Regression Implementation Report

Yongjie Kang
A20349674

1. Problem statement

The main motivation of this report is to show the performance of linear regression and polynomial regression fitting to given datasets. During the implementation process, we are going to use ordinary linear regression, different degrees polynomial regression and Gaussian kernel function on ridge regression. Here are 8 datasets, our goal is to find the best model for each dataset.

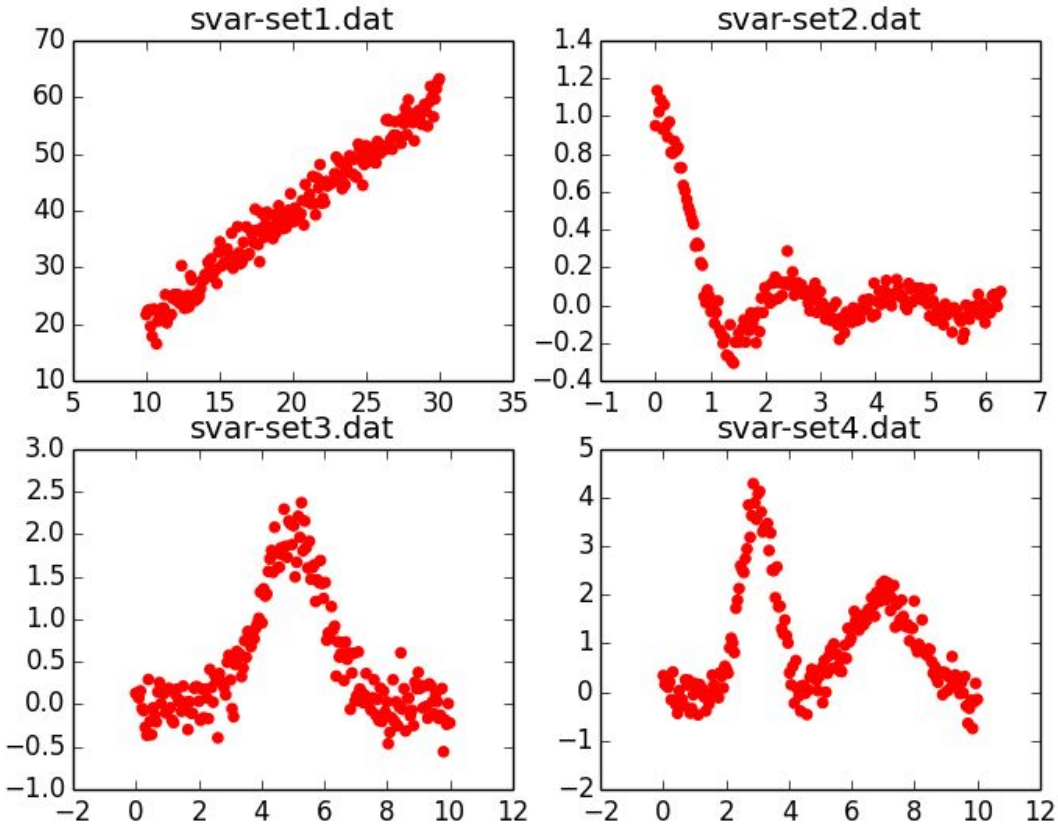
2. Proposed solution

For each datasets, we use both linear model and polynomial model to fit the data and select the model with the minimum testing error by 10 cross-validation.

