Game AI & Search

Wouter M. Koolen

2023-03-30

CWI & UT

Credentials

- Quarto
- Quoridor
- Scrabble
- some research papers on learning

Why are we here today?

Games

- Games as objects for studying learning about
- Game theory/methods as engines driving learning methods


THERE IS A GAME ENGINE
INSIDE YOUR GAME ENGINE
imgflip.com

PAC Learning

Definition of Learning

Suitable for passive and interactive learning

Definition of Learning [Valiant'84]

- Start with any computational problem. (However simple)
  - Sorting
  - Shortest path in graph, planning in MDP
  - Least squares, curve fitting
  - Saddle point (of matrix game)
  - Backward induction (of game DAG)
  - Linear / convex / submodular optimisation
  - ...
  - **Argmax** of finite dimensional vector:
    Given $\mu \in \mathbb{R}^K$ compute $\arg\max \mu_k$.

Definition of Learning [Valiant'84]

- Start with any computational problem. (However simple)
  - Sorting
  - Shortest path in graph, planning in MDP
  - Least squares, curve fitting
  - Saddle point (of matrix game)
  - Backward induction (of game DAG)
  - Linear / convex / submodular optimisation
  - ...
  - **Argmax** of finite dimensional vector:
    Given $\mu \in \mathbb{R}^K$ compute $\arg\max \mu_k$.
- Give the learner noisy access to the inputs
  - Pick $\mu_i = \mathbb{E}_{\nu_i}[X_i]$. Provide samples $X_1, X_2, \ldots$ i.i.d. from ν_i.
  - (Active learning) Let the learner *choose* experiments

Definition of Learning [Valiant'84]

- Start with any computational problem. (However simple)
  - Sorting
  - Shortest path in graph, planning in MDP
  - Least squares, curve fitting
  - Saddle point (of matrix game)
  - Backward induction (of game DAG)
  - Linear / convex / submodular optimisation
  - ...
  - **Argmax** of finite dimensional vector:
    Given $\mu \in \mathbb{R}^K$ compute $\arg\max \mu_k$.
- Give the learner noisy access to the inputs
  - Pick $\mu_i = \mathbb{E}_{\nu_i}[X_i]$. Provide samples $X_1, X_2, \ldots$ i.i.d. from ν_i.
  - (Active learning) Let the learner *choose* experiments
- Learner must return an ϵ-optimal solution w.p. $1 - \delta$

Definition of Learning [Valiant'84]

- Start with any computational problem. (However simple)
    - Sorting
    - Shortest path in graph, planning in MDP
    - Least squares, curve fitting
    - Saddle point (of matrix game)
    - Backward induction (of game DAG)
    - Linear / convex / submodular optimisation
    - ...
    - **Argmax** of finite dimensional vector:
      Given $\mu \in \mathbb{R}^K$ compute $\arg\max \mu_k$.
- Give the learner noisy access to the inputs
    - Pick $\mu_i = \mathbb{E}_{\nu_i}[X_i]$. Provide samples $X_1, X_2, \ldots$ i.i.d. from ν_i.
    - (Active learning) Let the learner *choose* experiments
- Learner must return an ϵ-optimal solution w.p. $1 - \delta$
- Questions: statistics (# samples), computation (algorithm design)

My Interest

PAC learning puts forward *crisp objective* metrics for judging learning algorithms.

- Approximation error: ϵ
- Confidence: δ
- Sample complexity: m
- Computational complexity: $O(n)$

My Interest

PAC learning puts forward *crisp objective* metrics for judging learning algorithms.

- Approximation error: ϵ
- Confidence: δ
- Sample complexity: m
- Computational complexity: $O(n)$

So we can reason about optimality in handling:

- Features of original computational problem
- Shape constraints assumptions (between means)
- (Non-)parametric assumptions on sampling processes
- Constraints on experiments
- Strategies/templates for dealing with uncertainty, “inference”

Why Games, Then?

Games provide a series of natural problems

- Beyond convex hull of what we know how to do $\Rightarrow$ Challenging
- Yet with hint of mathematical tractability $\Rightarrow$ Ripe
- Applications plentiful $\Rightarrow$ Useful

Why Games, Then?

