Topics in Reinforcement Learning: Lessons from AlphaZero for (Sub)Optimal Control and Discrete Optimization

Arizona State University
Course CSE 691, Spring 2022

Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html

Dimitri P. Bertsekas
dbertsek@asu.edu

Lecture 4

Adaptive Control
A Closer Look at Value and Policy Approximations in DP/RL

Outline

Overview of What we Have Done

- The first four lectures (including this one) are “big picture" lectures aiming to draw the broad conceptual framework of our subject and course
- Our focal point is approximation in value space and the AlphaZero paradigm (appropriately generalized): Using on-line play to improve on the results of off-line training
- Future lectures will go in greater depth into selected parts of the picture

What we have done so far in summary

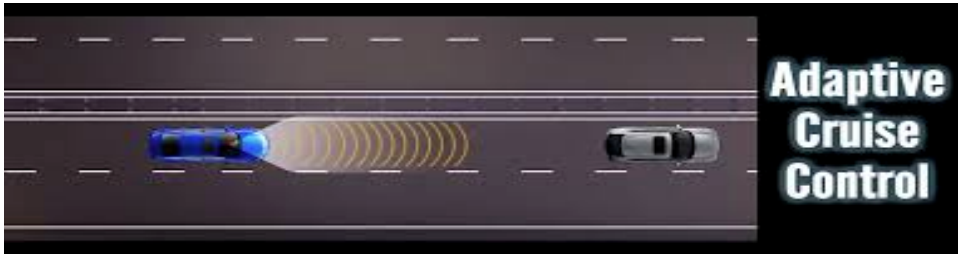
- Lecture 1: AlphaZero as a motivating example, off-line training and on-line play, generalities about RL/DP, exact finite horizon deterministic DP, examples
- Lecture 2: Exact finite horizon stochastic DP, overview of infinite horizon DP, examples, linear quadratic problems
- Lecture 3: Approximation in value space and the Newton method viewpoint in the context of linear quadratic problems. Examples of problem formulations and reformulations, including multiagent problems, POMDP, etc
- Lecture 4 (today): Broad issues of adaptive and model predictive control. Overview of conceptual and implementation aspects of approximation in value space and its connections to approximation in policy space

Overview of Where we are Going

Our plan for future lectures: We will cover in some depth and detail

- Rollout for finite horizon discrete-deterministic and stochastic problems
- More on model predictive control
- More on multiagent problems
- Off-line training using neural networks and approximation architectures
- Infinite horizon problems in greater theoretical detail
- Infinite horizon approximate DP/RL. Focus on variants of PI
- Off-line training of policies by policy gradient and random search methods
- Aggregation

Changing Problem Parameters: Adaptive Control (1960s $\rightarrow$)



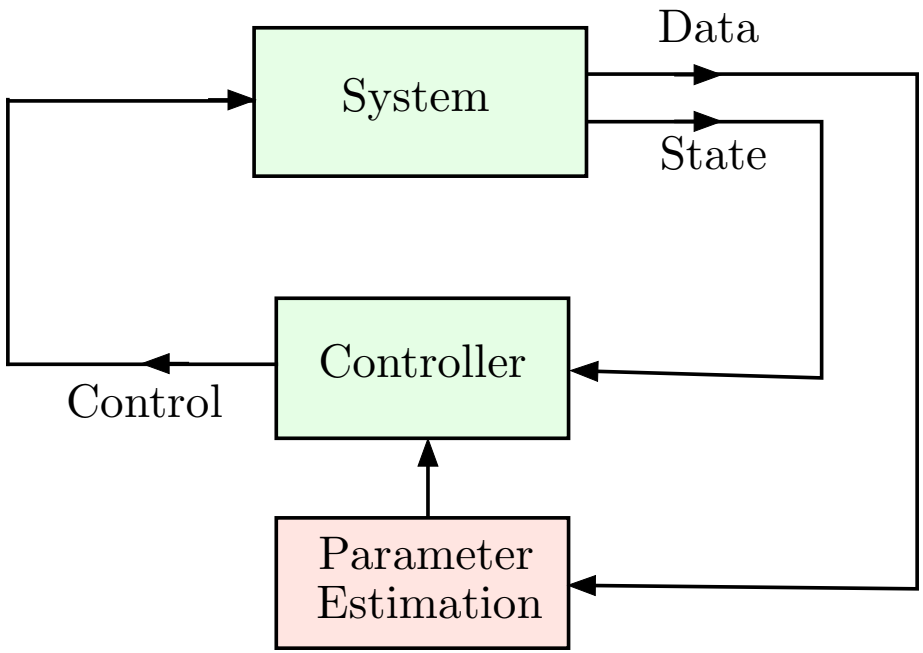
Example: A cruise control-type problem (keep velocity close to a target level)

- Control car velocity: $x_{k+1} = ax_k + bu_k$ ($a < 1$ models friction, wind drag, etc, $b > 0$ depends on road, number of passengers, etc)
- Cost over N stages: $(x_N - \bar{x})^2 + \sum_{k=0}^{N-1} \left((x_k - \bar{x})^2 + ru_k^2\right)$, where $r > 0$ is given
- ... but a, b, and $\bar{x}$ are changing all the time; they may be measured with error (?)

Adaptive control deals with such situations. Some possibilities:

- Ignore the changes in parameters; design a controller that is robust ("works" for a broad range of parameters). PID control is a time-honored relevant methodology
- Try to estimate the parameters, and use the estimates to modify the controller
  - On-line replanning by optimization; modify the controller to make it optimal for the current set of estimates. This is sophisticated/time consuming: needs on-line system identification, and optimal control computation. Has other pitfalls (identifiability; see notes)...
  - On-line replanning by rollout with a base policy whose cost function is computed using the current parameter estimates. This is a simpler (approximate) reoptimization ...

