Maximum Likelihood Estimators Assignment

Naveen Narayanan Meyyappan

18/11/2019

In statistics, maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model given observations, by finding the parameter values that maximize the likelihood of making the observations given the parameters. MLE can be seen as a special case of the maximum a posteriori estimation (MAP) that assumes a uniform prior distribution of the parameters, or as a variant of the MAP that ignores the prior and which therefore is unregularized.

In statistics, the likelihood function (often simply called the likelihood) expresses how probable a given set of observations is for different values of the statistical parameters.

The intuition behind maximizing a likelihood function is that it maximizes the likelihood that observed sample data came from true population. As we maximize this likelihood, parameters estimated using sampled data approach closest to the unknown population parameters.

Most of the time we take log of the likelihood function because it reduces the scale and makes it simpler to work with the data. The likelihood function involves multiplication of different data points. On taking log, the function involves addition rather than multiplication and makes it much easier to work with.

So maximizing the likelihood function is equivalent to maximizing its log-likelihood function.

In order to find the maximum of a function we take the first derivative and equate it zero. This is the point where the function achieves its local maximum or minimum.

In our approach for estimating the maximum likelihood parameters for a given distribution, we try to minimize the negative log-likelihood function.

Now let us take different theoretical probability distributions and try to find the maximum likelihood estimators

### Binomial Distribution

Binomial Distribution (size=10, p=0.5)

loglike.binomial = function(theta, x) # This function generates the first derivative of negative log likelihood for the given binomial distribution. dbinom function is the first derivative for the binomial distribution.  
{  
 ans <- sum(dbinom(x, theta[1], theta[2], log=TRUE))  
 return(-ans)   
}  
n = 1000  
x=rbinom(n,10,0.5)  
mu=mean(x)  
variance=var(x)  
q=variance/mu  
p=1-q  
size=as.numeric(round(mu/p))  
theta=c(size,p) #The binomial distribution has two parameter: size, p  
ans = optim(par=theta, fn=loglike.binomial, x=x) # optimize the likelihood function  
ans$par # Estimated parameter n and p for the given binomial distribution using likelihood function

## [1] 10.0000010 0.5023999

theta # Estimate the parameters using method of moments

## [1] 10.0000000 0.4984188

### Geometric Distribution

Geometric Distribution (p=0.2)

loglike.geometric = function(theta, x) #This function generates the negative log likelihood of the geometric distribution  
{  
 ans <- sum(dgeom(x, theta[1], log=TRUE))  
 return(-ans)   
}  
n = 1000  
x=rgeom(n,0.2) #Generates a random geometric distribution with p = 0.2  
p=1/mean(x)  
theta=c(p)  
ans = optim(par=theta, fn=loglike.geometric, x=x) #Optimizing the negative log likelihood function  
ans$par # The parameters estimated using MLE

## [1] 0.21494

theta # Parameters estimated using Method of Moments

## [1] 0.2738226

### Poisson Distribution

Poisson Distribution (lambda = 0.5)

loglike.poisson = function(theta, x) # This function generates the first derivative of negative log likelihood for the given Poisson distribution. dpois function is the first derivative for the Poisson distribution.  
{  
 ans <- sum(dpois(x, theta[1], log=TRUE))  
 return(-ans)   
}  
n = 1000  
x=rpois(n,0.5)  
lambda=mean(x)  
theta=c(lambda) #The Poisson distribution has only one parameter, lambda  
ans = optim(par=theta, fn=loglike.poisson, x=x, method="BFGS") # optimize the likelihood function   
ans$par # Estimated parameter lambda for the given Poisson distribution using likelihood function

## [1] 0.502

theta # Estimated parameter lambda for the given Poisson distribution using method of moments

## [1] 0.502

### Normal Distribution

Normal Distribution (mean=3, standard deviation=4)

loglike.normal = function(theta, x) # This function computes the first derivative of the negative log - likelihood function. dnorm is the first derivative of the normal function.  
{  
 ans = sum(dnorm(x, theta[1], theta[2], log=TRUE))  
 return(-ans)   
}  
n = 1000  
x = rnorm(n, mean=3, sd=4) # Creating a normal random distribution with mean = 3 and standard deviation = 4  
mu=mean(x)   
standarddeviation=sd(x)  
# The mean and standard deviation of the sample are sent as parameters for the distribution (theta)  
theta = c(mu,standarddeviation)  
ans = optim(par=theta, fn=loglike.normal, x=x, method="BFGS") # We try to minimize the loglike.normal function and find the parameters which satisfy the optimization condition (minimum of negative log-likelihood function)  
# The mean and standard deviation estimated using likelihood function   
ans$par

## [1] 2.941456 4.107735

# The mean and standard deviation estimated using method of moments  
theta

## [1] 2.941456 4.109791

### Exponential Distribution

Exponential Distribution (rate = 5)

loglike.exponential = function(x, theta) # Generating the negative log-likelihood function. dexp denotes the first derivative of the exponential distribution  
{  
 ans <- sum(dexp(x, theta, log = TRUE))  
 return(-ans)   
}  
n = 1000  
# Generating 100 random samples from exponential distribution  
x=rexp(n, 5)  
# Estimating rate using method of moments  
rate=1/mean(x)  
theta=c(rate)  
ans = optim(par=theta, fn=loglike.exponential, x=x, method="BFGS")  
# Parameters estimated using MLE  
ans$par

