Friday, 1 st March 2019.

ASSIGNMENT1-CSC4007 ADCANCED MACHINE LEARNING REPORT

By Tianbai Peng 40120405

Summary

This assignment report talks about the whole process of using machine learning theory to solve a real world problem. Thorough the processing of solving the problem, it gradually describes how the functions used for implementation and the outputs received in each sub-task. And it's also demonstrated the machine learning theory used to solve every sub-task. For example, what are the output got after executed, what is the meaning and interpretation of each result and the reason the results different from sub-task to sub-task and finally get the regular pattern combination with the data to find out the mysteries of machine learning.

1. Introduction

Machine learning is a very important research field in computer science and artificial intelligence and it is very popular in nowadays sociery, therefore it’s very necessary to gain more knowledge for this module. This assignment is very nice designed to let us know every kind of basic knowledge about machine learning. In this assignment, I mostly used linear regression to solve a real-world problem that builds a model to predict house prices. And this is what in the tasks:

• Using linear regression to fit a linear model to a given training data of house prices
• Using different features: linear, quadratic, cubic, radial basis function (RBF)
• Performing regularization to tackle over-fitting
• Using cross-validation to select hyper-parameters
• Interpreting, explaining, and reporting experiment results.

In the second part, the above content will be described in detail, and talk about my implements of each part of tasks.

2. The details of each tasks

2.1 Task1

In this task 1, I made use of the dataset given above to fit a linear model for house price prediction.

2.1.1 Task1.1

In this sub task I import the module of numpy and used loadtxt() function from numpy to load data from "boston.data", and then spilt it to train data(80%) and test data(20%) . From the result, we can see the data shape of each dataset:

```
data.shape: (506, 14)
trainingSet.shape: (404, 14)
testSet.shape: (102, 14)
```

Which clearly shows the data has been spilt to trainingSet(404 rows) and testSet(102 rows)

2.1.2 Task1.2

In this sub task, at the beginning I split both the trainset and testSet to input part as X and output part as y, after that I mostly used the linear regression without regularization to fit a linear model. In this task I add a prepend_one() function which used for prepending a one vector to X. The matrix need to be added one to the left because this is for calculate more easily. Firstly I used this function to add one to the training set and then use the powerful functions in numpy module to calculate the beta_. The function to calculate beta is:
β = (((X^T * X)^-1)* X^T)* y

And next add a error_computing() which used a method to calculate the error. The theory is to get the average of all square root error: sum of all sqrt of errors and divide by the total number of training instances. Finally I got the training error and testing error:

```
Training error is 0.23744921042027042
Testing error is 0.5670690268198522
```

From the outputs, we can see that the accuracy of this linear model is very high

2.1.3 Task1.3

In this sub task I used the linear regression without regularization to fit quadratic and cubic model. In this task I also use a prepend_one() function which added at task 1.2 which is for prepending a one vector to X. And I also add two functions to calculate the quadratic feature and cubic feature, the theory of these two function is let each elements multiply with other elements without repetition and got their phi_X, Firstly I used this two functions to add one to the training set and then use the powerful functions in numpy module to calculate the beta_. The function to calculate beta is the same:

β = (((X^T * X)^-1)* X^T)* y

But in this case, I found that the PHI_X for quadratic and cubic is a singular matrix, so that we cannot simply use the inv(), but for singular matrices or non-square matrices, there is no inverse matrix, therefore I used pinv() to find its pseudo-inverse.

After got it's regression model, I still use the error_computing() which has already defined in previous task to calculate the error. The theory is add all the errors then over the all count. Finally, I got the training error and testing error from the output:

```
Training error of quadratic feature is 0.10587977415314571
Testing error of quadratic feature is 1.2298227834650919
Training error of cubic feature is 0.0571241254559708
Testing error of cubic feature is 25.25932024211068
```

From the outputs, we can see that the accuracy of this linear model is very high when using the training data, but over the test set, the error is a little high especially the cubic feature, therefore we need to think about it may have an overfitting, the way to solve the overfitting is in the next part.

