ClusterTech

Abstract: In meteorological model with low resolution like 27-km, wind is often over-predicted and cold fronts are often predicted to come earlier than they actually occur in some seasons. Part of the reasons may lie in the pre-processing of static data or parameter sets, e.g., mountains are not fully represented in interpolated terrain data, static data or parameter sets that affect surface friction are not optimized for the region, etc. Without resorting to higher resolution, is it worthwhile to explore tricks in manipulating static data and parameters, say, optimizing against dominating wind directions for specific seasons of a region of interest, etc, so as to remedy errors in several common scenarios for China?

Data: These summer projects should involve public data only. We expect that WRF's Geographical Input Data Downloads, NASA's terrain data as an alternative to explore, and NCEP GFS / FNL data (for initial and boundary conditions) are used.

Idea: One idea is to vary the pre-processing (e.g. in the geogrid step) of static data. For example, Northwestern wind is often predicted to be too strong and cold front predicted to come too early. We may try some methods that make mountains to be higher / steeper, or surface friction to be larger in the model. Then, carry out iterative numerical experiments to see whether WRF would predict cold front more accurately in China. The project is an open R&D problem, and we are open to evaluate any idea suggested, by students.

Skills required: Scripting languages, like Python, NCL, etc. are expected if that can do the job. Otherwise, the most appropriate programming language to do the job, like Fortran if geogrid itself needs to be modified, is expected.

Charles River Advisors (Hong Kong) Ltd: A T+0 Stock Trading Strategy Based on Machine Learning Techniques Industry

Mentor: Hongsong Chou, Ph.D.
Charles River Advisors (Hong Kong) Ltd.

Abstract

In the financial markets, we call exchange-based transaction activities as those of the "secondary market", as compared to the "primary market" which often refers to initial public offerings (IPOs). For secondary market trading, the so-called "liquid products" - such as common stocks, index and commodity futures, exchange-traded options, etc. - can be traded multiple times throughout the day. That is, buy, sell, then buy, sell again, for a number of times throughout one trading day. Such financial trading activity is often called "T+0" trading. To profit from such trading activities, one often uses trading data to construct certain types of "signals" so as to forecast the future trajectories of the liquid products that are being traded. The trading decision (buy or sell) made at each step is based on many factors, among which such "signals" play key roles.

Signals are constructed using so-called "trading data", which often include (but are not limited to) observable market-related variables such as trade price, trade size, buy or sell directions, etc. One of the key concepts is "order book" or "limit order book". The order book is a snapshot of the market supply and demand at any moment, and is published by the exchange whenever it sees necessary. Because the order book is a snapshot about the market supply and demand, it often includes information such as, at that moment, how many orders are buy-orders and at what price and quantity (ditto for sell). At any moment, the "lowest" price that a market participant offers to sell is called an "Ask" or "Offer", and the "highest" price that a market participant offers to buy is called a "Bid". Both bid and ask are also called "quotes", and the simple average of the bid and the ask is called the "mid-quote". If the "Ask" is higher than the "Bid", there can be no transaction to happen. However, when the "Ask" is equal to or lower than the "Bid", one or a few transactions can happen, and the exchange will release such "trade" information (such as at what time, at what price and how many quantities of a stock have just been traded). All of the information such as "order book", trade, price, quantity, etc. Form the so-called "market data" that can be of rather high frequency nature, such as as a matter of a few milli-seconds.

The project:

1. Introduction: In this project, we will analyze high-frequency stock data of three types: snapshot data, individual trade data, and order queue data. With such data, we will first construct a few "factors" (or, "features" in machine learning jargon) and analyze the accuracy of forecasting future stock price returns when we combine these features. From practical point of view, assume that we are at time t during a live trading session. We will try to forecast the return between Pavg(t+dt) and P(t). Here, P(t) is the mid-quote at or immediately before time t, and Pavg(t+dt) is the arithmetic average of all the mid-quote values from t to t+dt (both ends inclusive). The value of dt is usually about 30 seconds, 60 seconds, 3 minutes, 5 minutes, and 10 minutes. For most of the time, we use 10 ticks, 20 ticks, 50 ticks, 100 ticks and 200 ticks

instead of seconds or minutes. We do the forecasting using necessary information between t-Dt and t, with Dt to be a few ticks or minutes.