Games provide a series of natural problems

- Beyond convex hull of what we know how to do $\Rightarrow$ Challenging
- Yet with hint of mathematical tractability $\Rightarrow$ Ripe
- Applications plentiful $\Rightarrow$ Useful

Best first move, PV, saddle point, value

Why Games, Then?

Games provide a series of natural problems

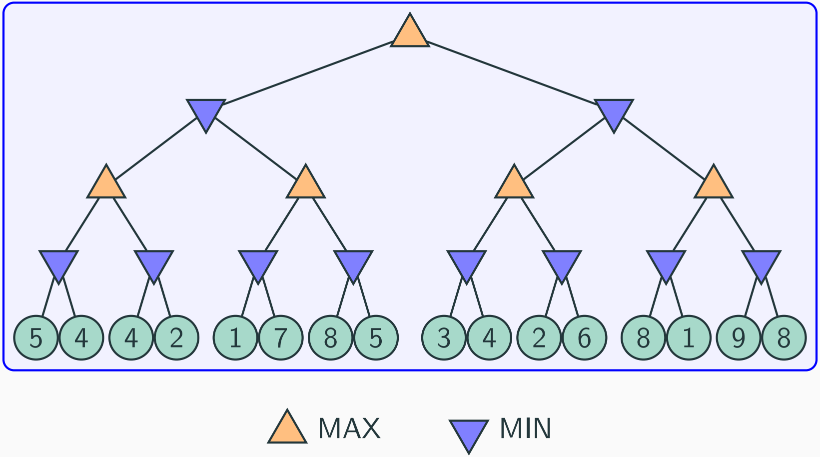
- Beyond convex hull of what we know how to do $\Rightarrow$ Challenging
- Yet with hint of mathematical tractability $\Rightarrow$ Ripe
- Applications plentiful $\Rightarrow$ Useful

Best first move, PV, saddle point, value

- Learning and Games: cross fertilization
- Learning results inspire systems
- Ambition to explain success of current game systems

