A Series of Lectures on Approximate Dynamic Programming

Dimitri P. Bertsekas

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

Lucca, Italy
June 2017

APPROXIMATE DYNAMIC PROGRAMMING II

Outline

Simulation-Based Approximation in Value Space

Lookahead Minimization

Cost-to-go Approximation

First ℓ Steps

"Future"

$$\min_{u_k,\mu_{k+1},\ldots,\mu_{k+\ell-1}} E\left\{ g_k(x_k,u_k,w_k) + \sum_{m=k+1}^{k+\ell-1} g_k\big(x_m,\mu_m(x_m),w_m\big) + \tilde{J}_{k+\ell}(x_{k+\ell}) \right\}$$

Monte Carlo tree search

Rollout: Simulation with fixed policy
Parametric approximation at the end

Rollout: A General Method to Compute Cost-to-Go Approximations

Computes the lookahead functions $\tilde{J}_k$ as the cost-to-go functions of some suboptimal policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, referred to as the base policy or base heuristic

Rollout implementation

- We may use rollout in one-step or multistep lookahead
- We may calculate the base policy costs $\tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big)$ needed in

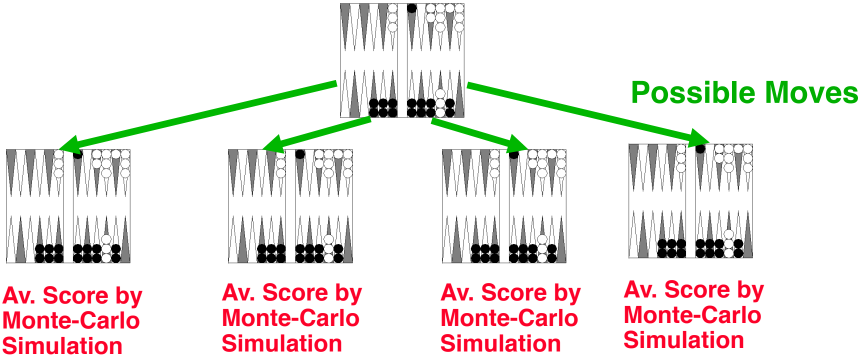
$$\min_{u_k \in U_k(x_k)} E\Big\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big)\Big\}$$

  (or its multistep version) analytically or by simulation
- The base policy costs $\tilde{J}_{k+1}$ may be calculated approximately over a rolling horizon, with a terminal cost approximation added at the end
- Simulation may be used for calculation of needed values of $\tilde{J}_{k+1}$
- The amount of simulation needed may be overwhelming (parallel computation helps). Simulation greatly simplifies if the problem is deterministic

Major fact about rollout

The rollout policy performs at least as well as the base policy. The improvement is often DRAMATIC. Relation to policy iteration method of infinite horizon DP

Example of Rollout: Backgammon



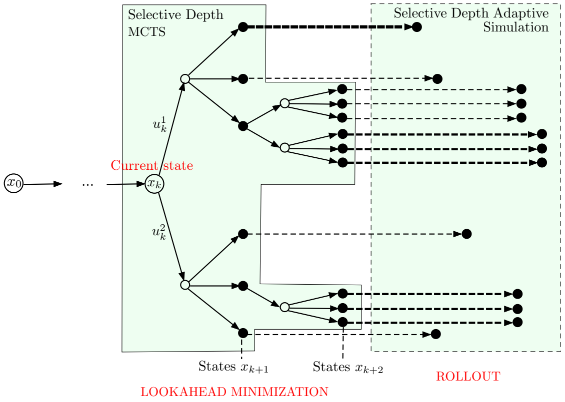
The original player (Tesauro, 1996):

- Involved one-step lookahead
- Base heuristic was a (relatively crude) backgammon player developed by different approximate DP methods
- The program played competitively to the best humans
- Was very time consuming (lots of parallelization of MC simulation)
- Subsequent improvements reduced the computation time

Example of Rollout: AlphaGo

Recent success: A Go program that plays at the level of the best humans

- Combines many of the ideas that we have discussed with awesome computing power and many heuristics
- Multistep lookahead (with Monte Carlo tree search and selective depth - see the next slide)
- Rollout with rolling horizon and cost function approximation (computed off-line with deep neural network)
- The base policy of the rollout is also computed off-line
- Massive on-line computation: 1920 CPUs and 280 GPUs, \$3000 electric bill per game!



