Integration and Monte-Carlo methods

Judd Chapters 7,8,9

David Childers (thanks to Y. Kryukov, K. Judd, and U. Doraszelski)

CMU, Tepper School of Business

April 3, 2023

Integration and Monte-Carlo

Univariate function integration:

- Newton-Cotes: piecewise
- Gaussian quadrature: global

Multivariate integration:

- Generalizations of univariate methods

Simulation (Monte-Carlo):

- Random number generation
- Variance reduction
- Optimization

Motivation

- Evaluate

$$I = \int_D f(x)dx,$$

  where $F : \mathbb{R}^d \to \mathbb{R}$ and $D \subset \mathbb{R}^d$.

  - Consumer / producer / total surplus
  - Expected utility, profit, etc.
  - Continuous time discounting
  - Bayesian posteriors, likelihoods

- Most integrals cannot be evaluated analytically
  - And evaluating $f(x)$ can be costly

Integration and function approximation

- Numerical integration – general approach:
  - Evaluate $f(x)$ for a few values of x (nodes)
  - Compute specially constructed weights for each node
  - Multiply evaluations by weights, and sum the results
- Different methods differ in choice of nodes and weights
- Choosing nodes and weights requires assumptions on the shape of the function
- We can integrate polynomials exactly;
  $\Rightarrow$ approximate our function with a polynomial
- Function approximation allows integration, but integration easier
  - Just need one feature, not a uniform representation

Univariate methods: overview

$f(x) : [a,b] \to \mathbb{R}$. (will deal with infinite intervals later)

$$I = \int_a^b f(x)dx,$$

Newton-Cotes: piecewise approximation to $f(x)$ (a spline)

1. Split $[a,b]$ into subintervals, e.g. of equal length h.
2. Over each interval, approximate f by a low-order polynomial.

Gaussian Quadrature: global polynomial approximation
Uses *orthogonal polynomial families*

1. Choose a family (and do the change of variable)
2. Compute nodes x_i, determine weights ω_i
3. Approximate integral as $\hat{I} = \sum \omega_i f(x_i)$

N-C: midpoint rule (step function)

1. Take n nodes, let $h = (b-a)/n$, and

$$x_i = a + (i - \frac{1}{2})h, \quad i = 1, \ldots, n,$$

2. Evaluate $f(x_i)$
3. Compute approximation:

$$\hat{I}_0 = \sum hf(x_i)$$

- Then, for some $\xi \in [a,b]$, for $f \in C^2$ by Taylor expansion

$$I = \int_a^b f(x)dx = \hat{I} + \frac{h^2(b-a)}{24} f''(\xi)$$

- $O(n^{-2})$ error.
- *Open rule*: no endpoints needed:
  - Useful if, e.g., discontinuous at boundary

Higher-order Newton-Cotes

- Trapezoid rule (piecewise-linear): $O(n^{-2})$ *closed rule* on C^2

$$\begin{aligned} f_i &= f(a+ih), \quad i = 0, \ldots, n, \\ \hat{I}_1 &= \frac{h}{2}\left(f_0 + 2f_1 + \ldots + 2f_{n-1} + f_n\right) \end{aligned}$$

$$I = \hat{I}_1 - \frac{h^2(b-a)}{12} f''(\xi).$$

- Simpson's rule (quadratic); n must be even: $O(n^{-4})$ on C^4

$$\begin{aligned} \int_{x_i}^{x_{i+2}} f(x)dx &= \frac{h}{3}\left[f_i + 4f_{i+1} + f_{i+2}\right] \\ \hat{I}_2 &= \frac{h}{3}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + \ldots + 4f_{n-1} + f_n\right) \end{aligned}$$

$$I = \hat{I}_2 - \tfrac{1}{180} h^4 (b-a) f^{(4)}(\xi).$$

Infinite intervals & change of variable

- Truncation risky unless tail known to decay fast: $\int_0^\infty f(x)dx \not\approx \int_0^b f(x)dx$
- Change of var.: $y = \phi(z)$, $\phi \in C^1$, $\phi' > 0$:

$$\int_a^b f(y)dy = \int_{\phi^{-1}(a)}^{\phi^{-1}(b)} f(\phi(z))\phi'(z)dz$$

- $(0,1)$ to $(0,+\infty)$: $\phi(z) = z/(1-z)$, $\phi'(z) = (1-z)^{-2}$

$$\int_0^{+\infty} f(x)dx = \int_0^1 f\left(\frac{z}{1-z}\right)(1-z)^{-2}dz$$

- $(0,1)$ to $(-\infty,+\infty)$: $\phi(z) = \ln[z/(1-z)]$, $\phi'(z) = 1/[z(1-z)]$
- Make sure derivative of integrand remains bounded

