

Topics in Reinforcement Learning:
AlphaZero, ChatGPT, Neuro-Dynamic Programming,
Model Predictive Control, Discrete Optimization, Applications
Arizona State University
Course CSE 691, Spring 2025

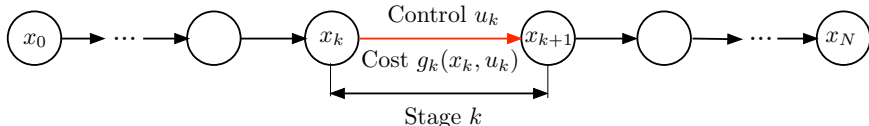
Links to Class Notes, Videolectures, and Slides at
<http://web.mit.edu/dimitrib/www/RLbook.html>

Prof. Dimitri P. Bertsekas (dimitrib@mit.edu)
and
Dr Yuchao Li (yuchaoli@asu.edu)

Lecture 2
Stochastic Finite and Infinite Horizon DP, Approximation in Value Space

- 1 Review of Finite Horizon Deterministic Problem - Approximation in Value Space
- 2 A Focus on Deterministic Rollout
- 3 Stochastic DP Algorithm
- 4 Linear Quadratic Problems - An Important Favorable Special Case
- 5 Approximation in Value Space - A Fundamental RL Approach
- 6 Approximation in Policy Space
- 7 Infinite Horizon - An Overview of Theory and Algorithms
- 8 Linear Quadratic Problems in Infinite Horizon

Review - Finite Horizon Deterministic Problem



- System

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \dots, N-1$$

where x_k : State, u_k : Control chosen from some set $U_k(x_k)$

- **Arbitrary state and control spaces** (e.g., vectors, chess positions, word strings)
- Cost function:

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

- For given initial state x_0 , minimize over control sequences $\{u_0, \dots, u_{N-1}\}$

$$J(x_0; u_0, \dots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

- Optimal cost function $J^*(x_0) = \min_{k=0, \dots, N-1} \min_{u_k \in U_k(x_k)} J(x_0; u_0, \dots, u_{N-1})$

Review - DP Algorithm for Deterministic Problems

Go backward to compute the optimal costs $J_k^*(x_k)$ of the x_k -tail subproblems (**off-line training** - involves lots of computation)

Start with

$$J_N^*(x_N) = g_N(x_N), \quad \text{for all } x_N,$$

and for $k = 0, \dots, N - 1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \left[g_k(x_k, u_k) + J_{k+1}^*(f_k(x_k, u_k)) \right], \quad \text{for all } x_k.$$

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

Go forward to construct optimal control sequence $\{u_0^*, \dots, u_{N-1}^*\}$ (**on-line play**)

Start with

$$u_0^* \in \arg \min_{u_0 \in U_0(x_0)} \left[g_0(x_0, u_0) + J_1^*(f_0(x_0, u_0)) \right], \quad x_1^* = f_0(x_0, u_0^*).$$

Sequentially, going forward, for $k = 1, 2, \dots, N - 1$, set

$$u_k^* \in \arg \min_{u_k \in U_k(x_k^*)} \left[g_k(x_k^*, u_k) + J_{k+1}^*(f_k(x_k^*, u_k)) \right], \quad x_{k+1}^* = f_k(x_k^*, u_k^*).$$

An alternative (and equivalent) form of the DP algorithm

- Generates the optimal **Q-factors**, defined for all (x_k, u_k) and k by

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + J_{k+1}^*(f_k(x_k, u_k))$$

- The optimal cost function J_k^* can be recovered from the optimal Q-factor Q_k^*

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} Q_k^*(x_k, u_k)$$

- The DP algorithm can be written in terms of Q-factors

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + \min_{u_{k+1} \in U_{k+1}(f_k(x_k, u_k))} Q_{k+1}^*(f_k(x_k, u_k), u_{k+1})$$

- Exact and approximate forms of this and other related algorithms, form an important class of RL methods known as **Q-learning**.