3. Single Variable Regression

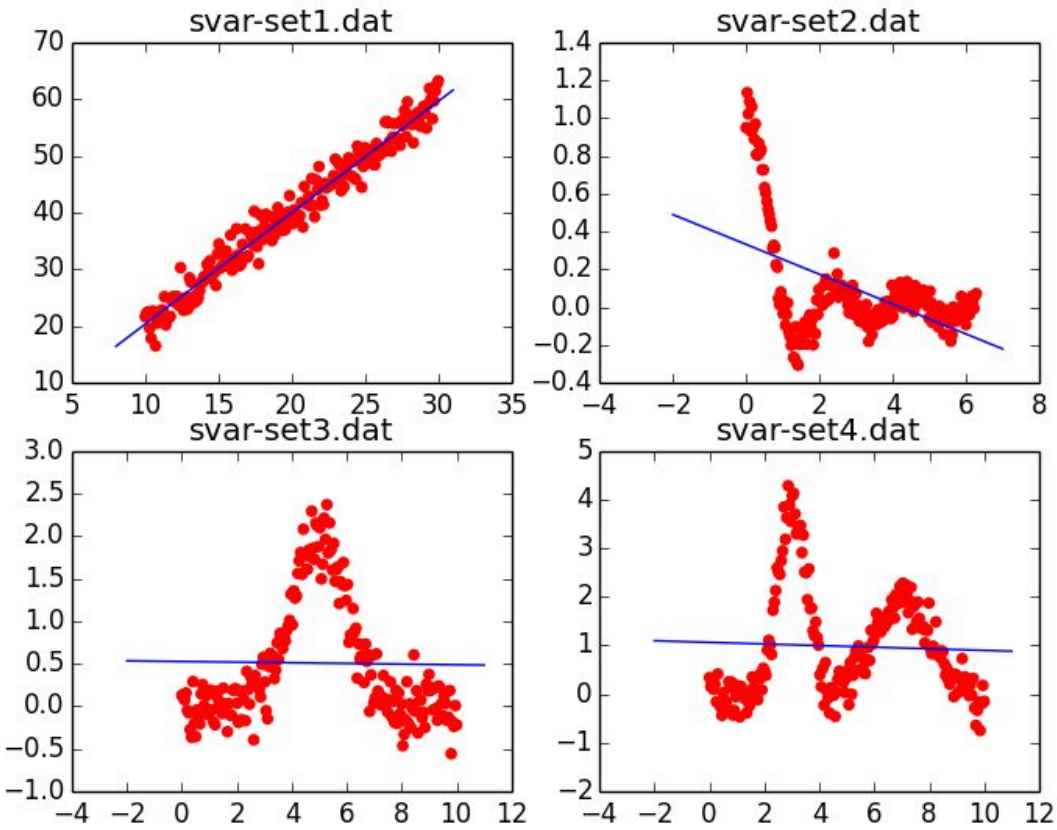
3.1 Plot and describe data

The graphs below are scatter plots of these four single-variable datasets. From these graphs, we are able to know that for the first dataset, linear model can fit the data well, since the feature and the label have relationship similar with linear relationship. For dataset 2, 3 and 4, obviously we can not use linear regression, then polynomial regression model can perform better.



