Oxford: Open-ended learning in symmetric zero-sum games

David Balduzzi w/ Marta Garnelo, Yoram Bachrach, Wojtek Czarnecki, Julien Perolat, Max Jaderberg, Thore Graepel

DeepMind

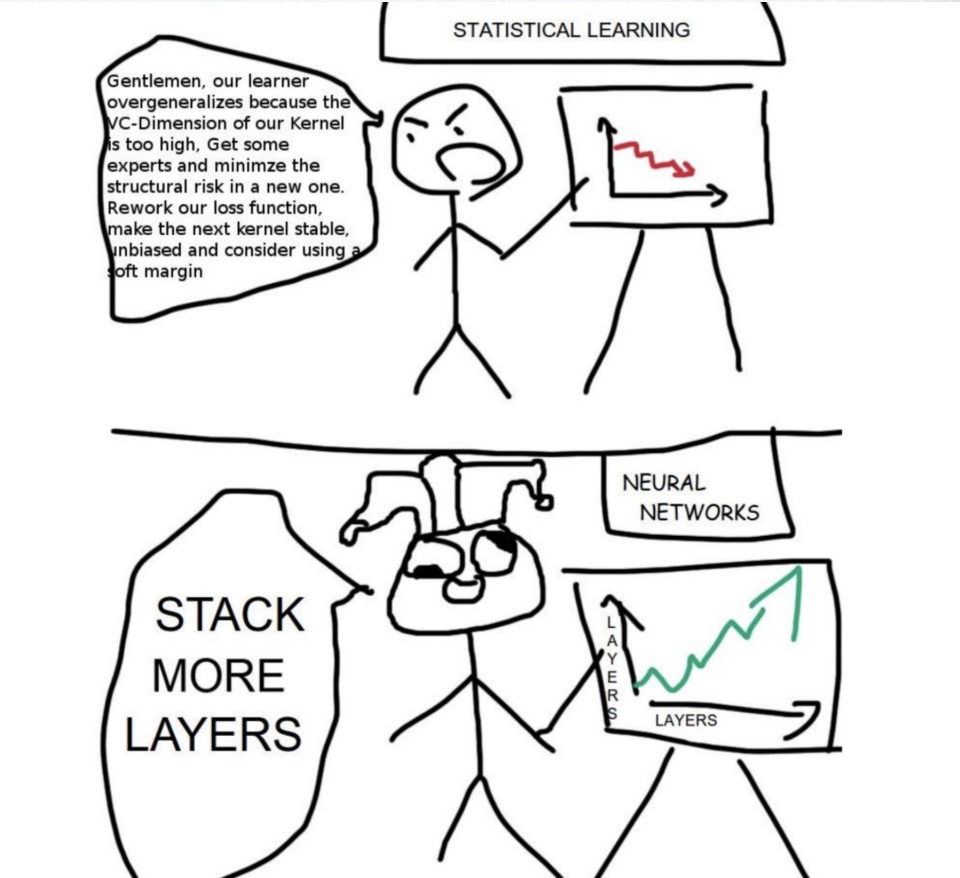
Context

Peak deep learning

"If you have a large big dataset, and you train a very big neural network, then success is guaranteed!"
Ilya Sutskever (NIPS 2014)

Peak deep learning

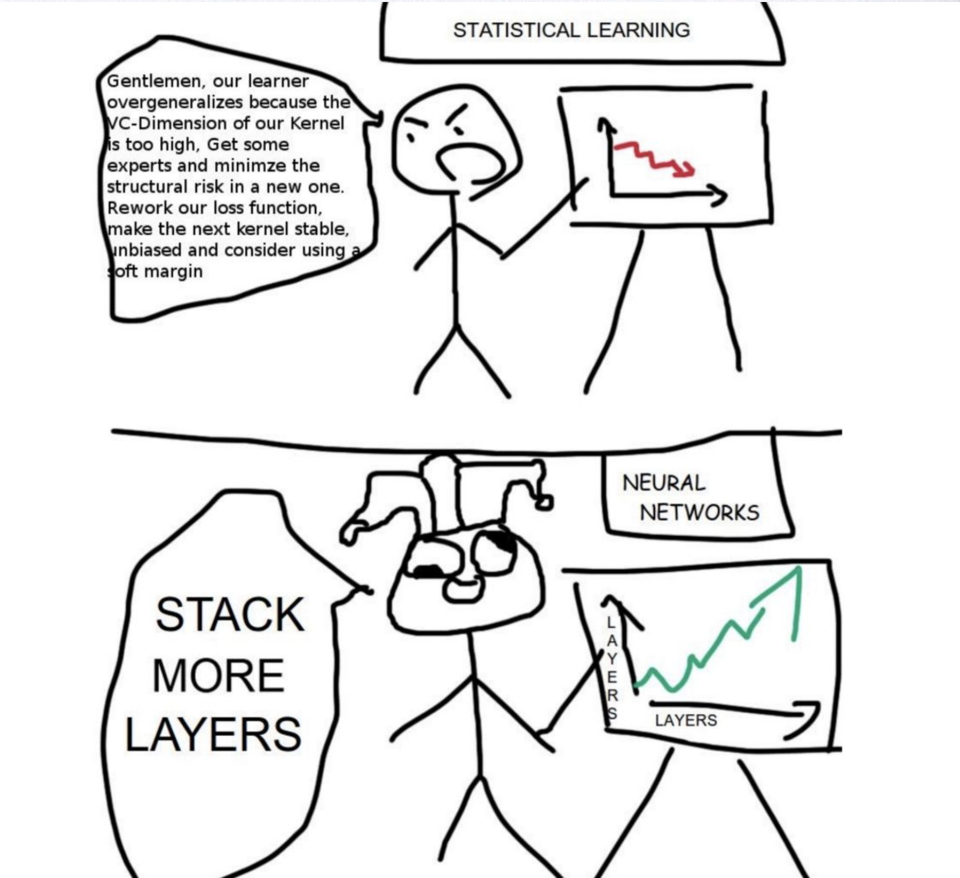
"If you have a large big dataset, and you train a very big neural network, then success is guaranteed!"
Ilya Sutskever (NIPS 2014)



Peak deep learning

“If you have a large big dataset, and you train a very big neural network, then success is guaranteed!”
Ilya Sutskever (NIPS 2014)

“If you formulate the right objective, and have enough capacity and compute, then success is guaranteed!”
Ilya’s law



DeepMind

Long ago, not so far away (mid-1800s, Cambridge):

First tutor: **"I'm teaching the most brilliant boy in Britain"**
Second tutor: **"Well, I'm teaching the best test-taker"**

Depending on the version of the story, the first boy was either **Lord Kelvin** or **James Clerk Maxwell**. The second boy indeed *scored highest* on the Mathematical Tripos, but is otherwise long forgotten.