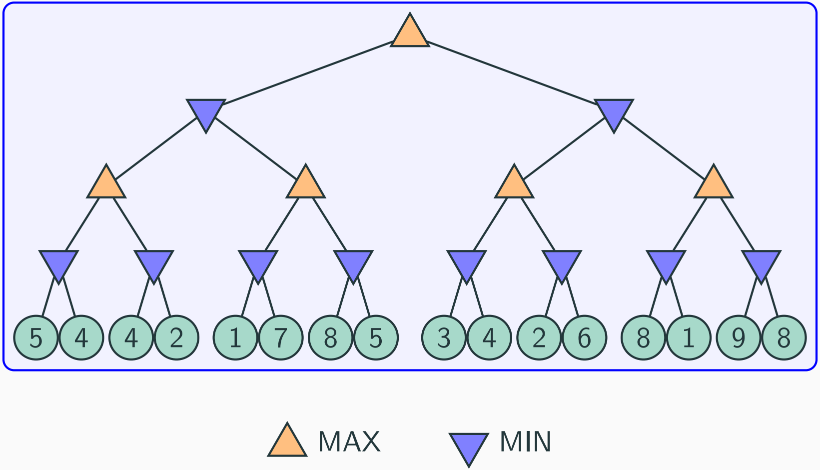
Tree Search

Deterministic Problem

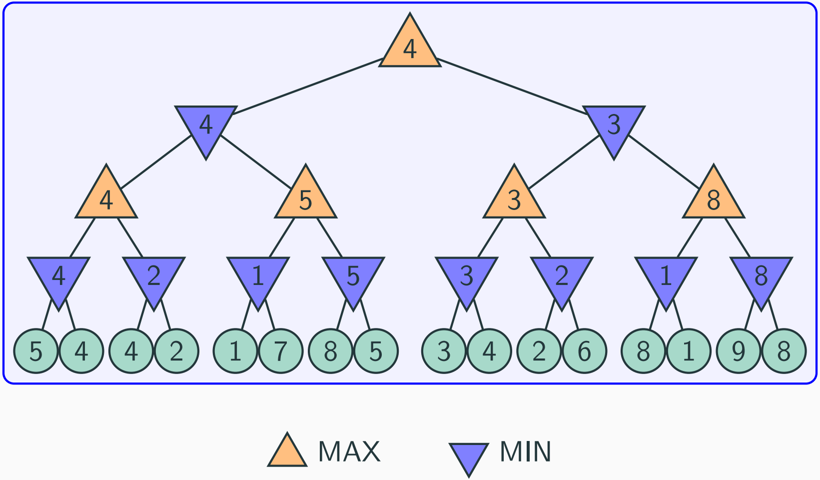


Problem
Given leaf values $\boldsymbol{\mu}$, compute $i^*(\boldsymbol{\mu}) := \operatorname*{argmax}_{a_1} \min_{a_2} \max_{a_3} \min_{a_4} \mu_{a_1 a_2 a_3 a_4}$

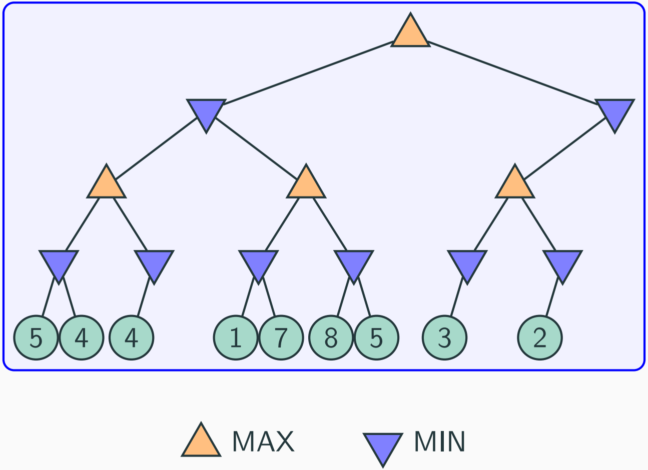
Backward Induction Computation



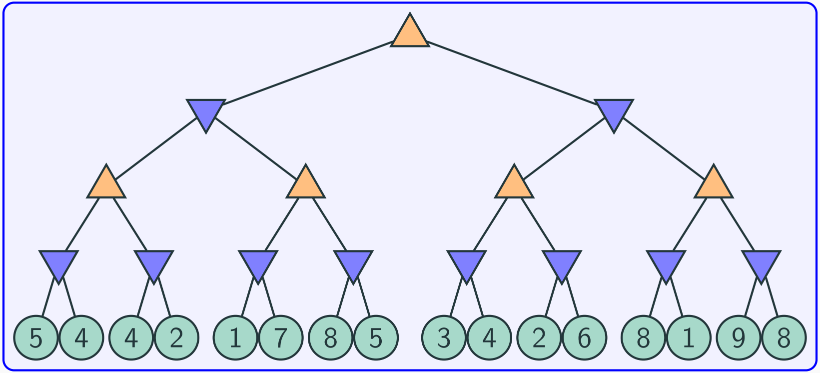
Backward Induction Computation



Alpha-beta Pruning

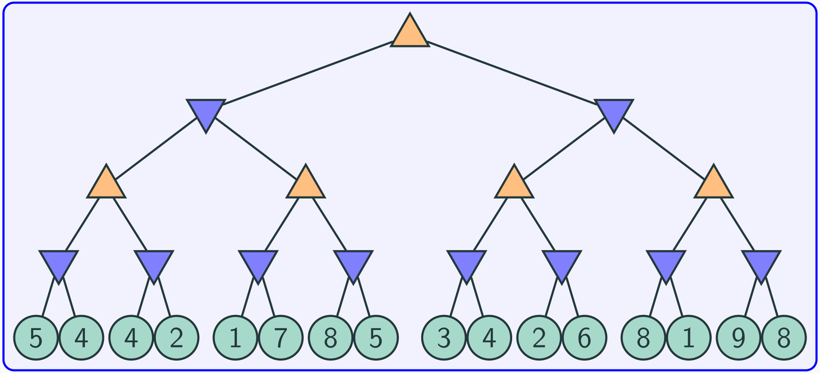


Model (Teraoka, Hatano, and Takimoto, 2014)



Maximin Action Identification Problem
Find best move at root from samples of leaves.

