Reevaluating Evaluation

Balduzzi et al

Motivation

- Evaluation on problems of common interest are the key drivers in ML
  - Go
  - Atari
  - Minecraft
  - MNIST
  - Etc
- Two main bodies of work:
  - Optimize new algorithms w.r.t these datasets
  - Propose a new benchmark

Adversarial Attacks

- Are our models really robust?

- How can we test against all attacks?

Granny Smith	85.6%
iPod	0.4%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.1%

Granny Smith	0.1%
iPod	99.7%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.0%

Self Play

- Agents train against copies of themselves

- We have trained agents to get superhuman play in e.g. Hanabi

- Policies learned through self-play:
    - may adopt arbitrary conventions

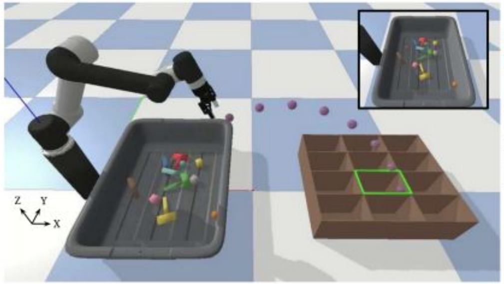
    - Do not play well with others

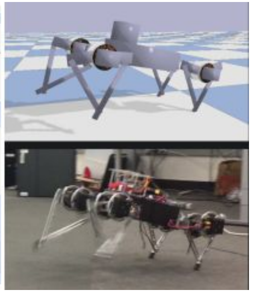
Many Competing Testbeds

MuJoCo

Gymnasium

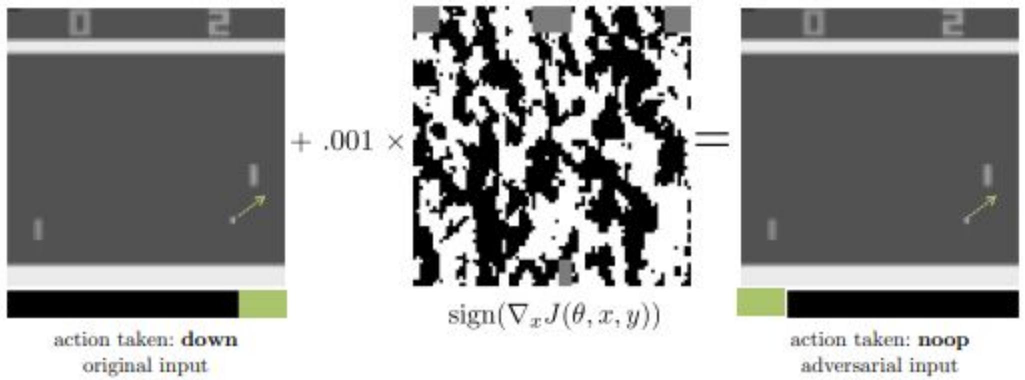
Gymnasium-Robotics





Common Thread

- Current methods do not account for non-stationary evaluation settings
- When the evaluation distribution is different from the training distribution, algorithms fail



Motivation

- Results are not used to evaluate and optimize **evaluations themselves**
- Therefore, our algorithms can be exploited
  - Adversarial attacks
    - We don’t know what attacks to test against
  - Self-Play
    - Can only test against each other
  - Proliferation of testing suites
    - Leads to cherry-picking what environment fits our algorithm the best

Guiding Questions: What does it mean to optimize an evaluation?

Do tasks/agents test what we think they test?

When is a task/agent redundant?

Which tasks (and agents) matter the most?

