ST340 Lab 6: Validation and the curse of dimensionality

2020–21

Validation

The dataset `SmokeCancer.csv` shows lung cancer rates by U.S. state in 2010, with a number of covariates such as Federal Year 2010 cigarette sales per 100,000.

(a) Read the data file on lung cancer and create a data frame with variables of interest.

```
X = read.table("data/SmokeCancer.csv", header=TRUE,sep=",",row.names=1)
LungCancer = data.frame(CigSalesRate=100000*X[,"FY2010Sales"]/X[,"Pop2010"],
                        X[,c("CigYouthRate","CigAdultRate","LungCancerRate")])
```

(b) Fit a linear model for LungCancerRate:

```
summary(lm(LungCancerRate~CigSalesRate+CigYouthRate+CigAdultRate,data=LungCancer))
```

(c) Write a function that takes a formula and does LOOCV (leave one out cross validation) with respect to the squared error of the linear model for the given formula. Use it to find a good linear model for `LungCancerRate` in terms of `CigSalesRate`, `CigYouthRate` and `CigAdultRate`. You could also try using transformations of the covariates by adding terms such as `I(CigSalesRate^2)` and `I(CigSalesRate*CigAdultRate)` to your formulae.

(By good, we mean that it is the optimal, in terms of cross-validation error, linear model using some or all of these covariates.)

```
LOOCV = function(formula, data){

  n = nrow(data)

  CV_values = vector(length = n)

  for(i in 1:n){
    data = data[-i,]
    model = lm(formula, data = data)
    CV_values[i] = sum((model$residuals)^2)
  }

  return(mean(CV_values))

}
```

```
LOOCV(data = LungCancer, formula = LungCancerRate ~ CigSalesRate + CigYouthRate + CigAdultRate)
```

```
## [1] 897.6543
```

(d) The Akaike Information criterion (AIC) and Bayesian Information criterion (BIC) are analytic approximations to the validation step. They are (different) ways of quantifying the trade-off between model complexity (in terms of, e.g. the number of parameters) and the fit to the training data (in terms of likelihood), defined as follows:

- Akaike Information criterion (AIC) = $(2 \times \#\text{parameters} - 2 \times \log(\text{likelihood}))$, and
- Bayesian information criterion (BIC) = $(\log(\text{amount of data}) \times \#\text{parameters} - 2 \times \log(\text{likelihood}))$.

  Write a function that takes a formula and then calculates AIC and BIC. Use your function to find a *good* linear model for `LungCancerRate`, as in (b).
  After plugging in the maximum likelihood estimators into the log likelihood we get
  AIC: $n \log\left(\frac{\text{RSS}}{n}\right) + 2p$
  BIC: $n \log\left(\frac{\text{RSS}}{n}\right) + p\log(n)$

```
AIC_BIC = function(formula, data){

  n = nrow(data)
  p = length(labels(terms(formula)))

  model = lm(formula, data = data)
  RSS   = sum((model$residuals)^2)

  AIC = n*(log(RSS/n)) + 2*p
  BIC = AIC - (2*p) + (p*log(n))

  return(list(AIC = AIC,BIC = BIC))

}
```

```
AIC_BIC(formula = LungCancerRate ~ CigSalesRate + CigYouthRate + CigAdultRate, data = LungCancer)
```

```
## $AIC
## [1] 172.3765
##
## $BIC
## [1] 178.172
```

```
LOOCV(formula = LungCancerRate ~ CigSalesRate + CigYouthRate + CigAdultRate, data = LungCancer)
```

```
## [1] 897.6543
```

The curse of dimensionality

Suppose N points are chosen uniformly at random in the D-dimensional hypercube $[0,1]^D$. Consider a smaller hypercube $H = [0,r]^D$ in the "corner" of $[0,1]^D$.

(a) How big does r have to be for there to be approximately one of the N points lying in H?

we want $\mathbb{E}[\{x \in [0,r]^D\}] = 1 \quad \iff \quad N\mathbb{P}(\{x \in [0,r]^D\}) = 1$
$\implies \mathbb{P}(x \le r)^D = \frac{1}{N}$
$\implies r = (\frac{1}{N})^{\frac{1}{D}}$

(b) How big does r have to be for there to be approximately 10 of the N points lying in H?
$r = (\frac{10}{N})^{\frac{1}{D}}$

(c) How big does r have to be for there to be approximately $\frac{N}{2}$ of the N points lying in H?
$r = \left(\frac{1}{2}\right)^D$

Check each of your answers by simulation.

Distance functions

(a) Write a function to calculate the ℓ_1 distances between pairs of row vectors in two matrices:

```
distances.l1 <- function(X,W) {

  xn = nrow(X); wn = nrow(W)

  l1.dist = matrix(nrow = xn, ncol = wn)

  rownames(l1.dist) = paste0('X',1:xn)
  colnames(l1.dist) = paste0('W',1:wn)

  for(i in 1:xn){
    for(j in 1:wn){

      l1.dist[i,j] = sum(abs(X[i,] - W[j,]))

    }
  }

  return(l1.dist)
}
```

(b) Write a similar function to calculate a matrix of pairwise ℓ_2 distances:

```
distances.l2 <- function(X,W) {


  xn = nrow(X); wn = nrow(W)

  l2.dist = matrix(nrow = xn, ncol = wn)

  rownames(l2.dist) = paste0('X',1:xn)
  colnames(l2.dist) = paste0('W',1:wn)

  for(i in 1:xn){
    for(j in 1:wn){

      l2.dist[i,j] = sqrt(sum((X[i,] - W[j,])^2))

    }
  }

  return(l2.dist)





}
```

(c) Write a similar function to calculate the Mahalanobis distance between the row vectors, given a $D \times D$ covariance matrix S:

```
distances.maha <- function(X,W,S) {
```

```r
xn = nrow(X); wn = nrow(W)

  mahal.dist = matrix(nrow = xn, ncol = wn)

  rownames(mahal.dist) = paste0('X',1:xn)
  colnames(mahal.dist) = paste0('W',1:wn)

  for(i in 1:xn){
    for(j in 1:wn){

      mahal.dist[i,j] = sqrt(t(X[i,] - W[j,])%*%(solve(S))%*%((X[i,] - W[j,])))

    }
  }

  return(mahal.dist)

}
```