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

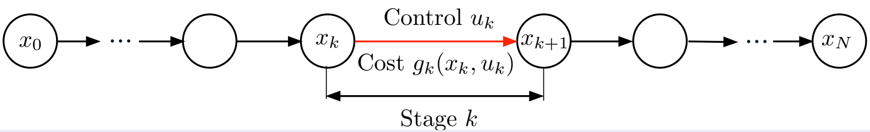
Arizona State University
Course CSE 691, Spring 2022

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

Dimitri P. Bertsekas
dbertsek@asu.edu

Lecture 2
Stochastic Finite and Infinite Horizon DP

Outline



- System

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

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

- Arbitrary state and control spaces
- 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 $\{u_0, \ldots, u_{N-1}\}$

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

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

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

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

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

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

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 $\{u_0^*, \ldots, u_{N-1}^*\}$ (on-line play)

Start with

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

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

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

Q-Factors for Deterministic Problems

An alternative (and equivalent) form of the DP algorithm

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

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

- 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

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

- 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

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

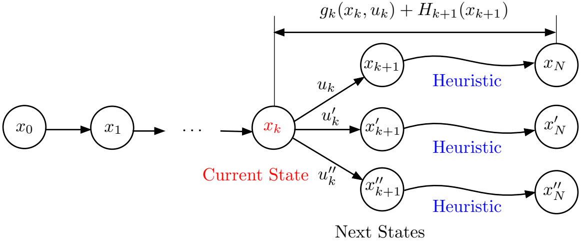
- Set $\tilde{x}_1 = f_0(x_0, \tilde{u}_0)$
- Sequentially, going forward, for $k = 1, 2, \ldots, N-1$, set

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

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
- On-line approximate optimization, e.g., solve on-line a shorter horizon problem by multistep lookahead minimization and simple terminal cost (often done in MPC)
- 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



Cost approximation by running a heuristic from states of interest

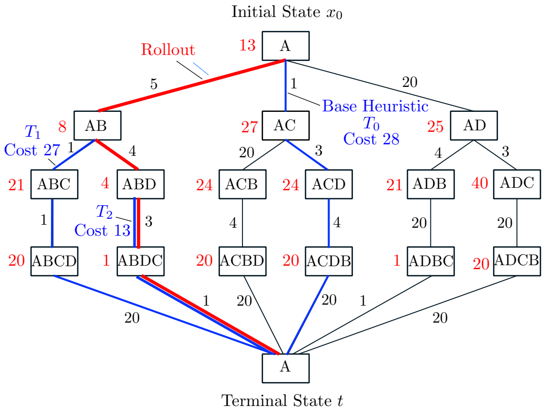
We generate a single system trajectory $\{x_0, x_1, \ldots, x_N\}$ by on-line play

- 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

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

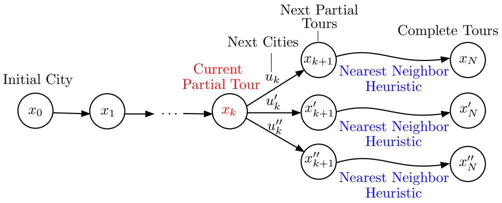
- We generate the next state $x_{k+1} = f_k(x_k, \tilde{u}_k)$ and repeat

Traveling Salesman Example

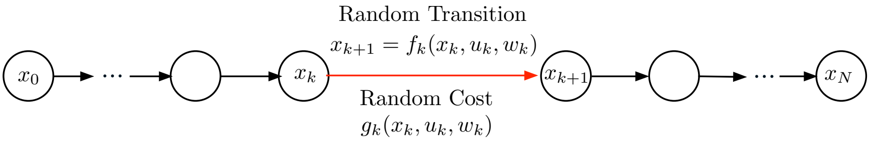


Matrix of Intercity Travel Costs

	5	1	20
20		1	4
1	20		1
20	4	3	



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 $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, 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 $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ the cost

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

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

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

- The optimal cost $J^*(x_0)$ is obtained at the last step: $J_0^*(x_0) = J^*(x_0)$.
- The optimal policy component μ_k^* can be constructed 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 $J_1^*, \ldots, J_{N-1}^*$

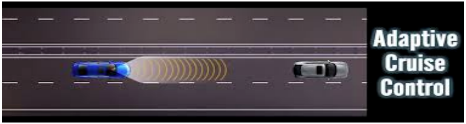
Sequentially, going forward, for $k = 0, 1, \ldots, N-1$, observe x_k and apply

