Machine Learning Engineer Nanodegree

Capstone Proposal

Sebastian Janisch

November 2016

Proposal

Domain Background

In quantitative asset management any available information that can be quantified is used to uncover potential market opportunities. These opportunities are typically caused by mispricing, where a stock's price is not reflective of its true (intrinsic) value. The ***Efficient Market Hypothesis*** (henceforth referred to as EMH) suggests that a "market in which prices always 'fully' reflect available information is called 'efficient'" (Fama, Efficient Capital Markets) indicating that in such efficient markets opportunities due to mispricing don't exist. Empirical analysis suggests however that this theory can, at least in part, be refuted (Sewell, The Efficient Market Hypothesis: Empirical Evidence), opening the playing field for a broad array of quantitative approaches to find said market inefficiencies.

Problem Statement

The assumption of all active asset management is that markets are not always efficient and can be outperformed (i.e. risk adjusted outperformance for given benchmark). This project aims to use model free machine learning techniques to devise an agent based system that generates trading decisions to profit in the stock market. Much of traditonal quantitative asset management uses a model based approach, where investment ideas are casted into a model which generates trading signals (Grinold/Kahn, Active Portfolio Management). This implies that a concrete statement is made as to whether or not a given piece of quantitative information is positive or negative (e.g. high corporate leverage indicates poor financial health and possibility of financial distress). The interpreation of a given quantitive metric is however not always so clear (e.g. some industries are characterised by higher leverage without posing a risk of financial distress). The system proposed in this project sets forward an alternative ***model free*** approach using Q-learning, where no statement is made as to the interpretation of a quantitative metric but it is rather left to the system to learn it autonomously.

Datasets and Inputs

Fortunately in the finance domain quantiative data is readily available and abundant. It is

understood that more complete, timely and extensive data can be obtained from vendors at cost, however, for the purpose of this project data readily available and free data sources shall suffice.

One of the most crucial information about a traded stock is surely its stock price. As pointed out by Sewell (see above) the *EMH* can be strongly refuted for daily stock prices, evidence for longer frequencies is thinner though. It is hence that for this project daily asset returns will be used as a primary data input.

Based on the daily (adjusted, i.e. corrected for stock splits and dividend payments) stock price alone alongside with daily trading volumes a whole field has emerged engaging in the discipline called *Technical Analysis* (Murphy, Technical Analysis of the Financial Markets). Using metrics that are essentially a function of stock price together with trading volume a whole battery of different metrics is created that is thought to help find market inefficiencies (e.g. Bollinger Bands). There is much controversy as to whether or not plain technical analysis is truly a strong predictor of future stock prices. Enriching the data set with information other than the stock price alone can add more insight into the details of a company behind a stock and as such yield a more predictive set of information. Fundamental quantitative data derived from financial statements (e.g. P/E ratio, profit margin, etc.) is hence used as a second set of input data into the learning algorithm.

The data source used will be the Quandl API to retrieve stock prices and financial statement data for given stocks.

Solution Statement

Given the above data source appropriate quantiative metrics will be selected to form the state space for the Q-Learning algorithm.

As a universe of stocks there will be a sample of US equities chosen across different industries.

Intuitively, financial statement ratios for different classes can be chosen. Popular classes are performance ratios (e.g. profit margin), activity ratios (e.g. asset turnover), financing ratios (e.g. leverage) and liquidity (e.g. interest coverage). Given that these metrics are all percent based they are easily discretized to form a state space.

On the side of pure stock price related metrics signals such as momentum, market beta, bollinger band values etc. will be chosen and discretized.

The generated state space will then be explored by randomly iterating over historical examples of stocks for which the above metrics are computed (approach of the Dyna-Q learning algorithm). The reward function will be based on the return generated for a given decision the agent took.

Evaluation Metrics

To evaluate the performance the actions carried out by the agent will be benchmarked against the return of a representative stock index. Given that the learner for the purpose of this project will be based on US stocks the S&P 500 index will be used as a benchmark. To ensure the return is compared apples to apples a risk adjusted measure will be used to quantify the

risk adjusted return of both benchmark index and our stock portfolio (see Sharpe Ratio).

Project Design

For the implementation of this project the below Python libraries will be used:

- Numpy (for numerical operations)
- Pandas (for tabulating and manipulating data)
- Quandl (to access the Quandl financial database
- Seaborn and Matplotlib (for charting)

An abstract Q-learning implementation will be implemented agnostic of the financial domain we operate in (i.e. implement state space, learning function, exploration of states given input data, etc.). This learning implementation will then be used to initially train the agent on historical data without executing actions or simulating any portfolio. Following this learning phase the agent will be executed against historical data, progressing through time. Based on input data the agent will at a given point in time take one of the below actions for a given stock:

- BUY
- SELL
- DO NOTHING (i.e. hold or don't buy)

Based on these actions a portfolio will be formed and its performance recorded through time. For the time frame of the execution performance measures will be formulated (see sharpe ratio above) and visualised. The important outcomes to determine are:

- the performance of the portfolio against the benchmark
- the performance through time (does it improve as the agent learns)

For simplicity the agent itself will be run as a basic command line python program which will generate csv based outputs (actions take at given time, etc). To then measure performance and visualise results an iPython Notebook seems to be the most suitable tool. It allows to read the produced csv outputs using Python, parse them and produce performance outputs.