Long ago, not so far away (mid-1800s, Cambridge):

First tutor: **"I'm teaching the most brilliant boy in Britain"**
Second tutor: **"Well, I'm teaching the best test-taker"**

Depending on the version of the story, the first boy was either **Lord Kelvin** or **James Clerk Maxwell**. The second boy indeed *scored highest* on the Mathematical Tripos, but is otherwise long forgotten.

Modern learning algorithms are outstanding test-takers

Intelligence is about more than taking tests
It's also about formulating useful problems

DeepMind

The problem problem*

Where do ~~problems~~ objectives come from?

* Tim Lillicrap

Where do problems come from?

Answer #1:

Someone packages a dataset into a loss function

e.g. ImageNet, CIFAR, MNIST

Where do problems come from?

Answer #1:

Someone packages a dataset into a loss function

e.g. ImageNet, CIFAR, MNIST

Answer #2:

Someone builds a task (that is, an environment sprinkled with rewards)

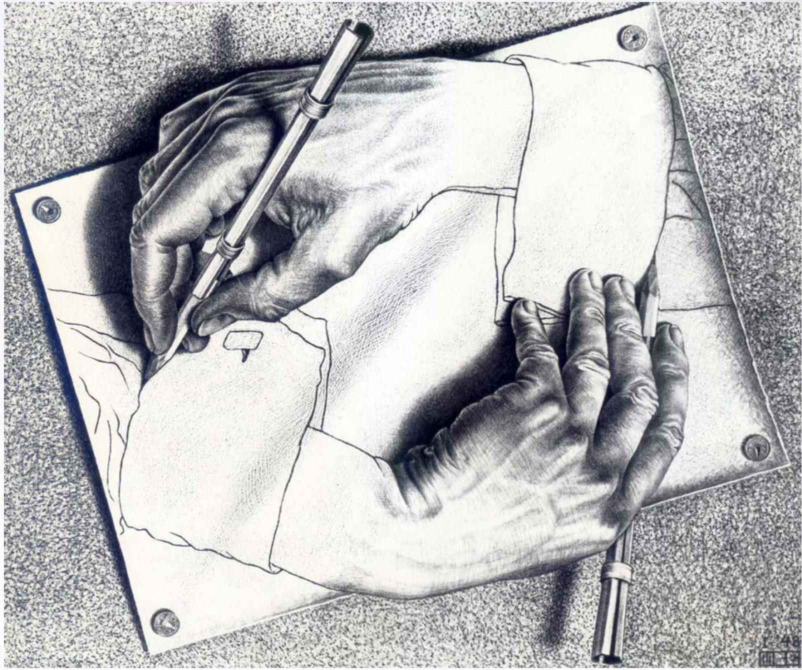
e.g. Arcade Learning Environment, DM-Lab, Open AI gym

Where do problems come from?

Answer #3:

Self-play in symmetric zero-sum games

The agent *is* the task -- create an outer loop that applies deep RL to itself



(Naive) self-play *is* an open-ended learning algorithm

It's pretty amazing

Algorithm 2 Self-play

input: agent $\mathbf{v}_1$
for $t = 1, \ldots, T$ **do**
  $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \phi_{\mathbf{v}_t}(\bullet)\right)$
end for
output: $\mathbf{v}_{T+1}$

(Naive) self-play *is* an open-ended learning algorithm

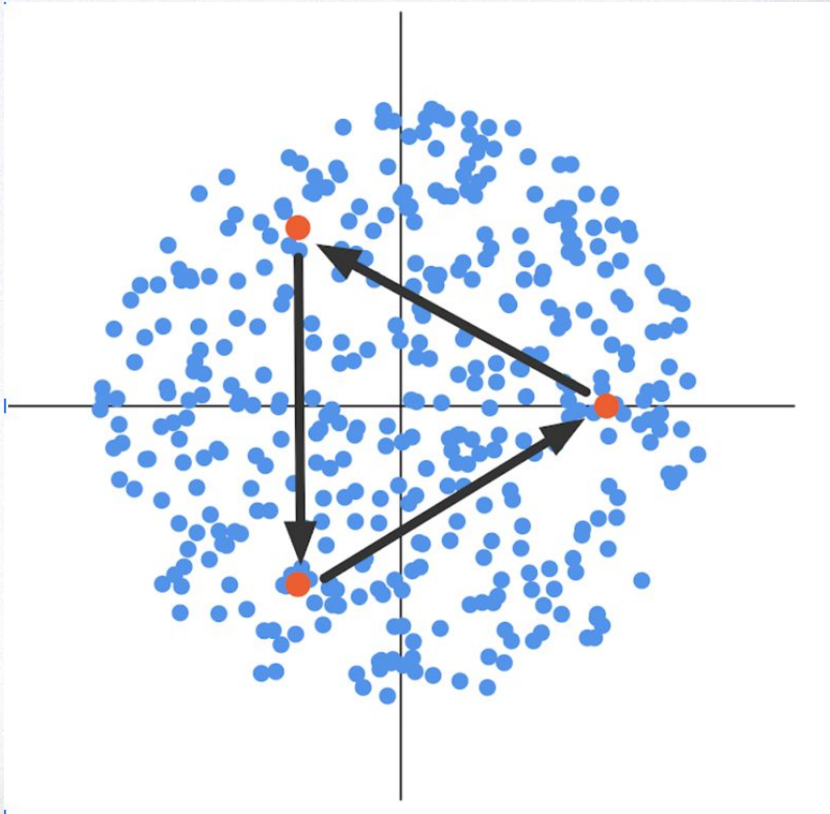
but ...

there are really simple examples

where it completely breaks down

It's **not** a general purpose learning

algorithm, *not even* for zero-sum games



DeepMind

Open questions

1. **When does self-play work, and why?**
    - If self-play is the answer then what is the question?
    - "works for some games but not others" isn't good enough

2. **What is the goal in (general) zero-sum games?**
    - Yes, to win, but against whom? The opponent matters
        - "Beating a pro" isn't a formal specification
    - Is there a general algorithm?

What does success look like?

#1: Supervised learning (e.g. ImageNet)

#2: Reinforcement learning on fixed environment (e.g. DM-Lab)

#3: Self-play (e.g. on Chess, Go, Shogi, … ; but **not** on everything)

What underlies machine learning's big wins?

1. **A transitive performance measure**
    - e.g. a loss function or discounted rewards
2. **An incremental improvement operator**
    - e.g. gradient descent, RL, evolutionary algorithms

What underlies machine learning's big wins?

