Abstract

Hyper-parameter optimization using **CMA-ES** significantly improved the performance of **Reinforcement Learning Algorithms** compared to the baselines OpenAI provided

CMA-ES as Hyperparameter Optimizer for Reinforcement Learning Algorithms

Yoonwoo Kim, Takahiro Shinozaki
Tokyo Institute of Technology

Overview

- RL Algorithms are notoriously sensitive to hyperparameters.
- It takes laborious effort of the researchers to achieve optimal performance that appears in papers.
- CMA-ES was used to optimize neural network structure in previous works.

Purpose

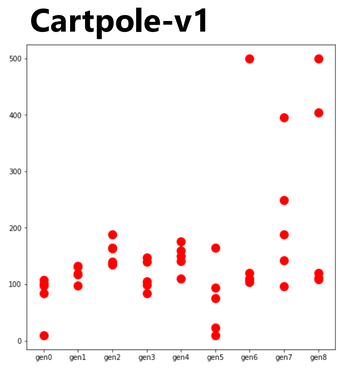
- Find optimal configuration of the algorithm without assistance of human experts.
- Scale to more expensive objective function using parallel computing.

Methods

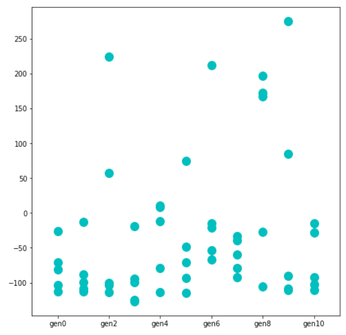
- Trained baseline Deep Q-Network configuration provided by OpenAI.
- Maximizing the reward of the DQN algorithm is the objective function for the CMA-ES.

RESULTS

Cartpole-v1



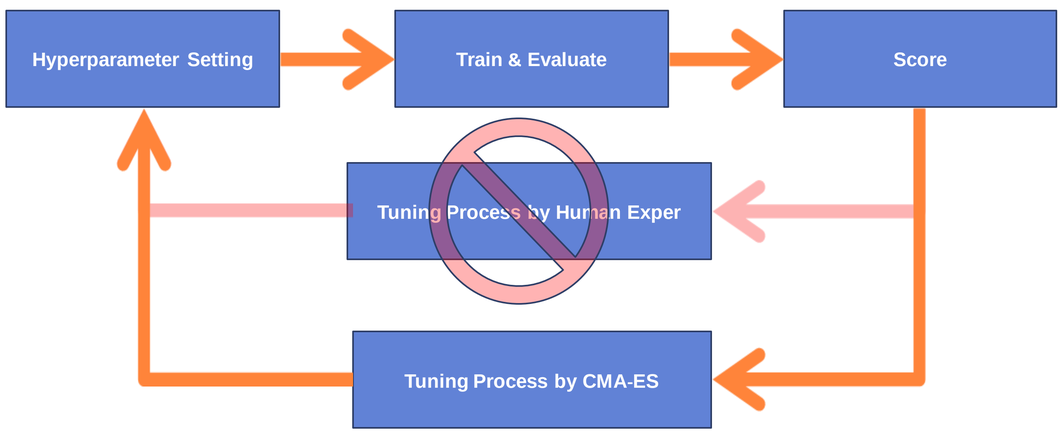
LunarLander-v2



	CartPole-v1	LunarLander-v2
Baseline Reward	110	-102
CMA-ES Reward	500	275

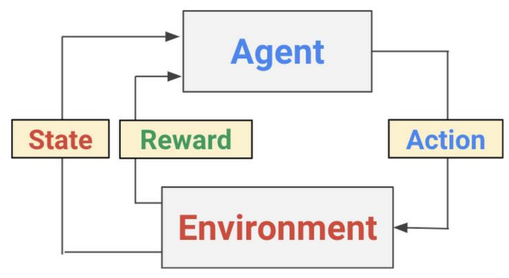
Conclusion

- DQN tuned by CMA-ES outperformed the baselines for both environments.



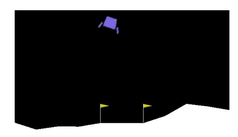
Reinforcement Learning

Form of machine learning where agent takes actions in an environment to maximize reward over sequence of steps.



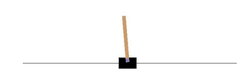
Environments

LunarLander-v2



- If lander moves away from landing pad it loses reward.
- Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points.
- Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points.

CartPole-v1



- The pendulum starts upright, and the goal is to prevent it from falling over.
- A reward of +1 is provided for every timestep that the pole remains upright.
- The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

CMA-ES

1. Calculate the fitness score.
- Implemented as minimization of the objective function.
2. Isolate the best 25% of the population in geeneration.
3. Using the best solutions and mean of the current generation calculate covariance matrix of the next generation.
4. Sample a new set of candidate solutions using the updated mean and covariance matrix.

DQN

- Uses deep neural network to calculate expected q-value.
- Agen takes action that has the highest expected q-value
- Q-Value is overall expected reward assuming agent is in state '*s*' and performs action '*a*'

DQN Hyperparameters

- gamma, learning rate, buffer size, exploration fraction, exploration final epsilon, target network update frequency, prioritized replay alpha, prioritized replay beta0, prioritized replay epsilon, number of nodes in hidden layer (2 layers for cartpole, 3 layers for lunar lander)

Tokyo Tech

Take a picture to see videos of a trained AI
https://youtu.be/7nTh392Lwms