Utilizing fundamental factors in reinforcement learning for active portfolio management

Oleg Mitsik
Stanford University
OMitsik@stanford.edu

***Abstract* — This paper examines how utilizing fundamental factors in reinforcement learning for active portfolio management can help to achieve better portfolio performance. Suggested deep recurrent Q-learning model is illustrated on a concentrated portfolio of 3 value stocks in the public utilities sector coupled with a risk-free asset. It is shown that the model outperformed benchmark portfolio on the test dataset, as measured by Sharpe ratio. Such an algorithmically managed portfolio could potentially complement the universe of exchange traded funds (ETFs), which are typically constructed to track some financial index.**

Keywords* — *Quantitative Finance, Algorithmic Trading, Quantamental Analysis, Reinforcement Learning, Q-Learning, DRQN, Neural Network, LSTM.

GitHub* — *https://github.com/OlegMitsik/AA228_Project

I. Introduction

Algorithmic trading has become increasingly popular these days, as evidenced by the fact that computer-driven investment companies are taking in investor money from the other strategies [1].

Traditionally, quantitative investment strategies were developed to perform technical analysis, such as finding arbitrage opportunities or sustainable patterns in prices of traded financial assets. Oppositely, human analysts usually performed fundamental analysis, i.e. evaluated fair price of financial assets based on their understanding of underlying business activities and future economic conditions.

Quantamental investment strategy is an emerging hybrid approach that combines both fundamental analysis and advanced quantitative methods, e.g. machine learning. This paper attempts to implement a reinforcement learning approach for active portfolio management which would take into account relevant fundamental factors of the managed financial assets while limiting the number of necessary portfolio rebalances. Reinforcement learning algorithms are applicable to this problem, since the frequency of portfolio rebalances should be limited to avoid high accumulated transaction costs.

The illustrative portfolio includes several public utility stocks, because fundamental factors are generally more useful in predicting performance of value stocks as opposed to growth stocks. For the utilities sector, typical fundamental factors may include fuel prices, interest rates, weather conditions, currency exchange rates and others.

II. Related work

The idea of using machine learning methods for portfolio management has been widely discussed in the scientific literature, but so far little attention has been paid to improving these methods with knowledge of fundamental factors, which affect performance of the underlying securities.

In [2], authors develop neural networks to estimate future returns and risks of securities and to use them for construction of the Markowitz efficient portfolio.

In [3], asset allocation is formalized as a Markovian Decision Problem, which can be optimized by applying dynamic programming or reinforcement learning based algorithms.

In [4], authors compare two major reinforcement learning approaches for trading systems and portfolios: Q-learning and recurrent reinforcement learning. They conclude that the later generally outperform the former, but the Q-learning approach can improve on the recurrent reinforcement learning method when trading single securities.

In [5], authors develop a method which, instead of considering forecasts of the individual securities for portfolio allocation, gets the allocation directly as the output of a neural network maximizing the risk-adjusted return.

Finally, in [6], a neural network architecture is developed to establish a portfolio management system similar to the Black-Litterman approach, distributing funds across various securities while simultaneously complying with specific allocation constraints.

III. Dataset overview

The dataset is based on ca. 2150 daily observations spanning the period from 1st July 2008 to 30th December

2016. The data has been collected from various sources which are described in [7-12].

As potential portfolio stocks, three large public utility companies located in the northeast region of the U.S. have been selected, namely Public Service Enterprise Group (PEG), Consolidated Edison (ED) and Eversource Energy (ES). The fourth potential portfolio asset is a risk-free asset with a constant 1-year return of 1.0%.

In addition to the prices of the abovementioned financial assets, the dataset includes a number of variables (fundamental factors) that usually have a great impact on performance of utility companies. In particular, fuel prices of oil, natural gas, coal and uranium prices are used to account for operating costs of power stations. Financial market and economic conditions evidenced by UST curve, S&P 500 index, VIX index and 5-year expected inflation rate are used as determinants of regulatory pricing mechanism, which determines a fair level of "return on equity" for regulated business lines of utility companies. Weather conditions which affect demand and supply of electricity are measured by average temperature, wind speed and precipitation level in the New York area. Finally, the state of the electricity wholesale market is evidenced by 1-day ahead load forecast and clearing prices in the PJM and New England regions.

