Machine Learning in Multi-Agent Planning

Geoff Gordon

ggordon@cs.cmu.edu
ML problems in multi-agent planning

Structured prediction problems
Interference from other agents
Help from other agents
Outline

Interference from other agents
  - No-regret algorithms

Help from other agents
  - Auction algorithms

Examples and experiments
Poker
Poker as a machine learning problem

Prediction problem:
- From: background knowledge, observations of players
- Predict: how should I play in next hand

How should I play:
- From: history of observations and actions in current hand
- Predict: should I bet on this round?
Why is poker hard?

Big output space
Don’t get information about “what ifs”
Structured output space
Neither adversarial nor cooperative
Why is poker hard?

Big output space
Don’t get information about “what ifs”
Structured output space
Neither adversarial nor cooperative
Output space

Behavior strategy: information state $\mapsto P(\text{action})$

$[0, 1]$ # info states

9 K pass pass bet call 8 7 K $\mapsto P(\text{bet}) = .72$
Structured prediction problem

- $\mathcal{X}$: input space
- $\mathcal{Y}$: prediction space
- $\mathcal{H}$: hypothesis space
- $\ell_t$: loss functions ($((y - 5)^2, D_{KL}(y \mid (.2, .3, .5)))$)

Will assume $\mathcal{H}$ is convex, $\subset \mathbb{R}^{d \times n}$

May have many vertices, many faces, or other complex features, but we have an efficient description
Poker as structured prediction

- $\mathcal{X}$: has Bill been drinking
- $\mathcal{Y}$: dist’n over behavior strategies
- $\mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$
- $\ell_t$: money I lose in hand $t$
Path planning w/ unknown costs
Path planning as structured prediction

\[ x = \begin{pmatrix} \text{day of week} \\ \text{phase of moon} \\ \text{recent eclipse} \end{pmatrix} \]

\[ y = \text{paths in map} \]

\[ H = x \mapsto y \]

\[ \ell(y) = (c + C') \cdot y \]
Adversarial path planning
Lookahead in multiagent planning
Lookahead, cont’d

\[ X = \text{features of current history} \]
\[ Y = \text{behavior strategies} \]
\[ H = X \mapsto Y \]
\[ \ell(y) = E(\text{heuristic(leaf)}) \] (note: what ifs)
One-card poker

You have won $0 so far.
You and the computer each ante $1.
Your card is A. What do you bet?
## Rock-paper-scissors

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>P</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>
Neither statistical nor adversarial

Possible approaches:
- Learn model of environment (incl. other agents)
- Compute equilibrium (minimax, Bayes-Nash, ...)

Problems:
- Catastrophic failure when not i.i.d.
- Assumes you know other agents’ motivations (inappropriate level of paranoia)
## Building models

<table>
<thead>
<tr>
<th>Sequence</th>
<th>$HTHTHTHTHTHHTH$ ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step $t$</td>
<td>1 2 3 4 5 ...</td>
</tr>
<tr>
<td>Predict</td>
<td>$T H T H T H T$ ...</td>
</tr>
<tr>
<td>Actual</td>
<td>$H T H T H H$ ...</td>
</tr>
</tbody>
</table>
Paranoia

Gambler

Dealer
### Tournament results

<table>
<thead>
<tr>
<th>Player</th>
<th>Win 1</th>
<th>Player</th>
<th>Win 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxPlayer</td>
<td>3.4856</td>
<td>Baazigar</td>
<td>2.3974</td>
</tr>
<tr>
<td>Victor</td>
<td>2.9968</td>
<td>Blitz</td>
<td>1.8700</td>
</tr>
<tr>
<td>Mouse</td>
<td>2.9552</td>
<td>CheatToLose</td>
<td>1.7122</td>
</tr>
<tr>
<td>TeamDiscoveryChannel</td>
<td>2.8924</td>
<td>OptPlayer (minimax)</td>
<td>1.5054</td>
</tr>
<tr>
<td>ActorCritic</td>
<td>2.8624</td>
<td>Corrado</td>
<td>0.7686</td>
</tr>
<tr>
<td>PatternRecognition</td>
<td>2.8572</td>
<td>KennyRogers</td>
<td>-0.7920</td>
</tr>
<tr>
<td>KillerPlayer</td>
<td>2.7906</td>
<td>RandomPlayer</td>
<td>-10.7776</td>
</tr>
<tr>
<td>YoavShohamAllStars</td>
<td>2.7594</td>
<td>SomeRegret</td>
<td>-22.4770</td>
</tr>
</tbody>
</table>