Orthogonal polynomials

- Functions form a linear space (infinite dimensional)
- Define inner product using a weighting function $w(x) > 0$:

$$\langle f, g \rangle = \int f(x) g(x) w(x) dx.$$

- Polynomial of degree k: $\phi_k(x) = \sum_{i=0}^{k} \alpha_i x^i$
- The family of polynomials $\{\phi_k\}$ is **mutually orthogonal** iff $\langle \phi_k, \phi_m \rangle = 0$ for all $k \neq m$.
  - Mononomials $(1, x, x^2, x^3, ...)$ are not orthogonal
  - Orthogonal families enable efficient computation
- Each family is defined by its weighting function

Gaussian quadrature

- Given a weighting function $w(x)$:

$$I = \int_a^b f(x)w(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i) = \hat{I},$$

  where $\{x_i\}_{i=1}^n$ are quadrature nodes, and $\{\omega_i\}_{i=1}^n$ are quadrature weights.
- Nodes and weights are chosen so the approximation is *exact* for all polynomials of degree $2n-1$ (Thm 7.2.1)
- If these polynomials are orthogonal w.r.t. $w(x)$:
  - The nodes are zeros of the degree n polynomial
  - Weights do not depend on f
  - $=>$ Take from tables or compute by $O(n^2)$ algorithms

Chebyshev & Legendre quadratures

- Let $f : [-1, 1] \to \mathbb{R}$, $\in C^\infty$, $f^{(n)}$ bounded.
- Gauss-**Chebyshev** quadrature:

$$\int_{-1}^{1} f(x)(1-x^2)^{-\frac{1}{2}} dx = \sum_{i=1}^{n} \omega_i f(x_i) + o(4^{-n}),$$

where

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \quad \omega_i = \frac{\pi}{n}, \quad i = 1, \ldots, n.$$

- Gauss-**Legendre** quadrature:

$$\int_{-1}^{1} f(x) dx = \sum_{i=1}^{n} \omega_i f(x_i) + o(4^{-n}),$$

see textbook for tables of x_i's and ω_i's

Gauss-Hermite quadrature

- Let $f : (-\infty, \infty) \to \mathbb{R}$, $\in C^\infty$, $f^{(n)}$ bounded. Then:

$$\int_{-\infty}^{\infty} f(x) e^{-x^2} dx = \sum_{i=1}^{n} \omega_i f(x_i) + o(K^{-n}),$$

  where x_i and ω_i, $i = 1, \ldots, n$, are tabulated.
- Typical use: expectation of a normal random variable
- Suppose $Y \sim N(\mu, \sigma^2)$. Then

$$\begin{aligned} E(f(Y)) &= \int_{-\infty}^{\infty} f(y) \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{y-\mu}{\sqrt{2}\sigma}\right)^2} dy \\ &= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} f(\sqrt{2}\sigma x + \mu) e^{-x^2} dx. \end{aligned}$$

Gauss-Laguerre quadrature

- Let $f : (0, \infty) \to \mathbb{R}$, $\in C^\infty$, $f^{(n)}$ bounded. Then:

$$\int_0^\infty f(x)e^{-x}dx = \sum_{i=1}^n \omega_i f(x_i) + o(K^{-n}),$$

  where x_i and ω_i, $i = 1, \ldots, n$, are tabulated.
- Typical use: Net present value of a stream of profits.
- Suppose $\rho > 0$ is the discount rate. Then

$$\int_0^\infty e^{-\rho t} f(t)dt = \frac{1}{\rho} \int_0^\infty e^{-x} f\left(\frac{x}{\rho}\right) dx.$$

Clenshaw-Curtis quadrature

- Alternative to Gauss-Legendre: $w(x) = 1$ on $[-1, 1]$
- Use Chebyshev points as nodes
  Use weights which exactly integrate Chebyshev interpolation
  - Computable in $O(n \log(n))$
- Exact only up to order $n - 1$ polynomials instead of $2n - 1$
- Handy when using Chebyshev also for function representation

Evaluation of quadrature

- For analytic functions, Gauss rules converge exponentially fast
  - Use tiny n in many econ applications
- For finitely differentiable f, Gauss and Clenshaw both polynomial
- With irregularities or unknown smoothness, use *adaptive* rules
  - Add points when error metric suggests inaccuracy
  - Save calculations by reusing nodes as grid refined
  - Curtis-Clenshaw points nested, can refine without re-evaluating
  - Gauss-Kronod rules allow refining Gaussian quadratures (`QuadGK` in Julia, `scipy.integrate.quad` in Python)
  - Newton-Cotes allows refinement locally for irregular functions

Multi-dimensional: product rules

- Product of nodes and weights across dimensions
- Approximate

$$\int_D F(x)dx$$

where $F : \mathbb{R}^d \to \mathbb{R}$ and $D \subset \mathbb{R}^d$, by

$$\sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \left(\prod_{j=1}^{d} \omega_{i_j} \right) F(x_{i_1}, \ldots, x_{i_d}),$$

- For dimension j, nodes $\left\{ x_{i_j} \right\}_{i_j=1}^{n_j}$ and weights $\left\{ \omega_{i_j} \right\}_{i_j=1}^{n_j}$ determined by a univariate Newton-Cotes or Gaussian scheme.
- Curse of dimensionality: $x \in \mathbb{R}^d \Rightarrow n^d$ nodes.

