APPROXIMATE DYNAMIC PROGRAMMING

A SERIES OF LECTURES GIVEN AT

CEA - CADARACHE

FRANCE

SUMMER 2012

DIMITRI P. BERTSEKAS

These lecture slides are based on the book: "Dynamic Programming and Optimal Control: Approximate Dynamic Programming," Athena Scientific, 2012; see

http://www.athenasc.com/dpbook.html

For a fuller set of slides, see

http://web.mit.edu/dimitrib/www/publ.html

*Athena is MIT's UNIX-based computing environment. OCW does not provide access to it.

APPROXIMATE DYNAMIC PROGRAMMING

BRIEF OUTLINE I

- **Our subject:**
  - Large-scale DP based on approximations and in part on simulation.
  - This has been a research area of great interest for the last 20 years known under various names (e.g., reinforcement learning, neuro-dynamic programming)
  - Emerged through an enormously fruitful cross-fertilization of ideas from artificial intelligence and optimization/control theory
  - Deals with control of dynamic systems under uncertainty, but applies more broadly (e.g., discrete deterministic optimization)
  - A vast range of applications in control theory, operations research, artificial intelligence, and beyond ...
  - The subject is broad with rich variety of theory/math, algorithms, and applications. Our focus will be mostly on algorithms ... less on theory and modeling

APPROXIMATE DYNAMIC PROGRAMMING

BRIEF OUTLINE II

- **Our aim:**
  - A state-of-the-art account of some of the major topics at a graduate level
  - Show how the use of approximation and simulation can address the dual curses of DP: dimensionality and modeling
- **Our 7-lecture plan:**
  - Two lectures on exact DP with emphasis on infinite horizon problems and issues of large-scale computational methods
  - One lecture on general issues of approximation and simulation for large-scale problems
  - One lecture on approximate policy iteration based on temporal differences (TD)/projected equations/Galerkin approximation
  - One lecture on aggregation methods
  - One lecture on stochastic approximation, Q-learning, and other methods
  - One lecture on Monte Carlo methods for solving general problems involving linear equations and inequalities

APPROXIMATE DYNAMIC PROGRAMMING

LECTURE 1

LECTURE OUTLINE

- Introduction to DP and approximate DP
- Finite horizon problems
- The DP algorithm for finite horizon problems
- Infinite horizon problems
- Basic theory of discounted infinite horizon problems

BASIC STRUCTURE OF STOCHASTIC DP

- Discrete-time system

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

  - k: Discrete time
  - x_k: State; summarizes past information that is relevant for future optimization
  - u_k: Control; decision to be selected at time k from a given set
  - w_k: Random parameter (also called "disturbance" or "noise" depending on the context)
  - N: Horizon or number of times control is applied

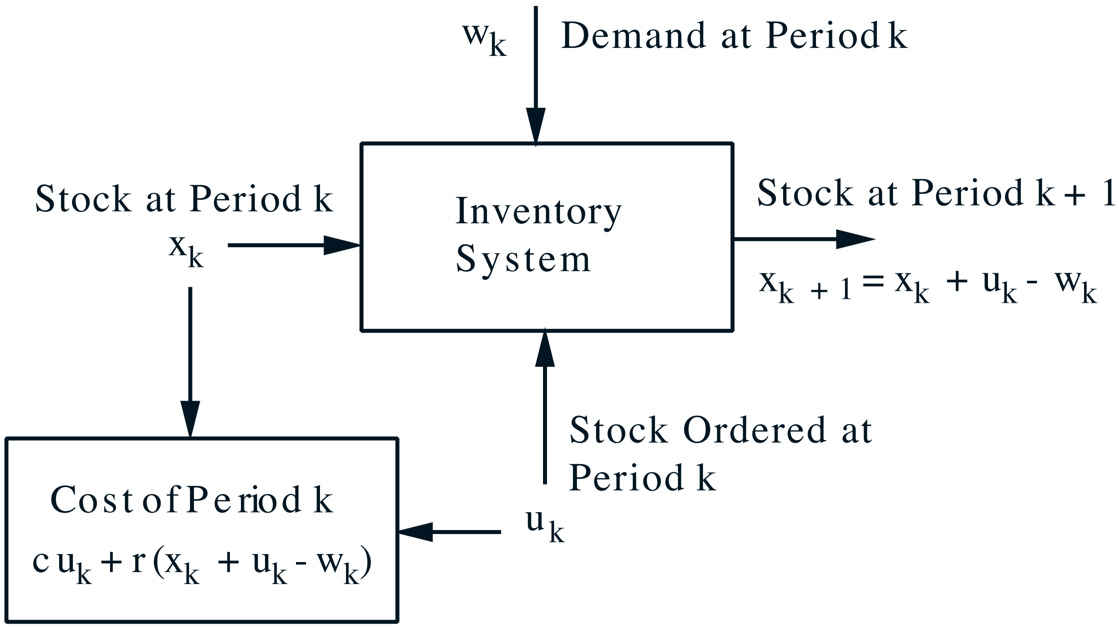
- Cost function that is additive over time

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

- Alternative system description: $P(x_{k+1} \mid x_k, u_k)$

$$x_{k+1} = w_k \quad \text{with } P(w_k \mid x_k, u_k) = P(x_{k+1} \mid x_k, u_k)$$

INVENTORY CONTROL EXAMPLE



- Discrete-time system

$$x_{k+1} = f_k(x_k, u_k, w_k) = x_k + u_k - w_k$$

- Cost function that is additive over time

$$E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\}$$
$$= E\left\{ \sum_{k=0}^{N-1} \big(cu_k + r(x_k + u_k - w_k)\big) \right\}$$

ADDITIONAL ASSUMPTIONS

- Optimization over policies: These are rules/functions

$$u_k = \mu_k(x_k), \qquad k = 0, \ldots, N-1$$

that map states to controls (closed-loop optimization, use of feedback)

- The set of values that the control u_k can take depend at most on x_k and not on prior x or u

- Probability distribution of w_k does not depend on past values $w_{k-1}, \ldots, w_0$, but may depend on x_k and u_k
  - Otherwise past values of w or x would be useful for future optimization

GENERIC FINITE-HORIZON PROBLEM

- System $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \ldots, N-1$
- Control contraints $u_k \in U_k(x_k)$
- Probability distribution $P_k(\cdot \mid x_k, u_k)$ of w_k
- Policies $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all x_k
- Expected cost of π starting at x_0 is

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

- Optimal cost function

$$J^*(x_0) = \min_\pi J_\pi(x_0)$$

- Optimal policy π^* satisfies

$$J_{\pi^*}(x_0) = J^*(x_0)$$

When produced by DP, π^* is independent of x_0.

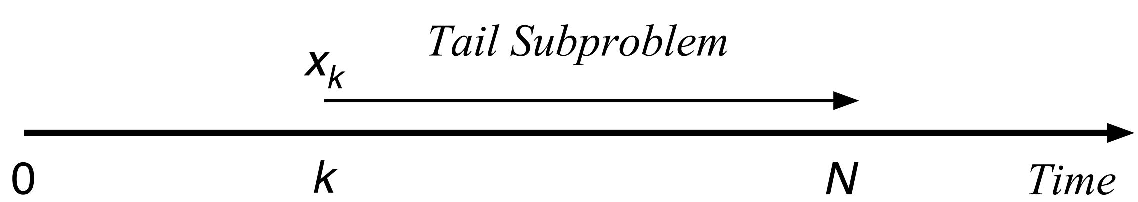
PRINCIPLE OF OPTIMALITY

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be optimal policy

- Consider the "tail subproblem" whereby we are at x_k at time k and wish to minimize the "cost-to-go" from time k to time N

$$E\left\{g_N(x_N) + \sum_{\ell=k}^{N-1} g_\ell\big(x_\ell, \mu_\ell(x_\ell), w_\ell\big)\right\}$$

and the "tail policy" $\{\mu_k^*, \mu_{k+1}^*, \ldots, \mu_{N-1}^*\}$



- Principle of optimality: The tail policy is optimal for the tail subproblem (optimization of the future does not depend on what we did in the past)

- DP solves ALL the tail subroblems

- At the generic step, it solves ALL tail subproblems of a given time length, using the solution of the tail subproblems of shorter time length

DP ALGORITHM

- $J_k(x_k)$: opt. cost of tail problem starting at x_k
- Start with

$$J_N(x_N) = g_N(x_N),$$

and go backwards using

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \mathop{E}_{w_k} \Big\{ g_k(x_k, u_k, w_k) + J_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\}, \quad k = 0, 1, \ldots, N-1$$

i.e., to solve tail subproblem at time k minimize

Sum of kth-stage cost + Opt. cost of next tail problem

starting from next state at time $k+1$

- Then $J_0(x_0)$, generated at the last step, is equal to the optimal cost $J^*(x_0)$. Also, the policy

$$\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$$

where $\mu_k^*(x_k)$ minimizes in the right side above for each x_k and k, is optimal

- Proof by induction

PRACTICAL DIFFICULTIES OF DP

- The curse of dimensionality
  - Exponential growth of the computational and storage requirements as the number of state variables and control variables increases
  - Quick explosion of the number of states in combinatorial problems
  - Intractability of imperfect state information problems
- The curse of modeling
  - Sometimes a simulator of the system is easier to construct than a model
- There may be real-time solution constraints
  - A family of problems may be addressed. The data of the problem to be solved is given with little advance notice
  - The problem data may change as the system is controlled – need for on-line replanning
- All of the above are motivations for approximation and simulation

COST-TO-GO FUNCTION APPROXIMATION

- Use a policy computed from the DP equation where the optimal cost-to-go function J_{k+1} is replaced by an approximation $\tilde{J}_{k+1}$.
- Apply $\overline{\mu}_k(x_k)$, which attains the minimum in

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

- Some approaches:
  (a) Problem Approximation: Use $\tilde{J}_k$ derived from a related but simpler problem
  (b) Parametric Cost-to-Go Approximation: Use as $\tilde{J}_k$ a function of a suitable parametric form, whose parameters are tuned by some heuristic or systematic scheme (we will mostly focus on this)
    – This is a major portion of Reinforcement Learning/Neuro-Dynamic Programming
  (c) Rollout Approach: Use as $\tilde{J}_k$ the cost of some suboptimal policy, which is calculated either analytically or by simulation

ROLLOUT ALGORITHMS

- At each k and state x_k, use the control $\overline{\mu}_k(x_k)$ that minimizes in

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

where $\tilde{J}_{k+1}$ is the cost-to-go of some heuristic policy (called the base policy).

- Cost improvement property: The rollout algorithm achieves no worse (and usually much better) cost than the base policy starting from the same state.

- Main difficulty: Calculating $\tilde{J}_{k+1}(x)$ may be computationally intensive if the cost-to-go of the base policy cannot be analytically calculated.
  - May involve Monte Carlo simulation if the problem is stochastic.
  - Things improve in the deterministic case.
  - Connection w/ Model Predictive Control (MPC)

INFINITE HORIZON PROBLEMS

- Same as the basic problem, but:
  - The number of stages is infinite.
  - The system is stationary.
- Total cost problems: Minimize

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

  - Discounted problems ($\alpha < 1$, bounded g)
  - Stochastic shortest path problems ($\alpha = 1$, finite-state system with a termination state) - we will discuss sparringly
  - Discounted and undiscounted problems with unbounded cost per stage - we will not cover
- Average cost problems - we will not cover
- Infinite horizon characteristics:
  - Challenging analysis, elegance of solutions and algorithms
  - Stationary policies $\pi = \{\mu, \mu, \ldots\}$ and stationary forms of DP play a special role

DISCOUNTED PROBLEMS/BOUNDED COST

- Stationary system

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots$$

- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

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

with $\alpha < 1$, and g is bounded [for some M, we have $|g(x,u,w)| \le M$ for all (x,u,w)]

- Boundedness of g guarantees that all costs are well-defined and bounded: $\big|J_\pi(x)\big| \le \frac{M}{1-\alpha}$

- All spaces are arbitrary - only boundedness of g is important (there are math fine points, e.g. measurability, but they don't matter in practice)

- Important special case: All underlying spaces finite; a (finite spaces) Markovian Decision Problem or MDP

- All algorithms essentially work with an MDP that approximates the original problem

SHORTHAND NOTATION FOR DP MAPPINGS

- For any function J of x

$$(TJ)(x) = \min_{u \in U(x)} \mathop{E}_{w} \Big\{ g(x, u, w) + \alpha J\big(f(x, u, w)\big) \Big\}, \ \forall\, x$$

- TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost function αJ.
- T operates on bounded functions of x to produce other bounded functions of x
- For any stationary policy μ

$$(T_\mu J)(x) = \mathop{E}_{w} \Big\{ g\big(x, \mu(x), w\big) + \alpha J\big(f(x, \mu(x), w)\big) \Big\}, \ \forall\, x$$

- The critical structure of the problem is captured in T and T_μ
- The entire theory of discounted problems can be developed in shorthand using T and T_μ
- This is true for many other DP problems

FINITE-HORIZON COST EXPRESSIONS

• Consider an N-stage policy $\pi_0^N = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ with a terminal cost J:

$$\begin{aligned} J_{\pi_0^N}(x_0) &= E\left\{\alpha^N J(x_k) + \sum_{\ell=0}^{N-1} \alpha^\ell g\big(x_\ell, \mu_\ell(x_\ell), w_\ell\big)\right\} \\ &= E\left\{g\big(x_0, \mu_0(x_0), w_0\big) + \alpha J_{\pi_1^N}(x_1)\right\} \\ &= (T_{\mu_0} J_{\pi_1^N})(x_0) \end{aligned}$$

where $\pi_1^N = \{\mu_1, \mu_2, \ldots, \mu_{N-1}\}$

• By induction we have

$$J_{\pi_0^N}(x) = (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_{N-1}} J)(x), \qquad \forall\ x$$

• For a stationary policy μ the N-stage cost function (with terminal cost J) is

$$J_{\pi_0^N} = T_\mu^N J$$

where T_μ^N is the N-fold composition of T_μ

• Similarly the optimal N-stage cost function (with terminal cost J) is $T^N J$

• $T^N J = T(T^{N-1} J)$ is just the DP algorithm

“SHORTHAND” THEORY – A SUMMARY

- Infinite horizon cost function expressions [with $J_0(x) \equiv 0$]

$$J_\pi(x) = \lim_{N\to\infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_N} J_0)(x), \quad J_\mu(x) = \lim_{N\to\infty} (T_\mu^N J_0)(x)$$

- Bellman's equation: $J^* = TJ^*, \ \ J_\mu = T_\mu J_\mu$
- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

- Value iteration: For any (bounded) J

$$J^*(x) = \lim_{k\to\infty} (T^k J)(x), \qquad \forall\ x$$

- Policy iteration: Given μ^k,
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

  - Policy improvement : Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$$

TWO KEY PROPERTIES

- Monotonicity property: For any J and J' such that $J(x) \le J'(x)$ for all x, and any μ

$$(TJ)(x) \le (TJ')(x), \qquad \forall\, x,$$

$$(T_\mu J)(x) \le (T_\mu J')(x), \qquad \forall\, x.$$

- Constant Shift property: For any J, any scalar r, and any μ

$$\big(T(J + re)\big)(x) = (TJ)(x) + \alpha r, \qquad \forall\, x,$$

$$\big(T_\mu(J + re)\big)(x) = (T_\mu J)(x) + \alpha r, \qquad \forall\, x,$$

where e is the unit function $[e(x) \equiv 1]$.

- Monotonicity is present in all DP models (undiscounted, etc)
- Constant shift is special to discounted models
- Discounted problems have another property of major importance: T and T_μ are contraction mappings (we will show this later)

CONVERGENCE OF VALUE ITERATION

- If $J_0 \equiv 0$,

$$J^*(x) = \lim_{k\to\infty} (T^k J_0)(x), \qquad \text{for all } x$$

Proof: For any initial state x_0, and policy $\pi = \{\mu_0, \mu_1, \ldots\}$,

$$\begin{aligned} J_\pi(x_0) &= E\left\{\sum_{\ell=0}^{\infty} \alpha^\ell g\big(x_\ell, \mu_\ell(x_\ell), w_\ell\big)\right\} \\ &= E\left\{\sum_{\ell=0}^{k-1} \alpha^\ell g\big(x_\ell, \mu_\ell(x_\ell), w_\ell\big)\right\} \\ &\qquad\qquad + E\left\{\sum_{\ell=k}^{\infty} \alpha^\ell g\big(x_\ell, \mu_\ell(x_\ell), w_\ell\big)\right\} \end{aligned}$$

The tail portion satisfies

$$\left| E\left\{\sum_{\ell=k}^{\infty} \alpha^\ell g\big(x_\ell, \mu_\ell(x_\ell), w_\ell\big)\right\}\right| \le \frac{\alpha^k M}{1-\alpha},$$

where $M \ge |g(x,u,w)|$. Take the min over π of both sides. **Q.E.D.**

BELLMAN'S EQUATION

- The optimal cost function J^* satisfies Bellman's Eq., i.e. $J^* = TJ^*$.

Proof: For all x and k,

$$J^*(x) - \frac{\alpha^k M}{1-\alpha} \leq (T^k J_0)(x) \leq J^*(x) + \frac{\alpha^k M}{1-\alpha},$$

where $J_0(x) \equiv 0$ and $M \geq |g(x,u,w)|$. Applying T to this relation, and using Monotonicity and Constant Shift,

$$\begin{aligned}(TJ^*)(x) - \frac{\alpha^{k+1} M}{1-\alpha} &\leq (T^{k+1} J_0)(x) \\ &\leq (TJ^*)(x) + \frac{\alpha^{k+1} M}{1-\alpha}\end{aligned}$$

Taking the limit as $k \to \infty$ and using the fact

$$\lim_{k\to\infty} (T^{k+1} J_0)(x) = J^*(x)$$

we obtain $J^* = TJ^*$. **Q.E.D.**

THE CONTRACTION PROPERTY

- Contraction property: For any bounded functions J and J', and any μ,

$$\max_x \big|(TJ)(x) - (TJ')(x)\big| \leq \alpha \max_x \big|J(x) - J'(x)\big|,$$

$$\max_x \big|(T_\mu J)(x) - (T_\mu J')(x)\big| \leq \alpha \max_x \big|J(x) - J'(x)\big|.$$

Proof: Denote $c = \max_{x \in S} \big|J(x) - J'(x)\big|$. Then

$$J(x) - c \leq J'(x) \leq J(x) + c, \qquad \forall\ x$$

Apply T to both sides, and use the Monotonicity and Constant Shift properties:

$$(TJ)(x) - \alpha c \leq (TJ')(x) \leq (TJ)(x) + \alpha c, \qquad \forall\ x$$

Hence

$$\big|(TJ)(x) - (TJ')(x)\big| \leq \alpha c, \qquad \forall\ x.$$

Q.E.D.

NEC. AND SUFFICIENT OPT. CONDITION

- A stationary policy μ is optimal if and only if $\mu(x)$ attains the minimum in Bellman's equation for each x; i.e.,

$$TJ^* = T_\mu J^*.$$

Proof: If $TJ^* = T_\mu J^*$, then using Bellman's equation ($J^* = TJ^*$), we have

$$J^* = T_\mu J^*,$$

so by uniqueness of the fixed point of T_μ, we obtain $J^* = J_\mu$; i.e., μ is optimal.