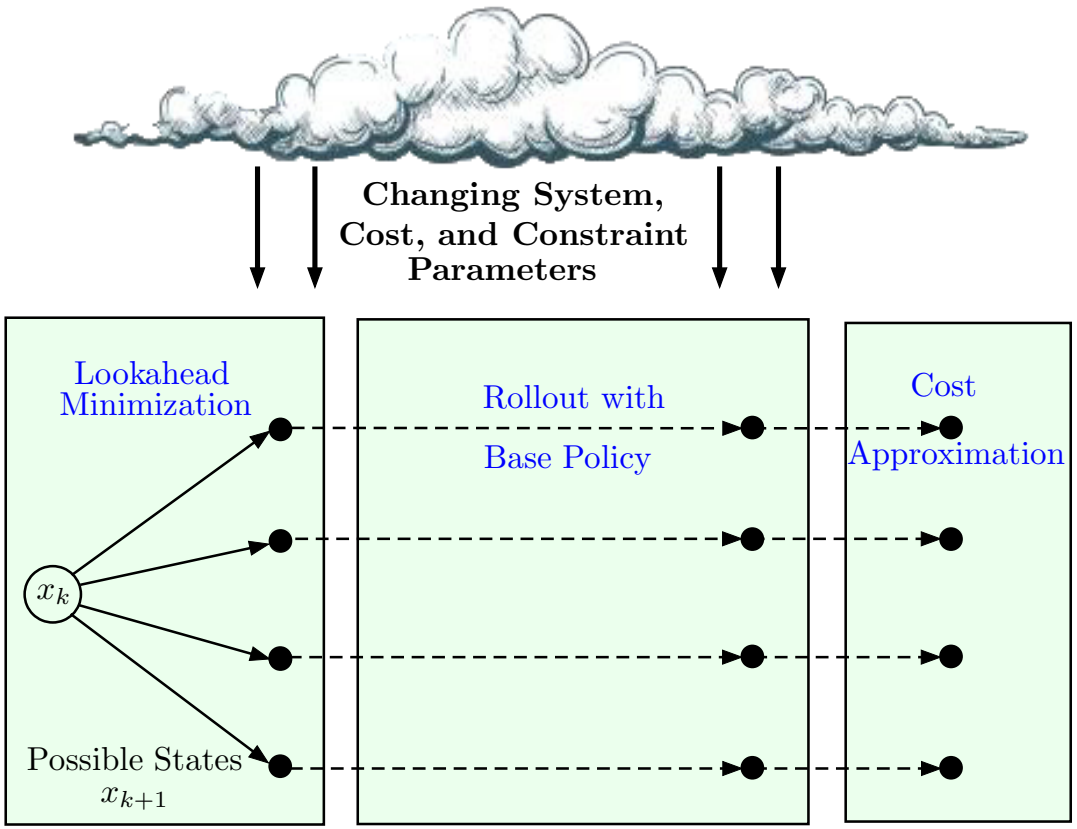
On-Line Replanning by Optimization (Indirect Adaptive Control)



Introduce on-line estimation of changing parameters

- Recompute the controller so it is optimal for the new set of parameters
- This can be time-consuming, so a suboptimal controller may be used instead (e.g., based on approximation in value space)

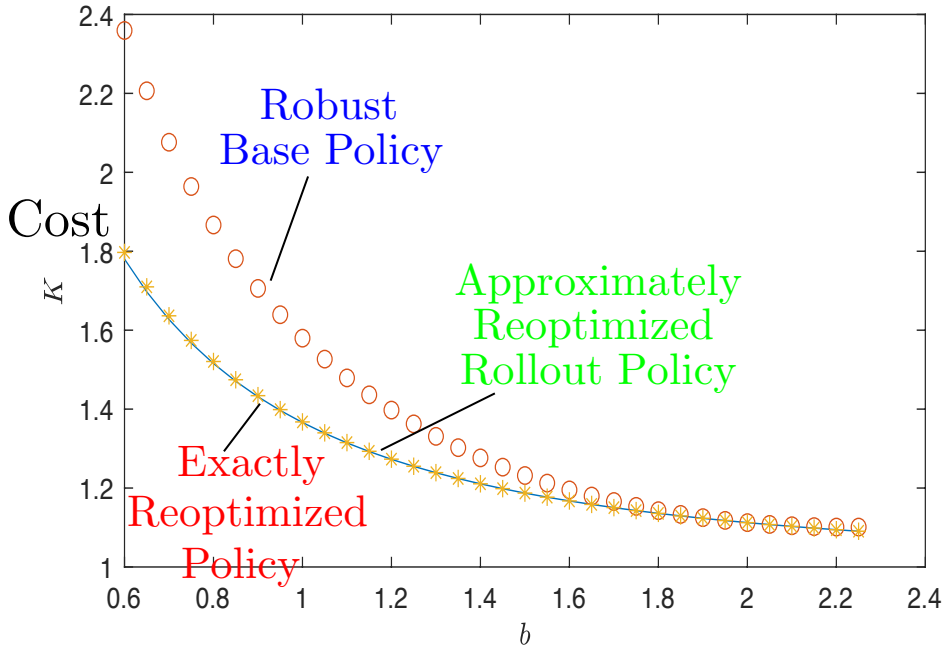
On-Line Replanning by Rollout/Approximation in Value Space



Use on-line replanning with rollout instead of controller reoptimization; this is faster

- Introduce new parameter estimates in the lookahead minimization and the rollout
- Continue to use the same base policy
- Possibly recalculate the base policy in the background

A Linear-Quadratic Example of On-Line Replanning



Performance comparison of on-line replanning by rollout and by optimization

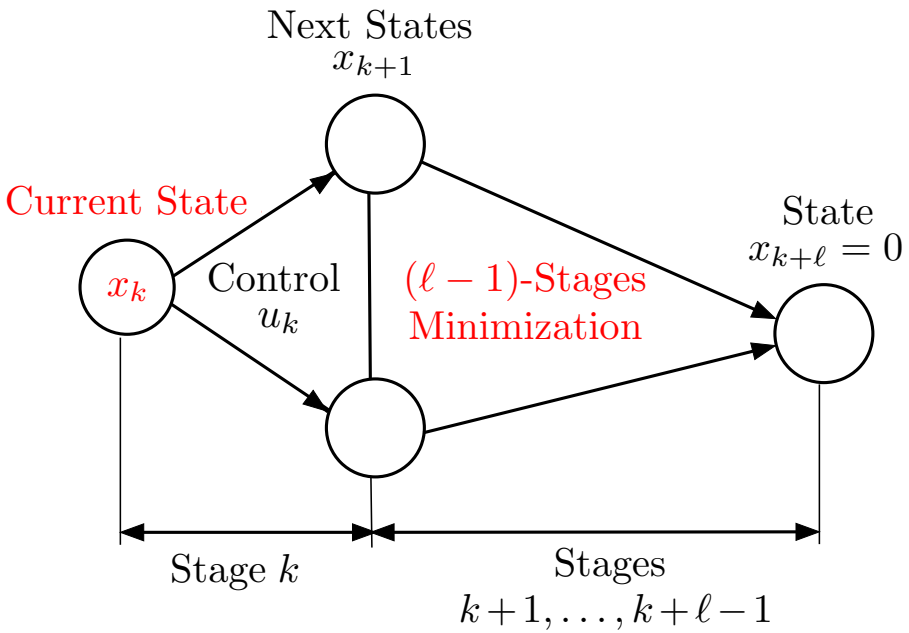
Note the effect of Newton's method

One-dimensional linear-quadratic example:

$$x_{k+1} = x_k + bu_k, \qquad \text{Cost} = \lim_{N\to\infty} \sum_{k=0}^{N-1} (x_k^2 + ru_k^2)$$