2.1.4 Task1.4

From previous task we found it may have an overfitting so that we may need add a ridge regression for the regularization. The theory of regularization is to avoid overfitting even when choosing a too complex model, and the formula for ridge regression is β(ridge)=(XT*X+λI)^−1 *XT *Y, which base on beta formula β = (((X^T * X)^-1)* X^T)* y, added a weighting factor to balance both constraints and Minimise the sum of all the regression coefficients to the square , in this case we assign λ = 0.5 .Then do the same procedure of 1.2 and 1.3 again, and finally get the results with the ridge regression:

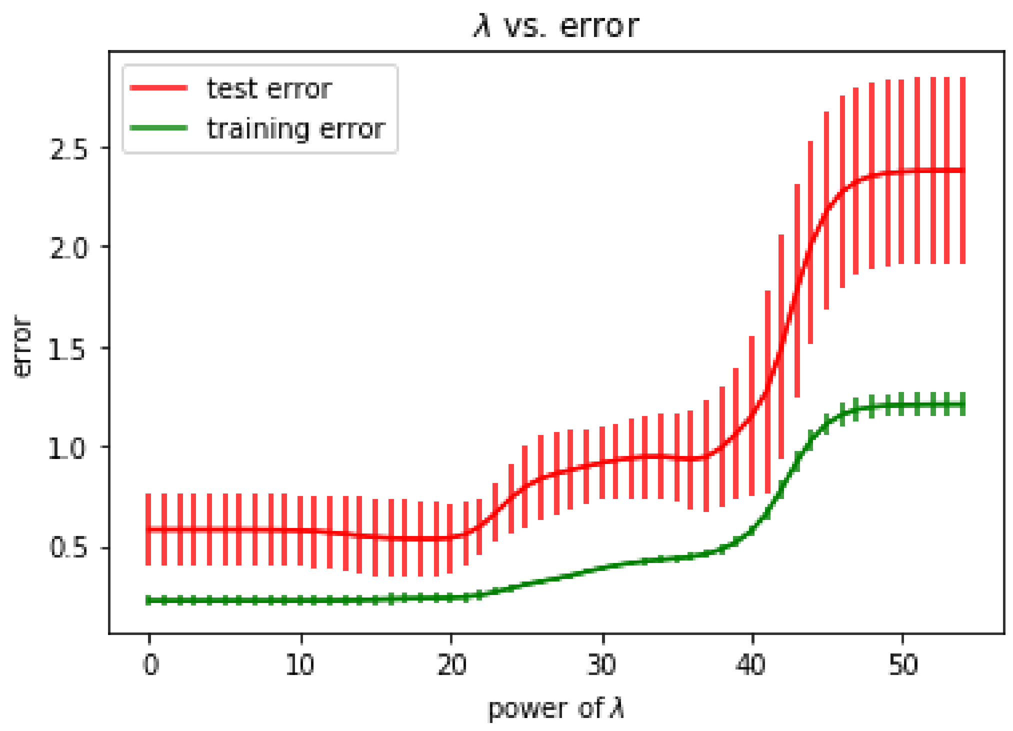
```
After ridge regression,Training error of linear feature is 0.237449
21042027042
After ridge regression,Testing error of linear feature is is 0.5670
690268198522
After ridge regression,Training error of quadratic feature is 0.109
74379335531416
After ridge regression,Testing error of quadratic feature is 1.3594
286779202887
After ridge regression,Training error of cubic feature is 0.0570987
4472064942
After ridge regression,Testing error of cubic feature is 25.3504900
57395834
```

From the outputs, we cannot see a significant change comparing to previous task which without a regularization, this may be the reason that the previous model did not have an over-fitting, so there is no need to do regularization. So, we can conclude that the cubic feature is not appropriate for this example, and linear feature and quadratic feature may be used for further research.

