Evolution Strategies using TensorForce

LSDPO (2017/2018) Project Presentation
Tudor Tiplea (tpt26)
What is TensorForce?

- Open-Source Reinforcement Learning Library
- Built on top of TensorFlow
- Provides a strict separation of agents, environments and update logic
- A number of out-of-the-box state-of-the-art RL algorithms already implemented:
  - A3C, DQN, Double-DQN, etc.
Why is it useful?

- Suppose you want to employ deep RL to control some aspect of your system
- Lots of resources and introductions to theoretical RL
- Also, lots of starter agents and their applications available online
- However, much of the existing code has several disadvantages. E.g.:
  - Tight integration with simulation platforms
  - Fixed network architectures
- TensorForce provides the out-of-the-box agents, but they are highly configurable
- It also employs a shift of paradigm: the environment calls out to the agent when it needs a decision rather than the other way around
Evolution Strategies

- An alternative to MDP-based RL techniques such as Q-learning or Policy Gradient
- A heuristic search procedure inspired by natural evolution
- At each iteration (generation):
  - Perturb a population of parameter vectors
  - Evaluate the objective function for each
  - Best performing ones are recombined to form the population at the next step
- Can be scaled and parallelised between multiple workers, with limited intercommunication
Non-parallelised algorithm

**Algorithm 1 Evolution Strategies**

1. **Input:** Learning rate $\alpha$, noise standard deviation $\sigma$, initial policy parameters $\theta_0$
2. **for** $t = 0, 1, 2, \ldots$ **do**
3. Sample $\epsilon_1, \ldots, \epsilon_n \sim \mathcal{N}(0, I)$
4. Compute returns $F_i = F(\theta_t + \sigma \epsilon_i)$ for $i = 1, \ldots, n$
5. Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
6. **end for**
Work plan

- Connect the existing weight update part of the simple ES algorithm to a model, producing the first agent
- Implement the parallelised ES agent to run in multi-threaded manner on my laptop
- Evaluate the two on simple environments (due to long training time) from OpenAI Gym
- Compare against already implemented agents such as A3C and DQN
Possible extensions

- First, set up an EC2 instance using a student account
- Evaluate the implemented agents in more complex environments, such as Atari 2600 games
- Extend the parallelised ES agent to run in a distributed manner, across multiple machines
- Evaluate the distributed ES agent
Questions

Thank you!
References
