Kolomeitsev Iurii

Online Portfolio Selection using Machine Learning

Early Research Project

Skolkovo Institute of Science and Technology, 2018

Contents

1 Introduction

Online portfolio selection (OLPS) is a theoretical and practical financial engineering problem, which aims to sequentially allocate capital among a set of assets in order to maximize long-term return. Nowadays there exists a variety of machine learning algorithms that solve this challenging problem. The main goal of this project is to study them, implement some of them, including benchmarks, state-of-the-art strategies and approaches involving deep learning, and finally to make a full comparison of all implemented strategies on different datasets and trading scenarios.

2 Related Work

In [1] the authors made an extensive survey of different online portfolio selection strategies including benchmarks and state-of-the-art algorithms and implemented OLPS toolbox in Matlab. However, their survey does not include approaches using deep learning and reinforcement learning.

In [2], [3] the authors investigate the Recurrent Reinforcement Learning approach for solving OLPS model for one asset. The performance functions that they consider for reinforcement learning are profit or wealth, economic utility, the Sharpe ratio and also they propose Differential Sharpe Ratio. However, they provide the solution of the OLPS problem only for one asset.

In this research we will consider multi-asset problem for all strategies and also for RRL models we will assume both buying and selling scenarios. For other strategies we will consider only buying scenarios.

3 Problem Formulation

The formulation of the portfolio optimization problem could be stated as follows. Let's consider m assets for n trading periods during which a trader buys and sells these financial instruments. Denote $p_t \in \mathbb{R}^m_+$ - close price vector at the t-th moment, p^i_t - close price of asset i at the t-th moment. Also let's denote $x_t = (y^1_t, \dots, y^m_t) \in \mathbb{R}^m_+$ as a price relative vector, where $y^i_t = \frac{p^i_t}{p^i_{t-1}}$. The sequence of price relative vectors for n periods will be written as: $x^n_1 = (x_1, \dots, x_n)$.

Before t-th period starts we should buy or sell assets according to a portfolio vector: $b_t = (\beta^1_t, \dots, \beta^m_t)$, where β^i_t is the fraction of wealth that we use to buy or sell the asset i. In all trading strategies except one, involving recurrent reinforcement learning, we assume that we can only buy assets, so $b_t \in \mathbb{R}^m_+$ and also no margin is allowed, so $b_t \in \Delta_m$, where $\Delta_m = \{b_t : b_t \in \mathbb{R}^m_+, \sum^m_{i=1} \beta^i_t = 1\}$.

At the beginning of the period we assume that $b_1 = \frac{1}{d}\mathbf{1}$. To solve the portfolio optimization problem we should make a strategy that will generate a new vector b_t at the beginning of t-th trading period and maximizes the portfolio cumulative wealth S_n. Denote $b^n_1 = (b_1, \dots, b_n)$ - strategy for n periods.

Portfolio period return:

$$s_t = b^T_t x_t = \sum^m_{i=1} \beta^i_t y^i_t$$

Portfolio cumulative wealth:

$$S_n(b^n_1, x^n_1) = S_0 \prod^n_{t=1} (b^T_t x_t),$$

where S_0 is the initial wealth. Further we will assume that $S_0 = 1$.

4 Strategies

In this section we will briefly describe the strategies which were implemented in this research project, namely: BAH, CRP, BCRP, Best Stock, Universal Portfolio, Exponential Gradient, Online Newton Step, Anticor, PAMR, OLMAR, RMR, BNN, CORN. All these strategies were implemented in Python and put into PyOlps library. Additionally, we describe Markowitz portfolio strategy that was used as a baseline. It's implementation was taken from portfolioopt Python library.

4.1 Benchmarks

4.1.1 Buy and Hold

Buy and Hold (BAH) strategy is investing wealth among a pool of assets with a portfolio b_1 and holding it until the end. So, the trader does not rebalance the portfolio at all. Portfolio cumulative wealth of BAH strategy:

$$S_n(BAH(b_1)) = b_1\big(\odot_{t=1}^n x_t\big)$$

Uniform Buy and Hold (UBAH): $b_1 = (\frac{1}{m}, \dots \frac{1}{m})$.

4.1.2 Best Stock

Best Stock strategy is a special case of BAH strategy when we put all our wealth on the stock with the best performance in hindsight:

$$b^* = \arg\max_{b \in \Delta_m} b\big(\odot_{t=1}^n x_t\big)$$

$$S_n(Best) = \max_{b \in \Delta_m} b\big(\odot_{t=1}^n x_t\big) = S_n(BAH(b^*))$$

4.1.3 Constant Rebalanced Portfolios

When we use Constant Rebalanced Portfolios (CRP) strategy we rebalance the portfolio to a fixed portfolio b every period. So, the strategy can be written as:

$$b_1^n = (b, \dots, b)$$

The portfolio cumulative wealth of CRP strategy:

$$S_n(CRP(b)) = \prod_{t=1}^n b^T x_t$$

Uniform Constant Rebalanced Portfolios (UCRP): $b = (\frac{1}{m}, \dots, \frac{1}{m})$.
We can calculate the optimal portfolio in the hindsight:

$$b^* = \arg\max_{b^n \in \Delta_m} \log S_n(CRP(b)) = \arg\max_{b^n \in \Delta_m} \sum_{t=1}^n \log(b^T x_t)$$

Using b^* we obtain Best Constant Rebalanced Portfolio (BCRP) strategy. The portfolio cumulative wealth of this strategy is:

$$S_n(BCRP) = \max_{b \in \Delta_m} S_n(CRP(b)) = S_n(CRP(b^*))$$

4.1.4 Markowitz portfolio

Markowitz portfolio is defined as the solution to the optimization problem:

$$\begin{aligned} & w^T \Sigma w \to \min \\ \text{s.t. } & w^T \hat{R} = \hat{R}_p \\ & |w^T| \mathbf{1} = 1, \end{aligned}$$

where $\hat{R} = \frac{1}{n}\sum_{i=1}^{n} R_i$ - vector of average returns of the assets, Σ - covariance matrix, $\hat{R}_p$ - the predefined minimum level of the return, typically $\hat{R}_p = 0.95$. During the optimization the model makes sure that the overall return of the portfolio is no less than the expected return and that the sum of weights is equal to one.