- Conversely, if the stationary policy μ is optimal, we have $J^* = J_\mu$, so

$$J^* = T_\mu J^*.$$

Combining this with Bellman's Eq. ($J^* = TJ^*$), we obtain $TJ^* = T_\mu J^*$. **Q.E.D.**

APPROXIMATE DYNAMIC PROGRAMMING

LECTURE 2

LECTURE OUTLINE

- Review of discounted problem theory
- Review of shorthand notation
- Algorithms for discounted DP
- Value iteration
- Policy iteration
- Optimistic policy iteration
- Q-factors and Q-learning
- A more abstract view of DP
- Extensions of discounted DP
- Value and policy iteration
- Asynchronous algorithms

DISCOUNTED PROBLEMS/BOUNDED COST

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots$$

- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

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

with $\alpha < 1$, and for some M, we have $|g(x, u, w)| \leq M$ for all (x, u, w)

- Shorthand notation for DP mappings (operate on functions of state to produce other functions)

$$(TJ)(x) = \min_{u \in U(x)} \mathop{E}_{w} \left\{ g(x, u, w) + \alpha J\big(f(x, u, w)\big) \right\}, \ \forall\, x$$

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ.

- For any stationary policy μ

$$(T_\mu J)(x) = \mathop{E}_{w} \left\{ g\big(x, \mu(x), w\big) + \alpha J\big(f(x, \mu(x), w)\big) \right\}, \ \forall\, x$$

"SHORTHAND" THEORY – A SUMMARY

- Cost function expressions [with $J_0(x) \equiv 0$]

$$J_\pi(x) = \lim_{k\to\infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0)(x), \quad J_\mu(x) = \lim_{k\to\infty} (T_\mu^k J_0)(x)$$

- Bellman's equation: $J^* = TJ^*, \quad J_\mu = T_\mu J_\mu$ or

$$J^*(x) = \min_{u\in U(x)} \mathop{E}_{w} \left\{ g(x,u,w) + \alpha J^*\big(f(x,u,w)\big) \right\}, \; \forall\, x$$

$$J_\mu(x) = \mathop{E}_{w} \left\{ g\big(x,\mu(x),w\big) + \alpha J_\mu\big(f(x,\mu(x),w)\big) \right\}, \; \forall\, x$$

- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

i.e.,

$$\mu(x) \in \arg\min_{u\in U(x)} \mathop{E}_{w} \left\{ g(x,u,w) + \alpha J^*\big(f(x,u,w)\big) \right\}, \; \forall\, x$$

- Value iteration: For any (bounded) J

$$J^*(x) = \lim_{k\to\infty} (T^k J)(x), \qquad \forall\, x$$

MAJOR PROPERTIES

- Monotonicity property: For any functions J and J' on the state space X such that $J(x) \le J'(x)$ for all $x \in X$, and any μ

$$(TJ)(x) \le (TJ')(x), \quad (T_\mu J)(x) \le (T_\mu J')(x), \ \forall\, x \in X.$$

- Contraction property: For any bounded functions J and J', and any μ,

$$\max_x \big|(TJ)(x) - (TJ')(x)\big| \le \alpha \max_x \big|J(x) - J'(x)\big|,$$

$$\max_x \big|(T_\mu J)(x) - (T_\mu J')(x)\big| \le \alpha \max_x \big|J(x) - J'(x)\big|.$$

- Compact Contraction Notation:

$$\|TJ - TJ'\| \le \alpha \|J - J'\|, \ \ \|T_\mu J - T_\mu J'\| \le \alpha \|J - J'\|,$$

where for any bounded function J, we denote by $\|J\|$ the sup-norm

$$\|J\| = \max_{x \in X} \big|J(x)\big|.$$

THE TWO MAIN ALGORITHMS: VI AND PI

- Value iteration: For any (bounded) J

$$J^*(x) = \lim_{k\to\infty} (T^k J)(x), \qquad \forall\ x$$

- Policy iteration: Given μ^k
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k}(x) = \mathop{E}_{w}\left\{g\big(x, \mu(x), w\big) + \alpha J_{\mu^k}\big(f(x, \mu^k(x), w)\big)\right\},\ \forall\ x$$

or $J_{\mu^k} = T_{\mu^k} J_{\mu^k}$

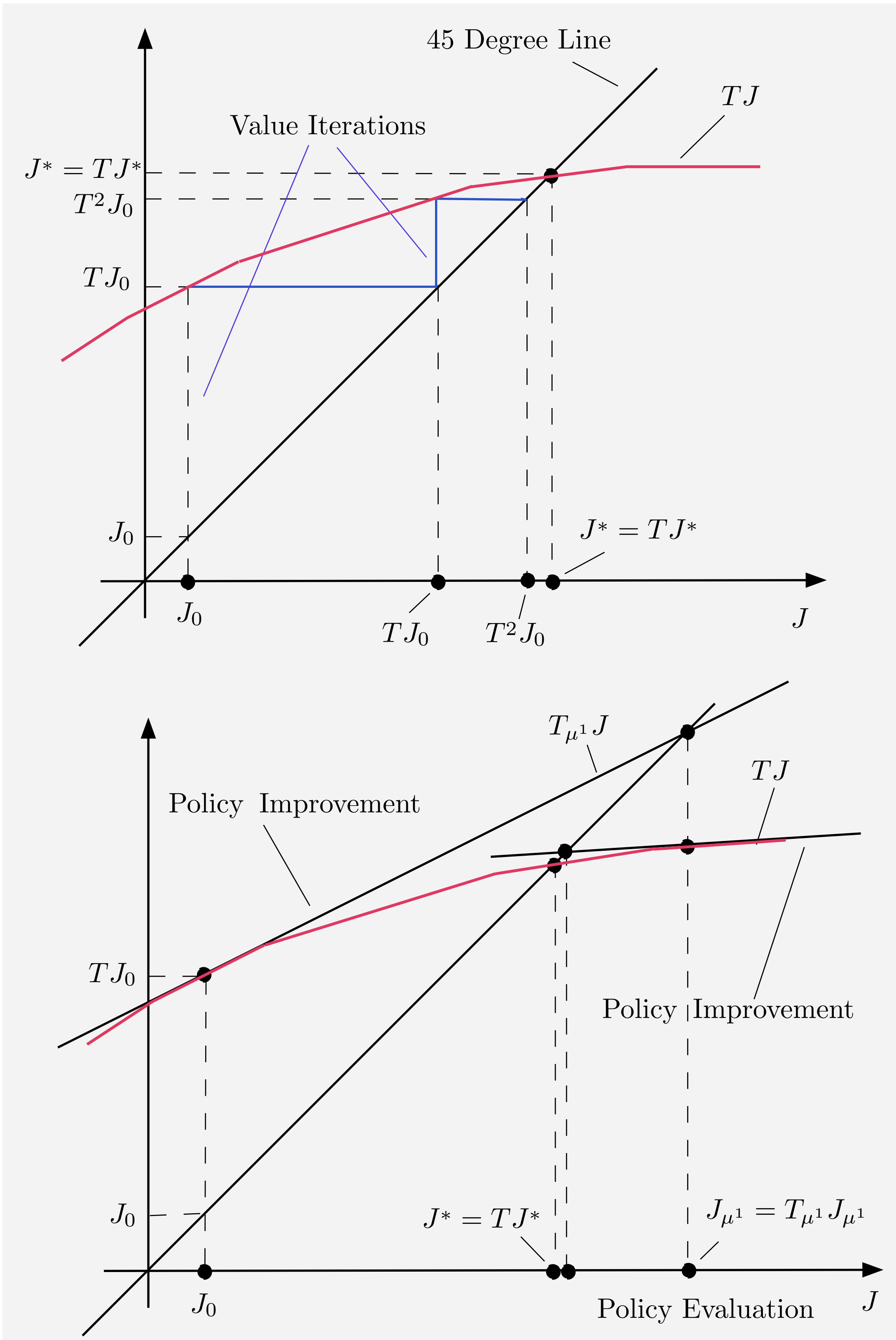
  - Policy improvement: Let μ^{k+1} be such that

$$\mu^{k+1}(x) \in \arg\min_{u\in U(x)} \mathop{E}_{w}\left\{g(x, u, w) + \alpha J_{\mu^k}\big(f(x, u, w)\big)\right\},\ \forall\ x$$

or $T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$

- For finite state space policy evaluation is equivalent to solving a linear system of equations
- Dimension of the system is equal to the number of states.
- For large problems, exact PI is out of the question (even though it terminates finitely)

INTERPRETATION OF VI AND PI



JUSTIFICATION OF POLICY ITERATION

- We can show that $J_{\mu^{k+1}} \leq J_{\mu^k}$ for all k
- Proof: For given k, we have

$$T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k} \leq T_{\mu^k} J_{\mu^k} = J_{\mu^k}$$

Using the monotonicity property of DP,

$$J_{\mu^k} \geq T_{\mu^{k+1}} J_{\mu^k} \geq T^2_{\mu^{k+1}} J_{\mu^k} \geq \cdots \geq \lim_{N\to\infty} T^N_{\mu^{k+1}} J_{\mu^k}$$

- Since

$$\lim_{N\to\infty} T^N_{\mu^{k+1}} J_{\mu^k} = J_{\mu^{k+1}}$$

we have $J_{\mu^k} \geq J_{\mu^{k+1}}$.

- If $J_{\mu^k} = J_{\mu^{k+1}}$, then J_{μ^k} solves Bellman's equation and is therefore equal to J^*
- So at iteration k either the algorithm generates a strictly improved policy or it finds an optimal policy
- For a finite spaces MDP, there are finitely many stationary policies, so the algorithm terminates with an optimal policy

APPROXIMATE PI

- Suppose that the policy evaluation is approximate,

$$\|J_k - J_{\mu^k}\| \leq \delta, \qquad k = 0, 1, \ldots$$

and policy improvement is approximate,

$$\|T_{\mu^{k+1}} J_k - T J_k\| \leq \epsilon, \qquad k = 0, 1, \ldots$$

where δ and ϵ are some positive scalars.

- Error Bound I: The sequence $\{\mu^k\}$ generated by approximate policy iteration satisfies

$$\limsup_{k \to \infty} \|J_{\mu^k} - J^*\| \leq \frac{\epsilon + 2\alpha\delta}{(1-\alpha)^2}$$

- Typical practical behavior: The method makes steady progress up to a point and then the iterates J_{μ^k} oscillate within a neighborhood of J^*.

- Error Bound II: If in addition the sequence $\{\mu^k\}$ terminates at $\overline{\mu}$,

$$\|J_{\overline{\mu}} - J^*\| \leq \frac{\epsilon + 2\alpha\delta}{1-\alpha}$$

OPTIMISTIC POLICY ITERATION

- Optimistic PI (more efficient): This is PI, where policy evaluation is done approximately, with a finite number of VI
- So we approximate the policy evaluation

$$J_\mu \approx T_\mu^m J$$

for some number $m \in [1, \infty)$

- Shorthand definition: For some integers m_k

$$T_{\mu^k} J_k = T J_k, \qquad J_{k+1} = T_{\mu^k}^{m_k} J_k, \qquad k = 0, 1, \ldots$$

- If $m_k \equiv 1$ it becomes VI
- If $m_k = \infty$ it becomes PI
- Can be shown to converge (in an infinite number of iterations)

Q-LEARNING I

- We can write Bellman's equation as

$$J^*(x) = \min_{u \in U(x)} Q^*(x,u), \qquad \forall\ x,$$

where Q^* is the unique solution of

$$Q^*(x,u) = E\left\{ g(x,u,w) + \alpha \min_{v \in U(\overline{x})} Q^*(\overline{x},v) \right\}$$

with $\overline{x} = f(x,u,w)$

- $Q^*(x,u)$ is called the optimal Q-factor of (x,u)
- We can equivalently write the VI method as

$$J_{k+1}(x) = \min_{u \in U(x)} Q_{k+1}(x,u), \qquad \forall\ x,$$

where Q_{k+1} is generated by

$$Q_{k+1}(x,u) = E\left\{ g(x,u,w) + \alpha \min_{v \in U(\overline{x})} Q_k(\overline{x},v) \right\}$$

with $\overline{x} = f(x,u,w)$

Q-LEARNING II

- Q-factors are no different than costs
- They satisfy a Bellman equation $Q = FQ$ where

$$(FQ)(x,u) = E\left\{ g(x,u,w) + \alpha \min_{v \in U(\overline{x})} Q(x,v) \right\}$$

where $\overline{x} = f(x,u,w)$

- VI and PI for Q-factors are mathematically equivalent to VI and PI for costs
- They require equal amount of computation ... they just need more storage
- Having optimal Q-factors is convenient when implementing an optimal policy on-line by

$$\mu^*(x) = \min_{u \in U(x)} Q^*(x,u)$$

- Once $Q^*(x,u)$ are known, the model [g and $E\{\cdot\}$] is not needed. Model-free operation.
- Later we will see how stochastic/sampling methods can be used to calculate (approximations of) $Q^*(x,u)$ using a simulator of the system (no model needed)

A MORE GENERAL/ABSTRACT VIEW

- Let Y be a real vector space with a norm $\|\cdot\|$
- A function $F : Y \mapsto Y$ is said to be a contraction mapping if for some $\rho \in (0,1)$, we have

$$\|Fy - Fz\| \leq \rho\|y - z\|, \qquad \text{for all } y, z \in Y.$$

ρ is called the modulus of contraction of F.

- Important example: Let X be a set (e.g., state space in DP), $v : X \mapsto \Re$ be a positive-valued function. Let $B(X)$ be the set of all functions $J : X \mapsto \Re$ such that $J(x)/v(x)$ is bounded over x.
- We define a norm on $B(X)$, called the weighted sup-norm, by

$$\|J\| = \max_{x \in X} \frac{|J(x)|}{v(x)}.$$

- Important special case: The discounted problem mappings T and T_μ [for $v(x) \equiv 1$, $\rho = \alpha$].

A DP-LIKE CONTRACTION MAPPING

- Let $X = \{1, 2, \ldots\}$, and let $F : B(X) \mapsto B(X)$ be a linear mapping of the form

$$(FJ)(i) = b_i + \sum_{j \in X} a_{ij}\, J(j), \qquad \forall\, i = 1, 2, \ldots$$

where b_i and a_{ij} are some scalars. Then F is a contraction with modulus ρ if and only if

$$\frac{\sum_{j \in X} |a_{ij}|\, v(j)}{v(i)} \le \rho, \qquad \forall\, i = 1, 2, \ldots$$

- Let $F : B(X) \mapsto B(X)$ be a mapping of the form

$$(FJ)(i) = \min_{\mu \in M} (F_\mu J)(i), \qquad \forall\, i = 1, 2, \ldots$$

where M is parameter set, and for each $\mu \in M$, F_μ is a contraction mapping from $B(X)$ to $B(X)$ with modulus ρ. Then F is a contraction mapping with modulus ρ.

- Allows the extension of main DP results from bounded cost to unbounded cost.

CONTRACTION MAPPING FIXED-POINT TH.

- Contraction Mapping Fixed-Point Theorem: If $F : B(X) \mapsto B(X)$ is a contraction with modulus $\rho \in (0,1)$, then there exists a unique $J^* \in B(X)$ such that

$$J^* = FJ^*.$$

Furthermore, if J is any function in $B(X)$, then $\{F^k J\}$ converges to J^* and we have

$$\|F^k J - J^*\| \leq \rho^k \|J - J^*\|, \qquad k = 1, 2, \ldots.$$

- This is a special case of a general result for contraction mappings $F : Y \mapsto Y$ over normed vector spaces Y that are *complete*: every sequence $\{y_k\}$ that is Cauchy (satisfies $\|y_m - y_n\| \to 0$ as $m, n \to \infty$) converges.

- The space $B(X)$ is complete (see the text for a proof).

GENERAL FORMS OF DISCOUNTED DP

- We consider an abstract form of DP based on monotonicity and contraction

- Abstract Mapping: Denote $R(X)$: set of real-valued functions $J : X \mapsto \Re$, and let $H : X \times U \times R(X) \mapsto \Re$ be a given mapping. We consider the mapping

$$(TJ)(x) = \min_{u \in U(x)} H(x, u, J), \qquad \forall\, x \in X.$$

- We assume that $(TJ)(x) > -\infty$ for all $x \in X$, so T maps $R(X)$ into $R(X)$.

- Abstract Policies: Let $\mathcal{M}$ be the set of "policies", i.e., functions μ such that $\mu(x) \in U(x)$ for all $x \in X$.

- For each $\mu \in \mathcal{M}$, we consider the mapping $T_\mu : R(X) \mapsto R(X)$ defined by

$$(T_\mu J)(x) = H\big(x, \mu(x), J\big), \qquad \forall\, x \in X.$$

- Find a function $J^* \in R(X)$ such that

$$J^*(x) = \min_{u \in U(x)} H(x, u, J^*), \qquad \forall\, x \in X$$

EXAMPLES

- Discounted problems (and stochastic shortest paths-SSP for $\alpha = 1$)

$$H(x,u,J) = E\big\{g(x,u,w) + \alpha J\big(f(x,u,w)\big)\big\}$$

- Discounted Semi-Markov Problems

$$H(x,u,J) = G(x,u) + \sum_{y=1}^{n} m_{xy}(u)J(y)$$

where m_{xy} are "discounted" transition probabilities, defined by the transition distributions

- Shortest Path Problems

$$H(x,u,J) = \begin{cases} a_{xu} + J(u) & \text{if } u \neq d, \\ a_{xd} & \text{if } u = d \end{cases}$$

where d is the destination. There is also a stochastic version of this problem.

- Minimax Problems

$$H(x,u,J) = \max_{w \in W(x,u)} \big[g(x,u,w) + \alpha J\big(f(x,u,w)\big)\big]$$

ASSUMPTIONS

- Monotonicity assumption: If $J, J' \in R(X)$ and $J \leq J'$, then

$$H(x, u, J) \leq H(x, u, J'), \qquad \forall\, x \in X,\ u \in U(x)$$

- Contraction assumption:
  - For every $J \in B(X)$, the functions $T_\mu J$ and TJ belong to $B(X)$.
  - For some $\alpha \in (0, 1)$, and all μ and $J, J' \in B(X)$, we have

$$\|T_\mu J - T_\mu J'\| \leq \alpha \|J - J'\|$$

- We can show all the standard analytical and computational results of discounted DP based on these two assumptions

- With just the monotonicity assumption (as in the SSP or other undiscounted problems) we can still show various forms of the basic results under appropriate assumptions

RESULTS USING CONTRACTION

- **Proposition 1:** The mappings T_μ and T are weighted sup-norm contraction mappings with modulus α over $B(X)$, and have unique fixed points in $B(X)$, denoted J_μ and J^*, respectively (cf. Bellman's equation).

Proof: From the contraction property of H.

- **Proposition 2:** For any $J \in B(X)$ and $\mu \in \mathcal{M}$,

$$\lim_{k\to\infty} T_\mu^k J = J_\mu, \qquad \lim_{k\to\infty} T^k J = J^*$$

(cf. convergence of value iteration).

Proof: From the contraction property of T_μ and T.

- **Proposition 3:** We have $T_\mu J^* = TJ^*$ if and only if $J_\mu = J^*$ (cf. optimality condition).

Proof: $T_\mu J^* = TJ^*$, then $T_\mu J^* = J^*$, implying $J^* = J_\mu$. Conversely, if $J_\mu = J^*$, then $T_\mu J^* = T_\mu J_\mu = J_\mu = J^* = TJ^*$.

