Emergent Complexity via Multi-Agent Competition:

`https://arxiv.org/pdf/1710.03748`

Paper Summary Notes

1 Main Ideas

Central Idea of the Paper: *Competitive multi-agent environments with self-play can produce behaviours that are far more complex than the environment itself, and come with a natural curriculum: for any skill level, an environment full of agents of this level will have the right level of difficulty*

This paper focuses on self-play in the domain of continuous control.

Instead of engineering complexity into the environment design by engineering dense reward functions, allow complexity to emerge due to the presence of other learning agents

1.1 Background

- in general, training an agent to perform a highly complex task requires a highly complex environment, which may be difficult to create
- *competitive multi-agent self-play environments alleviate this difficulty*:
  - even very simple competitive multi-agent environments can produce extremely complex behaviours (example: chess. Rules are simple, as the agent gets better, the task becomes harder)
  - when trained with self-play, the competitive multi-agent environment provides the agent with perfect curriculum
    - no matter how weak or strong, an agent is, a diverse set of agents of similar skill will provide the right challenge, facilitating rapid learning and avoiding getting stuck

2 Review

2.1 Multi-agent markov games

- A *Markov game for N agents* is a POMDP defined with:
  - $\mathcal{O}^1 \times ... \times \mathcal{O}^N$: observation of each agent
  - $\mathcal{A}^1 \times ... \times \mathcal{A}^N$: action of each agent
  - $\mathcal{S}$ global state
  - $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times ... \times \mathcal{A}^N \mapsto \mathcal{S}$ the transition function
  - r^i reward for agent i
  - $\pi_{\theta^i} : \mathcal{O}^i \times \mathcal{A}^i \longrightarrow [0, 1]$ stochastic policy for agent i

3 Competitive Environments

3.1 Run to Goal

- agent's start by facing each other in a 3d world and they have goals on the oposite side of the world. The agent that reaches the goal first wins. Reward: +1000 if first to goal, -1000 if second, and -1000 if neither reaches goal

3.2 You shall not pass

- Same as run to goal, but one of the agetns is a *blocker*, has the objective of blocking the other agent from reaching it's goal while not falling down. Blocker gets +1000 if opponent doesn't reach goal, 0 if not standing, and -1000 if opponent get's to goal.

3.3 Sumo

- In a round arena, goal of each agent is to knock the other to the ground or push them out of the ring. Winner gets +1000, loser -1000, and tie gies both -1000

3.4 Kick and Defend

- Standard penalty shootout. +1000 to attacker if scores, -1000 to goalie. Two additional rewards to goalie: +500 if still standing at the end of game and additional +500 if the goalie made contact with the ball [1]

4 Training Competitive Agents

4.1 High Level

- PPO algorithm
- Decentralized training (large batch sizes to counterbalance variance problem)
- GAE estimate from the full rollouts (not truncated steps), important since competition reward is sparse given at the end of episode

4.1.1 Policy and Value Functions

- MLP (2 hidden layer, 128 units each) for *run-to-goal* and *you shall not pass*
- LSTM (128-dimensional embedding, single layer lstm (10 step), output projected to action dimension) for *sumo* and *kick and defend*
- Gaussian policies with mean given by the output of the network and diagonal co-variance as trainable parameters
- Sperate policy and value networks

4.2 Exploration Curriculum

The success of agents in competitive games requires the agents to occasionally solve the task by random actions

How to learn motor skills which are a prerequisite for solving the games?

- They use simple dense rewards at each step to allow the agents to learn basic motor skills at the beginning of training they call this the **exploration reward**
- The **exploration reward** is annealed to 0 over the course of training, towards the **competition reward**, which are the sparse rewards described above

$$r_t = \alpha_t s_t + (1 - \alpha_t)\mathbb{1}[t == T]R \tag{1}$$

- s_t is the exploration reward
- R is the competition reward given at termination

[1]More realistic looking defending behaviours

- $\alpha_t : 1 \longrightarrow 0$ over the course of training

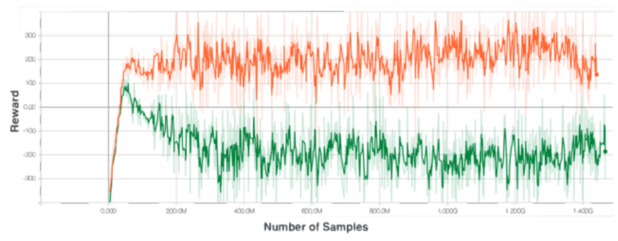
There are no dense exploration rewards for the complex emergent behaviours

4.3 Opponent Sampling

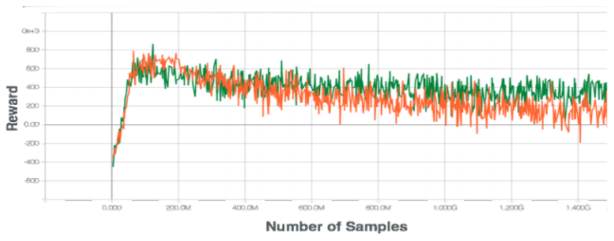
In the competitive multi-agent framework, all agents are simultaneously training in opponent pairs.