1. **A transitive performance measure**
   - e.g. a loss function or discounted rewards
2. **An incremental improvement operator**
   - e.g. gradient descent, RL, evolutionary algorithms

What drives **self-play**'s success?

1. **Improvement (local):** $(A_{t+1} > A_t)$ and $(A_t > A_{t-1})$ and ...
2. **Transitivity (global):** A_{final} beats all previous agents

DeepMind

Basics

On the varieties of zero-sum games

transitive: “relative skill determines who wins”

cyclic: “every strategy has a counter-strategy”

DeepMind

Convex → Nonconvex → Transitive → Nontransitive

Convex objectives (ML in early 2000s): Converge to **unique global optimum**, for any initial condition and for any reasonable algorithm.

Nonconvex objectives (deep learning, ~2010...): Converge to a **local optimum** that depends on choice of initial condition and choice of algorithm.

Transitive games (e.g. self-play in Go, 2016...): Optimize and generate a sequence of objectives of the **same type** and **increasing difficulty**. Best agent at hardest objective is best overall.

Nontransitive games (e.g. AlphaStar league, 2019...): Optimum depends on who you compete with. Rather than finding a single best agent, should **invest in training a diverse "ecology" of agents.**

DeepMind

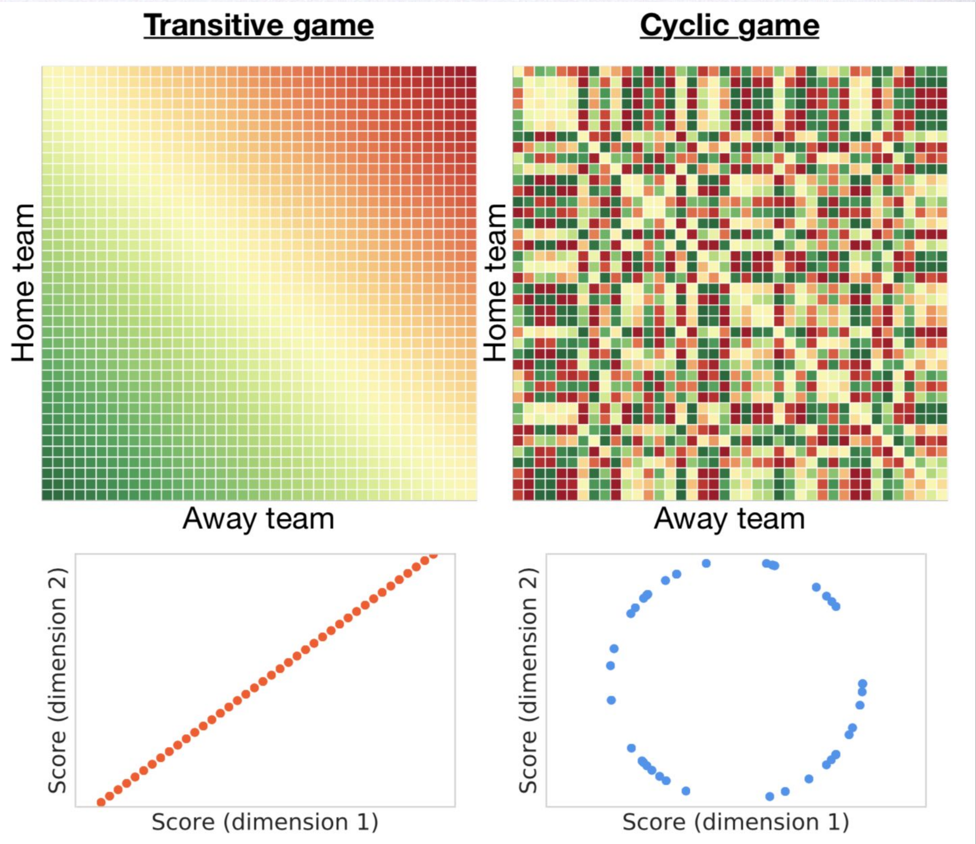
Functional-form game: a two-player symmetric zero-sum game, with differentiable parametrization:

$$\phi(\mathbf{v}, \mathbf{w})$$

Note: function approximator is “folded into” the game’s definition.

Theorem: Any symmetric zero-sum game decomposes into

[transitive] + **[cyclic]** components



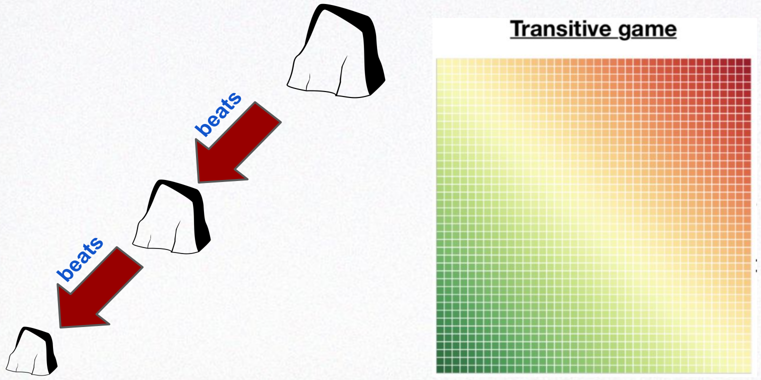
transitive: skill determines outcome

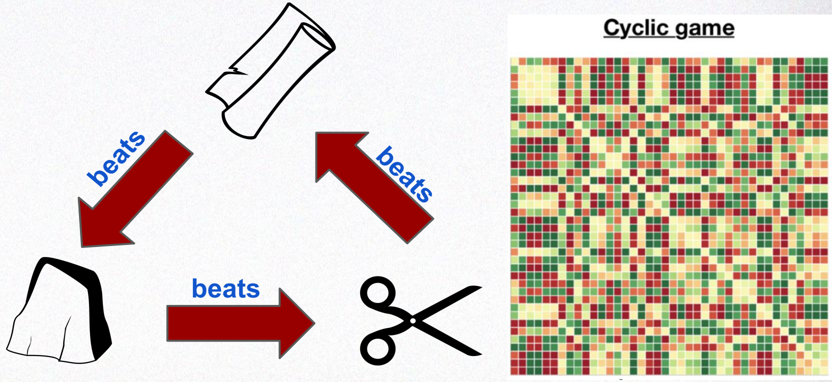
cyclic: every strategy has a counter-strategy