Model (Teraoka, Hatano, and Takimoto, 2014)



Maximin Action Identification Problem
Find best move at root from samples of leaves.

Model

Definition

A game tree is a min-max tree with leaves $\mathcal{L}$. A **bandit model** $\boldsymbol{\mu}$ assigns a distribution μ_ℓ to each leaf $\ell \in \mathcal{L}$.

Model

Definition

A game tree is a min-max tree with leaves $\mathcal{L}$. A **bandit model** $\boldsymbol{\mu}$ assigns a distribution μ_ℓ to each leaf $\ell \in \mathcal{L}$.

The maximin action (best action at the root) is

$$i^*(\boldsymbol{\mu}) \;:=\; \operatorname*{argmax}_{a_1} \min_{a_2} \max_{a_3} \min_{a_4} \cdots \; \mu_{a_1 a_2 a_3 a_4 \ldots}$$

Model

Definition

A game tree is a min-max tree with leaves $\mathcal{L}$. A **bandit model** $\boldsymbol{\mu}$ assigns a distribution μ_ℓ to each leaf $\ell \in \mathcal{L}$.

The maximin action (best action at the root) is

$$i^*(\boldsymbol{\mu}) \;:=\; \operatorname*{argmax}_{a_1} \min_{a_2} \max_{a_3} \min_{a_4} \cdots \; \mu_{a_1 a_2 a_3 a_4 \ldots}$$

Protocol

For $t = 1, 2, \ldots, \tau$:

- Learner picks a leaf $L_t \in \mathcal{L}$.
- Learner sees $X_t \sim \mu_{L_t}$

Learner recommends action $\hat{I}$

Objectives

Learner is δ-PAC if

$$\forall \boldsymbol{\mu} : \mathop{\mathbb{P}}_{\boldsymbol{\mu}}\left(\tau < \infty \wedge \hat{I} \neq i^*(\boldsymbol{\mu})\right) \leq \delta$$

Goal: minimise sample complexity $\mathbb{E}_{\boldsymbol{\mu}}[\tau]$ over all δ-PAC strategies.

Main Theorem I: Lower Bound

Define the *alternatives* to $\boldsymbol{\mu}$ by $\mathrm{Alt}(\boldsymbol{\mu}) = \{\boldsymbol{\lambda} | i^*(\boldsymbol{\lambda}) \neq i^*(\boldsymbol{\mu})\}$.

NB here i^* is best action at the root

Main Theorem I: Lower Bound

Define the *alternatives* to $\boldsymbol{\mu}$ by $\text{Alt}(\boldsymbol{\mu}) = \{\boldsymbol{\lambda} | i^*(\boldsymbol{\lambda}) \neq i^*(\boldsymbol{\mu})\}$.

NB here i^* is best action at the root

Theorem (Castro 2014; Garivier and Kaufmann 2016)

Fix a δ-correct strategy. Then for every bandit model $\boldsymbol{\mu}$

$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \;\geq\; T^*(\boldsymbol{\mu}) \ln \frac{1}{\delta}$$

where the characteristic time $T^*(\boldsymbol{\mu})$ is given by

$$\frac{1}{T^*(\boldsymbol{\mu})} \;=\; \max_{\boldsymbol{w} \in \triangle_K} \min_{\boldsymbol{\lambda} \in \text{Alt}(\boldsymbol{\mu})} \sum_{i=1}^{K} w_i \, \text{KL}(\mu_i \| \lambda_i).$$

Main Theorem II: Algorithm

Idea is consider the oracle weight map

$$\boldsymbol{w}^*(\boldsymbol{\mu}) \;:=\; \operatorname*{argmax}_{\boldsymbol{w}\in\triangle_K} \min_{\boldsymbol{\lambda}\in\mathrm{Alt}(\boldsymbol{\mu})} \sum_{i=1}^{K} w_i \,\mathrm{KL}(\mu_i\|\lambda_i)$$

and track the plug-in estimate: sample leaf $L_t \sim \boldsymbol{w}^*(\hat{\boldsymbol{\mu}}(t-1))$.

