Integration

Zhentao Shi

April 2, 2020

Integration

若于旷野中，积土成佛庙，乃至童子戏，聚沙为佛塔。如是诸人等，皆已成佛道。

In their mathematical definitions, integration and differentiation involve taking limit. However, our computer is a finite-precision machine that can handle neither arbitrarily small nor arbitrarily large numbers; it can, at best, approximate the limiting behavior. In this lecture, we first briefly talk about numerical differentiation and integration, and then we discuss stochastic methods with examples from econometrics. In particular, we will introduce simulated method of moments, indirect inference, and Markov Chain Monte Carlo (MCMC). These methods are beyond the in-class coverage of Econ5121A and Econ5150. Interested readers are referred to Cameron and Trivedi (2005) (Chapters 12 and 13) for details.

Numerical Methods

Numerical differentiation and integration are fundamental topics and of great practical importance. However, how the computer works out these operations has nothing to do with economics or econometrics; it is the content of a numerical analysis course. Here we quickly go over the numerical methods. Judd (1998) (Chapter 7) is an authoritative reference.

In undergraduate calculus, we have learned the analytic differentiation of many common functions. However, there are cases in which analytic forms are unavailable or too cumbersome to program. For instance, to find the optimum for the objective function $f : R^K \mapsto R$ by Newton's method, in principle we need to code the K-dimensional gradient and the $K \times K$-dimensional Hessian matrix. Programming up the gradient and the Hessian manually is a time-consuming and error-prone job. What is worse, whenever we change the objective function, which happens often at the experimental stage of research, we have to redo the gradient and Hessian. Therefore, it is more efficient to use numerical differentiation instead of coding up the analytical expressions, in particular in the trial-and-error stage.

The partial derivative of a multivariate function $f : R^K \mapsto R$ at a point $x_0 \in R^K$ is

$$\left.\frac{\partial f(x)}{\partial x_k}\right|_{x=x_0} = \lim_{\epsilon \to 0} \frac{f(x_0 + \epsilon \cdot e_k) - f(x_0 - \epsilon \cdot e_k)}{2\epsilon},$$

where $e_k = (0, \ldots, 0, 1, 0, \ldots, 0)$ is the identifier of the k-th coordinate. The numerical execution in a computer follows the basic definition to evaluate $f(x_0 \pm \epsilon \cdot e_k))$ with a small ϵ. But how small is small? Usually we try a sequence of ϵ's until the numerical derivative is stable. There are also more sophisticated algorithms.

In R, the package `numDeriv` conducts numerical differentiation, in which

- `grad` for a scalar-valued function;

- `jacobian` for a real-vector-valued function;
- `hessian` for a scalar-valued function;
- `genD` for a real-vector-valued function.

Integration is, in general, more difficult than differentiation. In R, `integrate` carries out one-dimensional quadrature, and `adaptIntegrate` in the package `cubature` deals with multi-dimensional quadrature. The reader is referred to the documentation for the algorithm behind numerical integrations.

Numerical methods are not panacea. Not all functions are differentiable or integrable. Before turning to numerical methods, it is always imperative to try to understand the behavior of the function at the first place. Some symbolic software, such as `Mathematica` or `Wolfram Alpha`, is a useful tool for this purpose. R is weak in symbolic calculation despite the existence of a few packages for this purpose.

Stochastic Methods

An alternative to numerical integration is the stochastic methods. The underlying principle of stochastic integration is the law of large numbers. Let $\int h(x)dF(x)$ be an integral where $F(x)$ is a probability distribution. We can approximate the integral by $\int h(x)dF(x) \approx S^{-1}\sum_{s=1}^{S} h(x_s)$, where x_s is randomly generated from $F(x)$. When S is large, a law of large numbers gives

$$S^{-1}\sum_{s=1}^{S} h(x_s) \xrightarrow{\text{p}} E[h(x)] = \int h(x)dF(x).$$

If the integration is carried out not in the entire support of $F(x)$ but on a subset A, then

$$\int_A h(x)dF(x) \approx S^{-1}\sum_{s=1}^{S} h(x_s)\cdot 1\{x_s \in A\},$$

where $1\{\cdot\}$ is the indicator function.