RESULTS USING MON. AND CONTRACTION

- Optimality of fixed point:

$$J^*(x) = \min_{\mu \in \mathcal{M}} J_\mu(x), \qquad \forall\, x \in X$$

- Furthermore, for every $\epsilon > 0$, there exists $\mu_\epsilon \in \mathcal{M}$ such that

$$J^*(x) \le J_{\mu_\epsilon}(x) \le J^*(x) + \epsilon, \qquad \forall\, x \in X$$

- Nonstationary policies: Consider the set Π of all sequences $\pi = \{\mu_0, \mu_1, \ldots\}$ with $\mu_k \in \mathcal{M}$ for all k, and define

$$J_\pi(x) = \liminf_{k \to \infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J)(x), \qquad \forall\, x \in X,$$

with J being any function (the choice of J does not matter)

- We have

$$J^*(x) = \min_{\pi \in \Pi} J_\pi(x), \qquad \forall\, x \in X$$

THE TWO MAIN ALGORITHMS: VI AND PI

- Value iteration: For any (bounded) J

$$J^*(x) = \lim_{k\to\infty} (T^k J)(x), \qquad \forall\, x$$

- Policy iteration: Given μ^k
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

  - Policy improvement: Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$$

- Optimistic PI: This is PI, where policy evaluation is carried out by a finite number of VI
  - Shorthand definition: For some integers m_k

$$T_{\mu^k} J_k = T J_k, \qquad J_{k+1} = T_{\mu^k}^{m_k} J_k, \qquad k = 0, 1, \ldots$$

  - If $m_k \equiv 1$ it becomes VI
  - If $m_k = \infty$ it becomes PI
  - For intermediate values of m_k, it is generally more efficient than either VI or PI

ASYNCHRONOUS ALGORITHMS

- Motivation for asynchronous algorithms
  - Faster convergence
  - Parallel and distributed computation
  - Simulation-based implementations

- General framework: Partition X into disjoint nonempty subsets $X_1, \ldots, X_m$, and use separate processor ℓ updating $J(x)$ for $x \in X_\ell$

- Let J be partitioned as

$$J = (J_1, \ldots, J_m),$$

where J_ℓ is the restriction of J on the set X_ℓ.

- Synchronous algorithm:

$$J_\ell^{t+1}(x) = T(J_1^t, \ldots, J_m^t)(x), \quad x \in X_\ell,\ \ell = 1, \ldots, m$$

- Asynchronous algorithm: For some subsets of times $\mathcal{R}_\ell$,

$$J_\ell^{t+1}(x) = \begin{cases} T(J_1^{\tau_{\ell 1}(t)}, \ldots, J_m^{\tau_{\ell m}(t)})(x) & \text{if } t \in \mathcal{R}_\ell, \\ J_\ell^t(x) & \text{if } t \notin \mathcal{R}_\ell \end{cases}$$

where $t - \tau_{\ell j}(t)$ are communication "delays"

ONE-STATE-AT-A-TIME ITERATIONS

- Important special case: Assume n "states", a separate processor for each state, and no delays
- Generate a sequence of states $\{x^0, x^1, \ldots\}$, generated in some way, possibly by simulation (each state is generated infinitely often)
- Asynchronous VI:

$$J_\ell^{t+1} = \begin{cases} T(J_1^t, \ldots, J_n^t)(\ell) & \text{if } \ell = x^t, \\ J_\ell^t & \text{if } \ell \neq x^t, \end{cases}$$

where $T(J_1^t, \ldots, J_n^t)(\ell)$ denotes the ℓ-th component of the vector

$$T(J_1^t, \ldots, J_n^t) = TJ^t,$$

and for simplicity we write J_ℓ^t instead of $J_\ell^t(\ell)$

- The special case where

$$\{x^0, x^1, \ldots\} = \{1, \ldots, n, 1, \ldots, n, 1, \ldots\}$$

is the Gauss-Seidel method

- We can show that $J^t \to J^*$ under the contraction assumption

ASYNCHRONOUS CONV. THEOREM I

- Assume that for all $\ell, j = 1, \ldots, m$, $\mathcal{R}_\ell$ is infinite and $\lim_{t\to\infty} \tau_{\ell j}(t) = \infty$

- Proposition: Let T have a unique fixed point J^*, and assume that there is a sequence of nonempty subsets $\{S(k)\} \subset R(X)$ with $S(k+1) \subset S(k)$ for all k, and with the following properties:

  (1) Synchronous Convergence Condition: Every sequence $\{J^k\}$ with $J^k \in S(k)$ for each k, converges pointwise to J^*. Moreover, we have

  $$TJ \in S(k+1), \qquad \forall\, J \in S(k),\ k = 0, 1, \ldots.$$

  (2) Box Condition: For all k, $S(k)$ is a Cartesian product of the form

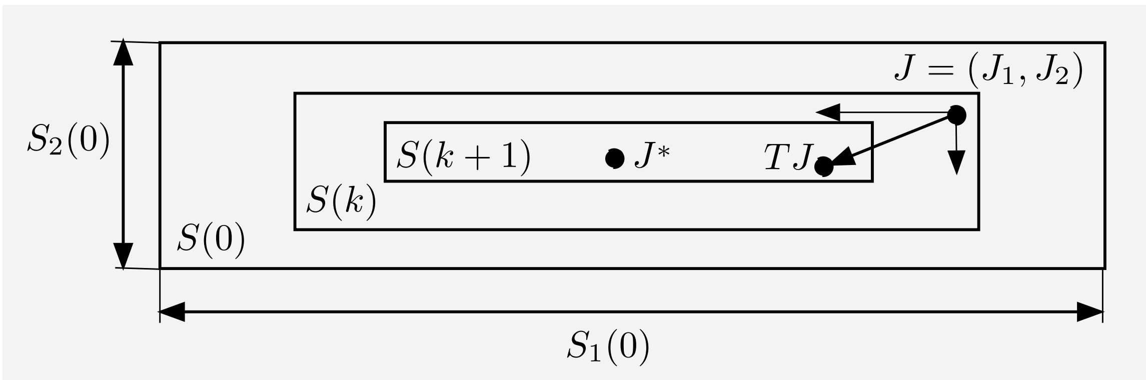
  $$S(k) = S_1(k) \times \cdots \times S_m(k),$$

  where $S_\ell(k)$ is a set of real-valued functions on X_ℓ, $\ell = 1, \ldots, m$.

Then for every $J \in S(0)$, the sequence $\{J^t\}$ generated by the asynchronous algorithm converges pointwise to J^*.

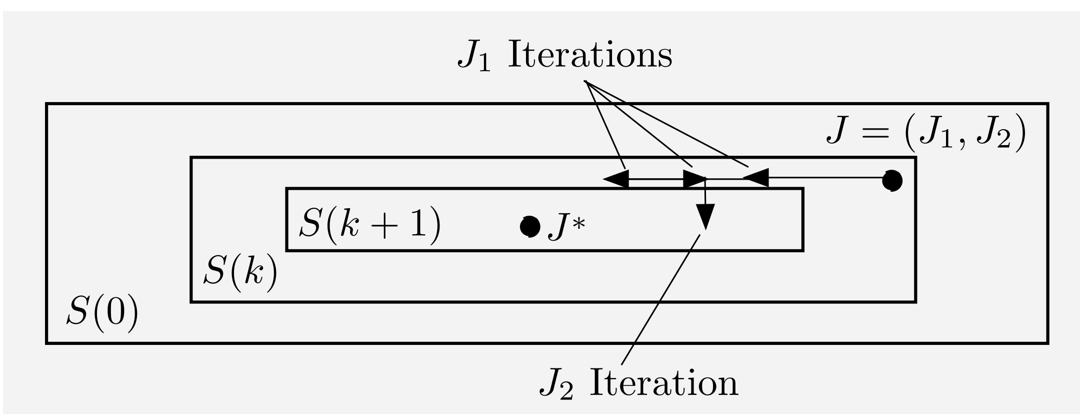
ASYNCHRONOUS CONV. THEOREM II

- Interpretation of assumptions:



A synchronous iteration from any J in $S(k)$ moves into $S(k+1)$ (component-by-component)

- Convergence mechanism:



Key: "Independent" component-wise improvement. An asynchronous component iteration from any J in $S(k)$ moves into the corresponding component portion of $S(k+1)$

APPROXIMATE DYNAMIC PROGRAMMING

LECTURE 3

LECTURE OUTLINE

- Review of theory and algorithms for discounted DP
- MDP and stochastic shortest path problems (briefly)
- Introduction to approximation in policy and value space
- Approximation architectures
- Simulation-based approximate policy iteration
- Approximate policy iteration and Q-factors
- Direct and indirect approximation
- Simulation issues

DISCOUNTED PROBLEMS/BOUNDED COST

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots$$

- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

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

with $\alpha < 1$, and for some M, we have $|g(x,u,w)| \leq M$ for all (x,u,w)

- Shorthand notation for DP mappings (operate on functions of state to produce other functions)

$$(TJ)(x) = \min_{u\in U(x)} \mathop{E}_{w} \left\{ g(x,u,w) + \alpha J\big(f(x,u,w)\big) \right\}, \ \forall\, x$$

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ

- For any stationary policy μ

$$(T_\mu J)(x) = \mathop{E}_{w} \left\{ g\big(x, \mu(x), w\big) + \alpha J\big(f(x, \mu(x), w)\big) \right\}, \ \forall\, x$$

MDP - TRANSITION PROBABILITY NOTATION

- Assume the system is an n-state (controlled) Markov chain
- Change to Markov chain notation
  - States $i = 1, \ldots, n$ (instead of x)
  - Transition probabilities $p_{i_k i_{k+1}}(u_k)$ [instead of $x_{k+1} = f(x_k, u_k, w_k)$]
  - Stage cost $g(i_k, u_k, i_{k+1})$ [instead of $g(x_k, u_k, w_k)$]
- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

$$J_\pi(i) = \lim_{N \to \infty} \mathop{E}_{\substack{i_k \\ k=1,2,\ldots}} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(i_k, \mu_k(i_k), i_{k+1}\big) \mid i_0 = i \right\}$$

- Shorthand notation for DP mappings

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J(j)\big), \quad i = 1, \ldots, n,$$

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g\big(i, \mu(i), j\big) + \alpha J(j)\Big), \quad i = 1, \ldots, n$$

“SHORTHAND” THEORY – A SUMMARY

- Cost function expressions [with $J_0(i) \equiv 0$]

$$J_\pi(i) = \lim_{k\to\infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0)(i), \quad J_\mu(i) = \lim_{k\to\infty} (T_\mu^k J_0)(i)$$

- Bellman's equation: $J^* = TJ^*$, $J_\mu = T_\mu J_\mu$ or

$$J^*(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \quad \forall\ i$$

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g\big(i,\mu(i),j\big) + \alpha J_\mu(j)\Big), \quad \forall\ i$$

- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

i.e.,

$$\mu(i) \in \arg \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)+\alpha J^*(j)\big), \quad \forall\ i$$

THE TWO MAIN ALGORITHMS: VI AND PI

- Value iteration: For any $J \in \Re^n$

$$J^*(i) = \lim_{k \to \infty} (T^k J)(i), \qquad \forall\ i = 1, \ldots, n$$

- Policy iteration: Given μ^k
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big)\Big(g\big(i, \mu^k(i), j\big) + \alpha J_{\mu^k}(j)\Big), \ \ i = 1, \ldots, n$$

or $J_{\mu^k} = T_{\mu^k} J_{\mu^k}$

  - Policy improvement: Let μ^{k+1} be such that

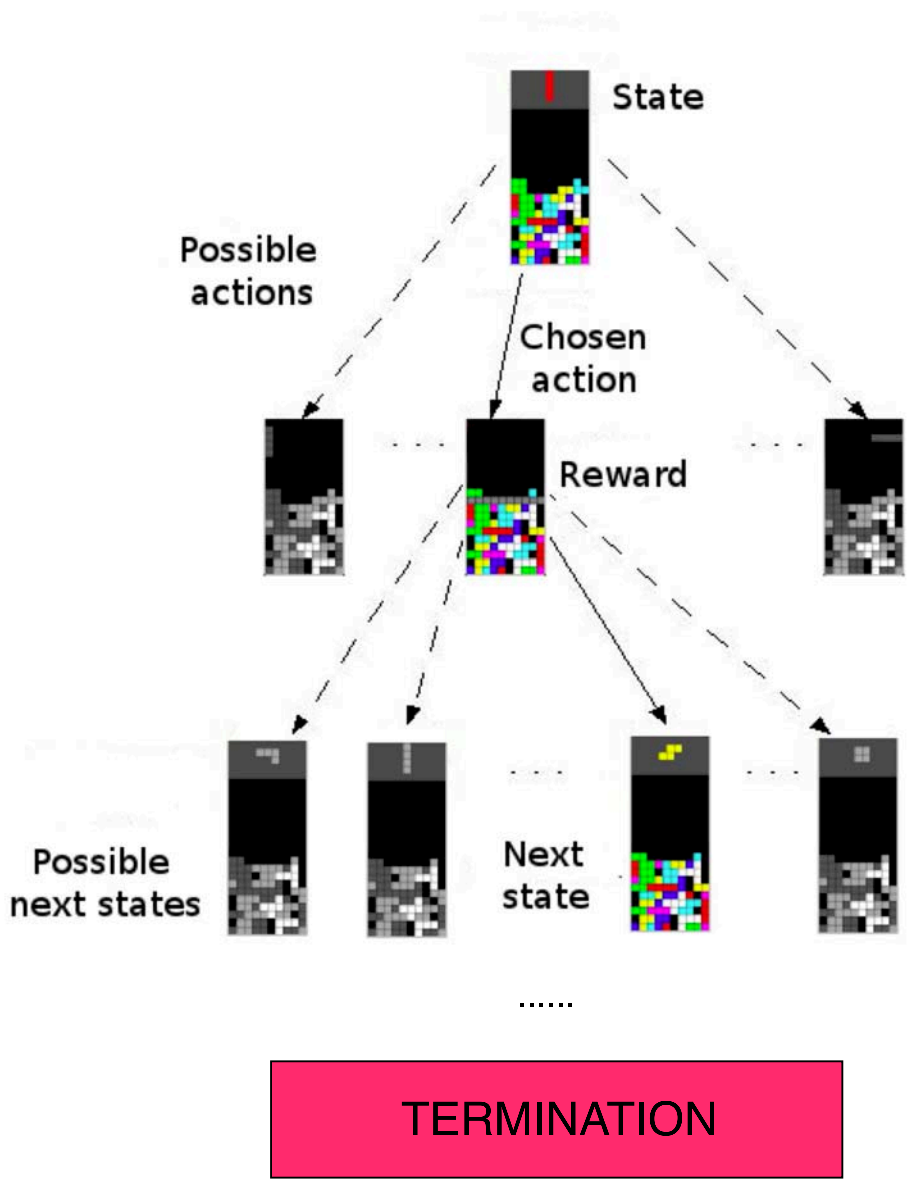
$$\mu^{k+1}(i) \in \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i, u, j) + \alpha J_{\mu^k}(j)\Big), \ \ \forall\ i$$

or $T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$

- Policy evaluation is equivalent to solving an $n \times n$ linear system of equations
- For large n, exact PI is out of the question (even though it terminates finitely)

STOCHASTIC SHORTEST PATH (SSP) PROBLEMS

- Involves states $i = 1, \ldots, n$ plus a special cost-free and absorbing termination state t
- Objective: Minimize the total (undiscounted) cost. Aim: Reach t at minimum expected cost
- An example: Tetris



SSP THEORY

- SSP problems provide a "soft boundary" between the easy finite-state discounted problems and the hard undiscounted problems.
  - They share features of both.
  - Some of the nice theory is recovered because of the termination state.
- Definition: A proper policy is a stationary policy that leads to t with probability 1
- If all stationary policies are proper, T and T_μ are contractions with respect to a common weighted sup-norm
- The entire analytical and algorithmic theory for discounted problems goes through if all stationary policies are proper (we will assume this)
- There is a strong theory even if there are improper policies (but they should be assumed to be nonoptimal - see the textbook)

GENERAL ORIENTATION TO ADP

- We will mainly adopt an n-state discounted model (the easiest case - but think of HUGE n).
- Extensions to SSP and average cost are possible (but more quirky). We will set aside for later.
- There are many approaches:
  - Manual/trial-and-error approach
  - Problem approximation
  - Simulation-based approaches (we will focus on these): "neuro-dynamic programming" or "reinforcement learning".
- Simulation is essential for large state spaces because of its (potential) computational complexity advantage in computing sums/expectations involving a very large number of terms.
- Simulation also comes in handy when an analytical model of the system is unavailable, but a simulation/computer model is possible.
- Simulation-based methods are of three types:
  - Rollout (we will not discuss further)
  - Approximation in value space
  - Approximation in policy space

APPROXIMATION IN VALUE SPACE

- Approximate J^* or J_μ from a parametric class $\tilde{J}(i, r)$ where i is the current state and $r = (r_1, \ldots, r_m)$ is a vector of "tunable" scalars weights.

- By adjusting r we can change the "shape" of $\tilde{J}$ so that it is reasonably close to the true optimal J^*.

- Two key issues:
  - The choice of parametric class $\tilde{J}(i, r)$ (the approximation architecture).
  - Method for tuning the weights ("training" the architecture).

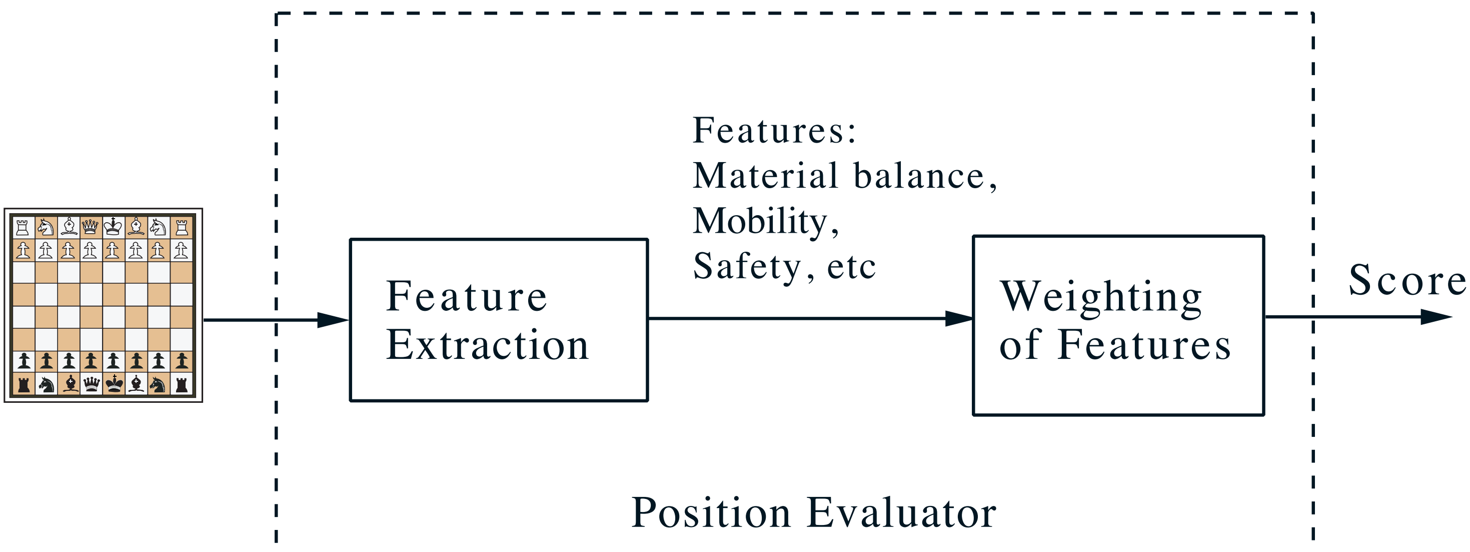
- Successful application strongly depends on how these issues are handled, and on insight about the problem.