Transitive and cyclic games

	Good	Better	Best	Avg
Good	0	-1	-2	-1.5
Better	1	0	-1	0.0
Best	2	1	0	1.5

	Rock	Paper	Scissors	Avg
Rock	0	-1	1	0.0
Paper	1	0	-1	0.0
Scissors	-1	1	0	0.0





DeepMind

Transitive games: where the opponent doesn't matter

$$\phi(\mathbf{v}, \mathbf{w}) = f(\mathbf{v}) - f(\mathbf{w})$$

- f assigns **rating** to players
- outcome is difference in ratings

→ algorithm: **optimize against a fixed opponent**

$\text{argmax}_{\mathbf{v}}\, \phi(\mathbf{v}, \mathbf{w})$ for any, fixed, $\mathbf{w}$

DeepMind

Monotonic games: what self-play is "meant for"

$$\phi(\mathbf{v}, \mathbf{w}) = \sigma(\, f(\mathbf{v}) - f(\mathbf{w})\,)$$

- σ is a monotonic function (e.g. **Elo's rating model**)
- no learning signal if you train against a weak opponent (gradients vanish)
    - optimizing against a fixed opponent doesn't work

$\rightarrow$ algorithm: **self-play**

Algorithm 2 Self-play

input: agent $\mathbf{v}_1$
for $t = 1, \ldots, T$ **do**
  $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \phi_{\mathbf{v}_t}(\bullet)\right)$
end for
output: $\mathbf{v}_{T+1}$

Cyclic games: where naive self-play breaks down

A game is cyclic if $\int_W \phi(\mathbf{v}, \mathbf{w}) \cdot d\mathbf{w} = 0$

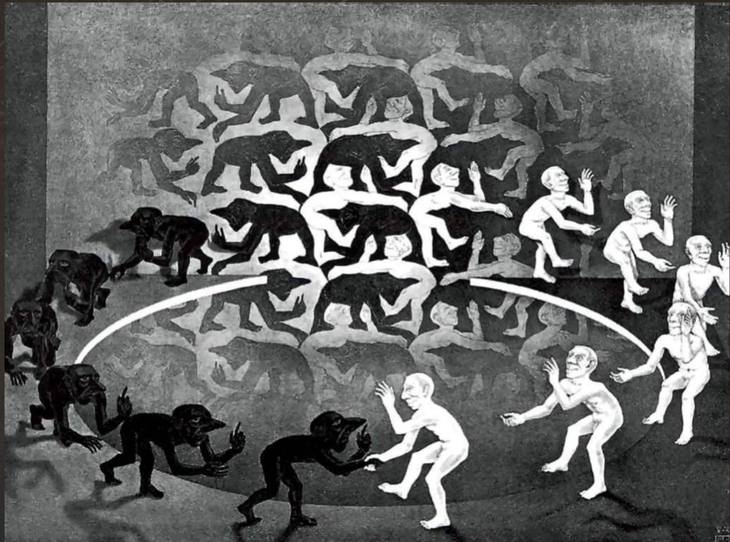
i.e **wins, against some opponents, are balanced by losses, against others.**

Implications:

- There is no best agent
- Measuring performance of individuals is nonsensical
  - which is better, paper or rock?

NEED: a **transitive objective**,

in **non-transitive games**!



How to convert **Agents** $\rightarrow$ **Objectives**

Definition: Gamescape is the convex hull of all objectives in a game

A game is a function that evaluates pairs of agents:

$$\Phi : W \times W \rightarrow R$$

$\Phi(\mathbf{v}, \mathbf{w})$ (e.g. probability that **v** beats **w**)

Fixing an opponent converts an **Agent** $\rightarrow$ **Objective function**

$$\mathbf{w} \rightarrow \Phi_{\mathbf{w}}(\ -\ ) := \Phi(\ -\ , \mathbf{w})$$

currying!

DeepMind

Gamescapes

Functional Gamescape: convex hull of all objective functions $\Phi_w(\,\text{-}\,)$

- lives in function space $\mathcal{F}(W, R)$
- intractable

Empirical Gamescape: convex mixture of all rows of evaluation matrix

- proxy for functional gamescape
- tractable

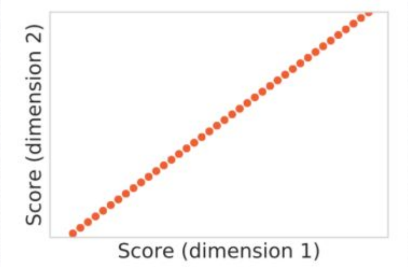
Fitness landscapes are a special case that arise when the game is transitive or monotone.