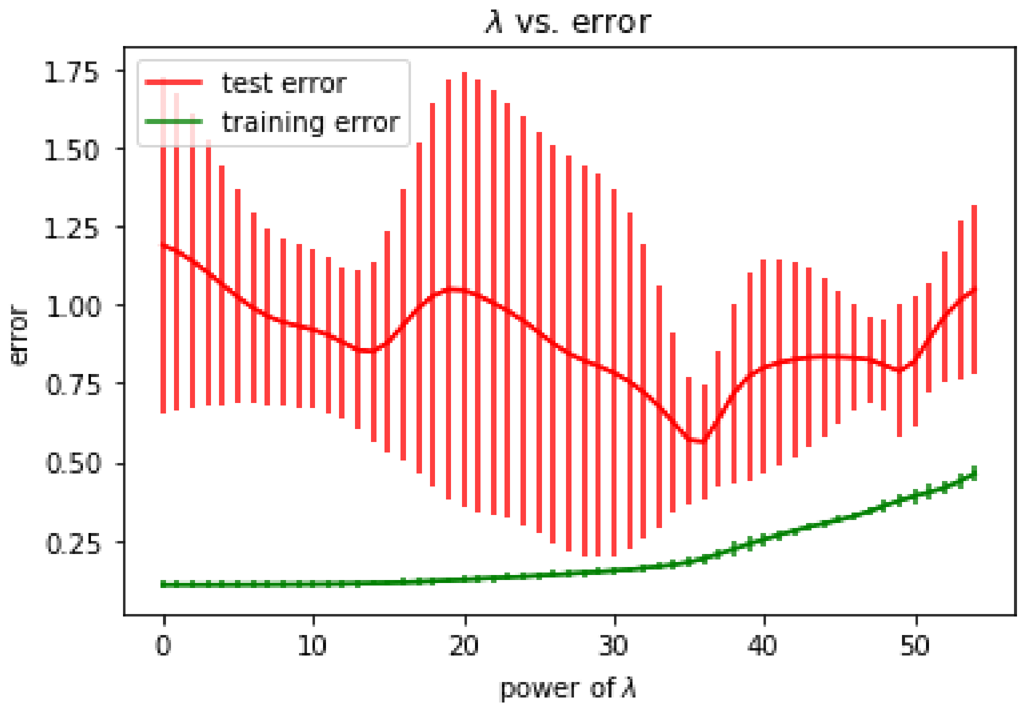
2.1.5 Task1.5

In this task I used 5-fold-cross-validation for ridge regression. cross-validation is a very efficient method to do the linear regression. In K-fold cross-validation, the original sample is randomly partitioned into K subsamples. Of the K subsamples, a single subsample is retained as the validation data for testing the model, and the remaining K − 1 subsamples are used as training data. The cross-validation process is then repeated K times (the folds), with each of the K subsamples used exactly once as the validation data.And in this case,we calculate lamda = 2^k with *k*={−15 ,−14 ,...,40}. Then I used the five_fold_cross_validation(feature) function to calculate the best λ and the testing error of the linear, quadratic and cubic models, we can pass the relative feature function as a parameter. And finally visualized the curves of training and validation error estimations (over 5 folds) together with their variances.

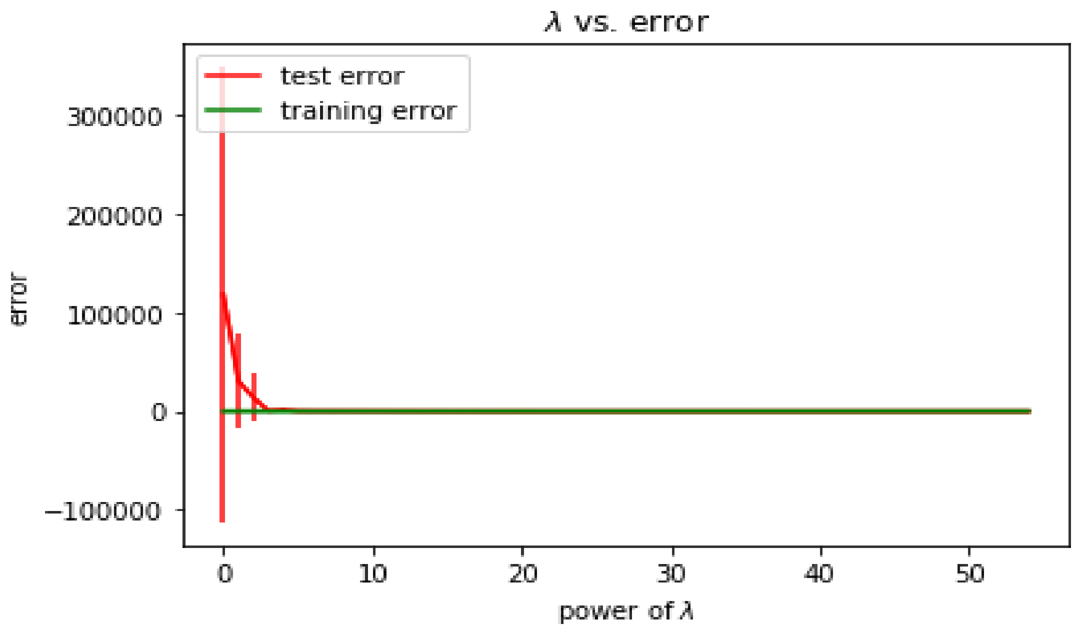
```
Linear Feature:
linear feature with five_fold_cross_validation:
best lamda =  8
best best_cost =  0.5335914564575702
```



```
Quadratic Feature:
quadratic feature with five_fold_cross_validation
best lamda =  2097152
best best_cost =  0.5643487193979444
```



```
Cubic Feature:
cubic feature with five_fold_cross_validation
best lamda =  549755813888
best best_cost =  0.8156613602767386
```



From the linear graph: As can be seen from the figure, the power of lamda value of 0~10 is the best. After 10 is become higher, the accuracy begins to decrease because of over fitting problems.