Graph 1

3.2 Fit a linear regression model to each data



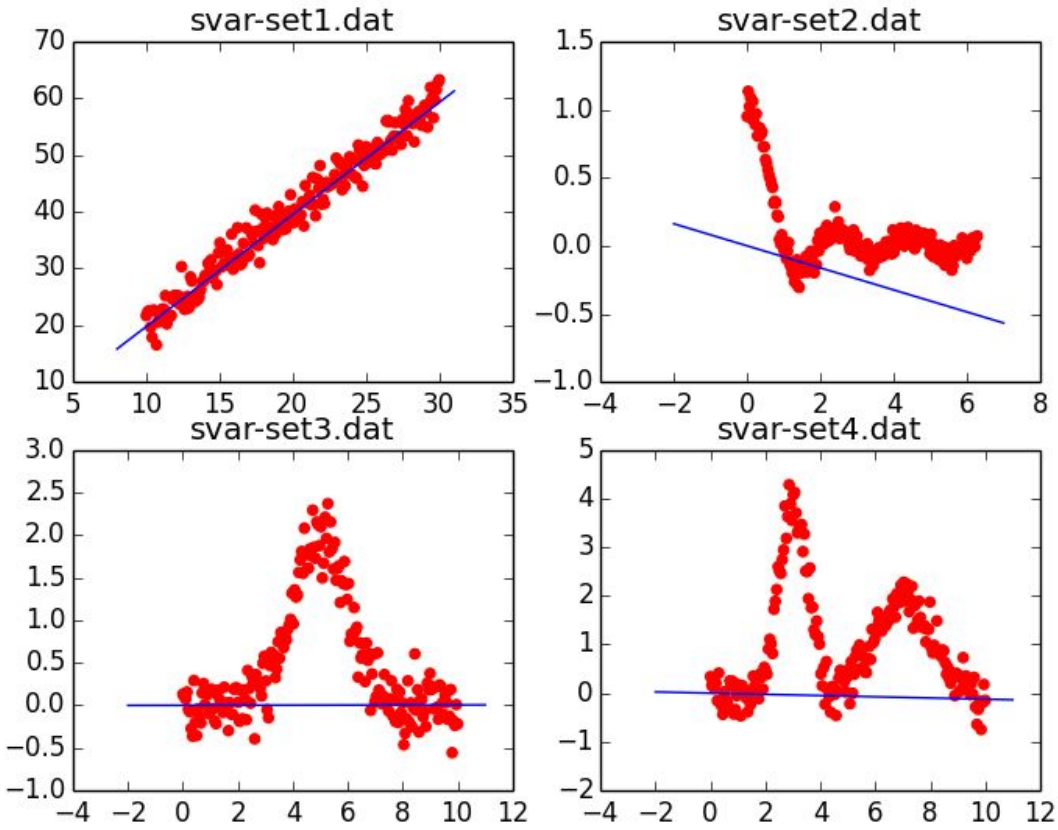
Graph 2

Dataset	Training Error(MSE)	Testing Error(MSE)
1	2.42709370177	3.49448623657
2	0.0696502911603	0.0382554680109
3	0.158104624807	0.325977754593
4	0.140020299841	0.641708517075

Table 1

After applying linear regression on these four datasets, we obtained the graphs plotting linear models on the top of original data. Obviously, linear regression worked well on the first data, for the other, although they had smaller training errors and testing errors, the models didn't fit the data, and the reason why they all had smaller errors is the range of the numbers in the last three datasets is smaller than the first.

3.3. Comparison the result with the given scikit-learn function



Graph-3 Given Regression Method

Dataset	Testing Error of Given Function(MSE)	Testing Error of My Function(MSE)
1	3.89	3.49448623657
2	0.06	0.0382554680109
3	0.40	0.325977754593
4	2.04	0.641708517075

Table 2

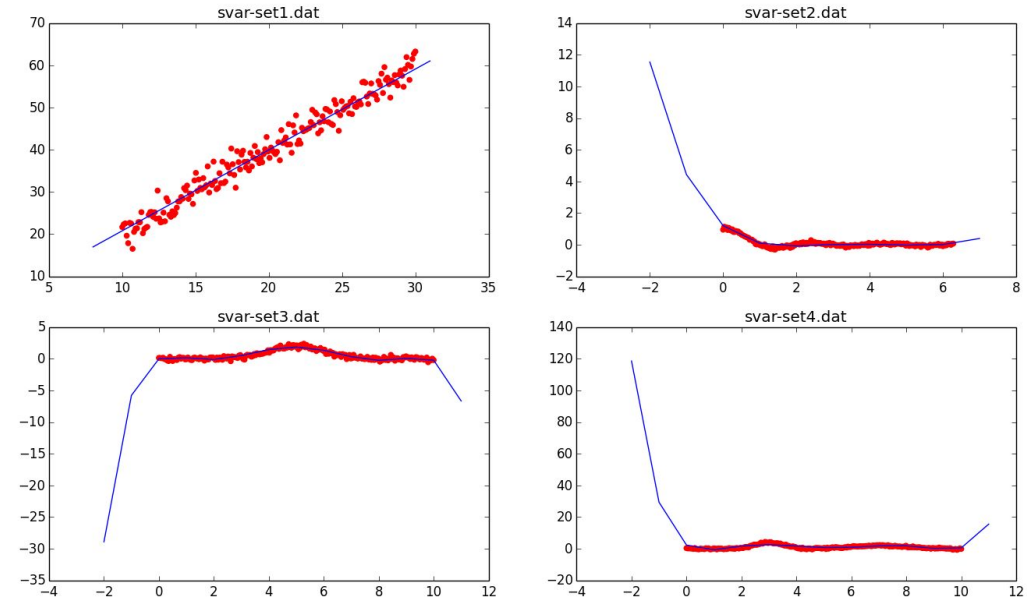
From table above, we notice that, averagely, our own function performed better than the given function according to the testing errors created by two methods.