- A simulator may be used, particularly when there is no mathematical model of the system (but there is a computer model).

- We will focus on simulation, but this is not the only possibility [e.g., $\tilde{J}(i, r)$ may be a lower bound approximation based on relaxation, or other problem approximation]

APPROXIMATION ARCHITECTURES

- Divided in linear and nonlinear [i.e., linear or nonlinear dependence of $\tilde{J}(i,r)$ on r].
- Linear architectures are easier to train, but non-linear ones (e.g., neural networks) are richer.
- Computer chess example: Uses a feature-based position evaluator that assigns a score to each move/position.



- Many context-dependent special features.
- Most often the weighting of features is linear but multistep lookahead is involved.
- In chess, most often the training is done by trial and error.

LINEAR APPROXIMATION ARCHITECTURES

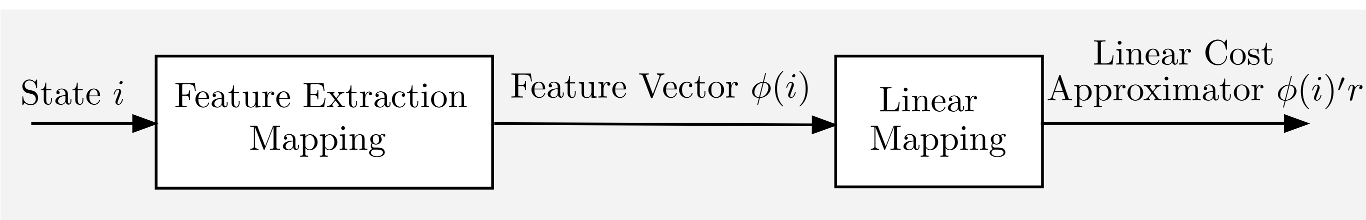
- Ideally, the features encode much of the nonlinearity inherent in the cost-to-go approximated

- Then the approximation may be quite accurate without a complicated architecture.

- With well-chosen features, we can use a linear architecture: $\tilde{J}(i,r) = \phi(i)'r$, $i = 1,\ldots,n$, or more compactly

$$\tilde{J}(r) = \Phi r$$

Φ: the matrix whose rows are $\phi(i)'$, $i = 1,\ldots,n$



- This is approximation on the subspace

$$S = \{\Phi r \mid r \in \Re^s\}$$

spanned by the columns of Φ (basis functions)

- Many examples of feature types: Polynomial approximation, radial basis functions, kernels of all sorts, interpolation, and special problem-specific (as in chess and tetris)

APPROXIMATION IN POLICY SPACE

- A brief discussion; we will return to it at the end.
- We parameterize the set of policies by a vector $r = (r_1, \ldots, r_s)$ and we optimize the cost over r
- Discounted problem example:
  - Each value of r defines a stationary policy, with cost starting at state i denoted by $\tilde{J}(i; r)$.
  - Use a random search, gradient, or other method to minimize over r

$$\bar{J}(r) = \sum_{i=1}^{n} p_i \tilde{J}(i; r),$$

    where $(p_1, \ldots, p_n)$ is some probability distribution over the states.
- In a special case of this approach, the parameterization of the policies is indirect, through an approximate cost function.
  - A cost approximation architecture parameterized by r, defines a policy dependent on r via the minimization in Bellman's equation.

APPROX. IN VALUE SPACE - APPROACHES

- Approximate PI (Policy evaluation/Policy improvement)
  - Uses simulation algorithms to approximate the cost J_μ of the current policy μ
  - Projected equation and aggregation approaches
- Approximation of the optimal cost function J^*
  - Q-Learning: Use a simulation algorithm to approximate the optimal costs $J^*(i)$ or the Q-factors

$$Q^*(i,u) = g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J^*(j)$$

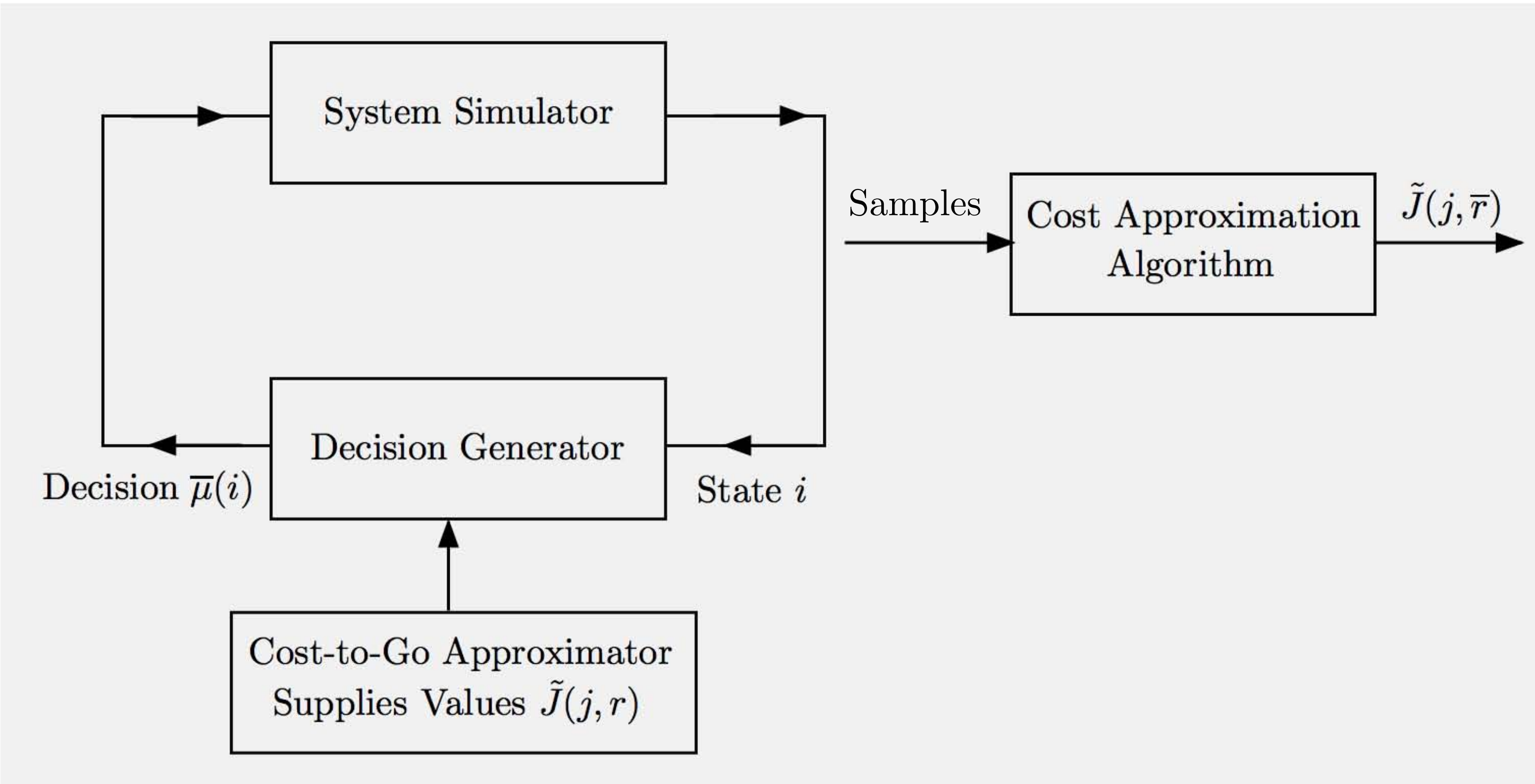
  - Bellman error approach: Find r to

$$\min_r E_i\Big\{ \big(\tilde{J}(i,r) - (T\tilde{J})(i,r)\big)^2 \Big\}$$

    where $E_i\{\cdot\}$ is taken with respect to some distribution
  - Approximate LP (we will not discuss here)

APPROXIMATE POLICY ITERATION

- General structure



- $\tilde{J}(j, r)$ is the cost approximation for the preceding policy, used by the decision generator to compute the current policy $\overline{\mu}$ [whose cost is approximated by $\tilde{J}(j, \overline{r})$ using simulation]

- There are several cost approximation/policy evaluation algorithms

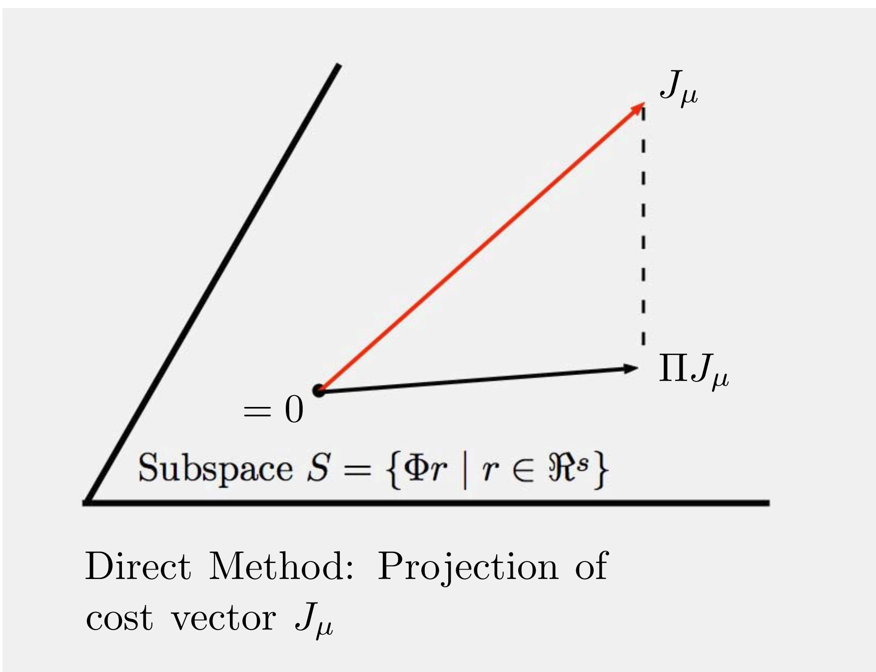
- There are several important issues relating to the design of each block (to be discussed in the future).

POLICY EVALUATION APPROACHES I

- Direct policy evaluation

- Approximate the cost of the current policy by using least squares and simulation-generated cost samples

- Amounts to projection of J_μ onto the approximation subspace



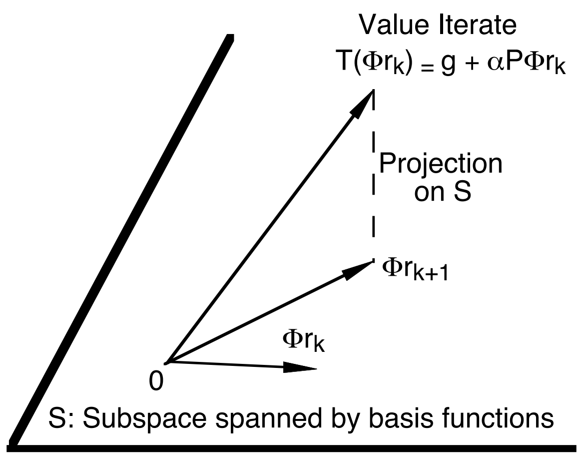
- Solution of the least squares problem by batch and incremental methods

- Regular and optimistic policy iteration

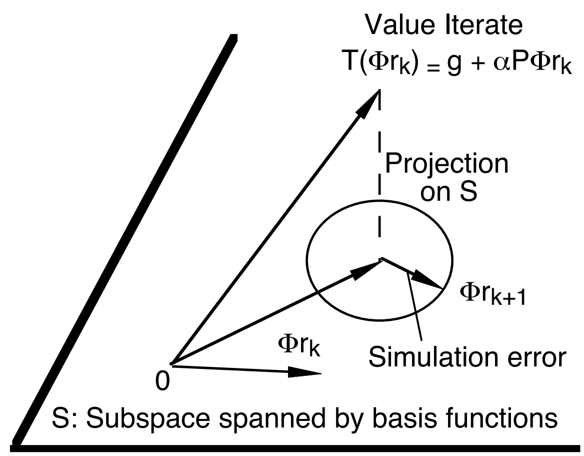
- Nonlinear approximation architectures may also be used

POLICY EVALUATION APPROACHES II

- Indirect policy evaluation



Projected Value Iteration (PVI)



Least Squares Policy Evaluation (LSPE)

- An example of indirect approach: Galerkin approximation
  - Solve the projected equation $\Phi r = \Pi T_\mu(\Phi r)$ where Π is projection w/ respect to a suitable weighted Euclidean norm
  - TD(λ): Stochastic iterative algorithm for solving$\Phi\ r = \Pi T_\mu(\Phi r)$
  - LSPE(λ): A simulation-based form of projected value iteration

$$\Phi r_{k+1} = \Pi T_\mu(\Phi r_k) + \text{ simulation noise}$$

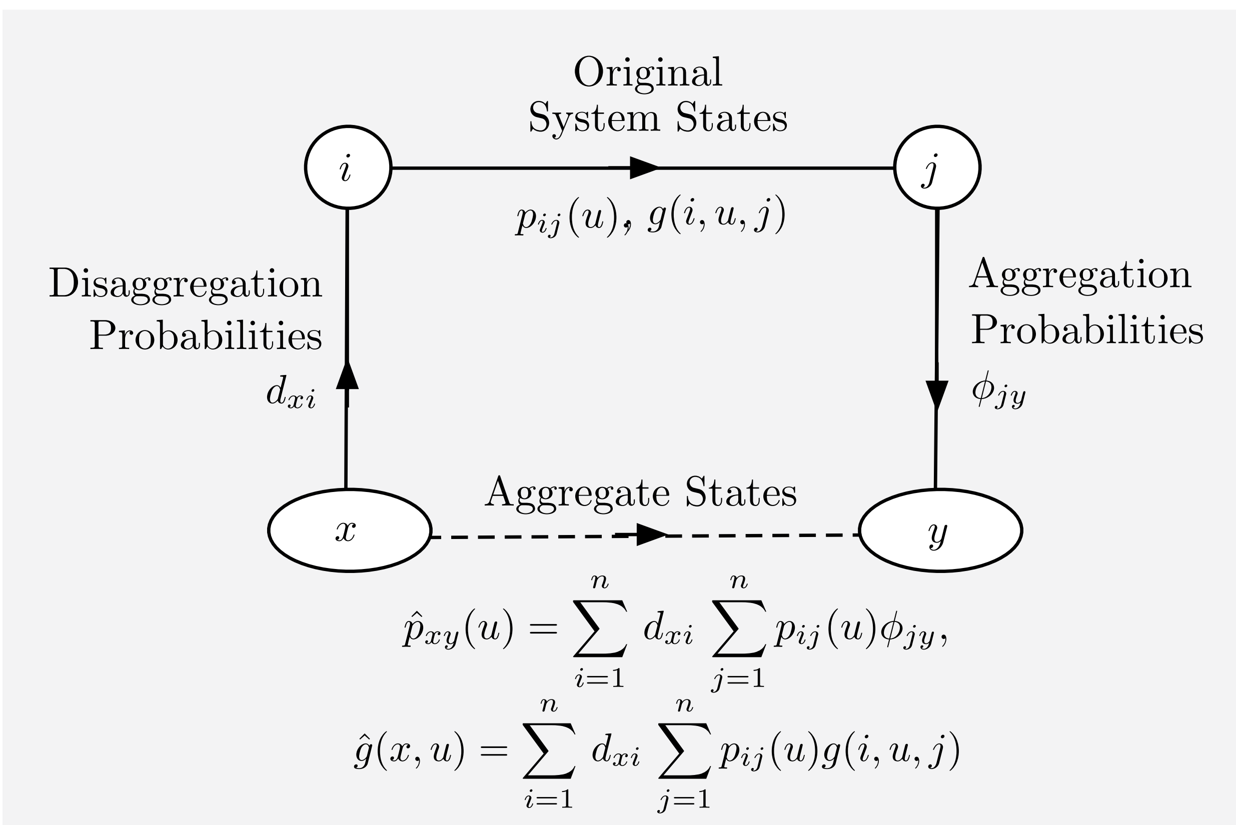
  - LSTD(λ): Solves a simulation-based approximation w/ a standard solver (Matlab)

POLICY EVALUATION APPROACHES III

- Aggregation approximation: Solve

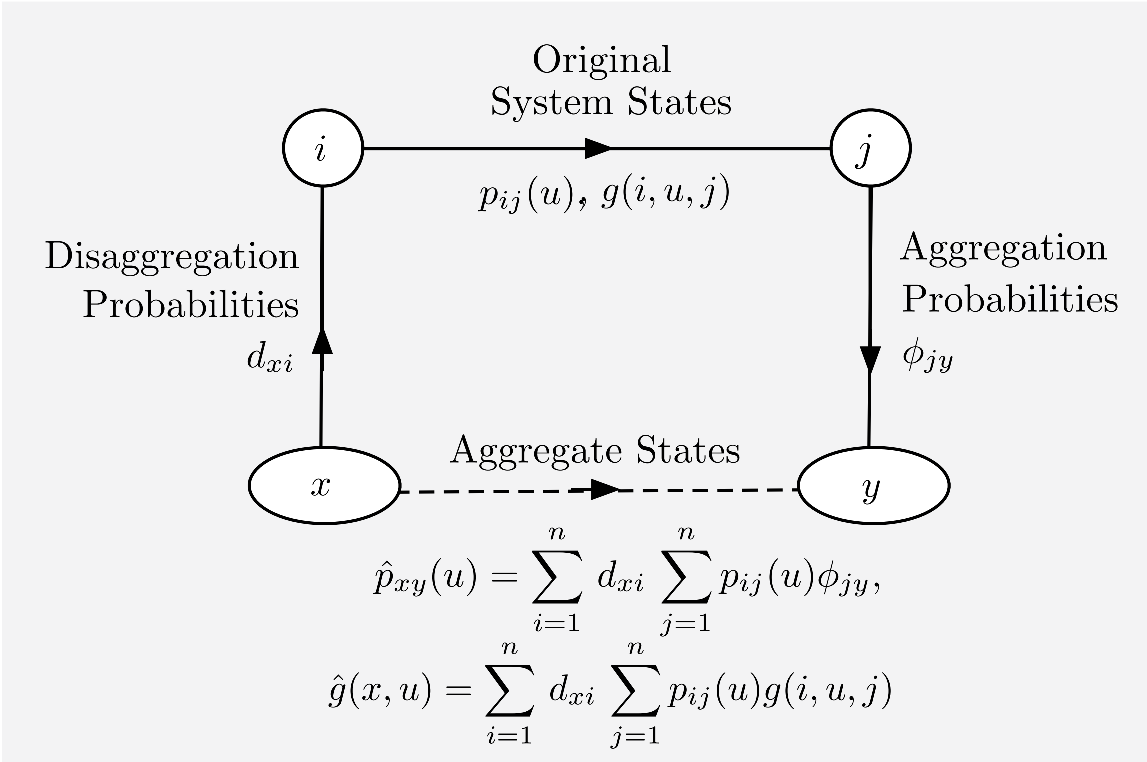
$$\Phi r = \Phi D T_\mu(\Phi r)$$

where the rows of D and Φ are prob. distributions (e.g., D and Φ "aggregate" rows and columns of the linear system $J = T_\mu J$).



- Several different choices of D and Φ.

