Reinforcement Learning and Optimal Control

ASU, CSE 691, Winter 2019

Dimitri P. Bertsekas
dimitrib@mit.edu

Lecture 3

Outline

1. Approximation in Value and Policy Space

2. General Issues of Approximation in Value Space

3. Special Multistep Lookahead Issues

4. Problem Approximation Schemes

Recall the Stochastic DP Algorithm

Produces the optimal costs $J_k^*(x_k)$ of the tail subproblems that start at x_k

Start with $J_N^*(x_N) = g_N(x_N)$, and for $k = 0, \ldots, N-1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big)\Big\}, \qquad \text{for all } x_k.$$

- The optimal cost $J^*(x_0)$ is obtained at the last step: $J_0^*(x_0) = J^*(x_0)$.

Online implementation of the optimal policy, given $J_1^*, \ldots, J_{N-1}^*$

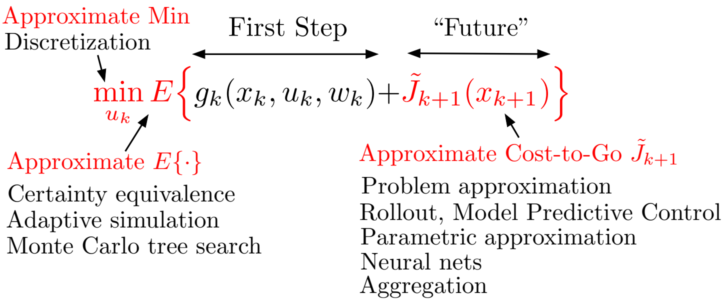
Sequentially, going forward, for $k = 0, 1, \ldots, N-1$, observe x_k and apply

$$u_k^* \in \arg\min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big)\Big\}.$$

The main difficulties: Too much computation, too much memory storage.

Approximation in value space: Use $\tilde{J}_k$ in place of J_k^*; possibly approximate $E\{\cdot\}$ and $\min_{u_k}$.

Approximation in Value Space: One-Step Lookahead



Approximate Min

Discretization

First Step

"Future"

$$\min_{u_k} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(x_{k+1}) \Big\}$$

Approximate $E\{\cdot\}$

Certainty equivalence

Adaptive simulation

Monte Carlo tree search

Approximate Cost-to-Go $\tilde{J}_{k+1}$

Problem approximation

Rollout, Model Predictive Control

Parametric approximation

Neural nets

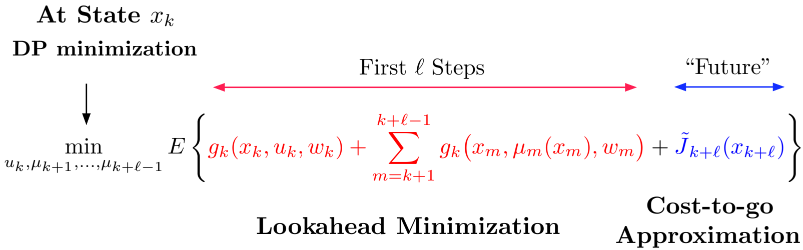
Aggregation

Approximation in value space uses $\tilde{J}_{k+1}$ (in place of J^*_{k+1}) and lookahead minimization, to construct suboptimal control law $\tilde{\mu}_k$ at time k.

Three main Issues; they can be addressed separately

- How to construct $\tilde{J}_k$, $k = 1, \ldots, N$.
- How to simplify $E\{\cdot\}$ operation.
- How to simplify min operation.

Approximation in Value Space: Multistep Lookahead



At State x_k

DP minimization

First ℓ Steps

"Future"

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

Lookahead Minimization

Cost-to-go Approximation

- At state x_k, we solve an ℓ-stage version of the DP problem with x_k as the initial state and $\tilde J_{k+\ell}$ as the terminal cost function.
- Use the first control of the ℓ-stage policy thus obtained, while discarding the others.

Can view ℓ-step lookahead as a special case of one-step lookahead:

The "effective" one-step lookahead function is the optimal cost function of an $(\ell - 1)$-stage DP problem with terminal cost $\tilde J_{k+\ell}$.