MCTS aims to alleviate the drawbacks of simulation-based stochastic rollout

- The simulated trajectories may be too long
- Based on simulation results, some of the controls u_k may be clearly inferior
- Some controls u_k that appear to be promising, may be worth exploring better through multistep lookahead
- Uses selective depth lookahead, length of simulation, and discarding of controls

Using a Parametric Approximation Architecture for Policies

- Parametrize policies with a parameter vector $r = (r_0, \ldots, r_{N-1})$:

$$\pi(r) = \big\{\tilde{\mu}_0(x_0, r_0), \ldots, \tilde{\mu}_{N-1}(x_{N-1}, r_{N-1})\big\}$$

- Compute off-line the parameters based on some optimization
- Great advantage: The on-line implementation of the policy is very fast

Possible use: Implement policies obtained by approximation in value space

Lookahead Minimization — First ℓ Steps; Cost-to-go Approximation — "Future"; DP minimization

$$\min_{u_k, \mu_{k+1}, \ldots, \mu_{k+\ell-1}} E\left\{ g_k(x_k, u_k, w_k) + \sum_{m=k+1}^{k+\ell-1} g_k\big(x_m, \mu_m(x_m), w_m\big) + \tilde{J}_{k+\ell}(x_{k+\ell}) \right\}$$

- Compute off-line many state-control pairs (x_k^s, u_k^s), $s = 1, \ldots, q$
- Train a policy approximation architecture on these pairs. For example by solving for each k the least squares problem

$$\min_{r_k} \sum_{s=1}^{q} \left\| u_k^s - \tilde{\mu}_k(x_k^s, r_k) \right\|^2 + \text{(Regularization term)}$$

- This idea applies more generally. Generate many "good" state-control pairs (x_k^s, u_k^s), using a software or human "expert" and train in policy space

Cost Optimization Approach

- Minimize the cost $J_{\pi(r)}(x_0)$ over r
- Aim directly for an optimal policy within the parametric class
- Gradient-based optimization may be a possibility
- Random search in the space of r is another possibility (cross entropy method)

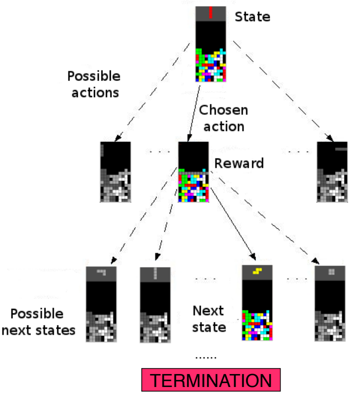
An important special case: Combination with approximation in value space

- For a given value space parametrization $r = (r_0, \ldots, r_{N-1})$, we define

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

- Has achieved success in a number of test problems (e.g., tetris)

An Example: Tetris (Often Used as Testbed in Competitions)



- Number of states $> 2^{200}$ (for 10×20 board)
- $J^*(x)$: optimal score starting from board position x
- Common choice: 22 features, readily recognized by tetris players as capturing important aspects of the board position (heights of columns, etc)
- Long history of successes and failures

Concluding Remarks

What we covered

- Approximate DP for finite horizon problems with perfect state information
- Approximation in value space
- Approximation in policy space; possibly in combination with approximation in value space

What we did not cover

- Approximate DP for infinite horizon problems
  - Lookahead and rollout ideas apply with essentially no change
  - Special training methods for approximation in value space
  - Temporal difference methods [e.g., TD(λ) and others]; TD(λ) is closely related with the proximal algorithm, but implemented by simulation (see internet videolecture)
- Imperfect state information problems can be converted to (much more complex) perfect state information problems. Approximate DP is essential for any kind of solution
- A variety of important lookahead/approximation in value space schemes: Model predictive control, open-loop feedback control, and others
- Alternative cost criteria: minimax/games, multiplicative/exponential cost, etc
- Approximation error bound analysis

Thank you!