Reinforcement Learning and Optimal Control

ASU, CSE 691, Winter 2019

Dimitri P. Bertsekas
dimitrib@mit.edu

Lecture 5

Outline

Recall Approximation in Value Space

Approximate Min

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

ONE-STEP 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

MULTISTEP LOOKAHEAD

At State x_k

DP minimization

First ℓ Steps

"Future"

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

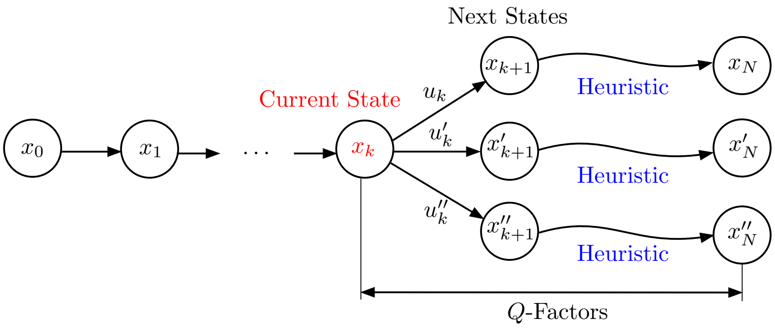
Lookahead Minimization

Heuristic Cost
Run the Base Policy

Use a suboptimal/heuristic policy at the end of limited lookahead

- The heuristic is called base policy (or default policy).
- The lookahead policy is called rollout policy.
- Policy improvement; connection with policy iteration.
- Involves simulation and on-line implementation; suitable for on-line replanning.
- Deterministic rollout lends itself to on-line implementation.

General Structure of Deterministic Rollout

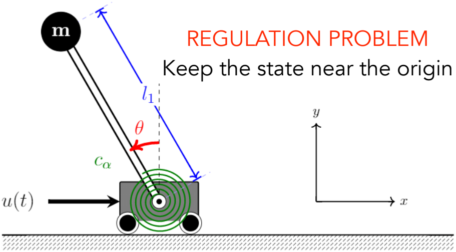


- At state x_k, for every pair (x_k, u_k), $u_k \in U_k(x_k)$, we generate a Q-factor

$$\tilde{Q}_k(x_k, u_k) = g_k(x_k, u_k) + H_{k+1}\big(f_k(x_k, u_k)\big)$$

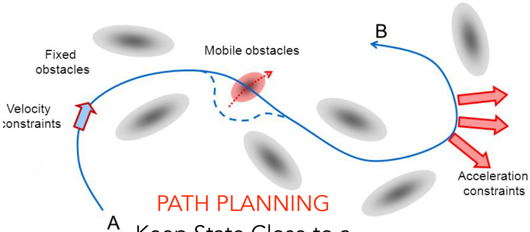
using the base heuristic [$H_{k+1}(x_{k+1})$ is the heuristic cost starting from x_{k+1}].

- We select the control u_k with minimal Q-factor.
- We move to next state x_{k+1}, and continue.
- A key question for today's lecture: What if we have a continuous/infinite control set?



REGULATION PROBLEM

Keep the state near the origin

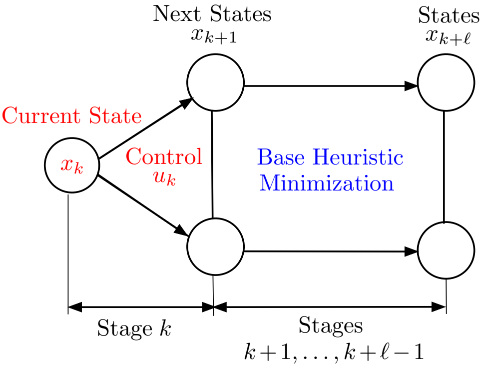


PATH PLANNING

Keep State Close to a Trajectory

Need to deal with state and control constraints; linear-quadratic is often inadequate

