Exploring Distributed Reinforcement Learning

Sean Parker

Introduction

TensorFlow

PyTorch

RAY

rllib

Motivation

- Many options for distributed training
  - Personally, I’ve only used Tensorflow before
  - Some may be easier or more difficult to use
- Ray, Horovod, PT/TF Distributed
- PT seems to be popular for quick prototyping, TF has more complex API

What’s the difference?

- Tensorflow
    - Offers multiple distribution “Strategies”
        - Mirrored, TPU, MultiWorker, Param Server, Central Storage and more
- PyTorch
    - Split into three components
        - Distributed data-parallel training, RPC-based distributed training, Collective communication
    - Support TPUs?
- Ray
    - Distributed execution engine - handles scheduling, management, fault tolerance
    - Can support TF + PT models, integrations with other libraries/frameworks

Comparison

Tensorflow:

- Define a ‘Strategy’
- Split dataset using strategy
- Build model using strategy scope
- Train

PyTorch:

- Define a ‘Process group’
- Build model
- Wrap in abstraction for Distributed training
- Train

Ray:

- Load data
- Build parallelizable model
- Train

Goal

- Comparison between Ray, PT, TF for distributed RL
- Ray should be easier to use, but maybe there is more overhead to learning how to setup & use Ray
- Explore newly introduced RaySDG
    - Promises simpler scalability, unified monitoring

Plan

- Research simple RL environment & possible agents to implement
- Explore how each system distributes computation over machines
- Build simple RL parallelizable agent
- Evaluation:
    - Quantitatively - running time + accuracy
    - Qualitatively - ease of use + intuitiveness

Thank you!

Questions?