3.4 Fit Polynomial Model to Data

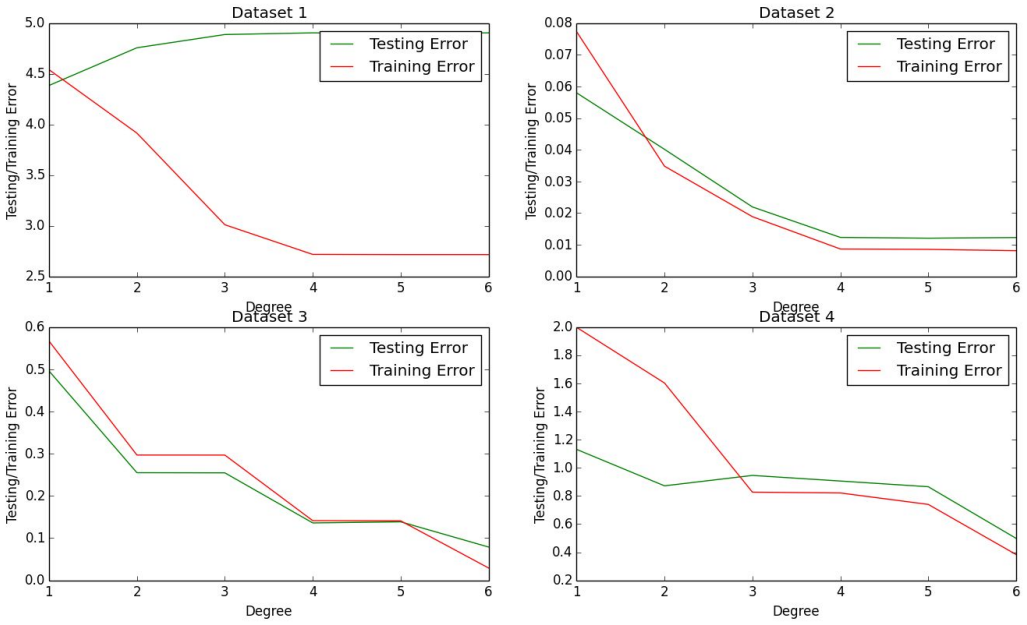
As we can see, the linear model didn't work well for dataset 2, 3 and 4. In this section, we are going to use polynomial model to fit our data. For each dataset, since it is a single variate dataset, the polynomial term is the summation of 1 to n power of x(feature vector). For example, if our initial vector is $[1, x]$, then 2 degree polynomial function is $[1, x, x^2]$, 3 degree polynomial function is $[1, x, x^2, x^3]$. As for each polynomial function, we also use 10-folder cross validation to select the one with smallest testing error. The output of one dataset is as follow:

```
########## Degree 1 ##########
Training Error: 0.0448958396912
Testing Error: 0.0678172959222
########## End degree 1##########


########## Degree 1 ##########
Training Error: 0.0663945972919
Testing Error: 0.0772342205048
########## End degree 1##########

...

########## Degree 1 ##########
Training Error: 0.0699452459812
Testing Error: 0.0698658519321
########## End degree 1##########

...

########## Degree 2 ##########
Training Error: 0.0337650477886
Testing Error: 0.0437033494314
########## End degree 2##########

...

########## Degree 6 ##########
Training Error: 0.0023648025468
Testing Error: 0.0201542364226
########## End degree 6##########
```

From the testing errors showed above, we are able to select the one with the smallest testing error and obtain its parameters. For each dataset, we plotted a polynomial model on the scatter graph. In this time, clearly polynomial regression worked better on dataset 2, 3 and 4 than linear regression.