POLICY EVALUATION APPROACHES IV



- Aggregation is a systematic approach for problem approximation. Main elements:
  - Solve (exactly or approximately) the "aggregate" problem by any kind of VI or PI method (including simulation-based methods)
  - Use the optimal cost of the aggregate problem to approximate the optimal cost of the original problem

- Because an exact PI algorithm is used to solve the approximate/aggregate problem the method behaves more regularly than the projected equation approach

THEORETICAL BASIS OF APPROXIMATE PI

- If policies are approximately evaluated using an approximation architecture such that

$$\max_i |\tilde{J}(i, r_k) - J_{\mu^k}(i)| \leq \delta, \qquad k = 0, 1, \ldots$$

- If policy improvement is also approximate,

$$\max_i |(T_{\mu^{k+1}}\tilde{J})(i, r_k) - (T\tilde{J})(i, r_k)| \leq \epsilon, \qquad k = 0, 1, \ldots$$

- Error bound: The sequence $\{\mu^k\}$ generated by approximate policy iteration satisfies

$$\limsup_{k\to\infty} \max_i \big(J_{\mu^k}(i) - J^*(i)\big) \leq \frac{\epsilon + 2\alpha\delta}{(1-\alpha)^2}$$

- Typical practical behavior: The method makes steady progress up to a point and then the iterates J_{μ^k} oscillate within a neighborhood of J^*.

THE USE OF SIMULATION - AN EXAMPLE

- Projection by Monte Carlo Simulation: Compute the projectionΠ J of a vector $J \in \Re^n$ on subspace $S = \{\Phi r \mid r \in \Re^s\}$, with respect to a weighted Euclidean norm $\|\cdot\|_\xi$.

- Equivalently, findΦ r^*, where

$$r^* = \arg\min_{r \in \Re^s} \|\Phi r - J\|_\xi^2 = \arg\min_{r \in \Re^s} \sum_{i=1}^n \xi_i \big(\phi(i)' r - J(i)\big)^2$$

- Setting to 0 the gradient at r^*,

$$r^* = \left(\sum_{i=1}^n \xi_i \phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^n \xi_i \phi(i) J(i)$$

- Approximate by simulation the two "expected values"

$$\hat{r}_k = \left(\sum_{t=1}^k \phi(i_t)\phi(i_t)'\right)^{-1} \sum_{t=1}^k \phi(i_t) J(i_t)$$

- Equivalent least squares alternative:

$$\hat{r}_k = \arg\min_{r \in \Re^s} \sum_{t=1}^k \big(\phi(i_t)' r - J(i_t)\big)^2$$

THE ISSUE OF EXPLORATION

- To evaluate a policy μ, we need to generate cost samples using that policy - this biases the simulation by underrepresenting states that are unlikely to occur under μ.

- As a result, the cost-to-go estimates of these underrepresented states may be highly inaccurate.

- This seriously impacts the improved policy $\overline{\mu}$.

- This is known as inadequate exploration - a particularly acute difficulty when the randomness embodied in the transition probabilities is "relatively small" (e.g., a deterministic system).

- One possibility for adequate exploration: Frequently restart the simulation and ensure that the initial states employed form a rich and representative subset.

- Another possibility: Occasionally generate transitions that use a randomly selected control rather than the one dictated by the policy μ.

- Other methods, to be discussed later, use two Markov chains (one is the chain of the policy and is used to generate the transition sequence, the other is used to generate the state sequence).

APPROXIMATING Q-FACTORS

- The approach described so far for policy evaluation requires calculating expected values [and knowledge of $p_{ij}(u)$] for all controls $u \in U(i)$.
- Model-free alternative: Approximate Q-factors

$$\tilde{Q}(i,u,r) \approx \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J_\mu(j)\big)$$

and use for policy improvement the minimization

$$\overline{\mu}(i) = \arg\min_{u \in U(i)} \tilde{Q}(i,u,r)$$

- r is an adjustable parameter vector and $\tilde{Q}(i,u,r)$ is a parametric architecture, such as

$$\tilde{Q}(i,u,r) = \sum_{m=1}^{s} r_m \phi_m(i,u)$$

- We can use any approach for cost approximation, e.g., projected equations, aggregation.
- Use the Markov chain with states (i,u) - $p_{ij}(\mu(i))$ is the transition prob. to $(j,\mu(i))$, 0 to other (j,u').
- Major concern: Acutely diminished exploration.

6.231 DYNAMIC PROGRAMMING

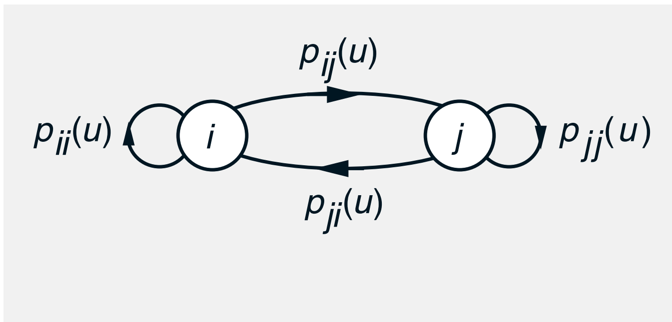
LECTURE 4

LECTURE OUTLINE

- Review of approximation in value space
- Approximate VI and PI
- Projected Bellman equations
- Matrix form of the projected equation
- Simulation-based implementation
- LSTD and LSPE methods
- Optimistic versions
- Multistep projected Bellman equations
- Bias-variance tradeoff

DISCOUNTED MDP

- System: Controlled Markov chain with states $i = 1, \ldots, n$ and finite set of controls $u \in U(i)$
- Transition probabilities: $p_{ij}(u)$



- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ starting at state i:

$$J_\pi(i) = \lim_{N \to \infty} E\left\{\sum_{k=0}^{N} \alpha^k g\big(i_k, \mu_k(i_k), i_{k+1}\big) \mid i = i_0\right\}$$

with $\alpha \in [0, 1)$

- Shorthand notation for DP mappings

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha J(j)\big), \quad i = 1, \ldots, n,$$

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g\big(i, \mu(i), j\big) + \alpha J(j)\big), \quad i = 1, \ldots, n$$

"SHORTHAND" THEORY – A SUMMARY

- Bellman's equation: $J^* = TJ^*$, $J_\mu = T_\mu J_\mu$ or

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \quad \forall\ i$$

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g\big(i,\mu(i),j\big) + \alpha J_\mu(j)\Big), \quad \forall\ i$$

- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

i.e.,

$$\mu(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \quad \forall\ i$$

THE TWO MAIN ALGORITHMS: VI AND PI

- Value iteration: For any $J \in \Re^n$

$$J^*(i) = \lim_{k\to\infty} (T^k J)(i), \qquad \forall\ i = 1, \ldots, n$$

- Policy iteration: Given μ^k
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big)\Big(g\big(i, \mu^k(i), j\big) + \alpha J_{\mu^k}(j)\Big), \ \ i = 1, \ldots, n$$

or $J_{\mu^k} = T_{\mu^k} J_{\mu^k}$

  - Policy improvement: Let μ^{k+1} be such that

$$\mu^{k+1}(i) \in \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha J_{\mu^k}(j)\big), \quad \forall\ i$$

or $T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$

- Policy evaluation is equivalent to solving an $n \times n$ linear system of equations

- For large n, exact PI is out of the question (even though it terminates finitely)

APPROXIMATION IN VALUE SPACE

- Approximate J^* or J_μ from a parametric class $\tilde{J}(i, r)$, where i is the current state and $r = (r_1, \ldots, r_m)$ is a vector of "tunable" scalars weights.

- By adjusting r we can change the "shape" of $\tilde{J}$ so that it is close to the true optimal J^*.

- Any $r \in \Re^s$ defines a (suboptimal) one-step lookahead policy

$$\tilde{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha \tilde{J}(j, r)\big), \quad \forall\, i$$

- We will focus mostly on linear architectures

$$\tilde{J}(r) = \Phi r$$

where Φ is an $n \times s$ matrix whose columns are viewed as basis functions

- Think n: HUGE, s: (Relatively) SMALL

- For $\tilde{J}(r) = \Phi r$, approximation in value space means approximation of J^* or J_μ within the subspace

$$S = \{\Phi r \mid r \in \Re^s\}$$

APPROXIMATE VI

- Approximates sequentially $J_k(i) = (T^k J_0)(i)$, $k = 1, 2, \ldots$, with $\tilde{J}_k(i, r_k)$
- The starting function J_0 is given (e.g., $J_0 \equiv 0$)
- After a large enough number N of steps, $\tilde{J}_N(i, r_N)$ is used as approximation $\tilde{J}(i, r)$ to $J^*(i)$
- Fitted Value Iteration: A sequential "fit" to produce $\tilde{J}_{k+1}$ from $\tilde{J}_k$, i.e., $\tilde{J}_{k+1} \approx T\tilde{J}_k$ or (for a single policy μ) $\tilde{J}_{k+1} \approx T_\mu \tilde{J}_k$
  - For a "small" subset S_k of states i, compute

$$(T\tilde{J}_k)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha \tilde{J}_k(j, r)\big)$$

  - "Fit" the function $\tilde{J}_{k+1}(i, r_{k+1})$ to the "small" set of values $(T\tilde{J}_k)(i)$, $i \in S_k$
  - Simulation can be used for "model-free" implementation
- Error Bound: If the fit is uniformly accurate within $\delta > 0$ (i.e., $\max_i |\tilde{J}_{k+1}(i) - T\tilde{J}_k(i)| \leq \delta$),

$$\limsup_{k \to \infty} \max_{i=1,\ldots,n} \big(\tilde{J}_k(i, r_k) - J^*(i)\big) \leq \frac{2\alpha\delta}{(1-\alpha)^2}$$

AN EXAMPLE OF FAILURE

- Consider two-state discounted MDP with states 1 and 2, and a single policy.
  - Deterministic transitions: $1 \to 2$ and $2 \to 2$
  - Transition costs $\equiv 0$, so $J^*(1) = J^*(2) = 0$.
- Consider approximate VI scheme that approximates cost functions in $S = \big\{(r, 2r) \mid r \in \Re\big\}$ with a weighted least squares fit; here $\Phi = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$
- Given $J_k = (r_k, 2r_k)$, we find $J_{k+1} = (r_{k+1}, 2r_{k+1})$, where for weights $\xi_1, \xi_2 > 0$, r_{k+1} is obtained as

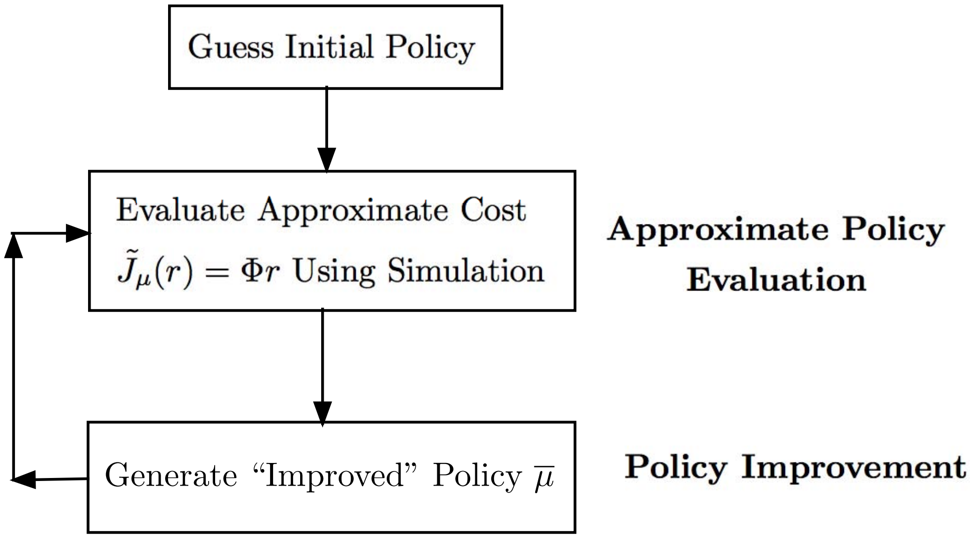
$$r_{k+1} = \arg\min_r \Big[\xi_1\big(r-(TJ_k)(1)\big)^2+\xi_2\big(2r-(TJ_k)(2)\big)^2\Big]$$

- With straightforward calculation

$$r_{k+1} = \alpha\beta r_k, \qquad \text{where } \beta = 2(\xi_1+2\xi_2)/(\xi_1+4\xi_2) > 1$$

- So if $\alpha > 1/\beta$, the sequence $\{r_k\}$ diverges and so does $\{J_k\}$.
- Difficulty is that T is a contraction, but$\Pi\, T$ (= least squares fit composed with T) is not
- Norm mismatch problem

APPROXIMATE PI



- Evaluation of typical policy μ: Linear cost function approximation $\tilde{J}_\mu(r) = \Phi r$, where Φ is full rank $n \times s$ matrix with columns the basis functions, and ith row denoted $\phi(i)'$.

- Policy "improvement" to generate $\overline{\mu}$:

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha\phi(j)'r\big)$$

- Error Bound: If

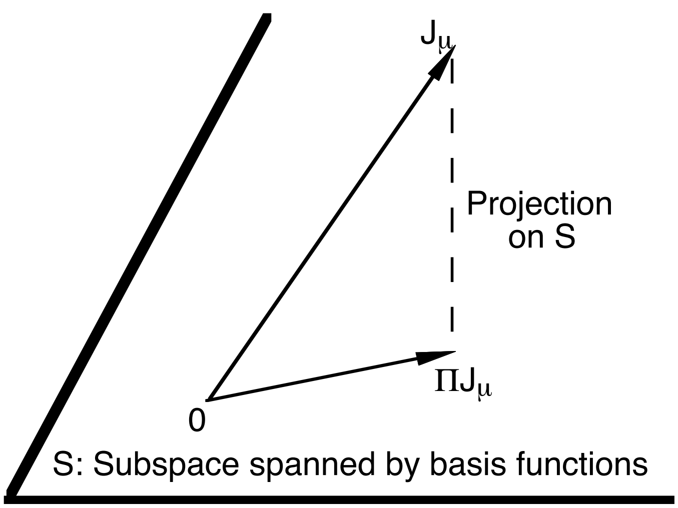
$$\max_i |\tilde{J}_{\mu^k}(i, r_k) - J_{\mu^k}(i)| \leq \delta, \qquad k = 0, 1, \ldots$$

The sequence $\{\mu^k\}$ satisfies

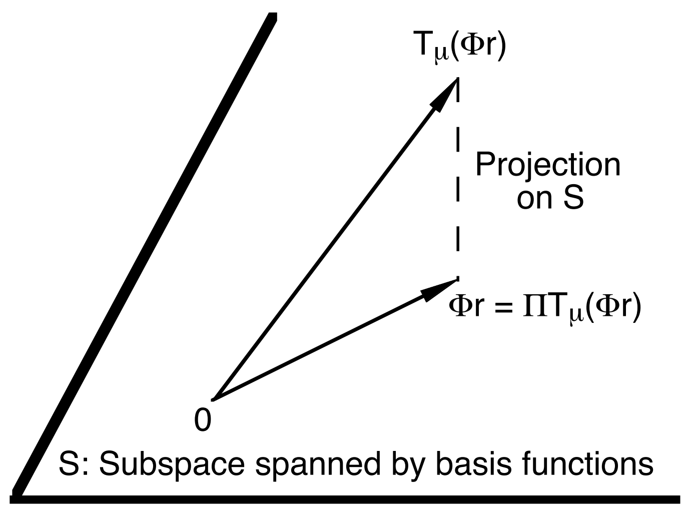
$$\limsup_{k\to\infty} \max_i \big(J_{\mu^k}(i) - J^*(i)\big) \leq \frac{2\alpha\delta}{(1-\alpha)^2}$$

POLICY EVALUATION

- Let's consider approximate evaluation of the cost of the current policy by using simulation.
  - Direct policy evaluation - Cost samples generated by simulation, and optimization by least squares
  - Indirect policy evaluation - solving the projected equation$\Phi\ r = \Pi T_\mu(\Phi r)$ where Π is projection w/ respect to a suitable weighted Euclidean norm



Direct Mehod: Projection of cost vector J_μ



Indirect method: Solving a projected form of Bellman's equation

- Recall that projection can be implemented by simulation and least squares

WEIGHTED EUCLIDEAN PROJECTIONS

- Consider a weighted Euclidean norm

$$\|J\|_\xi = \sqrt{\sum_{i=1}^{n} \xi_i \big(J(i)\big)^2},$$

where ξ is a vector of positive weights $\xi_1, \ldots, \xi_n$.

- Let Π denote the projection operation onto

$$S = \{\Phi r \mid r \in \Re^s\}$$

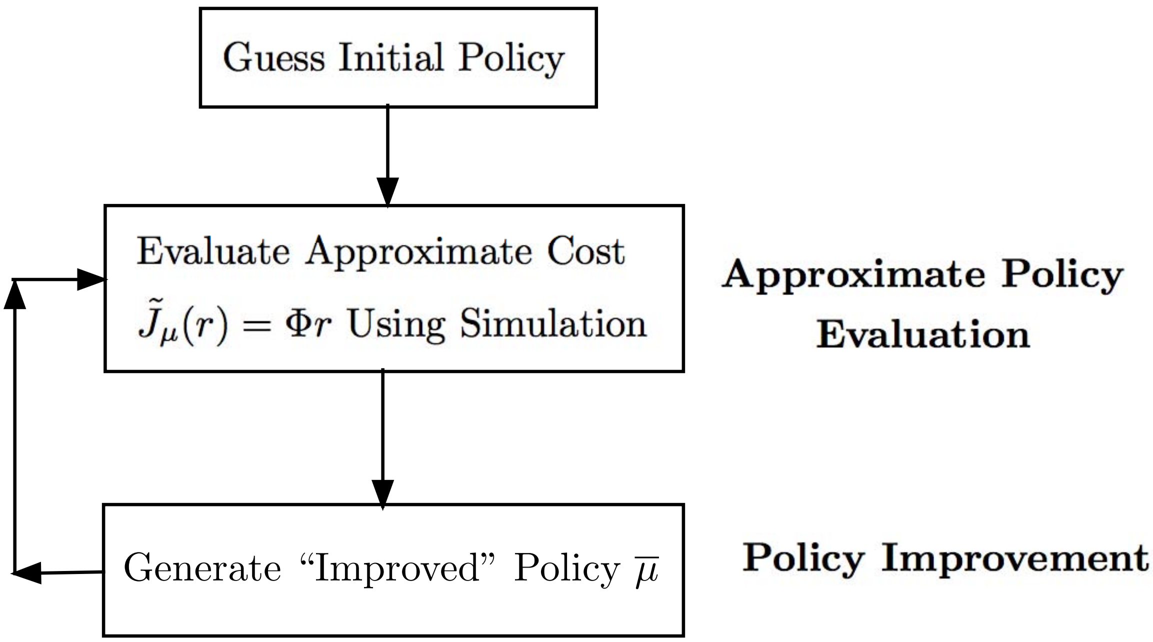
with respect to this norm, i.e., for any $J \in \Re^n$,

$$\Pi J = \Phi r^*$$

where

$$r^* = \arg\min_{r \in \Re^s} \|J - \Phi r\|_\xi^2$$

PI WITH INDIRECT POLICY EVALUATION



- Given the current policy μ:
  - We solve the projected Bellman's equation

$$\Phi r = \Pi T_\mu(\Phi r)$$

  - We approximate the solution J_μ of Bellman's equation

$$J = T_\mu J$$

    with the projected equation solution $\tilde{J}_\mu(r)$

KEY QUESTIONS AND RESULTS

- Does the projected equation have a solution?
- Under what conditions is the mapping ΠT_μ a contraction, so ΠT_μ has unique fixed point?
- Assuming ΠT_μ has unique fixed point Φr^*, how close is Φr^* to J_μ?
- Assumption: The Markov chain corresponding to μ has a single recurrent class and no transient states, i.e., it has steady-state probabilities that are positive

$$\xi_j = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} P(i_k = j \mid i_0 = i) > 0$$

- Proposition: (Norm Matching Property)
  - (a) ΠT_μ is contraction of modulus α with respect to the weighted Euclidean norm $\|\cdot\|_\xi$, where $\xi = (\xi_1, \ldots, \xi_n)$ is the steady-state probability vector.
  - (b) The unique fixed point Φr^* of ΠT_μ satisfies

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi$$

PRELIMINARIES: PROJECTION PROPERTIES

- Important property of the projection Π on S with weighted Euclidean norm $\|\cdot\|_\xi$. For all $J \in \Re^n$, $\overline{J} \in S$, the Pythagorean Theorem holds:

$$\|J - \overline{J}\|_\xi^2 = \|J - \Pi J\|_\xi^2 + \|\Pi J - \overline{J}\|_\xi^2$$

Proof: Geometrically, $(J - \Pi J)$ and $(\,\Pi J - \overline{J})$ are orthogonal in the scaled geometry of the norm $\|\cdot\|_\xi$, where two vectors $x, y \in \Re^n$ are orthogonal if $\sum_{i=1}^n \xi_i x_i y_i = 0$. Expand the quadratic in the RHS below:

$$\|J - \overline{J}\|_\xi^2 = \|(J - \Pi J) + (\,\Pi J - \overline{J})\|_\xi^2$$

- The Pythagorean Theorem implies that the projection is nonexpansive, i.e.,

$$\|\Pi J - \Pi \bar{J}\|_\xi \le \|J - \bar{J}\|_\xi, \qquad \text{for all } J, \bar{J} \in \Re^n.$$

To see this, note that

$$\begin{aligned}\left\|\Pi(J - \overline{J})\right\|_\xi^2 &\le \left\|\Pi(J - \overline{J})\right\|_\xi^2 + \left\|(I - \Pi)(J - \overline{J})\right\|_\xi^2 \\ &= \|J - \overline{J}\|_\xi^2\end{aligned}$$

PROOF OF CONTRACTION PROPERTY

- Lemma: If P is the transition matrix of μ,

$$\|Pz\|_\xi \leq \|z\|_\xi, \qquad z \in \Re^n$$

Proof: Let p_{ij} be the components of P. For all $z \in \Re^n$, we have

$$\|Pz\|_\xi^2 = \sum_{i=1}^n \xi_i \left(\sum_{j=1}^n p_{ij} z_j \right)^2 \leq \sum_{i=1}^n \xi_i \sum_{j=1}^n p_{ij} z_j^2$$

$$= \sum_{j=1}^n \sum_{i=1}^n \xi_i p_{ij} z_j^2 = \sum_{j=1}^n \xi_j z_j^2 = \|z\|_\xi^2,$$

where the inequality follows from the convexity of the quadratic function, and the next to last equality follows from the defining property $\sum_{i=1}^n \xi_i p_{ij} = \xi_j$ of the steady-state probabilities.

