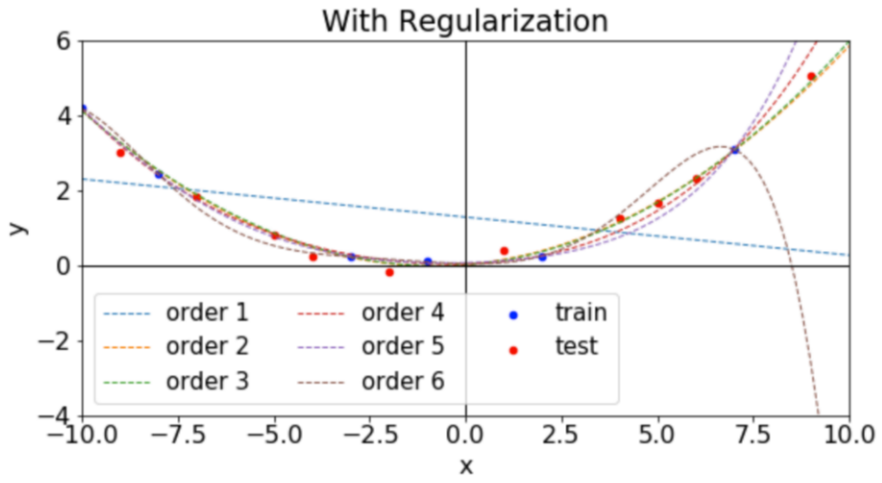
Q2(b)

With regularization, we can simply use the primal solution even for order 6: $\hat{w} = (P^T P + \lambda I)^{-1} P^T y$

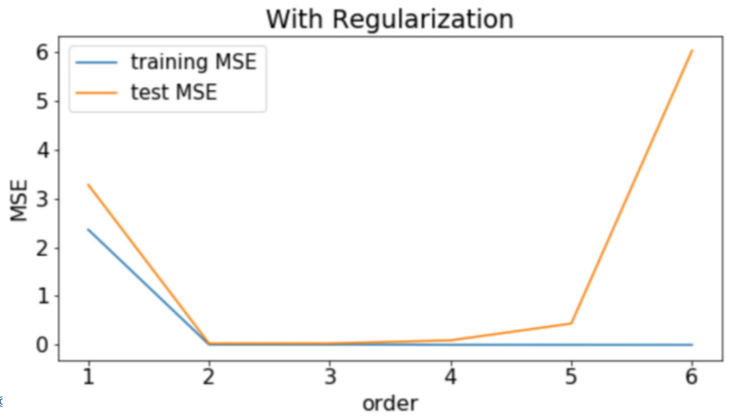


```
====== Regularization =======
Training MSE:  [2.3586   8.4565e-03   8.3560e-03   1.8080e-03   7.2650e-04   1.9348e-04]
Test MSE:      [3.2756   0.0302       0.0314       0.0939       0.4369       6.0202]
```

Q2(b)

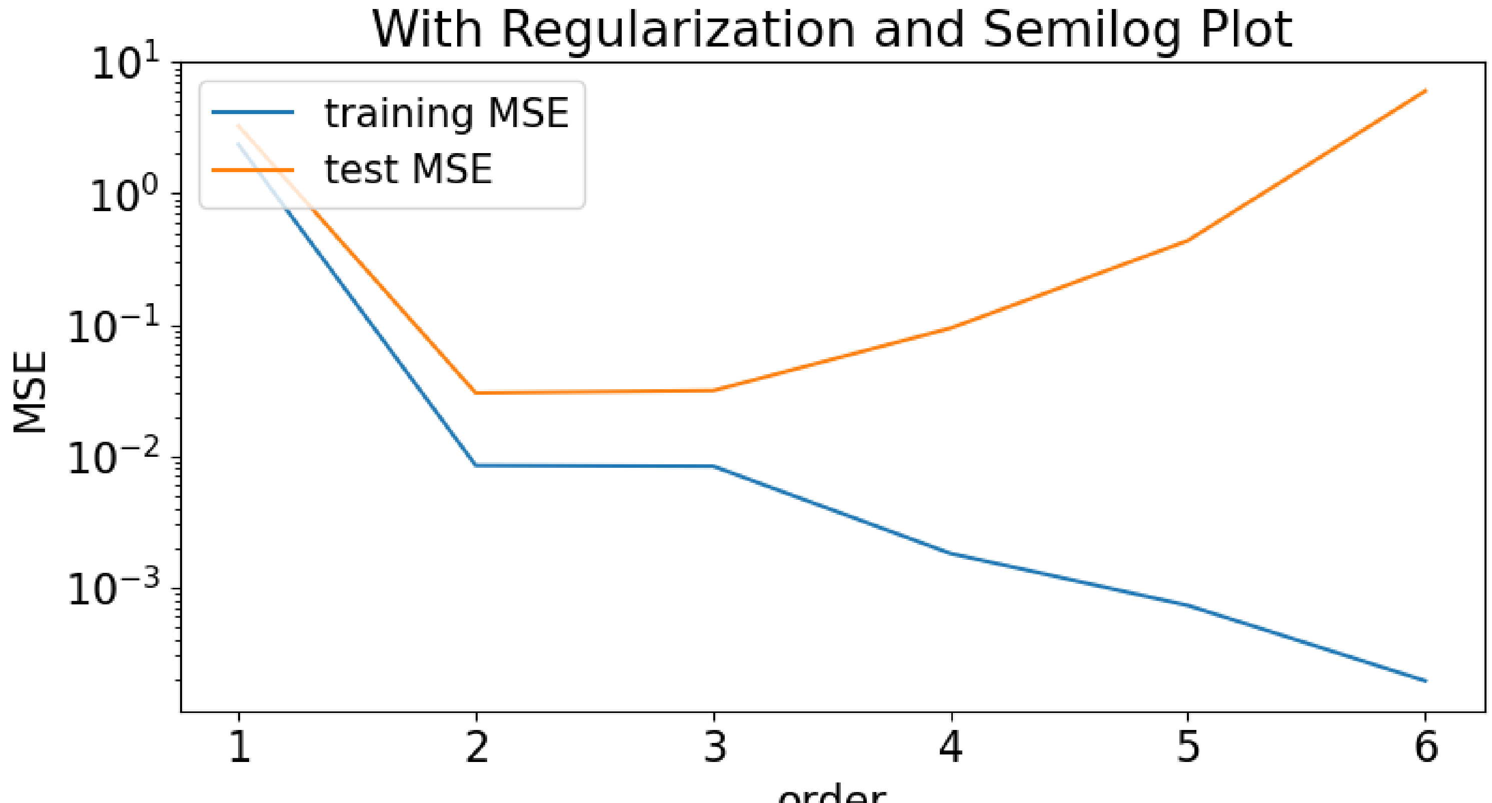


With the regularization, none of the polynomial curves passes through the training samples exactly. In the case of orders 5 and 6, test MSE dropped from 1.0548 (order 5) and 10.7674 (order 6) to 0.4369 (order 5) and 6.0202 (order 6) after regularization was added. Thus, the regularization reduces the overfitting



On the other hand, the regularization did not help orders 1 to 4. Observe that the test MSE actually went up. In this case, the regularization was overly strong, which ended up hurting these orders

Semilog Plot



THANK YOU