

MDTS4214_739_KNN regression

Debkanta Ghosh

2026-02-26

Question 1

Consider a setting with

$$n = 1000$$

observations generated as follows:

$$x_{1i} \sim N(0, 2^2), \quad x_{2i} \sim \text{Poisson}(\lambda = 1.5),$$

$$\varepsilon_i \sim N(0, 1),$$

$$y_i = -2 + 1.4x_{1i} - 2.6x_{2i} + \varepsilon_i.$$

```
rm(list=ls())
set.seed(123)
n=1000
x1=rnorm(n,0,2)
x2=rpois(n,1.5)
e=rnorm(n)
y=-2+1.4*x1-2.6*x2+e
```

The first 800 observations are used as the training set and the remaining 200 observations are used as the test set.

```
train.data=data.frame(y[1:800],x1[1:800],x2[1:800])
colnames(train.data)=c("y","x1","x2")
test.data=data.frame(y[801:1000],x1[801:1000],x2[801:1000])
colnames(test.data)=c("y","x1","x2")
#Train Data Overview
head(train.data)

##          y      x1  x2
## 1 -4.3903185 -1.1209513  0
## 2 -2.9517542 -0.4603550  0
## 3  1.4622853  3.1174166  0
## 4 -3.7755078  0.1410168  1
## 5 -3.1176393  0.2585755  1
## 6 -0.2706045  3.4301300  2

#Test Data overview
head(test.data)
```

```

##          y      x1 x2
## 1 -0.7945918 0.7125667 0
## 2 -6.6277609 -1.3160204 1
## 3 -2.1739932 1.7104044 1
## 4  1.6395734 2.3058725 0
## 5 -7.2026151 0.5525491 2
## 6 -5.6568293 0.2882093 2

```

(a)

Fit a multiple linear regression model of y on x_1 and x_2 . Calculate the test Mean Squared Error (MSE).

```

#1 Linear regression
fit1=lm(y~x1+x2,data=train.data)
summary(fit1)

##
## Call:
## lm(formula = y ~ x1 + x2, data = train.data)
##
## Residuals:
##    Min     1Q   Median     3Q    Max 
## -3.0727 -0.6573 -0.0125  0.6921  3.2412 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.07300   0.05382 -38.52   <2e-16 ***
## x1          1.38207   0.01767  78.21   <2e-16 ***
## x2         -2.55584   0.02768 -92.34   <2e-16 ***
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.98 on 797 degrees of freedom
## Multiple R-squared:  0.9492, Adjusted R-squared:  0.9491 
## F-statistic: 7445 on 2 and 797 DF,  p-value: < 2.2e-16

```

Fitted model is given by,

$$\hat{y} = -2.07300 + 1.38207 x_1 - 2.55584 x_2$$

with multiple R-squared = 0.9492. So this is a very good fit.

Calculate the test Mean Squared Error (MSE).

```

y.hat.test=predict(fit1,newdata=test.data)
mse.test=mean((test.data$y-y.hat.test)^2)
mse.test

## [1] 0.998901

```

Test MSE comes out as 0.998901 which is very low, implicating that the model is good. ### (b)
Fit a K-Nearest Neighbours (KNN) regression model for

$$k = 1,2,5,9,15.$$

Calculate the test MSE for each value of k .

```

k=c(1,2,5,9,15)
library(FNN)
X.train=matrix(c(x1[1:800],x2[1:800]),byrow=F,ncol=2)
X.test=matrix(c(x1[801:1000],x2[801:1000]),byrow=F,ncol=2)
f=function(k)
{
  pred=knn.reg(train=X.train,test=X.test,y=train.data$y,k)
  mean((test.data$y-pred$pred)^2)
}
test_mse=c(f(1),f(2),f(5),f(9),f(15))
cbind(k,test_mse)

##          k test_mse
## [1,] 1 2.219793
## [2,] 2 1.729587
## [3,] 5 1.303978
## [4,] 9 1.205371
## [5,] 15 1.235776

```

For five choices of K , we perform KNN regression on y and observe that for each choice of K , the MSE is low as well. So KNN regression in this case does not provide an improvement over linear regression.