In theory, we want to use an S as large as possible. In reality, we are constrained by the computer's memory and computing time. There is no clear guidance of the size of S in practice. Preliminary experiment can help decide an S that produces stable results.

Stochastic integration is popular in econometrics and statistics, thanks to its convenience in execution.

Example

Structural econometric estimation starts from economic principles. In an economic model, some elements unobservable to the econometrician dictate an economic agent's decision. Roy (1951) proposes such a structural model with latent variables, and the Roy model is the foundation of self-selection in labor economics.

In the original paper of the Roy model, an economic agent must be either a farmer or a fisher. The utility of being a farmer is $U_1^* = x'\beta_1 + e_1$ and that of being a fisher is $U_2^* = x'\beta_2 + e_2$, where U_1^* and U_2^* are latent (unobservable). The econometrician observes the binary outcome $y = \mathbf{1}\{U_1^* > U_2^*\}$. If (e_1, e_2) is independent of x, and

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & 1 \end{bmatrix}\right)$$

where σ_2 is normalized to be 1, we can write down the log-likelihood as

$$L(\theta) = \sum_{i=1}^{n} \{y_i \log P(U_{i1}^* > U_{i2}^*) + (1 - y_i) \log P(U_{i1}^* \leq U_{i2}^*)\}.$$

Let $\theta = (\beta_1, \beta_2, \sigma_1, \sigma_{12})$. Given a trial value θ, we can compute

$$p(\theta|x_i) = P\left(U_{i1}^*(\theta) > U_{i2}^*(\theta)\right) = P\left(x_i'(\beta_1 - \beta_2) > e_{i2} - e_{i1}\right).$$

Under the joint normal assumption, $e_{i2} - e_{i1} \sim N(0, \sigma_1^2 - 2\sigma_{12} + 1)$ so that

$$p(\theta|x_i) = \Phi\left(\frac{x_i'(\beta_1 - \beta_2)}{\sqrt{\sigma_1^2 - 2\sigma_{12} + 1}}\right)$$

where $\Phi(\cdot)$ is the CDF of the standard normal.

However, notice that the analytical form depends on the joint normal assumption and cannot be easily extended to other distributions. As long as the joint distribution of (e_{i1}, e_{i2}), no matter it is normal or not, can be generated from the computer, we can use the stochastic method. We estimate

$$\hat{p}(\theta|x_i) = \frac{1}{S} \sum_{i=1}^{S} \mathbf{1}\left(U_{i1}^{s*}(\theta) > U_{i2}^{s*}(\theta)\right),$$

where $s = 1, \ldots, S$ is the index of simulation and S is the total number of simulation replications.

Next, we match moments generated the theoretical model with their empirical counterparts. The choice of the moments to be matched is to be decided by the user. A set of valid choice for the Roy model example is

$$\begin{aligned}
g_1(\theta) &= n^{-1} \sum_{i=1}^{n} x_i (y_i - \hat{p}(\theta|x_i)) \approx 0 \\
g_2(\theta) &= n^{-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 - \bar{\hat{p}}(\theta|x_i)(1 - \bar{\hat{p}}(\theta|x_i)) \approx 0 \\
g_3(\theta) &= n^{-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \hat{p}(\theta|x_i))^2 \approx 0
\end{aligned}$$

where $\bar{y} = n^{-1} \sum_{i=1}^{n} y_i$ and $\bar{\hat{p}}(\theta) = n^{-1} \sum_{i=1}^{n} p(\theta|x_i)$. The first set of moments is justified by the independence of (e_{i1}, e_{i2}) and x_i so that $E[x_i y_i] = x_i E[y_i|x_i] = x_i p(\theta|x_i)$, and the second set matches the variance of y_i. Since the moment conditions $(g_j(\theta))_{j=1}^3$ equals the number of unknonwn parameters, these moment conditions just-identifies the parameter θ. We need to come up with more moment conditions for over-identification. Moreover, we need to choose a weighting matrix W to form a quadratic criterion for GMM in over-identification.

The above example can be viewed as an application of simulated maximum likelihood. In parallel, we can simulate a moment condition if its explicit form is unavailable. Pakes and Pollard (1989) provide the theoretical foundation of the simulated method of moments (SMM). Our co-teacher Prof. Guo (Guo, Xia, and Zhang 2018) recently applies SMM to estimate a structural labor model.