4.2 Follow-the-Winner Strategies

When using Follow-the-Winner strategy we increase the weights of more successful assets hoping that they will continue to rise in price.

4.2.1 Universal Portfolios

Universal Portfolios strategies are based on assigning the capital to a single class of base experts, let them run and then collect the wealth.This is similar to BAH strategy except that in UP strategies you have many sets of stocks and not just individual stocks.

In [4] Cover proposed the Universal Portfolio strategy. The main idea of UP strategy is to buy and hold parametrized CRP strategies over the whole simplex domain. Formally the update step can be written as:

$$b_{t+1} = \frac{\int_{\Delta_m} bS_t(b)d\mu(b)}{\int_{\Delta_m} S_t(b)d\mu},$$

where $d\mu$ is a portion of wealth invested to each CRP strategy.

The portfolio cumulative wealth of UP strategy:

$$S_n(UP) = \int_{\Delta_m} S_n(b)d\mu(b).$$

4.2.2 Exponential Gradient

The Exponential Gradient (EG) strategy is based on solving this optimization problem:

$$b_{t+1} \arg\max_{b\in\Delta_m} \eta \log bx_t - R(b, b_t),$$

where $R(b, b_t)$ is a regularization, η is the learning rate.

In the conventional EG strategy the regularization has the following form:

$$R(b, b_t) = \sum_{i=1}^{m} b_i \log \frac{b_i}{b_{t,i}}.$$

We can use 1-st order Taylor expansion of log at b_t:

$$\log bx_t \approx \log(b_t x_t) + \frac{x_t(b - b_t)}{b_t x_t}$$

Finally, we get the update step:

$$b_{t+1,i} = Cb_{t,i} \exp\big(\eta\frac{x_{t,i}}{b_t x_t}\big), \quad i = 1, \ldots, m$$

where C is the normalization term.

4.2.3 Online Newton Step

Online Newton Step (ONS) was proposed by Agarwal et. al. in [5]. The update step of ONS algorithm has the following form:

$$b_1 = (\frac{1}{m}, \ldots, \frac{1}{m}),$$

$$b_{t+1} = \prod_{\Delta_m}^{A_t} (\delta A_t^{-1} p_t),$$

where

$$A_t = \sum_{\tau=1}^{t} \Big(\frac{x_\tau x_\tau^T}{(b_\tau x_\tau)^2}\Big) + I_m,$$

$$p_t = (1 + \frac{1}{\beta}) \sum_{\tau=1}^{t} \frac{x_\tau}{b_\tau x_\tau},$$

where β - trade-off parameter, δ - scale parameter, $\prod_{\Delta_m}^{A_t}(\cdot)$ - projection to the simplex domain.

4.3 Follow-the-Loser Strategies

When using Follow-the-Winner strategy we transfer the wealth from winners to losers under the mean reversion assumption, which assumes that good-performing assets will perform poorly in the future and vice versa, poorly-performing assets will perform well.

4.3.1 Anti Correlation

Anti Correlation (Anticor) algorithm was proposed by Borodin et al. [6]. Using this approach we assume that market follows the mean reversion principle. The idea is to calculate the cross-correlation matrix between two market windows: $y_1 = \log(x_{t-2w+1}^{t-w})$ and $y_2 = \log(x_{t-w+1}^{t})$, where w is the window size.

$$M_{cov}(i,j) = \frac{1}{w-1}(y_{1,i} - \hat{y}_1)^T (y_{2,j} - \hat{y}_2)$$

$$M_{cor}(i,j) = \begin{cases} \frac{M_{cov}(i,j)}{\sigma_1(i)\sigma_2(j)}, & \sigma_1(i), \sigma_2(j) \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

Anticor algorithm moves the proportions from the stocks increased more to the stocks increased less, and the corresponding amounts are adjusted according to the cross-correlation matrix.

4.3.2 Passive Aggressive Mean Reversion

Passive Aggressive Mean Reversion (PAMR) was proposed in [7] by Li et. al. In this strategy we define ϵ-intensive loss function for the t^{th} period:

$$l_\epsilon(b, x_t) = \begin{cases} 0, & bx_t \leq \epsilon \\ bx_t - \epsilon & \text{otherwise} \end{cases}$$

where $0 \leq \epsilon \leq 1$ - sensitivity parameter that controls the mean reversion threshold. To make the step we should solve the optimization problem:

$$\begin{aligned} b_{t+1} = &\underset{b \in \Delta_m}{\arg\min} \frac{1}{2} \|b - b_t\|^2 \\ &\text{s.t. } l_\epsilon(b, x_t) = 0. \end{aligned}$$

Solving this problem we obtain:

$$b_{t+1} = b_t - \tau(x_t - \hat{x}_t \mathbf{1}),$$

where

$$\tau_t = \max\Big\{0, \frac{b_t x_t - \epsilon}{\|x_t - \hat{x}_t \mathbf{1}\|}\Big\}$$

4.3.3 Online Moving Average Reversion

PAMR strategy is single-period mean reversion. To exploit multiple-period mean reversion Li and Hoi proposed Online Moving Average Reversion (OLMAR) [8].

Instead of predicting next prices as last price: $\hat{p}_{t+1} = p_{t-1}$, as for example in PAMR algorithm, we can compute moving average: $\mathrm{MA}_t = \frac{1}{w}\sum_{i=t-w+1}^{t} p_i$. The next price relative:

$$\hat{x}_{t+1}(w) = \frac{\mathrm{MA}_t(w)}{p_t} = \frac{1}{w}\big(1 + \frac{1}{x_t} + \cdots + \frac{1}{\odot_{i=0}^{w-2} x_{t-i}}\big),$$

where w - the window size. To make the step we should solve the optimization problem:

$$b_{t+1} = \underset{b \in \Delta_m}{\arg\min} \frac{1}{2}\|b - b_t\|^2$$
$$\text{s.t. } b \cdot \hat{x}_{t+1} \geq \epsilon.$$

4.3.4 Robust Median Reversion

Huang et al. [9] used robust L_1-median estimator to exploit mean reversion and came up with Robust Median Reversion (RMR) strategy.