From the quadratic graph: As can be seen from the figure, the power of lamda value of 21~30 is the best. After 30 is become higher, the accuracy begins to decrease because of over fitting problems.

From the cubic graph: As can be seen from the figure, the more calculation done and the accuracy is more high

Overall the accuracy is very high over these models.

2.2 Task2

In this task, I used ridge regression with radial basis function features

(RBF) fit a non-linear model for house price prediction.

2.2.1 Task 2.1

In this sub task I used the linear regression fit an RBF model and with a ridge regression. In this task I used a RBF_Features() function which used for building a RBF feature. Before we doing the linear regression, we need choose some center points for the calculation, in this case, we are using all input instances in the training set and assigning them as the set of centers.

In this function the formula to calculate beta is same:
$\beta = (((X^T * X)^{-1}) * X^T) * y$
but in this formula:
$\varphi(x)=[1,\varphi1(x),\varphi2(x)]$

$$X=\begin{bmatrix} 1 & e^{-\|x1-c1\|/2\sigma^2} & e^{-\|x1-c2\|/2\sigma^2} \\ 1 & e^{-\|x2-c1\|/2\sigma^2} & e^{-\|x2-c2\|/2\sigma^2} \\ 1 & e^{-\|x3-c1\|/2\sigma^2} & e^{-\|x3-c2\|/2\sigma^2} \end{bmatrix} \quad \begin{matrix} \varphi(x1) \\ \varphi(x2) \\ \varphi(x3) \end{matrix}$$

Then used the same way to calculate the error, I got the training error and testing error and finally got the output:

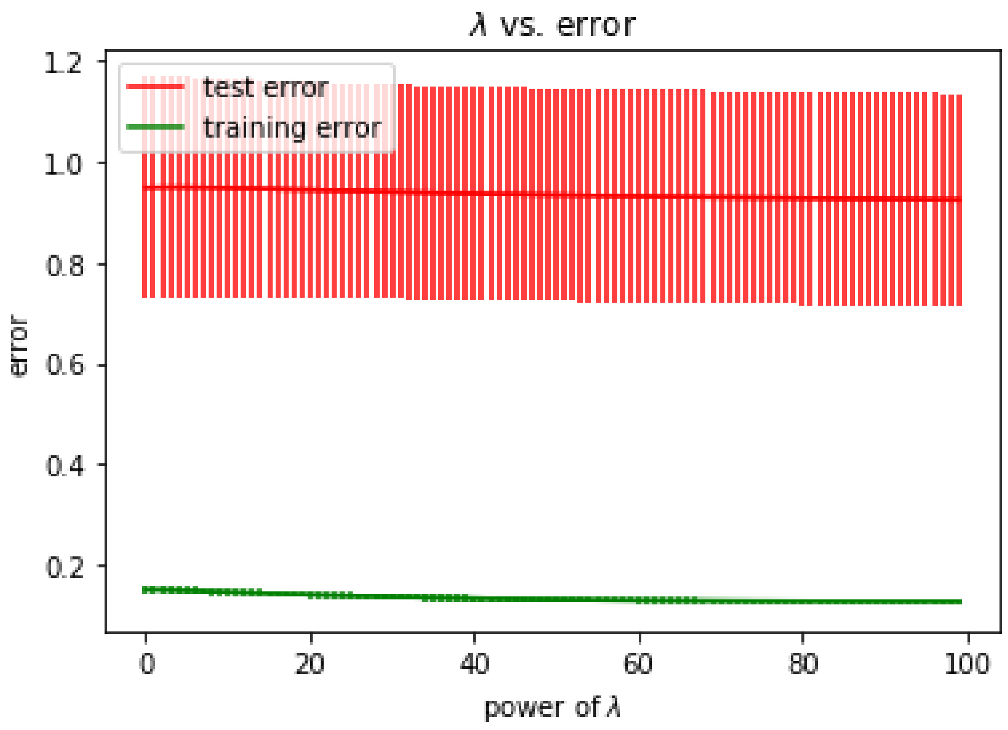
```
Training error of RBF feature is 0.151721001578446
Testing error of RBF feature is 0.9535275947919808
```

From the outputs, comparing this to the task 1.4, we can see that the accuracy of this RBF model is almost same with linear and quadratic model with regularization, and the accuracy of the RBF model is very high.