Solution

We want an **algorithm** that:

- automatically adapts to redundancies in evaluation data, so that results are not biased by the incorporation of easy tasks or weak agents

Deepmind puts forward one such algorithm called **Nash Averaging** where we play a game between:

- agents and tasks / datasets
- agents and other agents

Nash Averaging

- Play a meta-game on evaluation data
- The fundamental algebraic structure of tournaments and evaluation is antisymmetric
- **Answers Q2 and Q3** -- which tasks and agents do and do not matter is determined by a meta-game

Nash Averaging

Comes in two flavors:

- Agent vs Task(s)
  - Training an agent to e.g., solve atari games
  - Relatively easy to say solved vs unsolved vs % solved
- Agent vs Agent(s)
  - Training an agent to beat other agents at a specific game
  - Performance between agents is often quantified using **Elo** ratings

Rock-Paper-Scissors

$$\mathbf{A}_{ij} := \log \frac{p_{ij}}{1-p_{ij}}$$

- Zero-Sum Game
- Contains a **cycle**
  - A → B
  - B → C
  - C → A

- Values here are **log probabilities** of the ratio of win to loss

$\mathbf{A}$	A	B	C
A	0.0	4.6	-4.6
B	-4.6	0.0	4.6
C	4.6	-4.6	0.0

Rock-Paper-Scissors

$$\mathbf{A}_{ij} := \log \frac{p_{ij}}{1-p_{ij}}$$

- Matrix is **antisymmetric**
- A_ij + A_ ji = 0
- A + A^T = 0

A	A	B	C
A	0.0	4.6	-4.6
B	-4.6	0.0	4.6
C	4.6	-4.6	0.0

Nash Averaging (The Game, Very High Level)

- Two agents -- meta-players -- pick ‘teams’ of agents
- Their payoff is the expected log-odds of their respective team winning under the joint distribution
- The value of the metagame is zero
  - Nash equilibria are teams that are unbeatable in expectation

Nash Averaging

- Given antisymmetric logit matrix A (real or approximated)
- a two-player metagame with payoffs for the row and column meta-players
  - $\mu 1(p, q) = p^T Aq$
  - $\mu 2(p, q) = p^T Bq$
- $B = A^T$

What team would you build?

- Nash equilibria are teams that are unbeatable in expectation

	agent A	agent B	agent C	**Elo**
agent A	0.5	0.9	0.1	0
agent B	0.1	0.5	0.9	0
agent C	0.9	0.1	0.5	0

Nash Averaging in RPS

- In rock-paper-scissors, the only unbeatable-on-average team is the uniform distribution over the different players

- $p^* = q^* = [⅓, ⅓, ⅓]$

	agent A	agent B	agent C	**Elo**
agent A	0.5	0.9	0.1	0
agent B	0.1	0.5	0.9	0
agent C	0.9	0.1	0.5	0

- When is a task/agent redundant?
- Which tasks (and agents) matter the most?

What agent is the best now?

	agent A	agent B	agent C_1	agent C_2	**Elo**
agent A	0.5	0.9	0.1	0.1	-63
agent B	0.1	0.5	0.9	0.9	63
agent C_1	0.9	0.1	0.5	0.5	0
agent C_2	0.9	0.1	0.5	0.5	0

Properties of NA

CLAIM:

- **The MaxEnt Solution (p*, p*) is invariant to additional copies of an agent**

- I.e., adding redundant copies of an agent or task to the data should make no difference

There are many NE, which one to pick?

- row and column meta-players

  For **A** there is a unique NE at:

  - ($p*$, $p*$) solves
    - $\underline{\max_p}\ \underline{\min_q}\ p^T A q$
  - This NE has greater entropy than any other

What agent is the best now?

	agent A	agent B	agent C_1	agent C_2	**Elo**
agent A	0.5	0.9	0.1	0.1	-63
agent B	0.1	0.5	0.9	0.9	63
agent C_1	0.9	0.1	0.5	0.5	0
agent C_2	0.9	0.1	0.5	0.5	0

- Could say that B is better, but that's a quirk of the evaluation data

What team would you build?

	agent A	agent B	agent C_1	agent C_2	**Elo**
agent A	0.5	0.9	0.1	0.1	-63
agent B	0.1	0.5	0.9	0.9	63
agent C_1	0.9	0.1	0.5	0.5	0
agent C_2	0.9	0.1	0.5	0.5	0

The Upshot

- Objectively test algorithms against:
  - any dataset
  - all datasets
  - all tasks
  - other agents

The upshot upshot

- Provides a rigorous method of choosing how to sample parents in an evolutionary algorithm that preserves diversity!

- Can we use this to co-optimize agents and tasks?
    - Combine agent learning (RL) with Automatic Environment Design