Neuro-Dynamic Programming
An Overview

Dimitri Bertsekas
Dept. of Electrical Engineering
and Computer Science
M.I.T.

September 2006

BELLMAN AND THE DUAL CURSES

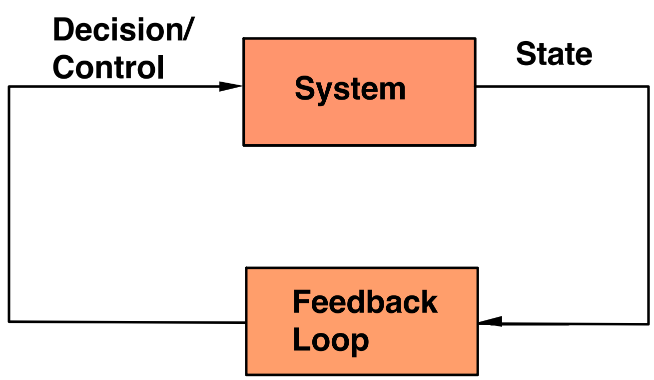
- Dynamic Programming (DP) is very broadly applicable, but it suffers from:
  - Curse of dimensionality
  - Curse of modeling
- We address “complexity” by using low-dimensional parametric approximations
- We allow simulators in place of models
- Unlimited applications in planning, resource allocation, stochastic control, discrete optimization
- Application is an art … but guided by substantial theory

OUTLINE

- Main NDP framework
- Discussion of two classes of methods, based on approximate value/policy iteration:
  - Actor-critic methods/LSPE
  - Rollout algorithms
- Additional classes of methods (not discussed): approximate linear programming, approximation in policy space
- References:
  - Neuro-Dynamic Programming (1996, Bertsekas + Tsitsiklis)
  - Reinforcement Learning (1998, Sutton + Barto)
  - Dynamic Programming: 3rd Edition (2006, Bertsekas)
  - Recent papers with V. Borkar, A. Nedic, and J. Yu
- Papers and this talk can be downloaded from http://web.mit.edu/dimitrib/www/home.html

DYNAMIC PROGRAMMING / DECISION AND CONTROL

- **Main ingredients:**
  - Dynamic system; state evolving in discrete time
  - Decision/control applied at each time
  - Cost is incurred at each time
  - There may be noise & model uncertainty
  - There is state feedback used to determine the control



APPLICATIONS

- Extremely broad range
- Sequential decision contexts
  - Planning (shortest paths, schedules, route planning, supply chain)
  - Resource allocation over time (maintenance, power generation)
  - Finance (investment over time, optimal stopping/option valuation)
  - Automatic control (vehicles, machines)
- Nonsequential decision contexts
  - Combinatorial/discrete optimization (breakdown solution into stages)
  - Branch and Bound/ Integer programming
- Applies to both deterministic and stochastic problems

ESSENTIAL TRADEOFF CAPTURED BY DP

- Decisions are made in stages
- The decision at each stage:
  - Determines the present stage cost
  - Affects the context within which future decisions are made
- At each stage we must trade:
  - Low present stage cost
  - Undesirability of high future costs

KEY DP RESULT: BELLMAN'S EQUATION

- Optimal decision at the current state minimizes the expected value of

  Current stage cost

  + Future stages cost

  (starting from the next state

  - using opt. policy)
- Extensive mathematical methodology
- Applies to both discrete and continuous systems (and hybrids)
- Dual curses of dimensionality/modeling

KEY NDP IDEA

- Use one-step lookahead with an "approximate" cost
- At the current state select decision that minimizes the expected value of

  Current stage cost

  + Approximate future stages cost

  (starting from the next state)

- Important issues:
  - How to construct the approximate cost of a state
  - How to understand and control the effects of approximation