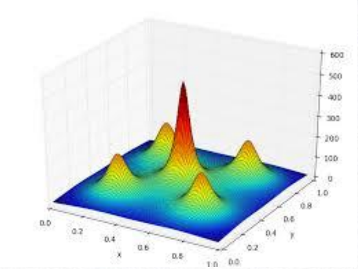
DeepMind

Landscape

Transitive: there's **one objective** ("improve skill") …

… so gamescape degenerates to **one dimensional fitness landscape**

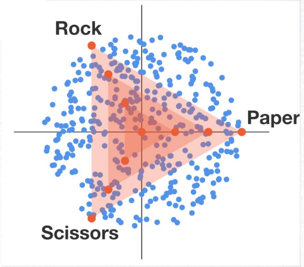




Gamescape

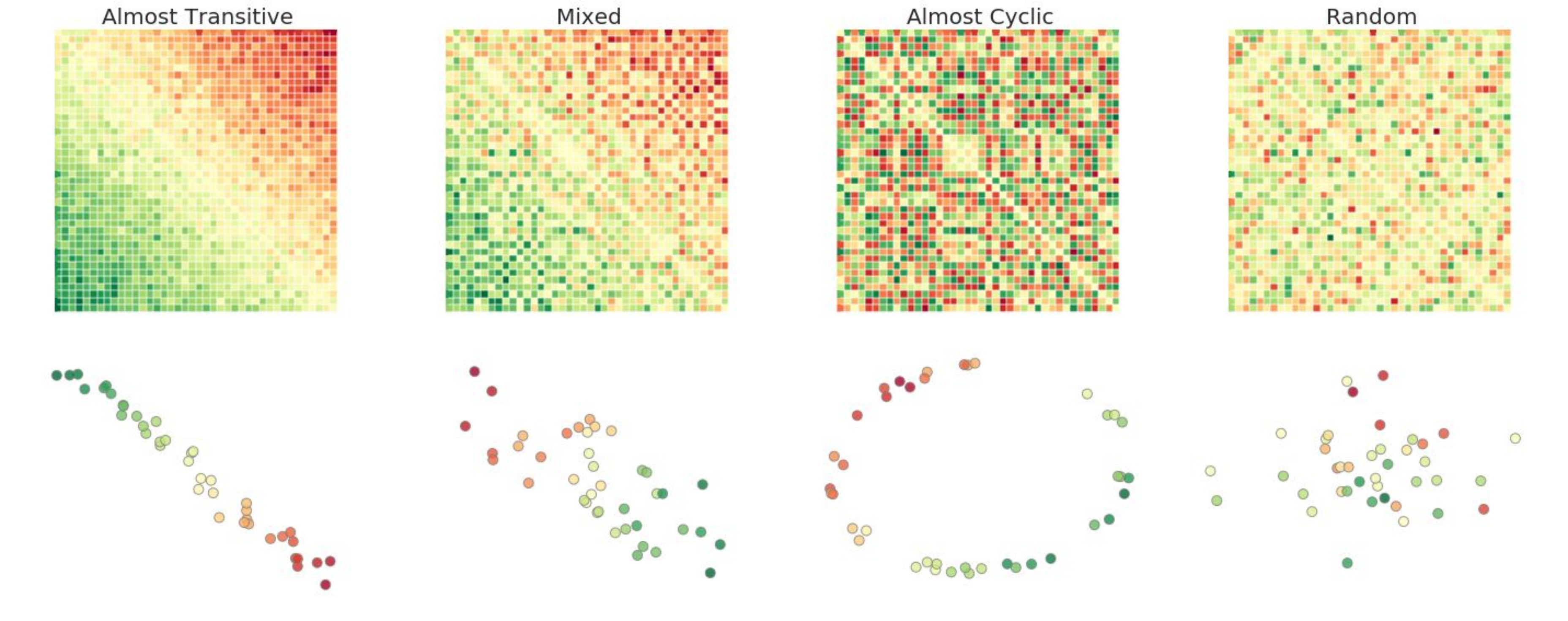
Nontransitive: different opponents are **different objectives** that pull in different directions

Gamescape is **multi-dimensional**



DeepMind

Gamescapes



mElo / Schur decomposition (roughly, in rank-2 case)

$$A_{n\times n} = W_{n\times 2} \cdot \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \cdot W^{T}_{2\times n}$$

{ agents } $\to$ { rows of **W** }

Entries of each row are the agent's **multi-dimensional Elo** scores

(let's forget about sigmoids)

Looooooong cycles

Is transitive geometry always 1-dim? **Yes**

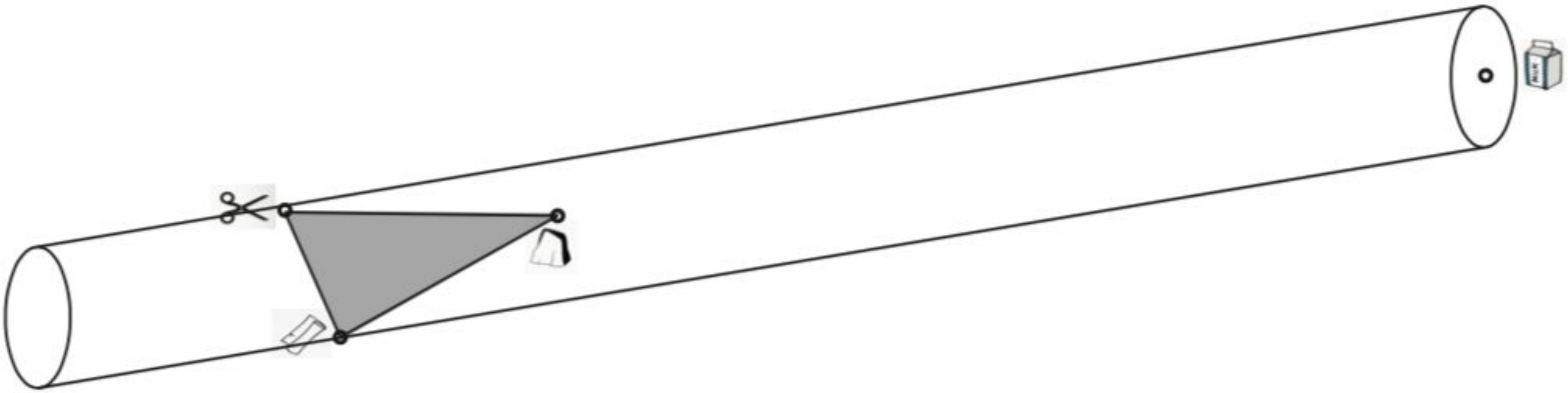
Is cyclic geometry always 2-dim? **No**

Rock → Paper → Scissors → Fire → Water → Air → Ether → Milk → … → Rock

length(cycle) = **{** 2n or 2n+1 **}** → **rank(A)** = **{** 2n-2 or 2n **}**, respectively.

What is the dimension of SC2's gamescape? No idea.

Ceci n’est pas une pipe



DeepMind

So what's **the transitive objective** in nontransitive games?

Individual performance is **meaningless** in nontransitive games.

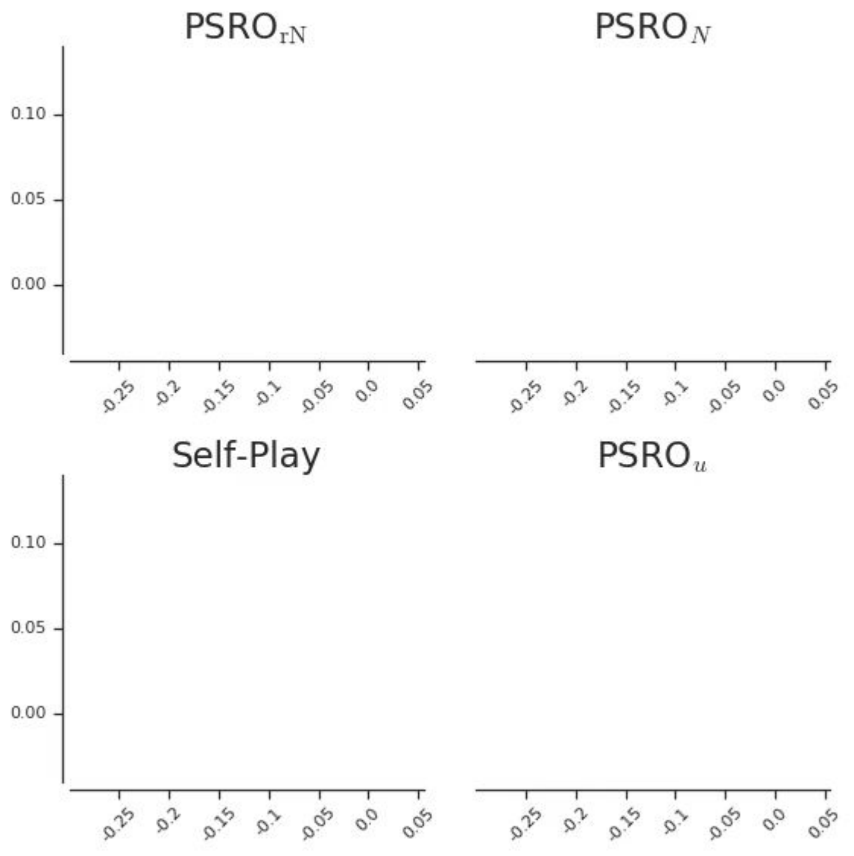
"Which is better, rock or paper?"

