SQUAREM

An R package for Accelerating Slowly Convergent Fixed-Point Iterations Including the EM and MM algorithms

Ravi Varadhan[1]

[1] Division of Geriatric Medicine & Gerontology
Johns Hopkins University
Baltimore, MD, USA

UseR! 2010
NIST, Gaithersburg, MD
July 22, 2010

Speed Is Not All That It's Cranked Up To Be

Evil deeds do not prosper; the slow man catches up with the swift - Homer (Odyssey)

What is a Fixed-Point Iteration?

$$x_{k+1} = F(x_k), \quad k = 0, 1, \ldots .$$

$F : \Omega \subset \mathbb{R}^p \mapsto \Omega$, and differentiable

- Most (if not all) iterations are FPI
- We are interested in contractive FPI
- Guaranteed convergence: $\{x_k\} \to x^*$

EM Algorithm

Let y, z, x, be observed, missing, and complete data, respectively.
The k-th step of the iteration:

$$\theta_{k+1} = \text{argmax } Q(\theta|\theta_k); \quad k = 0, 1, \ldots,$$

where

$$\begin{aligned} Q(\theta|\theta_k) &= E[L_c(\theta)|y, \theta_k], \\ &= \int L_c(\theta) f(z|y, \theta_k) dz, \end{aligned}$$

Ascent property: $L_{obs}(\theta_{k+1}) \geq L_{obs}(\theta_k)$

MM Algorithm

A *majorizing function*, $g(\theta|\,\theta^k)$:

$$\begin{aligned} f(\theta_k) &= g(\theta_k|\,\theta_k), \\ f(\theta_k) &\leq g(\theta|\,\theta_k), \ \forall\,\theta. \end{aligned}$$

- *To minimize $f(\theta)$, construct a majorizing function and minimize it (MM)*

$$\theta_{k+1} = \mathit{argmax}\, g(\theta|\theta_k); \quad k = 0, 1, \ldots$$

- *Descent property:* $f(\theta_{k+1}) \leq f(\theta_k)$
- *Is EM a subclass of MM or are they equivalent? It avoids the E-step.*

Least Squares Multidimensional Scaling

$$\text{Minimize}: \quad \sigma(X) = \frac{1}{2}\sum^{n}\sum^{n} w_{ij}(\delta_{ij} - d_{ij}(X))^2$$

over all $m \times p$ matrices X, where: $d_{ij} = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2}$

- Jan de Leeuw's SMACOF algorithm: $\xi_{k+1} = F(\xi)$,
- Has descent property: $\sigma(\xi_{k+1}) < \sigma(\xi_k)$
- An instance of MM algorithm

BLP Contraction Mapping

Previous Talk!

Power Method

To find the eigenvector corresponding to the largest (in magnitude) eigenvalue of an $n \times n$ matrix, A.

- Not all that academic - Google's PageRank algorithm!
- $x_{k+1} = A.x_k / \|A.x_k\|$
- Stop if $\|x_{k+1} - x_k\| \leq \varepsilon$
- Dominant eigenvalue (Rayleigh quotient) = $\frac{\langle A x_*, x_* \rangle}{\langle x_*, x_* \rangle}$
- Geometric convergence with rate $\propto \frac{|\lambda_1|}{|\lambda_2|}$
- Power method does not converge if $|\lambda_1| = |\lambda_2|$, but SQUAREM does!

Why Accelerate Convergence?

- These FPI are globally convergent
- Convergence is linear: Rate = $[\rho(J(x^*))]^{-1}$
- Slow convergence when spectral radius, $\rho(J(x^*))$, is large
- Need to be accelerated for practical application
- Without compromising on global convergence
- Without additional information (e.g. gradient, Hessian, Jacobian)

SQUAREM

- An R package implementing a family of algorithms for speeding-up any slowly convergent multivariate sequence
- Easy to use
- Ideal for high-dimensional problems
- Input: *fixptfn* = fixed-point mapping F
- Optional Input: *objfn* = objective function (if any)
- Two main control parameter choices: order of extrapolation and monotonicity
- Available on *R-forge* under **optimizer** project.
  install.packages(”SQUAREM”, repos = ”http://R-Forge.R-project.org”)

Upshot

SQUAREM works great!

- Significant acceleration (depends on the linear rate of F)
- Globally convergent (especially, first-order locally non-monotonic schemes)
- Finds the same or (sometimes) better fixed-points than FPI (e.g. EM, SMACOF, Power method)

SMACOF Results

Mores code data (de Leeuw 2008). 36 Morse signals compared - 630 dissimilarities & 69 parameters

Table: A comparison of the different schemes.

Scheme	# Fevals	# ObjEvals	CPU (sec)	ObjfnValue
SMACOF	1549	1549	471	0.0593
SQ1	213	141	55	0.0593
SQ2	140	57	32	0.0593
SQ3	113	33	24	0.0457
SQ3*	113	0	19	0.0457

Power Method - Part I

Generated a 1000×1000 (arbitrary) matrix with eigenvalues as follows:

```
eigvals <- c(2, 1.99, runif(997, 0, 1.9), -1.8)
```

A cool algorithm using the Soules matrix!

Table: A comparison of the different schemes: Average of 100 simulations

Scheme	# Fevals	CPU (sec)	Converged
Power	1687	8.8	100
SQ1	165	0.88	100
SQ2	121	0.69	100
SQ3	115	0.65	100

Power Method - Part II

Generated a 100×100 (arbitrary) matrix with eigenvalues as follows:

```
eigvals <- c(2, 1.99, runif(97, 0, 1.9), -2)
```

Table: A comparison of the different schemes: Average of 100 simulations

Scheme	# Fevals	CPU (sec)	Converged
Power	50000	3.46	0
SQ1	178	0.023	100
SQ2	130	0.031	100
SQ3	122	0.027	100

For Further Reading I

R. Varadhan, and C. Roland
Scandinavian Journal of Statistics.
2008.

C. Roland, R.Varadhan, and C.E. Frangakis
Numerical Mathematics.
2007.