Indirect Inference

Indirect inference (Smith Jr 1993, @gourieroux1993indirect) is yet another simulated-based estimation method. Indirect inference is extensively used in structural model estimatoin (Li 2010). Theoretical analysis of indirect inference reveals its nice properties in bias deduction via a proper choice of the binding function (Phillips 2012).

The basic idea of indirect inference is to recover the structural parameter from an *auxiliary model*—usually an reduced-form regression. The reduced-form regression ignores the underlying economic structure and is a purely statistical procedure; thus the reduced-form regression is relatively easier to implement. A *binding function* is a one-to-one mapping from the parameter space of the reduced-form to that of the structural form. Once the reduced-form parameter is estimated, we can recover the structural parameter via the binding function. When the reduced-form parameter can be expressed in closed-form, we can utilize the analytical form to match the theoretical prediction and the empirical outcomes, as in Shi and Zheng (2018). In most cases however, the reduced-form implied by the structural model does not have a closed-form expression so simulation becomes necessary.

The choice of the auxiliary model is not unique. In the Roy model example where θ is the structural parameter, a sensible starting point to construct the auxiliary model is the linear regression between y_i and x_i. A set of reduced-form parameters can be chosen as $\hat{b} = (\hat{b}_1, \hat{b}_2, \hat{b}_3)'$, where

$$\begin{aligned}
\hat{b}_1 &= (X'X)^{-1}X'y \\
\hat{b}_2 &= n^{-1}\sum_{y_i=1}(y_i - x_i'b_1)^2 = n^{-1}\sum_{y_i=1}(1 - x_i'b_1)^2 \\
\hat{b}_3 &= n^{-1}\sum_{y_i=0}(y_i - x_i'b_1)^2 = n^{-1}\sum_{y_i=0}(x_i'b_1)^2.
\end{aligned}$$

Here $\hat{b}_1$ is associated with β, and $(\hat{b}_2, \hat{b}_3)$ are associated with (σ_1, σ_{12}).

Now we consider the structural parameter. Given a trial value θ, the model is parametric and we can simulate artificial error (e_{i1}^*, e_{i2}^*) conditional on x_i. In each simulation experiment, we can decide y_i^*, and we can further estimate the reduced-form parameter $\hat{b}^* = \hat{b}^*(\theta)$ given the artificial data. $b(\theta)$ is the binding function. Conducting such simulation for S times, we measure the distance between $\hat{b}$ and $\hat{b}^*$ as

$$Q(\theta) = \left(\hat{b} - S^{-1}\sum_{s=1}^{S}\hat{b}^*(\theta)^s\right)' W \left(\hat{b} - S^{-1}\sum_{s=1}^{S}\hat{b}^*(\theta)^s\right)$$

where s indexes the simulation and W is a positive definite weighting matrix. The indirect inference estimator is $\hat{\theta} = \arg\min_\theta Q(\theta)$. That is, we seek the value of θ that minimizes the distance between the reduced-form parameter from the real data and that from the simulated artificial data.

Markov Chain Monte Carlo

If the CDF $F(X)$ is known, it is easy to generate random variables that follow such a distribution. We can simply compute $X = F^{-1}(U)$, where U is a random draw from Uniform$(0, 1)$. This X follows the distribution $F(X)$.

If the pdf $f(X)$ is known, we can generate a sample with such a distribution by *importance sampling*. Metropolis-Hastings algorithm (MH algorithm) is such a method. MH is one of the

Markov Chain Monte Carlo methods. It can be implemented in the R package `mcmc`. This page contains demonstrative examples of MCMC.