2. Feature construction:

- a) First of all, we will construct a "medium frequency" feature called "support and resistance level (SRL)". Basically, we will use the past 15 minute mid-quote of each individual snapshot data. For all the mid-quote values within this 15-minute window, we calculate the top 95% and bottom 5% cut-off values and use them as two threshold values called S-value (for support) and R-value (for resistance).
- b) Next, we will construct a group of features called "rounded trade size" or (RTS). There are three features in this group: RTS1 = 1 if the trade that happened immediately before t is of a size that is a multiple of 1000 (such as 1000, 2000, 3000, etc.) but not of 5000 and 10000; RTS1 = 1 if the trade that happened immediately before t is of a size that is a multiple of 5000 (but not of 10000); and RTS3 = 1 if the trade that happened immediately before t is of a size that is a multiple of 10000. RTS1/2/3 = 0 otherwise.
- c) Thirdly, we will construct a feature called "trade arrival intensity" or TAI. Define the number of individual trades that happened between the most recent two snapshots immediately before t as Nt, and the average/stdev of Nt between t-Dt and t is Navg/Nstd. Then, this feature is calculated as (Nt-Navg)/Nstd. We hope that this feature can capture the sudden increase or decrease in trading activity for the current time point, t. Dt is usually about 15 minutes or even longer.
- d) Fourthly, we will construct a group of features called "Order Queue Distribution" or OQD. Assume that the number of limit orders on bid1 (or ask1) at time t is Nb (or Na), and the total number of shares on bid1 (or ask1) at time t is Vb (or Va). Define the average queued order size at time t as OQD1b = Vb/Nb on bid1 and OQD1a = Va/Na on ask1. In a similar way as we defined TAI above, we calculate (OQD1b_t – OQD1b_avg)/OQD1b_stdev and (OQD1a_t – OQD1a_avg)/OQD1a_stdev as two features in the group of "Order Queue Distribution".
- e) (We may add a few more features as the project moves along).

3. Forecasting:With all the above defined features at time t, we will use different methods to construct a forecasting model for future price returns as discussed before. In general, we will use two approaches:

- a) Supervised Learning Approach: First, we will use a few basic machine learning techniques such as Decision Tree, Random Forrest, Logistic Regression, and so on. In this approach, we will essentially using mature supervised learning and/or regression schemes to construct the forecasting models and check their in-sample and out-of-sample forecasting accuracy.
- b) Deep Learning Approach: In this approach, we will try to use three methods: first of all, we will try to use basic ConvNet architecture to conduct forecasting; secondly, we will use the methods discussed in the two references: "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading " by Deng et al and " A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem" by Jiang et al. We will decide which one method in the two reference papers will be used as the project moves along.

Requirements for the team (It is not necessary that each individual team member meets all the requirements.)

- Basic knowledge of machine-learning models: supervised learning such as decision trees, SVM, etc.;
- deep learning models such as basic CNN, sequence models, etc.; experiences with standard machine
- learning packages such as TensorFlow and Keras;
- Basic knowledge of financial market trading facts, such as limit order books, trade and quote, etc.;

What we can provide

- Basic tutorials on trading strategy design and market data and how to process them;
- Some available machine learning infrastructure / model / data for different machine learning tasks.

References:

1. For order book related basics, the first three chapters of "Empirical Market Microstructure: The Institutions, Economics, and Econometrics of Securities Trading" by Joel Hasbrouck can be rather helpful;

2. Deep Direct Reinforcement Learning for Financial Signal Representation and Trading, by Yue Deng , Feng Bao, Youyong Kong, Zhiquan Ren and Qionghai Dai; IEEE Transactions on Neural Networks and Learning Systems (Volume: 28, Issue: 3, March 2017);

3. A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem, by Zhengyao Jiang, Dixing Xu, Jinjun Liang; https://arxiv.org/abs/1706.10059.

