Lecture 5: Numerical Optimization

Zhentao Shi

June 8, 2015

```
library(AER)
library(nloptr)
library(numDeriv)
library(optimx)
```

Introduction

Optimization is the key step to apply econometric extremum estimators. A general optimization problem is formulated as

$$\min_{\theta \in \Theta} f(\theta) \text{ s.t. } g(\theta) = 0, h(\theta) \leq 0,$$

where $f(\cdot)$ is a criterion function, $g(\theta) = 0$ is an equality constraint, and $h(\theta) \leq 0$ is an inequality constraint.

Up to now, most established numerical optimization algorithm can a local minimum if it exists. However, there is no guarantee to locate the global minimum when multiple local minima exist.

- unconstrained or constrained

Popular algorithms

- Newton-type algorithm
  - gradient
  - Hessian
- Quasi-Newton type algorithm
  - BFGS
- Nelder-Mead

R's optimization infrastructure has been improving. R Optimization Task View gives a brief survey of the available CRAN packages.

Recently, the package optimx effectively replaces R's default optimization commands. `optimx` delivers a unified interface for various widely-used optimization algorithms. Moreover, it facilitates comparison amongst optimization routines.

Example: pseudo Poisson maximum likelihood

If y_i is a continuous random variable, it obviously does not follow a possion distribution, whose support is non-negative integers. However, if the conditional mean model

$$E[y_i|x_i] = \exp(x_i\beta),$$

is stastified, we can still use the possion regression to obtain a consistent estimator of the parameter β even if y_i does not follow a conditional poisson distribution,

To implement optimization in `R`, the criterion must be written as a function of a sole argument– the optimization parameter. All data must be provided inside or outside of the function, but not via any additional argument.

```
# Poisson likelihood
poisson.loglik = function(b) {
    b = as.matrix(b)
    lambda = exp(X %*% b)
    ell = -sum(-lambda + y * log(lambda))
    return(ell)
}

## prepare the data
data("RecreationDemand")
y = RecreationDemand$trips
X = with(RecreationDemand, cbind(1, income))

## estimation
b.init = c(0, 1)  # initial value
b.hat = optimx(b.init, poisson.loglik, method = c("BFGS", "Nelder-Mead"), control = list
    abstol = 1e-07))
print(b.hat)
```

```
##                    p1          p2    value fevals gevals niter convcode
## BFGS        1.177411 -0.09994222 261.1141     99     21    NA        0
## Nelder-Mead 1.167261 -0.09703975 261.1317     53     NA    NA        0
##              kkt1 kkt2 xtimes
## BFGS         TRUE TRUE   0.01
## Nelder-Mead FALSE TRUE   0.00
```

Simply check `value` in the outcomes for the two algorithms.

In practice no algorithm suits all problems. It is always recommended to use **Monte Carlo simulation**, in which the true parameter is known, to check the accuracy of one's optimization routine before applying to an empirical problem, in which the true parameter is unknown.

Contour plot is a helpful tool to visualize the function surface in low dimension.

```
## contour plot
x.grid = seq(0, 2, 0.05)
x.length = length(x.grid)
y.grid = seq(-0.5, 0.2, 0.01)
y.length = length(y.grid)

z.contour = matrix(0, nrow = x.length, ncol = y.length)

for (i in 1:x.length) {
    for (j in 1:y.length) {
        z.contour[i, j] = poisson.loglik(c(x.grid[i], y.grid[j]))
    }
}

filled.contour(x.grid, y.grid, z.contour)
```

For problems that demand more accuracy, standalone solvers can be invoked via interfaces to R. For example, we can access NLopt through nloptr. However, standalone solvers usually have to be compiled and configured. These steps are often not as straightforward as installing most of Windows applications.