Approximation in Policy Space: The Major Alternative to Approximation in Value Space

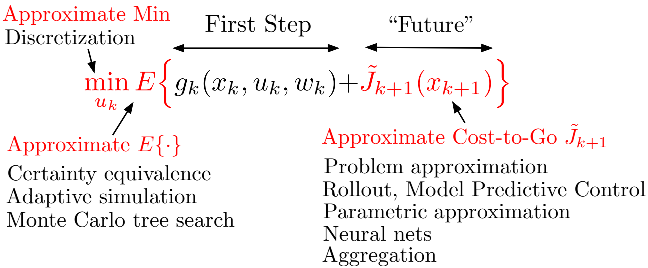
- Idea: Select the policy by optimization over a suitably restricted class of policies.
- The restricted class is usually a parametric family of policies $\mu_k(x_k, r_k)$, $k = 0, \ldots, N-1$, of some form, where r_k is a parameter (e.g., a neural net).
- Important advantage once the parameters r_k are computed: The computation of controls during on-line operation of the system is often much easier: At state x_k apply $u_k = \mu_k(x_k, r_k)$.

Approximation in policy space on top of approximation in value space

- Compute approximate cost-to-go functions $\tilde J_{k+1}$, $k = 0, \ldots, N-1$.
- This defines the corresponding suboptimal policy $\tilde\mu_k$, $k = 0, \ldots, N-1$, through one-step or multistep lookahead.
- Approximate $\tilde\mu_k$ using some form of regression and a training set consisting of a large number q of sample pairs $\big(x_k^s, u_k^s\big)$, $s = 1, \ldots, q$, where $u_k^s = \tilde\mu_k(x_k^s)$.
- Example: Introduce a parametric family of policies $\mu_k(x_k, r_k)$, $k = 0, \ldots, N-1$, of some form, where r_k is a parameter. Then estimate the parameters r_k by

$$r_k \in \arg\min_r \sum_{s=1}^{q} \big\| u_k^s - \mu_k(x_k^s, r) \big\|^2.$$

On-Line and Off-Line Lookahead Implementations



Approximate Min

Discretization

First Step

"Future"

$$\min_{u_k} E\Big\{ g_k(x_k,u_k,w_k) + \tilde J_{k+1}(x_{k+1}) \Big\}$$

Approximate $E\{\cdot\}$

Certainty equivalence

Adaptive simulation

Monte Carlo tree search

Approximate Cost-to-Go $\tilde J_{k+1}$

Problem approximation

Rollout, Model Predictive Control

Parametric approximation

Neural nets

Aggregation

- For many-state problems, the minimizing controls $\tilde\mu_k(x_k)$ are computed on-line (storage issue).
- Off-line methods: All the functions $\tilde J_{k+1}$ are computed for every k, before the control process begins.
- Examples of off-line methods: Neural network and other parametric approximations; also aggregation.
- On-line methods: The values $\tilde J_{k+1}(x_{k+1})$ are computed only at the relevant next states x_{k+1}, and are used to compute the control to be applied at the N time steps.
- Examples of on-line methods: Rollout and model predictive control.
- Rollout is well-suited for on-line replanning, but lots of on-line computation.

Simplifying the Minimization of the Expected Value in Lookahead Schemes

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

- If $U_k(x_k)$ is a finite set, the minimization can be done by brute force.
- If $U_k(x_k)$ is an infinite set, it may be replaced by a finite set through discretization.
- For deterministic problems and continuous control spaces, a more efficient alternative may be to use nonlinear programming techniques.
- For stochastic problems and continuous control spaces, we may use stochastic programming. Lookahead must be short because of the high branching factor of the lookahead tree when the problem is stochastic.

One possibility to deal with the $E\{\cdot\}$:

Assumed certainty equivalence, i.e., choose a typical value $\tilde{w}_k$ of w_k, and use the control $\tilde{\mu}_k(x_k)$ that solves the deterministic problem

$$\min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k, \tilde{w}_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, \tilde{w}_k)\big) \Big]$$

However, this may degrade performance significantly.

Model-Based Versus Model-Free Implementation

Our layman's use of the term "model-free": A method is called model-free if it involves calculations of expected values using Monte Carlo simulation.

Model-free is necessary when:

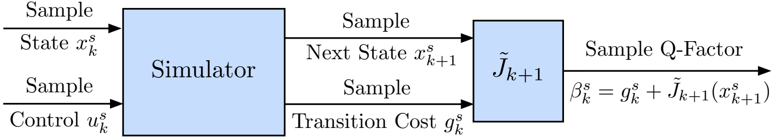
- A mathematical model of the probabilities $p_k(w_k \mid x_k, u_k)$ is not available but a computer model/simulator is. For any (x_k, u_k), it simulates sample probabilistic transitions to a successor state x_{k+1}, and generates the corresponding transition costs.
- When for reasons of computational efficiency we prefer to compute the expected value by using sampling and Monte Carlo simulation; e.g., approximate an integral or a huge sum of numbers by a Monte Carlo estimate.

