Contents

1 Distributed Multi Agent Learning for Portfolio Management

CMPT 756: Systems for Big Data Project Proposal

1.1 Subject

Our subject is on virtualized trading agents that can be easily deployed to cloud environments. Agents refer to deep reinforcement learning based cryptocurrency trading bots as outlined in this paper, and implemented in this repo. By deploying trading agents to virtualized cloud environments we are able to easily scale up the number of active bots. By having many bots deployed we are able to do interesting things like parallel grid search over hyper-parameters, inter-agent communication, and genetic programming based bot evolution.

1.2 Team

Shawn Anderson Sethuraman Annamalai Namita Shah

1.3 Work Plan

We have scheduled weekly meetings every Tuesday 2-5 pm. Meetings will serve the following purposes:

- Share progress of past week, get all team mates on same page
- Brainstorm next steps and project ideas
- Collaborative work session for project tasks

In addition to meetings we will all work as individuals throughout the week to stay up to date, and accomplish assigned tasks for the project.

1.4 Project Proposal

1.4.1 Introduction:

In Deep Reinforcement for the Financial Portfolio Management Problem, the authors build a trading bot that re-optimizes a cryptocurrency portfolio at every time step. Our goal is to encapsulate the software that the authors release along with the paper, thus being able to spin up instances of the code(agents) in a cloud environment. Once the initial phase of cloud deployment is accomplished, there are various interesting scientific routes that could be taken from there.

1.4.2 Task 0: Deploy Remote Agent

Create a cloud environment template which will be ready to run the PGPortfolio software. Be able to instantiate an agent environment with an API call. Have Agent request initialization parameters from server. Have agent download its appropriate training and testing data. Perform hardware queries for dynamic optimization (Attempt GPU?). Begin training and testing.

1.4.3 Task 1: Centralized Database

Maintain a database which exposes a web API as a RESTful interface. This database will keep track of all agents that have been deployed, their parameters, and performance metrics. A newly deployed agent will make a request to the API for it's initialization parameters. From maintaining this data, we are able to retrieve interesting statistics like which agents performed the best over which periods, how many agents have been active at what times, which agent made the most Bitcoin, ect.

1.4.4 Task 2: Plotting Performance

Once we have remote agents posting performance metrics to the data base, we would like to have some sort of visualization of these metrics. Visualization will serve as a sanity check to ensure that things are working properly, as well as it will serve as a powerful analysis and debugging tool as we move on to more advanced tasks.

1.4.5 Task 3: Hyper-parameter optimization

When agents spawn, they will request initialization parameters from the server. Having a centralized initialization state dispatcher allows us to search the hyper-parameter space intelligently. A naive example would be to perform an exhaustive grid search over the hyper-parameters. But we can do better.

1.4.6 Task 4: Hyper-parameter evolution

Distributed Evolutionary Algorithms in Python (DEAP) is a framework for state-space optimization using evolutionary approaches like populations, generations, fitness, mutation and prefix-trees. This could be used as the back-end for our parameter-initialization dispatcher.

1.4.7 Task 5: Additional learning features

In the paper, the agents are able to perform well with only historic price movement as input. Specifically, only three features: High, Low, Closing, for each time period (30 seconds). Additional features could be tested, for example, trade volume.

1.4.8 Task 6: Transfer Learning

Transfer learning is a method in deep learning in which internal representations learned in one setting are transferred to another setting. If we could have each agent post it's learned parameters to a URL encoded as a prefix-tree, or genetic representation that describes that agent. Then perhaps we could have agents literally inherit the genes of their ancestors.

1.4.9 Task 7: Inter-agent communication

Imagine two agents sharing information, for example, one agent could tell it's prediction's to another agent by concatenating it's output to another agent's input. Or consider inter-agent trading, which could circumvent market transaction fees.

1.4.10 What technologies are you using? What do you need?

1. SFU-Cloud (VM Template, DB host)
2. Python: Tensorflow, Django REST Framework, DEAP, SCOOP
3. Database Backend
4. Poloniex API for Data
5. Optional: GPU

1.4.11 Questions / Concerns

1. Should agents stream data directly from source(poloniex) or from a central database?
   - We don't want to hammer the poloniex API, but it would be way easier if agents get their own data
2. Why must this approach be distributed? Why is it not the same to simply run agents as separate processes on a single machine?
   - Perhaps we should spin up multiple agents as separate processes within cloud environments?

1.4.12 Task Division

We will all work together to accomplish task 0. This will assure that all group members become familiar with the software and deployment technique. Other than that, we will mostly work together on tasks, but to introduce some paralellism, we will say Shawn is responsible for tasks 1,2; Sethu is responsible for 3,4; Namita is responsible for 5,6. Task 7 will remain as a bonus.

1.4.13 Timeline

1. Optimal Task 0 is finished before March 2nd. Each member has acheived results on one of their two tasks by March 9th. Most members have acheived results on their second task by March 23rd. Remaining time is used to run experiments, investigate task 7 or additional tasks that arrise, and write the report.

2. Likely Task 0 is finished before March 2nd. Each member has acheived results on one of their two tasks by March 16th. Each member has acheived results on their second task by April 1st. Remaining time is used to run experiments, investigate task 7, and write the report.

3. Backup Task 0 is finished before March 9th. Some members have acheived results on one of their two tasks by March 16th. Members collaborate to assure the 5/8 tasks are acheived by April 1st. Remaining time is used to run experiments, investigate remaining tasks, and write the report.

1.5 References

1. Deep Reinforcement Learning for Portfolio Optimization

2. DEAP: Distributed Evolutionary Algorithms in Python

3. SCOOP: Scalable COncurrent Operations in Python

4. Django Rest Framework