METHODS TO COMPUTE AN APPROXIMATE COST

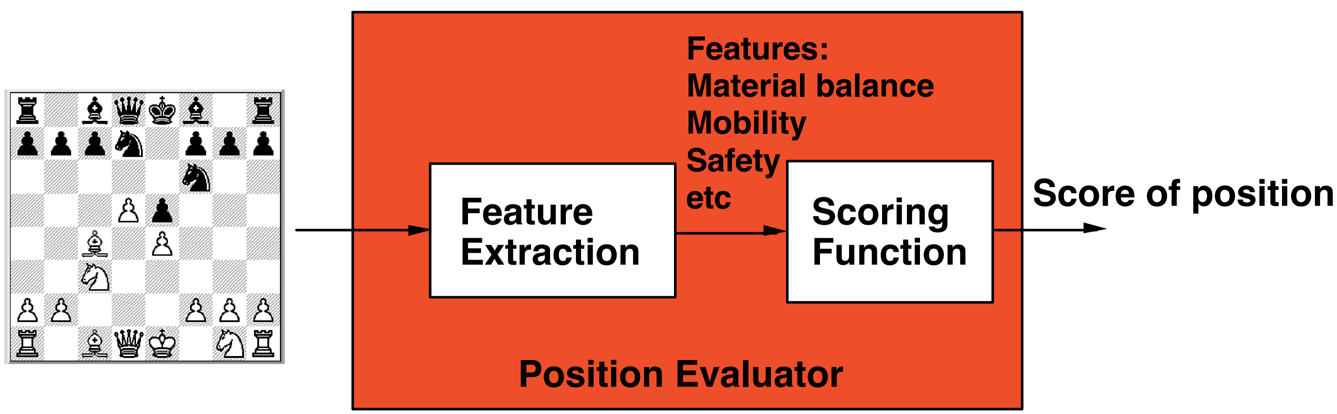
- Parametric approximation algorithms
  - Use a functional approximation to the optimal cost; e.g., linear combination of basis functions
  - Select the weights of the approximation
  - One possibility: Hand-tuning, and trial and error
  - Systematic DP-related policy and value iteration methods (TD-Lambda, Q-Learning, LSPE, LSTD, etc)
- Rollout algorithms
  - Use the cost of the heuristic (or a lower bound) as cost approximation
  - Obtain by simulation this cost, starting from the state of interest

SIMULATION AND LEARNING

- Simulation (learning by experience): used to compute the (approximate) cost-to-go -- a key distinctive aspect of NDP
- Important advantage: detailed system model not necessary - use a simulator instead
- In case of parametric approximation: off-line learning
- In case of a rollout algorithm: on-line learning (we learn only the cost values needed by on-line simulation)

PARAMETRIC APPROXIMATION: CHESS PARADIGM

- Chess playing computer programs
- State = board position
- Score of position: “Important features” appropriately weighted



TRAINING

- In chess: Weights are “hand-tuned”
- In more sophisticated methods: Weights are determined by using simulation-based training algorithms
- Temporal Differences TD(λ), Q-Learning, Least Squares Policy Evaluation LSPE(λ), Least Squares Temporal Differences LSTD(λ), etc
- All of these methods are based on DP ideas of policy iteration and value iteration

POLICY IMPROVEMENT PRINCIPLE

- Given a current policy, define a new policy as follows:

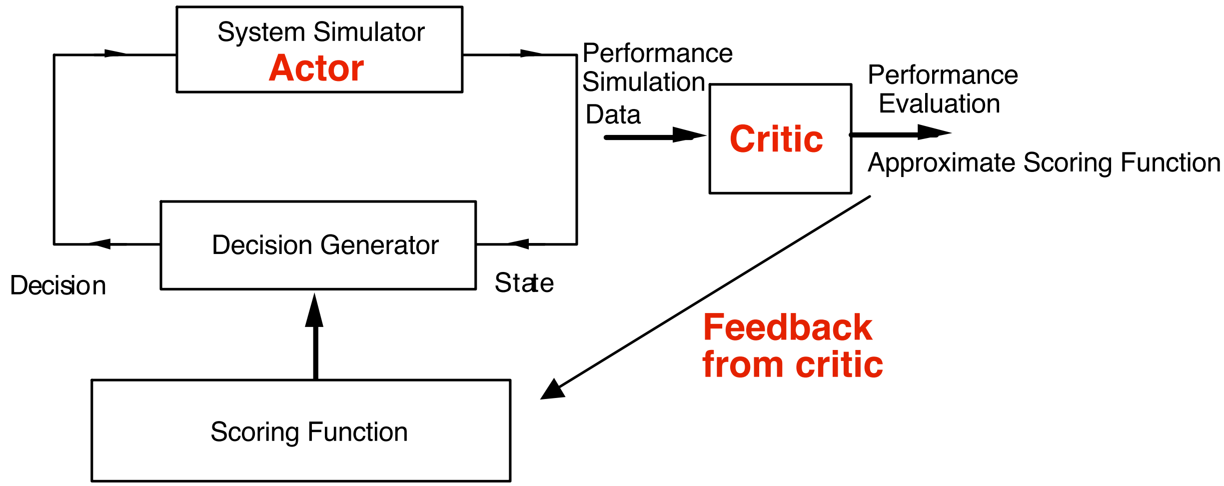
  At each state minimize

  Current stage cost + cost-to-go of current policy (starting from the next state)

- Policy improvement result: New policy has improved performance over current policy
- If the cost-to-go is approximate, the improvement is “approximate”
- Oscillation around the optimal; error bounds

ACTOR/CRITIC SYSTEMS

- Metaphor for policy evaluation/improvement
- Actor implements current policy
- Critic evaluates the performance; passes feedback to the actor
- Actor changes/"improves" policy



APPROXIMATE POLICY EVALUATION

- Consider stationary policy μ w/ cost function J
- Satisfies Bellman's equation:

$$J = T(J) = g_\mu + \alpha P_\mu J \qquad \text{(discounted case)}$$

- Subspace approximation

$$J \sim \Phi r$$

Φ: matrix of basis functions

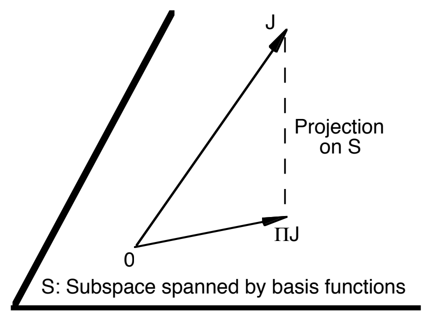
r: parameter vector

DIRECT AND INDIRECT APPROACHES

- **Direct:** **Use simulated cost samples and least-squares fit**

$$J \sim \Pi J$$

Approximate the cost

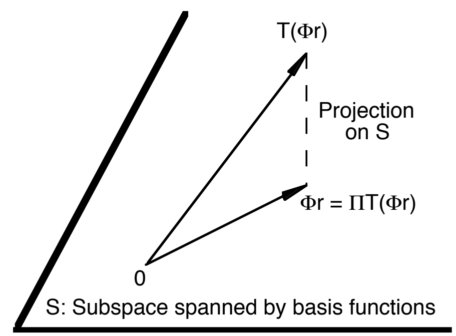


Direct Mehod: Projection of cost vector J

- **Indirect:** **Solve a projected form of Bellman's equation**

$$\Phi r = \Pi T(\Phi r)$$

Approximate the equation



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

DIRECT APPROACH

- Minimize over r; least squares

$$\Sigma \big(\text{Simulated cost sample of } J(i) - (\Phi r)_i \big)^2$$

- Each state is weighted proportionally to its appearance in the simulation
- Works even with nonlinear function approximation (in place of Φr)
- Gradient or special least squares methods can be used
- Problem with large error variance

INDIRECT POLICY EVALUATION

- The most popular simulation-based methods solve the Projected Bellman Equation (PBE)
- TD(λ): (Sutton 1988) - stochastic approximation method
- LSTD(λ): (Barto & Bradtke 1996, Boyan 2002) - solves by matrix inversion a simulation generated approximation to PBE, optimal convergence rate (Konda 2002)
- LSPE(λ): (Bertsekas, Ioffe 1996, Borkar, Nedic 2004, Yu 2006) - uses projected value iteration to find fixed point of PBE
- We will focus now on LSPE