We replace J_k^* with an approximation \tilde{J}_k during on-line play

- Start with

$$\tilde{u}_0 \in \arg \min_{u_0 \in U_0(x_0)} \left[g_0(x_0, u_0) + \tilde{J}_1(f_0(x_0, u_0)) \right]$$

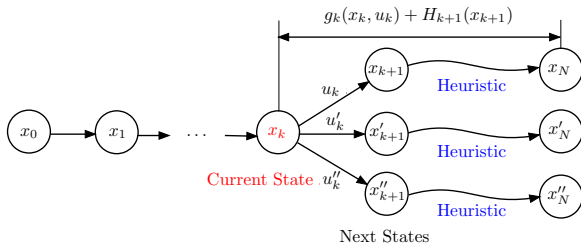
- Set $\tilde{x}_1 = f_0(x_0, \tilde{u}_0)$
- Sequentially, going forward, for $k = 1, 2, \dots, N - 1$, set

$$\tilde{u}_k \in \arg \min_{u_k \in U_k(\tilde{x}_k)} \left[g_k(\tilde{x}_k, u_k) + \tilde{J}_{k+1}(f_k(\tilde{x}_k, u_k)) \right], \quad \tilde{x}_{k+1} = f_k(\tilde{x}_k, \tilde{u}_k)$$

How do we compute $\tilde{J}_{k+1}(x_{k+1})$? This is one of the principal issues in RL

- **Off-line problem approximation:** Use as \tilde{J}_{k+1} the optimal cost function of a simpler problem, computed off-line by exact DP
- **Parametric cost approximation:** Obtain $\tilde{J}_{k+1}(x_{k+1})$ from a parametric class of functions $J(x_{k+1}, r)$, where r is a parameter, e.g., training using data and a NN.
- **Rollout with a heuristic:** We will focus on this for the moment.

Rollout for Finite-State Deterministic Problems



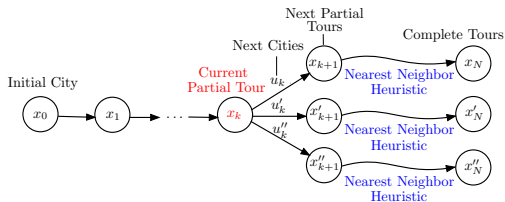
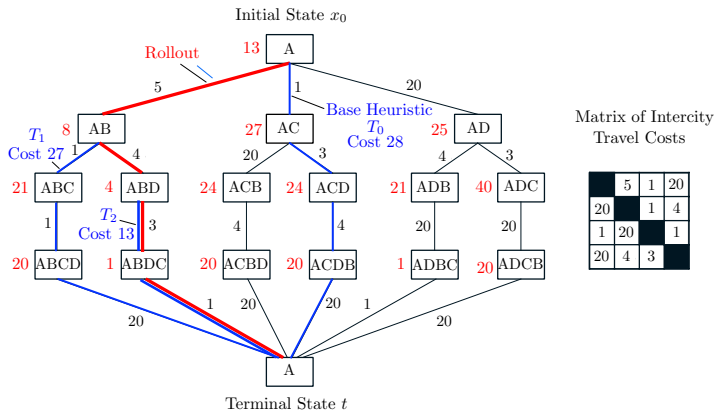
Optimal cost approximation by running a heuristic from states of interest

- Upon reaching x_k , we compute for all $u_k \in U_k(x_k)$, the corresponding next states $x_{k+1} = f_k(x_k, u_k)$
- From each of the next states x_{k+1} we run the heuristic and compute the heuristic cost $H_{k+1}(x_{k+1})$
- We apply \tilde{u}_k that minimizes over $u_k \in U_k(x_k)$, the (heuristic) Q-factor

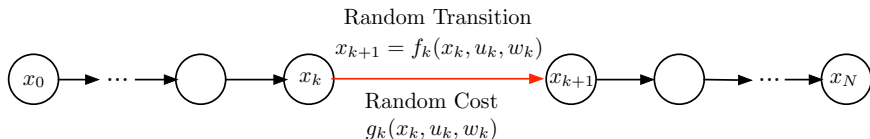
$$g_k(x_k, u_k) + H_{k+1}(x_{k+1})$$

- We generate the next state $x_{k+1} = f_k(x_k, \tilde{u}_k)$ and repeat

Example of Rollout: Traveling Salesman w/ Nearest Neighbor Heuristic



Stochastic DP Problems - Perfect State Observation (We Know x_k)



