# Quantitative Risk Management
# Second Case for Week 4

John Dodson

November 13, 2019

## Numerical Approach to Maximum Likelihood Estimation

Gradient descent (Newton) methods for minimizing a real-valued function are based on the observation that if a function (in a single variable here) $u \mapsto h(u)$ is sufficiently regular near its minimum $u^\star$, then $h'(u^\star) = 0$ and

$$h'(u) \approx h'(u^\star) + h''(u^\star)(u - u^\star)$$

for $u$ near $u^\star$, so

$$u^\star \approx u - \frac{h'(u)}{h''(u)}$$

If $h''(u_j) > 0$ for each $j$, then an iterative scheme

$$u_{j+1} = u_j - \gamma_j \frac{h'(u_j)}{h''(u_j)} \quad \text{for} \quad j = 1, 2, \ldots$$

for $0 < \gamma_j \leq 1$, will converge to $u^\star$, as long as $u_0$ is close enough to $u^\star$, and $\gamma_j$ not too large.

### Multivariate optimization

If $u$ is an element of a vector space, the scheme generalizes to

$$u_{j+1} = u_j - \gamma_j \left[ \left. \frac{\partial h^2}{\partial u' \partial u} \right|_{u_j} \right]^{-1} \left. \frac{\partial h}{\partial u'} \right|_{u_j} \tag{1}$$

where the gradient is a column vector and the Hessian is a positive definite matrix.

In a generic unconstrained optimization setting, Newton methods can be burdensome because they require implementations for all of the first and second partial derivatives of the objective function.

### Approximate Fisher information

The authors of the BHHH method in [2] noted that, in the case of numerical maximum likelihood estimation, this burden is reduced substantially because the Fisher information of a random variable $X$ can be expressed as *either* the expected value of the Hessian *or* the covariance of the gradient of the log-likelihood with respect to the parameters.

$$\frac{\partial^2}{\partial \theta' \, \partial \theta} \operatorname{E}\left[-\log f(X;\theta)\right] = \operatorname{cov}\left[\frac{\partial \log f(X;\theta)}{\partial \theta'}\right]$$

So, if our problem is to identify the entropy-minimizing parameters

$$\hat{\theta} = \arg\min_{\theta} H(X;\theta)$$

where the entropy

$$H(X;\theta) = \mathrm{E}\left[-\log f(X)\right] \approx \frac{1}{n}\sum_{i=1}^{n} -\log f(x_i;\theta)$$

for an i.i.d. sample $\{x_i\}_{i=1,2,\ldots,n}$, we effectively have the objective function

$$h(u) = \frac{1}{n}\sum_{i=1}^{n} -\log f(x_i;u) \tag{2}$$

We still need to be able to evaluate the first partials for each $u_j$ by hand; but in terms of these the Hessian can be approximated by

$$\left.\frac{\partial h^2}{\partial u' \partial u}\right|_{u_j} \approx \frac{1}{n}\sum_{i=1}^{n} \left.\frac{\partial\left(-\log f(x_i;u)\right)}{\partial u'}\right|_{u_j} \left.\frac{\partial\left(-\log f(x_i;u)\right)}{\partial u}\right|_{u_j} \tag{3}$$

which is guaranteed to be a positive definite matrix as long as all of the parameters are distinct.

## Line search

We need to ensure in each step that $\gamma_j$ is not too big. The method employed in BHHH seems to be based on the prior work in [1].

The goal with this is to make sure that the magnitude of the gradient of $h(\cdot)$ at each step is always decreasing. Choose a constant $0 < \delta < \frac{1}{2}$. Start an inner iteration at $k = 0$ with the tentative assumption that $\gamma_j^{(0)} = 1$:

$$u_{j+1}^{(k)} = u_j - \gamma_j^{(k)} \left[\left.\frac{\partial h^2}{\partial u' \partial u}\right|_{u_j}\right]^{-1} \left.\frac{\partial h}{\partial u'}\right|_{u_j}$$

If $u_{j+1}^{(k)}$ is valid and

$$h\left(u_{j+1}^{(k)}\right) - h\left(u_j\right) < \delta \left(u_{j+1}^{(k)} - u_j\right)' \left.\frac{\partial h}{\partial u'}\right|_{u_j} \tag{4}$$

proceed with $u_{j+1} = u_{j+1}^{(k)}$. If not, progressively try

$$\gamma_j^{(k+1)} = 2^{-(k+1)}$$

for $k = 1, 2, \ldots$ until condition (4) is met.

Note that the line search sub-routine presents an opportunity to validate that the new candidate for the parameters satisfies any required constraints, such as the positivity of magnitudes[1].

[1] Unconstrained optimization is generally ineffective if the optimal value lies on a domain boundary. For problems of this variety, convex programming techniques may be more appropriate.

### Worked example

Let's consider the problem of determining the maximum likelihood estimates of the parameters of a Generalized Pareto random variable $X$ from an i.i.d. sample. The probability density function is

$$f(x;\beta,\xi) = \begin{cases} \frac{1}{\beta}\left(1-\xi\frac{x}{\beta}\right)^{-1/\xi-1} & \text{for} \quad \begin{cases} \xi > 0 \text{ and } x \leq 0 \\ -1 < \xi < 0 \text{ and } \frac{\beta}{\xi} < x \leq 0 \end{cases} \\ \frac{1}{\beta}\exp\left(-\frac{x}{\beta}\right) & \text{for} \quad \xi = 0 \text{ and } x \leq 0 \end{cases}$$

for scale parameter $\beta > 0$ and left tail index parameter $\xi > -1$. Note that, in spite of the apparent break at $\xi = 0$, $u \mapsto f(x;u)$ is smooth in both parameters $u = (\beta,\xi)'$ throughout their domains for all $x$ in the support.

