

Assignment 4

Maximum Likelihood Estimators for various probability distributions

Maximum Likelihood Estimators

Maximum Likelihood Estimation (MLE) is a method of estimating the parameters of a statistical model, given observations.

The method of maximum likelihood is used with a wide range of statistical analyses. As an example, suppose that we are interested in the heights of adult female penguins, but are unable to measure the height of every penguin in a population (due to cost or time constraints). Assuming that the heights are normally distributed with some unknown mean and variance, the mean and variance can be estimated with MLE while only knowing the heights of some sample of the overall population. MLE would accomplish that by taking the mean and variance as parameters and finding particular parametric values that make the observed results the most probable given the normal model.

```
In [1]: 1 #importing libraries
        2 library(Rlab)
        3 library(pracma)
        4
        5 #supressing warning in Jupyter Notebook
        6 options(warn=-1)
```

Rlab 2.15.1 attached.

Attaching package: 'Rlab'

The following objects are masked from 'package:stats':

```
dexp, dgamma, dweibull, pexp, pgamma, pweibull, qexp, qgamma,
qweibull, rexp, rgamma, rweibull
```

The following object is masked from 'package:datasets':

```
precip
```

```
In [2]: 1 # Bernoulli Distribution
2 mle_bernoulli <- function(data){
3   p <- mean(data)
4   return(p)
5 }
6
7 # Binomial Distribution
8 mle_binomial <- function(data){
9   n <- length(data)
10  p <- (1/length(data))*(sum(data)/n)
11  return(p)
12 }
13
14 # Geometric Distribution
15 mle_geometric <- function(data){
16   p <- 1.0/(mean(data))
17   return(p)
18 }
19
20 # Poisson Distribution
21 mle_poisson <- function(data){
22   estimated_lambda <- mean(data)
23   return(estimated_lambda)
24 }
25
26 # Uniform Distribution
27 mle_uniform <- function(data){
28   a <- min(data)
29   b <- max(data)
30   return(c(a, b))
31 }
32
33 # Normal Distribution
34 mle_normal <- function(data){
35   # Estimating the parameters
36   mu <- mean(data)
37   var <- sum((data - mu)**2)/(length(data) - 1)
38   return(c(mu, var))
39 }
40
41 # Exponential Distribution
42 mle_exponential <- function(data){
43   theta <- mean(data)
44   return(theta)
45 }
46
47 # Gamma Distribution
48 mle_gamma <- function(data){
49   data <- data + 1e-6
50   s = log(mean(data)) - (sum(log(data)))/length(data)
51   alpha <- ((3 - s) + sqrt( ((s-3)**2) + (24*s) ))/(12*s)
52   beta <- mean(data)/alpha
53   return(c(alpha, beta))
54 }
55
56 # Beta Distribution
```

```

57 mle_beta <- function(data){
58   data_mean <- mean(data)
59   data_variance <- (sum(data * data))/length(data)
60   alpha <- ((data_mean ^ 2) - (data_mean * data_variance))/(data_variance - (
61   beta <- (alpha * (1 - data_mean))/(data_mean)
62
63   final_val <- c(alpha, beta)
64
65   # We will run the optimisation step for 100 iterations
66   for(index in 1:100){
67     g1 <- digamma(alpha) - digamma(alpha + beta) - (sum(log(data))/length(da
68     g2 <- digamma(beta) - digamma(alpha + beta) - (sum(log(1 - data))/length(
69     g <- c(g1, g2)
70
71     G1_val <- trigamma(alpha) - trigamma(alpha + beta)
72     G2_val <- -trigamma(alpha + beta)
73     G3_val <- trigamma(beta) - trigamma(alpha + beta)
74     G <- matrix(c(G1_val, G2_val, G2_val, G3_val), nrow = 2, ncol = 2, byrow
75     G_inverse <- inv(G)
76
77     # Final values
78     final_val <- final_val - t(G_inverse %*% g)
79     alpha <- final_val[1]
80     beta <- final_val[2]
81   }
82
83   return(c(c(alpha, beta)))
84 }
85
86 # Chi Square Distribution
87 mle_chisq <- function(data){
88   # Intitial values for v from MOM estimator
89   p_tilda <- mean(data)
90
91   # We will use some approximations using the second derivative
92   n <- length(data)
93   del_p_numerator <- (-n/gamma(p_tilda/2) * digamma(p_tilda/2)) - (((n * log(
94   del_p_denominator <- (-n * trigamma(p_tilda/2)/4)
95   del_p <- del_p_numerator/del_p_denominator
96
97   estimated_p <- (p_tilda + del_p)/2
98   return(estimated_p)
99 }

```

Goodness of Fit Test

The goodness of fit of a statistical model describes how well it fits a set of observations. Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question.