Idea: population-level objectives!

Grow the gamescape, to get "diverse, effective agents"

→ find strategic dimensions of game

→ and best ways of executing them



Population-level performance

Definition:

Given (m x n) evaluation matrix A for populations P and Q.

Let (p, q) be any Nash Equilibrium on zero-sum meta-game on A.

Relative performance of populations:

$$\mathbf{v(P, Q) := p^T.A.q}$$

(doesn't depend on choice of Nash).

Intuition:

- I pick my mixture of champions
- You pick yours (simultaneously)
- We play them out and
- See who wins, how much, in expectation

Population-level performance

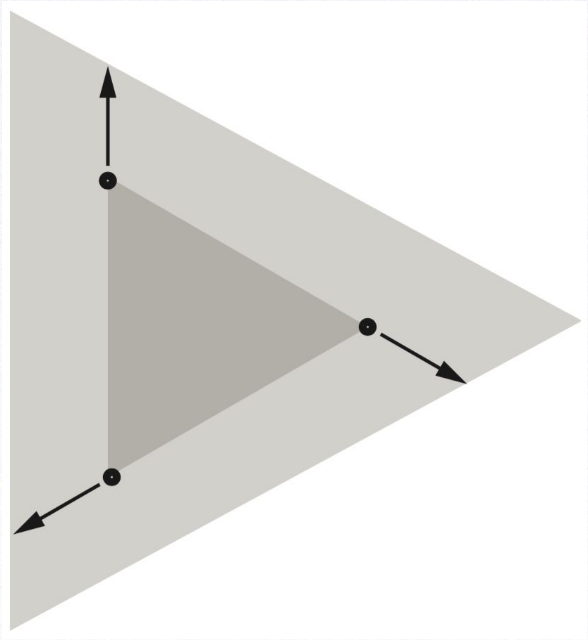
Lemma (transitivity for populations):

If agents in P are convex combinations of agents in Q then

$$v(P, Q) <= 0$$

Implication:

Algorithms that "grow polytope" are guaranteed to improve the population-level performance.



Two algorithms: one old and one new

Two ways to **track growth** ...

1. Population performance
2. Effective diversity

... and two algorithms

1. Response to Nash (double oracle)
   - train against Nash
2. Response to **rectified** Nash
   - train against agents in Nash that you beat
   - game-theoretic niches

Algorithm 3 Response to Nash (PSRO$_\mathrm{N}$)

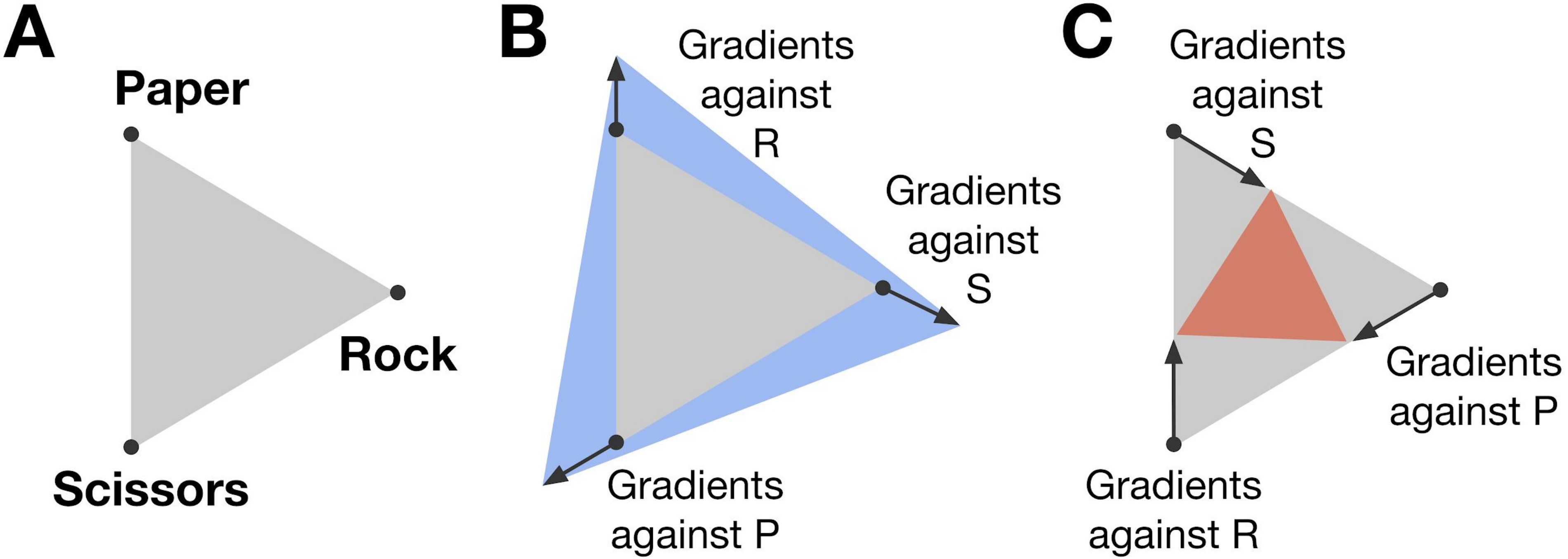
input: population $\mathfrak{P}_1$ of agents
for $t = 1, \ldots, T$ **do**
  $\mathbf{p}_t \leftarrow$ Nash on $\mathbf{A}_{\mathfrak{P}_t}$
  $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \sum_{\mathbf{w}_i \in \mathfrak{P}_t} \mathbf{p}_t[i] \cdot \phi_{\mathbf{w}_i}(\bullet)\right)$
  $\mathfrak{P}_{t+1} \leftarrow \mathfrak{P}_t \cup \{\mathbf{v}_{t+1}\}$
end for
output: $\mathfrak{P}_{T+1}$

Algorithm 4 Response to rectified Nash (PSRO$_\mathrm{rN}$)

