Report

Pingxuan Huang 15307130283

linear regression and nonlinear basis

1. 1

(I) Analysis

In the original fitting process of this question, since the fitting model is set to: $y_i = w^T x_i$ (no offset term is set) so the fitting result is not good. Then the model hypothesis is changed to: $y_i = w^T x_i + \beta$. After that, calculating w^T and β by the least-squares method can improve the fitting accuracy.

(II) Specific process

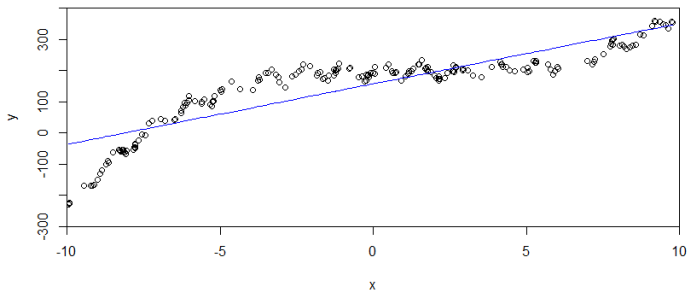
The principle of *the least squares method* is to make RSS (the residual sum of squares（$(y_1 - \hat{\beta}_0 - \hat{\beta}_1 * x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 * x_2)^2 + \cdots\cdots$）the smallest.

Through mathematical derivation, we know that according to the formula $\hat{\beta}_1 = \sum_1^n \frac{\sum_1^n (x_i - \bar{x}) * (y_i - \bar{y})}{\sum_2^n (x_i - \bar{x})^2}$ and $\beta_0 = \bar{y} - \overline{\beta_1} * \bar{x}$, the slope and intercept satisfying the condition can be respectively determined to obtain a fitted straight line with a minimum RSS.

The updated least squares function code is:

```
leastSquaresBias<-function(X,Y){   # Least squares
  xm<-mean(X);
  ym<-mean(Y);
  P1<-sum(t((X-xm))*(Y-ym))/sum(t((X-xm))*(X-xm));
  P0<-ym-P1*xm;
  c(P1,P0);
}
```

The new image obtained by fitting is：



The new training/test error are：

```
[1] "The training error is: 3551.34587065661"
> paste("The testing error is:",sum((y_yu_te-y_te)^2)/length(X_te))
[1] "The testing error is: 3393.86909807158"
```

Compared to the original error：

```
Training error = 28122.82
Test error = 28298.97
```

There is a significant drop in error, and it can be seen from the image that the new fitted line fits the original data more closely. It can be seen that a significant performance improvement can be achieved by fitting the data through the least squares method.

1. 2

(I) Analysis

This problem is a typical "linear basis function" fitting problem which can be solved by MLE.

(II) Specific process

For the data, the model can be assumed as: $y(x, w) = \sum_{i=0}^{n-1} w_j \phi_j(x) = w^T \phi(x)$, where $\phi(x)$ represents a basic function related to x. And in this problem, $\phi_j(x)=x^j$.

Now suppose that under the conditions determined by x,β,w, the value of y satisfies the normal distribution: $p(t|x, w, \beta) = N(t|y(x, w), \beta^{-1})$， then for the input data set X, the total probability of getting all values is: $p(t|X, w, \beta) = \prod_1^n N(t_n|y(x_n, w), \beta^{-1})$. The purpose of maximum likelihood is to maximize the value of p by changing w, that is, to maximize the likelihood of a vested result.

Let the derivation of ln^p equal to 0: $\sum_{i=1}^{n} t_n \phi(x_n)^T - w^T(\sum_{n=1}^{N} \phi(x_n)\phi(x_n)^T) = 0$， then we can get that $w = (\phi^T\phi)^{-1}\Phi^T t$, where ϕ denotes X_{poly}. Then, according to $w_0 = \bar{t} - \sum_1^n w_j \overline{\Phi_j}$, we can figure out w_0.

