Self-adaptive Robot and Evolution

Yuhao (Henry) Zhou
University of Toronto | Vector Institute
Overview

• Motivation
  • Search for optimal structure
  • Learn a good controller

• Related work
• Algorithm
• Experiments
Motivation:
The Problem of Finding Optimal Robot Structure
Motivation:
The Problem of Adapting Controllers Given a Fixed Structure
Overview

- Motivation
- Related work
  - Search for optimal structure
  - Learn a good controller
- Algorithm
- Experiments
Related Work:
Searching the Best Structure

• Neural Architecture Search
  • Outer-loop: Lots of potential architecture
  • Inner-loop: Train the neural network

• Evolutionary Strategy (ES) or Genetic Algorithms
  • Inner-loop: Random search for controller weights
Related Work: Training the Agent’s Controller

• Reinforcement learning (RL) for mastering locomotion control problems.

• Model-based:
  • Pros: Faster to train
  • Cons: Requires engineering / Slow to simulate

• Model-free:
  • Pros. Fast to simulate
  • Cons: Sample inefficient
Overview

• Motivation
• Related work
• Algorithm
  • Representation of agents’ topology
  • Representation of agents’ policy using NerveNet
  • Amortized fitness
  • Neural topology pruning
• Experiments
Algorithm: Representation of the agent’s topology

- Every species is associated with the topology graph and node attributes
  - $G = (V, E)$
  - $\{a(u) \mid u \in V\}$
Algorithm: Representation of the agent’s policy

• NerveNet: a graph neural net served as the policy
  • For better inheritance of the controller weights in new structure (The weight vector is of the same shape)
Algorithm: Representation of the agent’s policy

- NerveNet++: to speed up training
Algorithm: Performance Metric: Amortized Fitness

• Key idea:
  • Avoid training till convergence to save computation resource on one species.
  • Spread the training across generations.
• Within each generation, each species get same number of updates.
Algorithm: Neural Topology Pruning (NTP)

- Key idea: avoid wasting computation resources on species that have low expected fitness
Algorithm: Neural Topology Pruning (NTP)

• Key idea: avoid wasting computation resources on species that have low expected fitness

• NTP based on Thompson Sampling:
  • Regression-only model to predict reward tend to overfit.
  • Bayesian optimization framework to balance trade-off between exploration and exploitation.
    • Follow “dropout as a Bayesian approximation” and perform dropout during inference.
Algorithm 1 Neural Topological Evolution

1: Initialize $\mathcal{N}$ species with weights and topology $\{\theta_i, G_i\}$
2: while True do
3:     for species $i$ alive do
4:         Train and evaluate Amortized Fitness $\xi_i$ of the species using NerveNet++.
5:     end for
6:     Eliminate $\beta \mathcal{N}$ species with the worst fitness score
7:     Mutate new species with Policy Inheritance
8:     Neural Topology Pruning
9: end while
Overview

- Motivation
- Related work
- Algorithm
- Experiments
  - Environment settings: Fish and Walker
  - Baseline
  - Fine-tuning species
  - Pruning
  - Qualitative result
Experiments: Environment Settings
Experiments: Baseline

(a) Results on fish environment.  (b) Results on walker environment.

Figure 2: The performance of the topology search for Brute-force, MLP and NTE.
Experiments: Fine-tuning Species

(a) Fine-tuning fish3d.
(b) Fine-tuning leg-walker.
(c) Fine-tuning cheetah.
Experiments:
Fine-tuning Species

(a) Fine-tuning fish3d.
(b) Fine-tuning leg-walker.
(c) Fine-tuning cheetah.
Experiments: Pruning Species
Experiments:
Qualitative Result

Gen:5–Spc:103–R:22.41
Future Work

• NTE result:
  • Competitive agents interacting in same environment
  • Cooperative agents interacting in same environment
  • More complex environments
  • Model-based method