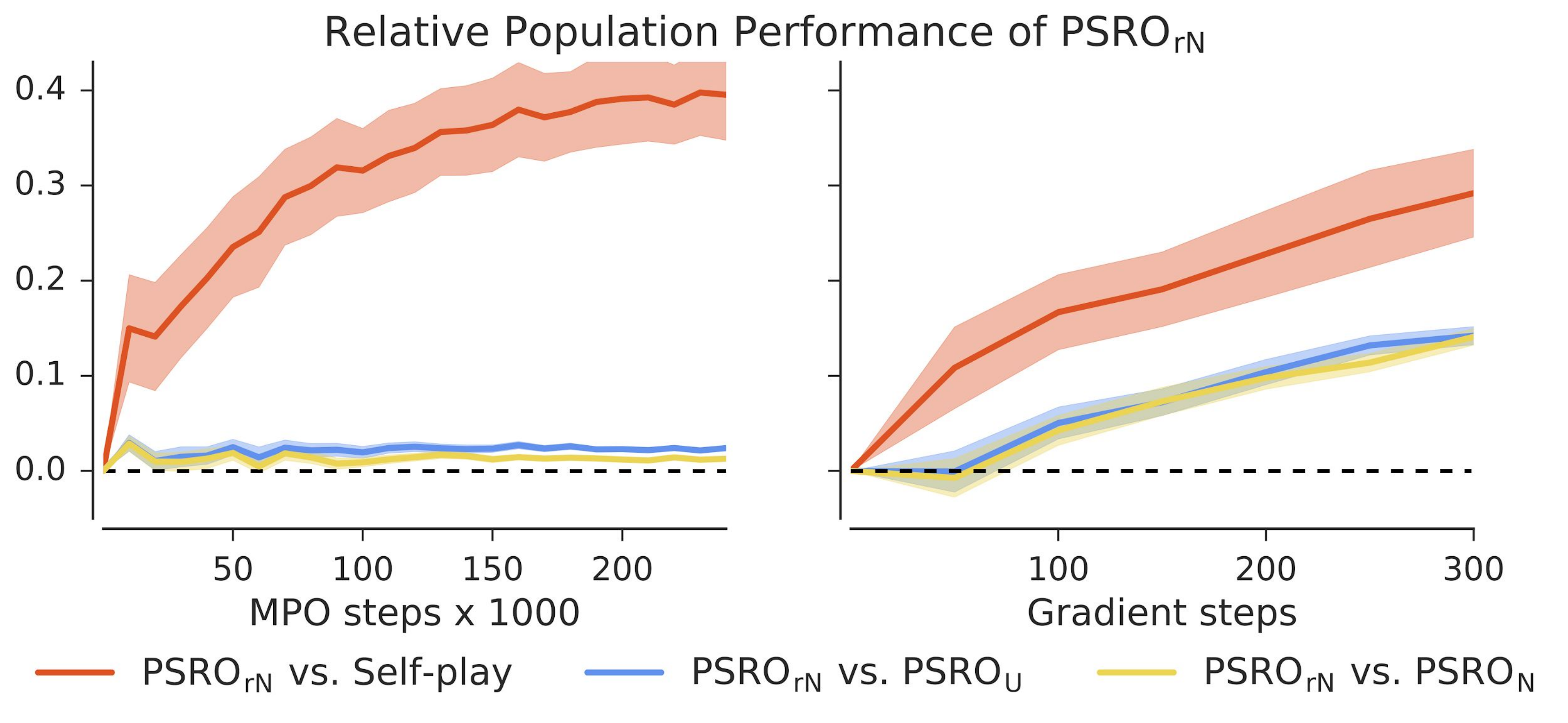
input: population $\mathfrak{P}_1$
for $t = 1, \ldots, T$ **do**
  $\mathbf{p}_t \leftarrow$ Nash on $\mathbf{A}_{\mathfrak{P}_t}$
  for agent $\mathbf{v}_t$ with positive mass in $\mathbf{p}_t$ **do**
    $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \sum_{\mathbf{w}_i \in \mathfrak{P}_t} \mathbf{p}_t[i] \cdot \lfloor \phi_{\mathbf{w}_i}(\bullet) \rfloor_+\right)$
  end for
  $\mathfrak{P}_{t+1} \leftarrow \mathfrak{P}_t \cup \{\mathbf{v}_{t+1} : \text{updated above}\}$
end for
output: $\mathfrak{P}_{T+1}$

