Topics in Reinforcement Learning: AlphaZero, ChatGPT, Neuro-Dynamic Programming, Model Predictive Control, Discrete Optimization, Applications

Arizona State University
Course CSE 691, Spring 2025

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

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

Lecture 11
Adversarial/Minimax Problems, Minimax Rollout, Approximation in Both Value and Policy Space, Solution by MPC Methods, Computer Chess

Outline

Minimax Control - Robust Control - Games Against Nature

A worst case point of view of the uncertainty/disturbances

- The disturbances w_k are chosen by an adversarial and omniscient decision maker
- Mathematically, instead of a probabilistic description of w_k, assume a set membership description $w_k \in W_k$

Minimax Control Problems (Finite Horizon - Infinite Horizon is Similar)

- System is $x_{k+1} = f_k(x_k, u_k, w_k)$. We assume a set membership constraint $w_k \in W_k(x_k, u_k)$ [it may depend on (x_k, u_k)]
- The minimax control problem is to find a policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ with $\mu_k(x_k) \in U_k(x_k)$ for all x_k and k, which minimizes the cost function

$$J_\pi(x_0) = \max_{\substack{w_k \in W_k(x_k, \mu_k(x_k)) \\ k=0,1,\ldots,N-1}} \left[g_N(x_N) + \sum_{k=0}^{N-1} g_k\big(x_k, \mu_k(x_k), w_k\big) \right]$$

- The DP algorithm (max in place of $E\{\cdot\}$): Starting with $J_N^*(x_N) = g_N(x_N)$,

$$J_k^*(x_k) = \min_{u_k \in U(x_k)} \max_{w_k \in W_k(x_k, u_k)} \Big[g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big) \Big]$$

- Approximation in value space with one-step lookahead applies at state x_k a control

$$\tilde{u}_k \in \arg\min_{u_k \in U(x_k)} \max_{w_k \in W_k(x_k, u_k)} \Big[g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big]$$

- Similar to the stochastic case ... but the max operation is nonlinear and Monte Carlo simulation is unavailable (this affects everything: e.g., rollout etc)

Exact DP Approaches - Zero-Sum Game Theory and its Relation to Minimax Control

Connection to zero-sum games

Exact DP-based minimax approaches have been pursued in the context of game theory traditionally

Zero-sum game problems involve two players and a cost function; one player aims to minimize the cost and the other aims to maximize the cost

- They involve TWO minimax control problems:
  - The min-max problem where the minimizer chooses policy first and the maximizer chooses policy second with knowledge of the minimizer's policy
  - The max-min problem where the maximizer chooses policy first and the minimizer chooses policy second with knowledge of the maximizer's policy
  - Generally, we have Max-Min optimal value $\leq$ Min-Max optimal value
- Game theory is particularly interested on conditions that guarantee that Max-Min value $=$ Min-Max value.

Two Exceptional Minimax Control Problems for which Min-Max = Max-Min

- In an antagonistic practical RL context, Min-Max=Max-Min is unlikely to hold
- However there are at least two important exceptions

Discounted Markov games (Shapley 1953)

- Finite number of states and player controls, discounted, transition probabilities $p_{ij}(u, w)$, costs $g(i, u, w, j)$
- Randomized policies (u and w are probability distributions over the corresponding players' controls)
- Exact policy iteration is an interesting subject: It involves convergence difficulties, but can be modified to work (see the discussion and references in the course textbook)

Linear-quadratic games (classical control subject from the 60s; see e.g., the books by Basar)

- Linear system and quadratic cost for both players
- They can be solved in closed form
- We will discuss them later

Approximation in Value Space for Minimax Control

At State x_k → **Control** $\tilde{u}_k$

First ℓ Steps — "Future"

$$\min_{u_k,\mu_{k+1},\ldots,\mu_{k+\ell-1}} \; \max_{w_k,\ldots,w_{k+\ell-1}} \left[g_k(x_k,u_k,w_k) + \sum_{t=k+1}^{k+\ell-1} g_t\big(x_t,\mu_t(x_t),w_t\big) + \tilde{J}_{k+\ell}(x_{k+\ell}) \right]$$