IV. Methodology

A. Technology stack

The algorithm was implemented on an Intel 2.2 GHz CPU system with 8 GB RAM and 64-bit Windows 10 operating system. The programming environment was based on Python 3.5 interpreter. Additional frameworks mainly included Numpy for operations with vectors, Pandas for operations with large datasets, Itertools for generation of item combinations and Keras for developing a deep neural network based on Tensorflow backend.

B. Preprocessing

The purpose of preprocessing of the dataset basically was to ensure stationarity of the analyzed time series data. Thus, log-difference transformation was applied to the variables that were susceptible to exhibit deterministic trend or seasonality.

Furthermore, the dataset was split into training and testing subsets. The training dataset consisted of observations from 1st July 2008 to 30th December 2015. The testing dataset consisted of observations from 1st January 2016 to 30th December 2016.

C. Action space

The action space was defined as a set of weights for each asset in the managed portfolio. The different combinations of asset weights were discretized in a way that all non-zero weights within each combination of assets were equal. For example, if some combination of assets included 1st, 2nd and 4th assets out of four possible assets, the corresponding portfolio weights were {1/3, 1/3, 0, 1/3}. Thus, the total number of discretized actions could be calculated as the following function of the number of portfolio assets:

$$||A|| = \sum_{k=1}^{n} C_k^n = 2^n - 1,$$

where n is the number of assets in the portfolio.

D. Reward function

The reward function, D_t, in the reinforcement learning algorithm corresponds to the differential Sharpe ratio, defined as follows:

$$A_t = A_{t-1} + \theta \Delta A_t, \;\; B_t = B_{t-1} + \theta \Delta B_t$$

$$\Delta A_t = R_t - A_{t-1}, \;\; \Delta B_t = {R_t}^2 - B_{t-1}$$

$$D_t = \frac{B_{t-1}\Delta A_t - 0.5 A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{1.5}}$$

Parameter R_t corresponds to the actual portfolio return in the period t, while parameter θ is set to be equal to 0.02. It was shown in [13] that models trained to maximize the differential Sharpe ratio achieve better risk-adjusted returns than those trained to maximize profit.

The actual portfolio returns are computed according to the following formula:

$$R_t = \sum_{k=1}^{n} \left(r_{t,k} \cdot A_{t,k} - tc \cdot \left| A_{t,k} - A_{t-1,k} \right| \right),$$

where tc denotes transaction costs equal to 5 bps per trade and $r_{t,k}$ is the actual return of k^{th} asset in the period t.

E. Q-learning algorithm

There are two major approaches used in reinforcement learning for automated portfolio management: deep recurrent Q-learning (DRQN) and recurrent reinforcement learning (RRL). Although it has been shown that the RRL approach generally has more stable performance and higher computational efficiency, this approach requires implementation of the backpropagation through time technique, which in turn requires analytical calculation of partial derivatives of the reward function. Thus, the DRQN approach is relatively easier to implement, allowing to try different model specifications by simply changing the structure of the Q-network.

For the purposes of the Q-Learning algorithm, the state of the process consists of two groups: partially observed and fully observed. The partially observed group included fundamental factors. The reason why they are considered partially-observed states is that they imperfectly reflect the consensus opinion of the investor community about future cashflow of the analyzed financial assets, while this opinion is the true determinant of the current asset prices. The fully observed group includes current portfolio allocation.

The exploration-exploitation dilemma is resolved according to the ε-greedy strategy. The value of ε is decaying over 500 iterations from the start value of 1.0 to the end value of 0.001.

The initial portfolio allocation is assumed to be 100% risk-free asset.

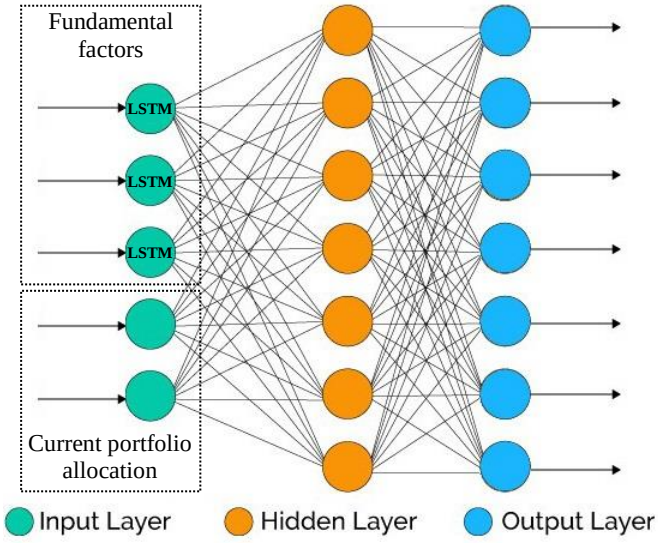
Parameters for Q-learning update formula are $\gamma = 0.95$ and $\alpha = 0.5$. Overall, the pseudocode of the algorithm can be summarized as follows:

UPDATE ε_t ACCORDING TO THE DECAY RULE
SET INITIAL PORTFOLIO ALLOCATION AS 100% RISK-FREE ASSET
FOR EACH OBSERVATION IN THE TRAINING DATASET:
  SELECT ACTION A_t ACCORDING TO ε_t-GREEDY STRATEGY
  OBSERVE REWARD D_t AND NEW STATE S_{t+1}
  $Q(S_t, A_t) = (1-\alpha)\cdot Q(S_t, A_t) + \alpha\cdot(D_t + \gamma\cdot \text{MAX}_A\, Q(S_{t+1}, A_{t+1}))$
  $S_t = S_{t+1}$
END FOR

F. Deep recurrent Q-network

Since the quantity of possible states of the automated portfolio management process is very high, it is not practical to maintain full Q-Matrix in the Q-Learning algorithm. Instead, the Q-Matrix is approximated by a deep neural network.

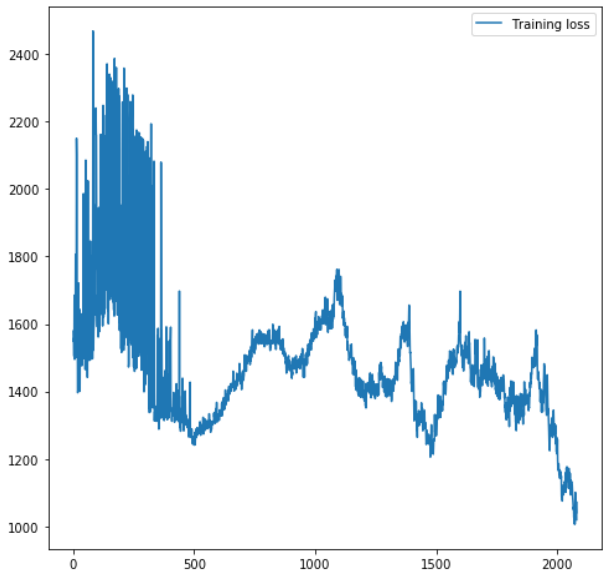
The structure of the neural network is recurrent and is designed to mirror the nature of the process states. Thus, the first layer of the network is hybrid, consisting of five recurrent LSTM cells and four standard feedforward cells. It was shown in [14] that LSTM-based Q-network can successfully integrate information through time and replicate performance of deep Q-Networks when other partially observed equivalents are used to account for time dependency. Thus, fundamental factors are used as inputs of the LSTM cells, while current portfolio allocation is received by the standard feedforward cells. Two additional layers of the network are dense. The output of the network reflects the total number of actions, with each output corresponding to the value of Q-Function for the given action. The following figure summarizes the structure of the used Q-network:



Because the size of the training dataset was relatively low, the experience replay used to train the Q-network included all training observations. The batch size of training algorithm was set at 64 and the epoch size – at 10. MSE was used as the loss functon and ADAM – as optimization method.

V. RESULTS

The average runtime of the learning algorithm was approximately 15 minutes per 100 iterations. Training loss, measured as absolute difference between predicted and target Q-values for a given experience replay, did not exhibit any converging pattern. A representative chart of the training loss behavior is presented below:



As a result, it was decided to stop the algorithm early – after 2,000 iterations.

