deependersingla / deep_trader

This project uses reinforcement learning on stock market and agent tries to learn trading. The goal is to check if the agent can learn to read tape. The project is dedicated to hero in life great Jesse Livermore.

148 commits | 2 branches | 0 releases | 3 contributors

Branch: master | New pull request | Find File | Clone or download

deependersingla Update README.md — Latest commit 82af6ba on Jan 22, 2018

data	nifty data added	3 years ago
ib	Data gathering process done	3 years ago
q_learning	wip for supervised learning	3 years ago
supervised	wip for supervised learning	3 years ago
tensor-reinforcement	TODO added	3 years ago
README.md	Update README.md	a year ago
__init__.py	init file added	3 years ago
deep_thoughts.md	Fix broken Markdown headings	2 years ago
prepare_data.py	nifty data added	3 years ago
questions	updated questions	3 years ago
requirements.txt	readme and requirement.txt	3 years ago
run.py	wip for supervised learning	3 years ago

README.md

Reinforcement-trading

This project uses Reinforcement learning on stock market and agent tries to learn trading. The goal is to check if the agent can learn to read tape. The project is dedicated to hero in life great Jesse Livermore and one of the best human i know Ryan Booth https://github.com/ryanabooth.

One Point to note, the code inside tensor-reinforcement is the latest code and you should be reading/running if you are interested in project. Leave other directories, I am not working on them for now
. To read my thought journal during ongoing development
https://github.com/deependersingla/deep_trader/blob/master/deep_thoughts.md

Before this I have used RL here: http://somedeepthoughtsblog.tumblr.com/post/134793589864/maths-versus-computation

Now I run a company on RL trading, so I can't answer questions related to the project.

Steps to reproduce DQN

a) cd tensor-reinforcement
b) Copy data from https://drive.google.com/file/d/0B6ZrYxEMNGR-MEd5Ti0tTEJjMTQ/view and https://drive.google.com/file/d/0B6ZrYxEMNGR-Q0YwWWVpVnJ3YmM/view?usp=sharing into tensor-reinforcement directory.
b) Create a directory saved_networks inside tensor_reinforcement for saving networks.
c) python dqn_model.py

Steps to reproduce PG

a) cd tensor-reinforcement
b) Create a directory saved_networks inside tensor_reinforcement for saving networks.
c) python pg_model.py

For the first iteration of the project

Process:
Intially I started by using Chainer for the project for both supervised and reinforcement learning. In middle of it AlphaGo (https://research.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html) came because of it I shifted to read Sutton book on RL (https://webdocs.cs.ualberta.ca/~sutton/book/the-book.html), AlphaGo and related papers, David Silver lectures (http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html, they are great).

I am coming back to project after some time a lot has changed. All the cool kids even DeepMind (the gods) have started using TensorFlow. Hence, I am ditching Chainer and will use Tensorflow from now. Exciting times ahead.

Policy network

I will be starting with simple feed-forward network. Though, I am also inclined to use convolutional network reason, they do very well when the minor change in input should not change ouput. For example: In image recognizition, a small pixel values change doesn't meam image is changed. Intutively stocks numbers look same to me, a small change should not trigger a trade but again the problem here comes with normalization. With normalization the big change in number will be reduced to a very small in inputs hence its good to start with feed-forward.

Feed-forward

I want to start with 2 layer first, yes that just vanilla but lets see how it works than will shift to more deeper network. On output side I will be using a sigmoid non-linear function to get value out of 0 and 1. In hidden layer all neurons will be RELU. With 2 layers, I am assuming that first layer w1 can decide whether market is bullish, bearish and stable. 2nd layer can then decide what action to take based on based layer.

Training

I will run x episode of training and each will have y time interval on it. Policy network will have to make x*y times decision of whether to hold, buy or short. After this based on our reward I will label every decison whether it was good/bad and update network. I will again run x episode on the improved network and will keep doing it. Like MCTS where things average out to optimality our policy also will start making more positive decision and less negative decision even though in training we will see policy making some wrong choices but on average it will work out because we will do same thing million times.

Episodic

I plan to start with episodic training rather than continous training. The major reason for this is that I will not have to calculate reward after every action which agent will make which is complex to do in trading, I can just make terminal reward based on portfolio value after an entire episode (final value of portfolio - transaction cost occur inside the episode - initial value of portfolio). The other reason for doing it that I believe it will motivate agent to learn trading on episodes, which decreases risk of any outlier events or sentiment change in market.

This also means that I have to check the hypothesis on:
a) Episodes of different length
b) On different rewards terminal reward or rewards after each step inside an episode also.
As usual like every AI projects, there will be a lot of hit and trial. I should better write good code and store all results properly so that I can compare them to see what works and what don't. Ofcourse the idea is to make sure agent remain profitable while trading.

More info here https://docs.google.com/document/d/12TmodyT4vZBViEbWXkUIgRW_qmL1rTW00GxSMqYGNHU/edit

Data sources

1. For directly running this repo, use this data source and you are all setup: https://drive.google.com/open?id=0B6ZrYxEMNGR-MEd5Ti0tTEJjMTQ
2. Nifty Data: https://drive.google.com/folderview?id=0B8e3dtbFwQWUZ1I5dklCMmE5M2M&ddrp=1%20%E2%81%A0%E2%81%A0%E2%81%A0%E2%81%A09:05%20PM%E2%81%A0%E2%81%A0%E2%81%A0%E2%81%A0%E2%81%A0
3. Nifty futures:http://www.4shared.com/folder/Fv9Jm0bS/NSE_Futures
4. Google finance
5. Interative Brokers, I used IB because I have an account with them.

For reading on getting data using IB
https://www.interactivebrokers.com/en/software/api/apiguide/tables/historical_data_limitations.htm
https://www.interactivebrokers.com/en/software/api/apiguide/java/historicaldata.htm symbol: stock -> STK, Indices -> IND

Reinforcement learning resources

https://github.com/aikorea/awesome-rl , this is enough if you are serious