- Using the lemma, the nonexpansiveness of Π, and the definition $T_\mu J = g + \alpha P J$, we have

$$\|\Pi T_\mu J - \Pi T_\mu \bar{J}\|_\xi \leq \|T_\mu J - T_\mu \bar{J}\|_\xi = \alpha \|P(J - \bar{J})\|_\xi \leq \alpha \|J - \bar{J}\|_\xi$$

for all $J, \bar{J} \in \Re^n$. Hence ΠT_μ is a contraction of modulus α.

PROOF OF ERROR BOUND

- LetΦ r^* be the fixed point of ΠT. We have

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi.$$

Proof: We have

$$\begin{aligned}
\|J_\mu - \Phi r^*\|_\xi^2 &= \|J_\mu - \Pi J_\mu\|_\xi^2 + \big\|\Pi J_\mu - \Phi r^*\big\|_\xi^2 \\
&= \|J_\mu - \Pi J_\mu\|_\xi^2 + \big\|\Pi T J_\mu - \Pi T(\Phi r^*)\big\|_\xi^2 \\
&\leq \|J_\mu - \Pi J_\mu\|_\xi^2 + \alpha^2 \|J_\mu - \Phi r^*\|_\xi^2,
\end{aligned}$$

where

- The first equality uses the Pythagorean Theorem
- The second equality holds because J_μ is the fixed point of T and Φr^* is the fixed point of ΠT
- The inequality uses the contraction property of ΠT.

Q.E.D.

MATRIX FORM OF PROJECTED EQUATION

- Its solution is the vector $J = \Phi r^*$, where r^* solves the problem

$$\min_{r \in \Re^s} \left\| \Phi r - (g + \alpha P \Phi r^*) \right\|_\xi^2.$$

- Setting to 0 the gradient with respect to r of this quadratic, we obtain

$$\Phi' \Xi \big(\Phi r^* - (g + \alpha P \Phi r^*) \big) = 0,$$

where Ξ is the diagonal matrix with the steady-state probabilities $\xi_1, \ldots, \xi_n$ along the diagonal.

- This is just the orthogonality condition: The error$\Phi\ r^* - (g + \alpha P \Phi r^*)$ is "orthogonal" to the subspace spanned by the columns of Φ.

- Equivalently,

$$Cr^* = d,$$

where

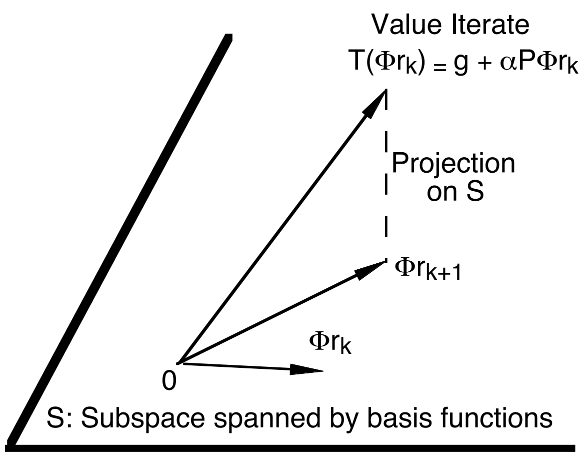
$$C = \Phi' \Xi (I - \alpha P) \Phi, \qquad d = \Phi' \Xi g.$$

PROJECTED EQUATION: SOLUTION METHODS

- Matrix inversion: $r^* = C^{-1}d$
- Projected Value Iteration (PVI) method:

$$\Phi r_{k+1} = \Pi T(\Phi r_k) = \Pi\big(g + \alpha P \Phi r_k\big)$$

Converges to r^* because$\Pi\, T$ is a contraction.



- PVI can be written as:

$$r_{k+1} = \arg\min_{r \in \Re^s} \big\|\Phi r - (g + \alpha P \Phi r_k)\big\|_\xi^2$$

By setting to 0 the gradient with respect to r,

$$\Phi' \Xi \big(\Phi r_{k+1} - (g + \alpha P \Phi r_k)\big) = 0,$$

which yields

$$r_{k+1} = r_k - (\Phi' \Xi \Phi)^{-1}(C r_k - d)$$

SIMULATION-BASED IMPLEMENTATIONS

- Key idea: Calculate simulation-based approximations based on k samples

$$C_k \approx C, \qquad d_k \approx d$$

- Matrix inversion $r^* = C^{-1}d$ is approximated by

$$\hat{r}_k = C_k^{-1} d_k$$

This is the LSTD (Least Squares Temporal Differences) Method.

- PVI method $r_{k+1} = r_k - (\Phi'\Xi\Phi)^{-1}(Cr_k - d)$ is approximated by

$$r_{k+1} = r_k - G_k(C_k r_k - d_k)$$

where

$$G_k \approx (\Phi'\Xi\Phi)^{-1}$$

This is the LSPE (Least Squares Policy Evaluation) Method.

- Key fact: C_k, d_k, and G_k can be computed with low-dimensional linear algebra (of order s; the number of basis functions).

SIMULATION MECHANICS

- We generate an infinitely long trajectory $(i_0, i_1, \ldots)$ of the Markov chain, so states i and transitions (i, j) appear with long-term frequencies ξ_i and p_{ij}.

- After generating the transition (i_t, i_{t+1}), we compute the row $\phi(i_t)'$ of Φ and the cost component $g(i_t, i_{t+1})$.

- We form

$$C_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\big(\phi(i_t) - \alpha\phi(i_{t+1})\big)' \approx \Phi'\Xi(I - \alpha P)\Phi$$

$$d_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) g(i_t, i_{t+1}) \approx \Phi'\Xi g$$

Also in the case of LSPE

$$G_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\phi(i_t)' \approx \Phi'\Xi\Phi$$

- Convergence based on law of large numbers.

- C_k, d_k, and G_k can be formed incrementally. Also can be written using the formalism of temporal differences (this is just a matter of style)

OPTIMISTIC VERSIONS

- Instead of calculating nearly exact approximations $C_k \approx C$ and $d_k \approx d$, we do a less accurate approximation, based on few simulation samples
- Evaluate (coarsely) current policy μ, then do a policy improvement
- This often leads to faster computation (as optimistic methods often do)
- Very complex behavior (see the subsequent discussion on oscillations)
- The matrix inversion/LSTD method has serious problems due to large simulation noise (because of limited sampling)
- LSPE tends to cope better because of its iterative nature
- A stepsize $\gamma \in (0, 1]$ in LSPE may be useful to damp the effect of simulation noise

$$r_{k+1} = r_k - \gamma G_k (C_k r_k - d_k)$$

MULTISTEP METHODS

- Introduce a multistep version of Bellman's equation $J = T^{(\lambda)}J$, where for $\lambda \in [0,1)$,

$$T^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^{\ell} T^{\ell+1}$$

Geometrically weighted sum of powers of T.

- Note that T^{ℓ} is a contraction with modulus α^{ℓ}, with respect to the weighted Euclidean norm $\|\cdot\|_{\xi}$, where ξ is the steady-state probability vector of the Markov chain.

- Hence $T^{(\lambda)}$ is a contraction with modulus

$$\alpha_{\lambda} = (1-\lambda)\sum_{\ell=0}^{\infty} \alpha^{\ell+1}\lambda^{\ell} = \frac{\alpha(1-\lambda)}{1-\alpha\lambda}$$

Note that $\alpha_{\lambda} \to 0$ as $\lambda \to 1$

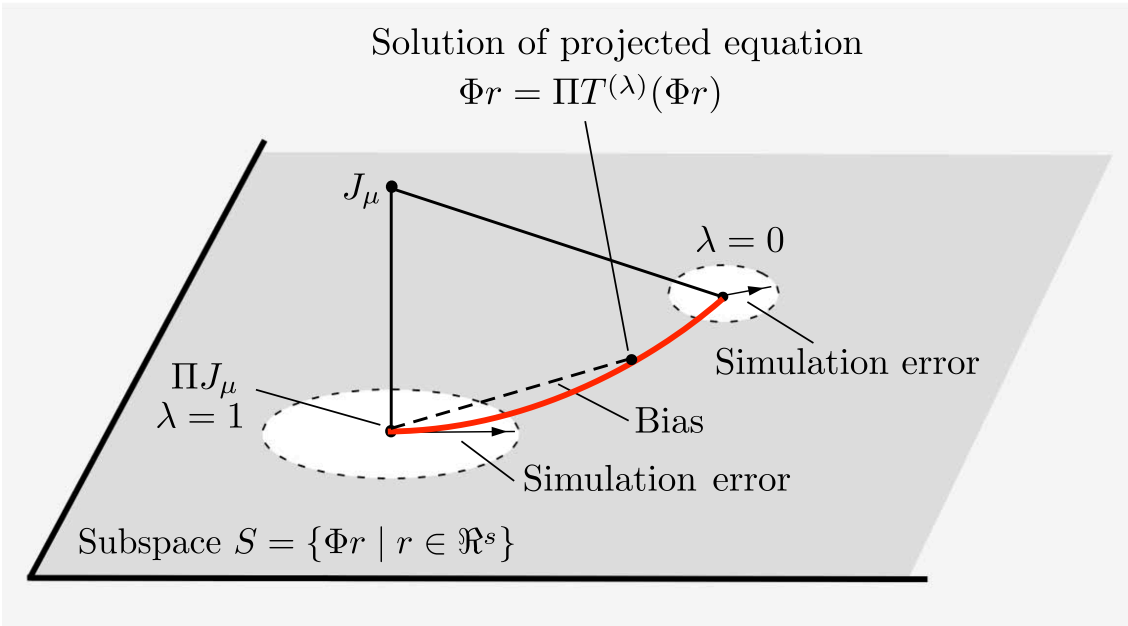
- T^t and $T^{(\lambda)}$ have the same fixed point J_{μ} and

$$\|J_{\mu} - \Phi r_{\lambda}^{*}\|_{\xi} \leq \frac{1}{\sqrt{1-\alpha_{\lambda}^{2}}}\,\|J_{\mu} - \Pi J_{\mu}\|_{\xi}$$

where Φr_{λ}^{*} is the fixed point of $\Pi T^{(\lambda)}$.

- The fixed point Φr_{λ}^{*} depends on λ.

BIAS-VARIANCE TRADEOFF



- Error bound $\|J_\mu - \Phi r_\lambda^*\|_\xi \le \frac{1}{\sqrt{1-\alpha_\lambda^2}} \|J_\mu - \Pi J_\mu\|_\xi$

- As $\lambda \uparrow 1$, we have $\alpha_\lambda \downarrow 0$, so error bound (and the quality of approximation) improves as $\lambda \uparrow 1$. In fact

$$\lim_{\lambda \uparrow 1} \Phi r_\lambda^* = \Pi J_\mu$$

- But the simulation noise in approximating

$$T^{(\lambda)} = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}$$

increases

- Choice of λ is usually based on trial and error

MULTISTEP PROJECTED EQ. METHODS

- The projected Bellman equation is

$$\Phi r = \Pi T^{(\lambda)}(\Phi r)$$

- In matrix form: $C^{(\lambda)} r = d^{(\lambda)}$, where

$$C^{(\lambda)} = \Phi' \Xi \big(I - \alpha P^{(\lambda)}\big)\Phi, \qquad d^{(\lambda)} = \Phi' \Xi g^{(\lambda)},$$

with

$$P^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \alpha^\ell \lambda^\ell P^{\ell+1}, \quad g^{(\lambda)} = \sum_{\ell=0}^{\infty} \alpha^\ell \lambda^\ell P^\ell g$$

- The LSTD(λ) method is

$$\big(C_k^{(\lambda)}\big)^{-1} d_k^{(\lambda)},$$

where $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ are simulation-based approximations of $C^{(\lambda)}$ and $d^{(\lambda)}$.

- The LSPE(λ) method is

$$r_{k+1} = r_k - \gamma G_k \big(C_k^{(\lambda)} r_k - d_k^{(\lambda)}\big)$$

where G_k is a simulation-based approx. to $(\Phi' \Xi \Phi)^{-1}$

- TD(λ): An important simpler/slower iteration [similar to LSPE(λ) with $G_k = I$ - see the text].

MORE ON MULTISTEP METHODS

- The simulation process to obtain $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ is similar to the case $\lambda = 0$ (single simulation trajectory $i_0, i_1, \ldots$ more complex formulas)

$$C_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \sum_{m=t}^{k} \alpha^{m-t} \lambda^{m-t} \big(\phi(i_m) - \alpha\phi(i_{m+1})\big)',$$

$$d_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \sum_{m=t}^{k} \alpha^{m-t} \lambda^{m-t} g_{i_m}$$

- In the context of approximate policy iteration, we can use optimistic versions (few samples between policy updates).
- Many different versions (see the text).
- Note the λ-tradeoffs:
  - As $\lambda \uparrow 1$, $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ contain more "simulation noise", so more samples are needed for a close approximation of r_λ (the solution of the projected equation)
  - The error bound $\|J_\mu - \Phi r_\lambda\|_\xi$ becomes smaller
  - As $\lambda \uparrow 1$,$\Pi\, T^{(\lambda)}$ becomes a contraction for arbitrary projection norm

6.231 DYNAMIC PROGRAMMING

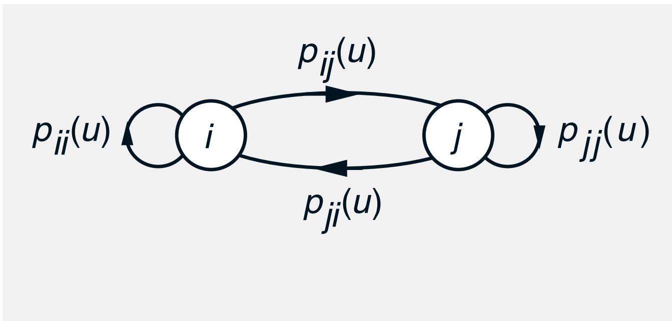
LECTURE 5

LECTURE OUTLINE

- Review of approximate PI
- Review of approximate policy evaluation based on projected Bellman equations
- Exploration enhancement in policy evaluation
- Oscillations in approximate PI
- Aggregation – An alternative to the projected equation/Galerkin approach
- Examples of aggregation
- Simulation-based aggregation

DISCOUNTED MDP

- System: Controlled Markov chain with states $i = 1, \ldots, n$ and finite set of controls $u \in U(i)$
- Transition probabilities: $p_{ij}(u)$



- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ starting at state i:

$$J_\pi(i) = \lim_{N\to\infty} E\left\{\sum_{k=0}^{N} \alpha^k g\big(i_k, \mu_k(i_k), i_{k+1}\big) \mid i = i_0\right\}$$

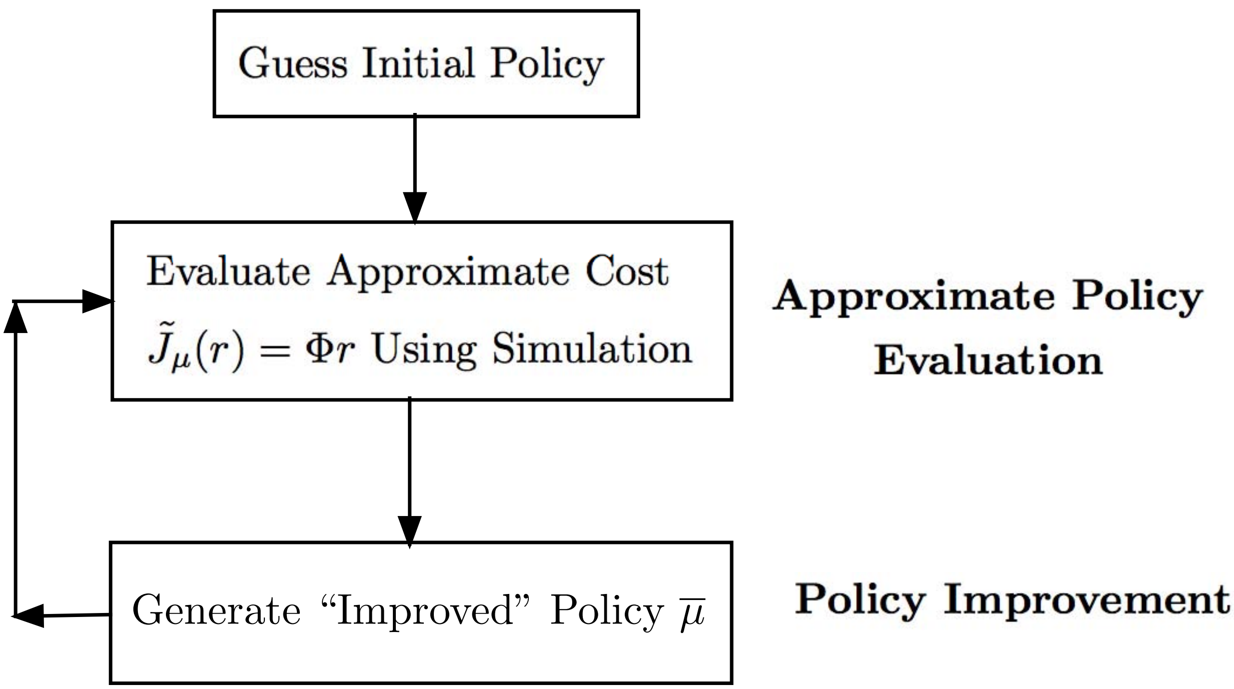
with $\alpha \in [0, 1)$

- Shorthand notation for DP mappings

$$(TJ)(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)+\alpha J(j)\big), \quad i = 1, \ldots, n,$$

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g\big(i,\mu(i),j\big)+\alpha J(j)\Big), \quad i = 1, \ldots, n$$

APPROXIMATE PI



- Evaluation of typical policy μ: Linear cost function approximation

$$\tilde{J}_\mu(r) = \Phi r$$

where Φ is full rank $n \times s$ matrix with columns the basis functions, and ith row denoted $\phi(i)'$.

