10FoldCV

Ye

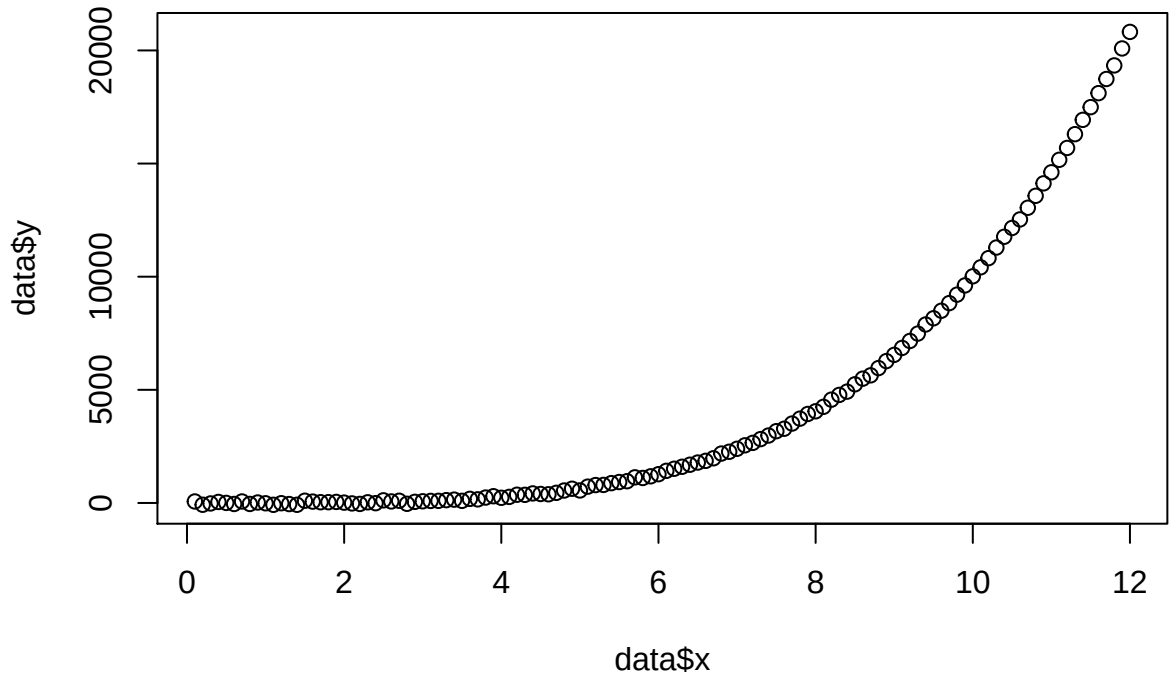
7/21/2017

Generate dataset with polynormial degree = 4

```
set.seed(2333)
N = 120
tF <- function(x) x^4
X = 0.1*(1:N)
sigma = 40
data = data.frame(y= tF(X) + rnorm(N,sd=sigma),x=X)
head(data)
```

```
##            y   x
## 1  65.435915 0.1
## 2 -79.227158 0.2
## 3 -21.557846 0.3
## 4  44.910880 0.4
## 5   1.663197 0.5
## 6 -44.952580 0.6
```

```
plot(data$x, data$y)
```




```
print(mean(data$y)/sigma)
```

```
## [1] 105.8139
```

The SNR (signal-to-noise ratio) is defined as the ratio of signal mean to noise standard deviation.

Perform the k-fold cross validation with k = 10

```
set.seed(2111)
nEval = 10  # last nEval values - evaluation set
n = N - nEval # n values in train set
xTrain = data[c(sample(n),(n+1):N),] # reshuffle train set, keep test set unchanged
nFold = 10
resCV = numeric(nFold)
(testSize = floor(n/nFold))
```

```
## [1] 11
```

```
for(k in 1:6) {   # k is the degree of fitted polynomial
  if(k>1) {
    xTrain = cbind(xTrain,xTrain$x^k)  #create polynomial of degree k
    names(xTrain)[ncol(xTrain)] = paste0('x',k)
  }
  for(i in 1:nFold) {
    # select train and test sets
    testInd = (1+(i-1)*testSize):(i*testSize)  #make test fold
    train = xTrain[(1:n)[-testInd],]  #exclude test fold
    test = xTrain[testInd,]
    model <- lm(y~.,data=train)
    resCV[i] = sum((predict(model,test)-test$y)^2)
  }
  cat(k,mean(resCV),'\n')  #like print, but more efficient
}
```

```
## 1 44526554
## 2 3777896
## 3 87244.2
## 4 21182.93
## 5 21066.29
## 6 21614.55
```

The conclusion is that the cross validation did not prevent overfitting in test, which will lead to large error especially when extrapolating the data.