The main idea of this strategy is to estimate next price vector using robust L_1-estimator:

$$\hat{p}_{t+1} = L_1 med_{t+1}(w) = \mu_{t+1},$$

where w - window size and μ:

$$\mu_{t+1} = \underset{\mu}{\arg\min} \sum_{i=0}^{w-1} \|p_{t-i} - \mu\|$$

Then the price relative will be:

$$\hat{x}_{t+1}(w) = \frac{L_1 med_{t+1}(w)}{p_t} = \frac{\mu_{t+1}}{p_t}.$$

4.4 Pattern-Matching Strategies

Pattern-Matching based strategies may utilize both winners and losers. These approaches consider the non i.i.d. market and maximize the conditional expectation of the return using past observations.

The Pattern-Matching strategies basically consist of two steps: Sample Selection and Portfolio Optimization.

In the first step we select an index set I of similar historical price relatives that will be used to predict the next price relative. Each sample price relative $x_i, i \in I$ is assigned with probability $P_i, i \in I$. Commonly used probability values are: $P_i = \frac{1}{|I|}$.

Portfolio Optimization step:

$$b_{t+1} = \underset{b \in \Delta_m}{\arg\min}\, U(b, I),$$

where $U(b, I)$ - utility function. The default utility function is: $U(b, I) = \sum_{i \in I} \log b^T x_i$.

Log optimal portfolio:

$$U_L(b, I(x_1^t)) = \mathbb{E}\big(\log b \cdot x \big| x_i, i \in I(x_1^t)\big) = \sum_{i \in I(x_1^t)} P_i \log b \cdot x_i,$$

where P_i - probability assigned to a similar price relative $x_i, i \in C(x_1^t)$.

4.4.1 Nonparametric Nearest Neighbor Log-optimal Strategy

Nonparametric Nearest Neighbor Log-optimal Strategy (BNN) was proposed by Gyorfi et al. [10]. This strategy combines the nearest neighbor sample selection and log-optimal utility function.

The nonparametric nearest neighbor (NN) sample selection searches the price relatives using this expression:

$$I_N(x_1^t, w) = \{w < i < t+1 : x_{i-w}^{i-1}\text{is among the } l \text{ NNs of } x_{t-w+1}^t\},$$

where l - number of neighbors that is the threshold parameter.

4.4.2 Correlation-driven Nonparametric Learning Strategy

Correlation-driven nonparametric learning approach (CORN) was proposed by Li et al. [11]. This method combines the correlation driven sample selection and log-optimal utility function.

The correlation driven sample selection:

$$I_C(x_1^t, w) = \big\{w < i < t+1 : \frac{cov(x_{i-w}^{i-1}, x_{t-w+1}^t)}{std(x_{i-w}^{i-1})std(x_{t-w+1}^t)} \geq \rho\big\},$$

where ρ is a pre-defined correlation coefficient threshold.

5 Reinforcement Learning

Reinforcement Learning is a machine learning technique when a system (agent) is trained by interacting with an environment and uses the received feedback from its own actions and experiences to learn the optimal behavior. An action moves an agent to a new state and agent gets some reward that can be positive or negative. The goal is to find a suitable action model that would maximize the total cumulative reward of the agent. The figure 1 represents the basic idea and elements involved in the reinforcement learning model.

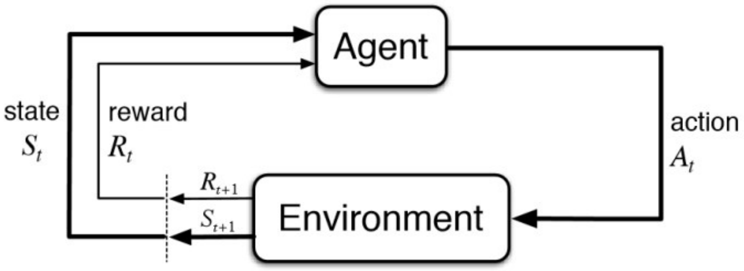


Figure 1: reinforcement learning scenario

Formally we can formulate the problem as follows. Let's denote $s \in S$ as environment's state, $a \in A$ - agent's action, $r \in R$ - reward, $P(s_{t+1}|s_t, a_t)$ - probabilities of moving from one state to another by making an action a_t. The cumulative reward is: $R = \sum_t r_t$.

Denote $\pi(a|s)$ as the agent's strategy that is the probability of taking action a in the state s. The main goal is to find a strategy that maximizes the cumulative reward:

$$\pi(a|s) : \mathbb{E}_\pi(R) \to \max$$

5.1 Recurrent Reinforcement Learning

In OLPS problem we should use not only features and returns on the previous time period, but consider the past with specified window size. Thus, we have a recurrent structure of the problem and the basic reinforcement learning scenario is not enough for solving this task. In our case we should use the Recurrent Reinforcement Learning (RRL). RRL model remembers hidden state to generate the next output. Namely, we will use Recurrent Neural Network (RNN) as the model to predict the portfolio vectors.

In the Figure 2 we can see the basic RNN model's structure. RNN networks take not only the input as the ordinary neural networks, but also the previous output of the networks. Therefore, such networks can find long-term dependencies in the data. However, basic RNN suffers from vanishing gradients problem, when small numbers are being multiplied many times, so they fail to find really long-term dependencies. That's why Long Short-Term Memory (LSTM) network was developed.