Multi-dimensional: Mononomials

- The complete set of polynomials of total degree k is

$$\Phi = \left\{ \prod_{j=1}^{d} \phi_{k_j}(x_j) \middle| \sum_{j=1}^{n} k_j \leq k \right\},$$

  where $\phi_{k_j}(x_j)$ is a polynomial of degree k_j in x_j.
- Impose exact integral of polynomials in Φ, i.e. find nodes $\{x^i\}_{i=1}^n$ and weights $\{\omega_i\}_{i=1}^n$ such that

$$\sum_{i=1}^{n} \omega_i \phi(x^i) = \int_D \phi(x) dx, \quad \forall \phi \in \Phi.$$

  $|\Phi|$ conditions, $nd + n$ unknowns
- Efficient solution methods exist in special cases
- Approximate as $\hat{I} = \sum_{i=1}^{n} \omega_i F(x^i)$
- Smolyak sparse grids make high dimensional ($d \approx 10 - 20$) integrals feasible if f reasonably smooth

Monte - Carlo methods

- Standard quadrature rules rely on properties of function
- An alternative independent of dimension or smoothness
- $\Rightarrow$ Generate nodes randomly: $X_i \sim U[D]$
- $\Rightarrow$ Assign equal weights:

$$I = \int_0^1 f(x)dx = E[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) = \hat{I}$$

- Law of Large Numbers and Central Limit theorem ensure *eventual* convergence, at rate $N^{-1/2}$
- Very slow relative to quadrature in low dimensions
- Close to optimal in high dimensions
  - Certain tricks to speed it up
- One requirement: random numbers

Quasi-random numbers: Weyl

- Ignore statistical properties; focus on suitability for numeric integration
- Sequence $x_j \in [0,1]$ is **equidistributed** if:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} f\left(x_j\right) = \int_0^1 f\left(x\right) dx$$

- E.g. *Weyl sequence*: for any irrational number θ:

$$y_j = \mathrm{mod}\left\{j\theta\right\}$$

  - $\Rightarrow$ sequence y_j is equidistributed
- For multidimensional draws, each dimension uses a different sequence (i.e. different θ)
- Better results than classical pseudo-random numbers in large samples

Quasi-random numbers: Halton

- Another criterion: discrepancy = difference from Uniform density
- Let $X = \{x_1, ..., x_N\} \subset [0,1]$. **Discrepancy** is:

$$D_N(X) = \sup_{0 \le a < b \le 1} \left| \frac{1}{N} count\{X \cap [a,b]\} - (b-a) \right|$$

  - *count* means number of elements
  - lowest possible value: $D_N(X) = 1/(N+1)$, achieved by $\{x_i = i/(N+1)\}$
  - we want a practical sequence that comes close
- E.g. *Halton sequence*, for a prime number p:
  - First p elements are numbers i/p, $i = 0, ..., p-1$
  - Then numbers i/p^2 $(i < p^2)$ that weren't picked yet
  - Then i/p^3, and so on: refining the grid
- Matlab: `haltonset()` to setup, `net()` to draw
- Julia: Option `LowDiscrepancySequence` in QuasiMonteCarlo.jl
- Python: `Halton` in scipy.stats.qmc

Variance reduction techniques

- **Antithetic variates**. Monotonic $f : [0,1] \to \mathbb{R}$
  $\Rightarrow cov\{f(x), f(1-x)\} < 0$, and

$$\hat{I}_f^a = \frac{1}{2N} \sum_{i=1}^{N} [f(x_i) + f(1-x_i)]$$

  has lower variance than plain Monte Carlo.

- **Control variates**: ϕ is similar to f and easy to integrate.

$$\int_0^1 f(x)dx = \int_0^1 \phi(x)dx + \int_0^1 (f(x) - \phi(x))dx$$

  $\Rightarrow$ integrate ϕ analytically, Monte Carlo on $(f - \phi)$.