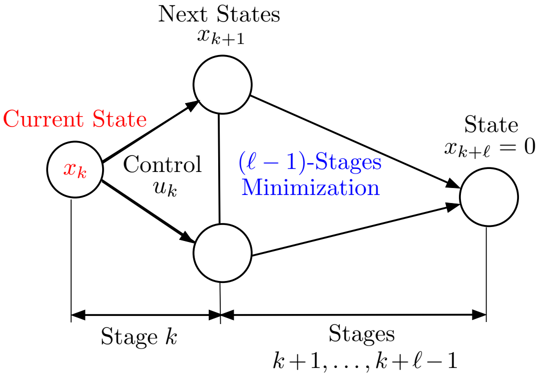
On-Line Rollout for Deterministic Infinite-Spaces Problems



Suppose the control space is infinite

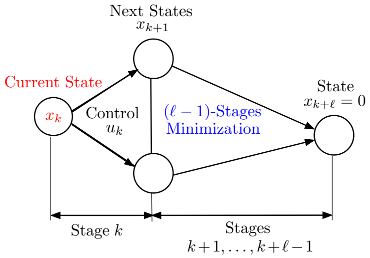
- One possibility is discretization of $U_k(x_k)$; but excessive number of Q-factors.
- Another possibility is to use optimization heuristics that look $(\ell - 1)$ steps ahead.
- Seemlessly combine the kth stage minimization and the optimization heuristic into a single ℓ-stage deterministic optimization.
- Can solve it by nonlinear programming/optimal control methods (e.g., quadratic programming, gradient-based).

Model Predictive Control for Regulation Problems



- System: $x_{k+1} = f_k(x_k, u_k)$
- Cost per stage: $g_k(x_k, u_k) \geq 0$, the origin 0 is cost-free and absorbing.
- State and control constraints: $x_k \in X_k$, $u_k \in U_k(x_k)$ for all k
- At x_k solve an ℓ-step lookahead version of the problem, requiring $x_{k+\ell} = 0$ while satisfying the state and control constraints.
- If $\{\tilde{u}_k, \ldots, \tilde{u}_{k+\ell-1}\}$ is the control sequence so obtained, apply $\tilde{u}_k$.

Relation to Rollout

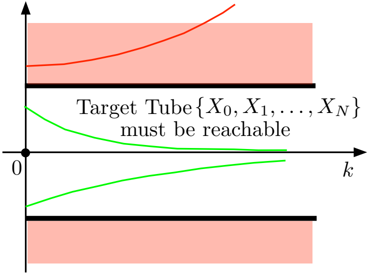


- It is rollout with base heuristic the $(\ell - 1)$-step min (0 is cost-free and absorbing).
- This heuristic is sequentially improving (not sequentially consistent)

$$\min_{u_k \in U_k(x_k)} \left[g_k(x_k, u_k) + H_{k+1}\big(f_k(x_k, u_k)\big)\right] \leq H_k(x_k)$$

  where $H_k(x_k)$, $H_{k+1}(x_{k+1})$: optimal heuristic costs starting at x_k and x_{k+1}.
- Sequential improvement implies "stability": $\sum_{k=0}^{\infty} g_k(x_k, u_k) \leq H_0(x_0) < \infty$, where $\{x_0, u_0, x_1, u_1, \ldots\}$ is the state and control sequence generated by MPC.
- Major issue: How do we know that the optimization of the base heuristic is solvable (e.g., there exists ℓ such that we can drive $x_{k+\ell}$ to 0 for all $x_k \in X_k$ while observing the state and control constraints).