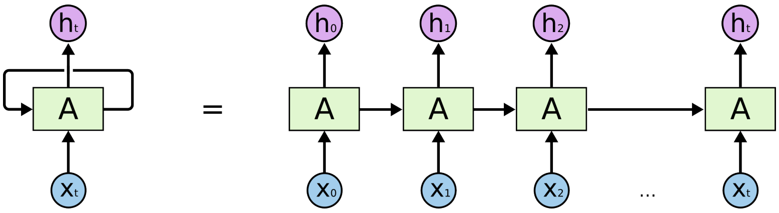


Figure 2: Recurrent Neural Network

5.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) network is a special kind of architecture of recurrent neural networks designed to find long-term dependencies that was proposed in [12]. The LSTM model is shown in the Figure 3.

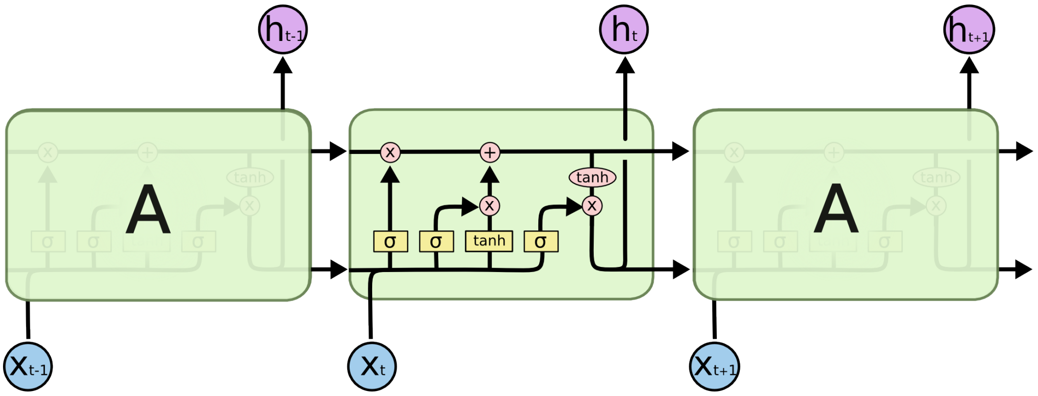


Figure 3: LSTM model

Basically LSTM network consists of 4 layers instead of one as in the standard RNN. Let's see what happens in each layer in more details.

The first layer computes how hard should we remember the previous information:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f).$$

In the second layer we figure out how important is the information that we get from new signal:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i),$$

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C).$$

In the third layer we compute the new state that is going onwards:

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t.$$

In the last forth layer the output of LSTM cell is being computed:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o),$$

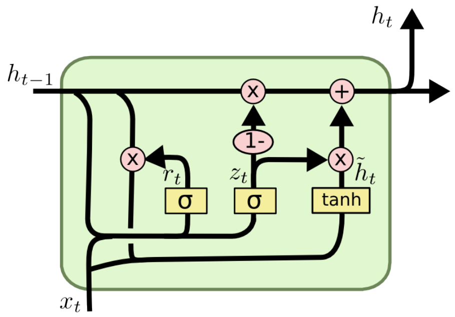
$$h_t = o_t * \tanh(C_t).$$

There are several modifications of LSTM model. One of the most popular is the Gated Recurrent Unit (GRU).

5.3 Gated Recurrent Unit

Gated Recurrent Unit (GRU) model was originally proposed in [13]. In this type of architecture layers that utilize the input and the previous output of the network are being combined into one update gate. Also, the output and the hidden states are united. In the Figure 4 one can see GRU model and all related formulas.

Often GRU networks work better than LSTM networks. This happens because GRU models have less parameters, so they are simpler to fit properly and do not tend to overfit much. In this project both LSTM and GRU based RRL models were tested and compared with each other.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 4: GRU model

5.4 RRL model for OLPS problem

Let's now describe the solution of the OLPS problem using RRL model that was introduced in [2] and [3] for one asset. As before denote: p_t - price of the asset at the time t. As input data we will use price relatives shifted by 1 towards zero:

$$r_t = \frac{p_t}{p_{t-1}} - 1$$

The decision function:

$$F_t = \tanh\Big(\sum_{i=0}^{m-1} w_i(t) r_{t-i} + w_m(t) F_{t-1} + w_{m+1}\nu\Big)$$

If $F_t > 0$ then a trader is in the LONG position (is buying the asset), if $F_t < 0$ then a trader is in the SHORT position (is selling the asset). w_i are the parameters that are needed to optimize. And ν is the bias. Sometimes sign function is considered instead of tanh.

The rate of return of the asset could be computed like this:

$$R_t = r_t * F_t$$

As the optimizing functional let's take the Sharpe Ratio:

$$S_T = \frac{\hat{R}_T}{\sigma(R_T)},$$

where $\hat{R}_T = \frac{1}{T}\sum_{k=0}^{T} R_k$, $\sigma(R_t) = \sqrt{\frac{1}{T}\sum_{k=0}^{T}(R_k - \hat{R}_T)^2}$.

In order to get better results let's also consider the smoothing of the optimized functional by exponential moving average:

$$EMA_t = \theta p_t + (1-\theta) EMA_{t-1},$$

where p_t is the value of the function at the moment t and $\theta \in [0, 1]$.

Considering a moving average of Sharpe Ratio we will get The Differential Sharpe Ratio:

$$\hat{S}(T) = \frac{A_t}{B_t},$$

where

$$A_t = A_{t-1} + \theta(R_t - A_{t-1}) = A_{t-1} + \theta \Delta A_t,$$
$$B_t = B_{t-1} + \theta(R_t^2 - B_{t-1}) = B_{t-1} + \theta \Delta B_t,$$

where A_t and B_t are exponential moving estimates of the first and second moments of R_t.