- System $x_{k+1} = f_k(x_k, u_k, w_k)$ with **random "disturbance" w_k** (e.g., physical noise, market uncertainties, demand for inventory, unpredictable breakdowns, etc)
- Cost function: $E \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\}$
- **Policies $\pi = \{\mu_0, \dots, \mu_{N-1}\}$** , where μ_k is a "closed-loop control law" or "feedback policy"/a function of x_k . A **"lookup table" for the control $u_k = \mu_k(x_k)$ to apply at x_k** .
- **An important point:** Using feedback (i.e., choosing controls with knowledge of the state) is beneficial in view of the stochastic nature of the problem.
- For given initial state x_0 , minimize over all $\pi = \{\mu_0, \dots, \mu_{N-1}\}$ the cost

$$J_\pi(x_0) = E \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

- Optimal cost function: $J^*(x_0) = \min_\pi J_\pi(x_0)$. Optimal policy: $J_{\pi^*}(x_0) = J^*(x_0)$

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 $k = 0, \dots, N - 1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}^*(f_k(x_k, u_k, w_k)) \right\}, \quad \text{for all } x_k.$$

- The optimal cost $J^*(x_0)$ is obtained at the last step: $J_0^*(x_0) = J^*(x_0)$.
- The optimal policy component μ_k^* can be constructed (off-line) simultaneously with J_k^* , and consists of the minimizing $u_k^* = \mu_k^*(x_k)$ above.

Alternative (on-line) implementation of the optimal policy, given J_1^*, \dots, J_{N-1}^*

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

$$u_k^* \in \arg \min_{u_k \in U_k(x_k)} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}^*(f_k(x_k, u_k, w_k)) \right\}.$$

Issues: Need to know J_{k+1}^* , compute $E_{w_k} \{ \cdot \}$ for each u_k , minimize over all u_k

One-dimensional linear-quadratic problem

- System is $x_{k+1} = ax_k + bu_k + w_k$ (a and b are given scalars, $E\{w_k\} = 0$)
- Cost over N stages: $qx_N^2 + \sum_{k=0}^{N-1} (qx_k^2 + ru_k^2)$, where $q > 0$ and $r > 0$ are given scalars
- The DP algorithm starts with $J_N^*(x_N) = qx_N^2$, and generates J_k^* according to

$$J_k^*(x_k) = \min_{u_k} E_{w_k} \{ qx_k^2 + ru_k^2 + J_{k+1}^*(ax_k + bu_k + w_k) \}, \quad k = 0, \dots, N-1$$

- **Main result:** DP algorithm can be carried out in closed form to yield

$$J_k^*(x_k) = K_k x_k^2 + \text{const}, \quad \mu_k^*(x_k) = L_k x_k$$

where K_k and L_k can be explicitly computed

- $\mu_k^*(x_k)$ does not depend on the distribution of w_k as long as it has 0 mean:
Certainty Equivalence (a common approximation idea for other problems)

This result generalizes to multidimensional linear-quadratic problems

$x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$; the scalars a, b, q, r are replaced by matrices A, B, Q, R .

Derivation - DP Algorithm starting from Terminal Cost $J_N^*(x_N) = qx_N^2$

$$\begin{aligned}
 J_{N-1}^*(x_{N-1}) &= \min_{u_{N-1}} E\{qx_{N-1}^2 + ru_{N-1}^2 + J_N^*(ax_{N-1} + bu_{N-1} + w_{N-1})\} \\
 &= \min_{u_{N-1}} E\{qx_{N-1}^2 + ru_{N-1}^2 + q(ax_{N-1} + bu_{N-1} + w_{N-1})^2\} \\
 &= \min_{u_{N-1}} [qx_{N-1}^2 + ru_{N-1}^2 + q(ax_{N-1} + bu_{N-1})^2 + \underbrace{2qE\{w_{N-1}\}}_{=0}(ax_{N-1} + bu_{N-1}) + q\underbrace{E\{w_{N-1}^2\}}_{=\sigma^2}] \\
 &= qx_{N-1}^2 + \min_{u_{N-1}} [ru_{N-1}^2 + q(ax_{N-1} + bu_{N-1})^2] + q\sigma^2
 \end{aligned}$$

Minimize by setting to zero the derivative: $0 = 2ru_{N-1} + 2qb(ax_{N-1} + bu_{N-1})$, to obtain

$$\mu_{N-1}^*(x_{N-1}) = L_{N-1}x_{N-1} \quad \text{with} \quad L_{N-1} = -\frac{abq}{r + b^2q}$$

and by substitution, $J_{N-1}^*(x_{N-1}) = K_{N-1}x_{N-1}^2 + q\sigma^2$, where $K_{N-1} = \frac{a^2rq}{r+b^2q} + q$