Minimax Control Problem
with ℓ-Step Horizon

- Any cost function approximation $\tilde{J}_{k+\ell}(x_{k+\ell})$ is permissible
- Terminal cost approximation $\tilde{J}_{k+\ell}(x_{k+\ell})$ may be obtained by off-line training
- The "three approximations" view is valid (min approx, max approx, $\tilde{J}_{k+\ell}$ approx)
- The ℓ-step minimax control problem is solved by DP (over a "minimax tree" in finite state problems)
- There are variants with selective step lookahead, incremental pruning, etc
- The solution may be facilitated by special techniques, e.g., "alpha-beta pruning"
- This is the algorithm that most two-person game programs use for on-line play (including chess programs)

Rollout for Minimax Control in Discrete Spaces Problems

At State x_k

Cost of Base Policy Corresponding to $x_{k+1} = f_k(x_k, u_k, w_k)$ and $w_{k+1}, \ldots, w_{N-1}$

$$\min_{u_k \in U_k(x_k)} \max_{w_k, \ldots, w_{N-1}} \Big[g_k(x_k, u_k, w_k) + \tilde{J}_{\pi,k+1}\big(f_k(x_k, u_k, w_k), w_{k+1}, \ldots, w_{N-1}\big) \Big]$$

Rollout Control $\tilde{u}_k$

LONGEST PATH PROBLEM
Optimal Control with
Control Variables $w_k, \ldots, w_{N-1}$

- At state x_k: For $u_k \in U_k(x_k)$, compute the Q-factor of the base policy π

$$\tilde{Q}_k(x_k, u_k) = \max_{w_k, \ldots, w_{N-1}} \Big[g_k(x_k, u_k, w_k) + \tilde{J}_{\pi,k+1}\big(f_k(x_k, u_k, w_k), w_{k+1}, \ldots, w_{N-1}\big) \Big]$$

  This is a longest path problem
- Rollout control: $\tilde{u}_k \in \arg\min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k, u_k)$
- Any policy can be used as base policy (must be a legitimate policy for sequential consistency)
- Sequential consistency implies cost improvement
- Variants: Truncated, simplified, fortified, constrained, certainty equivalence etc

How Can we Implement Off-Line Training in Minimax Problems?

An interesting question:

How do various off-line training algorithms for one-player problems extend when approximations are used in the min-max and max-min problems?

Some possibilities:

- We can certainly improve either the minimizer's policy or the maximizer's policy by rollout, assuming a fixed policy for the opponent
- Can the policies be improved simultaneously? In practice this seems to work “often" ... but there is no reliable theory on this question ...
- In symmetric games like chess: What if a common policy is trained for both players?
- Examples where this does not work exist!
- We will next consider another approach: Introduce a fixed (but very skilled) “nominal" adversary, in place of the true adversary. Then train against this adversary.
- This converts the minimax problem to a one-player optimization problem, which can be dealt with an MPC-like methodology

A Fifteen-Minute Break

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

Combined Approximation in Value and Policy Space for Minimax Control

Motivation: Address the difficulties of minimax approximation in value space, rollout, and policy iteration

- We replace/approximate the maximizer with a nominal opponent, i.e. a known fixed policy $w_k = \nu(x_k, u_k)$
- Then the system and the cost function depend on a single player - the minimizer:

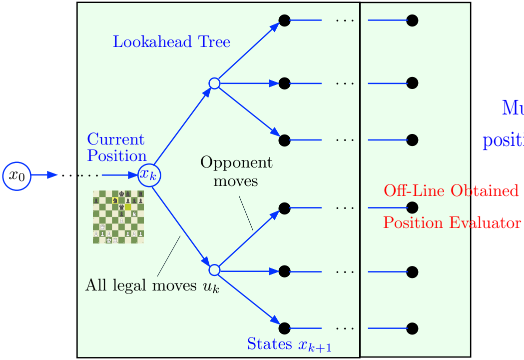
$$x_{k+1} = f\big(x_k, u_k, \nu(x_k, u_k)\big), \qquad g\big(x_k, u_k, \nu(x_k, u_k)\big)$$

- This is a problem that can be approached by one-player methods: approximation in value space/MPC - a much simpler problem

Issues:

- How do we determine the nominal opponent's policy $\nu(x, u)$?
  - Maximization against some “good policy of the minimizer"
  - Heuristics, like a list of adversarial responses to a list of (x, u) scenarios?
- How does the nominal opponent approximation affect the performance?
- In what follows, we will focus on this issue using examples.