Quadratic cost coefficient as b changes. Base policy is optimal for $b = 2$

Model Predictive Control - A Form of Approximation in Value Space



System: $x_{k+1} = f(x_k, u_k)$

Cost: $g(x_k, u_k) \geq 0$, for all (x_k, u_k)

The system can be kept at the origin at zero cost by some control

Consider undiscounted infinite horizon; we want to keep the system near 0

- Original form of MPC: We minimize the cost function over the next ℓ stages while requiring $x_{k+\ell} = 0$
- We apply the first control of the minimizing sequence, discard the other controls
- This is rollout w/ base heuristic the min that drives $x_{k+\ell}$ to 0 in $(\ell - 1)$ steps
- Well-suited for on-line replanning
- A variant that uses a terminal cost approximation instead of $x_{k+\ell} = 0$; can be viewed as rollout/approximation in value space with single or multistep lookahead

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

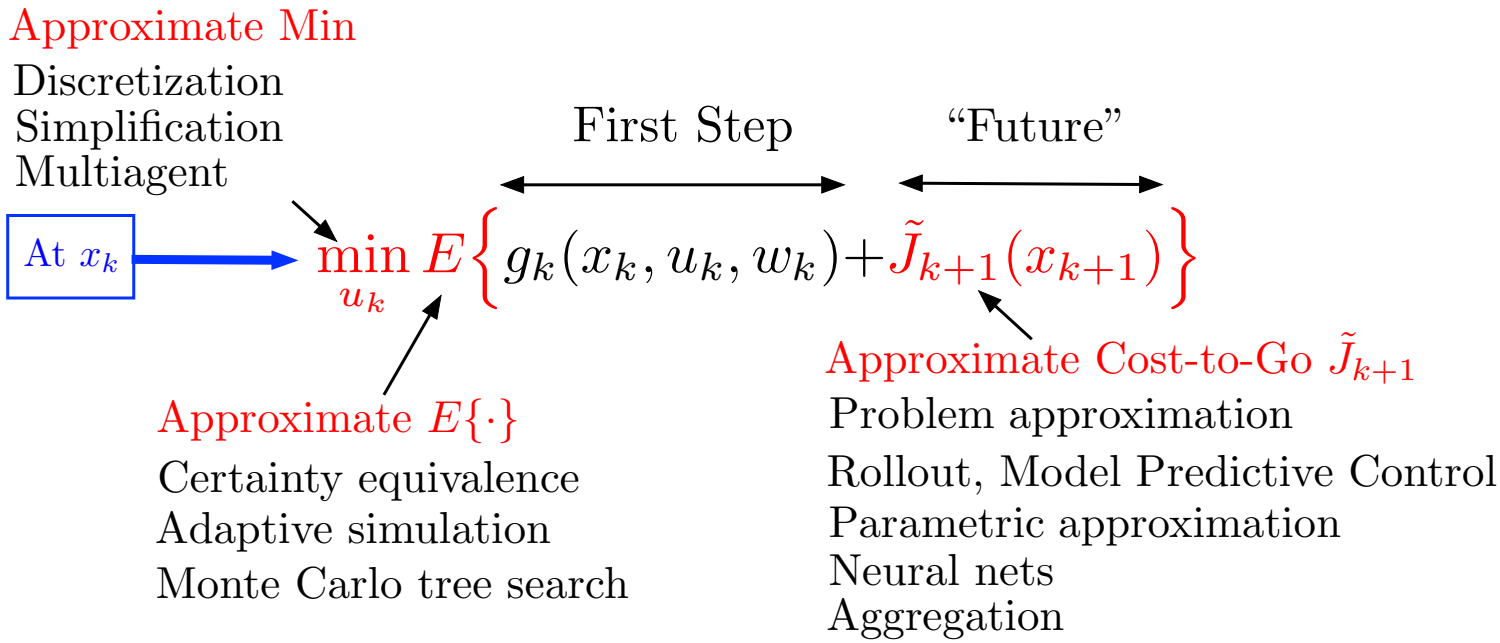
$$u_k^* \in \arg \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big) \Big\}.$$

The main difficulties: Too much computation, too much memory storage for J_k^*.

Approximation in value space:

Use $\tilde{J}_{k+1}$ in place of J_{k+1}^*

Approximation in Value Space: One-Step Lookahead



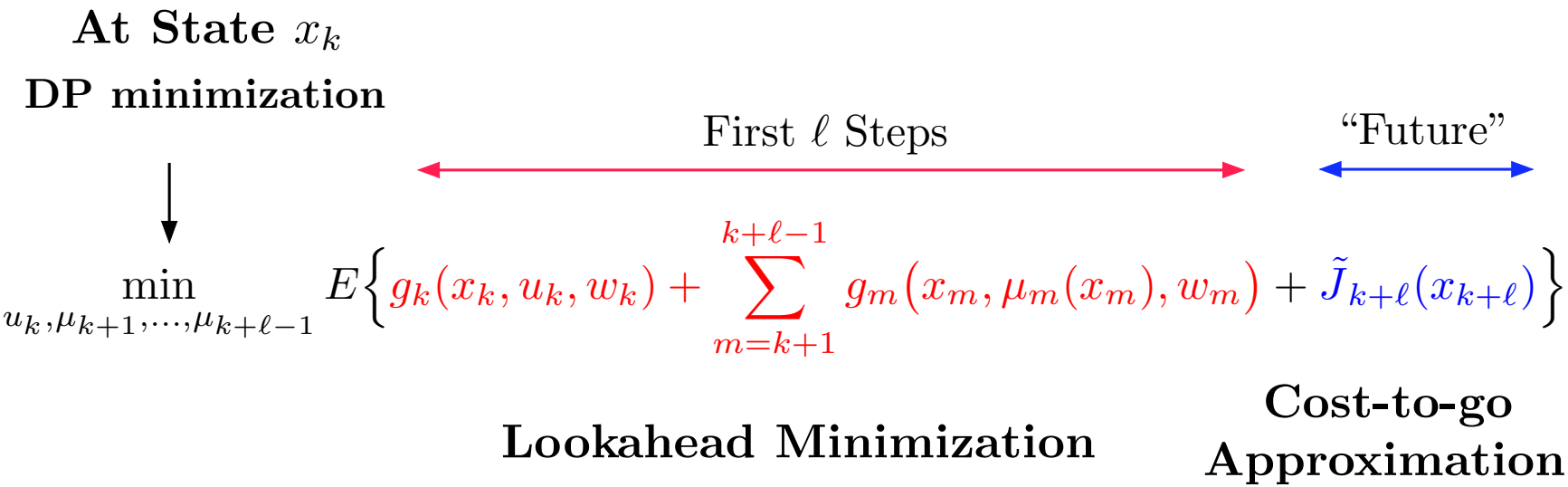
Approximation in value space and one-step lookahead minimization defines:

- The control to use at state x_k for on-line play
- A suboptimal control law $\tilde{\mu}_k$

The three approximations; they can be addressed separately

- How to construct the cost function approximations $\tilde{J}_{k+1}$?
- How to simplify $E\{\cdot\}$ operation?
- How to simplify min operation?