Main Theorem II: Algorithm

Idea is consider the oracle weight map

$$\boldsymbol{w}^*(\boldsymbol{\mu}) \;:=\; \operatorname*{argmax}_{\boldsymbol{w}\in\triangle_K} \min_{\boldsymbol{\lambda}\in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{i=1}^{K} w_i \,\mathrm{KL}(\mu_i\|\lambda_i)$$

and track the plug-in estimate: sample leaf $L_t \sim \boldsymbol{w}^*(\hat{\boldsymbol{\mu}}(t-1))$.

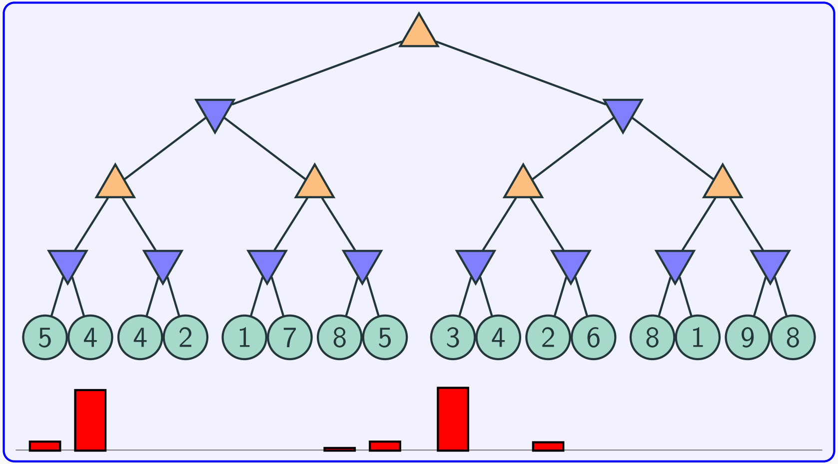
Theorem (Degenne and Koolen, 2019)

Take set-valued interpretation of argmax *defining* $\boldsymbol{w}^*$. *Then* $\boldsymbol{\mu} \mapsto \boldsymbol{w}^*(\boldsymbol{\mu})$ *is upper-hemicontinuous and convex-valued. Suitable tracking ensures that as* $\hat{\boldsymbol{\mu}}(t) \to \boldsymbol{\mu}$, *any choice* $\boldsymbol{w}_t \in \boldsymbol{w}^*(\hat{\boldsymbol{\mu}}(t-1))$ *have*

$$\min_{\boldsymbol{w}\in\boldsymbol{w}^*(\boldsymbol{\mu})} \|\boldsymbol{w}_t - \boldsymbol{w}\|_\infty \to 0$$

Track-and-Stop is asymptotically optimal: $\limsup_{\delta\to 0} \frac{\mathbb{E}_{\boldsymbol{\mu}}[\tau]}{\ln\frac{1}{\delta}} = T^*(\boldsymbol{\mu})$.

Example with oracle weights



On Computation

To compute a gradient (in $\boldsymbol{w}$) we need to differentiate

$$\boldsymbol{w} \;\mapsto\; \min_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{i=1}^{K} w_i \, \mathrm{KL}(\mu_i \| \lambda_i)$$

An optimal $\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})$ can be found by binary search for common value plus tree reasoning in $O(|\mathcal{L}|)$.

AI

But where do we put the AI?

One possible answer:

End-to-End-Deep-Learning
Excellent performance at relatively little understanding. (Silver et al., 2017)

Another possible answer:

Algorithms with Predictions
Heuristics and guarantees can typically be combined.

- Algorithm with worst-case performance guarantee . . .
- . . . that improves whenever predictions are good
- Example: node ordering for (α, β) pruning.

