A Series of Lectures on Approximate Dynamic Programming Lecture 4

Dimitri P. Bertsekas

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

University of Cyprus
September 2017

APPROXIMATE DYNAMIC PROGRAMMING III

Outline

1. Approximation in Policy Space

2. Tail Problem Approximation

Using a Parametric Approximation Architecture for Policies

- Parametrize policies with a parameter vector $r = (r_0, \ldots, r_{N-1})$:

$$\pi(r) = \{\tilde{\mu}_0(x_0, r_0), \ldots, \tilde{\mu}_{N-1}(x_{N-1}, r_{N-1})\}$$

- Compute/train off-line the parameters using some optimization
- Great advantage: After off-line training, the on-line calculation of the controls is very fast

An important use: Implement policies obtained by approximation in value space

- Train off-line a cost function approximation and compute many state-control pairs (x_k^s, u_k^s), $s = 1, \ldots, q$
- Train a policy approximation architecture on these pairs. For example by solving for each k the least squares problem

$$\min_{r_k} \sum_{s=1}^{q} \left\| u_k^s - \tilde{\mu}_k(x_k^s, r_k) \right\|^2 + \text{(Regularization term)}$$

- This idea applies more generally. Generate many “good" state-control pairs (x_k^s, u_k^s), using a software or human “expert" and train in policy space

Cost Optimization Approach for Training a Policy Architecture

- Minimize the cost $J_{\pi(r)}(x_0)$ over r
- Aim directly for an optimal policy within the parametric class
- Gradient-based optimization may be a possibility
- Random search in the space of r is another possibility (e.g., cross entropy method)

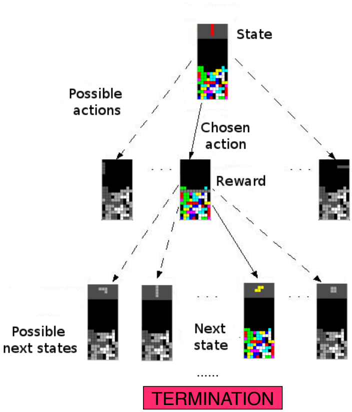
An important special case: Policy parametrization through cost function parametrization

- For a given value space parametrization $r = (r_0, \ldots, r_{N-1})$, we define

$$\tilde{\mu}_k(x_k, r_k) = \arg \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k), r_k\big) \Big\}$$

- Has achieved success in a number of test problems (e.g., tetris)

An Example: Tetris (Often Used as Testbed in Competitions)



- Number of states $> 2^{200}$ (for 10×20 board)
- $J^*(x)$: optimal score starting from board position x
- Common choice: 22 features, readily recognized by tetris players as capturing important aspects of the board position (heights of columns, etc)
- Long history of successes and failures

Lookahead Minimization

Cost-to-go Approximation

First ℓ Steps

"Future"

$$\min_{u_k,\mu_{k+1},\ldots,\mu_{k+\ell-1}} E\left\{ g_k(x_k,u_k,w_k) + \sum_{m=k+1}^{k+\ell-1} g_k\big(x_m,\mu_m(x_m),w_m\big) + \tilde{J}_{k+\ell}(x_{k+\ell}) \right\}$$

Tail problem approximation

Tail Problem Approximation Ideas

Obtain $\tilde{J}_{k+\ell}$ as the cost-to-go of a simplified problem
which is solved exactly or approximately

Enforced decomposition of interconnected subsystems

Applies to problems involving a collection I of interconnected subsystems, with each subsystem $i \in I$ applying control u_k^i at time k

- One-at-a time optimization: Obtain $\tilde{J}_{k+\ell}$ by optimizing one subsystem at a time, with controls of other subsystems fixed at nominal values
- Constraint relaxation: Artificially decouple subsystems by modifying the constraint set
- Lagrangean relaxation: Artificially decouple subsystems by using Lagrange multipliers (we will not cover)

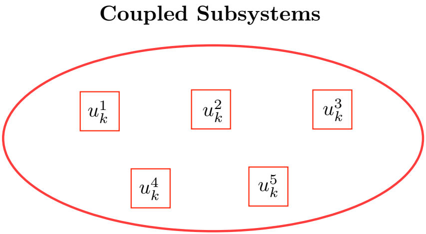
Probabilistic approximation

Simplify the probabilistic structure (e.g., replace random variables with deterministic)

Aggregation

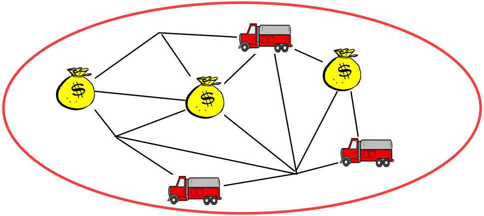
Reduce the size of the problem; e.g., by “combining" states into aggregate states