Code:

Code to figure out W:

```
W<-solve(t(Xpoly)%*%Xpoly)%*%t(Xpoly)%*%y;
```

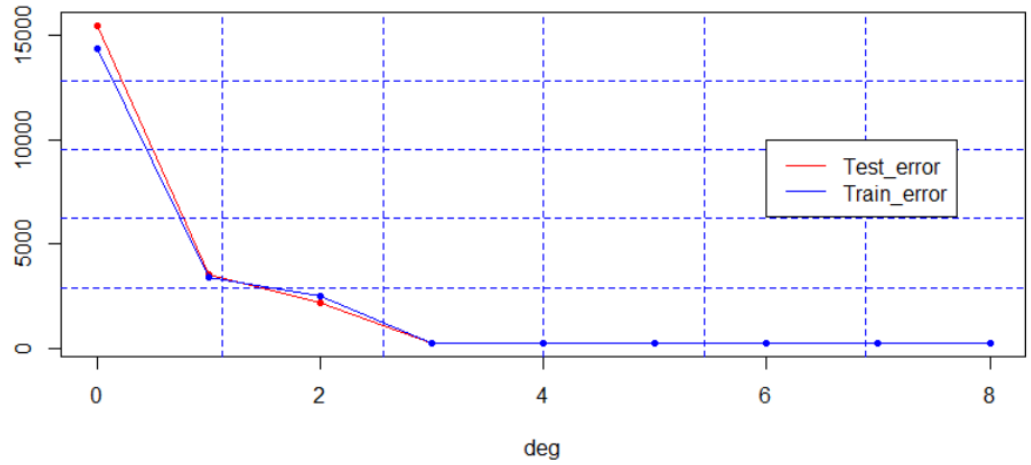
Code to solve w_0 (Where W_fuzhu is the mean of each column of Xpoly) is:

```
W_fuzhu<-vector();
  for(i in 1:(deg+1)){
    W_fuzhu[i]=mean(Xpoly[,i]);
  }
  W0<-mean(y)-t(W)%*%W_fuzhu;
```

Train error\Test error:

deg	*Train error*	*Test error*
0	*15480.52*	*14390.76*
1	*3551.346*	*3393.869*
2	*2167.992*	*2480.725*
3	*252.0461*	*242.8049*
4	*251.4616*	*242.1264*
5	*251.1435*	*239.5449*
6	*248.5828*	*246.0054*
7	*247.011*	*242.8876*
8	*241.4064*	*245.9662*
9	*NA*	*NA*
10	*NA*	*NA*

Among them, when deg is equal to 9/10, Since $\phi^T\phi$ is a singular matrix, the fitting cannot be performed.



(III) Data Analysis

As the above error data analysis shows, when deg is not 1, both the Test_error and the Train_error are decreasing and the downward trend is the same, which means that the polynomial fitting can get better results than the simple linear fitting. At the same time, it can be known that the Test_error will gradually become smaller due to the more complicated linear fitting in the process of deg becoming larger, but the over-fitting phenomenon may be more evident when deg becomes larger. Obviously, Train_error does not necessarily decrease as deg increases.

Regularization

(I) Analysis

The fitting method and process of this paper are clear: firstly, the data is standardized and divided into training set and test set. Then, the regression is carried out by Ridge Regression, and the cross-validation method is used again to find δ^2, which is the best fitting effect.

(II) Specific process

The introduction of data standardization in the text is detailed, so we mainly introduce the method of regularization (Ridge Regression) and cross-validation.

The purpose of the ridge regression is to minimize the value of $\|y - x_\theta\|_2^2 + \delta^2\|\theta\|_2^2$, and derive it to obtain $w = (\delta^2 I + \Phi^T\Phi)^{-1}\Phi^T t$, then use the above formula to find W_0.

The cross-validation method I used in this experiment is the "leave one cross-validation" method. Each group of data in the data set is used as test data, and other data is used as training data set for training, and then predicted by the obtained model, and the formula $CV = \frac{1}{n}\sum_1^n (y_i - \hat{y})^2$ will help us to find the CV corresponding to each δ^2. Finally we will select the minimum value which is the best δ^2.