Graph 4 Fitting Polynomial Model to Data



Graph 5 Degree-Testing/Training Error

By our polynomial regression function, we could get the relationship between the complexity of model and the testing/training error it creates. Obviously, for the first dataset, with the increment of model complexity, training error decreased a lot, but at the same time, the testing error increased, this caused by overfitting data. For other datasets, both training error and testing

error declined when model complexity became higher. **So we select models as follow for each dataset.**

```
Dataset 1
Optimal Degree 1
Optimal Paremeters [[ 1.64145263]
 [ 1.91578303]]

Dataset 2
Optimal Degree 5
Optimal Paremeters [[  1.20377214e+00]
 [ -1.92440348e+00]
 [  1.03445436e+00]
 [ -2.41371746e-01]
 [  2.45119164e-02]
 [ -8.38395116e-04]]

Dataset 3
Optimal Degree 6
Optimal Paremeters [[ -2.00704961e-02]
 [  1.65240939e+00]
 [ -2.53418627e+00]
 [  1.28798836e+00]
 [ -2.69300910e-01]
 [  2.47152958e-02]
 [ -8.29829168e-04]]

Dataset 4
Optimal Degree 6
Optimal Paremeters [[  2.29400367e+00]
 [ -1.05639355e+01]
 [  1.12414151e+01]
 [ -4.39923442e+00]
 [  7.93293245e-01]
 [ -6.69154078e-02]
 [  2.13753310e-03]]
```

3.5 Performance of Model after Reducing Training Data

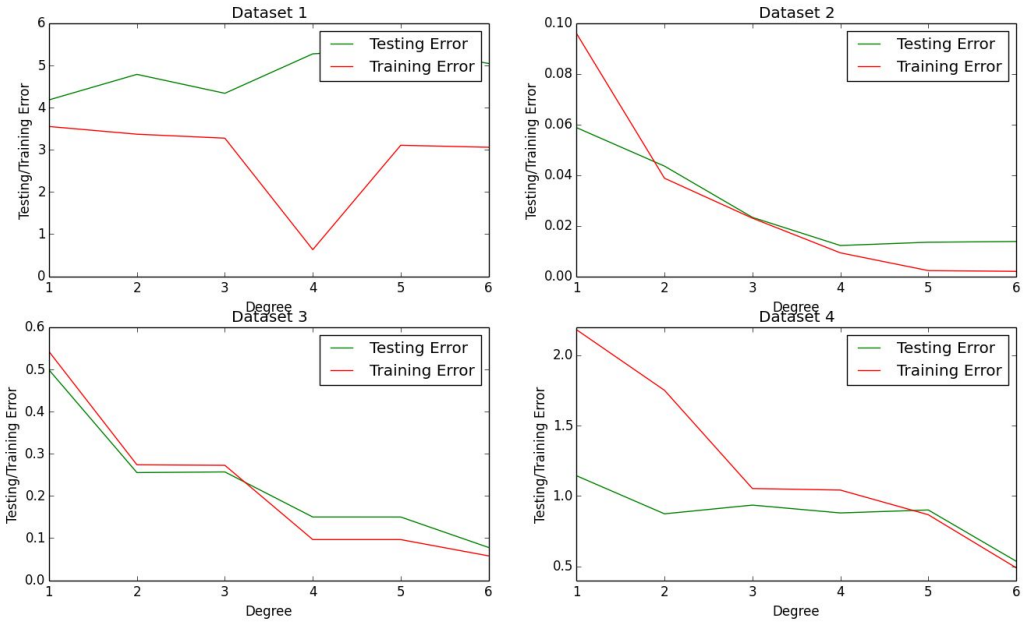
The size of training data plays a big role for getting a good model. In this section, we are going to reduce the size of training data by 25% and compare the testing error with the error we obtained before.