The Differential Sharpe Ratio:

$$D_t = \frac{d\hat{S}_t}{d\theta}|_{\theta=0} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}}$$

Let's compute the gradient of DSR:

$$\frac{dD_t}{dw_t} = \frac{dD_t}{dR_t}(\frac{dR_t}{dF_t}\frac{dF_t}{dw_t} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{dw_{t-1}})$$

$$\frac{dD_t}{dR_t} = \frac{B_{t-1} - A_{t-1}R_t}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}}$$

6 Experiments

6.1 Dataset description

As a dataset let's take 6 popular stock market indices for 8 years: S&P 500, Hang Seng, CSI 300, DJIA, Nikkei 225, Nifty 50.

The time period is from July 2008 till August 2016: all in all 2005 trading days. We split the data into train and test, train is 75% and test is 25%. We fit our RRL model on train set and then compare it with other benchmarks and state-of-the art algorithms on test set (mainly from PyOlps package that was also implemented under this research project).

While training deep learning models considering plenty of features can give substantial profit. So let's use not only Closing price data, but also other features connected with the Index (all in all 19 features).

We will use these features: Open, High, Low, and Close prices, moving average convergence divergence (MACD), commodity channel index (CCI), average true range - volatility of price (ATR), Bollinger Band (BOLL), exponential moving average for 20 days (EMA20), moving average for 5 and 10 days (MA5, MA10), 6 and 12 month momentum (MTM6, MTM12), price rate of change - speed at which a stock's price is changing (ROC), stochastic momentum index (SMI), William's Variable Accumulation Distribution - buying and selling pressure (WVAD), US Dollar Index - macroeconomic variable, interbank Offered Rate / Federal Fund Rate - macroeconomic variable (SHIBOR - Shanghai Interbank Offered Rate, HIBOR - Hong Kong Interbank Offered Rate).

To make the experiments closer to reality we consider the commission rate for transactions. In all experiments it is equal to 0.01% both for selling and buying.

6.2 Comparison with baselines

The following strategies are considered as baselines: UCRP, EG and Markowitz. The comparison between basic RRL model and baselines is shown in the Figure 5. We can see that the RLL model can achieve the quality higher than baselines by both metrics: Sharpe and Total return.

The comparison between RRL with DSR model and baselines is shown in the Figure 6. We can see that using DSR leads to better quality of the model. The Sharpe Ratios of basic RRL and RRL with DSR are 1.339 and 1.969 respectively. The final returns of basic RRL and RRL with DSR are 0.36 and 0.43 respectively. So the profit of using DSR is significant. Apart from higher quality DSR has lower computation complexity in comparison with basic Sharpe Ratio: computing SR at every point leads to $\mathcal{O}(n^2)$ algorithm but computing DSR leads only to linear complexity. So, the RRL model with DSR is much faster and is more suitable for solving OLPS problem.