Traditional On-Line Play in Computer Chess

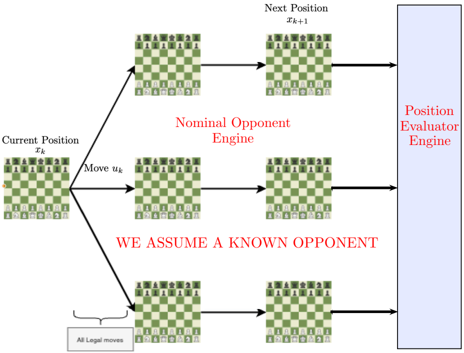


Multistep lookahead with position evaluation at the end

Our suggested alternative:

We replace the opponent moves with the moves of a “good" chess engine (this makes chess a one-player game that can be addressed with MPC)

MPC-MC (MetaChess): An Algorithm for Better Computer Chess



We use two available chess engines as components (a meta algorithm)

- The nominal opponent engine: Predicts the move of the true opponent of MPC-MC (exactly or approximately)
- The position evaluator engine: The base engine; evaluates any given position

Reference: Gundawar, Li, and Bertsekas, “Superior Computer Chess with Model Predictive Control, Reinforcement Learning, and Rollout," arXiv:2409.06477, 2024

MPC-MC: Computational Results Using the Stockfish (SK) Base Engine

We tested two variants of the algorithm

- Standard version
- Fortified version (based on fortified rollout ideas ... it plays a little better against very strong opponents, a little worse against weaker opponents)

Table: MPC-MC vs SK

SK Strength	Exact. Known Opponent		Approx. Known Opponent	
	Standard	Fortified	Standard	Fortified
0.5 secs	7.5-2.5	8-2	8-2	7-3
2 secs	5-5	5.5-4.5	5.5-4.5	6.5-3.5
5 secs	5-5	5.5-4.5	10-10	10.5-9.5

- We use MPC-MC (one-step lookahead), with SK as both the position evaluator and the nominal opponent, to play against SK
- Similar (but better) results obtained with other engines
- We can obtain better results with multistep lookahead
- Parallel computation is essential to reduce the move generation time

Our Speculation on Performance Improvement

Consistent observation:

MPC-MC improves the play of weak base engines (decisively), and the play of strong/world champion base engines (narrowly)

Why is this happening? (After all MPC-MC's lookahead is only one step longer)

- Observation: MPC-MC plays mostly the same moves as the base engine (SK) ... but varies in about 10-20 percent of the moves
- Occasionally, the base engine misses some hard-to-find strong moves at the first step of lookahead, which MPC-MC does not
- An important fact: The base engine prunes the lookahead tree at every step of lookahead
- By contrast, MPC-MC does not prune anything. Thus the first step of lookahead is exact, so MPC-MC performs a true Newton step
- This is all counterintuitive. However, it is consistent with the Newton step theory, and with much experimentation from other RL and MPC case studies
- A important point: The MPC-MC architecture applies to any deterministic two-player game

Minimax Linear Quadratic Problems - Closed Form Solution

- System is $x_{k+1} = ax_k + b_1 u_k + b_2 w_k$, and cost over infinite stages: $\sum_{k=0}^{\infty}(qx_k^2 + r_1 u_k^2 - r_2 w_k^2)$, where $q > 0$, $r_1 > 0$, and $r_2 > 0$ are given
- Objective: Minimize the cost over u_k, while it is maximized over w_k
- Closed form solution - Main result: The optimal cost J^* is given by $J^*(x) = K^* x^2$, with $K^* \geq 0$ being the unique nonnegative value satisfies $K = F(K)$, where

$$F(K) = \frac{a^2 r_1 r_2 K}{r_1 r_2 - (r_1 b_2^2 - r_2 b_1^2)K} + q \qquad \text{(the denominator is assumed positive)}$$