Similarly, going backwards (starting with $K_N = q$), we obtain for all k :

$$J_k^*(x_k) = K_k x_k^2 + \sigma^2 \sum_{m=k}^{N-1} K_{m+1}, \quad \mu_k^*(x_k) = L_k x_k, \quad K_k = \frac{a^2 r K_{k+1}}{r + b^2 K_{k+1}} + q, \quad L_k = -\frac{ab K_{k+1}}{r + b^2 K_{k+1}}$$

Observations and generalizations

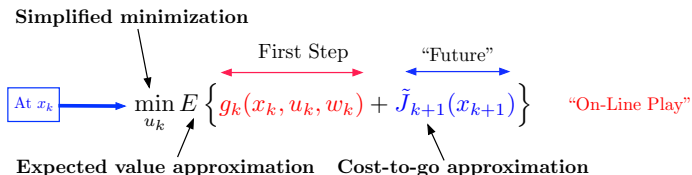
- The solution does not depend on the distribution of w_k , only on the mean (which is 0), i.e., we have **certainty equivalence** (the stochastic problem can be replaced by a deterministic problem)
- Generalization to **multidimensional problems**, nonzero mean w_k , etc
- Generalization to **infinite horizon**
- Generalization to problems where the **state is observed partially through linear measurements**: Optimal policy involves an extended form of certainty equivalence

$$L_k E\{x_k \mid \text{measurements}\}$$

where $E\{x_k \mid \text{measurements}\}$ is provided by an estimator (e.g., Kalman filter)

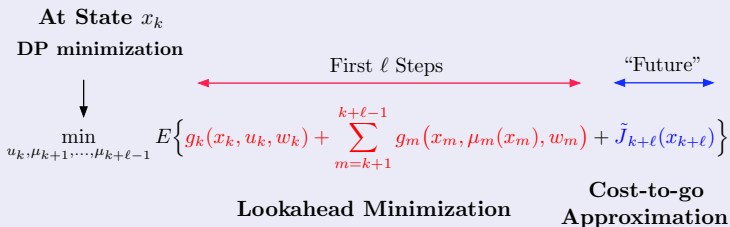
- Linear-quadratic problems are often used as a starting point for other lines of investigations and approximations:
 - ▶ **Problems with safety/state constraints** [Model Predictive Control (MPC)]
 - ▶ **Problems with control constraints** (MPC)
 - ▶ **Unknown or changing system parameters** (adaptive control)

Approximation in Value Space - The Three Approximations



Important variant: Use **multistep lookahead**

Multistep lookahead (performance - computational overhead tradeoff)



Constructing Approximations

Approximate Min

Discretization
Selective Minimization

At x_k

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

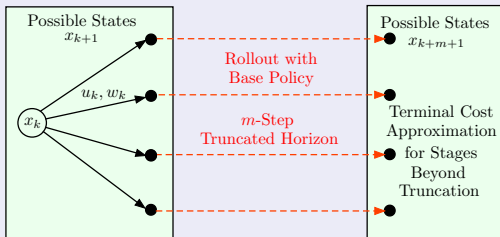
Approximate $E\{\cdot\}$
Adaptive simulation
Monte Carlo tree search

First Step

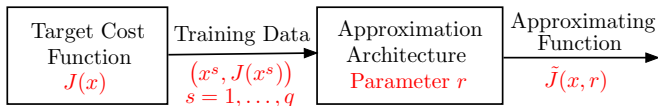
“Future”

Approximate Cost-to-Go \tilde{J}_{k+1}
Certainty equivalence
Problem approximation
Rollout, Model Predictive Control
Parametric approximation
Neural nets
Aggregation

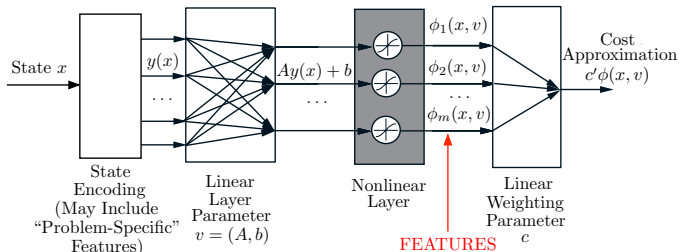
An example: Truncated rollout with base policy and terminal cost approximation (however obtained, e.g., off-line training)



Approximation Architectures/Neural Networks as Cost Function or Q-Factor Approximators