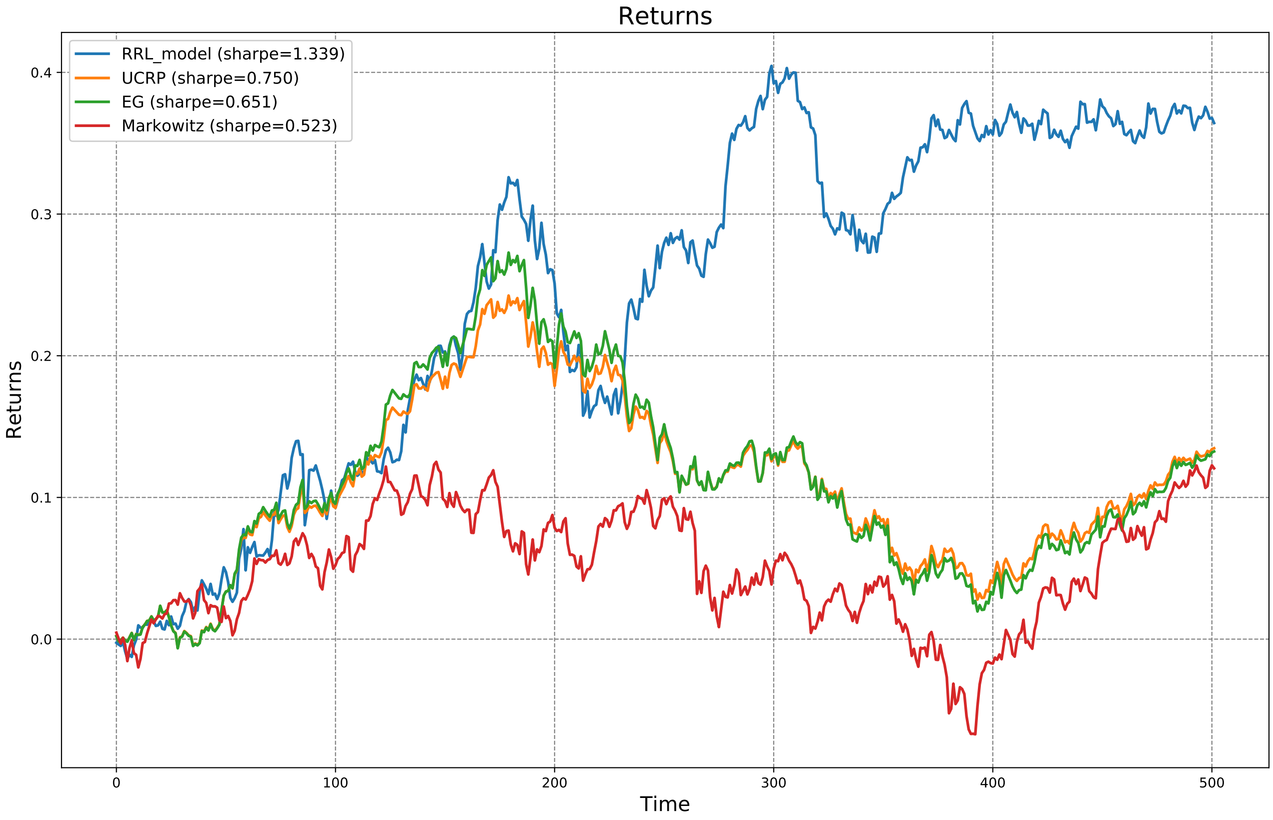


Figure 5: RRL returns compared with baselines

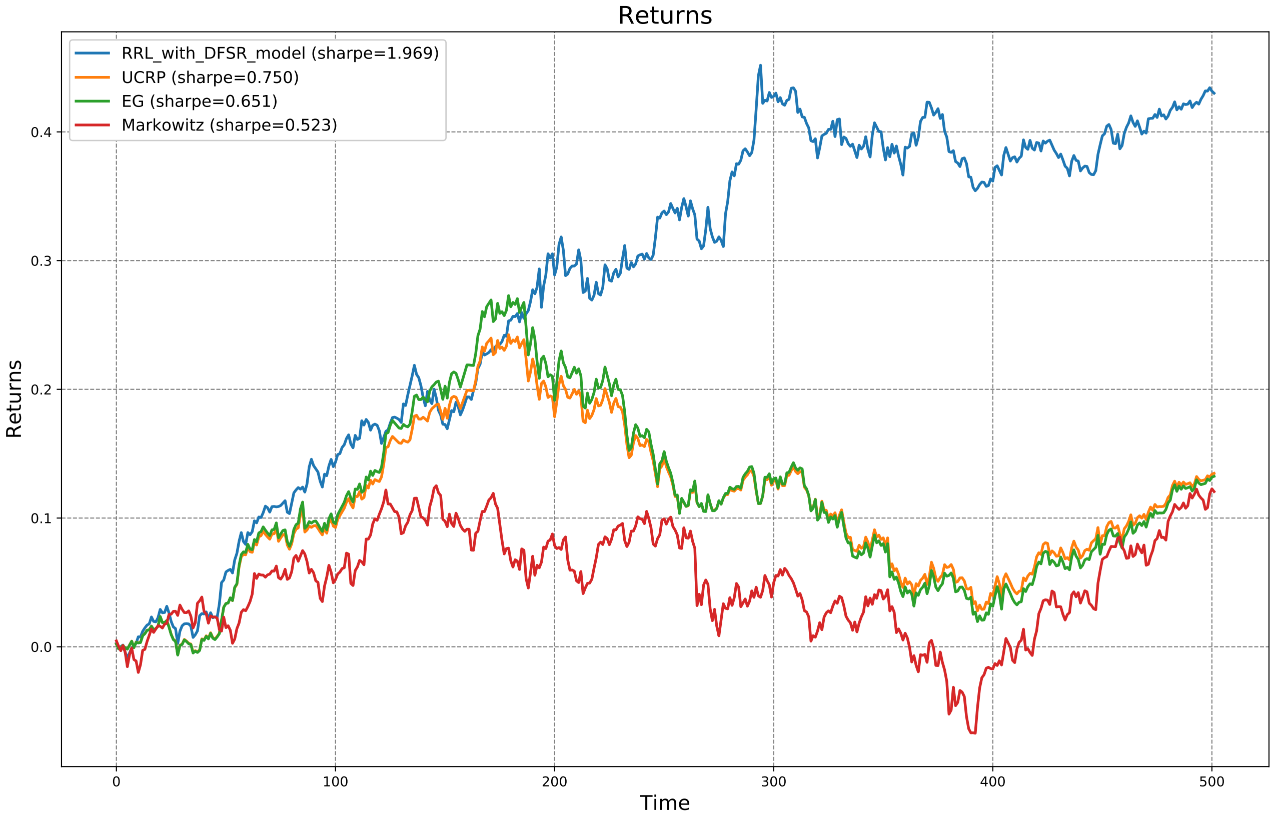
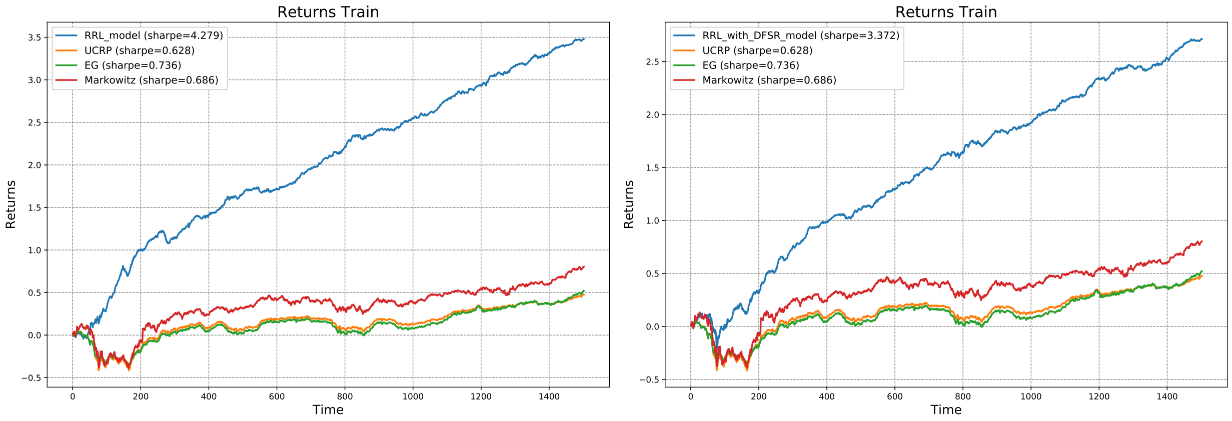


Figure 6: RRL with Differential Sharpe Ratio returns

In the Figure 7 we can see the comparison of the strategies on train set. We can see that RRL models highly outperforms baselines. Also, we see that basic RRL model overfits more than RRL with DSR. The final returns of RRL model and RRL with DSR are 3.5 and 2.61 respectively.



(a) RRL

(b) RRL with DSR

Figure 7: RRL returns on train set

6.3 LSTM and GRU comparison

Let's compare LSTM and GRU based architectures. The comparison is shown in the Figure 8 and Table 1. We can see that RRL with GRU model slightly outperforms RRL with LSTM model by both metrics: Sharpe Ratio and Returns.

