Randomization to Solve Markov Decision Process Problems

Amy Rhee

Motivation

Recall the McCall worker problem:

$$V(w) = \max_{a,r}\{\frac{w}{1-\beta}, c + \beta \int_0^B v(w')dF(w')\}$$

How did we numerically compute the optimal policy?

Motivation

Recall the McCall worker problem:

$$V(w) = \max_{a,r}\{\frac{w}{1-\beta}, c + \beta \int_0^B v(w')dF(w')\}$$

How did we numerically compute the optimal policy?

- Method of Successive Approximations: Discretize the state space, say $S = \{1, ..., n\}$ and compute the value function at each possible state $s \in S$, i.e. $v^* = (v_1, ..., v_n)$, etc....

Motivation

Recall the McCall worker problem:

$$V(w) = \max_{a,r}\{\frac{w}{1-\beta}, c + \beta \int_0^B v(w')dF(w')\}$$

How did we numerically compute the optimal policy?

- Method of Successive Approximations: Discretize the state space, say $S = \{1, ..., n\}$ and compute the value function at each possible state $s \in S$, i.e. $v^* = (v_1, ..., v_n)$, etc....

The McCall worker problem is an example of a Markov decision process, namely a discrete Markov decision process

Motivation

- Structural estimation of discrete Markov decision processes typically require numerical methods since closed-form solutions for the decision rule are rare.
- **Curse of dimensionality** in approximating solutions to MDPs: Discretizing the state space $[0,1]^d$ to approximate the solution to an MDP problem requires the number of grid points to grow exponentially with dimension d
  - In other words, there is an exponential rise in time and space required to approximate a solution to an MDP problem as the number of state and controls increases[1]

[1] More technically, a DDP suffers from curse of dimensionality if $comp(d) = \Omega(2^d)$; i.e. lower bound on min computational cost grows exponentially fast as dimension d increases

Motivation: Monte Carlo Integration

Multivariate integration suffers from curse of dimensionality problem:

- Integrating an r times differentiable multivariate function on a d-dimensional unit cube $[0,1]^d$ using quadrature rules:

$$\sum_{i_1=1}^{N} \cdots \sum_{i_d=1}^{N} w_{i_1} \cdots w_{i_d} f(x_{i_1}, \ldots, x_{i_d})$$

  - Error is $\mathcal{O}(1/N^m)$, where $m = r/d$ and N is number of sample points at which we evaluate the integrand
- Using Monte Carlo integration: Error is $\mathcal{O}(1/\sqrt{N})$, i.e., if we want to make CI 10x smaller, then we need 100x more samples

MC Integration Algorithm

Integral of interest:

$$\iint \cdots \int f(x_1, ..., x_n) dx_1 \cdots dx_n \equiv \int_V f(\vec{x}) dV$$

1. Generate bounding box B $= [A_1, B_1] \times \cdots \times [A_m, B_m]$, where area of integration $V \subset B$
2. Generate points $\{x_i\}_{i=1}^N \sim unif(B)$
3. Define $g(x_i) = f(x_i)$ if $x_i \in V$ or 0 else
4. Return $\frac{1}{N} \sum_{i=1}^N g(x_i) \cdot \mathrm{Vol}(B)$

MC Integration Algorithm

Integral of interest:

$$\int\int \cdots \int f(x_1, ..., x_n) dx_1 \cdots dx_n \equiv \int_V f(\vec{x}) dV$$

1. Generate bounding box B $= [A_1, B_1] \times \cdots \times [A_m, B_m]$, where area of integration $V \subset B$
2. Generate points $\{x_i\}_{i=1}^N \sim unif(B)$
3. Define $g(x_i) = f(x_i)$ if $x_i \in V$ or 0 else
4. Return $\frac{1}{N} \sum_{i=1}^N g(x_i) \cdot \text{Vol}(B)$

Why this matters: Randomization can (sometimes) help us compute faster!

Rust (1997b): Using Randomization to Break the Curse of Dimensionality

If MC integration helps us beat the curse of dimensionality problem with multivariate integration, maybe this can work for DDP problems... but there are also some differences in the two problems

- Requires calculating an infinite number of multivariate integrals at each possible conditioning state $s \in S$
- Nonlinear since current value function V is the *max* of the conditional expectation of the future value function

Rust (1997b): Using Randomization to Break the Curse of Dimensionality

If MC integration helps us beat the curse of dimensionality problem with multivariate integration, maybe this can work for DDP problems... but there are also some differences in the two problems

- Requires calculating an infinite number of multivariate integrals at each possible conditioning state $s \in S$
- Nonlinear since current value function V is the *max* of the conditional expectation of the future value function

Rust (1997b): Shows that randomization breaks curse of dimensionality for DDPs

Set-up

Consider a discrete Markov decision process (DDP) consisting of:

- States s_t and controls d_t, where $t = 0, 1, ..., T$ and the set of possible controls A is finite

Set-up

Consider a discrete Markov decision process (DDP) consisting of:

- States s_t and controls d_t, where $t = 0, 1, ..., T$ and the set of possible controls A is finite
- A DM defined by primitives (u, p, β)
  - $u(s_t, d_t)$: utility function representing DM's preferences at time t
  - $p(s_{t+1}|s_t, d_t)$: a Markov transition probability representing the DM's subjective beliefs about uncertain future states
  - $\beta \in (0, 1)$: time-discounting factor for utility (in future periods)

Set-up

Every period, the DM acts optimally according to the optimal decision rule d_t given by:

$$d_t = \delta(s_t) =_\delta \mathbb{E}_\delta[\sum_t \beta^t u(s_t, d_t)|s_0 = s] \tag{1}$$

Set-up

Every period, the DM acts optimally according to the optimal decision rule d_t given by:

$$d_t = \delta(s_t) =_\delta \mathbb{E}_\delta[\sum_t \beta^t u(s_t, d_t)|s_0 = s] \tag{1}$$

Goal: To find this optimal decision rule δ to maximize/minimize this expectation wrt the controlled stochastic process $\{s_t, d_t\}$ induced by the decision rule $\delta = \{\delta_0, ..., \delta_T\}$

Set-up

First step: Construct value function

$$V(s) \equiv \max_{\delta} \mathbb{E}_{\delta}[\sum_{t} \beta^{t} u(s_t, d_t)|s_0 = s] \tag{2}$$

Second step: Use dynamic programming to "solve" Bellman equation

- Need to use numerical methods

Random Bellman Operator

We define the random bellman operator $\mathbf{\Gamma}_N : B \to B$, where $B = C([0,1]^d)$ as

$$\mathbf{\Gamma}_N[V](s) \equiv \max_{d \in A} u(s,d) + \beta \frac{1}{N} \sum_{k=1}^{N} V(\tilde{s}_k) p(\tilde{s}_k | s, d) \tag{3}$$

where $\{\tilde{s}_1, ..., \tilde{s}_N\}$ are IID draws wrt some measure λ from the unit hypercube $[0,1]^d$ at which the MC integral (sample average) is calculated

Random Bellman Operator

Our random bellman operator $\mathbf{\Gamma}_N[V]$ converges to the true bellman operator $\mathbf{\Gamma}[V]$, and the approximation error $\|\mathbf{\Gamma}_N[V] - \mathbf{\Gamma}[V]\|$ is bounded by approximation error between the maximum of two linear operators:

$$\mathbf{\Gamma}_{a,N}[V](s) = \max u(s,d) + \beta \frac{1}{N} \sum_{k=1}^{N} V(\tilde{s}_k) p(\tilde{s}_k | s, d)$$

$$\mathbf{\Gamma}_{a}[V](s) = \max u(s,d) + \beta \int V(s') p(s' | s, d) \lambda(ds')$$

Asymptotic Guarantees

We first impose the regularity conditions that u and p are Lipschitz continuous functions of s

- In particular, u and p are bounded by polynomial functions of d

Error Bounds for Random Bellman Operator

Expected approximation error of the random bellman operator to the true bellman operator is $\mathcal{O}_p(\frac{1}{\sqrt{N}})$, independent of the dimension d

- I.e., approximation error decreases at rate $1/\sqrt{N}$

Error Bounds for Random Bellman Operator

Expected approximation error of the random bellman operator to the true bellman operator is $\mathcal{O}_p(\frac{1}{\sqrt{N}})$, independent of the dimension d

- I.e., approximation error decreases at rate $1/\sqrt{N}$

What we really need to beat the curse of dimensionality: Show that expectation of the approximation errors do not increase exponentially fast in d, i.e. need a uniform bound on the expectation of the approximation error

Error Bounds for Random Bellman Operator

For each $N \geq 1$, the expected error in the random Bellman operator satisfies the uniform bound:

$$\sup_{p \in BL(K_p)} \sup_{V \in B(0,K_c)} \mathbb{E}[\|\boldsymbol{\Gamma}_N[V] - \boldsymbol{\Gamma}[V]\|] \leq \frac{\gamma(d)|A|K_p K_v}{\sqrt{N}} \tag{4}$$

where bounding constant $\gamma(d) = \sqrt{\frac{\pi}{2}}\beta(1 + d\sqrt{\pi}C)$

Using this result and a random version of the algorithm of successive approximations, Rust (1997b) shows that the random Bellman operator breaks the curse of dimensionality and thus changes DDP problems from exponential time to polynomial time complexity

- In Rust's results, the optimal values of a (polynomially) large sample of states are sufficient statistics for the near optimal values everywhere

Bray (2021 WP): Critique 20+ years later

Proves that results of Rust (1997b) are very limited because it requires that almost all but a vanishingly small fraction of state variables to behave as if they were IID r.v.s, i.e. memoryless

- Only $\approx \log(d)$ of d state variables may have memory of past states and controls for Rust's results to hold

Conclusion and Remarks

- Dynamic problems typically have curse of dimensionality, making them exponentially hard
- Random algorithms can be faster than deterministic ones in approximating solutions, but only for certain types of problems