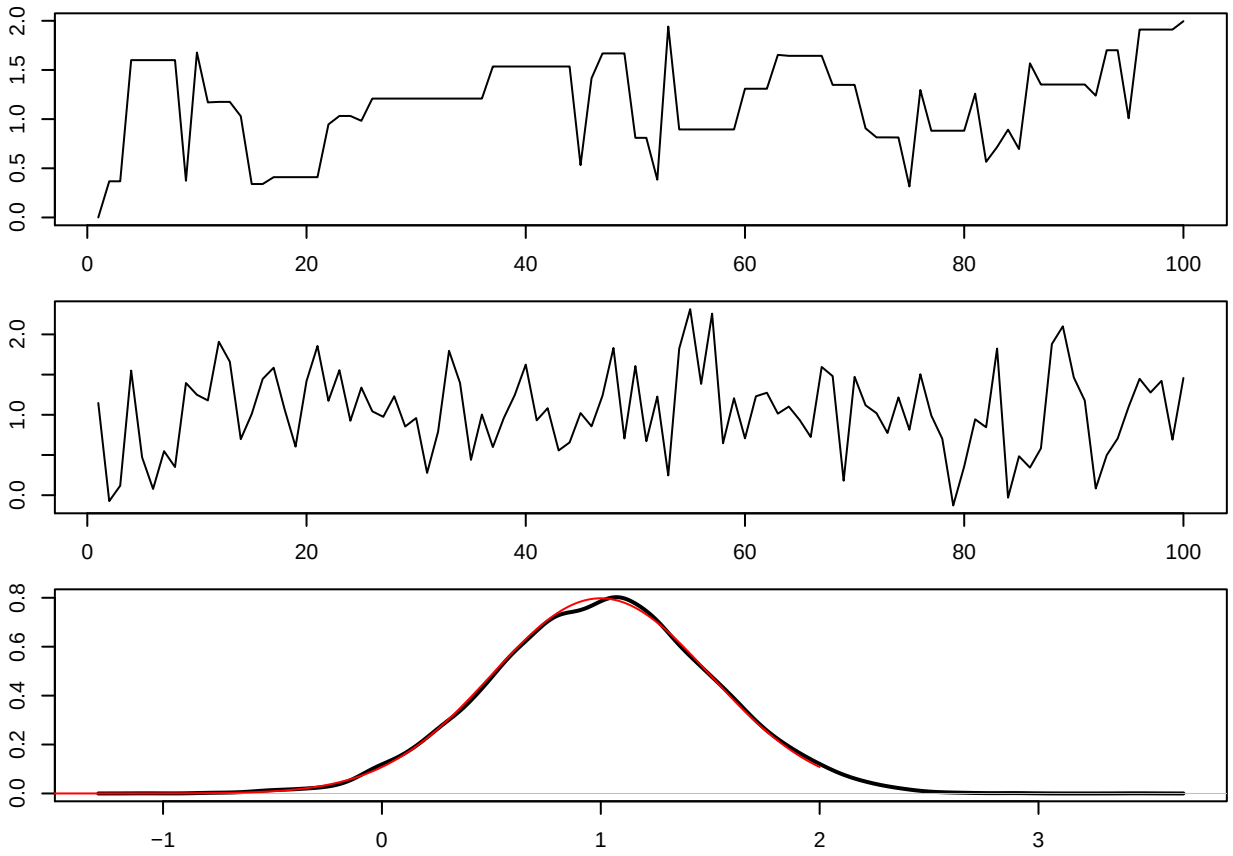
Metropolis-Hastings Algorithm

The underlying theory of the MH requies long derivation, but implementation is straightforward. Here we use MH to generate a sample of normally distributed observations with $\mu = 1$ and $\sigma = 0.5$. In the function `metrop`, we provide the logarithm of the density of

$$\log f(x) = -\frac{1}{2}\log(2\pi) - \log\sigma - \frac{1}{2\sigma^2}(x-\mu)^2$$

, and the first term can be omitted as it is irrelevant to the parameter.

```
library(mcmc)
h <- function(x, mu = 1, sd = 0.5) {
  y <- -log(sd) - (x - mu)^2 / (2 * sd^2)
} # un-normalized function (doesn't need to integrate as 1)

out <- metrop(obj = h, initial = 0, nbatch = 100, nspac = 1)
set.seed(123)

par(mfrow = c(3, 1))
par(mar = c(2, 2, 1, 1))
plot(out$batch, type = "l") # a time series with flat steps

out <- metrop(obj = h, initial = 0, nbatch = 100, nspac = 10)
plot(out$batch, type = "l") # a time series looks like a white noise

out <- metrop(obj = h, initial = 0, nbatch = 10000, nspac = 10)
plot(density(out$batch), main = "", lwd = 2)

xbase <- seq(-2, 2, 0.01)
ynorm <- dnorm(xbase, mean = 1, sd = 0.5)
lines(x = xbase, y = ynorm, type = "l", col = "red")
```




```
# the plot compare the density of the simulated sample
# with the standard normal pdf
```