Dataset	Testing Error(MSE) of Data Reduced	Testing Error(MSE)
1	3.30725708008	3.49448623657
2	0.0389640837908	0.0382554680109

3	0.30919175148	0.325977754593
4	0.640606594086	0.641708517075

Table 3

After reducing training data and comparing the testing error with our linear regression model created before, we notice that the difference between two methods' MSE is small. Since the size of each single variate dataset is small(200 each), so the reduction of training influenced the performance little.



Graph 6 Degree-Testing/Training Error (Data Reduced)

When considering polynomial regression, our models didn't influenced by the reduction too much, either.

4. Multiple Variables Regression

4.1 Mapping Data to a Higher Dimension

Solving Multiple variables regression is more difficult than solving linear regression, not only training model becomes more complex, but also the time consuming increases a lot. In order to create a best model for each dataset, we use both explicit method as well as iterative method, also known as gradient descent. By loading "mvar-set1.dat" dataset, we can obtain a 2500×3 data matrix, the first two columns are feature matrix , the last column is label vector.

Mapping the 2-dimensional feature vector [1, x1,x2] to a higher dimension, for example, 6-dimensional vector, this feature vector will become [1, x1, x2, x1x2, x1^2, x2^2].
Then our feature matrix will be transformed from:

```
array([[ 1.        ,  1.67346942,  0.44897959],
       [ 1.        , -0.04081633,  0.53061223],
       [ 1.        , -0.77551019, -1.59183669],
       ...,
       [ 1.        , -0.2857143 ,  1.42857146]])
```

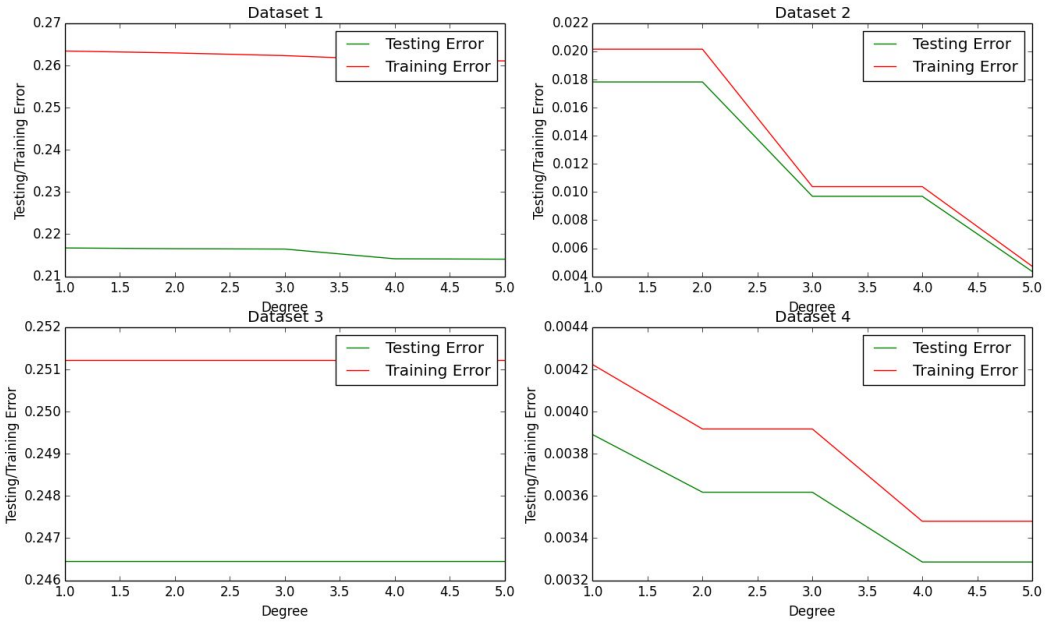
to:

```
array([[  1.00000000e+00,   1.67346942e+00,   4.48979586e-01,
          2.80049991e+00,   7.51353610e-01,   2.01582669e-01],
       [  1.00000000e+00,  -4.08163257e-02,   5.30612230e-01,
          1.66597244e-03,  -2.16576416e-02,   2.81549339e-01],
       [  1.00000000e+00,  -7.75510192e-01,  -1.59183669e+00,
          6.01416058e-01,   1.23448558e+00,   2.53394405e+00],
       ...,
       [  1.00000000e+00,  -2.85714298e-01,   1.42857146e+00,
          8.16326604e-02,  -4.08163293e-01,   2.04081642e+00]]), shape=(2500, 6)
```