- Policy "improvement" to generate $\overline{\mu}$:

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \big(g(i,u,j) + \alpha \phi(j)' r \big)$$

EVALUATION BY PROJECTED EQUATIONS

- We discussed approximate policy evaluation by solving the projected equation

$$\Phi r = \Pi T_\mu(\Phi r)$$

Π: projection with a weighted Euclidean norm

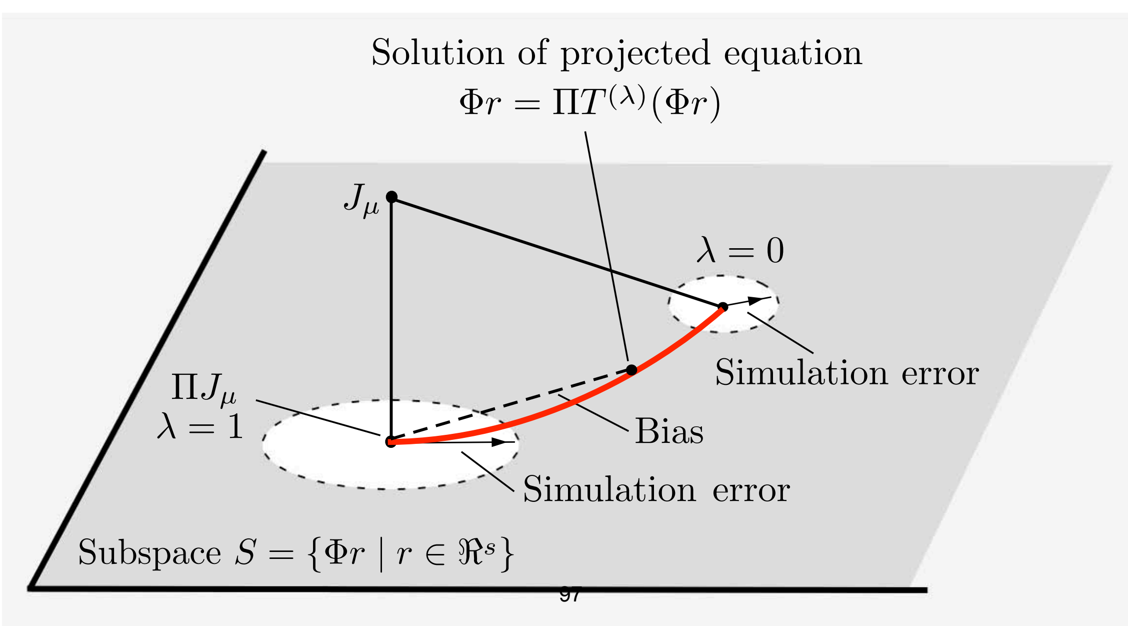
- Implementation by simulation (single long trajectory using current policy - important to make ΠT_μ a contraction). LSTD, LSPE methods.

- Multistep option: SolveΦ $r = \Pi T_\mu^{(\lambda)}(\Phi r)$ with

$$T_\mu^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^\ell T_\mu^{\ell+1}$$

  – As $\lambda \uparrow 1$, $\Pi T^{(\lambda)}$ becomes a contraction for any projection norm

  – Bias-variance tradeoff



POLICY ITERATION ISSUES: EXPLORATION

- **1st major issue:** exploration. To evaluate μ, we need to generate cost samples using μ

- This biases the simulation by underrepresenting states that are unlikely to occur under μ.

- As a result, the cost-to-go estimates of these underrepresented states may be highly inaccurate.

- This seriously impacts the improved policy $\overline{\mu}$.

- This is known as inadequate exploration - a particularly acute difficulty when the randomness embodied in the transition probabilities is "relatively small" (e.g., a deterministic system).

- Common remedy is the off-policy approach: Replace P of current policy with a "mixture"

$$\overline{P} = (I - B)P + BQ$$

where B is diagonal with diagonal components in $[0, 1]$ and Q is another transition matrix.

- LSTD and LSPE formulas must be modified ... otherwise the policy $\overline{P}$ (not P) is evaluated. Related methods and ideas: importance sampling, geometric and free-form sampling (see the text).

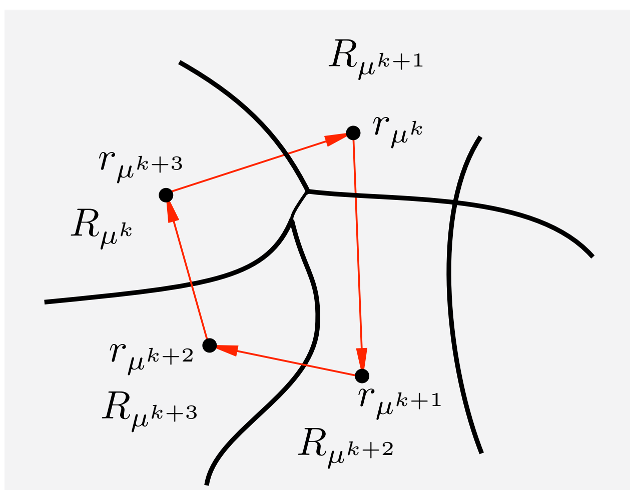
POLICY ITERATION ISSUES: OSCILLATIONS

- **2nd major issue:** oscillation of policies

- Analysis using the greedy partition: R_μ is the set of parameter vectors r for which μ is greedy with respect to $\tilde{J}(\cdot, r) = \Phi r$

$$R_\mu = \big\{ r \mid T_\mu(\Phi r) = T(\Phi r) \big\}$$

- There is a finite number of possible vectors r_μ, one generated from another in a deterministic way



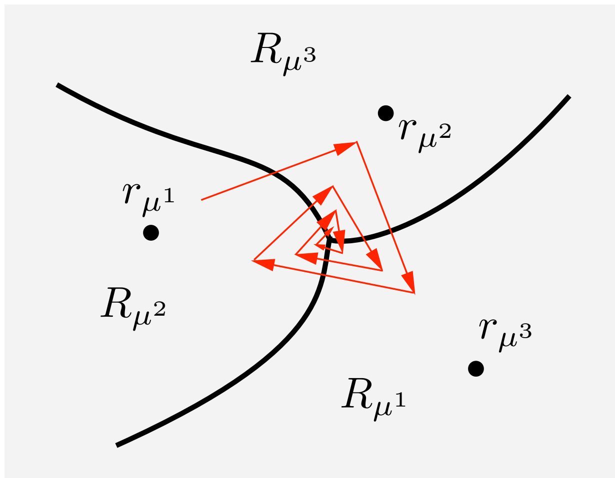
- The algorithm ends up repeating some cycle of policies $\mu^k, \mu^{k+1}, \ldots, \mu^{k+m}$ with

$$r_{\mu^k} \in R_{\mu^{k+1}},\, r_{\mu^{k+1}} \in R_{\mu^{k+2}}, \ldots, r_{\mu^{k+m}} \in R_{\mu^k};$$

- Many different cycles are possible

MORE ON OSCILLATIONS/CHATTERING

- In the case of optimistic policy iteration a different picture holds



- Oscillations are less violent, but the "limit" point is meaningless!
- Fundamentally, oscillations are due to the lack of monotonicity of the projection operator, i.e., $J \leq J'$ does not imply $\Pi J \leq \Pi J'$.
- If approximate PI uses policy evaluation

$$\Phi r = (W T_\mu)(\Phi r)$$

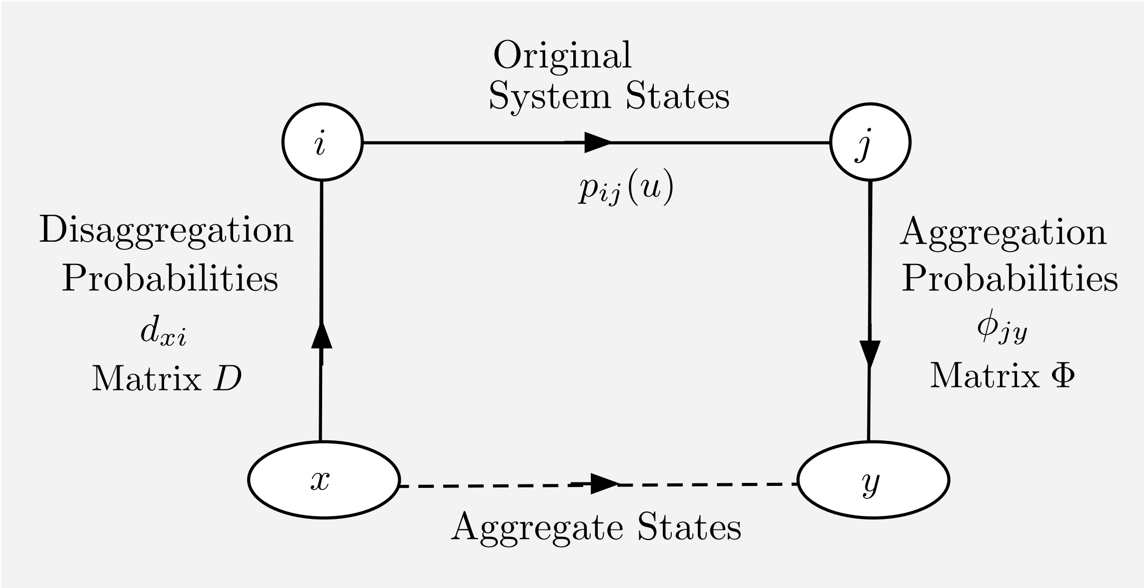
with W a monotone operator, the generated policies converge (to a possibly nonoptimal limit).

- The operator W used in the aggregation approach has this monotonicity property.

PROBLEM APPROXIMATION - AGGREGATION

- Another major idea in ADP is to approximate the cost-to-go function of the problem with the cost-to-go function of a simpler problem.
- The simplification is often ad-hoc/problem-dependent.
- Aggregation is a systematic approach for problem approximation. Main elements:
  - Introduce a few "aggregate" states, viewed as the states of an "aggregate" system
  - Define transition probabilities and costs of the aggregate system, by relating original system states with aggregate states
  - Solve (exactly or approximately) the "aggregate" problem by any kind of VI or PI method (including simulation-based methods)
  - Use the optimal cost of the aggregate problem to approximate the optimal cost of the original problem
- Hard aggregation example: Aggregate states are subsets of original system states, treated as if they all have the same cost.

AGGREGATION/DISAGGREGATION PROBS



- The aggregate system transition probabilities are defined via two (somewhat arbitrary) choices
- For each original system state j and aggregate state y, the aggregation probability ϕ_{jy}
  - Roughly, the "degree of membership of j in the aggregate state y."
  - In hard aggregation, $\phi_{jy} = 1$ if state j belongs to aggregate state/subset y.
- For each aggregate state x and original system state i, the disaggregation probability d_{xi}
  - Roughly, the "degree to which i is representative of x."
  - In hard aggregation, equal d_{xi}

AGGREGATE SYSTEM DESCRIPTION

- The transition probability from aggregate state x to aggregate state y under control u

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}, \quad \text{or } \hat{P}(u) = DP(u)\Phi$$

where the rows of D and Φ are the disaggregation and aggregation probs.

- The expected transition cost is

$$\hat{g}(x,u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)g(i,u,j), \quad \text{or } \hat{g} = DPg$$

- The optimal cost function of the aggregate problem, denoted $\hat{R}$, is

$$\hat{R}(x) = \min_{u\in U}\left[\hat{g}(x,u) + \alpha\sum_{y}\hat{p}_{xy}(u)\hat{R}(y)\right], \qquad \forall\, x$$

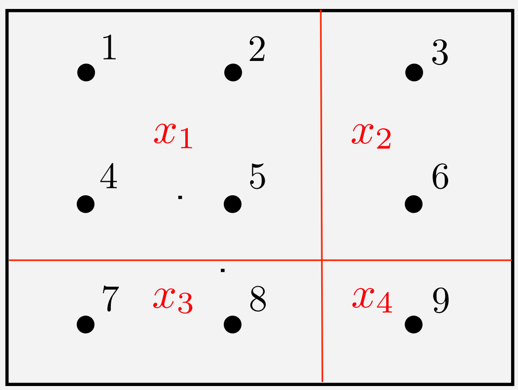
Bellman's equation for the aggregate problem.

- The optimal cost function J^* of the original problem is approximated by $\tilde{J}$ given by

$$\tilde{J}(j) = \sum_{y}\phi_{jy}\hat{R}(y), \qquad \forall\, j$$

EXAMPLE I: HARD AGGREGATION

- Group the original system states into subsets, and view each subset as an aggregate state
- Aggregation probs.: $\phi_{jy} = 1$ if j belongs to aggregate state y.



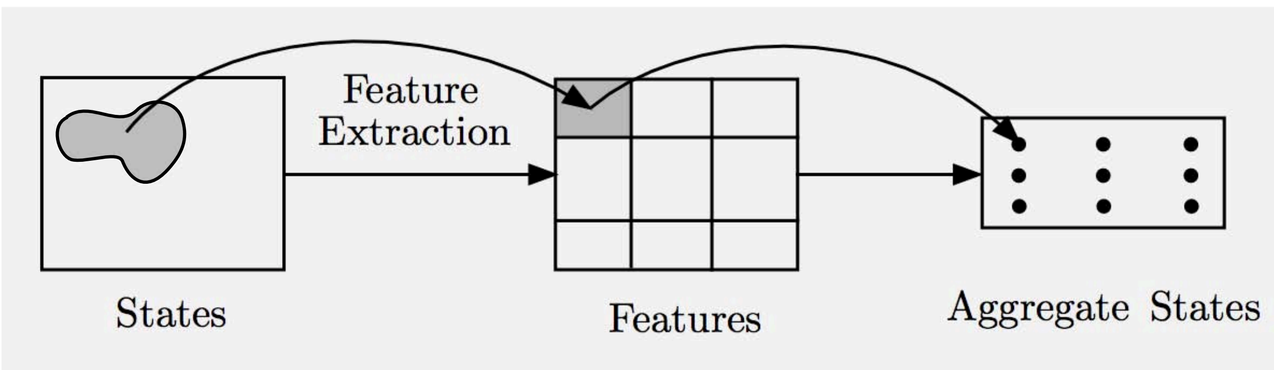
$$\Phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Disaggregation probs.: There are many possibilities, e.g., all states i within aggregate state x have equal prob. d_{xi}.
- If optimal cost vector J^* is piecewise constant over the aggregate states/subsets, hard aggregation is exact. Suggests grouping states with “roughly equal” cost into aggregates.
- A variant: Soft aggregation (provides “soft boundaries” between aggregate states).

EXAMPLE II: FEATURE-BASED AGGREGATION

- Important question: How do we group states together?

- If we know good features, it makes sense to group together states that have "similar features"



- A general approach for passing from a feature-based state representation to an aggregation-based architecture

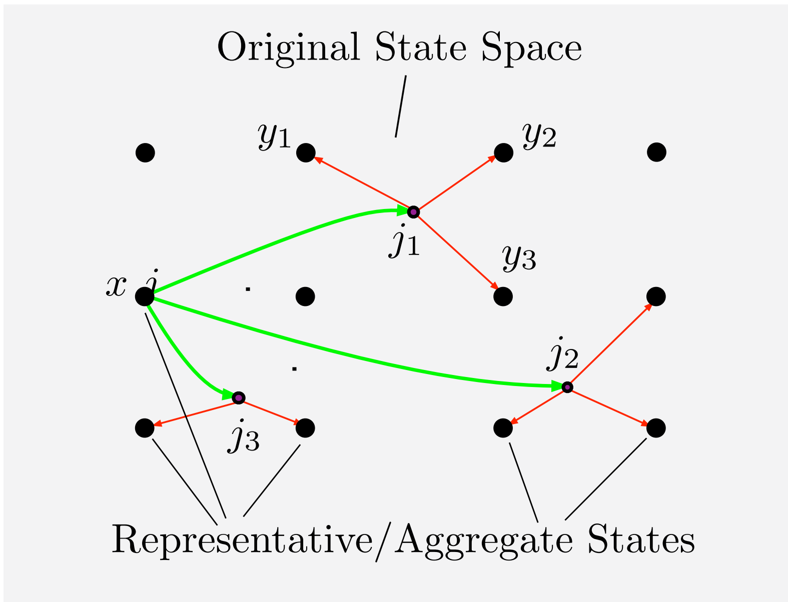
- Essentially discretize the features and generate a corresponding piecewise constant approximation to the optimal cost function

- Aggregation-based architecture is more powerful (nonlinear in the features)

- ... but may require many more aggregate states to reach the same level of performance as the corresponding linear feature-based architecture

EXAMPLE III: REP. STATES/COARSE GRID

- Choose a collection of "representative" original system states, and associate each one of them with an aggregate state



- Disaggregation probabilities are $d_{xi} = 1$ if i is equal to representative state x.

- Aggregation probabilities associate original system states with convex combinations of representative states

$$j \sim \sum_{y \in \mathcal{A}} \phi_{jy} y$$

- Well-suited for Euclidean space discretization

- Extends nicely to continuous state space, including belief space of POMDP

EXAMPLE IV: REPRESENTATIVE FEATURES

- Here the aggregate states are nonempty subsets of original system states (but need not form a partition of the state space)

- Example: Choose a collection of distinct "representative" feature vectors, and associate each of them with an aggregate state consisting of original system states with similar features

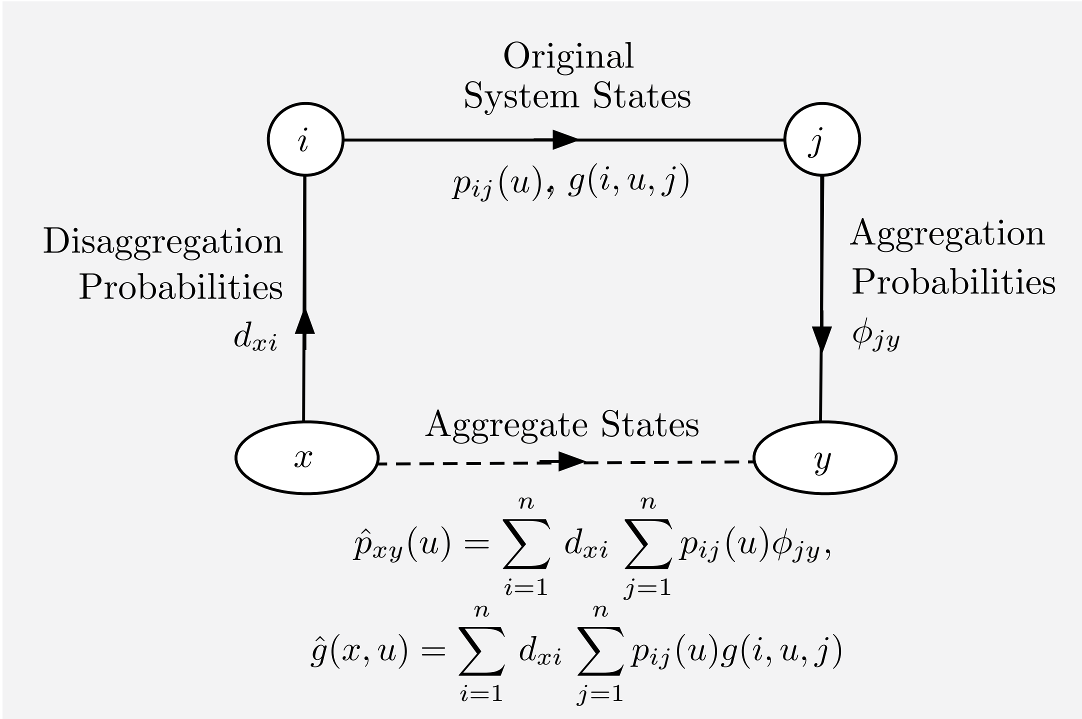
- Restrictions:
  - The aggregate states/subsets are disjoint.
  - The disaggregation probabilities satisfy $d_{xi} > 0$ if and only if $i \in x$.
  - The aggregation probabilities satisfy $\phi_{jy} = 1$ for all $j \in y$.