The negative log-likelihood is

$$-\log f(x;u) = \begin{cases} \log\beta + \left(1+\frac{1}{\xi}\right)\log\left(1-\xi\frac{x}{\beta}\right) & \xi \neq 0 \\ \log\beta - \frac{x}{\beta} & \xi = 0 \end{cases}$$

The components of the gradient are

$$\frac{\partial(-\log f(x;u))}{\partial\beta} = \begin{cases} \frac{1}{\beta}\left(1-\left(1+\frac{1}{\xi}\right)\left(1-\frac{1}{1-\xi\frac{x}{\beta}}\right)\right) & \xi \neq 0 \\ \frac{1}{\beta}\left(1+\frac{x}{\beta}\right) & \xi = 0 \end{cases}$$

$$\frac{\partial(-\log f(x;u))}{\partial\xi} = \begin{cases} \frac{1}{\xi}\left(1+\frac{1}{\xi}\right)\left(1-\frac{1}{1-\xi\frac{x}{\beta}}\right) - \frac{1}{\xi^2}\log\left(1-\xi\frac{x}{\beta}\right) & \xi \neq 0 \\ -\frac{x}{\beta}\left(1+\frac{x}{2\beta}\right) & \xi = 0 \end{cases}$$

for $x$ in the support.

Matching first and second moments gives us a reasonable seed value to start the search for the maximum likelihood estimates.

$$\xi_0 = \frac{1}{2}\left(1-\frac{\mathrm{E}[X]^2}{\mathrm{var}[X]}\right)$$
$$\beta_0 = -\mathrm{E}[X]\frac{1}{2}\left(1+\frac{\mathrm{E}[X]^2}{\mathrm{var}[X]}\right)$$

assuming $\xi < \frac{1}{2}$ so that the expected value and variance exist.

This is coded in the appendix. Samples of simulated data drawn from a Generalized Pareto with $\beta = 1$ and $\xi = 0$ are fit to a tolerance of $10^{-8}$, which seems to require about 4–8 total iterations for a sample size of one hundred. Larger samples converge faster.

## References


[1] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, January 1966.

[2] Ernst K. Berndt, Bronwyn H. Hall, Robert E. Hall, and Jerry A. Hausman. Estimation and inference in nonlinear structural models. *Annals of Economic and Social Measurement*, 3(4):653–665, October 1974.

## Julia[2] implementation (`fall4case.jl`)

```
module Fall4case

using Statistics
using LinearAlgebra

"validate inputs for GP"
function GP_valid(x,β,ξ)
	if β≤0. || ξ<-1. || maximum(x)>0.
		return false
	end
	if ξ<0. && minimum(x)≤β/ξ
		return false
	end
	return true
end

"Generalied Pareto negative log-likelihood"
function GP(x,β,ξ)
	if !GP_valid(x,β,ξ)
		return NaN
	end
	if abs(ξ)<eps()
		return log(β).-x/β
	end
	return log(β).+(1+1/ξ)log.(1 .-ξ*x/β)
end

"β partial of GP negative log-liklihood"
function GP_β(x,β,ξ)
	if !GP_valid(x,β,ξ)
		return NaN
	end
	if abs(ξ)<eps()
		return (1 .+x/β)/β
	end
	return (1 .-(1+1/ξ)*(1 .-1 ./(1 .-ξ*x/β)))/β
end

"ξ partial of GP negative log-likelihood"
function GP_ξ(x,β,ξ)
	if !GP_valid(x,β,ξ)
		return NaN
```

[2] https://julialang.org/

```
        end
        if abs(ξ)<eps()
                return -x/β.*(1 .+x/2β)
        end
        return (1+1/ξ)*(1 .-1 ./(1 .-ξ*x/β))/ξ.-log.(1 .-ξ*x/β)/ξ^2
end

# simulated data: variates from ξ=0, β=1
x = log.(rand(100))

"objective"
function h(u)
        (β,ξ) = u
        return mean(GP(x,β,ξ))
end

"gradient"
function h_grad(u)
        (β,ξ) = u
        return mean(
             [GP_β(x,β,ξ) GP_ξ(x,β,ξ)]
             ,dims=1)
end

"approximate hessian"
function h_hess(u)
        (β,ξ) = u
        return cov(
            [GP_β(x,β,ξ) GP_ξ(x,β,ξ)]
            )
end

"approximate lower bound of estimator variance"
function cr_approx(u)
        return inv(length(x)h_hess(u))
end

"Newton method minimizer"
function newtMin(h,h_grad,h_hess,u0
                     ;maxiter=100,tol=1.e-8,δ=1.e-4)
        u1 = u0
        h1 = h(u1)
        if isnan(h1)
                throw(DomainError(u0,"invalid initial value"))
        end
```

```
        while maxiter>0
                u0 = u1
                h0 = h1
                k = 0
                while maxiter>0 && (k==0 || isnan(h1)
                      || h1-h0>δ*dot(u1-u0,h_grad(u0)))
                        u1 = u0-2.0^k*h_grad(u0)/h_hess(u0)
                        h1 = h(u1)
                        k -= 1
                        maxiter -= 1
                end
                if abs(h1-h0)<tol
                        return u1
                end
        end
        return u0
end

# initial parameter values from moment matching
ξ0 = (1-mean(x)^2/var(x))/2
β0 = -mean(x)*(1-ξ0)

"maximum likelihood estimate for GP parameters"
mle = newtMin(h,h_grad,h_hess,[β0 ξ0])

"approximate Cramér-Rao lower bound on standard errors"
se = sqrt.(diag(cr_approx(mle))')

export x,mle,se

end # Fall4case
```