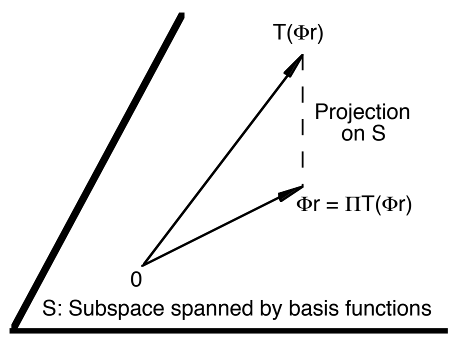
LEAST SQUARES POLICY EVALUATION (LSPE)

- Consider α-discounted Markov Decision Problem (finite state and control spaces)
- We want to approximate the solution of Bellman equation:

$$J = T(J) = g_\mu + \alpha P_\mu J$$

- It solves the projected Bellman equation

$$\Phi r = \Pi T(\Phi r)$$

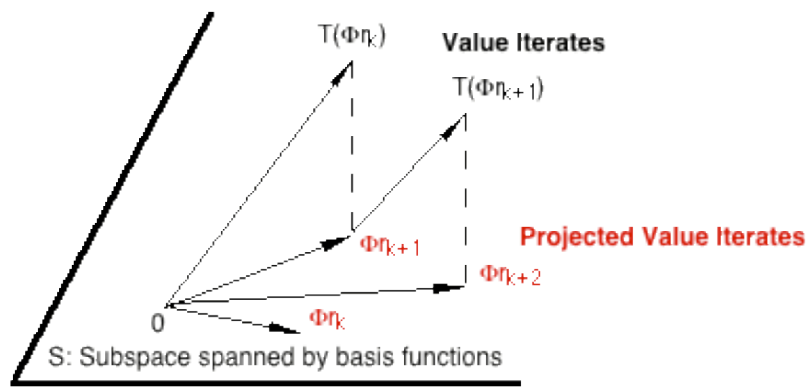


Indirect method: Solving a projected form of Bellman's equation

PROJECTED VALUE ITERATION

- **Value iteration:** $J_{t+1} = T(J_t)$
- **Projected Value iteration:** $\Phi r_{t+1} = \Pi T(\Phi r_t)$

  where Φ is a matrix of basis functions and Π is projection w/ respect to some weighted Euclidean norm $\|\cdot\|$
- **Norm mismatch issue:**
  - Π is nonexpansive with respect to $\|\cdot\|$
  - T is a contraction w/ respect to the sup norm
- **Key Question:** When is ΠT a contraction w/ respect to some norm?



PROJECTION W/ RESPECT TO DISTRIBUTION NORM

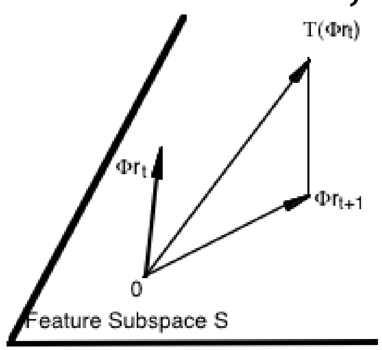
- Consider the steady-state distribution norm
  - Weight of ith component: the steady-state probability of state i in the Markov chain corresponding to the policy evaluated
  - Projection with respect to this norm can be approximated by simulation-based least-squares
- Remarkable Fact: If Π is projection w/ respect to the distribution norm, then ΠT is a contraction for discounted problems
- Average cost story is more complicated (Yu and Bertsekas, 2006)

