
Applied Data Analysis

R-Laboratory 7

Newton Raphson – Model Selection – Gamma Regression

Useful packages and functions:

- AIC()
- BIC()
- step()
- glm()

Task 24

Let $Y \sim \Gamma(\alpha, \beta)$ be gamma distributed with shape parameter $\beta > 0$ and rate parameter $\alpha > 0$, i.e. Y has the pdf

$$f(y; \alpha, \beta) = \frac{\alpha^\beta}{\Gamma(\beta)} y^{\beta-1} \exp(-\alpha y), \quad y > 0,$$

where Γ denotes the gamma-function.

- (a) Implement an R-function with your own implementation of the Newton-Raphson algorithm (see Algorithm II.2.26 in the lecture) to compute the maximum likelihood estimator of α for an iid sample $Y_1, \dots, Y_n \stackrel{\text{iid}}{\sim} \Gamma(\alpha, \beta)$, $n \in \mathbb{N}$, where β is assumed to be known. As a termination criterion, check if $|\alpha^{(t)} - \alpha^{(t+1)}| < \varepsilon(1 + |\alpha^{(t)}|)$ with $\varepsilon = 10^{-8}$ holds, where $\alpha^{(t)}$ denotes the value assigned to α at iteration $t \in \mathbb{N}$. Implement a second termination criterion $\delta|\alpha^{(t)}| < |\alpha^{(t+1)}|$ for $\delta > 0$ to check if the algorithm diverges. The function should get the following input variables:

- The observed sample $\mathbf{y} = (y_1, \dots, y_n)$ for an arbitrary $n \in \mathbb{N}$,
- the starting value $\alpha^{(0)} > 0$,
- the termination criterion values $\varepsilon, \delta > 0$ with default values $\varepsilon = 10^{-8}$ and $\delta = 100$.

The function should return a list with the following output variables:

- a boolean indicator if the algorithm converged or diverged,
- the final guess for α .
- the number of iterations T required for convergence

Test the function on a simulated $\Gamma(2, 2)$ sample of size $n = 20$ generated by the `rgamma(20, 2, 2)` function.

- (b) For the sample sizes $n = 10, 100, 500, 1000$, create 1000 data sets of the $\Gamma(2, 2)$ distribution and estimate the rate parameter $\alpha = 2$ using the MLE and the Newton-Raphson algorithm of (a). Compare the mean of the MLEs with the true value of α and \mathcal{I}^{-1} with the sample variance of the MLEs for all n , where \mathcal{I} denotes the Fisher information at $\alpha = 2$.

Hint: If the Newton-Raphson algorithm diverges in some cases, then only take the values of $\hat{\alpha}$ for which the algorithm converges to compute the sample mean and variance of the $\hat{\alpha}$ s, reporting the percentage of times the algorithm diverged.

Task 25

- (a) Download the file *Windmill.dat* from RWTHmoodle and load it as a data frame into your workspace. Transform the attribute `bin1` to type factor.
- (b) Divide the data in training and testing data randomly. The training data should consist of about $\frac{2}{3}$ of the rows of *Windmill.dat*.
- (c) Fit on this data frame a liner model with formula

$$\text{Cspd} \sim \text{Spd1} * \text{Spd1Lag1} + \text{Spd2} * \text{Spd2Lag1} + \text{Spd3} * \text{Spd3Lag1} + \text{Spd4} * \text{Spd4Lag1} + \text{Spd1sin1} + \text{Spd1cos1} + \text{bin1} + \text{Dir1}.$$

and compute the AIC and BIC for this model.

- (d) Next compute the AIC and BIC for the linear model with formula

$$\text{Cspd} \sim \text{Spd1} + \text{Spd1Lag1} + \text{Spd2} + \text{Spd2Lag1} + \text{Spd3} + \text{Spd3Lag1} + \text{Spd4} + \text{Spd4Lag1} + \text{Spd1sin1} + \text{Spd1cos1} + \text{bin1} + \text{Dir1}.$$

Compute the AIC and BIC. Which of the models considered in (c) and (d) do you prefer?

- (e) (i) Search for the linear model with lowest AIC value out of all models nested in the model (c) using a backward, a forward strategy and both. What do you observe?

Hint: You may use the function `step` with the model of (c) considered the most complex model.

- (ii) Now search for the linear model with lowest BIC value analogously.

Hint: If n denotes the number of rows of the data set, you can search with respect to BIC by setting `k=log(n)` in the `step` function.

- (f) Compare the models selected in (e) (i) and (ii) using the testing data and compute the so called *predicted residual sum of squares* (PRESS)

$$\text{PRESS} = \sum_{i=1}^n (y_i - \hat{\mu}_i)^2,$$

where $\hat{\mu}_1, \dots, \hat{\mu}_n$ denote the predicted values of a model considering the test data set. Which of the models do you prefer based on this criterion?

Task 26

Let $X \sim \mathcal{U}(20, 80)$ be uniformly distributed on $[20, 80]$ and let $Y|X = x \sim \Gamma(1, \beta(x))$ be gamma-distributed conditional on X with constant rate parameter $\alpha = 1$ and shape parameter specified by $\log(\beta(x)) = -2 + 0.08x$.