Question 2

Now suppose the response variable is generated according to the nonlinear model

$$y_i = \frac{1}{-2 + 1.4x_{1i} - 2.6x_{2i} + 2.9x_{1i}^2} + 3.1\sin(x_{2i}) - 1.5x_{1i}x_{2i}^2 + \varepsilon_i.$$

```

y.new=1/(-2+1.4*x1-2.6*x2+2.9*x1*x1)+3.1*sin(x2)-1.5*x1*x2*x2+e
train.data.new=data.frame(y.new[1:800],x1[1:800],x2[1:800])
colnames(train.data.new)=c("y","x1","x2")
test.data.new=data.frame(y.new[801:1000],x1[801:1000],x2[801:1000])
colnames(test.data.new)=c("y","x1","x2")
head(train.data.new) #Train data overview

##          y      x1      x2
## 1 12.581968 -1.1209513 0
## 2 -0.799890 -0.4603550 0
## 3 -0.869362  3.1174166 0
## 4  2.793949  0.1410168 1

```

```

## 5 3.093778 0.2585755 1
## 6 -15.603221 3.4301300 2

head(test.data.new) #test data overview

##          y      x1 x2
## 1 2.3351481 0.7125667 0
## 2 3.6929787 -1.3160204 1
## 3 0.2336686 1.7104044 1
## 4 0.4714204 2.3058725 0
## 5 -1.4531284 0.5525491 2
## 6 2.0767028 0.2882093 2

```

(a)

Fit a multiple linear regression model of y on x_1 and x_2 .
Calculate the test Mean Squared Error (MSE).

```

fit3=lm(y~x1+x2,data=train.data.new)
summary(fit3)

##
## Call:
## lm(formula = y ~ x1 + x2, data = train.data.new)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -241.819 -5.150  -0.051   5.566 155.662 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -0.2369    1.0272  -0.231  0.81764    
## x1          -6.3747    0.3373 -18.901 < 2e-16 ***
## x2           1.3849    0.5283   2.621  0.00892 ** 
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 18.7 on 797 degrees of freedom
## Multiple R-squared:  0.3146, Adjusted R-squared:  0.3129 
## F-statistic: 182.9 on 2 and 797 DF,  p-value: < 2.2e-16

```

Fitted model:

$$\hat{y} = -0.2369 - 6.3747 x_1 + 1.3849 x_2$$

with multiple R-squared = 0.3146 which means this fit is not good.

Then we find out the test mse

```

y.hat.test.new=predict(fit3,newdata=test.data.new)
mse.test.new=mean((test.data.new$y-y.hat.test.new)^2)
mse.test.new

```

```
## [1] 205.1776
```

Test MSE = 205.1776 means the model is not good.

(b)

Fit K-Nearest Neighbours regression models for

$$k = 1,2,5,9,15.$$

Calculate the test MSE for each value of k .

```
library(FNN)
X.train.new=matrix(c(x1[1:800],x2[1:800]),byrow=F,ncol=2)
X.test.new=matrix(c(x1[801:1000],x2[801:1000]),byrow=F,ncol=2)
f2=function(k){
  pred.new=knn.reg(train=X.train.new,test=X.test.new,y=y.new[1:800],k)
  mean((test.data.new$y-pred.new$pred)^2)
}
mse.knn.new=c(f2(1),f2(2),f2(5),f2(9),f2(15))
cbind(k,mse.knn.new)

##      k mse.knn.new
## [1,]  1   47.52490
## [2,]  2   54.48942
## [3,]  5   59.77963
## [4,]  9   62.10913
## [5,] 15   63.72010
```

Here we observe that test MSE has been reduced drastically due to KNN regression compared to multiple linear regression.

(c)

Compare the performance of linear regression and KNN regression models and comment on the results.

Parametric models perform well when the assumed functional form is correctly specified. However, when the true relationship between predictors and the response variable is nonlinear, nonparametric methods such as K-nearest neighbours regression often provide superior predictive performance.