Total winnings in 10,000 games (5,000 each as Dealer and Gambler)
No regret

Regret = $\rho = \text{how much do I wish I had done something else?}$

E.g., opp played $RRRRPRRRRSRRRR$, I played at random

Lots of regret for not playing “$P$ all the time”

(Lots of negative regret for “$S$ all the time”)
Comparison class

Allowable “something else” = comparison class

Little class: easy to get algorithms, but low regret isn’t impressive

Big class: hard to get algorithms, but low regret inspires confidence

Typical:
- all constant $h \in \mathcal{H}$ (e.g. “$R$ all the time”)
- simple rules for modifying $h$ (e.g. $R \leftrightarrow P$)
No-regret algorithms

Guarantee $\rho$ grows slower than $O(t)$, often $O(\sqrt{t})$

Average regret $\frac{\rho}{t} \to 0$ as $t \to \infty$ at rate $1/\sqrt{t}$

Guarantee is for all sequences of opp plays

$\Rightarrow$ approach equilibrium if opponent plays well, something like CLT if opponent plays obliviously
No-regret intuition

Choose actions with positive regret
Regret for chosen action can’t increase
When two actions have similar regrets, randomize
Algorithm recipe

Pick a potential function, $F(S)$
Keep track of regret vector, $S_t$
Compute $F'(S_t) = f(S_t) = \bar{H}$
Normalize to get $H = \alpha \bar{H} \in \mathcal{H}$
Play $H$
Regret vector and potential fn

\[ \rho(H) = S_t \cdot H = \text{regret vs. } H \]

safe set = \( \{ S \mid (\forall H \in \mathcal{H}) \; S \cdot H \leq 0 \} \)

In safe set \( \Rightarrow \rho \leq 0 \)

Potential: large outside safe set, zero inside, bounded curvature
Regret in RPS

\[ x_t = \begin{pmatrix} 1 & \text{if I played R} \\ 1 & \text{if I played P} \\ 1 & \text{if I played S} \end{pmatrix} \]

\[ y_t = \text{same for opponent} \]

\[ M y_t = \text{my payoffs for each action at time } t \]

\[ r_t = x_t \cdot M y_t = \text{my payoff} \]

\[ s_t = M y_t - r_t 1 = \text{my regret vector} \]

\[ s = \sum_t s_t \quad \rho = \max s \]
Safe set in RPS

Machine Learning in Multi-Agent Planning – p.30/59
Potential

\[ F(s) = \| [s]_+ \|^2 \]
Algorithm for RPS

Given $s$

Compute $s_+$

Renormalize to get $q = \alpha s_+$

Randomize according to $q$

“Regret matching” [Hart & Mas-Colell]
One-card poker

Define $\mathcal{X}, \mathcal{Y}, \mathcal{H}$ and $\ell$

Define regret vector $S$

Pick $F$ (will be $F(S) =$ squared distance from safe set)

Show how to compute $F'(S)$
One-card poker

\[ \mathcal{X} = \{1\} \]

\[ \mathcal{Y} = \text{(randomized) poker strategies} \]

\[ \mathcal{H} = \mathcal{Y} \]

\[ \ell_t = \text{money lost on round } t \]
Regret vector

Poker strategies are $H = \{h \mid Ah + b = 0, \ h \geq 0\}$

Payoffs are $r \cdot h$

Regret is

$$S_t = \begin{pmatrix} r_t \\ -r_t \cdot h_t \end{pmatrix}$$

Now $(h, 1) \cdot S = h \cdot r_t - r_t \cdot h_t$

Safe set is $H^\perp = \{y \mid y \cdot h \leq 0 \ \forall h \in H\}$
Safe set is $H^\perp = \{y \mid y \cdot h \leq 0 \ \forall h \in H\}$
One-card poker game tree
Information states and actions

Gambler:
- Holding 2-A in first round
- Holding 2-A in second round
- Pass or bet for each

Dealer:
- Holding 2-A after pass
- Holding 2-A after bet
- Pass or bet for each
Sequences