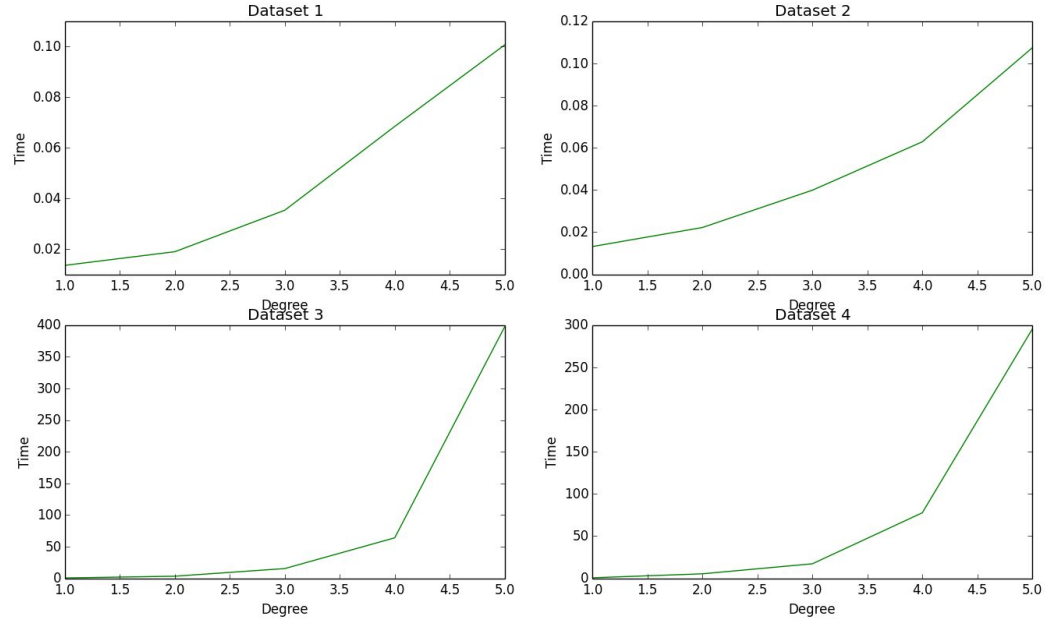
If we increase the dimension of our feature matrix, this could make a nonlinear model in lower dimension become linear in a higher dimension. So in this section, we transform our data to train a linear model in higher dimension to evaluate its performance.

4.2 Train Model

In order to select a model best fitting the data without too much time costing, here we consider both testing error the model creating and the time it costing.



Graph 7 Degree-Testing/Training Error



Graph 8 Degree-Time

We've got two graphs showed as above. For the dataset 1 and 3, clearly the difference between each degree is small, but the time costed increased a lot, so for these two ,according to Occam's razor, "among competing hypotheses the one with the fewest assumptions should be selected.", we select polynomial functions with maximum degree equals to 1. For dataset 2 and 4, it's true that more complex model gave us smaller testing error, but considering the time

costed, we'll select model with maximum degree 3. Since this polynomial regression model decreases much testing error without too much increasing in time costing.