(III) Code and Result

Code to group data:

```
chouqu<-sample(1:97,50)
X_tr<-X[chouqu,];
y_tr<-y[chouqu];
```

```
chouqu<-chouqu*(-1);
X_te<-X[chouqu,];
y_te<-y[chouqu]
```

Code to normalize the data:

```
#standerdizaion
S<-vector()     #SE
M<-vector()     #Mean
for(a in 1:8){    #standerdize elements
  S[a]<-sd(X_tr[,a]);
  M[a]<-mean(X_tr[,a]);
  X_tr[,a]<-(X_tr[,a]-M[a])/S[a];
}

My<-mean(y_tr);
y_tr<-(y_tr-My);
```

Ridge regression code:

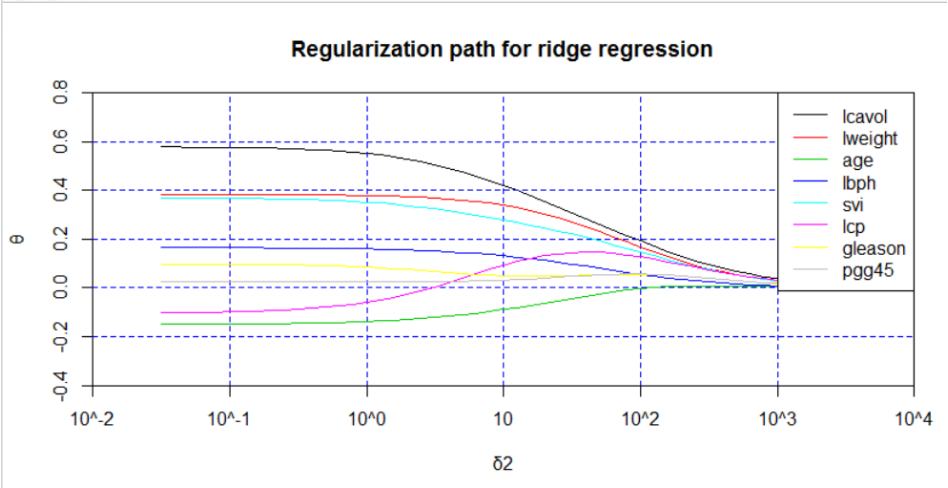
```
# Ridge regression
ridge<-function (X, y, d2){
X_t<-t(X)%*%X;
Nl<-dim(X_t)[2];
X_t<-X_t+diag(Nl)*d2;
W<-solve(X_t)%*%t(X)%*%y;

#Calculate W0
W<-W[,1];
W_fuzhu<-vector();
for(i in 1:8){
W_fuzhu[i]=mean(X[,i]);
}
W0<-mean(y)-t(W)%*%W_fuzhu;
W<-c(W0,W);
W
}
```

Code for plots（1）

```
#Plot
```

```
D2<-seq(-1.5,3.5,0.1)
D2<-10^D2
Der<-matrix(nrow=8,ncol=length(D2));
for(i in 1:length(D2)){
  Der[,i]<-ridge(X_tr,y_tr,D2[i])
}
# Scale the axis
D2_p<-log10(D2)
with(iris, plot(main="Regularization path for ridge regression",xlim=c(-2,4),ylim=c(-0.4,0.8),D2_p,Der[1,],type="l",col=1,pch=20,cex=1,xlab="δ2",ylab="θ",xaxt="n",yaxs="i",xaxs="i"))
grid(nx=6,ny=6,lwd=1,lty=2,col="blue")

for(i in 2:8) lines(D2_p,Der[i,],col=i,type='l',pch=20)
axis(side=1,at = c(-2,-1,0,1,2,3,4),labels = c('10^-2','10^-1','10^0','10','10^2','10^3','10^4'))

#Legend
legend(x=3.0,y=0.8,c('lcavol','lweight','age','lbph','svi','lcp','gleason','pgg45'),col=1:8,lty=1)
```

Note: Request is achieved by taking the log of the x value.