2.2.2 Task 2.2

In this task I used 5-fold-cross-validation which is almost same as task 1.5, but in this task, I used different k where *k*=1,2,…, 100 to calculate a best bandwidth σ^2=0.2∗*k* Then I used the five_fold_cross_validation() function to calculate the best λ and the testing error of the RBF models, the RBF feature is inside the cross validation. And finally visualized the curves of training and validation error estimations (over 5 folds) together with their variances.

```
RBF feature with five_fold_cross_validation:
best bandwith =  20.0
best best_cost =  0.924312890419797
```

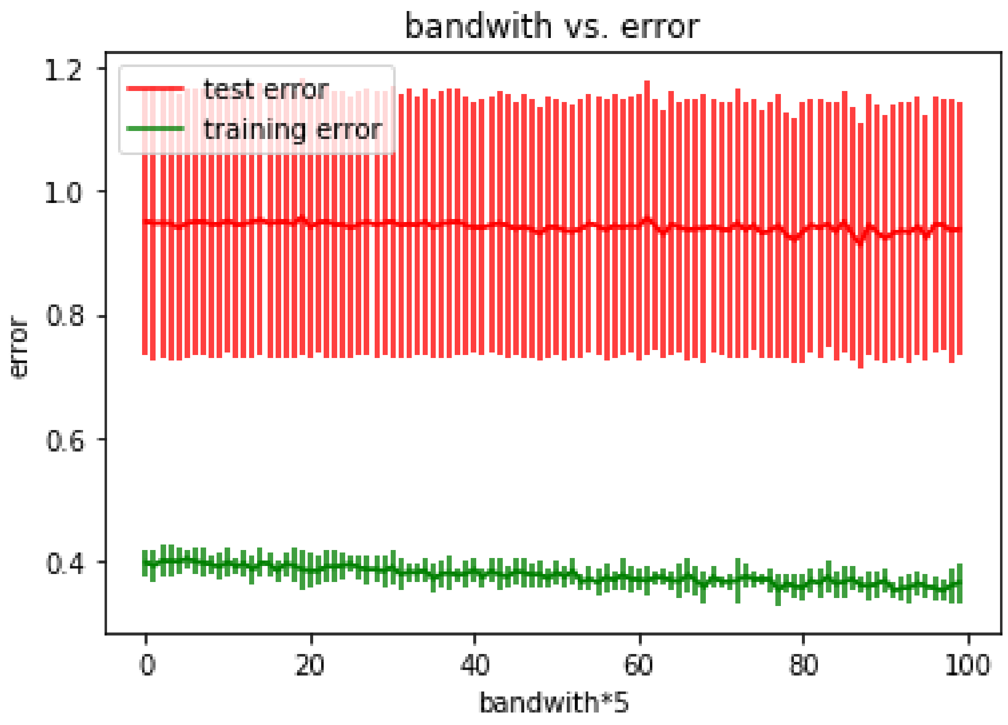


From the output, after comparing to the task2.1 we can see there is no much difference between their outputs, but this model become a little more accuracy.

2.2.3 Task 2.3

In this task the running procedure is almost same as task 2.2 but the centers are different. At this time, I randomly select 100 instances from the training set and let them as the center list rather than using all input instances as the set of centers. After I put the parameters to the five_fold_cross_validation() function , the next steps are totally same as sub-task 2.2.

```
RBF feature with five_fold_cross_validation:
best bandwith =  17.6
best best_cost =  0.9124032420469679
```



From the output, after comparing to the task2.2 we can see there is no much difference between their outputs but the curve shows a characteristic of fluctuations, not as smooth as before, what's more, this model become a little more accuracy.

3. Conclusion

This is a very good assignment which helps us totally understand the basic knowledge about linear regression and basic machine learning. Thorough doing the assignment, I consider that there still be a lot of points need me to master, especially the cross-validation part and how to improve the accuracy and even boosting the performance of a machine learning program. To this end, my next step is to learn other unknown information about machine learning for building a better AI programs in future.