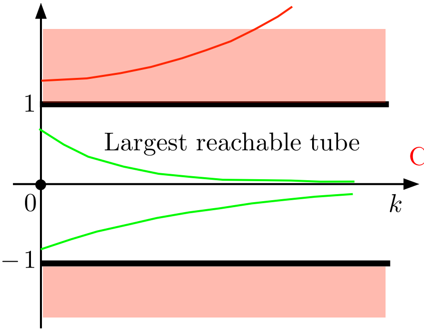
Reachability of Target Tubes (DPB, 1969, PhD Thesis)



- The tube of state constraint sets $\{X_0, X_1, \ldots, X_N\}$ is reachable if the state x_k can be kept within it for all k when the initial state x_0 belongs to X_0.
- If $\{X_0, X_1, \ldots, X_N\}$ is not reachable, MPC will not work; if it is reachable MPC will "typically" work. We may try to extract a reachable subset $\{\overline{X}_0, \overline{X}_1, \ldots, \overline{X}_N\}$, with $\overline{X}_k \subset X_k$, for all k. Then use $\overline{X}_k$ in place of X_k.
- Reachability algorithm: Start with $\overline{X}_N = X_N$, and proceed backwards

$$\overline{X}_k = \{x_k \in X_k \mid \text{for some } u_k \in U_k(x_k) \text{ we have } f_k(x_k, u_k) \in \overline{X}_{k+1}\}.$$

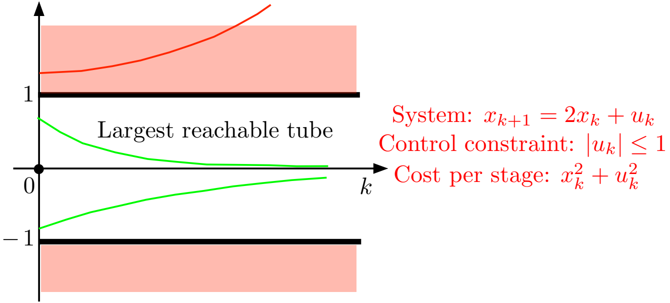
- Generally, it is difficult to compute the sets $\overline{X}_k$ of the target tube, but algorithms that produce inner approximations have been constructed.



System: $x_{k+1} = 2x_k + u_k$
Control constraint: $|u_k| \leq 1$
Cost per stage: $x_k^2 + u_k^2$

- Is it true that $\{(-1, 1), (1, 1), \ldots, (-1, 1)\}$ is the largest reachable tube?
- Is the tube
$$\{[-2, 2], [-2, 2], \ldots, [-2, 2]\}$$
reachable? How about the tube
$$\{[-1/2, 1/2], [-1/2, 1/2], \ldots, [-1/2, 1/2]\}$$
- How will MPC with $\ell = 2$ work starting from $x_0 = 1/2$ and from $x_0 = 2$?

Some Answers (see the textbook for details)



- If $|x_k| > 1$ the state cannot be brought back towards 0; if $|x_k| < 1$ it can.
- If $|x_k| \le 1/2$ the state can be driven to 0 in one step; if $1/2 < |x_k| < 1$ the state can be driven to 0 in finitely many steps (the number increases as $|x_k|$ is closer to 1).
- If $|x_k| = 1$ the state can at best be kept where it is.
- $\{[-1, 1], [-1, 1], \ldots, [-1, 1]\}$ is the largest reachable tube.
- $\{(-1, 1), (1, 1), \ldots, (-1, 1)\}$ is the largest tube from within which the state can be driven to 0 in a finite number of steps.
- For $\ell = 2$, MPC must start from $|x_0| \le 1/2$. It has the form $\tilde{u}_k = -(5/3)x_k$. It is stable and drives the state to 0 asymptotically (not in a finite number of steps).

Parametric Approximation in Value Space

Approximation Architectures

- A class of functions $\tilde{J}(x, r)$ that depend on x and a vector $r = (r_1, \ldots, r_m)$ of m "tunable" scalar parameters (or weights).
- We adjust r to change $\tilde{J}$ and "match" the cost function approximated.
- Training the architecture: The algorithm to choose r (typically use data/regression).
- Architectures are linear or nonlinear, depending on whether $\tilde{J}(x, r)$ is linear or nonlinear in r.
- Architectures are feature-based if they depend on x via a feature vector $\phi(x)$,

$$\tilde{J}(x, r) = \hat{J}\big(\phi(x), r\big),$$

  where $\hat{J}$ is some function. Idea: Features capture dominant nonlinearities.
- A linear feature-based architecture:

$$\tilde{J}(x, r) = \sum_{\ell=1}^{m} r_\ell \phi_\ell(x),$$

  where r_ℓ and $\phi_\ell(x)$ are the ℓth components of r and $\phi(x)$.
- Local vs global: Change in a single weight affects $\tilde{J}$ locally vs globally.