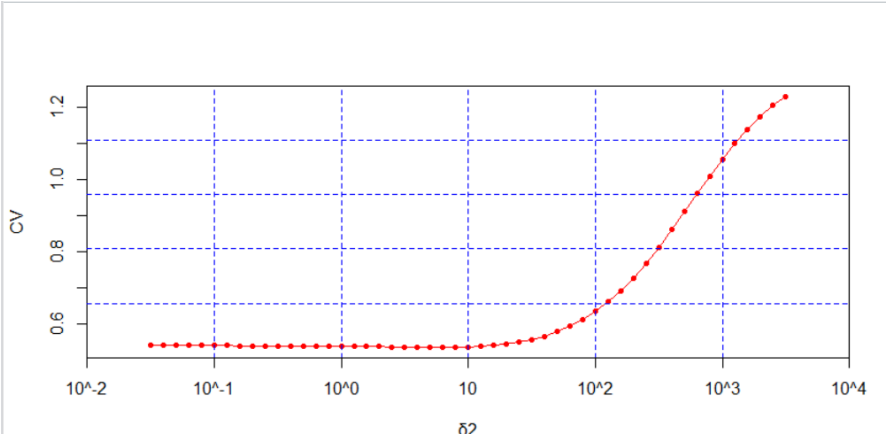
Detailed data of the corresponding Train error \Test error forδ2 is shown in the attached file

CV calculation code:

```
CV<-vector()
for(i in 1:length(D2)){
  tem<-0;
  for(k in 1:97){
    #Separate train dataset
    XX_te<-XX[k,];
    yy_te<-yy[k];
    k<-k*(-1);
    XX_tr<-XX[k,];
    yy_tr<-yy[k];

    #standardize train dataset
    S<-vector()     #SE
    M<-vector()     #Mean
    for(a in 1:8){    #standardize
      S[a]<-sd(XX_tr[,a]);
      M[a]<-mean(XX_tr[,a]);
      XX_tr[,a]<-(XX_tr[,a]-M[a])/S[a];
    }

    Myy<-mean(yy_tr);
    yy_tr<-(yy_tr-Myy);
    #standardize test dataset
    for(a in 1:8){
      XX_te[a]<-(XX_te[a]-M[a])/S[a];
    }

    #Forecast and calculate error
    #Training
    W<-ridge_2(XX_tr,yy_tr,D2[i])
    W0<-W[1];
    W<-W[-1];

    #Forecast
    yy_yu_te[1]<-t(W)%*%XX_te+W0+Myy;

    tem<-tem+(yy_yu_te-yy_te)^2;
```

```
  }
  CV[i]<-tem/97;
}
```

The relationship between CV and δ^2:



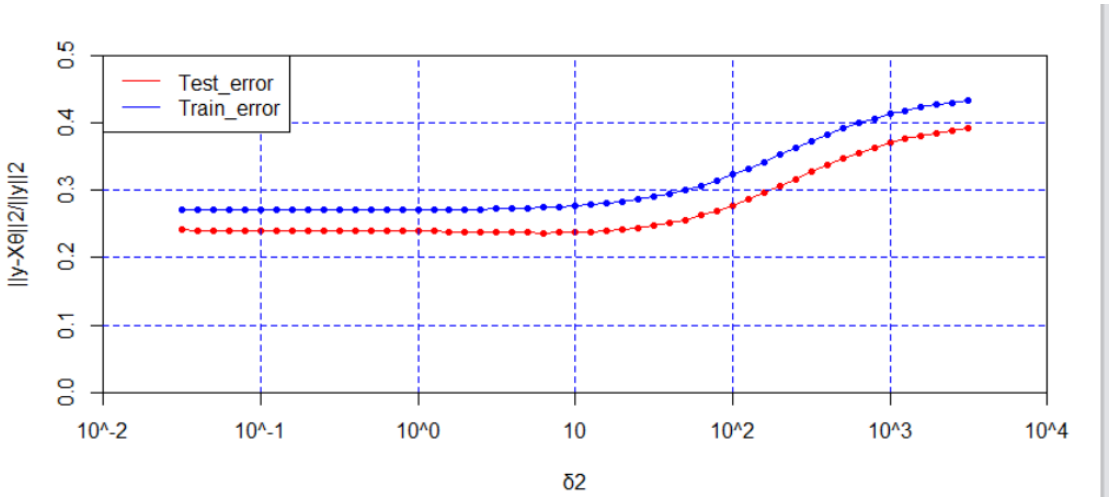
The best δ^2 is:

```
> cat('最合适的δ2为：',D2[which.min(CV)])
最合适的δ2为： 6.309573
```