Importance sampling

- Suppose $p(x)$ is a common pdf. Note

$$I_f = \int_0^1 f(x)dx = \int_0^1 \frac{f(x)}{p(x)} p(x)dx = E_{p(x)}\left[\frac{f(x)}{p(x)}\right]$$

- Draw $\{x_i\}_{i=1}^N$ from pdf $p(x)$, approximate I_f by

$$\hat{I}_f^p = \frac{1}{N}\sum_{i=1}^N \frac{f(x_i)}{p(x_i)}.$$

  $\hat{I}_f^p$ is an unbiased estimate of I_f with variance

$$V^p = \frac{1}{N}\left(\int_0^1 \frac{f(x)^2}{p(x)}dx - I_f^2\right).$$

- If $f(x) \geq 0$ and $p(x) = f(x)/I_f$, then $V^p = 0$.
  In practice, find a p that is similar to f.
- Thin-tails problem: $f(x)^2/p(x)$ can blow up V^p.

8.4 Stochastic approximation

- Solving expected utility problem

$$\max_{x \in X} U(x), \qquad U(x) = E_Z[u(x, Z)]$$

- Do not want to compute $E_Z[u(x, Z)]$ for every x
- Given a draw z_k, $u_x(x, z_k)$ is an unbiased estimate of $U_x(x)$
- Steepest descent: $x_{k+1} = x_k + \lambda_k U_x(x_k)$, $\lambda_k = \arg\max...$
- Stochastic gradient: $x_{k+1} = x_k + \lambda_k u_x(x_k, z_k)$,
  λ_k – predetermined (e.g. $\lambda_k = 1/k$)
- Many small steps; direction is random but correct on average.
- Robbins-Monro conditions for convergence:
  $\lambda_k \to 0, \qquad \sum_{k=1}^{\infty} \lambda_k = \infty, \qquad \sum_{k=1}^{\infty} \lambda_k^2 < \infty.$

8.5 Simulated objective

- Same expected utility problem

$$\max_{x \in X} U(x), \qquad U(x) = E_Z[u(x, Z)]$$

- Draw $\{z_i\}_{i=1}^N$, replace $U(x) = E(u(x, Z))$ by

$$\hat{U}(x) = \frac{1}{N} \sum_{i=1}^N u(x, z_i)$$

- Draw $\{z_i\}_{i=1}^N$ once, maximize $\hat{U}(x)$ instead of $U(x)$.
- Natural interpretation: replacing continuous distribution of Z with a discrete one.
- Computing $\hat{U}_x$ is easy if you have u_x.
- In econometrics, need $N >>$ sample size for accurate inference

MCMC: Markov Chain Monte Carlo

- Distribution $p(z)$ only known up to scale: $p(z) = \frac{f(z)}{\int f(z)dz}$
  - E.g. Bayesian Posterior: $p(z|y) = \frac{p(y|z)p(z)}{\int p(y|z)p(z)dz}$
- Direct sampling would require knowing $\int f(z)dz$ already
- Construct Markov chain $\tilde{p}(z^{t+1}|z^t)$ s.t. paths $\{z^t\}_{t=1}^T$ have distribution drawn from $p(z)$, so $\frac{1}{T}\sum_{t=1}^T g(z^t) \xrightarrow{p} \int g(z)p(z)dz$
- Metropolis-Hastings: starting at z^0, repeat T times:
  1. Draw $\tilde{z}'$ from *proposal distribution* $g(z'|z^t)$, e.g. $z' \sim N(z^t, \Sigma)$
  2. Calculate *Acceptance probability* $A = \min(1, \frac{f(z')}{f(z^t)}\frac{g(z^t|z')}{g(z'|z^t)})$
  3. With probability A, *accept*: $z^{t+1} = z'$
     Otherwise, *reject*: $z^{t+1} = z^t$
- Method only uses ratio, so never need normalizing constant
- Many variants: modify proposal distribution or chain itself

MCMC Variants and extensions

- Gibbs: Sample $p(z_1, \ldots, z_n)$ by alternately sampling $p(z_i|z_{-i})$
  - Exploits closed form conditionals for coordinates
- Hybrid/Hamiltonian Monte Carlo adds momentum term to draw
  - Uses gradients for faster sampling
- Sequential Monte Carlo/particle filter combines MCMC and IS
  - Latent variables: integrate out then again to get posterior
- Burn-in, thinning, tempering, and diagnostics can help
- Use *Probabilistic Programming Language* e.g. Turing or Stan to build and sample from distribution

References

- Julia: QuantEcon, Quadrature.jl,
- `SciPy`: integrate, stats.qmc
- MCMC and Bayesian computation
  - Chi Feng The Markov-chain Monte Carlo Interactive Gallery
  - Ed Herbst and Frank Schorfheide (2016) *Bayesian Estimation of DSGE Models* or Schorfheide handbook chapter for macro applications
  - Nicolas Chopin and James Ridgway (2017) "Leave Pima Indians Alone: Binary Regression as a Benchmark for Bayesian Computation" Statist. Sci. 32(1): 64-87 for sampler choices