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

Issues: Need to know J_{k+1}^*, compute expectation for each u_k, minimize over all u_k

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

A Very Favorable Case: Linear-Quadratic Problems



An example of a linear-quadratic problem

- Keep car velocity constant (like oversimplified cruise control): $x_{k+1} = x_k + bu_k + w_k$
- Here $x_k = v_k - \bar{v}$ is the deviation between the vehicle's velocity v_k at time k from desired level $\bar{v}$, and b is given
- u_k is unconstrained; w_k has 0-mean and variance σ^2
- 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
- Consider a more general problem where the system is $x_{k+1} = ax_k + bu_k + w_k$
- The DP algorithm starts with $J_N^*(x_N) = qx_N^2$, and generates J_k^* according to

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

- DP algorithm can be carried out in closed form to yield $J_k^*(x_k) = K_k x_k^2 + \text{const}, \ \mu_k^*(x_k) = L_k x_k$: K_k and L_k can be explicitly computed
- The solution 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)

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

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} E\{qx_{N-1}^2 + ru_{N-1}^2 + J_N^*(ax_{N-1} + bu_{N-1} + w_{N-1})\}$$

$$= \min_{u_{N-1}} E\{qx_{N-1}^2 + ru_{N-1}^2 + q(ax_{N-1} + bu_{N-1} + w_{N-1})^2\}$$

$$= \min_{u_{N-1}} \left[qx_{N-1}^2 + ru_{N-1}^2 + (ax_{N-1} + bu_{N-1})^2 + 2\underbrace{E\{w_{N-1}\}}_{=0}(ax_{N-1} + bu_{N-1}) + q\underbrace{E\{w_{N-1}^2\}}_{=\sigma^2}\right]$$

$$= qx_{N-1}^2 + \min_{u_{N-1}} \left[ru_{N-1}^2 + q(ax_{N-1} + bu_{N-1})^2\right] + q\sigma^2$$

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

Similarly, going backwards, we obtain for all k:

$$J_k^*(x_k) = P_kx_k^2 + \sigma^2\sum_{m=k}^{N-1} P_{m+1}, \quad \mu_k^*(x_k) = L_kx_k, \quad P_k = \frac{a^2rP_{k+1}}{r + b^2P_{k+1}} + q, \quad L_k = -\frac{abP_{k+1}}{r + b^2P_{k+1}}$$

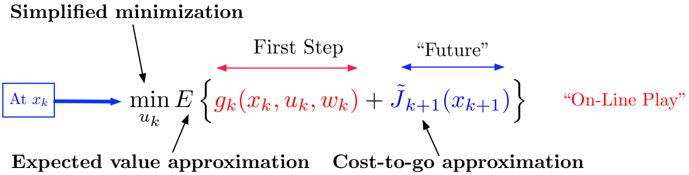
Linear-Quadratic Problems in General

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
- Generalization to multidimensional problems, nonzero mean disturbances, 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

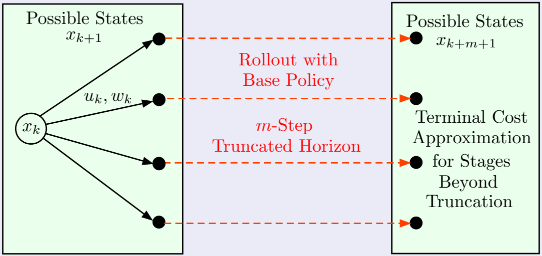
$$L_k E\{x_k \mid \text{measurements}\}$$

  where $E\{x_k \mid \text{measurements}\}$ is provided by an estimator (e.g., Kalman filter)
- Linear systems and quadratic cost are 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)



Important variants: Use multistep lookahead, replace $E\{\cdot\}$ by limited simulation (e.g., a "certainty equivalent" of w_k), multiagent rollout (for multicomponent control problems)

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

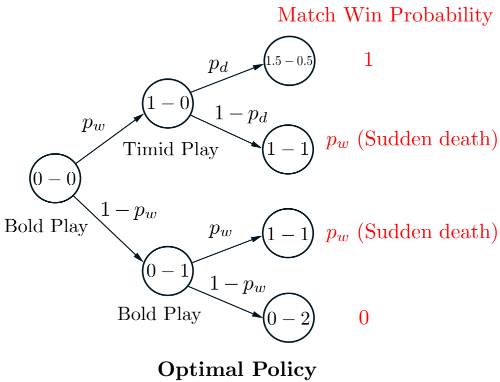


Let's Take a 15-min Working Break: Catch your Breath, Collect your Questions, and Consider the Following Challenge Puzzle