LSPE: SIMULATION-BASED IMPLEMENTATION

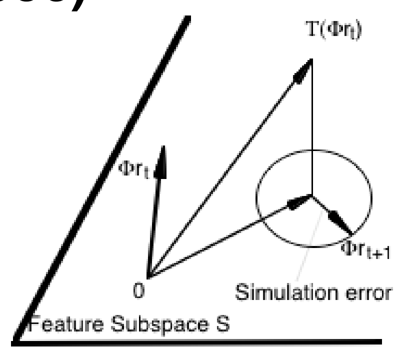
- Simulation-based implementation of $\Phi r_{t+1} = \Pi T(\Phi r_t)$ with an infinitely long trajectory, and least squares

  $\Phi r_{t+1} = \Pi T(\Phi r_t)$ + Diminishing simulation noise

- Interesting convergence theory (see papers at www site)
- Optimal convergence rate; much better than TD(λ), same as LSTD (Yu and Bertsekas, 2006)



Value Iteration with Linear Function Approximation



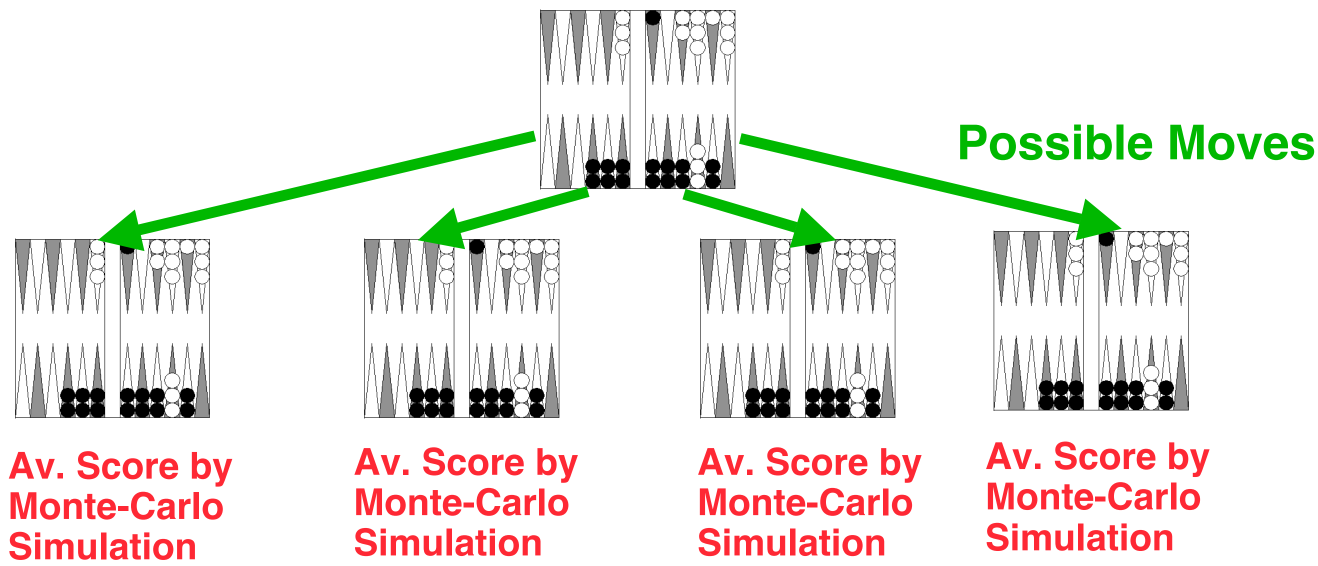
Simulation-Based Value Iteration with Linear Function Approximation

SUMMARY OF ACTOR-CRITIC SYSTEMS

- A lot of mathematical analysis, insight, and practical experience are now available
- There is solid theory for policy evaluation methods with linear function approximation: TD(λ), LSPE(λ), LSTD(λ)
- Typically, improved policies are obtained early, then oscillation “near” the optimum
- On-line computation is small
- Training is challenging and time-consuming
- Less suitable when problem data changes frequently

ROLLOUT POLICIES: BACKGAMMON PARADIGM

- On-line (approximate) cost-to-go calculation by simulation of some base policy (heuristic)
- Rollout: Use action w/ best simulation results
- Rollout is one-step policy iteration