Enforced Decomposition: One Subsystem at a Time

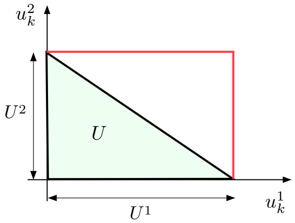


- Let $u_k = (u_k^1, \ldots, u_k^n)$, with u_k^i corresponding to the ith subsystem
- To compute cost-to-go approximation $\tilde{J}_k(x_k)$:
  - Start with subsystem 1, optimize over $(u_k^1, \ldots, u_{N-1}^1)$, with all future controls of other subsystems $i \neq 1$ held at nominal values $(\tilde{u}_k^i, \ldots, \tilde{u}_{N-1}^i)$
  - Fix the nominal values of subsystem 1 to the optimal sequence thus obtained
  - Repeat for all subsystems $i = 2, \ldots, n$ (with intermediate adjustment of the nominal control values)

Example: Optimize the Routes of n Vehicles Through a Road Network



- Aim: Execute a number of tasks with given values
- The value of a task is collected only once; a finite horizon is assumed
- This is a very complex combinatorial problem
- The single vehicle problem is typically much simpler (e.g., can be solved exactly or with a high-quality heuristic)
- Solve (suboptimally) the tail subproblem one-vehicle-at-a-time. The nominal decisions of the other vehicles can be determined using some heuristic



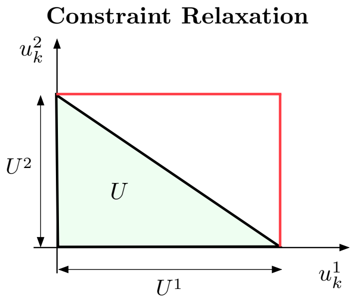
- Let $x_k = (x_k^1, \dots, x_k^n)$, $u_k = (u_k^1, \dots, u_k^n)$, $w_k = (w_k^1, \dots, w_k^n)$, with (x_k^i, u_k^i, w_k^i) corresponding to the ith subsystem
- Assume that the only coupling between subsystems is the control constraint

$$(u_k^1, \dots, u_k^n) \in U, \qquad \text{e.g., } u_k^i \in U^i,\ u_k^1 + \cdots + u_k^n \le b_k$$

- Approximate U with a decomposed constraint $U^1 \times \dots \times U^n$
- The problem decomposes into n decoupled subproblems. Let $\tilde{J}_k^i$ be the optimal cost to go functions for the ith decoupled subproblem (obtained by DP off-line)
- Use one-step lookahead with cost-to-go approximation

$$\tilde{J}_{k+1}(x_{k+1}) = \tilde{J}_{k+1}^1(x_{k+1}^1) + \cdots + \tilde{J}_{k+1}^n(x_{k+1}^n)$$

Example: Production Planning



A work center producing n product types

- x_k^i, u_k^i, w_k^i: the amounts stored, produced, and demanded of product i at time k
- State is the stock vector $x_k = (x_k^1, \ldots, x_k^n)$, where $x_{k+1}^i = x_k^i + u_k^i - w_k^i$
- U represents the (shared) production capacity of the work center
- In a more complex version (involving equipment failures), U depends on a random parameter α_k that changes according to a Markov chain

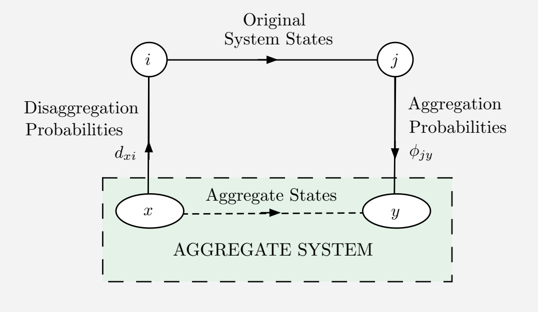
Probabilistic Approximation

Modify the probability distributions $P(w_k \mid x_k, w_k)$ to simplify the calculation of $\tilde{J}_{k+\ell}$ and/or the lookahead minimization

Certainty equivalent control (inspired by linear-quadratic control problems)

- Replace uncertain quantities with deterministic nominal values
- The lookahead and tail problems are deterministic so they can be solved by DP or by special deterministic methods
- Use expected values or forecasts to determine nominal values; update policy when forecasts change (on-line replanning)
- A variant: Partial certainty equivalence. Fix only some uncertain quantities to nominal values
- A generalization: Approximate $E\{\cdot\}$ by limited simulation

Tail Problem Approximation by Aggregation



- Construct a “smaller" aggregate tail problem by introducing aggregate states
- Use the exact costs-to-go of the aggregate tail problem as approximate costs-to-go for the original

Aggregation examples:

- State discretization-interpolation schemes
- Grouping of states into subsets, which serve as aggregate states
- Feature-based discretization; aggregate problem operates in the space of features

Concluding Remarks

What we covered

- Approximate DP for finite horizon problems with perfect state information
- Approximation in value space
- Approximation in policy space; possibly in combination with approximation in value space

What we did not cover

- Approximate DP for infinite horizon problems
  - Lookahead and rollout ideas apply with essentially no change
  - Special training methods for approximation in value space
  - Temporal difference methods [e.g., TD(λ) and others]; TD(λ) is closely related with the proximal algorithm, but implemented by simulation (see internet videolecture)
- Imperfect state information problems can be converted to (much more complex) perfect state information problems. Approximate DP is essential for any kind of solution
- A variety of important lookahead/approximation in value space schemes: Model predictive control, open-loop feedback control, and others
- Alternative cost criteria: minimax/games, multiplicative/exponential cost, etc
- Approximation error bound analysis

Thank you!