- Training against most recent opponent leads to imbalance in training, where one agent becomes more skilled than the other early in training
- Training against *random old versions* of the opponent works much better

For self-play this means that the policy at any time should be able to defeat random older versions of itself, thus ensuring continual learning



(a) Latest available opponent



(b) Random old opponent

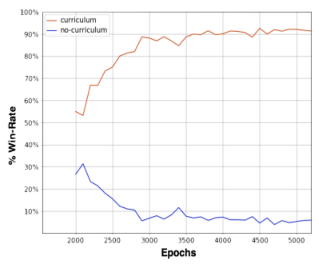
5 Experiments

5.1 Learned Behaviours

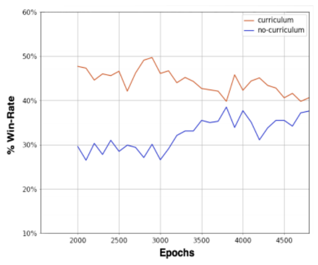
- Videos in paper show interesting complex emergent behaviours
- Run-to-goal: blocking, standing strongly, using legs to topple the opponent, and running towards the goal
- You shall not pass: blocking agent learns to block by raising it's hands while the other agent learns to duck in order to cross
- Sumo: "faking" behaviours, trying to deceive the opponent, know opponent using head
- Kick and defend: fooling behaviour in the kicker motions where it moves left and right quickly once close to the ball to fool the defender. THe defender learned to defend by moving in response to the motion of the kicker and using its hands and legs to obstruct the ball

5.2 Effect of Exploration Curriculum

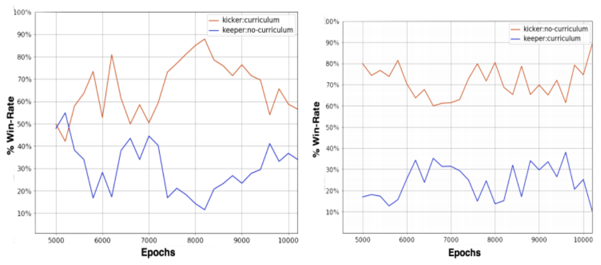
- exploration is crucial for learning, as otherwise the agents are unable to explore the sparse competition reward
- however, learned behaviours are mostly a result of the natural curriculum arising out of hte multi-agent competition, and not due to the dense exploration reward



(a) Humanoid Sumo



(b) Ant Sumo



(c) Kick-and-Defend

- Non-annealed agent's lose to annealed agents, suggesting that the exploration reward alone cannot explain the emergent complexity

5.3 Effect of Opponent Sampling

- consider what happens when we alter the uniform random opponent window of sampling

$$\theta_{opponent} \longleftarrow Uniform(\delta v, v) \tag{2}$$

  - v = latest parameter weights
  - $\delta \in \{1.0, 0.8, 0.5, 0\}$

δ	1.0	0.8	0.5	0.0	$\mathbb{E}$[Win]
1.0	-	0.26	0.13	0.37	0.25
0.8	0.46	-	0.22	0.52	0.40
0.5	0.59	0.58	-	0.73	**0.63**
0.0	0.55	0.36	0.16	-	0.35
$\mathbb{E}$[Loss]	0.53	0.40	**0.17**	0.54	-

(a) Humanoid Sumo

δ	1.0	0.8	0.5	0.0	$\mathbb{E}$[Win]
1.0	-	0.37	0.35	0.29	0.34
0.8	0.36	-	0.38	0.33	0.36
0.5	0.36	0.39	-	0.33	0.36
0.0	0.51	0.49	0.49	-	**0.50**
$\mathbb{E}$[Loss]	0.41	0.42	0.41	**0.32**	-

(b) Ant Sumo

- training against latest opponent leads to worst performance
- uniform random over the entire history has highest win rate for *Ant* and training with $\delta = 0.5$ has highest win rate with *Humanoid*
  - And with random policy on a small arena is still a good opponent while a Humanoid with random policy is unable to stand and thus always looses in a few steps

5.4 Learning Robust Policies

Preventing over fitting to competitive environments

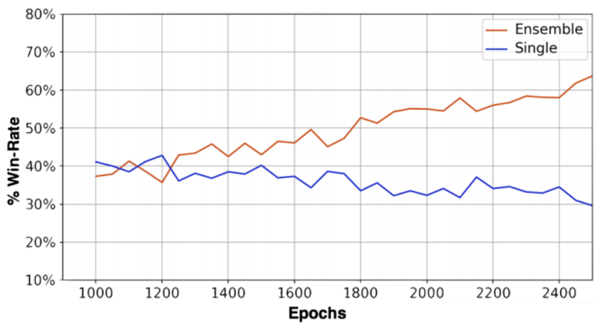
5.4.1 Randomization in World

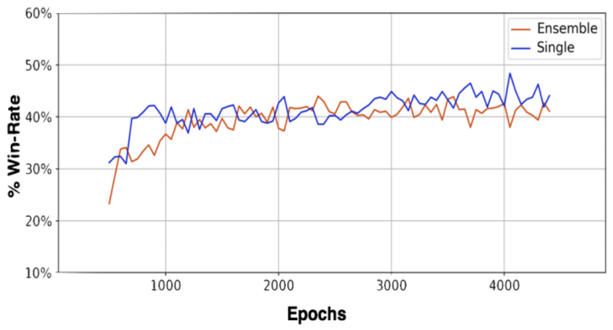
- Introduce randomness into the environment
  - random ball position, arena radius etc
  - Found that increasing randomization throughout training did best

Preventing over fitting to a particular opponent behaviour

5.4.2 Competing against ensemble of policies

- Aggregate pool (ensemble) of policies and select one at random for each rollout
  - similar to multi-task learning
  - creates a natural distribution for multi-taks learning, where each good opponent behaviour is a task: (learn a set of strategies, and apply the right one)





- Train a pool of 3 ensemble policies and take the highest average return in last 500 iteration as the representative *ensemble*
- Train three independent policies via self-play and again take the highest average return in last 500 iteration as the representative of *single*
- **Humanoid [left]**: makes a big difference, (harder problem)
- **Ant [right]** no difference