Training can be done with specialized optimization software such as gradient-like or other least-squares methods
Alternatives include Temporal Difference (TD) methods



A Single Layer Neural Network [Parameter Vector: $r = (A, b, c)$]

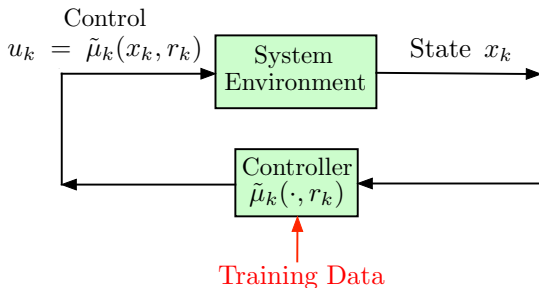
A Fifteen-Minute Break

All our lectures will have a 15-minute break, somewhere in the middle

Catch our breath and think about issues relating to the first half of the lecture.

A short discussion/questions/answers period will follow each break.

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 $\tilde{\mu}_k(x_k, r_k)$, $k = 0, \dots, N - 1$, of some form, where r_k is a parameter (e.g., a neural net)
- Methods used for optimization/off-line training: **Random search, policy gradient, machine learning/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

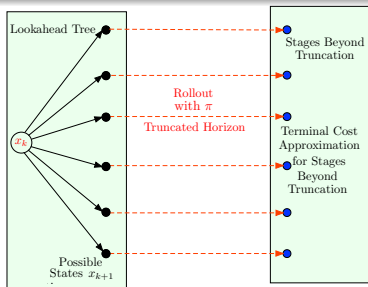
An Important Conceptual Difference Between Approximation in Value and in Policy Space

$$\text{At } x_k \rightarrow \min_{u_k} E \left\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(x_{k+1}) \right\}$$

← First Step ← "Future" →

Approximation in value space is primarily an "on-line play" method

with off-line training used optionally to construct cost function approximations for one-step or multistep lookahead



Approximation in policy space is primarily an "off-line training" method

which may be used optionally to provide a policy for on-line rollout

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

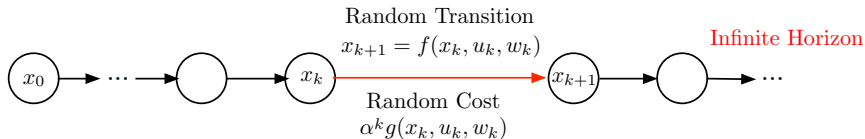
- Given functions \tilde{J}_{k+1} , how do we simplify the computation of $\tilde{\mu}_k$?
- Idea: **Use (off-line) approximation in policy space to "learn" $\tilde{\mu}_k$** : Approximate $\tilde{\mu}_k$ using a training set of a large number of sample pairs (x_k^s, u_k^s) , $s = 1, \dots, q$, where $u_k^s = \tilde{\mu}_k(x_k^s)$:

$$u_k^s \in \arg \min_{u \in U_k(x_k)} E \left\{ g_k(x_k^s, u, w_k) + \tilde{J}_{k+1}(f_k(x_k^s, u, w_k)) \right\} \quad (\text{off-line})$$

- Example**: Introduce a parametric family of randomized policies $\mu_k(x_k, r_k)$, $k = 0, \dots, N - 1$, of some form (e.g., a neural net), where r_k is a parameter. Then estimate the parameters r_k by **least squares fit**:

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

Infinite Horizon Problems



Infinite number of stages, and stationary system and cost

- System $x_{k+1} = f(x_k, u_k, w_k)$ with state, control, and random disturbance.
- Policies $\pi = \{\mu_0, \mu_1, \dots\}$ with $\mu_k(x) \in U(x)$ for all x and k .
- Cost of stage k : $\alpha^k g(x_k, \mu_k(x_k), w_k)$.
- $0 < \alpha \leq 1$ is the **discount factor**. If $\alpha < 1$ the problem is called **discounted**.
- Cost of a policy $\pi = \{\mu_0, \mu_1, \dots\}$: The limit as $N \rightarrow \infty$ of the N -stage costs

$$J_\pi(x_0) = \lim_{N \rightarrow \infty} E_{w_k} \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\}$$

- Optimal cost function $J^*(x_0) = \min_\pi J_\pi(x_0)$.
- Problems with $\alpha = 1$ typically include a special **cost-free termination state** t . The objective is to reach (or approach) t at minimum expected cost.

