Algorithms_II_Lab-02

August 18, 2021

0.1 Lab02

Class will focus on learning the probability of heads from a series of coin flips using the maximum likelihood approach. We will learn how to simulate a series of coin flips, program the loglikelihood function into R, optimize this likelihood and approximate the distribution of maximum likelihood estimates for the probability of heads using the Fisher Information Matrix.

0.1.1 Simulate a coin flip using the `rbinom` function

We will simulate a coinflip using the `rbinom` fucntion. Recall that the Binomial distribution or Binom(N,θ) assigns probabilities to the values 0,1,2,. . . ,N.

The probability mass function (pmf) for the Binomial distribution is

$$p(x|N,\theta) = \binom{N}{x}\theta^{x}(1-\theta)^{N-x} \tag{1}$$

and the pmf for the Bernoulli is θ when $x = 1$ and $1-\theta$ when $x = 0$. We can write this pmf in a single line like this

$$p(x|\theta) = \theta^{x}(1-\theta)^{1-x} \tag{2}$$

By setting the Binomial distribution parameter $N = 1$, we have the following pmf

$$\begin{align} p(x|N=1,\theta) &= \binom{1}{x}\theta^{x}(1-\theta)^{1-x} \tag{3}\\ &= \theta^{x}(1-\theta)^{1-x} \tag{4}\end{align}$$

where $\binom{1}{x}$ counts the number of ways to return 1 item (which is equal to one).

This means we can simulate values from a Bern(θ) distribution by simulating from a Binom($1,\theta$) distribution. In R, we can simulate values from a Binomial distribution using the `rbinom` function,

```
[ ]: rbinom(size=10,p=0.25,n=30)
```

where the input **size** corresponds to the parameter N, the input p corresponds to θ, and n controls the number of values you wish to generate.

Lets create a function that simulates N coin flips with a probability p os heads and probability $1-p$ of tails.

```
[ ]: simulateNflips = function(numflips,p){
       samples = rbinom(n=numflips,size=1,p=p)
       return(samples)
     }
```

```
[ ]: simulateNflips(100,0.45)
```

Now that we have a convienant function, lets simulate 1,000 coin flips with a prob of 0.25 for heads and store this vector in a variable called **samples**.

```
[ ]: samples = simulateNflips(1000,0.25)
```

0.1.2 The loglikelihood function for the Bernoulli distribution

For a single sample For a single data point, if we knew the value of θ we would assign the foloowing probability

$$p(x|\theta) = \theta^x (1-\theta)^{1-x} \tag{5}$$

In other words, if we saw the value 1 we would assign that the probability θ and if we saw 0 we would assign that value a $1-\theta$ probability of occuring.

However, for a given data point we can view the above probability as a function of θ

$$g(\theta) = p(x|\theta) = \theta^x (1-\theta)^{1-x} \tag{6}$$

This function is called the likelihood. The log likelihood is then

$$\begin{aligned} \log(g(\theta)) &= \log[\theta^x (1-\theta)^{1-x}] && (7) \\ &= x\log(\theta) + (1-x)\log(1-\theta) && (8) \end{aligned}$$

For N samples For N data points that we represent as $x_1, x_2, \cdots, x_N$ where x_i is a 1 or 0, the log likelihood becomes

$$p(x_1, x_2, \cdots, x_N|\theta) = \left[\theta^{x_1}(1-\theta)^{1-x_1}\right]\left[\theta^{x_2}(1-\theta)^{1-x_2}\right]\cdots\left[\theta^{x_N}(1-\theta)^{1-x_N}\right] \tag{9}$$

$$\begin{aligned} \log\left[g(\theta)\right] &= \log\left\{\left[\theta^{x_1}(1-\theta)^{1-x_1}\right]\left[\theta^{x_2}(1-\theta)^{1-x_2}\right]\cdots\left[\theta^{x_N}(1-\theta)^{1-x_N}\right]\right\} && (10) \\ &= \sum_{i=1}^{N} x_i\log(\theta) + (1-x_i)\log(1-\theta) && (11) \\ &= \log(\theta)\sum_{i=1}^{N} x_i + \log(1-\theta)\left(N - \sum_{i=1}^{N} x_i\right) && (12) \\ & && (13) \end{aligned}$$