NLopt provides an extensive list of algorithms.

```
## optimization with NLoptr

opts = list(algorithm = "NLOPT_LN_NELDERMEAD", xtol_rel = 1e-07)
# check the solver status

res_NM = nloptr(x0 = b.init, eval_f = poisson.loglik, opts = opts)
print(res_NM)
```

```
##
## Call:
## nloptr(x0 = b.init, eval_f = poisson.loglik, opts = opts)
##
##
## Minimization using NLopt version 2.4.0
##
## NLopt solver status: 5 ( NLOPT_MAXEVAL_REACHED: Optimization stopped
```

```
## because maxeval (above) was reached. )
##
## Number of Iterations....: 100
## Termination conditions:  xtol_rel: 1e-07
## Number of inequality constraints:  0
## Number of equality constraints:    0
## Current value of objective function:  261.114078295402
## Current value of controls: 1.177398 -0.09993993
```

```
opts = list(algorithm = "NLOPT_LN_NELDERMEAD", xtol_rel = 1e-07, maxeval = 500)

res_NM = nloptr(x0 = b.init, eval_f = poisson.loglik, opts = opts)
print(res_NM)
```

```
##
## Call:
## nloptr(x0 = b.init, eval_f = poisson.loglik, opts = opts)
##
##
## Minimization using NLopt version 2.4.0
##
## NLopt solver status: 4 ( NLOPT_XTOL_REACHED: Optimization stopped because
## xtol_rel or xtol_abs (above) was reached. )
##
## Number of Iterations....: 118
## Termination conditions:  xtol_rel: 1e-07 maxeval: 500
## Number of inequality constraints:  0
## Number of equality constraints:    0
## Optimal value of objective function:  261.114078295329
## Optimal value of controls: 1.177397 -0.09993984
```

```
## 'SLSQP' is indeed the BFGS algorithm in NLopt, though 'BFGS' doesn't
## appear in the name
opts = list(algorithm = "NLOPT_LD_SLSQP", xtol_rel = 1e-07)

poisson.loglik.grad = function(b) {
    b = as.matrix(b)
    lambda = exp(X %*% b)
    ell = -colSums(-as.vector(lambda) * X + y * X)
    return(ell)
}

# check the numerical gradient and the analytical gradient
```

```r
b = c(0, 0.5)
grad(poisson.loglik, b)
```

```
## [1]  6542.46 45825.40
```

```r
poisson.loglik.grad(b)
```

```
##            income
##  6542.46 45825.40
```

```r
res_BFGS = nloptr(x0 = b.init, eval_f = poisson.loglik, eval_grad_f = poisson.loglik.gra
    opts = opts)
print(res_BFGS)
```

```
##
## Call:
##
## nloptr(x0 = b.init, eval_f = poisson.loglik, eval_grad_f = poisson.loglik.grad,
##     opts = opts)
##
##
## Minimization using NLopt version 2.4.0
##
## NLopt solver status: 4 ( NLOPT_XTOL_REACHED: Optimization stopped because
## xtol_rel or xtol_abs (above) was reached. )
##
## Number of Iterations....: 38
## Termination conditions:  xtol_rel: 1e-07
## Number of inequality constraints:  0
## Number of equality constraints:    0
## Optimal value of objective function:  261.114078295329
## Optimal value of controls: 1.177397 -0.09993984
```

Contrained optimization in R

- `optimx` can handle simple box-constrained problems.
- `constrOptim` can handle linear constrained problems.
- Some algorithms in `nloptr`, for example, `NLOPT_LD_SLSQP`, can handle nonlinear constrained problems.
- `Rdonlp2` is an alternative for general nonlinear constrained problems. Rdonlp2 is a package offered by `Rmetric` project. It can be installed by `install.packages("Rdonlp2", repos="http://R-Forge.R-project.org")`

Convex optimization

If a function is convex in its argument, then a local minimum is a global minimum. Convex optimization is particularly important in high-dimensional problems.

Example

- linear regression model MLE
- Lasso
- empirical likelihood

Rmosek is an interface in R to access Mosek, a high-quality commercial solver dedicated to convex optimization. Mosek provides free academic licenses.

Extended Readings

- Boyd and Vandenberghe (2004): Convex Optimization
- Buhlmann and van de Geer (2011): Statistics for High-Dimensional Data. Chapter 2
- Owen (2000): Empirical Likelihood
- Nash (2014): On Best Practice Optimization Methods in R