k -stages opt. cost \rightarrow Infinite horizon opt. cost as $k \rightarrow \infty$

- We have $J^*(x) = \lim_{k \rightarrow \infty} J_k(x)$, for all x , where for any k , $J_k(x)$ = k -stages optimal cost starting from x , and is generated by

$$J_k(x) = \min_{u \in U(x)} E_w \left\{ g(x, u, w) + \alpha J_{k-1}(f(x, u, w)) \right\}, \quad J_0(x) \equiv 0 \quad (\text{VI})$$

- Derivation using DP: Let $V_{N-k}(x)$ be the optimal cost-to-go starting at x with k stages to go,

$$V_{N-k}(x) = \min_{u \in U(x)} E_w \left\{ \alpha^{N-k} g(x, u, w) + V_{N-k+1}(f(x, u, w)) \right\}, \quad V_N(x) \equiv 0$$

- Define $J_k(x) = V_{N-k}(x) / \alpha^{N-k}$ to obtain Eq. (VI)

J^* satisfies Bellman's equation: Take the limit in Eq. (VI)

$$J^*(x) = \min_{u \in U(x)} E_w \left\{ g(x, u, w) + \alpha J^*(f(x, u, w)) \right\}, \quad \text{for all } x$$

Optimality condition: Let $\mu^*(x)$ attain the min in the Bellman equation for all x

The policy $\{\mu^*, \mu^*, \dots\}$ is optimal. (This type of policy is called **stationary**.)

Value iteration (VI): Generates finite horizon opt. cost function sequence $\{J_k\}$

$$J_k(x) = \min_{u \in U(x)} E_w \left\{ g(x, u, w) + \alpha J_{k-1}(f(x, u, w)) \right\}, \quad J_0 \text{ is "arbitrary" (?)}$$

Policy Iteration (PI): Generates sequences of policies $\{\mu^k\}$ and their cost functions $\{J_{\mu^k}\}$; μ^0 is "arbitrary"

The typical iteration starts with a policy μ and generates a new policy $\tilde{\mu}$ in two steps:

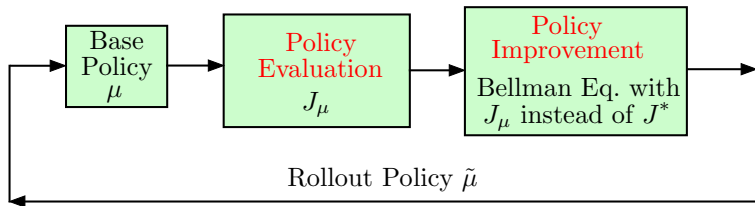
- **Policy evaluation step**, which computes J_μ the cost function of the (base) policy μ
- **Policy improvement step**, which computes the improved (rollout) policy $\tilde{\mu}$ using the one-step lookahead minimization

$$\tilde{\mu}(x) \in \arg \min_{u \in U(x)} E_w \left\{ g(x, u, w) + \alpha J_\mu(f(x, u, w)) \right\}$$

There are several options for policy evaluation to compute J_μ

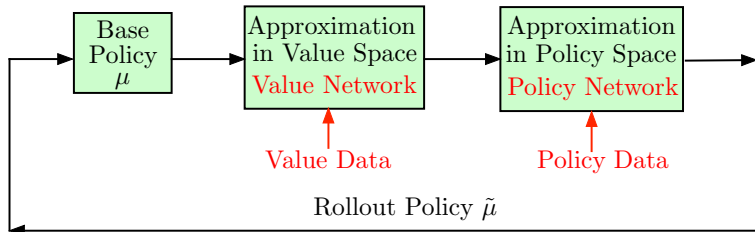
- Solve Bellman's equation for μ [$J_\mu(x) = E\{g(x, \mu(x), w) + \alpha J_\mu(f(x, \mu(x), w))\}$] by using VI or other method (it is linear in J_μ)
- Use simulation (**on-line Monte-Carlo, Temporal Difference (TD) methods**)

Exact and Approximate Policy Iteration



Important facts (to be discussed later):

- PI yields in the limit an optimal policy (?)
- PI is faster than VI; can be viewed as Newton's method for solving Bellman's Eq.
- PI can be implemented approximately, with a value and (perhaps) a policy network