Lets spend time implementing a function called `loglikelihood` that takes as input two variables: a single number between 0 and 1 called `theta` and a vector of 0s and 1s called `samples`. This function will return the loglikelihood above.

```
[ ]: loglikelihood = function(theta,samples){
       N = length(samples)
       numberOfOnes  = sum(samples)
       numberOfZeros = N-numberOfOnes
       LL = numberOfOnes*log(theta) + numberOfZeros*log(1-theta)
       return(LL)
     }
```

0.1.3 Visualizing the loglikelihood function for a sample of data

We can plot the loglikelihood for values of theta from 0 to 1 by 0.01 to get a sense of where the loglikelihood is highest. Because we simulated our data `samples` using $\theta = 0.25$ we expect that the loglikelihood will be highest around 0.25.

```
[ ]: thetas = seq(0,1,0.01)
     LLs = loglikelihood(thetas,samples)
     plot(thetas,LLs)
```

0.1.4 Finding the maximum of the loglikelihood using an optimization routine

To find the θ that maximizes the loglikelihood function we will use the R function called `optimize`. The optimize function takes as input: a function of one variable that we wish to optimize and an interval of potential values that will maximize this function.

First, we need to give the `optimize` function our `loglikelihood` fucntion. But we have a problem. The loglikelihood function we created takes two inputs: θ and `samples`. We need to create a new function (lets call it `LL`), often called a **wrapper**, that takes as input only theta and returns the loglikehood.

```
[ ]: LL = function(theta){
       return(loglikelihood(theta,samples))
     }
```

We can now call the optimize function and input LL. The second argument is the interval of possible values that could optimize this loglikelihood. The possible values of θ are in the interval from 0 to 1, and so we can provide the optimize function this inverval. Finally, the default behavior of the optimize function is to find a minimum value, not maximum. We can change this behavior by providing the optimize function one optional input, maximum, that is either TRUE or FALSE.

```
[ ]: optimize(LL, interval=c(0,1),maximum=TRUE)
```

The optimize function returns a new object we have not yet seen in R—a **list**. This list contains the maximum θ value and also the loglikelihood value for this maximum θ value.

0.1.5 The List (an R programming aside)

In R, a list can be used to store items of different types and reference them with a key. To create a list, use the list function and include as many R objects as you like.

For example, we can create a list called L that contains a a key "A" that correpsonds to a string "B" and a key "C" that corresponds to a vector. Keys need to be strings.

```
[ ]: L = list("A"="B","C" = c(1,2,3))
     L
```

Items in the list can be accessed by naming the list and enclosing two sets of square brackets around a key.

```
[ ]: L[["A"]]
```

```
[ ]: L[["C"]]
```

An alternative way to access a list is by using the $ operator. Name the list, type a $, and then the key.

```
[ ]: L$C
```

We now see that the optimize function returns a list of two keys

```
[ ]: O = optimize(LL, interval=c(0,1),maximum=TRUE)
     O
```

The first key is called "maximum" and returns the maximum theta for our log likelihood, in other words our maximum likelihood estiamtor of the parameter θ. The second key returns the loglikelihood value for that maximum θ.