Principal example: Calculations of approximate Q-factors in lookahead schemes

$$E\Big\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big)\Big\}$$

(assuming $\tilde{J}_{k+1}$ has been computed).

Model-Free Q-Factor Calculation for Stochastic Problems



- Use the simulator to collect a large number of “representative" samples of state-control-successor states-stage cost quadruplets $(x_k^s, u_k^s, x_{k+1}^s, g_k^s)$, and corresponding sample Q-factors

$$\beta_k^s = g_k^s + \tilde J_{k+1}(x_{k+1}^s), \qquad s = 1, \dots, q$$

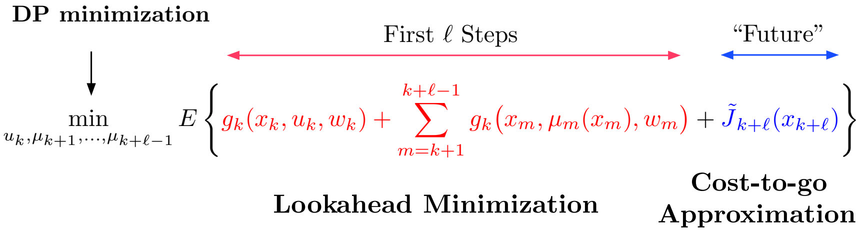
- Introduce a parametric family of Q-factors $\tilde Q_k(x_k, u_k, r_k)$.
- Determine the parameter vector $\bar r_k$ by the least-squares regression

$$\bar r_k \in \arg\min_{r_k} \sum_{s=1}^{q} \left(\tilde Q_k(x_k^s, u_k^s, r_k) - \beta_k^s\right)^2$$

- Use the policy

$$\tilde\mu_k(x_k) \in \arg\min_{u_k \in U_k(x_k)} \tilde Q_k(x_k, u_k, \bar r_k)$$

Multistep Lookahead Issues



DP minimization

First ℓ Steps

"Future"

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

Lookahead Minimization

Cost-to-go Approximation

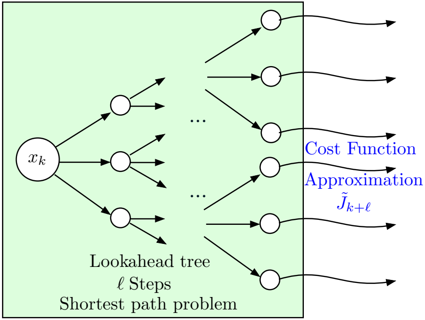
Main hope

- Minimization over many steps will work better than minimization over few steps (with long enough lookahead we are optimal).
- By using a long-step lookahead, we can afford a simpler/less accurate cost-to-go approximation.

Main Issue

Minimization over many stages is costly; stochastic problems are harder because of a larger branching factor of the lookahead tree.

Multistep Lookahead and Deterministic Problems



If the problem is deterministic and finite-state, the lookahead minimization is a shortest path problem and may be solved on-line.

If the problem is deterministic and continuous-state/control, the lookahead minimization may be quickly solvable by nonlinear programming (model predictive control case).

If the problem is stochastic and finite-state, the lookahead minimization can be split into a first stochastic step and a deterministic remainder; i.e., use a deterministic shortest path problem approximation for the remaining steps.

Let’s Take a Working Break to Consider the Following Challenge Questions

Question 1

Consider one-step lookahead with two different cost function approximations $\tilde{J}_{k+1}$ and $\hat{J}_{k+1}$. Assume that $\tilde{J}_{k+1}$ is “much closer" to J^*_{k+1} (in any way you may think of).

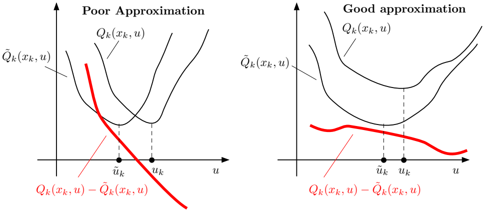
Will $\tilde{J}_{k+1}$ produce a better policy than $\hat{J}_{k+1}$?

Question 2

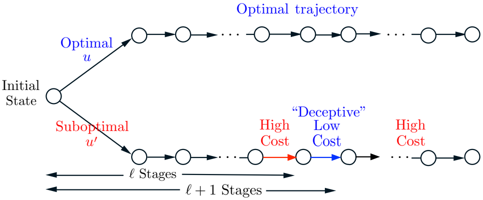
Consider multistep lookahead.