Approximation in Value Space: Multistep Lookahead

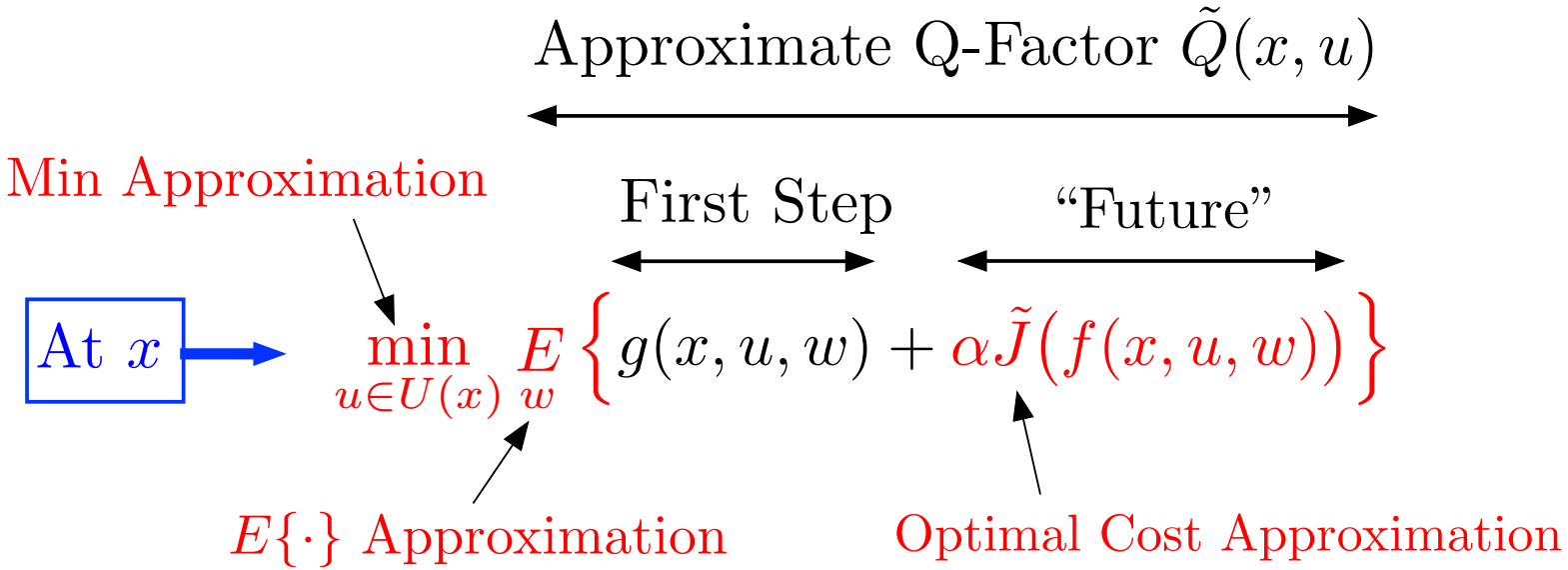


- At state x_k, we solve an ℓ-stage version of the DP problem with x_k as the initial state and $\tilde{J}_{k+\ell}$ as the terminal cost function
- Use the first control of the ℓ-stage policy thus obtained, discard the others

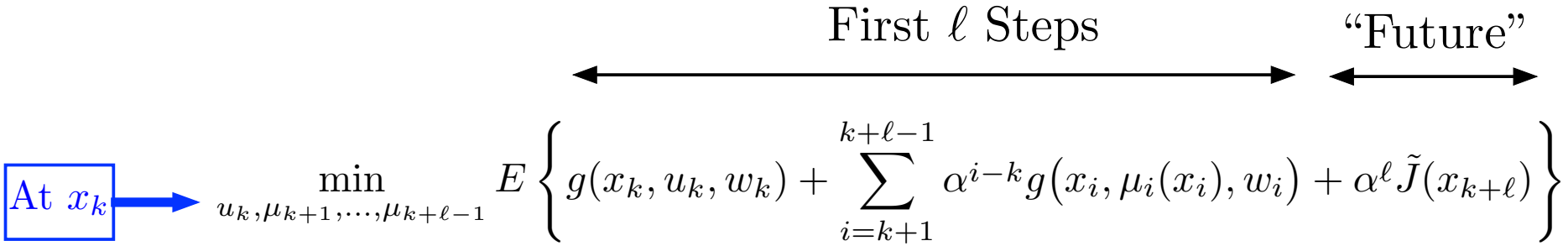
We can view ℓ-step lookahead as a special case of one-step lookahead:

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

The three (largely independent) approximations apply

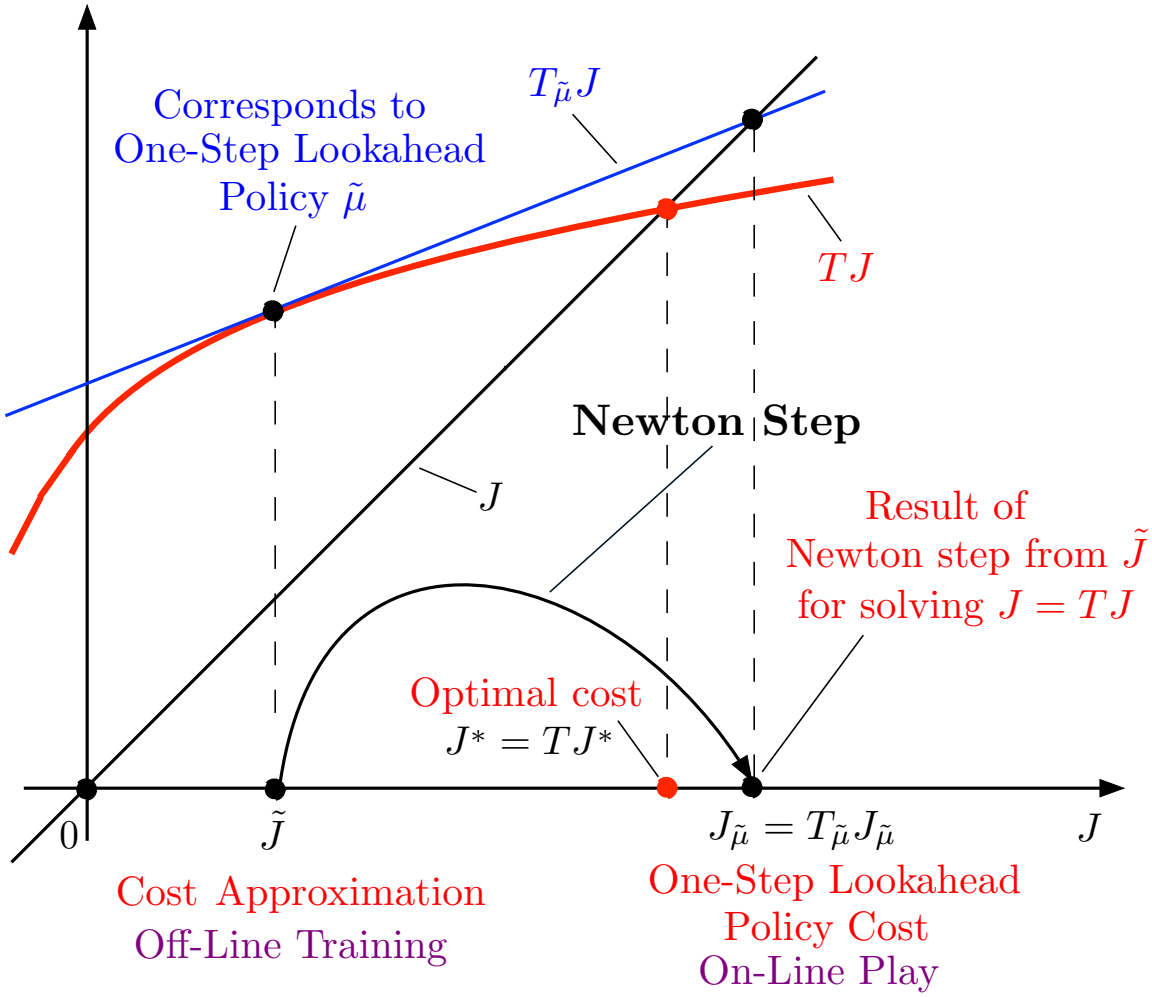


One-Step Lookahead



Multistep Lookahead

Newton Step View of One-Step Lookahead Minimization



$$(TJ)(x) = \min_{u \in U(x)} E\Big\{ g(x,u,w) + \alpha J\big(f(x,u,w)\big) \Big\}, \qquad \text{for all } x$$

$$(T_\mu J)(x) = E\Big\{ g\big(x,\mu(x),w\big) + \alpha J\big(f(x,\mu(x),w)\big) \Big\}, \qquad \text{for all } x$$

Local and Global Performance Estimates

- The Newton step ($\tilde{J}$ to $J_{\tilde{\mu}}$) interpretation suggests a local superlinear performance estimate (for $\tilde{J}$ near J^*)

$$\max_x \left|J_{\tilde{\mu}}(x) - J^*(x)\right| = o\Big(\max_x \left|\tilde{J}(x) - J^*(x)\right|\Big)$$

- When $\tilde{J}$ is far from J^*, the difference $\max_x \left|J_{\tilde{\mu}}(x) - J^*(x)\right|$ may be large and even infinite when $\tilde{\mu}$ is unstable
- There are global estimates (don't depend on $\tilde{J} \approx J^*$), including

$$\max_x \left|J_{\tilde{\mu}}(x) - J^*(x)\right| \leq \frac{2\alpha^\ell}{1-\alpha} \max_x \left|\tilde{J}(x) - J^*(x)\right|$$

  for ℓ-step lookahead, and α-discounted problems
- These global estimates tend to be overly conservative and not representative of the performance of approximation in value space schemes when $\tilde{J}$ is near J^*
- Example: For finite spaces α-discounted MDP, $\tilde{\mu}$ can be shown to be optimal if

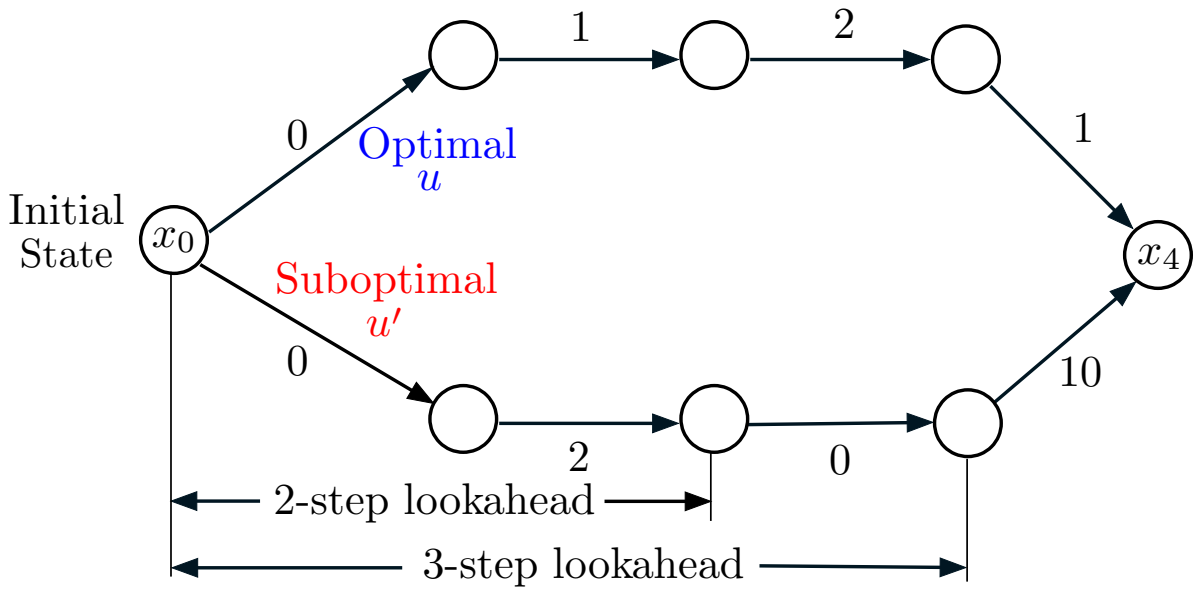
$$\max_x \left|\tilde{J}(x) - J^*(x)\right|$$

  is sufficiently small (the Bellman operator is piecewise linear). The global performance bound is way off!