- If every original system state i belongs to some aggregate state we obtain hard aggregation

- If every aggregate state consists of a single original system state, we obtain aggregation with representative states

- With the above restrictions $D\Phi = I$, so $(\Phi D)(\Phi D) = \Phi D$, and ΦD is an oblique projection (orthogonal projection in case of hard aggregation)

APPROXIMATE PI BY AGGREGATION



- Consider approximate policy iteration for the original problem, with policy evaluation done by aggregation.

- Evaluation of policy μ: $\tilde{J} = \Phi R$, where $R = DT_\mu(\Phi R)$ (R is the vector of costs of aggregate states for μ). Can be done by simulation.

- Looks like projected equation $\Phi R = \Pi T_\mu(\Phi R)$ (but with ΦD in place of Π).

- Advantages: It has no problem with exploration or with oscillations.

- Disadvantage: The rows of D and Φ must be probability distributions.

DISTRIBUTED AGGREGATION I

- We consider decomposition/distributed solution of large-scale discounted DP problems by aggregation.
- Partition the original system states into subsets $S_1, \ldots, S_m$
- Each subset S_ℓ, $\ell = 1, \ldots, m$:
  - Maintains detailed/exact local costs

    $$J(i) \quad \text{for every original system state } i \in S_\ell$$

    using aggregate costs of other subsets
  - Maintains an aggregate cost $R(\ell) = \sum_{i \in S_\ell} d_{\ell i} J(i)$
  - Sends $R(\ell)$ to other aggregate states
- $J(i)$ and $R(\ell)$ are updated by VI according to

$$J_{k+1}(i) = \min_{u \in U(i)} H_\ell(i, u, J_k, R_k), \qquad \forall\, i \in S_\ell$$

with R_k being the vector of $R(\ell)$ at time k, and

$$H_\ell(i, u, J, R) = \sum_{j=1}^{n} p_{ij}(u) g(i, u, j) + \alpha \sum_{j \in S_\ell} p_{ij}(u) J(j) + \alpha \sum_{j \in S_{\ell'},\, \ell' \neq \ell} p_{ij}(u) R(\ell')$$

DISTRIBUTED AGGREGATION II

- Can show that this iteration involves a sup-norm contraction mapping of modulus α, so it converges to the unique solution of the system of equations in (J, R)

$$J(i) = \min_{u \in U(i)} H_\ell(i, u, J, R), \quad R(\ell) = \sum_{i \in S_\ell} d_{\ell i} J(i),$$
$$\forall\ i \in S_\ell,\ \ell = 1, \ldots, m.$$

- This follows from the fact that $\{d_{\ell i} \mid i = 1, \ldots, n\}$ is a probability distribution.
- View these equations as a set of Bellman equations for an "aggregate" DP problem. The difference is that the mapping H involves $J(j)$ rather than $R\big(x(j)\big)$ for $j \in S_\ell$.
- In an asynchronous version of the method, the aggregate costs $R(\ell)$ may be outdated to account for communication "delays" between aggregate states.
- Convergence can be shown using the general theory of asynchronous distributed computation (see the text).

6.231 DYNAMIC PROGRAMMING

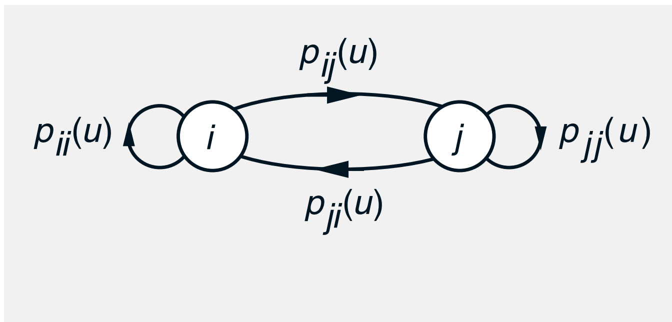
LECTURE 6

LECTURE OUTLINE

- Review of Q-factors and Bellman equations for Q-factors
- VI and PI for Q-factors
- Q-learning - Combination of VI and sampling
- Q-learning and cost function approximation
- Approximation in policy space

DISCOUNTED MDP

- System: Controlled Markov chain with states $i = 1, \ldots, n$ and finite set of controls $u \in U(i)$
- Transition probabilities: $p_{ij}(u)$



- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ starting at state i:

$$J_\pi(i) = \lim_{N \to \infty} E\left\{ \sum_{k=0}^{N} \alpha^k g\big(i_k, \mu_k(i_k), i_{k+1}\big) \mid i = i_0 \right\}$$

with $\alpha \in [0, 1)$

- Shorthand notation for DP mappings

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J(j)\big), \quad i = 1, \ldots, n,$$

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g\big(i, \mu(i), j\big) + \alpha J(j)\Big), \quad i = 1, \ldots, n$$

THE TWO MAIN ALGORITHMS: VI AND PI

- Value iteration: For any $J \in \Re^n$

$$J^*(i) = \lim_{k\to\infty} (T^k J)(i), \qquad \forall\ i = 1,\ldots,n$$

- Policy iteration: Given μ^k
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big)\Big(g\big(i,\mu^k(i),j\big)+\alpha J_{\mu^k}(j)\Big),\ \ i = 1,\ldots,n$$

  or $J_{\mu^k} = T_{\mu^k} J_{\mu^k}$

  - Policy improvement: Let μ^{k+1} be such that

$$\mu^{k+1}(i) \in \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)+\alpha J_{\mu^k}(j)\big),\quad \forall\, i$$

  or $T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$

- We discussed approximate versions of VI and PI using projection and aggregation

- We focused so far on cost functions and approximation. We now consider Q-factors.

BELLMAN EQUATIONS FOR Q-FACTORS

- The optimal Q-factors are defined by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \quad \forall\ (i,u)$$

- Since $J^* = TJ^*$, we have $J^*(i) = \min_{u\in U(i)} Q^*(i,u)$ so the optimal Q-factors solve the equation

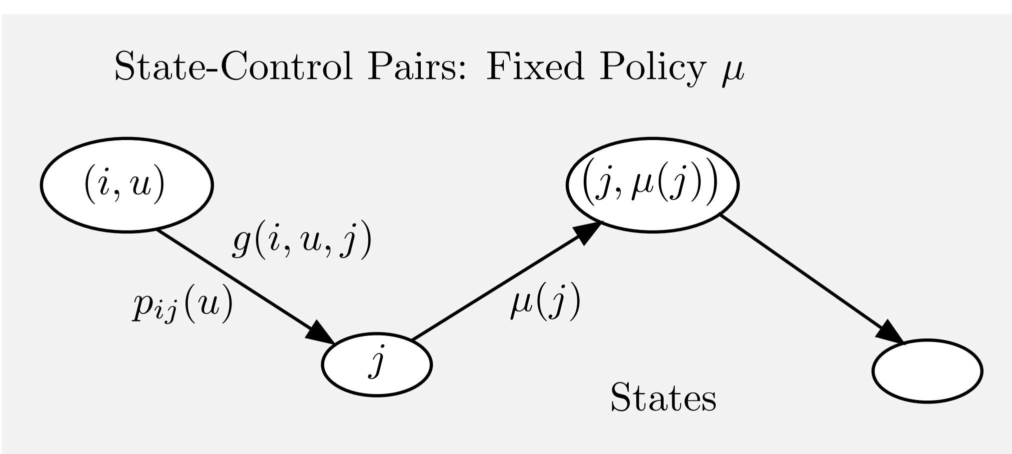
$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{u'\in U(j)} Q^*(j,u')\right)$$

- Equivalently $Q^* = FQ^*$, where

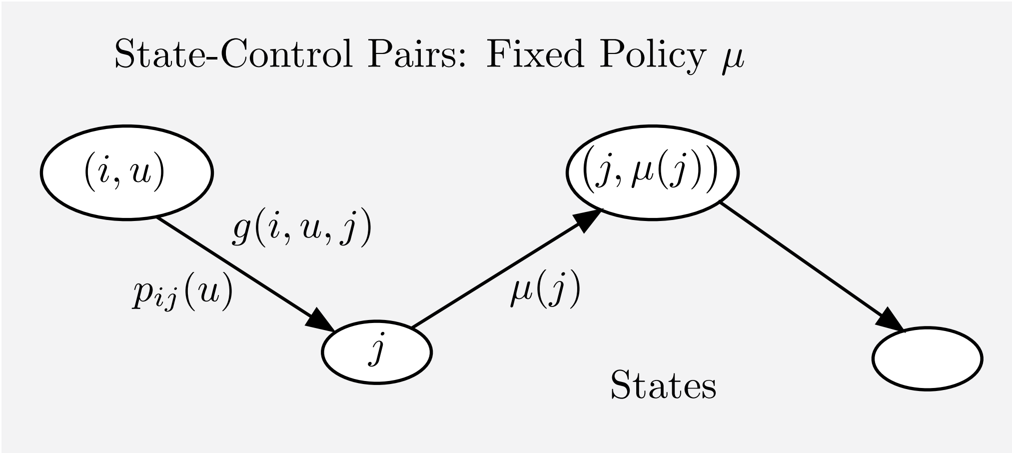
$$(FQ)(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{u'\in U(j)} Q(j,u')\right)$$

- This is Bellman's Eq. for a system whose states are the pairs (i,u)

- Similar mapping F_μ and Bellman equation for a policy μ: $Q_\mu = F_\mu Q_\mu$



SUMMARY OF BELLMAN EQS FOR Q-FACTORS



- Optimal Q-factors: For all (i,u)

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \min_{u' \in U(j)} Q^*(j,u') \right)$$

Equivalently $Q^* = FQ^*$, where

$$(FQ)(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \min_{u' \in U(j)} Q(j,u') \right)$$

- Q-factors of a policy μ: For all (i,u)

$$Q_\mu(i,u) = \sum_{j=1}^{n} p_{ij}(u) \Big(g(i,u,j) + \alpha Q_\mu\big(j,\mu(j)\big) \Big)$$

Equivalently $Q_\mu = F_\mu Q_\mu$, where

$$(F_\mu Q)(i,u) = \sum_{j=1}^{n} p_{ij}(u) \Big(g(i,u,j) + \alpha Q\big(j,\mu(j)\big) \Big)$$

WHAT IS GOOD AND BAD ABOUT Q-FACTORS

- All the exact theory and algorithms for costs applies to Q-factors
  - Bellman's equations, contractions, optimality conditions, convergence of VI and PI
- All the approximate theory and algorithms for costs applies to Q-factors
  - Projected equations, sampling and exploration issues, oscillations, aggregation
- A MODEL-FREE (on-line) controller implementation
  - Once we calculate $Q^*(i,u)$ for all (i,u),

$$\mu^*(i) = \arg\min_{u \in U(i)} Q^*(i,u), \qquad \forall\ i$$

  - Similarly, once we calculate a parametric approximation $\tilde{Q}(i,u,r)$ for all (i,u),

$$\tilde{\mu}(i) = \arg\min_{u \in U(i)} \tilde{Q}(i,u,r), \qquad \forall\ i$$

- The main bad thing: Greater dimension and more storage! [Can be used for large-scale problems only through aggregation, or other cost function approximation.]

Q-LEARNING

- In addition to the approximate PI methods adapted for Q-factors, there is an important additional algorithm:
  - Q-learning, which can be viewed as a sampled form of VI
- Q-learning algorithm (in its classical form):
  - Sampling: Select sequence of pairs (i_k, u_k) (use any probabilistic mechanism for this, but all pairs (i, u) are chosen infinitely often.)
  - Iteration: For each k, select j_k according to $p_{i_k j}(u_k)$. Update just $Q(i_k, u_k)$:

$$Q_{k+1}(i_k,u_k) = (1 - \gamma_k)Q_k(i_k, u_k) + \gamma_k \left(g(i_k, u_k, j_k) + \alpha \min_{u' \in U(j_k)} Q_k(j_k, u') \right)$$

    Leave unchanged all other Q-factors: $Q_{k+1}(i, u) = Q_k(i, u)$ for all $(i, u) \neq (i_k, u_k)$.
  - Stepsize conditions: γ_k must converge to 0 at proper rate (e.g., like $1/k$).

NOTES AND QUESTIONS ABOUT Q-LEARNING

$$Q_{k+1}(i_k,u_k) = (1-\gamma_k)Q_k(i_k,u_k) + \gamma_k\left(g(i_k,u_k,j_k) + \alpha \min_{u' \in U(j_k)} Q_k(j_k,u')\right)$$

- Model free implementation. We just need a simulator that given (i,u) produces next state j and cost $g(i,u,j)$
- Operates on only one state-control pair at a time. Convenient for simulation, no restrictions on sampling method.
- Aims to find the (exactly) optimal Q-factors.
- Why does it converge to Q^*?
- Why can't I use a similar algorithm for optimal costs?
- Important mathematical (fine) point: In the Q-factor version of Bellman's equation the order of expectation and minimization is reversed relative to the cost version of Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big)$$

CONVERGENCE ASPECTS OF Q-LEARNING

- Q-learning can be shown to converge to true/exact Q-factors (under mild assumptions).

- Proof is sophisticated, based on theories of stochastic approximation and asynchronous algorithms.

- Uses the fact that the Q-learning map F:

$$(FQ)(i,u) = E_j\Big\{g(i,u,j) + \alpha \min_{u'} Q(j,u')\Big\}$$

is a sup-norm contraction.

- Generic stochastic approximation algorithm:
  - Consider generic fixed point problem involving an expectation:

$$x = E_w\big\{f(x,w)\big\}$$

  - Assume $E_w\big\{f(x,w)\big\}$ is a contraction with respect to some norm, so the iteration

$$x_{k+1} = E_w\big\{f(x_k,w)\big\}$$

    converges to the unique fixed point
  - Approximate $E_w\big\{f(x,w)\big\}$ by sampling

STOCH. APPROX. CONVERGENCE IDEAS

- For each k, obtain samples $\{w_1, \ldots, w_k\}$ and use the approximation

$$x_{k+1} = \frac{1}{k} \sum_{t=1}^{k} f(x_k, w_t) \approx E\big\{f(x_k, w)\big\}$$

- This iteration approximates the convergent fixed point iteration $x_{k+1} = E_w\big\{f(x_k, w)\big\}$
- A major flaw: it requires, for each k, the computation of $f(x_k, w_t)$ for all values w_t, $t = 1, \ldots, k$.
- This motivates the more convenient iteration

$$x_{k+1} = \frac{1}{k} \sum_{t=1}^{k} f(x_t, w_t), \qquad k = 1, 2, \ldots,$$

that is similar, but requires much less computation; it needs only one value of f per sample w_t.

- By denoting $\gamma_k = 1/k$, it can also be written as

$$x_{k+1} = (1 - \gamma_k) x_k + \gamma_k f(x_k, w_k), \quad k = 1, 2, \ldots$$

- Compare with Q-learning, where the fixed point problem is $Q = FQ$

$$(FQ)(i, u) = E_j\Big\{g(i, u, j) + \alpha \min_{u'} Q(j, u')\Big\}$$

Q-FACTOR APROXIMATIONS

- We introduce basis function approximation:

$$\tilde{Q}(i,u,r) = \phi(i,u)'r$$

- We can use approximate policy iteration and LSPE/LSTD for policy evaluation

- Optimistic policy iteration methods are frequently used on a heuristic basis

- Example: Generate trajectory $\{(i_k, u_k) \mid k = 0, 1, \ldots\}$.

- At iteration k, given r_k and state/control (i_k, u_k):

  (1) Simulate next transition (i_k, i_{k+1}) using the transition probabilities $p_{i_k j}(u_k)$.

  (2) Generate control u_{k+1} from

$$u_{k+1} = \arg \min_{u \in U(i_{k+1})} \tilde{Q}(i_{k+1}, u, r_k)$$

  (3) Update the parameter vector via

$$r_{k+1} = r_k - (\text{LSPE or TD-like correction})$$

- Complex behavior, unclear validity (oscillations, etc). There is solid basis for an important special case: optimal stopping (see text)

APPROXIMATION IN POLICY SPACE

- We parameterize policies by a vector $r = (r_1, \ldots, r_s)$ (an approximation architecture for policies).

- Each policy $\tilde{\mu}(r) = \{\tilde{\mu}(i; r) \mid i = 1, \ldots, n\}$ defines a cost vector $J_{\tilde{\mu}(r)}$ (a function of r).

- We optimize some measure of $J_{\tilde{\mu}(r)}$ over r.

- For example, use a random search, gradient, or other method to minimize over r

$$\sum_{i=1}^{n} p_i J_{\tilde{\mu}(r)}(i),$$

where $(p_1, \ldots, p_n)$ is some probability distribution over the states.

- An important special case: Introduce cost approximation architecture $V(i, r)$ that defines indirectly the parameterization of the policies

$$\tilde{\mu}(i; r) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \big(g(i, u, j) + \alpha V(j, r) \big), \; \forall\, i$$

- Brings in features to approximation in policy space

APPROXIMATION IN POLICY SPACE METHODS

- Random search methods are straightforward and have scored some impressive successes with challenging problems (e.g., tetris).

- Gradient-type methods (known as policy gradient methods) also have been worked on extensively.

- They move along the gradient with respect to r of

$$\sum_{i=1}^{n} p_i J_{\tilde{\mu}(r)}(i),$$

- There are explicit gradient formulas which have been approximated by simulation

- Policy gradient methods generally suffer by slow convergence, local minima, and excessive simulation noise

FINAL WORDS AND COMPARISONS

- There is no clear winner among ADP methods
- There is interesting theory in all types of methods (which, however, does not provide ironclad performance guarantees)
- There are major flaws in all methods:
  - Oscillations and exploration issues in approximate PI with projected equations
  - Restrictions on the approximation architecture in approximate PI with aggregation
  - Flakiness of optimization in policy space approximation
- Yet these methods have impressive successes to show with enormously complex problems, for which there is no alternative methodology
- There are also other competing ADP methods (rollout is simple, often successful, and generally reliable; approximate LP is worth considering)
- Theoretical understanding is important and nontrivial
- Practice is an art and a challenge to our creativity!