Code for plots（2）

```
Tr_err<-vector();
Te_err<-vector();

for(k in 1:length(D2)){     # Training and Test set error
  W<-ridge_2(X_tr,y_tr,D2[k])
  W0<-W[1];
  W<-W[-1];

  y_yu_tr<-vector();#Forecast+error
  y_yu_te<-vector();

  for(a in 1:50){
    y_yu_tr[a]<-t(W)%*%X_tr[a,]+W0+My;
  }

  for(a in 1:47){
    y_yu_te[a]<-t(W)%*%X_te[a,]+W0+My;
  }
```

```
  Tr_err[k]<-sqrt(sum((y_yu_tr-y_tr-My)^2))/sqrt(sum((y_tr+My)^2));
  Te_err[k]<-sqrt(sum((y_yu_te-y_te)^2))/sqrt(sum(y_te^2));

}
#Plots
with(iris, plot(ylim=c(0,0.5),xlim=c(-
2,4),D2_p,Te_err,type="o",col=2,pch=20,cex=1,xlab="δ2",ylab="||y-
Xθ||2/||y||2",xaxt="n",xaxs="i",yaxs="i"))
grid(nx=6,ny=5,lwd=1,lty=2,col="blue")

lines(D2_p,Tr_err,col=4,type='o',pch=20)
axis(side=1,at = c(-2,-1,0,1,2,3,4),labels = c('10^-2','10^-
1','10^0','10','10^2','10^3','10^4'))

#Legend
legend(x=-2,y=0.5,c('Test_error','Train_error'),col=c(2,4),lty=1)
```



Annex

When the δ2 is 0.03162278 The training error is: 0.04732565
When the δ2 is 0.03162278 The testing error is: 0.07078023
When the δ2 is 0.03981072 The training error is: 0.0473257
When the δ2 is 0.03981072 The testing error is: 0.0707758
When the δ2 is 0.05011872 The training error is: 0.04732579
When the δ2 is 0.05011872 The testing error is: 0.07077027
When the δ2 is 0.06309573 The training error is: 0.04732592
When the δ2 is 0.06309573 The testing error is: 0.0707634
When the δ2 is 0.07943282 The training error is: 0.04732612
When the δ2 is 0.07943282 The testing error is: 0.07075488
When the δ2 is 0.1 The training error is: 0.04732645
When the δ2 is 0.1 The testing error is: 0.07074436
When the δ2 is 0.1258925 The training error is: 0.04732696
When the δ2 is 0.1258925 The testing error is: 0.07073143
When the δ2 is 0.1584893 The training error is: 0.04732776
When the δ2 is 0.1584893 The testing error is: 0.07071564
When the δ2 is 0.1995262 The training error is: 0.04732902
When the δ2 is 0.1995262 The testing error is: 0.07069654
When the δ2 is 0.2511886 The training error is: 0.04733098
When the δ2 is 0.2511886 The testing error is: 0.07067365
When the δ2 is 0.3162278 The training error is: 0.04733402
When the δ2 is 0.3162278 The testing error is: 0.07064661
When the δ2 is 0.3981072 The training error is: 0.04733873
When the δ2 is 0.3981072 The testing error is: 0.07061526
When the δ2 is 0.5011872 The training error is: 0.04734597
When the δ2 is 0.5011872 The testing error is: 0.07057976
When the δ2 is 0.6309573 The training error is: 0.04735703
When the δ2 is 0.6309573 The testing error is: 0.07054089
When the δ2 is 0.7943282 The training error is: 0.04737378
When the δ2 is 0.7943282 The testing error is: 0.07050029
When the δ2 is 1 The training error is: 0.04739888
When the δ2 is 1 The testing error is: 0.07046083
When the δ2 is 1.258925 The training error is: 0.04743606
When the δ2 is 1.258925 The testing error is: 0.07042694
When the δ2 is 1.584893 The training error is: 0.04749041
When the δ2 is 1.584893 The testing error is: 0.07040481
When the δ2 is 1.995262 The training error is: 0.04756867
When the δ2 is 1.995262 The testing error is: 0.0704024
When the δ2 is 2.511886 The training error is: 0.0476795
When the δ2 is 2.511886 The testing error is: 0.07042895
When the δ2 is 3.162278 The training error is: 0.0478338
When the δ2 is 3.162278 The testing error is: 0.07049399