Let's Take a Break; Consider the Following Challenge Question

Will longer lookahead produce a better policy than shorter lookahead?

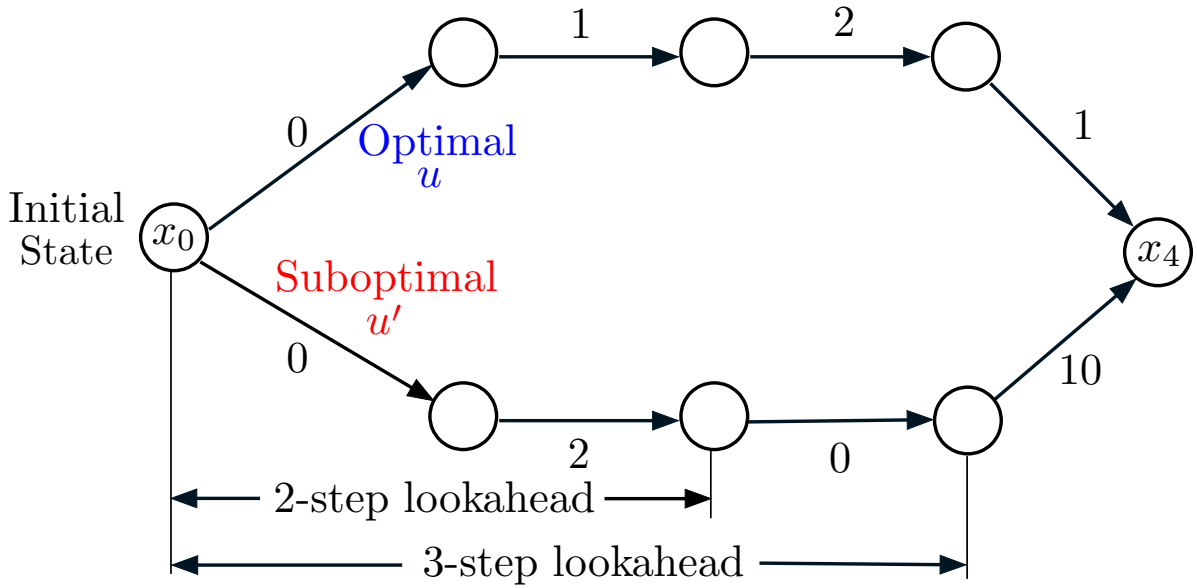
Consider the following example



Two controls, u, u', and cost function approximation $\tilde{J}_k(x_k) \equiv 0$.

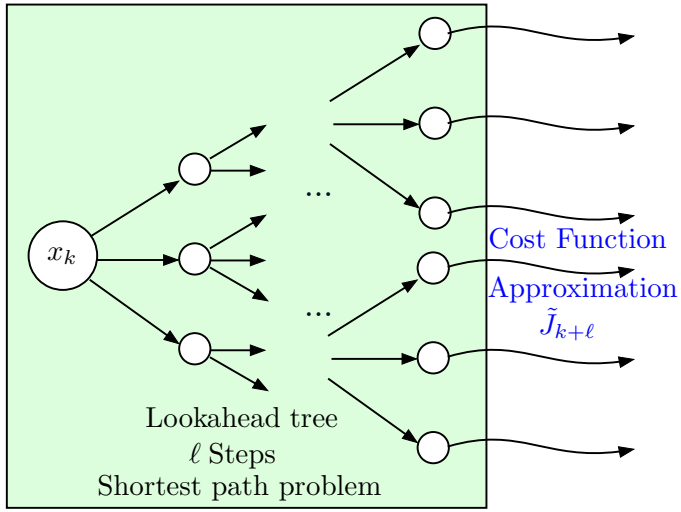
There is a choice only at x_0.

The Answer is that “Usually" Longer is Better, but NOT for this Example



Problem with “edge effects": u will be preferred based on 2-step lookahead. u' will be preferred based on 3-step lookahead

Multistep Lookahead with Deterministic Optimizations



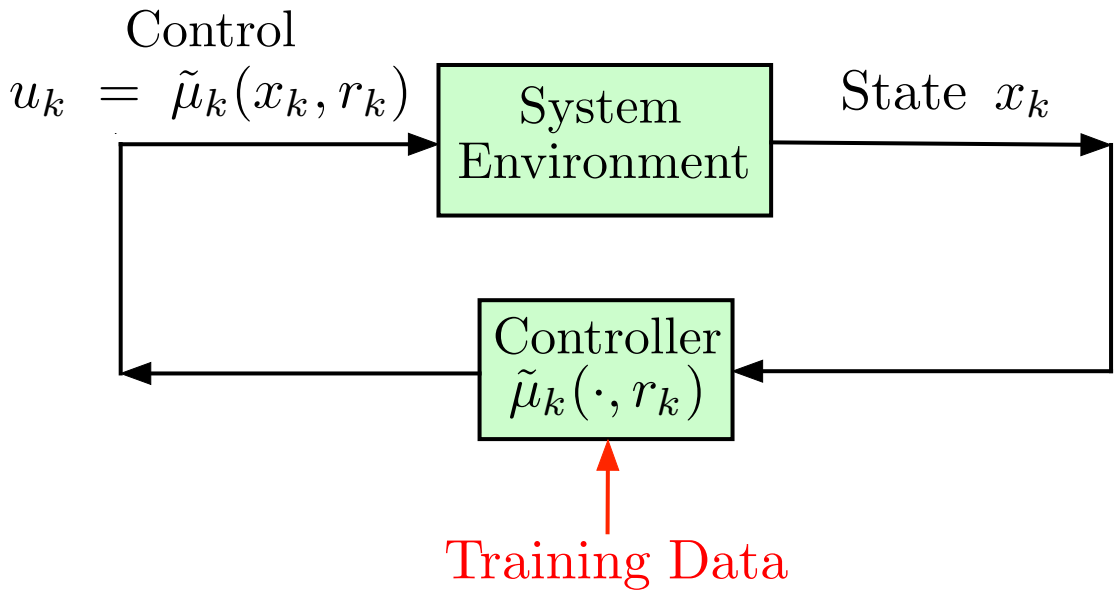
If the problem is deterministic with finite control space, the lookahead minimization is a shortest path problem and may be solved on-line