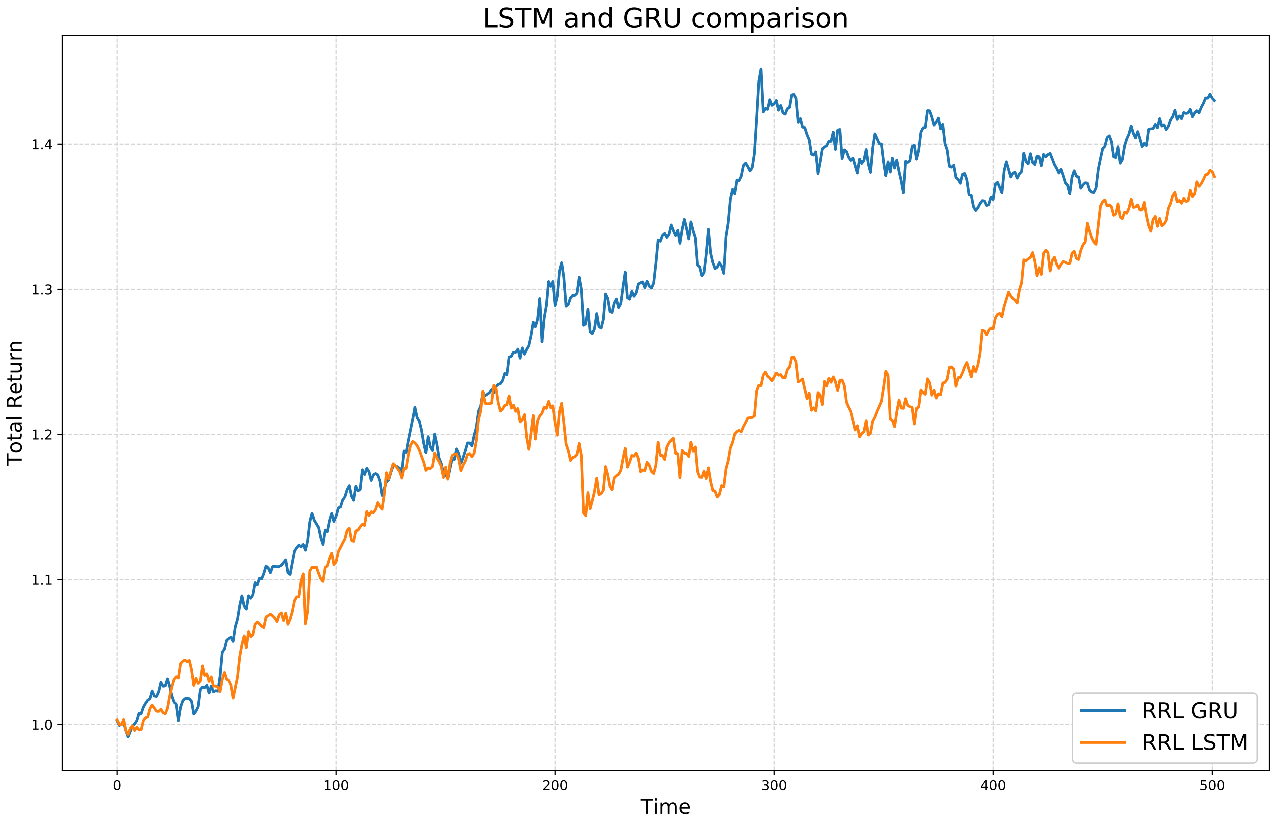


Figure 8: LSTM and GRU comparison.

Strategy	Sharpe	Returns
RRL GRU	1.969383	1.430047
RRL LSTM	1.799645	1.377632

Table 1: LSTM and GRU comparison.

7 Results

Let's compare RRL strategy with strategies from PyOlps library. Namely, we will compare GRU-based RRL with DSR strategy with these benchmark and some state-of-the-art algorithms: UBAH, BCRP, Best Stock, EG, ONS, Anticor(Anticor), UP, UCRP, EG, Anticor, OLMAR-1, OLMAR-2, RMR, PAMR, BNN, CORN. In the Figure 9 we can see the results of the final comparison.

