Numerical Dynamic Programming

Jesús Fernández-Villaverde
University of Pennsylvania

Introduction

- In the last set of lecture notes, we reviewed some theoretical background on numerical programming.

- Now, we will discuss numerical implementation.

- Two issues:

  1. Finite versus infinite time.

  2. Discrete versus continuous state space.

Finite Time

- Problems where there is a terminal condition.

- Examples:

  1. Life cycle.

  2. Investment.

- Why are finite time problems nicer? Backward induction.

Infinite Time

- Problems where there is no terminal condition.

- Examples:

  1. Industry dynamics.

  2. Business cycle dynamics.

- However, we will need the equivalent of a terminal condition: transversality condition.

Discrete State Space

- We can solve problems up to floating point accuracy.

- Why is this important?

  1. ε-equilibria.

  2. Estimation.

- However, how realistic are models with a discrete state space.

Infinite State Space

- More common cases in economics.
- Problem: we will always have to rely on a numerical approximation.
- Interaction of different approximation errors.
- Bounds?

Different Strategies

1. Value Function Iteration.

2. Policy Function Iteration.

3. Projection.

4. Perturbation.

Value Function Iteration

- Well known, basic algorithm of dynamic programming.
- We have tight convergence properties and bounds on errors.
- Well suited for parallelization.
- It will always (perhaps quite slowly) work.

How Do We Implement The Operator?

- We come back to our two distinctions: finite versus infinite time and discrete versus continuous state space.

- Then we need to talk about:

  1. Initialization.

  2. Discretization.

Value Function Iteration in Finite Time

- We begin with the Bellman operator:

$$\Gamma\left(V^t\right)(s) = \max_{a \in A(s)} \left[u(s,a) + \beta \int V^{t'}\left(s'\right) p\left(ds'|s,a\right) \right]$$

- Specify V^T and apply Bellman operator:

$$V^{T-1}(s) = \max_{a \in A(s)} \left[u(s,a) + \beta \int V^{T}\left(s'\right) p\left(ds'|s,a\right) \right]$$

- Iterate until first period:

$$V^{1}(s) = \max_{a \in A(s)} \left[u(s,a) + \beta \int V^{2}\left(s'\right) p\left(ds'|s,a\right) \right]$$

Value Function Iteration in Infinite Time

- We begin with the Bellman operator:

$$\Gamma\left(V\right)\left(s\right)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V\left(s'\right)p\left(ds'|s,a\right)\right]$$

- Specify V^0 and apply Bellman operator:

$$V^1\left(s\right)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V^0\left(s'\right)p\left(ds'|s,a\right)\right]$$

- Iterate until convergence:

$$V^T\left(s\right)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V^{T-1}\left(s'\right)p\left(ds'|s,a\right)\right]$$

Normalization

- Before initializing the algorithm, it is usually a good idea to normalize problem:

$$V(s) = \max_{a \in A(s)} \left[(1-\beta)\, u(s,a) + \beta \int V\left(s'\right) p\left(ds' | s, a\right) \right]$$

- Two advantages:

  1. We save one iteration.

  2. Stability properties.

  3. Convergence bounds are interpretable.

Initial Value in Finite Time Problems

- Usually, economics of the problem provides natural choices.
- Example: final value of an optimal expenditure problem is zero.
- However, some times there are subtle issues.
- Example: what is the value of dying? And of bequests? OLG.

Initial Guesses for Infinite Time Problems

- Theorems tell us we will converge from any initial guess.
- That does not mean we should not be smart picking our initial guess.
- Several good ideas:
  1. Steady state of the problem (if one exists). Usually saves at least one iteration.
  2. Collapsing one or more dimensions of the problem. Which one?

Discretization

- In the case where we have a continuous state space, we need to discretize it into a grid.
- How do we do that?
- Dealing with curse of dimensionality.
- Do we let future states lie outside the grid?