COST IMPROVEMENT PROPERTY

- Generic result: Rollout improves on base heuristic
- A special case of policy iteration/policy improvement
- In practice, substantial improvements over the base heuristic(s) have been observed
- Major drawback: Extensive Monte-Carlo simulation
- Extension to multiple heuristics:
  - From each next state, run multiple heuristics
  - Use as value of the next state the best heuristic value
  - Cost improvement: The rollout algorithm performs at least as well as each of the base heuristics
- Interesting special cases:
  - The classical open-loop feedback control policy (base heuristic is the optimal open-loop policy)
  - Model predictive control (major applications in control systems)

STOCHASTIC PROBLEMS

- Major issue: Computational burden of Monte-Carlo simulation
- Motivation to use "approximate" Monte-Carlo
- Approximate Monte-Carlo by certainty equivalence: Assume future unknown quantities are fixed at some typical values
- Advantage : Single simulation run per next state, but some loss of optimality
- Extension to multiple scenarios (see Bertsekas and Castanon, 1997)

DETERMINISTIC PROBLEMS

- ONLY ONE simulation trajectory needed
- Use heuristic(s) for approximate cost-to-go calculation
  - At each state, consider all possible next states, and run the heuristic(s) from each
  - Select the next state with best heuristic cost
- Straightforward to implement
- Cost improvement results are sharper (Bertsekas, Tsitsiklis, Wu, 1997, Bertsekas 2005)
- Extension to constrained problems

ROLLOUT ALGORITHM PROPERTIES

- Forward looking (the heuristic runs to the end)
- Self-correcting (the heuristic is reapplied at each time step)
- Suitable for on-line use
- Suitable for replanning
- Suitable for situations where the problem data are a priori unknown
- Substantial positive experience with many types of optimization problems, including combinatorial (e.g., scheduling)

RELATION TO MODEL PREDICTIVE CONTROL

- **Motivation: Deal with state/control constraints**
- **Basic MPC framework**
  - Deterministic discrete time system $x_{k+1} = f(x_k,u_k)$
  - Control contraint U, state constraint X
  - Quadratic cost per stage: $x'Qx+u'Ru$
- **MPC operation: At the typical state x**
  - Drive the state to 0 in m stages with minimum quadratic cost, while observing the constraints
  - Use the 1st component of the m-stage optimal control sequence, discard the rest
  - Repeat at the next state

ADVANTAGES OF MPC

- It can deal explicitly with state and control constraints
- It can be implemented using standard deterministic optimal control methodology
- Key result: The resulting (suboptimal) closed-loop system is stable (under a "constrained controllability assumption" - Keerthi/Gilbert, 1988)
- Connection with infinite-time reachability
- Extension to problems with set-membership description of uncertainty

CONNECTION OF MPC AND ROLLOUT

- MPC <==> Rollout with suitable base heuristic

- Heuristic: Apply the (m-1)-stage policy that drives the state to 0 with minimum cost

- Stability of MPC <==> Cost improvement of rollout

- Base heuristic stable ==> Rollout policy is also stable

EXTENSIONS

- **The relation with rollout suggests more general MPC schemes:**
  - **Nontraditional control and/or state constraints**
  - **Set-membership disturbances**

- **The success of MPC should encourage the use of rollout**

RESTRICTED STRUCTURE POLICIES

- General suboptimal control scheme
- At each time step: Impose restrictions on future information or control
- Optimize the future under these restrictions
- Use 1st component of the restricted policy
- Recompute at the next step
- Special cases:
  - Rollout, MPC: Restrictions on future control
  - Open-loop feedback control: Restrictions on future information
- Main result for the suboptimal policy so obtained:

  It has better performance than the restricted policy

CONCLUDING REMARKS

- NDP is a broadly applicable methodology; addresses optimization problems that are intractable in other ways
- Many off-line and on-line methods to choose from
- Interesting theory
- No need for a detailed model; a simulator suffices
- Computational requirements are substantial
- Successful application is an art
- Many questions remain