When the δ2 is 3.981072 The training error is: 0.04804493
When the δ2 is 3.981072 The testing error is: 0.07060583
When the δ2 is 5.011872 The training error is: 0.04832919
When the δ2 is 5.011872 The testing error is: 0.07076996
When the δ2 is 6.309573 The training error is: 0.0487065
When the δ2 is 6.309573 The testing error is: 0.07098795
When the δ2 is 7.943282 The training error is: 0.0492017
When the δ2 is 7.943282 The testing error is: 0.07125786
When the δ2 is 10 The training error is: 0.04984632
When the δ2 is 10 The testing error is: 0.0715767
When the δ2 is 12.58925 The training error is: 0.05068077
When the δ2 is 12.58925 The testing error is: 0.07194514
When the δ2 is 15.84893 The training error is: 0.05175634
When the δ2 is 15.84893 The testing error is: 0.07237382
When the δ2 is 19.95262 The training error is: 0.05313611
When the δ2 is 19.95262 The testing error is: 0.07288968
When the δ2 is 25.11886 The training error is: 0.05489398
When the δ2 is 25.11886 The testing error is: 0.07354059
When the δ2 is 31.62278 The training error is: 0.05711116
When the δ2 is 31.62278 The testing error is: 0.07439669
When the δ2 is 39.81072 The training error is: 0.05987036
When the δ2 is 39.81072 The testing error is: 0.07554751
When the δ2 is 50.11872 The training error is: 0.06324809
When the δ2 is 50.11872 The testing error is: 0.0770948
When the δ2 is 63.09573 The training error is: 0.06730608
When the δ2 is 63.09573 The testing error is: 0.07914194
When the δ2 is 79.43282 The training error is: 0.07208231
When the δ2 is 79.43282 The testing error is: 0.0817807
When the δ2 is 100 The training error is: 0.07758188
When the δ2 is 100 The testing error is: 0.08507666
When the δ2 is 125.8925 The training error is: 0.08376811
When the δ2 is 125.8925 The testing error is: 0.08905455
When the δ2 is 158.4893 The training error is: 0.09055567
When the δ2 is 158.4893 The testing error is: 0.09368634
When the δ2 is 199.5262 The training error is: 0.09780861
When the δ2 is 199.5262 The testing error is: 0.0988854
When the δ2 is 251.1886 The training error is: 0.1053466
When the δ2 is 251.1886 The testing error is: 0.1045102
When the δ2 is 316.2278 The training error is: 0.1129602
When the δ2 is 316.2278 The testing error is: 0.1103785
When the δ2 is 398.1072 The training error is: 0.1204337
When the δ2 is 398.1072 The testing error is: 0.11629
When the δ2 is 501.1872 The training error is: 0.1275687
When the δ2 is 501.1872 The testing error is: 0.1220512

When the δ2 is 630.9573 The training error is: 0.1342042
When the δ2 is 630.9573 The testing error is: 0.1274975
When the δ2 is 794.3282 The training error is: 0.1402284
When the δ2 is 794.3282 The testing error is: 0.1325062
When the δ2 is 1000 The training error is: 0.1455805
When the δ2 is 1000 The testing error is: 0.137002
When the δ2 is 1258.925 The training error is: 0.1502461
When the δ2 is 1258.925 The testing error is: 0.1409528
When the δ2 is 1584.893 The training error is: 0.1542473
When the δ2 is 1584.893 The testing error is: 0.1443627
When the δ2 is 1995.262 The training error is: 0.1576311
When the δ2 is 1995.262 The testing error is: 0.1472611
When the δ2 is 2511.886 The training error is: 0.1604597
When the δ2 is 2511.886 The testing error is: 0.1496935
When the δ2 is 3162.278 The training error is: 0.1628014
When the δ2 is 3162.278 The testing error is: 0.1517136