Will longer lookahead produce a better policy than shorter lookahead?

The Answers are NO and NO



Constant shift in Q-factor does not affect the minimizing control. For a good suboptimal policy, the “slope" of the Q-factor difference $Q_k(x_k, u) - \tilde{Q}_k(x_k, u)$ should be small.



Problem with “edge effects": Two controls, u (optimal) and u' (suboptimal), and cost function approximation $\tilde{J}_k(x_k) \equiv 0$. u will be preferred based on ℓ-step lookahead. u' will be preferred based on $(\ell + 1)$-step lookahead.

Recall Lookahead Methods Involving Problem Approximation

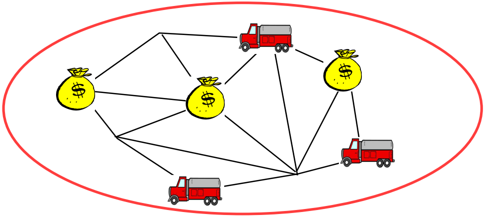
$$\tilde{\mu}_k(x_k) \in \arg \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\}$$

$\tilde{J}_{k+1}$ is the optimal cost function of a simpler problem
How this is done is typically problem-dependent

Examples

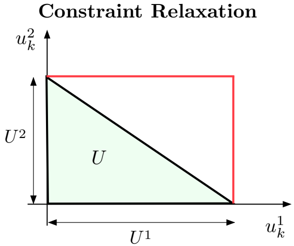
- Enforced decomposition of systems that consist of weakly coupled subsystems. Leads to single subsystem computations.
- Probabilistic approximation. Enforced certainty equivalence. Leads to deterministic minimizations with no expected values to compute.
- Aggregation. Construct a “smaller" aggregate problem by introducing aggregate states.

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).
- At a given state: Solve (suboptimally) the tail subproblem one-vehicle-at-a-time.

Enforced Decomposition: Constraint Decoupling by Relaxation



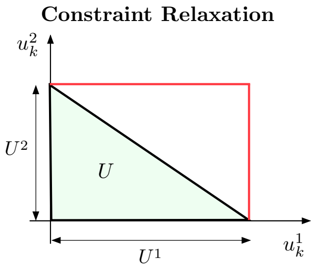
- Let $x_k = (x_k^1, \ldots, x_k^n)$, $u_k = (u_k^1, \ldots, u_k^n)$, $w_k = (w_k^1, \ldots, w_k^n)$, with (x_k^i, u_k^i, w_k^i) corresponding to the ith subsystem.
- Assume that the only coupling between subsystems is the control constraint

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

- Approximate U with a decomposed constraint $U^1 \times \ldots \times U^n$.
- 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 the 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 $x_k = (x_k^1, \dots, x_k^n)$, 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

Another example in the text: "Restless" multiarmed bandit problems. (Select one out of *n* projects to work on at each stage.)

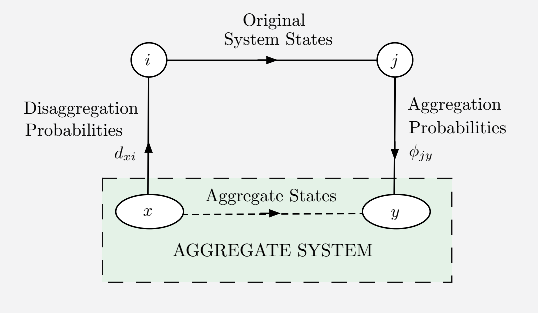
Probabilistic Approximation

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 could be solvable by DP or by special deterministic methods on-line.
- Use expected values or forecasts to determine nominal values; update policy when forecasts change (on-line replanning).
- Use state estimates instead of belief states.
- A variant: Partial certainty equivalence. Fix only some uncertain quantities to nominal values.
- A generalization: Approximate $E\{\cdot\}$ by limited simulation.

Problem Approximation by Aggregation (to be discussed in detail later)



- Construct a “smaller" aggregate 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.

About the Next Lecture

We will cover:

- Rollout for deterministic and stochastic problems
- Monte Carlo tree search
- Model predictive control

PLEASE READ AS MUCH OF SECTIONS 2.3, 2.4 AS YOU CAN

PLEASE DOWNLOAD THE LATEST VERSIONS FROM MY WEBSITE