Generic Example Architectures

- Piecewise constant approximation (local): Partition the state space into subsets $S_1, \ldots, S_m$. Let the ℓth feature be defined by membership in the set S_ℓ, i.e., $\phi_\ell(x) = 1$ if $s \in S_\ell$ and $\phi_\ell(x) = 0$ if $s \notin S_\ell$. The architecture

$$\tilde{J}(x, r) = \sum_{\ell=1}^{m} r_\ell \phi_\ell(x),$$

  is piecewise constant with value r_ℓ for all x within the set S_ℓ.

- Quadratic polynomial approximation (global): $\tilde{J}(x, r)$ is quadratic in the components x^i of x. Consider features

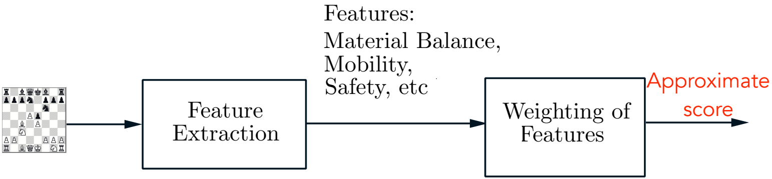
$$\phi_0(x) = 1, \qquad \phi_i(x) = x^i, \qquad \phi_{ij}(x) = x^i x^j, \quad i, j = 1, \ldots, n.$$

  A linear feature-based approximation architecture:

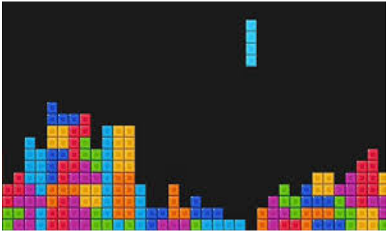
$$\tilde{J}(x, r) = r_0 + \sum_{i=1}^{n} r_i x^i + \sum_{i=1}^{n} \sum_{j=i}^{n} r_{ij} x^i x^j$$

  The parameter vector r has components r_0, r_i, and r_{ij}.

- General polynomial architectures: Polynomials in the components $x^1, \ldots, x^n$. Another possibility: Polynomials of features.

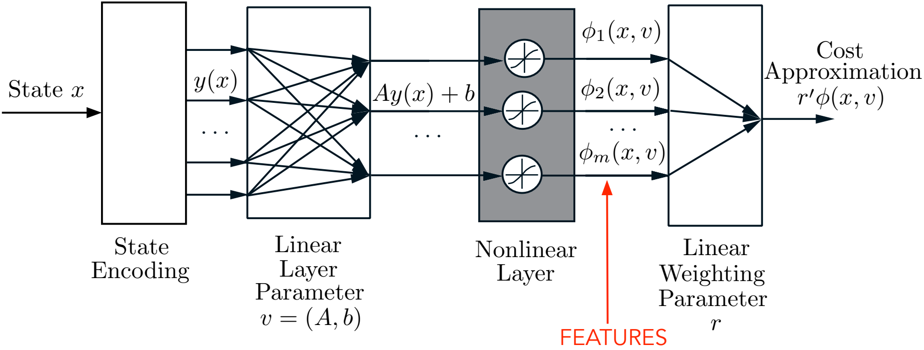


Chess



Tetris

Neural Nets: An Architecture that does not Require Knowledge of Features



About the Next Lecture

We will cover:

- Training of parametric approximation architectures
- Neural networks; how do we use
- Sequential Dynamic Programming Approximation
- Q-factor Parametric Approximation

PLEASE READ AS MUCH OF SECTIONS 3.1.3, 3.2-3.4 AS YOU CAN

PLEASE DOWNLOAD THE LATEST VERSIONS FROM MY WEBSITE