## [1] 5.108832

# Parameters estimated using method of moments  
theta

## [1] 5.108832

### Gamma Distribution

Gamma Distribution (alpha=2, beta=5)

loglike.gamma = function(theta, x) # Generating the negative log likelihood function for gamma function. dgamma denotes first derivative of gamma   
{  
 ans = sum(dgamma(x, theta[1], theta[2], log=TRUE))  
 return(-ans)   
}  
n = 1000  
x=rgamma(n,2,5) # Generating the Gamma distribution  
# Estimating the parameters alpha and beta using method of moments and storing the result in theta  
mu=mean(x)  
variance=var(x)  
beta=mu/variance  
alpha=mu\*beta  
theta = c(alpha,beta)  
ans = optim(par=theta, fn=loglike.gamma, x=x, method="BFGS")  
# Parameters estimated using MLE  
ans$par

## [1] 1.923178 4.876447

# Parameters estimated using Method of Moments  
theta

## [1] 1.904187 4.828294

### Chi-square Distribution

Chi-square Distribution (degrees of freedom = 30)

loglike.chisquare = function(x, theta) # Generating the negative log - likelihood for the Chi-square distribution  
{  
 ans <- sum(dchisq(x, theta[1], log = TRUE))  
 return(-ans)   
}  
n = 1000  
x=rchisq(n,30) # Generating 100 random samples from a Chi-square distribution with 30 degrees of freedom  
  
# Calculating df using method of moments  
df=mean(x)  
theta=c(df)  
ans = optim(par=theta, fn=loglike.chisquare, x=x, method="BFGS")  
# Estimators using MLE  
ans$par

## [1] 30.05462

# Estimators using Method of Moments  
theta

## [1] 30.0818

### Beta Distribution

Beta Distribution (alpha=2, beta=5)

loglike.beta = function(theta, x) # This function generates the first derivative of the negative log likelihood function. dbeta function is the first derivative of beta distribution  
{  
 ans = sum(dbeta(x, theta[1], theta[2], log=TRUE))  
 return(-ans)   
}  
n = 1000  
x=rbeta(n,2,5)  
mu=mean(x)  
variance=var(x)  
# Calculating the parameters of beta distribution using method of moments and storing the result as a list in theta  
alpha = (((mu^2)-(mu^3)-(variance\*mu))/variance)  
beta = (alpha\*(1-mu))/mu  
theta=c(alpha,beta)  
ans = optim(par=theta, fn=loglike.beta, x=x, method="BFGS") # Minimizing the negative log likelihood function  
ans$par # The parameters alpha and beta estimated using MLE

## [1] 2.028878 5.007678

theta # The parameters alpha and beta estimated using Method of Moments

## [1] 1.986331 4.920316

## KS Test

In statistics, the Kolmogorov–Smirnov test (K–S test or KS test) is a nonparametric test of the equality of continuous (or discontinuous), one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample K–S test), or to compare two samples (two-sample K–S test).The Kolmogorov–Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples. The null distribution of this statistic is calculated under the null hypothesis that the sample is drawn from the reference distribution (in the one-sample case) or that the samples are drawn from the same distribution (in the two-sample case).

Let us take the beta distribution and perform ks test based on goodness of fit.

p0 <- ks.test(x, "pbeta", 2, 5, exact = FALSE)$statistic # Generating the statistics for KS test of beta distribution   
p1 <- NULL # Initializing NULL value for the same  
loglike.beta = function(theta, x) # This function generates the first derivative of the negative log likelihood function. dbeta function is the first derivative of beta distribution  
{  
 ans = sum(dbeta(x, theta[1], theta[2], log=TRUE))  
 return(-ans)   
}  
n = 1000  
for (i in 1:100) {  
 x=rbeta(n,2,5)  
 mu=mean(x)  
 variance=var(x)  
 # Calculating the parameters of beta distribution using method of moments and storing the result as a list in theta  
 alpha = (((mu^2)-(mu^3)-(variance\*mu))/variance)  
 beta = (alpha\*(1-mu))/mu  
 theta=c(alpha,beta)  
 ans = optim(par=theta, fn=loglike.beta, x=x, method="BFGS") # Optimizing the negative log likelihood function  
 x\_star <- rbeta (n, ans$par[1], ans$par[2]) # Generating the beta distribution using parameters estimated using MLE method  
 p\_star <- ks.test(x, "pbeta", ans$par[1], ans$par[2], exact = FALSE)$statistic  
 p1 <- c(p1, p\_star) #Updating p1 with the staticstics from ks test for beta distribution generated using MLE  
}  
pvalue <- sum(p1 > p0)/100  
pvalue

## [1] 0.69

## References

1. <https://towardsdatascience.com/a-gentle-introduction-to-maximum-likelihood-estimation-9fbff27ea12f>
2. <https://faculty.washington.edu/ezivot/econ424/maximumLikelihood.r>
3. <https://www.r-bloggers.com/fitting-a-model-by-maximum-likelihood/>
4. <https://jblevins.org/log/r-mle>
5. <https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test>