RL: Policy Search

Gradient-free Optimization

Marius Lindauer

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

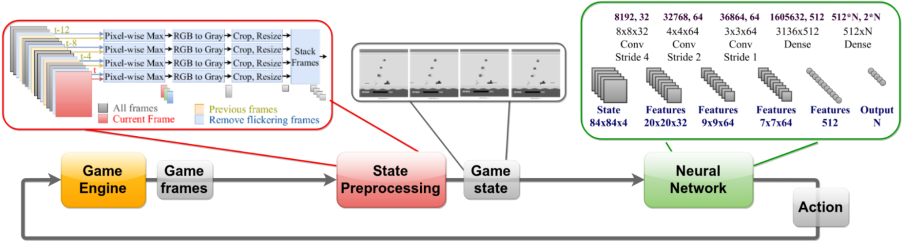
- Policy based reinforcement learning is an optimization problem over θ

$\rightsquigarrow$ 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?
- 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 ∈ {1 … μ} do
    w_j = (log(μ+0.5) − log(j)) / (Σ_{k=1}^{μ} log(μ+0.5) − log(k))
for t = 0, 1, …, T do
    for i = 1, 2, …, λ do
        ε_i ~ N(0, I)
        s_i ← F(θ_t + σ · ε_i)
    Sort (ε_1, …, ε_λ) according to s in ascending order
    Update policy: θ_{t+1} ← θ_t + σ · Σ_{j=1}^{μ} w_j · ε_j
```

$$w_j = \frac{log(\mu+0.5)-log(j)}{\sum_{k=1}^{\mu} log(\mu+0.5)-log(k)}$$

Output: Return best found policy θ_t

Progressive Episode Length [Fuks et al. 2019]