New Approximated Problem

- Exact problem:

$$V(s) = \max_{a \in A(s)} \left[(1-\beta)\, u(s,a) + \beta \int V\left(s'\right) p\left(ds' | s, a\right) \right]$$

- Approximated problem:

$$\widehat{V}(s) = \max_{a \in \widehat{A}(s)} \left[(1-\beta)\, u(s,a) + \beta \sum_{k=1}^{N} \widehat{V}\left(s'_k\right) p_N\left(s'_k | s, a\right) \right]$$

Grid Generation

- Huge literature on numerical analysis on how to efficiently generate grids.

- Two main issues:

  1. How to select points s_k.

  2. How to approximate p by p_N.

- Answer to second issue follows from answer to first problem.

- We can (and we will) combine strategies to generate grids.

Uniform Grid

- Decide how many points in the grid.
- Distribute them uniformly in the state space.
- What is the state space is not bounded?
- Advantages and disadvantages.

Non-uniform Grid

- Use economic theory or error analysis to evaluate where to accumulate points.
- Standard argument: close to curvatures of the value function.
- Problem: this an heuristic argument.
- Self-confirming equilibria in computations.

Quadrature Grid

- Tauchen and Hussey (1991).

- Motivation: quadrature points in integrals

$$\int f(s)\, p(s)\, ds \simeq \sum_{k=1}^{N} f(s_k)\, w_k$$

- Gaussian quadrature: we require previous equation to be exact for all polynomials of degree less than or equal to $2N - 1$.

Stochastic Grid

- Randomly chosen grids.
- Rust (1995): it breaks the curse of dimensionality. Why?
- How do we generate random numbers?

Interpolation

- Discretization also generates the need for interpolation.
- Simpler approach: linear interpolation.
- Problem: in one than more dimension, linear interpolation may not preserve concavity.
- Shape-preserving splines: Schumaker scheme.

Multigrid Algorithms

- Old tradition in numerical analysis.

- Basic idea: solve first a problem in a coarser grid and use it as a guess for more refined solution.

- Examples:

  1. Differential equations.

  2. Projection methods.

  3. Dynamic programming (Chow and Tsitsiklis, 1991).

Applying the Algorithm

- After deciding initialization and discretization, we still need to implement each step:

$$V^T(s) = \max_{a \in A(s)} \left[u(s,a) + \beta \int V^{T-1}\left(s'\right) p\left(ds'|s,a\right) \right]$$

- Two numerical operations:

  1. Maximization.

  2. Integral.

Maximization

- We need to apply the max operator.
- Most costly step of value function iteration.
- Brute force (always works): check all the possible choices in the grid.
- Sensibility: using a Newton or quasi-Newton algorithm.

Brute Force

- Some times we do not have any other alternative. Examples: problems with discrete choices, constraints, non-differentiabilities, etc.

- Even if brute force is expensive, we can speed things up quite a bit:

  1. Previous solution.

  2. Monotonicity of choices.

  3. Concavity (or quasi-concavity) of value and policy functions.

Newton or Quasi-Newton

- Much quicker.

- However:

  1. Problem of global convergence.

  2. We need to compute derivatives.

- We can mix brute force and Newton-type algorithms.

Accelerator

- Maximization is the most expensive part of value function iteration.
- Often, while we update the value function, optimal choices are not.
- This suggests a simple strategy: apply the max operator only from time to time.
- How do we choose the optimal timing of the max operator?

How Do We Integrate?

- Exact integration.
- Approximations: Laplace's method.
- Quadrature.
- Monte Carlo simulations.

Convergence Assessment

- How do we assess convergence?

- By the contraction mapping property:

$$\left\|V - V^k\right\|_\infty \leq \frac{1}{1-\beta}\left\|V^{k+1} - V^k\right\|_\infty$$

- Relation of value function iteration error with Euler equation error.

Error Analysis

- We can use errors in Euler equation to refine grid.
- How?
- Advantages of procedure.
- Problems.