Magnum Research Ltd

Portfolio Management using Reinforcement Learning

Industry Mentors: Dr. Don HUANG and Dr. Joseph CHEN
Magnum Research Ltd, HK

Abstract

Portfolio management is the decision making process of continuously reallocating an amount of fund into a number of different financial assets, aiming to maximize the return while restraining the risk. Over the past two decades, the development of automated trading strategies via reinforcement learning (RL) has been an active area of research. Reinforcement learning algorithms typically do not assume a complete model of the underlying system and is a potentially powerful approach for solving challenging financial problems such as portfolio management. There are two main categories of RL algorithms: (1) value function (VF) learning, a typical example of which is Q-learning algorithm; and (2) direct reinforcement (DR), a typical example of which is recurrent reinforcement learning (RRL) algorithm. While some researchers have successfully implemented Q-learning algorithm to a two-stock portfolio, and showed it promising to outperform the preset benchmarks [1], some authors claimed that Q-learning can suffer from the curse of dimensionality and is more difficult to use than RRL approach [2]. Therefore, we would like to proposed this topic to investigate the effectiveness of Q-learning algorithm and RRL algorithm in portfolio management, and their corresponding limitations.

The project

In this project, students will run practical experiments on reinforcement learning algorithms to explore their applications on portfolio management. The goal is to get the conclusion on the effectiveness of applying reinforcement learning techniques to portfolio management, and whether Q-learning will outperform RRL algorithms, or vice versa. The output could be in multiple forms:

1. Performance of portfolio using reinforcement learning, giving specific time period, underlying assets, and appropriate benchmarks.
2. Comparison of performances of portfolio using Q- learning and recurrent reinforcement learning.

The algorithms will be tested on stock market data or cryptocurrency data. The data and backtesting engine can be provided by zipline (https://github.com/quantopian/zipline), an open source pythonic algorithmic trading library. We will also use open source neural network library such as Keras (https://github.com/keras-team/keras) to implement the algorithms.

Requirements for the team (It is not necessary that each individual team member meets all the requirements.)

- Proficient in Python programming and commonly used libraries such as Numpy and Pandas.

- Basic knowledge of machine-learning models: Q-learning model, recurrent reinforcement learning model.
- Basic knowledge of financial portfolio management.
- Basic knowledge of partial differentiation and linear algebra.

What we can provide

- Guidance on reinforcement learning theory.
- Some available machine learning infrastructure / model / data for different reinforcement learning tasks.
- Possibly some data resource for running cryptocurrency jobs.

Magnum Research Limited

Magnum Research targets the pain points of wealth management process by bringing automation and customization. With services covering all three aspects of finance, algorithm and information technology, Magnum aims to assist financial institutions and their clients with outstanding asset allocation service, which eventually leads to transparent, stable, instant and affordable wealth management experiences for users worldwide. Magnum finishes its Series A Financing Led by Alibaba Entrepreneurs Fund in June 2017. Magnum holds SFC Type 1 and Type 4 licenses.
https://aqumon.com/

References

[1] Portfolio Management using Reinforcement Learning
http://cs229.stanford.edu/proj2016/report/JinElSaawy-PortfolioManagementusingReinforcementLearning-report.pdf
[2] Reinforcement Learning for Trading
http://papers.nips.cc/paper/1551-reinforcement-learning-for-trading.pdf
[3] Recurrent Reinforcement Learning with Expected Maximum Drawdown
https://www.researchgate.net/profile/Steve_Yang2/publication/317622713_An_adaptive_portfolio_trading_system_A_risk-return_portfolio_optimization_using_recurrent_reinforcement_learning_with_expected_maximum_drawdown/links/59f3401da6fdcc075ec339c5/An-adaptive-portfolio-trading-system-A-risk-return-portfolio-optimization-using-recurrent-reinforcement-learning-with-expected-maximum-drawdown.pdf
[4] Trading using Recurrent Reinforcement Learning and LSTM Neural Networks
https://arxiv.org/pdf/1707.07338.pdf
[5] Deep Reinforcement Learning Framework for Portfolio Management
https://arxiv.org/pdf/1706.10059.pdf%EF%BC%89
[6] Keras
https://keras.io/
[7] Zipline
http://www.zipline.io/

Tencent (SPIA): Sketch-to-image Generation

In this project, you are going to implements a neural network to convert a sketch image to a real photo at high resolution. The Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of images from two different domains. In this project, you are going to extend the cycleGAN by implementing a multi-scale cycleGAN which combine global and local discriminator to convert a sketch image to a high quality high resolution photo realistic image which advances the results of the cycleGAN.

Useful materials
https://github.com/junyanz/CycleGAN