Sequence = history of my observations and actions, ending in my action

Ex: J pass bet bet

Special case: $\epsilon$ = empty sequence

Sequences for all players $\Rightarrow$ leaf of game tree

1CP: 780 leaves, 638 internals, 52 seqs/player
Sequence weights

Let $s$, $a$ be my info state, action

$$x_{sa} = \prod_{\text{ancestors} \ (s', a') \text{ of } s} P(a' | s')$$

$s'$ is my info state, $s' \neq s$, $a'$ leads towards $s$

Ex: J pass raise call

$\Rightarrow P(\text{pass} | J) P(\text{call} | J \text{ pass raise})$
Convex sequence weight set

At any choice point, split flow from most recent ancestor I control
Linear payoffs

Strategies $x(\text{sequence}), y(\text{sequence})$

Payoff = $\sum_{\text{leaves}} r_i P(i \mid x, y)$

Ex: $J K \text{ pass raise call}$

$P(J)P(K)P(\text{pass} \mid J)P(\text{raise} \mid K \text{ pass})$

$P(\text{call} \mid J \text{ pass raise})$

$P(J)P(K)x(J \text{ pass raise call})y(K \text{ pass raise})$
The algorithm

Given game $A$, $a$ and regret vector $s$, solve:

$\min_{x, \lambda} \left( \|x\|^2/2 - s \cdot x \right)$

$Ax = \lambda a \quad x, \lambda \geq 0$

Then play $x/\lambda$ (or arbitrarily if $\lambda = 0$)

Note: if $A = (1, 1, 1, \ldots)$ and $a = 1$, then $x = [s]_+$

(regret matching)
One-card poker self-play
Play against fixed Gambler

![Graph showing the comparison of different bounds and values over iterations. The graph includes lines for Gambler bound, Dealer bound, Avg payoff, and Minimax value. The x-axis represents iterations (0 to 250), and the y-axis represents the payoff values ranging from -0.4 to 0.6.]
Help, I need somebody
Task allocation

[image credit: CMU’s FIRE project]
Traffic control
Basic auction algorithm

1. [everyone] Start with nominal plan
2. [auctioneer] Identify a point of interaction
3. [bidders] Calculate and submit bids
4. [auctioneer] Clear auction
5. Repeat from 2
[e.g., contract net protocol]
Auction applied to traffic control

1. Plan paths assuming we know edge costs, store cost-to-goal
2. Execute until conflict
3. Look ahead at alternate routes, calculate bids
4. Assign best feasible combination of paths
5. Repeat from 2
Chicken and egg

Look-ahead is inefficient w/o good cost-to-goal function

Cost-to-goal function requires good price estimates

Accurate price estimates require successful look-ahead
Path planning as LP

\[-f_{xy} - f_{xz} + 1 = 0\]

\[f_{xy} - f_{yg} = 0\]

\[f_{xz} - f_{zg} = 0\]

\[f_{yg} + f_{zg} - f_g = 0\]

\[f_{xy}, f_{xz}, f_{yg}, f_{zg}, f_g \geq 0\]
Congestion, single-agent case

\[-f_{xy} - f_{xz} + 1 = 0\]
\[f_{xy} - f_{yg} = 0\]
\[f_{xz} - f_{zg} = 0\]
\[f_{yg} + f_{zg} - f_{g} = 0\]

\[f_{zg} \leq 1/2\]

\[f_{xy}, f_{xz}, f_{yg}, f_{zg}, f_{g} \geq 0\]
Congestion, two agents

\[-f_{xy} \quad -f_{xz} \quad +1 = 0\]
\[-g_{xy} \quad -g_{xz} \quad +1 = 0\]

\[\ldots\]
\[f_{zg} + g_{zg} \leq 1\]
\[f_{xy}, f_{xz}, \ldots, g_{xy}, \ldots \geq 0\]
In general

\[
\max_{f_i} \sum_i c_i \cdot f_i \\
\forall i: \ A_i f_i = b_i \\
(*) \quad \sum_i C_i f_i = d \\
\forall i: \ f_i \geq 0
\]
Overall algorithm

Send prices to robots
Robots plan individually, using prices to avoid congested areas
Report paths back to master
Master finds best combination of known paths, recalculates prices
Repeat until done

Return: cost-to-goal for each robot
Example: fuel constraint

[Animation]
Auctions and no-regret in paintball

[joint work with Curt Bererton, Sebastian Thrun]
Lookahead search

[joint work with Rosemary Emery-Montemerlo, Sebastian Thrun]