Open Problems

Problem: Multiple answers and $\epsilon > 0$

We have seen Best Action Identification. Instance of PAC learning with $\epsilon = 0$.

Lower bound complexity essentially governed by certain gaps between leaf means.

We can learn **faster** if we accept any ϵ-optimal move.

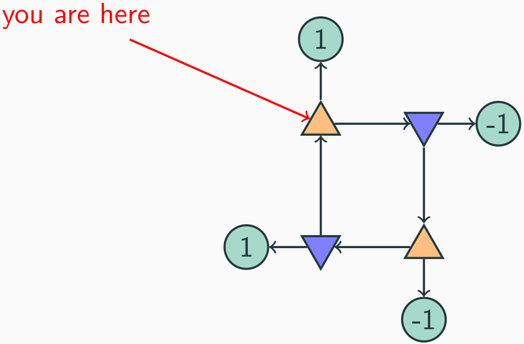
Theory currently underdeveloped:

- Asymptotic optimality as $\delta \to 0$ results exist (Degenne and Koolen, 2019) but are of questionable *practicality*.
- My perspective: lower bounds need sharpening
- Practical schemes based on confidence intervals Kaufmann and Koolen, 2017

Problem
What is the sample complexity of ϵ Best Action Identification?

Problem: Cyclic game graphs

Consider a two-player zero-sum **cyclic** game graph.



Convention: cycling forever results in value 0.

Problem
What is the sample complexity of finding the best move in a cyclic game?

Problem: Facing the Curse of Dimensionality

Game trees are intractably big. Practical systems (Silver et al., 2017)

- Progressively expand strategic horizon
- Cluster similar nodes
- Exploit background knowledge
-

Problem: Mixtures and $\epsilon > 0$

With Julia Olkovskaya (in progress) we found an intriguing possibility.

Observation
In bandits, it appears much easier to find a ϵ-good mixture of arms than to find a **individual** ϵ best arm.

- Statistics improve
- Computation streamlines

Problem
What is the sample complexity of ϵ- Best Mixture of Actions Identification?

Problem: Oracle Strategy Computation

Algorithms based on lower bound problems of the form

$$\operatorname*{argmax}_{\boldsymbol{w} \in \triangle_K} \min_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{i=1}^{K} w_i \, \mathrm{KL}(\mu_i \| \lambda_i)$$

With the underlying deterministic problem determining what Alt means.

Problem
As a function of Alt, *what is the computational complexity of optimisation of the above objective?*

Gradient descent vs cutting plane vs interior point methods? Can we do Nesterov acceleration? Scaling with the dimension K and perhaps the problem rank (Kaufmann and Koolen, 2021)?

Conclusion

Games and **Learning** fertile research areas with interesting, challenging, promising intersection.

References

Castro, R. M. (Nov. 2014). “Adaptive sensing performance lower bounds for sparse signal detection and support estimation”. In: *Bernoulli* 20.4, pp. 2217–2246.

Degenne, R. and W. M. Koolen (Dec. 2019). “Pure Exploration with Multiple Correct Answers”. In: *Advances in Neural Information Processing Systems (NeurIPS) 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 14591–14600.

Garivier, A. and E. Kaufmann (2016). “Optimal Best arm Identification with Fixed Confidence”. In: *Proceedings of the 29th Conference On Learning Theory (COLT)*.

Kaufmann, E. and W. M. Koolen (Nov. 2021). “Mixture Martingales Revisited with Applications to Sequential Tests and Confidence Intervals”. In: *Journal of Machine Learning Research* 22.246, pp. 1–44.

Kaufmann, E. and W. M. Koolen (Dec. 2017). "Monte-Carlo Tree Search by Best Arm Identification". In: *Advances in Neural Information Processing Systems (NeurIPS) 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, pp. 4904–4913.

Silver, D. et al. (Oct. 2017). "Mastering the game of Go without human knowledge". In: *Nature* 550, pp. 354–359.

Teraoka, K., K. Hatano, and E. Takimoto (2014). "Efficient Sampling Method for Monte Carlo Tree Search Problem". In: *IEICE Transactions* 97-D.3, pp. 392–398.