Randomly generate $n = 25$ independent observations from this model. Fit the model in R using the `glm` function. Calculate $\text{corr}(y - \hat{\mu}, \hat{\mu})$. Do the same for $n = 100$, $n = 1000$ and $n = 10000$ and summarize how the computed correlations depend on n .

```

# Task 24
# (a)
n=10
beta=2
alpha_true=2
y=rgamma(n, beta, alpha_true)

newton_increment<-function(alpha,x,beta=2){
  n <- length(x)
  score <- n*beta/alpha - sum(x)
  hessian <- (-n*beta/alpha^2)
  return(score/hessian)
}

newton_raphson<-function(x,alpha_0,epsilon=1e-8,delta=100){
  rep=TRUE #should algorithm should be repeated in next step?
  alpha_cur=alpha_0 #current value for alpha
  T.iteration=0 #number of iterations T
  while(rep){ #while rep==TRUE
    alpha_next=alpha_cur-newton_increment(alpha_cur,x) #newton step
    if(abs(alpha_next-alpha_cur) < epsilon*(1+abs(alpha_cur))){ #stopping criterion
      rep=FALSE
      conv=TRUE
    }
    if(delta*abs(alpha_cur) < abs(alpha_next)){ #criterion to check if algorithm diverges
      rep=FALSE
      conv=FALSE
    }
    alpha_cur=alpha_next
    T.iteration=T.iteration+1
  }
  return(list(convergence=conv,alpha=alpha_cur,iterations=T.iteration))
}

print(newton_raphson(y,2))

#(b)
for(n in c(10,100,500,1000)){
  alpha=rep(0, 1000)
  conv_ind = rep(FALSE,1000)
  for(i in 1:1000){
    y=rgamma(n, beta, alpha_true)
    result_it_i = newton_raphson(y,2)
    conv_ind[i] = result_it_i$convergence
    alpha[i]= result_it_i$alpha
  }
  # select only alpha estimates for which Newton-Raphson converged
  alpha = alpha[conv_ind] #take the values for alpha which were estimated
  J=n*beta/alpha_true^2 #fisher information
  cat(mean(alpha),"t",var(alpha),"t", J^(-1),"n") #mean of alpha, sample variance, inverse Fisher
information
  alpha_norm=(alpha-mean(alpha))/sd(alpha) #standardize
  hist(alpha_norm, freq=FALSE)
  curve(dnorm col="red" add=TRUE) #see theorem 11.2.22 approximate normal distribution

```

#####TASK25#####

a) load data and factorize bin1

```
windmil.task25 = read.table("R-Lab-Datasets/Windmill.dat", header = TRUE, sep = " ")
```

```
windmil.task25$bin1 = as.factor(windmil.task25$bin1)
```

b) divide the training and testing data randomly

```
set.seed(2020)
```

```
len.windmill = nrow(windmil.task25)
```

```
training.id = which(rbinom(len.windmill, 1, 0.666) == 1)
```

```
training.windmill = windmil.task25[training.id, ]
```

```
test.windmill = windmil.task25[-training.id, ]
```

c) Fit the model

```
#Cspd ~ Spd1 * Spd1Lag1 + Spd2 * Spd2Lag1 + Spd3 * Spd3Lag1 + Spd4 * Spd4Lag1+
```

```
# Spd1sin1 + Spd1cos1 + bin1 + Dir1.
```

```
complex.formula1 = CSpd ~ Spd1 * Spd1Lag1 + Spd2 * Spd2Lag1 + Spd3 * Spd3Lag1 + Spd4 *  
  Spd4Lag1+Spd1sin1 + Spd1cos1 + bin1 + Dir1
```

```
fit1 = lm(CSpd ~ Spd1 * Spd1Lag1 + Spd2 * Spd2Lag1 + Spd3 * Spd3Lag1 + Spd4 *  
  Spd4Lag1+Spd1sin1 + Spd1cos1 + bin1 + Dir1,  
  data = training.windmill)
```

```
# > AIC(fit1)
```

```
# [1] 3307.401
```

```
# > BIC(fit1)
```

```
# [1] 3455.71
```

```
fit2 = lm(CSpd ~ Spd1 + Spd1Lag1 + Spd2 + Spd2Lag1 + Spd3 + Spd3Lag1 + Spd4 +  
  Spd4Lag1+Spd1sin1 + Spd1cos1 + bin1 + Dir1,  
  data = training.windmill)
```

```
# > AIC(fit2)
```

```
# [1] 3310.718
```

```
# > BIC(fit2)
```

```
# [1] 3440.488
```

find the best fits using AIC

```
best.fit = step(fit1, direction = "backward")
```

null model can be used to find the best models

```
lm.pseudo = lm(CSpd ~ 1, data = training.windmill)
```

```
best.fit.1.AIC1 = step(lm.pseudo,complex.formula1, direction = "forward")
```

```
best.fit.1.AIC2 = step(lm.pseudo,complex.formula1, direction = "both")
```

find the best fits using BIC

```
best.fit = step(fit1, direction = "backward", k = log(n))
```

null model can be used to find the best models

```
lm.pseudo = lm(CSpd ~ 1, data = training.windmill)
```

```
best.fit.1.BIC1 = step(lm.pseudo,complex.formula1, direction = "forward", k = log(n))
```

```
best.fit.1.BIC2 = step(lm.pseudo,complex.formula1, direction = "both", k = log(n))
```

#f) PRESS

```
predicted.fit1 = predict(best.fit.1.AIC1, newdata = test.windmill)
```

```
PRESS = sum((test.windmill$CSpd - predicted.fit1)^2)
```