A chess match puzzle

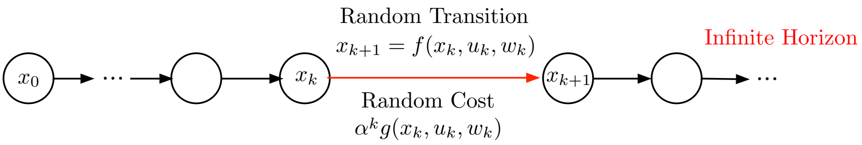
- A chess player plays against a chess computer program a two-game match.
- A win counts for 1, a draw counts for 1/2, and a loss counts for 0, for both player and computer.
- "Sudden death" games are played if the score is tied at 1-1 after the two games.
- The chess player can choose to play each game in one of two possible styles:
  - Bold play (wins with probability $p_w < 1/2$ and loses with probability $1 - p_w$) or
  - Timid play (draws with probability $p_d < 1$ and loses with probability $1 - p_d$).
- The style for the 2nd game is chosen after seeing the outcome of the 1st game.
- Note that the player plays worse than the computer (on the average), regardless of chosen style of play, and must play bold at least one game to have any chance to win.
- Speculate on the optimal policy of the player.
- Is it possible for the player to have a better than 50-50 chance to win the match, even though the computer is the better player?

Answer: Depending on p_w and p_d, Player's Win Prob. May be $> 1/2$



- The optimal policy: Play bold in the 1st game. Then play bold again if the 1st game is lost, and timid if the 1st game is won (see the full DP solution in DPB, DP textbook, Vol. I, Chapter 1; available from Google Books).
- Example: For $p_w = 0.45$ and $p_d = 0.9$, optimal style of play policy gives a match win probability of roughly 0.53 (a simple DP calculation that you can try).
- Intuition: The player can use feedback, while the computer cannot.

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 $\pi = \{\mu_0, \mu_1, \ldots\}$ with $\mu_k(x) \in U(x)$ for all x and k.
- Cost of stage k: $\alpha^k g\big(x_k, \mu_k(x_k), w_k\big)$.
- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$: The limit as $N \to \infty$ of the N-stage costs

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

- $0 < \alpha \leq 1$ is the discount factor. If $\alpha < 1$ the problem is called discounted.
- 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.

Intuition: N-stages opt. costs –> Infinite horizon opt. cost

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

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

- Define $J_k(x) = V_{N-k}(x)/\alpha^{N-k}$, i.e., reverse the time index and divide with α^{N-k}:

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

- $J_N(x)$ is equal to $V_0(x)$, the N-stages optimal cost starting from x
- So for any k, $J_k(x)$ = k-stages optimal cost starting from x. Intuitively:

$$J^*(x) = \lim_{k \to \infty} J_k(x), \qquad \text{for all } x$$

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

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

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

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

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

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

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:

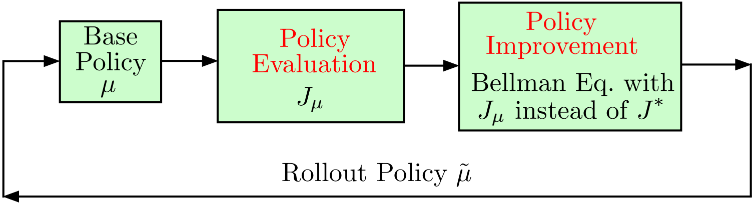
- Policy evaluation step, which computes the cost function J_μ (base) policy μ
- Policy improvement step, which computes the improved (rollout) policy $\tilde{\mu}$ using the one-step lookahead minimization

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

There are several options for policy evaluation to compute J_μ

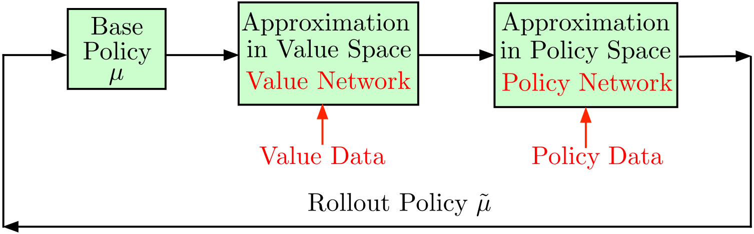
- Solve Bellman's equation for μ [$J_\mu(x) = E\{g(x,\mu(x),w) + \alpha J_\mu(f(x,\mu(x),w))\}$] by using VI or other method (it is linear in J_μ)
- Use simulation (on-line Monte-Carlo, Temporal Difference (TD) methods)