The generated graph consists of three panels. The first panel is a time series where the marginal distribution of each observations follows $N(1, 0.5^2)$. The time dependence is visible, and flat regions are observed when the Markov chain rejects a new proposal so the value does not update over two periods. To reduced time dependenc, the middle panel collects the time series every 10 observations on the Markov chain. No flat region is observed in this subgraph and the serial correlation is weakened. The third panel compares the kernel density of the simulated observations (black curve) with the density function of $N(1, 0.5^2)$ (red curve).

Laplace-type Estimator: An Application of MCMC

For some econometric estimators, finding the global optimizer is known to be difficult, because of irregular behavior of the objective function. Chernozhukov and Hong (2003)'s *Laplace-type estimator* (LTE) is an alternative to circumvent the challenge in optimization. LTE transforms the value of the criterion function of an extremum estimator into a probability weight

$$f_n(\theta) = \frac{\exp(-L_n(\theta))\pi(\theta)}{\int_\Theta \exp(-L_n(\theta))\pi(\theta)}$$

where $L_n(\theta)$ is an criterion function (say, OLS criterion, (negative) log likelihood criterion, or GMM criterion), and $\pi(\theta)$ is the density of a prior distribution. The smaller is the value of the objective function, the larger it weighs.

We use MCMC to simulate the distribution of θ. From a Bayesian's viewpoint, $f_n(\theta)$ is the posterior distribution. However, Chernozhukov and Hong (2003) use this distribution for classical estimation and inference, and they justify the procedure via frequentist asymptotic theory. Once $f_n(\theta)$ is known, then *asymptotically* the point estimator equals its mean under the quadratic loss function, and equals its median under the absolute-value loss function.

The code block below compares the OLS estimator with the LTE estimator in a linear regression model.