If the problem has continuous-state/control

- In the deterministic case, the lookahead minimization may be quickly solvable by nonlinear programming (MPC case)
- In the stochastic case, the lookahead minimization may be solvable by stochastic programming (a deterministic opt. method to be discussed later)

If the problem is stochastic with finite state and control spaces, 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

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 $\tilde{\mu}_k(x_k, r_k)$, $k = 0, \ldots, N-1$, of some form, where r_k is a parameter (e.g., a neural net)
- Methods used for optimization: Random search, policy gradient, classification (to be discussed later)
- Important advantage once the parameters r_k are computed: The on-line computation of controls is often much faster ... at state x_k apply $u_k = \tilde{\mu}_k(x_k, r_k)$
- Important disadvantage: It does not allow for on-line replanning ... no Newton step

The approximate cost-to-go functions $\tilde{J}_{k+1}$ define a suboptimal policy $\tilde{\mu}_k$ through one-step or multistep lookahead minimization

- Given functions $\tilde{J}_{k+1}$, how do we simplify the computation of $\tilde{\mu}_k$?
- Idea: Approximate $\tilde{\mu}_k$ using some form of least squares and a training set of a large number q of sample pairs (x_k^s, u_k^s), $s = 1, \ldots, q$, where $u_k^s = \tilde{\mu}_k(x_k^s)$:

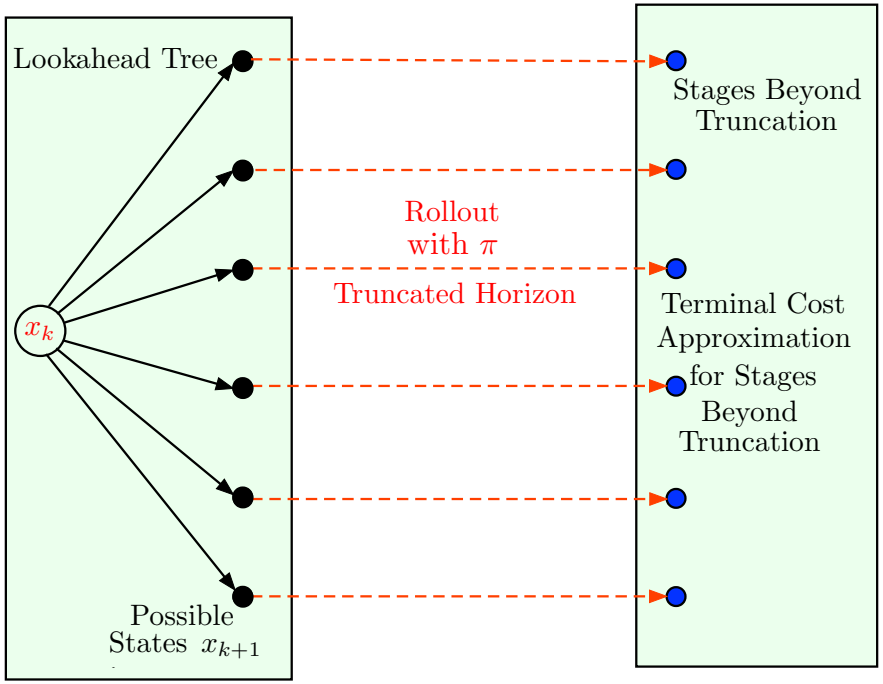
$$u_k^s \in \arg \min_{u \in U_k(x_k)} E\Big\{ g_k(x_k^s, u, w_k) + \tilde{J}_{k+1}\big(f_k(x_k^s, u, w_k)\big) \Big\}$$

  Similarly for multistep lookahead. (But the Newton step interpretation is lost.)
- Example (for finite number of controls): Introduce a parametric family of randomized policies $\mu_k(x_k, r_k)$, $k = 0, \ldots, N-1$, of some form (e.g., a neural net), where r_k is a parameter. Then estimate the parameters r_k by least squares fit:

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

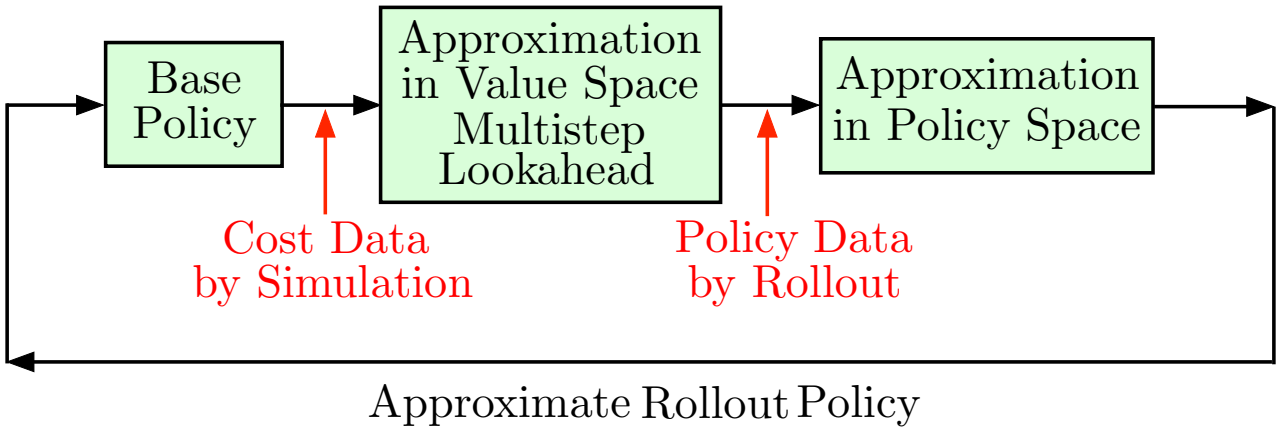
- Relation to classification methods ... policy <–> classifier; more on this later.