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



A More Abstract Notational View

Bellman's equation, VI, and PI can be written using Bellman operators

Recall Bellman's equation

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

It can be written as a fixed point equation: $J^*(x) = (TJ^*)(x)$, where T is the Bellman operator that transforms a function $J(\cdot)$ into a function $(TJ)(\cdot)$

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

Shorthand theory using Bellman operators:

- VI is the fixed point iteration $J_{k+1} = TJ_k$
- There is a Bellman operator T_μ for any policy μ and corresponding Bellman Eq. $J_\mu(x) = (T_\mu J_\mu)(x) = E\{g(x,\mu(x),w) + \alpha J_\mu(f(x,\mu(x),w))\}$
- PI is written compactly as $J_{\mu^k} = T_{\mu^k} J_{\mu^k}$ (policy evaluation) and $T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$ (policy improvement)

The abstract view is very useful for theoretical analysis, intuition, and visualization

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

Finite horizon results (quadratic optimal cost, linear optimal policy) suggest:

- $J^*(x) = K^* x^2$ where K^* is some positive scalar
- The optimal policy has the form $\mu^*(x) = L^* x$ where L^* is some scalar
- To characterize K^* and L^*, we plug $J(x) = Kx^2$ into the Bellman equation

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

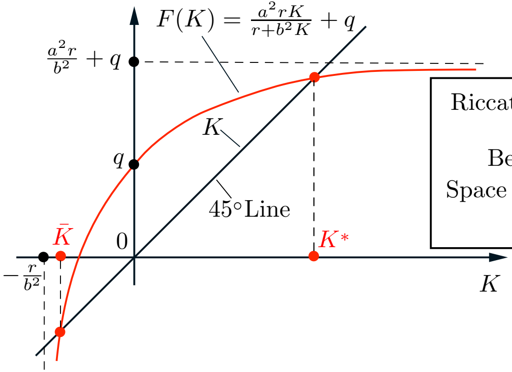
  where $F(K) = \frac{a^2 rK}{r+b^2K} + q$ with the minimizing u being equal to $-\frac{abK}{r+b^2K}x$
- Thus the Bellman equation is solved by $J^*(x) = K^* x^2$, with K^* being a solution of the Riccati equation

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

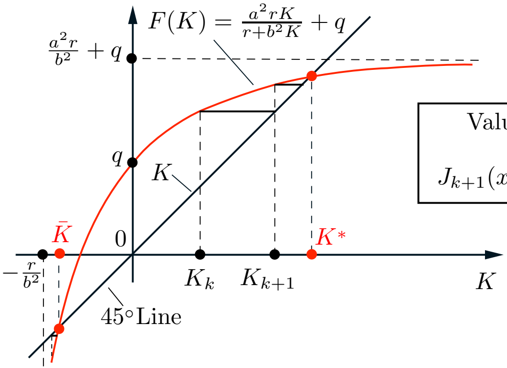
  and the optimal policy is linear:

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

Graphical Solution of Riccati Equation



Riccati Equation: $K = F(K)$
from
Bellman Equation on
Space of Quadratic Functions
$J(x) = Kx^2$



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

About the Next Lecture

Linear quadratic problems and Newton step interpretations

- Approximation in value space as a Newton step for solving the Riccati equation
- Rollout as a Newton step starting from the cost of the base policy
- Policy Iteration as repeated Newton steps

Problem formulations and reformulations

- How do we formulate DP models for practical problems?
- Problems involving a terminal state (stochastic shortest path problems)
- Problem reformulation by state augmentation (dealing with delays, correlations, forecasts, etc)
- Problems involving imperfect state observation (POMDP)
- Multiagent problems - Nonclassical information patterns
- Systems with unknown or changing parameters - Adaptive control

PLEASE READ SECTIONS 1.5 and 1.6 OF THE CLASS NOTES (AMAP)

1ST HOMEWORK (DUE IN ONE WEEK): Exercise 1.1 of the Class Notes