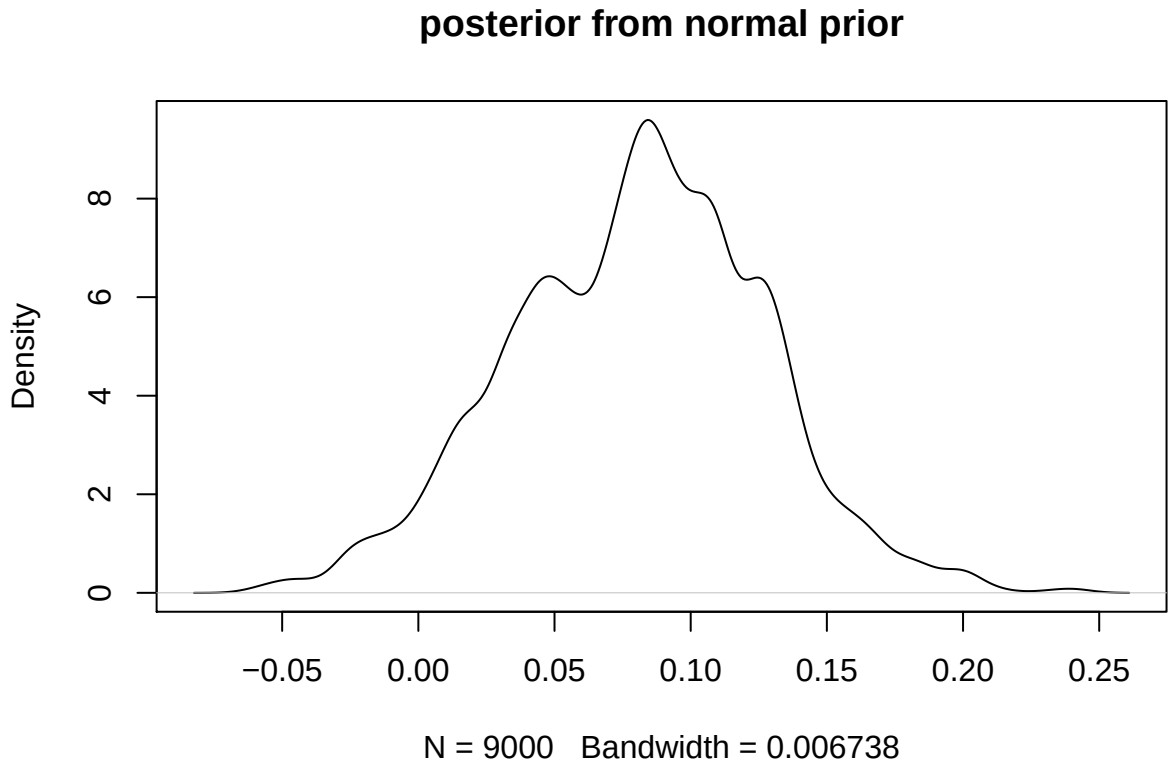
```
# set.seed(101)
library(mcmc)
set.seed(321)

# DGP
n <- 500
b0 <- c(.1, .1)
X <- cbind(1, rnorm(n))
Y <- X %*% b0 + rnorm(n)

b_OLS <- summary(lm(Y ~ -1 + X))
# "-1" because X has contained intercept
print(coef(b_OLS))
```

```
##      Estimate Std. Error  t value   Pr(>|t|)
## X1 0.10717458 0.04476623 2.394094 0.01703037
## X2 0.08460624 0.04539900 1.863615 0.06296414
```

```
# Laplace-type estimator
L <- function(b) -0.5 * sum((Y - X %*% b)^2) - 0.5 * crossprod(b - c(0, 0))
# notice the "minus" sign of the OLS objective function
# here we use a normal prior around (0,0).
# results are very similar if we replace it with a flat prior so that
# L <- function(b) -0.5*sum((Y - X %*% b)^2)

nbatch <- 10000
out <- metrop(obj = L, initial = c(0, 0), nbatch = nbatch, nspac = 20)

# summarize the estimation
bhat2 <- out$batch[-(1:round(nbatch / 10)), 2] # remove the burn in
bhat2_point <- mean(bhat2)
bhat2_sd <- sd(bhat2)
bhat2_CI <- quantile(bhat2, c(.025, .975))

# compare with OLS
print(cat(
  "The posterior mean =", bhat2_point, " sd = ", bhat2_sd,
  " and C.I. = ", bhat2_CI, "\n"
))
```

```
## The posterior mean = 0.08052159  sd =  0.04625461  and C.I. =  -0.01346313 0.1683149
## NULL
```

```
plot(density(bhat2), main = "posterior from normal prior")
```



References

Cameron, A Colin, and Pravin K Trivedi. 2005. *Microeconometrics: Methods and Applications*. Cambridge University Press.

Chernozhukov, Victor, and Han Hong. 2003. "An Mcmc Approach to Classical Estimation." *Journal of Econometrics* 115 (2): 293–346.

Gourieroux, Christian, Alain Monfort, and Eric Renault. 1993. "Indirect Inference." *Journal of Applied Econometrics* 8 (S1): S85–S118.

Guo, Naijia, Xiaoyu Xia, and Junsen Zhang. 2018. "A Matching Model of Intergenerational Co-Residence and Its Application in China." The Chinese University of Hong Kong.

Judd, Kenneth L. 1998. *Numerical Methods in Economics*. MIT press.

Li, Tong. 2010. "Indirect Inference in Structural Econometric Models." *Journal of Econometrics* 157 (1): 120–28.

Pakes, Ariel, and David Pollard. 1989. "Simulation and the Asymptotics of Optimization Estimators." *Econometrica*, 1027–57.

Phillips, Peter CB. 2012. "Folklore Theorems, Implicit Maps, and Indirect Inference." *Econometrica* 80 (1): 425–54.

Roy, Andrew Donald. 1951. "Some Thoughts on the Distribution of Earnings." *Oxford Economic Papers* 3 (2): 135–46.

Shi, Zhentao, and Huanhuan Zheng. 2018. "Structural Estimation of Behavioral Heterogeneity." *Journal of Applied Econometrics* 33 (5): 690–707.

Smith Jr, Anthony A. 1993. "Estimating Nonlinear Time-Series Models Using Simulated Vector Autoregressions." *Journal of Applied Econometrics* 8 (S1): S63–S84.