From Policies to Values by Rollout



- Start with some policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, a base policy, possibly obtained through approximation in policy space
- Use one-step or multistep lookahead rollout where $\tilde{J}_{k+1}(x_{k+1}) \approx J_{k+1,\pi}(x_{k+1})$
- The policy $\tilde{\pi} = \{\tilde{\mu}_0, \ldots, \tilde{\mu}_{N-1}\}$ thus obtained is the truncated rollout policy
- Important issue: How to compute $J_{k+1,\pi}(x_{k+1})$?
  - For deterministic problems: Run π from x_{k+1} once and accumulate stage costs
  - For stochastic problems: Run π from x_{k+1} many times and Monte Carlo average

Combined Approximation in Value and Policy Space



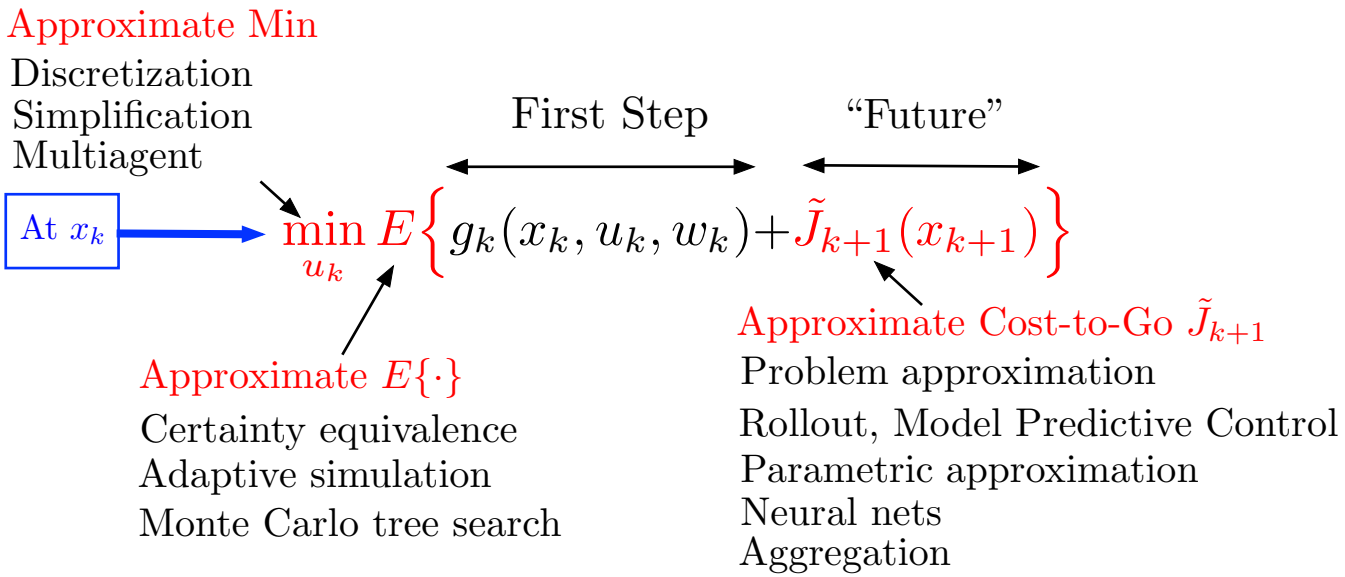
Perpetual rollout and policy improvement

- A fundamental property: In its idealized form (no approximations) each new policy has no worse cost function than the preceding one, i.e., for all x_k and k,

$$J_{k,\tilde{\pi}}(x_k) \leq J_{k,\pi}(x_k)$$

- Thus the algorithm is capable of self-improvement or self-learning (no external training data is needed)
- Its natural extension to infinite horizon problems is the policy iteration (PI) algorithm, and its foundation is the policy improvement property
- With approximations, self-improvement is approximate (to within an error bound)
- There are many variations of this scheme: Optimistic PI, Q-learning, temporal differences, etc. They involve challenging implementation issues

Balance Between Off-Line and On-Line Computations



- Off-line methods (primarily): All the functions $\tilde{J}_{k+1}$ are computed off-line for every k (also base policy, if on-line rollout is involved), before the control process begins.
- Examples of off-line methods: Use of neural network and other parametric approximations in the context of PI-like methods; also aggregation.
- Approximation in policy space is essentially an off-line method.
- On-line methods (primarily): 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 MPC.

Probabilistic Approximations: Simplifying the Expected Value Calculation

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

Assume certainty equivalence

- Replace uncertain quantities with deterministic nominal values.
- Then 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).
- A major possibility for POMDP: 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.

Model-Based vs Model-Free

What does model-free mean? Is it good or bad? There is no free lunch in RL

We will not deal with interactions with the environment for combined model identification and control (this is hard)

For us it's all model-based (but the model may be a computer/simulation model)

Monte Carlo simulation is useful when:

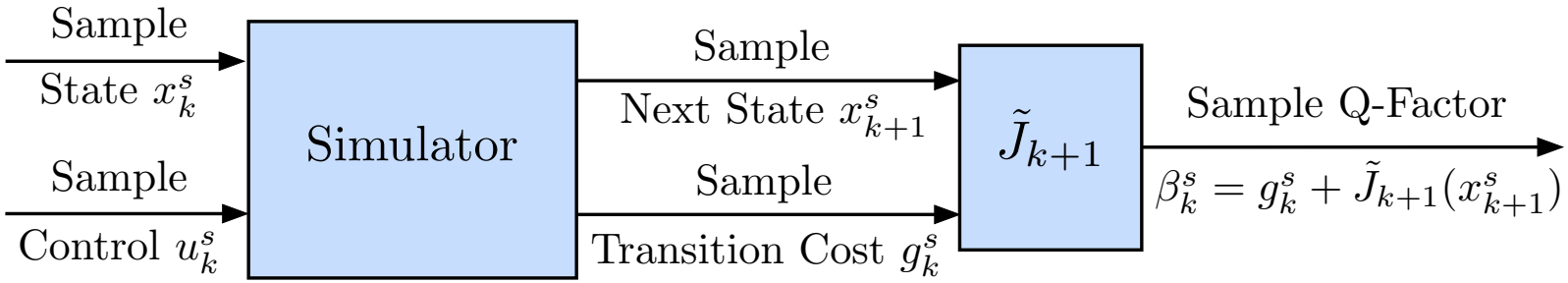
- A mathematical model of the probabilities $p_k(w_k \mid x_k, u_k)$ is not available but a computer model/simulator is. It simulates sample probabilistic transitions to a successor state x_{k+1}
- When for reasons of computational efficiency we prefer to compute expected values by using sampling and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

A common example: Model-free calculations of approximate Q-factors

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

See the next slide

Off-Line Model-Free Approximation in Value Space



- 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

$$\beta_k^s = g_k^s + \tilde{J}_{k+1}(x_{k+1}^s), \qquad s = 1, \ldots, q$$

- 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 fit

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

- Use the policy

$$\tilde{\mu}_k(x_k) \in \arg\min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k, u_k, \bar{r}_k)$$

- Note: The least squares fit introduces errors and the Newton step property is lost

About the Next Lecture

We will cover:

- Deterministic rollout and variations
- Rollout for stochastic problems

Homework to be announced next week

Watch videolecture 5 from 2021 ASU course offering