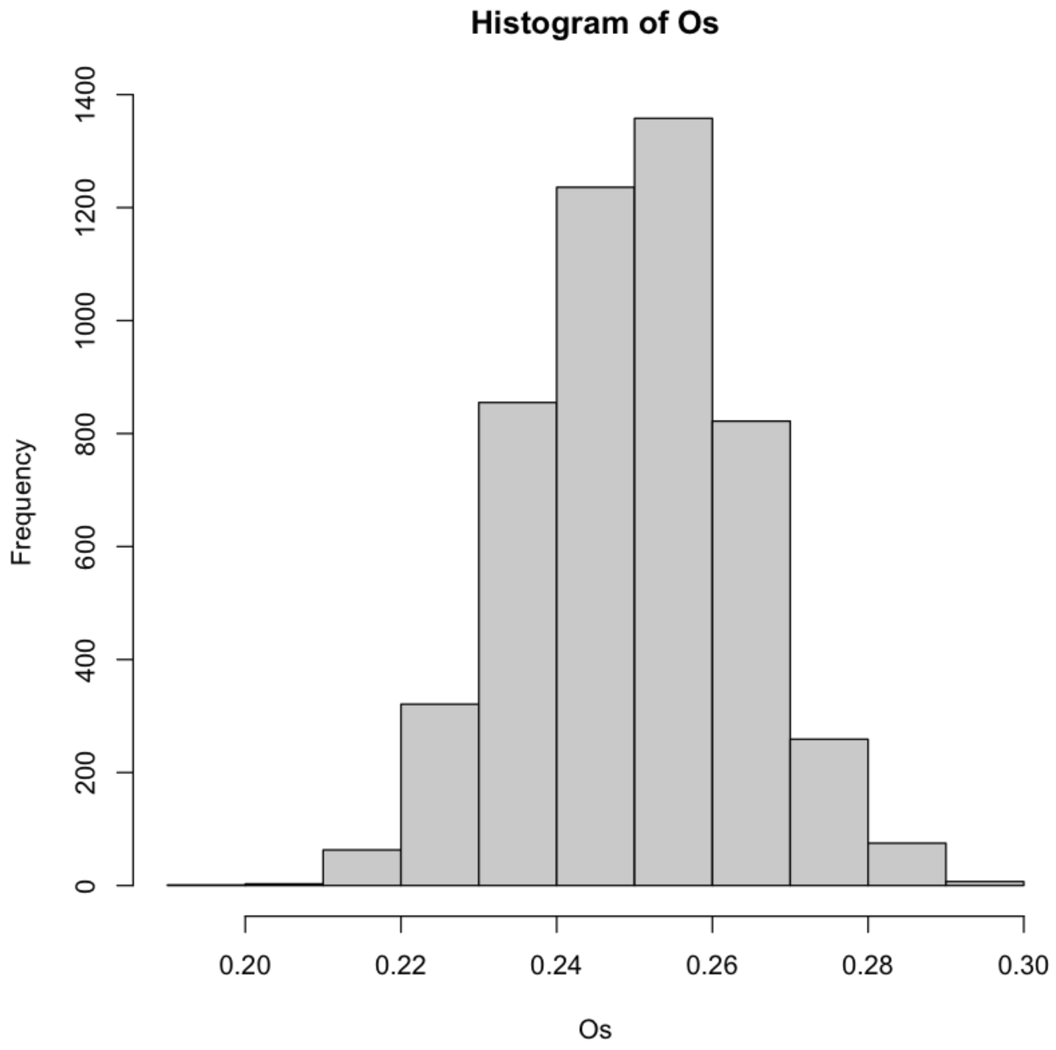
```
[ ]: mle = O$maximum
     mle
```

0.1.6 Variance of our MLE

Let's explore the variability in our maximum likelihood estimator (MLE) by simulating 1000 coin flips 5000 times, recording the MLE, and then plotting a histogram of the MLE values.

```
[51]: Os = rep(0,5000)                     # Create a vector of 5,000 zeros that will␣
       ↪store our MLEs
      for(i in 1:5000){                    # Run a loop 5,000 times
        samples = simulateNflips(1000,0.25) # Flip 1,000 coins with a prob of heads of␣
       ↪0.25
        LL = function(theta){              # build a function that incorporates the␣
       ↪data above
          return(-1*loglikelihood(theta,samples))
        }
```

```
  O = optimize(LL, interval=c(0,1))   # Optimize!
  Os[i] = O$minimum                   # Whoa, minimum? what happened here? Hint:␣
 ↪look up three lines
}
hist(Os)                              #Plot a histogram of our 5,000 MLEs
```



We see that the MLE is centered near $\theta = 0.25$ but does have some variability around this θ value. The shape of this histogram likes awfully "Normal". This is because we learned in class that the MLE has the following sampling distribution

$$\theta_{\text{MLE}} \sim \mathcal{N}\left(\theta, \mathcal{I}^{-1}\right) \tag{14}$$

where θ is the mean parameter and $\mathcal{I}/N$ the variance around the MLE.