4.3 Comparison Between Explicit Method and Iterative Method

Using gradient descent equation, we can iteratively compute each parameter by

$$\theta'_i = \theta_i^{(j-1)} - \gamma \frac{\partial}{\partial x_i} \nabla J(\theta^{(j-1)})$$

The pseudocode is as follow:

```
def gradient_descent(data, theta, learning_rate, iterative_num):
    for i from 1 to iterative_num:
         for each parameter in theta:
             theta[i] -= learning_rage * gradient_of_cost_function
             # here we update theta homogeneously.
    return theta
```

By implementation process, We found the program was too slow to even get a satisfied result, so we replaced it with stochastic gradient descent, which is computing gradient of $J(\theta)$ by choosing a random data from training data. Here is the result.

Dataset	Explicit Method		Stochastic Gradient Descent	
	Testing Error(MSE) of Data	Time Costing	Testing Error(MSE) of Data	Time Costing
1	0.22	0.012778997421264648	0.488	0.090900182724
2	0.018	0.013655900955200195	0.02560	0.108880996704
3	0.25	0.6571710109710693	0.2529	1.30189204216
4	0.0039	0.6442530155181885	0.0044	1.30418109894

Table 4

By comparing the time costing and testing error 2 methods created, it's easy to know that for the bigger datasets, for example, dataset 3 and dataset 4, SGD(Stochastic Gradient Descent) was working better due to the less testing error it created. However, for smaller datasets, dataset 1 and dataset 2, SGD didn't perform well as it did in bigger datasets.

4.4 Use Gaussian Kernel Function

The cost function of dual linear regression is defined as:

$$J(\alpha) = (X\theta - Y)^T(X\theta - Y),\ \theta = X^T\alpha$$

By taking the derivative of the function above, and let it be zero, we are able to get:

$$\alpha = (XX^T)^{-1}Y$$

After obtaining the equation, we can compute α by loading dataset and solving this regression problem by using the same approach as we did before. Unfortunately, using our datasets, we'd got Gram Matrix(XX^T) being singular. So we decided to use ridge regression instead.

The cost function of ridge regression defined as:

$$J(\theta) = (X\theta - Y)^T(X\theta - Y) + \lambda||\theta||^2$$

Letting the derivative of this function be 0 will give us

$$\alpha = (XX^T + \lambda I)^{-1}Y$$

This regression could make sure to solve the "singular" problem. Then we compute the kernel matrix of training data feature matrix with testing data feature matrix K, in the end, the predicted value of our testing data is equal to inner production of alpha transpose and K:

$$y_{hat} = \alpha^T K$$

We selected degree 1 polynomial model for primal linear regression. The result showed as follow:

Dataset	Primal Linear Regression		Dual Linear Regression	
	Testing Error(MSE) of Data	Time Costing	Testing Error(MSE) of Data	Time Costing
1	0.22	0.012778997421264648	0.25	167.1417
2	0.018	0.013655900955200195	0.023616926997296239	165.3581
3	0.25	0.6571710109710693	Didn't get due to long time costing	Over 5 minutes
4	0.0039	0.6442530155181885	Didn't get due to long time costing	Over 5 minutes

Table 5

From the table above, we can see that the model performance on prediction on testing data, primal linear regression got higher scores, also primal linear model cost less time than dual linear regression.

5. Summary

Training 8 datasets and selecting one model from them, we are able to know how overfitting influence the result of first dataset. During the polynomial model training process, we've got a

conclusion that more complex model would give us less training error. So for dataset 2, 3 and 4, we chose polynomial model with maximum degree 5, 6 and 6 respectively. For dataset 2, degree 6 would overfit dataset so we selected maximum degree 5 instead.

As for multiple variables datasets, we implemented primal linear regression, dual linear regression. In the process of solving primal problem, we also made comparison between explicit approach and iterative approach and found iterative method was computation expensive, even though they both gave us models with small testing errors.

For the most cases, our programs did good jobs and gave us valuable outputs and graphs, but for some specific computation expensive methods, we didn't get model from them, we should've done better for this if we could spend more time on computation to deal with these issues.