In [3]:

```

1  # Goodness of fit function
2  gfit <- function(distribution, nboot = 1000, data)
3  {
4    mle_name = get(paste("mle_", distribution, sep = ""))
5    theta_hat = mle_name(data)
6    n <- length(data)
7
8    if(distribution == "poisson"){
9      q_hat <- qpois(c(1:n)/(n+1), theta_hat)
10
11      D0 <- ks.test(data, q_hat)$statistic
12      Dvec <- NULL
13
14      for(i in 1:nboot){
15        x_star <- rpois(n, theta_hat)
16        theta_hat_star <- mle_name(x_star)
17
18        q_hat_star <- qpois(c(1:n)/(n+1), theta_hat_star)
19        D_star <- ks.test(x_star, q_hat_star)$statistic
20        Dvec <- c(Dvec, D_star)
21      }
22      p_value <- sum(Dvec > D0)/nboot
23      return(p_value)
24    }
25    else if(distribution == "normal"){
26      q_hat <- qnorm(c(1:n)/(n+1), mean = theta_hat[1], sd = theta_hat[2])
27
28      D0 <- ks.test(data, q_hat)$statistic
29      Dvec <- NULL
30
31      for(i in 1:nboot){
32        x_star <- rnorm(n, mean = theta_hat[1], sd = theta_hat[2])
33        theta_hat_star <- mle_name(x_star)
34
35        q_hat_star <- qnorm(c(1:n)/(n+1), mean = theta_hat_star[1], sd = theta_hat_star[2])
36        D_star <- ks.test(x_star, q_hat_star)$statistic
37        Dvec <- c(Dvec, D_star)
38      }
39      p_value <- sum(Dvec > D0)/nboot
40      return(p_value)
41    }
42    else if(distribution == "uniform"){
43      q_hat <- qunif(c(1:n)/(n+1), theta_hat[1], theta_hat[2])
44
45      D0 <- ks.test(data, q_hat)$statistic
46      Dvec <- NULL
47
48      for(i in 1:nboot){
49        x_star <- runif(n, theta_hat[1], theta_hat[2])
50        theta_hat_star <- mle_name(x_star)
51
52        q_hat_star <- qunif(c(1:n)/(n+1), theta_hat_star[1], theta_hat_star[2])
53        D_star <- ks.test(x_star, q_hat_star)$statistic
54        Dvec <- c(Dvec, D_star)
55      }
56      p_value <- sum(Dvec > D0)/nboot

```

```

57     return(p_value)
58 }
59 else if(distribution == "gamma"){
60     q_hat <- qgamma(c(1:n)/(n+1), shape = theta_hat[1], scale = theta_hat[2])
61
62     D0 <- ks.test(data, q_hat)$statistic
63     Dvec<-NULL
64
65     for(i in 1:nboot){
66         x_star <- rgamma(n, shape = theta_hat[1], scale = theta_hat[2])
67         theta_hat_star <- mle_name(x_star)
68
69         q_hat_star <- qgamma(c(1:n)/(n+1), shape = theta_hat_star[1], scale = theta_hat_star[2])
70         D_star <- ks.test(x_star, q_hat_star)$statistic
71         Dvec <- c(Dvec, D_star)
72     }
73     p_value <- sum(Dvec > D0)/nboot
74     return(p_value)
75 }
76 else if(distribution == "beta"){
77     q_hat <- qbeta(c(1:n)/(n+1), shape1 = theta_hat[1], shape2 = theta_hat[2])
78
79     D0 <- ks.test(data, q_hat)$statistic
80     Dvec<-NULL
81
82     for(i in 1:nboot){
83         x_star <- rbeta(n, shape1 = theta_hat[1], shape2 = theta_hat[2])
84         theta_hat_star <- mle_name(x_star)
85
86         q_hat_star <- qbeta(c(1:n)/(n+1), shape1 = theta_hat_star[1], shape2 = theta_hat_star[2])
87         D_star <- ks.test(x_star, q_hat_star)$statistic
88         Dvec <- c(Dvec, D_star)
89     }
90     p_value <- sum(Dvec > D0)/nboot
91     return(p_value)
92 }
93 else if(distribution == "exponential"){
94     q_hat <- qexp(c(1:n)/(n+1), theta_hat)
95
96     D0 <- ks.test(data, q_hat)$statistic
97     Dvec<-NULL
98
99     for(i in 1:nboot){
100         x_star <- rexp(n, theta_hat)
101         theta_hat_star <- mle_name(x_star)
102
103         q_hat_star <- qexp(c(1:n)/(n+1), theta_hat_star)
104         D_star <- ks.test(x_star, q_hat_star)$statistic
105         Dvec <- c(Dvec, D_star)
106     }
107     p_value <- sum(Dvec > D0)/nboot
108     return(p_value)
109 }
110 }