DeepMind

Response to Rectified Nash

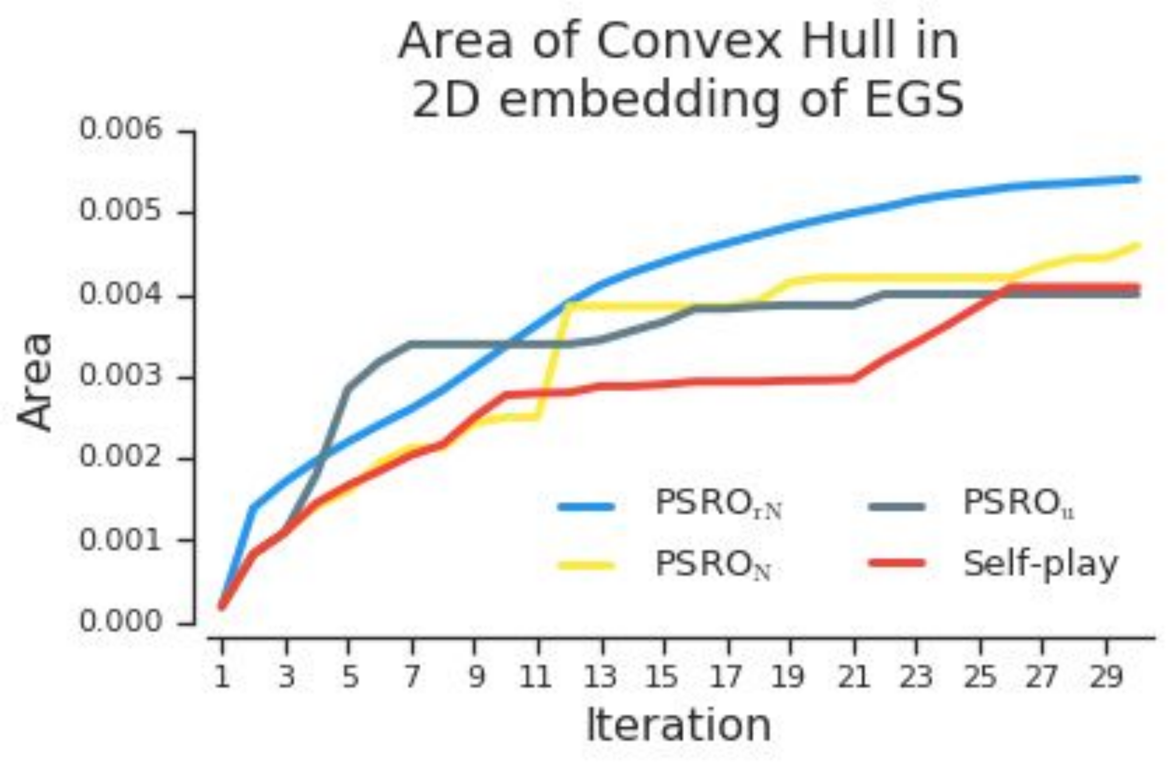


Performance on Blotto and differentiable Lotto



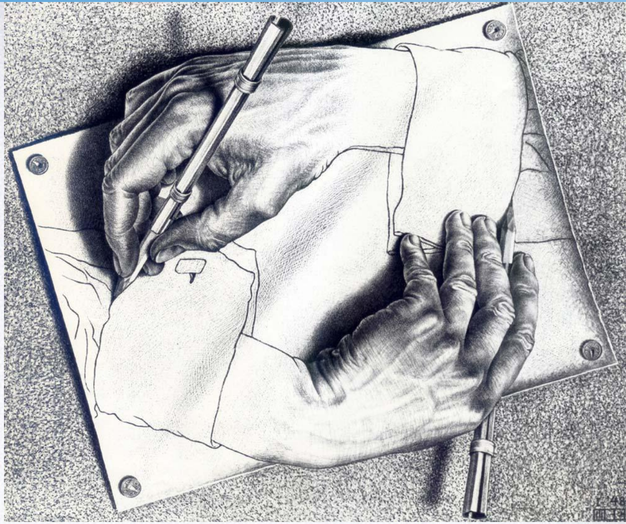
DeepMind

Growth of gamescapes in differentiable Lotto

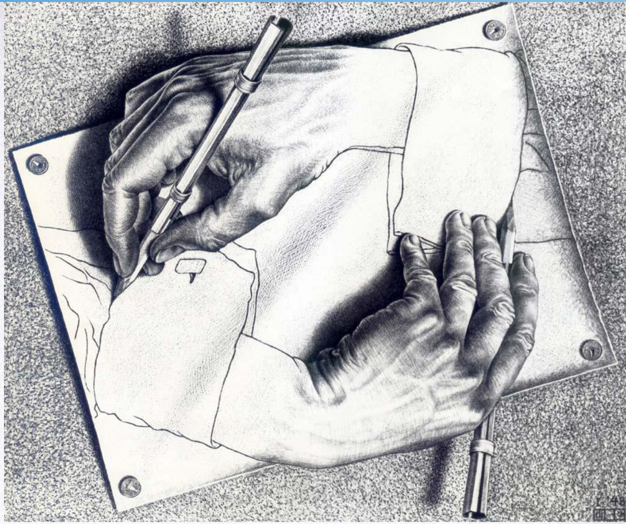


Discussion

Back to the problem problem



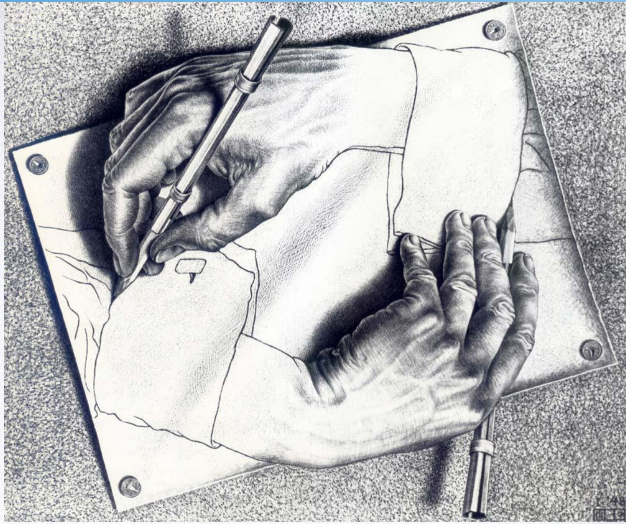
Back to the problem problem



1. **How can algorithms formulate problems?**
2. **Which problems are useful?**

DeepMind

Back to the problem problem



1. **How can algorithms formulate problems?**
2. **Which problems are useful?**

Formulating problems is an **inter-agent** phenomenon

- in a zero-sum game, **"*my* behavior is *your* problem"**

Usefulness is **semantically grounded** in **winning** in ZSGs

- so **Nash formulates useful problems** via, say, response-to-Nash, response-to-rectified-Nash

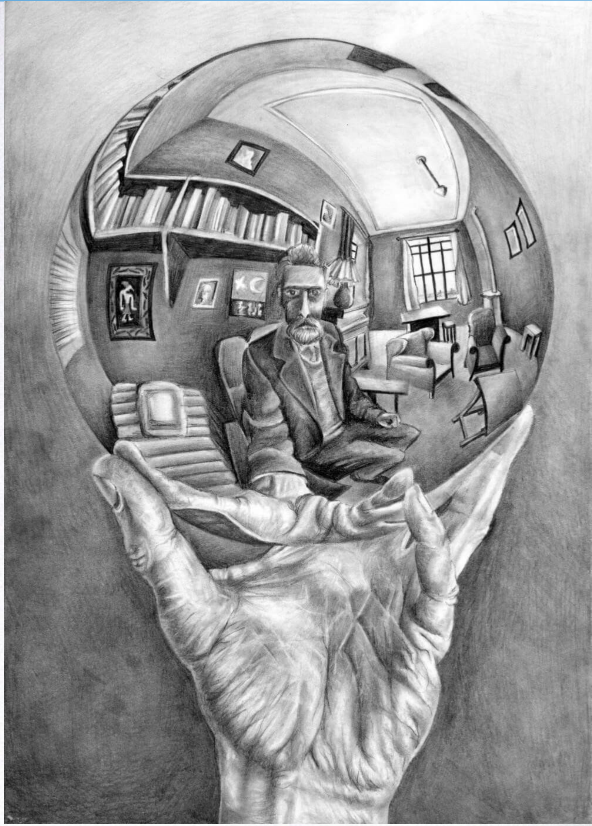
DeepMind

Comment on human problem posing

- The model describes posing-and-solving problems as a population-level activity

- This is how it works for humans
  - At least, population aspects are under-rated;
  - individuals get credit for work of populations

- Even humans like Einstein didn't pose **new** problems
  - "Everyone" knew Maxwell's equations aren't invariant under Galilean transforms
  - Einstein "just" took the problem seriously



Summary

Success of modern machine boils down to

- applying *many* **incremental improvements**
- to **transitive objectives**, so improvements **add up**

This talk:

- tools for **formulating objectives in a nontransitive setting**
- and **algorithms for optimizing them**

details: https://arxiv.org/abs/1901.08106