Quantitative Risk Management
Case for Week 3

John A. Dodson

September 27, 2021

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 γ_j not too large.

Multivariate optimization

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

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

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

Approximate Fisher information

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.

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

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

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 γ_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 $\gamma_j^{(0)} = 1$:

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

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 conditions for positivity and stationarity of the conditional variance.

Implementation for GARCH quasi-MLE

Let's consider specifically the conditional quasi-MLE for GARCH(1,1),

$$\sigma_i^2 = \alpha_0 + \alpha_1 \varepsilon_{i-1}^2 + \beta_1 \sigma_{i-1}^2$$

for a timeseries of invariants $(X_i)_i$ where $\varepsilon_i = X_i - \mathrm{E}[X_i|\mathcal{F}_{i-1}]$ and $\sigma_i^2 = \mathrm{var}[X_i|\mathcal{F}_{i-1}]$.

The (quasi[1], conditional) negative log-likelihood for a sample $(x_i)_{i=1,\ldots,n}$ is

$$h(u) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(\log\left(2\pi\sigma_i^2\right) + \frac{\varepsilon_i^2}{\sigma_i^2}\right)$$

for parameters $u = (\alpha_0, \alpha_1, \beta_1)$, residuals $\varepsilon_i = x_i - \mu_i$, and unconditional variance $\sigma_1^2 = \alpha_0/(1-\beta_1-\alpha_1)$. We can assume that $\mu_i \equiv 0$ for daily returns.

BHHH is a quasi[2]-Newton method requiring an explicit gradient along with the objective function, from which the Hessian can be approximated from the Fisher information.

The partials are all of the form

$$\frac{\partial h}{\partial u_j} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2\sigma_i^2}\left(1 - \frac{\varepsilon_i^2}{\sigma_i^2}\right)\frac{\partial \sigma_i^2}{\partial u_j}$$

and the partials of the conditional variance are themselves linear recursions.

$$\begin{aligned}
\frac{\partial \sigma_1^2}{\partial \alpha_0} &= \frac{1}{1-\beta_1-\alpha_1} \quad , \qquad \frac{\partial \sigma_i^2}{\partial \alpha_0} = 1 + \beta_1 \frac{\partial \sigma_{i-1}^2}{\partial \alpha_0} \quad i = 2, \ldots, n \\
\frac{\partial \sigma_1^2}{\partial \alpha_1} &= \frac{1}{(1-\beta_1-\alpha_1)^2} \quad , \qquad \frac{\partial \sigma_i^2}{\partial \alpha_1} = \varepsilon_{i-1}^2 + \beta_1 \frac{\partial \sigma_{i-1}^2}{\partial \alpha_1} \quad i = 2, \ldots, n \\
\frac{\partial \sigma_1^2}{\partial \beta_1} &= \frac{1}{(1-\beta_1-\alpha_1)^2} \quad , \qquad \frac{\partial \sigma_i^2}{\partial \beta_1} = \sigma_{i-1}^2 + \beta_1 \frac{\partial \sigma_{i-1}^2}{\partial \beta_1} \quad i = 2, \ldots, n
\end{aligned}$$

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.

[1] in the sense of maximum entropy residuals

[2] in the sense of approximate Hessian

Julia implementation

```
module Fall3case

using Statistics, LinearAlgebra

"GARCH(1,1) at (α₀, α₁, β₁)"
function garch(ε,θ)
    (α₀,α₁,β₁) = θ
    σ² = fill(NaN,length(ε))
    if α₀>0 && α₁≥0 && β₁≥0 && α₁+β₁<1
        σ²[1] = α₀/(1-β₁-α₁)
        for i = 2:length(σ²)
            σ²[i] = α₀+α₁*ε[i-1]^2+β₁*σ²[i-1]
        end
    end
    return σ²
end

"GARCH(1,1) partials wrt (α₀, α₁, β₁)"
function garch_grad(ε,θ)
    (α₀,α₁,β₁) = θ
    σ² = garch(ε,θ)
    grad = fill([NaN;NaN;NaN],length(σ²))
        if α₀>0 && α₁≥0 && β₁≥0 && α₁+β₁<1
        grad[1] = [
            1/(1-β₁-α₁);
            α₀/(1-β₁-α₁)^2;
            α₀/(1-β₁-α₁)^2 ]
        for i = 2:length(grad)
            grad[i] = [
                1+β₁*grad[i-1][1];
                ε[i-1]^2+β₁*grad[i-1][2];
                σ²[i-1]+β₁*grad[i-1][3] ]
        end
    end
    return grad
end

"negative quasi log-likelihood for GARCH"
function qmle_obj(ε,θ)
    σ² = garch(ε,θ)
    return (log.(2π*σ²)+ε.^2 ./σ²)/2
end

"negative quasi log-likelihood for GARCH (α₀,α₁,β₁) partials"
```

```
function qmle_grad(ε,θ)
    σ² = garch(ε,θ)
    return (1 .-ε.^2 ./σ²)./(2*σ²).*garch_grad(ε,θ)
end

"Newton's method minimizer"
function newtMin(h_obj::Function,h_grad::Function,h_hess::Function,u₀::Vector
                    ;maxiter=500,tol=1.e-14,δ=1.e-4)
    u₁ = u₀
    h₁ = h_obj(u₁)
    if isnan(h₁)
        throw(DomainError(u₀,"invalid initial value"))
    end
    N = maxiter
    while N>0
        u₀ = u₁
        h₀ = h₁
        k = 0
        while N>0 && (k==0 || isnan(h₁)
            || h₁-h₀>δ*dot(u₁-u₀,h_grad(u₀)))
            u₁ = u₀-2.0^k*h_hess(u₀)\h_grad(u₀)
            h₁ = h_obj(u₁)
            k -= 1
            N -= 1
        end
        if abs(h₁-h₀)<tol
            return u₁
        end
    end
    return u₀
end

"BHHH solver for maximum likelihood estimates"
function bhhh(x::Vector,obj::Function,grad::Function,θ₀::Vector)
    h_obj = θ->mean(obj(x,θ))
    h_grad = θ->mean(grad(x,θ))
    h_hess = θ->cov(grad(x,θ))
    return newtMin(h_obj,h_grad,h_hess,θ₀)
end

"GARCH(1,1) fit"
function garch_fit(ε)
   θ₀ = [var(ε)*(1.0-0.4-0.4),0.4,0.4]
   return bhhh(ε,qmle_obj,qmle_grad,θ₀)
end
```

```
export garch,garch_fit

end # module
```