Coordinated Multi-Agent Control Utilizing Deep Reinforcement Learning

Alex Gao, Kevin Frans
Henry M. Gunn High School

Introduction

In today's world, nearly everything is automated. But behind this veil of automation, a programmer has to explicitly tell the computer how to act in each situation. As more complex problems are reached, this manual method becomes impractical, as it requires excessive amounts of work.

With autonomous learning, we allow the computer to figure out a strategy on its own. This strategy is referred to as a **policy**, and the computer program following it is an **agent**.

In a standard setup, many simulations are run on an unknown task, starting from a random policy. How the simulation data can be used to improve the policy is an active area of research, especially with the rise of modern computing power.

Objectives

In this experiment, our goal is to identify new learning methods in order to:

- Improve performance on the task. Each task defines success in a different manner, and an optimal strategy may involve cooperation between multiple agents.
- Speed up training time. An algorithm that requires less simulations can learn at a faster rate. Additionally, algorithms that require less compute power can be executed faster in terms of wall-clock time.

Reinforcement Learning

When a human child learns to walk, it isn't told how. Instead, the child keeps trying things until it figures out how each of its muscles should move. Reinforcement learning, a subset of machine learning, follows the same methodology. Through continuous trial and error, a policy can gradually be improved upon.

We represent our policy as a **deep neural network**, mapping from an observation vector (positions and angles of joints) to an agent's actions (joint torque). The policy is stochastic: actions are represented as normal distributions, parametrized by a mean and a standard deviation. This network can be thought of as an agent's "brain".

Each task gives out a reward value based on how well the agent is performing. This reward is different in each task, ranging from 'how far forward the agent has traveled' to 'how close is the agent to a desired point'.

By repeatedly running simulations of a task, we can record which actions, on average, resulted in a higher reward.

Utilizing this information, we backpropagate through the neural network, increasing the likelihood of actions that have proven to be beneficial.

This paradigm is known as the **policy gradient**, and is the basis for how our learning agent is trained.

Hypothesis

Traditional methods use one big policy to account for all the agents actions. However, this setup has trouble scaling into more complex tasks. **Instead, we split this large network into many smaller, distinct networks — one for each agent.** We view each of a robot's joints as separate agents, allowing independent decisions to be made. In practice, this means training distinct neural networks for each joint. Our method is **decentralized**, and each network can be trained and executed independently of the others.

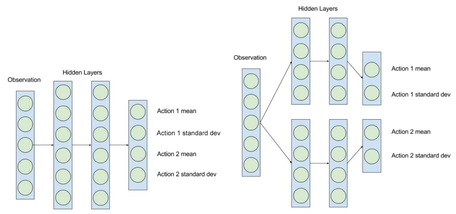


Figure: Left: traditional network. Right: decentralized network.

Tasks

Figure: Swimmer

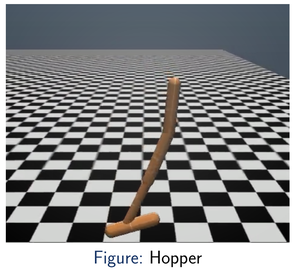


Figure: Hopper

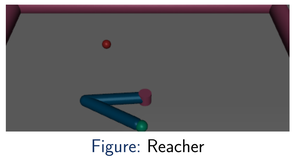


Figure: Reacher

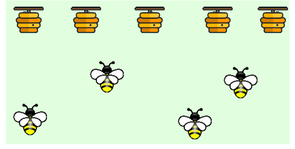


Figure: Delivery

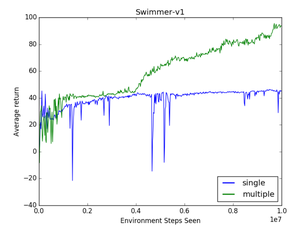
Rationale

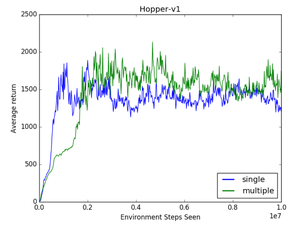
Large neural networks, especially those with multiple objectives, often struggle to encode observations correctly. For example, a robot will move its leg joint differently, depending on the position of its foot. However, this information is not useful in controlling its right arm. When using a single network, the network is forced to encode all information, which can lead to confusion.

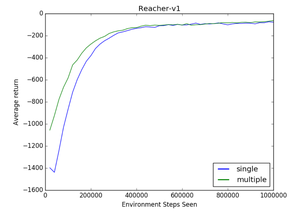
By utilizing multiple networks, we allow **each network to specialize**, and only encode the information useful to its specific role in the task.

Additionally, the decentralization of our policy enables scaling up of training, through **parallelization**. In a centralized setup, the entire network would have to be updated before the next update could start. With the policy split into separate networks, each can be trained on different computers, in parallel – allowing for faster training when scaling up.

Results







Algorithm

Algorithm 1 Decentralized Policy Gradient

initialize π for every agent
for iteration = 0,1,2,... until convergence **do**
  for episode = 0,1,2, ... 10000 **do**
    Reset environment
    for timestep = 0,1,2,... until episode end **do**
      Receive observation
      Every agent takes action according to policy π
      Record tuple (observation, action, reward)
    Estimate value function Q(s,a) using recorded tuples
    Compute gradient to increase expected value
    Update policies in gradient direction

Conclusion

We tested our algorithm's performance on a variety of standard control tasks, and showed an **increase of up to two times the performance of the single-network method**. Our method also learned an optimal solution in less simulations. Repeated trials on each task validated our results.

While the total computational power of a decentralized method may be higher, the neural networks can be trained in parallel across multiple computers, effectively reducing the computational time. This is more beneficial for practical use, since the smaller neural networks can be distributed among each agent without intercommunication.

Our method's improvements on the industry standard, in terms of performance and training time, allow for deep reinforcement learning to be applied in many practical scenarios. Real-world problems such as assembly line robotics have traditionally been explicitly coded; however, as autonomous learning becomes more efficient, artificial intelligence may take over these jobs.

Future Work

Our work provides a reliable basis for practical cooperative learning in a multitude of environments, and paves the way for future research in the emergent field of multi-agent control.

Our robust algorithm has countless practical applications requiring cooperative learning that can be explored in future work, including:

- Drone package delivery network management and collision avoidance
- Robotic bee pollination to proxy bee extinction
- Planetary exploration with teams of rovers
- Prosthetics for disabled persons