```

Wrapper functions for Goodness of Fit for Maximum Likelihood Estimator

```
In [4]: 1 mle_wrapper <- function(distribution, population = 0){
2       p = 0.5
3       lambda = 0.5
4       a = 0
5       b = 100
6       theta = 2
7       alpha = 4.7
8       beta = 2.9
9       dog = 5
10
11      if (distribution == "bernoulli"){
12        if (population == 0){
13          p = 0.5
14          data = rbinom(10000, 1, p)
15        }
16        print("Population parameter: ")
17        print(p)
18        sampled = sample(data, 1000)
19        param_estimate <- mle_bernoulli(sampled)
20        print("Parameter Estimates: ")
21        print(param_estimate)
22      }
23      else if (distribution == "binomial"){
24        if (population == 0){
25          n = 1000
26          p = 0.5
27          data = rbinom(10000, n, p)
28        }
29        print("Population parameters: ")
30        print(paste(p, ",", n))
31        sampled = sample(data, 1000)
32        param_estimate <- mle_binomial(sampled)
33        print("Parameter Estimates: ")
34        print(param_estimate)
35      }
36      else if (distribution == "geometric"){
37        if (population == 0){
38          p = 0.5
39          data = rgeom(10000, p)
40        }
41        print("Population parameters: ")
42        print(p)
43        sampled = sample(data, 1000)
44        param_estimate <- mle_geometric(sampled)
45        print("Parameter Estimates: ")
46        print(param_estimate)
47      }
48      else if (distribution == "poisson"){
49        if (population == 0){
50          lambda = 0.5
51          data = rpois(10000, lambda)
52        }
53        print("Population parameters: ")
54        print(lambda)
55        sampled = sample(data, 1000)
56        param_estimate <- mle_poisson(data)
```

```
57     print("Parameter Estimates: ")
58     print(param_estimate)
59
60     # Doing parametric bootstrap of MLE using ks test
61     p_value <- gfit(distribution, data = data)
62     print("The p-value is: ")
63     print(p_value)
64 }
65 else if (distribution == "uniform"){
66     if (population == 0){
67         a = 0
68         b = 100
69         data = runif(10000,a,b)
70     }
71     print("Population parameters: ")
72     print(paste(a,",",b))
73     sampled = sample(data, 1000)
74     estimator <- mle_uniform(sampled)
75     print("Parameter Estimates: ")
76     print(estimator)
77
78     # Doing parametric bootstrap of MLE using ks test
79     p_value <- gfit(distribution, data = data)
80     print("The p-value is: ")
81     print(p_value)
82 }
83 else if (distribution == "normal"){
84     if (population == 0){
85         data = rnorm(10000, 0, 1)
86     }
87     print("Population mean: ")
88     print(mean(data))
89     print("Population variance: ")
90     print(var(data))
91     sampled = sample(data, 1000)
92     param_estimate <- mle_normal(sampled)
93     print("Parameter Estimates: ")
94     print(param_estimate)
95
96     # Doing parametric bootstrap of MLE using ks test
97     p_value <- gfit(distribution, data = data)
98     print("The p-value is: ")
99     print(p_value)
100 }
101 else if (distribution == "exponential"){
102     if (population == 0){
103         theta = 2
104         data = rexp(10000, theta)
105     }
106     print("Population parameter: ")
107     print(theta)
108     sampled = sample(data, 1000)
109     param_estimate <- mle_exponential(data = sampled)
110     print("Parameter Estimates: ")
111     print(param_estimate)
112
113     # Doing parametric bootstrap of MLE using ks test
```



```

114     p_value <- gfit(distribution, data = data)
115     print("The p-value is: ")
116     print(p_value)
117   }
118   else if (distribution == "gamma"){
119     if (population == 0){
120       alpha = 5
121       beta = 20
122       data = rgamma(10000, shape = alpha, scale = beta)
123     }
124     print("Population parameters: ")
125     print(paste(alpha,"",beta))
126     sampled = sample(data, 1000)
127     param_estimate <- mle_gamma(sampled)
128     print("Parameter Estimates: ")
129     print(param_estimate)
130
131     # Doing parametric bootstrap of MLE using ks test
132     p_value <- gfit(distribution, data = data)
133     print("The p-value is: ")
134     print(p_value)
135   }
136   else if (distribution == "beta"){
137     if (population == 0){
138       alpha = 4.7
139       beta = 2.9
140       data = rbeta(10000, shape1 = alpha, shape2 = beta)
141     }
142     print("Population parameters: ")
143     print(paste(alpha,"",beta))
144     sampled = sample(data, 1000)
145     param_estimate <- mle_beta(sampled)
146     print("Parameter Estimates: ")
147     print(param_estimate)
148
149     # Doing parametric bootstrap of MLE using ks test
150     p_value <- gfit(distribution, data = data)
151     print("The p-value is: ")
152     print(p_value)
153   }
154   else if (distribution == "chi square"){
155     if (population == 0){
156       dog = 5
157       data = rchisq(10000, df = dog)
158     }
159     print("Population parameter: ")
160     print(dog)
161     sampled = sample(data, 1000)
162     param_estimate <- mle_chisq(sampled)
163     print("Parameter Estimates: ")
164     print(param_estimate)
165   }
166 }