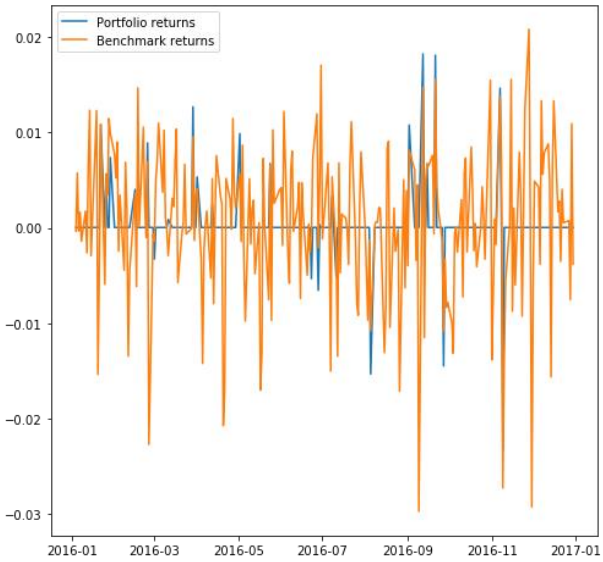
In order to evaluate performance of the algorithmically managed portfolio, it should be compared with a passively constructed benchmark portfolio. Two common benchmarks are the equal-weighted portfolio and the market-weighted portfolio. This paper focuses on the former one.

Although it is interesting to compare total returns of the algorithmically managed portfolio with the benchmark, this comparison would not be correct, as the algorithm was designed to maximize differential Sharpe ratio, i.e. a risk-adjusted return measure. Therefore, total Sharpe ratio on the test dataset can be used for the comparison. The resulting return measures are presented below:

	Portfolio	Benchmark
Total return	8.43%	11.23%
Sharpe ratio	1.42	0.81

As expected, the algorithm did not beat the benchmark portfolio on the total return basis. However, it has significantly outperformed it in terms of Sharpe ratio.

Looking at daily portfolio returns explains how the algorithm decided on portfolio allocation:



Basically, the algorithm preferred to keep all money in the risk-free asset unless there was a strong signal in the fundamental factors that some of the stocks may experience a series of positive returns.

VI. Conclusion

This paper presented an implementation of a deep recurrent Q-Learning algorithm for automated portfolio management which could take into account fundamental factors that normally drive performance of the underlying financial assets. I believe that the model suggested in this paper is quite natural solution for the automated portfolio management problem. Although the current implementation of the model already showed promising results on the test dataset, additional research should be conducted in this area to prove its viability and improve its efficiency.

Future research can study multiple areas. First, such parameters as θ and γ have a physical meaning and depend on investor's risk profile. So, additional models could be introduced to evaluate reasonable values of these parameters. Second, the number of discretized actions can be increased to allow building more precise investment strategies. Third, the exploitation-exploration trade-off can be handled with more advanced techniques, e.g. SoftMax. Fourth, the depth of the Q-Network could be increased with additional hidden layers or more complex versions of the Q-Learning algorithm could be employed, such as double Q-learning [15] or dueling network architecture [16]. Lastly, more data points should be collected for model testing, possibly on an hourly or more frequent basis. Also, data sources should be homogenies to avoid appearance of any artifacts.

VII. References

[1] CNBC News (https://www.cnbc.com/2017/07/12/machines-hedge-funds-humans-quants.html).

[2] D. Toulson and S. Toulson. Use of neural network ensembles for portfolio selection and risk management. 1996.

[3] R. Neuneier. Optimal Asset Allocation using Adaptive Dynamic Programming. 1996.

[4] J. Moody and M. Saffell. Reinforcement Learning for Trading Systems and Portfolios. 1998.

[5] J. Franke and M. Klein. Optimal portfolio management using neural networks - a case study. 1999.

[6] H. Zimmermann, R. Neuneier, and R. Grothmann. Active portfolio-management based on error correction neural networks. 2001.

[7] Yahoo Finance Historical Prices (http://finance.yahoo.com/).

[8] Quandl Core Financial Data & Alternative Data (www.quandl.com).

[9] FRED Economic Data (http://fred.stlouisfed.org/).

[10] Investing.com Historical Data (www.investing.com).

[11] U.S. Energy Information Association (http://www.eia.gov).

[12] Open Weather Map Historical Data (http://openweathermap.org).

[13] J. Moody, L. Wu, Y. Liao and M. Saffell. Performance functions and reinforcement learning for trading systems and portfolios. 1997.

[14] M. Hausknecht and P. Stone. Deep Recurrent Q-Learning for Partially Observable MDPs. 2015.

[15] H. Hasselt, A. Guez and D. Silver. Deep Reinforcement Learning with Double Q-learning. 2015.

[16] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot and N. Freitas. Dueling Network Architectures for Deep Reinforcement Learning. 2015.