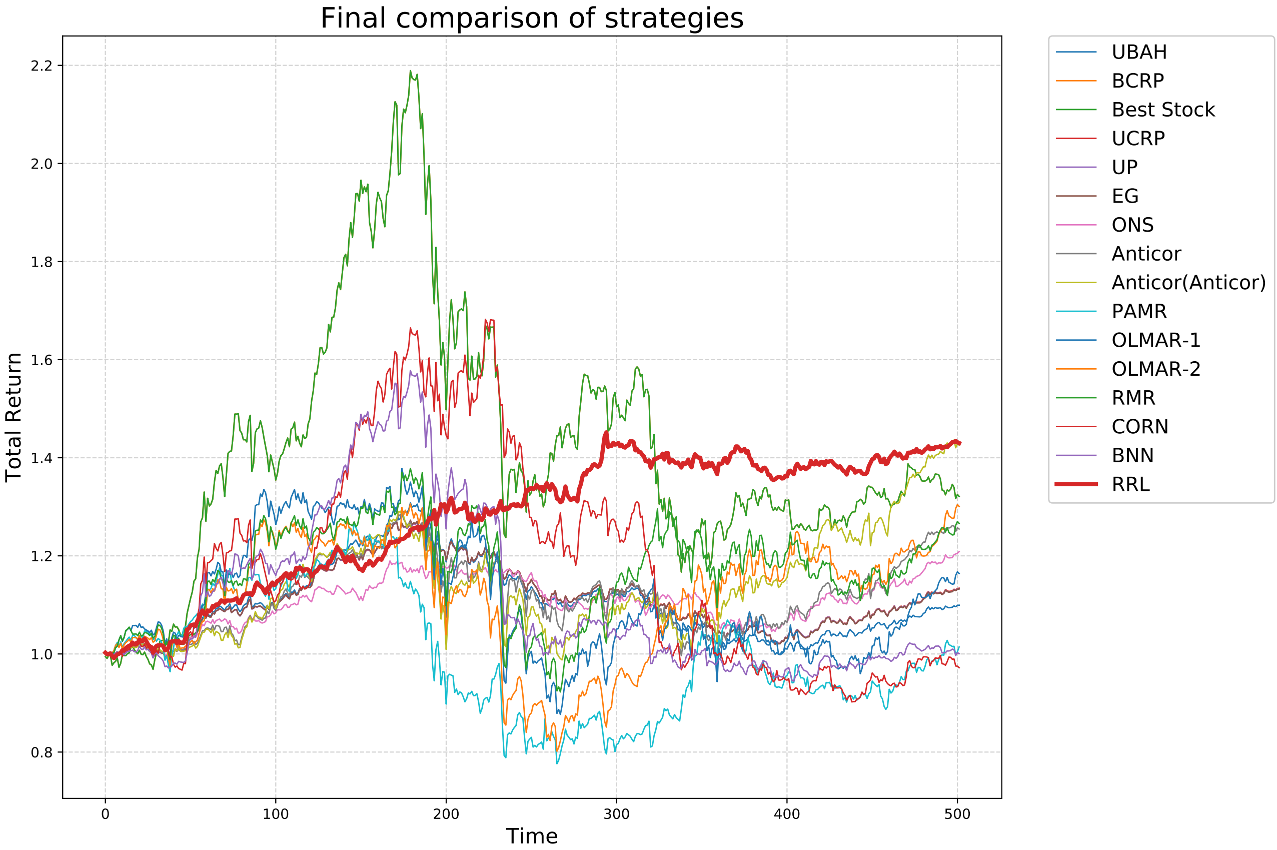


Figure 9: Final results: RRL with DSR

We can see in the figure that starting from around 2/3 period of our test dataset, RRL strategy (thick red line) outperforms all other strategies. Moreover, we see that, unlike many other strategies, RRL is smooth and straightforward, it doesn't have any big jumps. This is also confirmed by the high Sharpe ratio: 1.969 which is almost 2 times higher than the non-RRL predecessor ONS: 1.016. In the Table 2 we report the exact values of Sharpe Ratio and Returns for each strategy sorted by Sharpe.

If we have high Sharpe that means we have low risks of the portfolio. This is a very big advantage of RRL strategy as having low risks is one of the main priorities of traders.

Other algorithms that are in the top (Total Return) at the end of test period are: Anticor(Anticor), Best Stock, OLMAR-1, OLMAR-2, RMR, Anticor (listed in the order of their descending quality).

This shows us that using reinforcement learning with deep learning for solving portfolio selection task can give very profitable and stable results.

8 Conclusion

During this research project many benchmark and state-of-the-art OLPS algorithms were implemented in Python. The implemented strategies are: BAH, CRP, BCRP, Best Stock, Universal Portfolio, Exponential Gradient, Online Newton Step, Anticor, PAMR, OLMAR, RMR, BNN, CORN.

Also, Recurrent Reinforcement Learning (RRL) strategy was investigated and implemented. This model uses reinforcement learning and deep learning approaches such as RNN, LSTM, GRU. LSTM and GRU approaches were compared with each other. All implemented strategies were compared on

Strategy	Sharpe	Returns
RRL with DSR	**1.969383**	**1.430047**
RRL	1.136594	1.251198
ONS	1.016901	1.208643
Anticor(Anticor)	0.840856	1.425623
UP	0.662710	1.134559
UCRP	0.661377	1.134104
EG	0.650659	1.132446
Anticor	0.594237	1.254682
OLMAR-2	0.488413	1.300776
UBAH	0.443488	1.099442
RMR	0.396724	1.266272
BCRP	0.322003	1.321177
Best Stock	0.322003	1.321177
OLMAR-1	0.250928	1.164077
PAMR	0.026447	1.014283
BNN	0.001162	1.000596
CORN	-0.039490	0.971911

Table 2: Final results sorted by Sharpe ratio.

different datasets. It turned out that in the experiments the best strategy was GRU-based RRL with DSR.

9 GitHub repository

In my repository on GitHub: `https://github.com/kolomeytsev/online_portfolio_selection` you can get the code and the comparison of all models.

References

[1] Bin Li and Steven Hoi. Online portfolio selection: A survey. 46, 12 2012.

[2] C. Gold. Fx trading via recurrent reinforcement learning. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.*, pages 363–370, March 2003.

[3] Moody John, Wu Lizhong, Liao Yuansong, and Saffell Matthew. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(5-6):441–470.

[4] Cover Thomas M. Universal portfolios. *Mathematical Finance*, 1(1):1–29.

[5] Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E. Schapire. Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 9–16, New York, NY, USA, 2006. ACM.

[6] Allan Borodin, Ran El-Yaniv, and Vincent Gogan. Can we learn to beat the best stock. *CoRR*, abs/1107.0036, 2011.

[7] Bin Li, Peilin Zhao, Steven C. H. Hoi, and Vivekanand Gopalkrishnan. Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning*, 87:221–258, 2012.

[8] Bin Li, Steven C.H. Hoi, Doyen Sahoo, and Zhi-Yong Liu. Moving average reversion strategy for on-line portfolio selection. *Artif. Intell.*, 222(C):104–123, May 2015.

[9] D. Huang, J. Zhou, B. Li, S. C. H. Hoi, and S. Zhou. Robust median reversion strategy for online portfolio selection. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2480–2493, Sept 2016.

[10] László Györfi, Frederic Udina, and Harro Walk. Nonparametric nearest neighbor based empirical portfolio selection strategies. 99:999–9999, 01 2003.

[11] B. Li, D. Huang, and S. C. H. Hoi. CORN: Correlation-Driven Nonparametric Learning Approach for Portfolio Selection - an Online Appendix. *ArXiv e-prints*, June 2013.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(9):1735–1780, November 1997.

[13] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.