We learned in class that the Fisher information for the Bernoulli distribution for a single data point is

$$\mathcal{I}(\theta) = \frac{1}{\theta(1-\theta)} \tag{15}$$

and or N data points is

$$\mathcal{I}(\theta) = \frac{N}{\theta(1-\theta)} \tag{16}$$

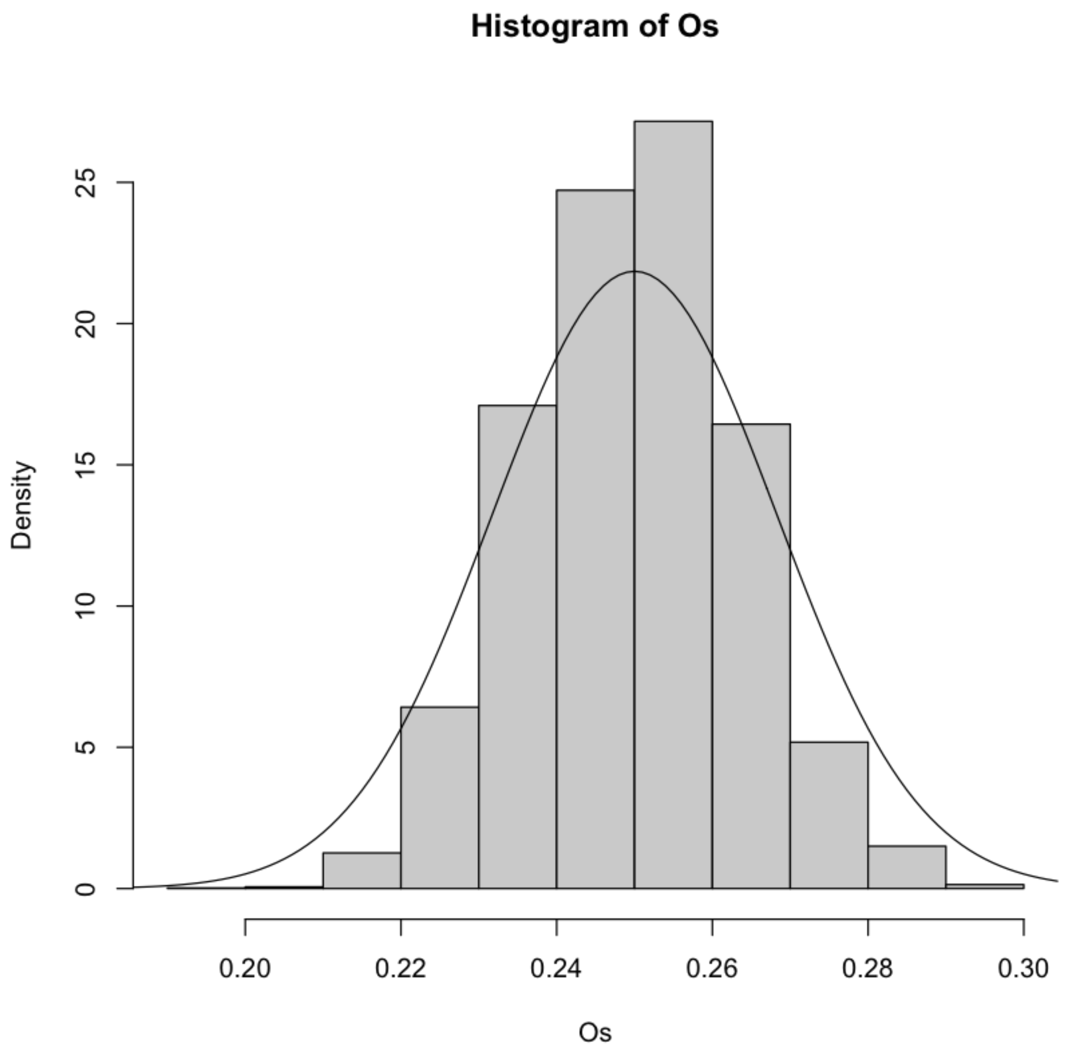
Lets plot our empricial distribution of MLEs and also the above normal distribution to see how well they line up.

[55]:
```
fisherInfo = function(MLE,samples){
  N = length(samples)
  return(N/MLE*(1-MLE)) # This is the Fisher info for N samples
}

thetas = seq(0,1,0.001)

# This returns the density of a Normal distribution with mean, and sd for all␣
 ↪the x values
dens = dnorm(x = thetas, mean = 0.25, sd = sqrt(1/fisherInfo(0.25,samples)))

hist(Os,probability=TRUE) # The options probability=TRUE estimates a probability␣
 ↪density from your histogram
lines(thetas,dens)
```


Histogram of Os
Density
0
5
10
15
20
25
0.20
0.22
0.24
0.26
0.28
0.30
Os