Deterministic Linear Quadratic Problem - Infinite Horizon, Undiscounted

Linear system $x_{k+1} = ax_k + bu_k$; quadratic cost per stage $g(x, u) = qx^2 + ru^2$

Bellman equation: $J(x) = \min_u \{qx^2 + ru^2 + J(ax + bu)\}$

Take the limit as $N \rightarrow \infty$ in the N -step horizon results

- $J_k(x) = K_k x^2$ so $J^*(x) = K^* x^2$, where $K^* = \lim_{k \rightarrow \infty} K_k$
- The optimal policy is $\mu^*(x) = L^* x$, where L^* is obtained from the Bellman Eq.
- To characterize K^* and L^* , we plug $J(x) = Kx^2$ into the Bellman equation

$$Kx^2 = \min_u \{qx^2 + ru^2 + K(ax + bu)^2\} = \dots = F(K)x^2$$

where $F(K) = \frac{a^2 rK}{r + b^2 K} + q$ with the minimizing u being equal to $-\frac{abK}{r + b^2 K} x$

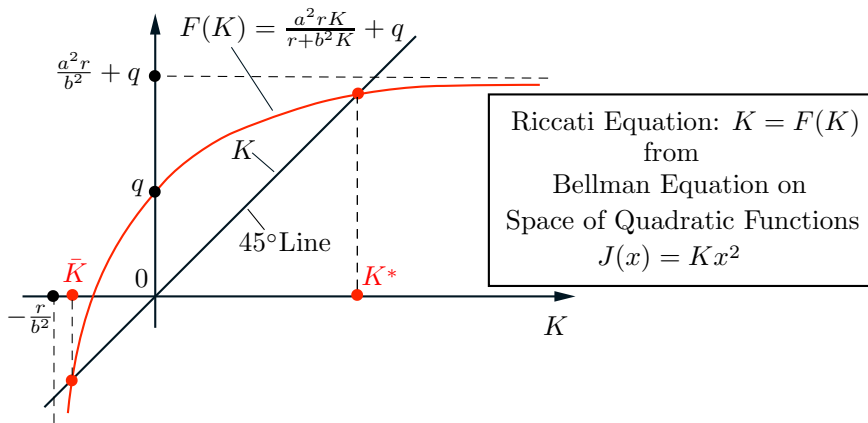
- Thus the Bellman equation is solved by $J^*(x) = K^* x^2$, with K^* being a solution of the Riccati equation

$$K^* = F(K^*) = \frac{a^2 rK^*}{r + b^2 K^*} + q$$

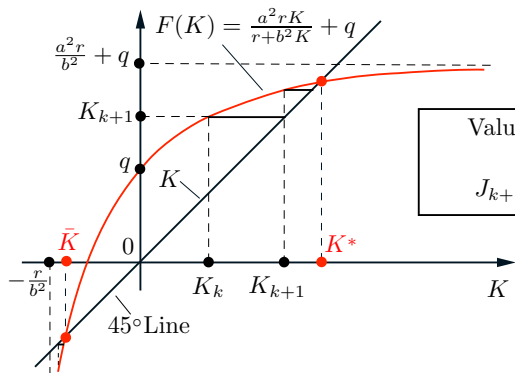
and the optimal policy is linear:

$$\mu^*(x) = L^* x \quad \text{with} \quad L^* = -\frac{abK^*}{r + b^2 K^*}$$

Graphical Solution of the Riccati Equation



Visualization of VI



Value Iteration: $K_{k+1} = F(K_k)$
from
 $J_{k+1}(x) = K_{k+1}x^2 = F(K_k)x^2$

About the Next Two Lectures

Newton step interpretations (for linear quadratic problems)

- Generally: **Approximation in value space is a Newton step for solving the Riccati equation, starting from the cost approximation**
- Special case: **Rollout is a Newton step starting from the base policy cost**
- **Policy Iteration consists of repeated Newton steps**

Problem formulations and reformulations

- How do we formulate DP models for practical problems?
- Problems with a **terminal state** (stochastic shortest path problems)
- Reformulation by **state augmentation** (delays, correlations, forecasts, etc)
- Problems with imperfect state observation (**POMDP**)
- Problems with **multiple agents**
- Model predictive control (**MPC**)
- Systems with unknown or changing parameters - **Adaptive control**

PLEASE READ AS MUCH OF THE TEXTBOOK AS YOU CAN
2ND HOMEWORK (DUE IN ONE WEEK): Exercise 1.1(b),(c) of the textbook