Approx. in value space: Terminal cost approx. K and actual/nominal opponent

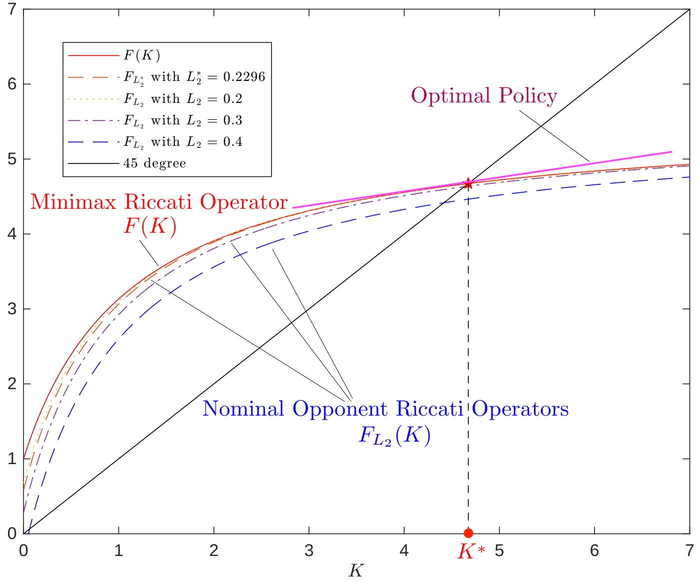
- Actual opponent: $\tilde{\mu}(x) \in \arg\min_u \max_w \{qx^2 + r_1 u^2 - r_2 w^2 + K(ax + b_1 u + b_2 w)^2\}$
- The minimizing u and maximizing w satisfies $u = \tilde{L}_1 x$ and $w = \tilde{L}_2 x$, where

$$\tilde{L}_1 = -\frac{ab_1 r_2 K}{r_1 r_2 - (r_1 b_2^2 - r_2 b_1^2)K}, \qquad \tilde{L}_2 = \frac{ab_2 r_1 K}{r_1 r_2 - (r_1 b_2^2 - r_2 b_1^2)K}$$

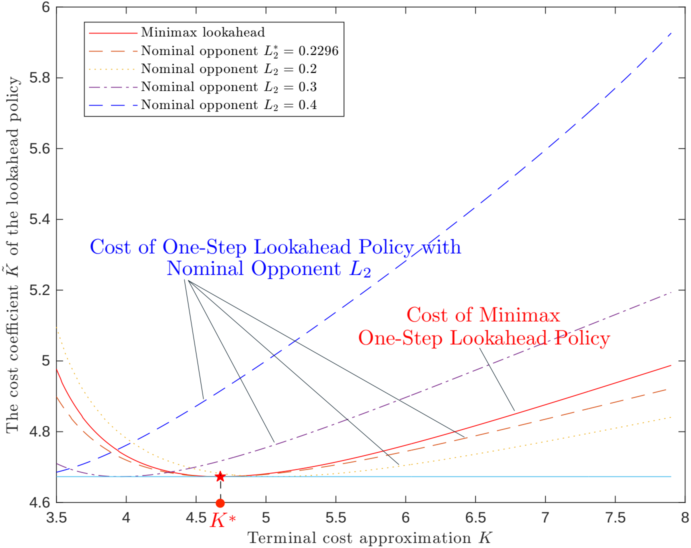
- Nom. opponent: $\tilde{\mu}(x) \in \arg\min_u \{qx^2 + r_1 u^2 - r_2 (L_2 x)^2 + K(ax + b_1 u + b_2 L_2 x)^2\}$
- The minimizing u satisfies $u = \tilde{L}_1 x$, where

$$\tilde{L}_1 = -\frac{\hat{a} b_1 K}{r_1 + b_1^2 K}, \qquad \hat{a} = a + b_2 L_2 \quad \text{(Note: } a \text{ is changed to } \hat{a}\text{)}$$

Minimax and Nominal Opponent Riccati Operators



Costs for the One-Step Lookahead Policy as a Function of the Cost Approximation K and the Nominal Opponent L_2



Final Notes

In the next lecture we will cover approximation in policy space, policy gradient methods

My lecture at the ASU Math Dept Friday, April 11th, at 1:30 PM in Wexler 206

- “Abstract Dynamic Programming and Reinforcement Learning"
- This is a mathematically oriented lecture on the foundations of our course
- Video recording to be posted at the course website by Saturday, April 12th