RL: Policy Search

Gradient-free Optimization

Marius Lindauer

Leibniz Universität Hannover

Winter Term 2021

Policy optimization

- Policy based reinforcement learning is an optimization problem over θ

$\rightsquigarrow$ Find policy parameters θ^* that maximize $V(s_0, \theta^*)$

- We can use gradient-free approaches (a.k.a. black-box optimization)
  - Hill climbing
  - Simplex / amoeba / Nelder Mead
  - Genetic algorithms
  - Cross-Entropy method
  - Covariance Matrix Adaptation (CMA)

Policy optimization

- Policy based reinforcement learning is an optimization problem over θ

$\rightsquigarrow$ Find policy parameters θ^* that maximize $V(s_0, \theta^*)$

- We can use gradient-free approaches (a.k.a. black-box optimization)
  - Hill climbing
  - Simplex / amoeba / Nelder Mead
  - Genetic algorithms
  - Cross-Entropy method
  - Covariance Matrix Adaptation (CMA)
- gradient-free optimizers are (often) designed for
  - many function evaluations $\rightarrow$ possible in RL
  - parallel computation $\rightarrow$ possible in RL
  - a few to hundreds of dimensions $\rightarrow$ RL?

Policy optimization

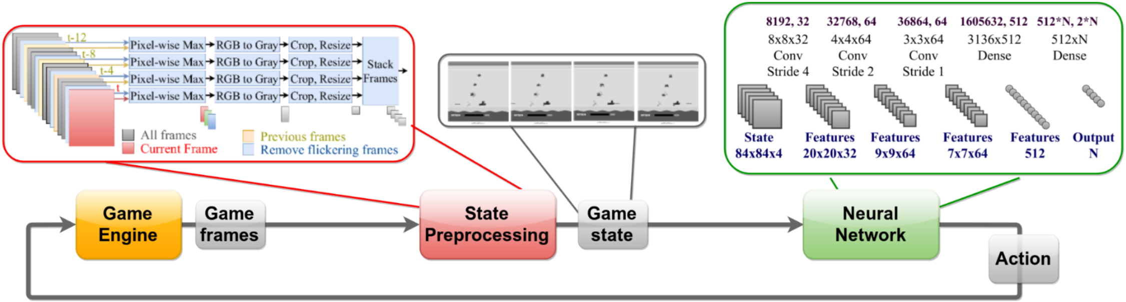
- Policy based reinforcement learning is an optimization problem over θ

$\leadsto$ Find policy parameters θ^* that maximize $V(s_0, \theta^*)$

- We can use gradient-free approaches (a.k.a. black-box optimization)
  - Hill climbing
  - Simplex / amoeba / Nelder Mead
  - Genetic algorithms
  - Cross-Entropy method
  - Covariance Matrix Adaptation (CMA)
- gradient-free optimizers are (often) designed for
  - many function evaluations $\rightarrow$ possible in RL
  - parallel computation $\rightarrow$ possible in RL
  - a few to hundreds of dimensions $\rightarrow$ RL?
- if we encode the policy π_θ as a DNN, we might have millions of dimensions (i.e., parameters in θ)

Policy Optimization with Evolutionary Strategies

- Evolutionary Strategies can perform surprisingly well nevertheless [Salimans et al. 2017], [Chrabaszcz et al. 2018], [Fuks et al. 2019]



Sources: [Chrabaszcz et al. 2018], [Fuks et al. 2019]

Policy Optimization with Evolutionary Strategies

[Chrabaszcz et al. 2018]

Algorithm 1 Canonical Evolution Strategy

Input:
θ_0 - Initial policy vector parameters
T - time budget
λ - Population size
μ - Parent population size
σ - Mutation step-size
$F(\theta)$ - Fitness function for policy evaluation

for $j \in \{1 \dots \mu\}$ **do**
    $w_j = \frac{log(\mu+0.5)-log(j)}{\sum_{k=1}^{\mu} log(\mu+0.5)-log(k)}$

for $t = 0, 1, \dots, T$ **do**
    for $i = 1, 2, \dots, \lambda$ **do**
        $\epsilon_i \sim \mathcal{N}(0, I)$ $s_i \leftarrow F(\theta_t + \sigma \cdot \epsilon_i)$
    Sort $(\epsilon_1, \dots, \epsilon_\lambda)$ according to s in ascending order
    Update policy: $\theta_{t+1} \leftarrow \theta_t + \sigma \cdot \sum_{j=1}^{\mu} w_j \cdot \epsilon_j$

Output: Return best found policy θ_t

Progressive Episode Length [Fuks et al. 2019]