```

Calling MLE for various distribution

```
In [5]: 1 #MLE for Bernoulli Distribution
        2 mle_wrapper('bernoulli')
```

```
[1] "Population parameter: "
[1] 0.5
[1] "Parameter Estimates: "
[1] 0.489
```

```
In [6]: 1 #MLE for Binomial Distribution
        2 mle_wrapper('binomial')
```

```
[1] "Population parameters: "
[1] "0.5 , 1000"
[1] "Parameter Estimates: "
[1] 0.500358
```

```
In [7]: 1 #MLE for Geometric Distribution
        2 mle_wrapper('geometric')
```

```
[1] "Population parameters: "
[1] 0.5
[1] "Parameter Estimates: "
[1] 1.091703
```

```
In [8]: 1 #MLE for Poisson Distribution
        2 mle_wrapper('poisson')
```

```
[1] "Population parameters: "
[1] 0.5
[1] "Parameter Estimates: "
[1] 0.4981
[1] "The p-value is: "
[1] 0.324
```

```
In [9]: 1 #MLE for Uniform Distribution
        2 mle_wrapper('uniform')
```

```
[1] "Population parameters: "
[1] "0 , 100"
[1] "Parameter Estimates: "
[1] 0.007363642 99.960501422
[1] "The p-value is: "
[1] 0.726
```

```
In [10]: 1 #MLE for Normal Distribution
         2 mle_wrapper('normal')
```

```
[1] "Population mean: "
[1] -0.004709556
[1] "Population variance: "
[1] 0.9681644
[1] "Parameter Estimates: "
[1] -0.02587081 1.00099350
[1] "The p-value is: "
[1] 0.988
```

```
In [11]: 1 #MLE for Exponential Distribution
        2 mle_wrapper('exponential')
```

```
[1] "Population parameter: "
[1] 2
[1] "Parameter Estimates: "
[1] 0.4950785
[1] "The p-value is: "
[1] 0.589
```

```
In [12]: 1 #MLE for Gamma Distribution
        2 mle_wrapper('gamma')
```

```
[1] "Population parameters: "
[1] "5 , 20"
[1] "Parameter Estimates: "
[1] 5.002731 20.369871
[1] "The p-value is: "
[1] 0.547
```

```
In [13]: 1 #MLE for Beta Distribution
        2 mle_wrapper('beta')
```

```
[1] "Population parameters: "
[1] "4.7 , 2.9"
[1] "Parameter Estimates: "
[1] 4.504253 2.771081
[1] "The p-value is: "
[1] 0.683
```

```
In [14]: 1 #MLE for Chi Squared Distribution
        2 mle_wrapper('chi square')
```

```
[1] "Population parameter: "
[1] 5
[1] "Parameter Estimates: "
[1] 9.284379
```

References:

- https://en.wikipedia.org/wiki/Maximum_likelihood_estimation (https://en.wikipedia.org/wiki/Maximum_likelihood_estimation)
- https://en.wikipedia.org/wiki/Goodness_of_fit (https://en.wikipedia.org/wiki/Goodness_of_fit)