Mitigating Risk in Machine Learning-Based Portfolio Management: A Deep Reinforcement Learning Approach

1st Fidel Esteves do Nascimento
Electronic and Computer Engineering
Institute of Aeronautics and Technology
Sao Jose dos Campos, Brazil
fidel.nascimento@ita.ga.br

2nd Paulo André Lima de Castro
Electronic and Computer Engineering
Institute of Aeronautics and Technology
Sao Jose dos Campos, Brazil
pauloac@ita.br

***Abstract*—Many researchers when using machine learning approaches for investing focus on predicting returns, but overlook risk management or portfolio allocation. That may lead to risk management problems or even uncontrolled strategy risk. In order to address such vulnerability, we propose a framework for training an allocation strategy that models the investment process as a Markov decision process. We use a risk decomposition technique and a deep reinforcement learning agent model to compute variances and update portfolio variables efficiently. We tested our framework with a LSTM-based agent for predicting stock movements and found that it was able to offset systematic risk and achieve reasonable returns. The proposed low-dimensional framework may contribute to more effective portfolio management based on reinforcement learning.**

***Index Terms*—AI, Reinforcement Learning, Trading**

I. Introduction

Investment activity involves a wide range of concerns, such as diversification, market impact, broker costs, bet size, timing, and accuracy. One important aspect of investment is portfolio allocation, which seeks to maximize profits while controlling risk. Markowitz's approach to portfolio allocation has been influential, but it has also been criticized for its practical limitations. To solve the bet sizing problem, many methods have emerged that seek to estimate the risk of a portfolio based on the covariance matrix of its assets. However, the dimensionality of the covariance matrix can make it highly susceptible to estimation errors.

To consistently create profitable portfolios, investors need to perform forecasts about future performances. Forecasting involves using raw forecasts to estimate the next return of the target assets. The accuracy and timing of these forecasts are crucial for allocating money in the right direction and at the right time. However, predicting future movements of financial assets is a challenging problem with a low signal-to-noise ratio. The efficient market hypothesis (EMH) states that asset prices are a reflection of all available information at a given moment in time. While there are some anomalies in the financial market that make opposite evidence to the EMH, these anomalies do not have statistical significances as high as other well-known problems that humans have shown to be capable of solving.

To improve the predictive power of the forecast formula, investors have turned to machine learning techniques. These techniques can be used to learn from large volumes of data and identify patterns that are not easily observable by humans. However, they also have limitations, such as overfitting, data snooping bias, and model instability. To address these limitations, investors need to carefully design their models, validate them using out-of-sample data, and implement them in a disciplined and systematic way. In short, investment activity is a complex and challenging problem that requires a combination of mathematical, statistical, and machine learning techniques, as well as a deep understanding of market dynamics and human behavior.

When it comes to machine learning papers on portfolio allocation, most of them focus on price prediction and return maximization, rather than risk control and portfolio allocation [1]. The existing approaches can be classified into two clusters, but they have their limitations. In this article, we introduce a new approach to portfolio allocation using reinforcement learning (RL) and risk decomposition.

The first approach to portfolio allocation involves using supervised learning to predict the direction of next prices movements, followed by a naive portfolio allocation, such as equal weights. The second approach uses a reinforcement learning algorithm to perform weights allocation among the assets. However, the input size increases linearly with the number of assets, which is a major disadvantage of this approach.

We propose a new approach that uses a reinforcement learning method that receives the covariance structure with a fixed input shape, regardless of the number of negotiated stocks. To do this, we use a risk decomposition method that has been proven to be a good way to estimate portfolio risk even before reinforcement learning. We reduce the input size by using only two or three features per asset and computing an even more accurate correlation matrix. This method reduces the number of points from whatever you choose the time series size to two or three, which is already a great dimension reduction.

Our approach allows the RL model to look at one stock at a time, which allows it to know how the entire portfolio risk would change when changing the weights of that specific stock. As a result, the input for risk control decreases from an entire matrix of timeseries to a simple vector of two or three risk measures. This approach allows us to feed the model with other features with proper trading signal without being obfuscated by the input matrix.

Our approach to portfolio allocation using reinforcement learning and risk decomposition is a promising new method that addresses some of the limitations of existing approaches. With our approach, automated portfolio management can keep the benefits of reinforcement learning, such as being trained to maximize profits with or without costs, market constraints like liquidity and several other market properties, without the obligation of high-dimensional inputs. We hope that our approach can contribute to the development of better trading algorithms and risk management strategies in the future.

II. Related work

Markowitz's classic work on portfolio selection, [2], established the concept of risk minimization and presented frameworks for optimal allocation among assets. However, due to the noisy trading environment, these frameworks tend to produce poor results in out-of-sample data. To address this issue, several methods were developed, including heuristic methods that balance estimation risk and assumption risk. These methods include naive risk parity, inverse of variance, inverse of VAR, Hierarchical Risk Parity, network distance allocation, and minimum-variance and risk parity. Mean-variance optimization is also used, which incorporates risk, co-risk, and return estimates.

With the rise of RL in investment activity, there has been an increase in its application to diversification and optimal allocation. Works such as [3], [4] and [5] have explored RL in investment, while [6] have used supervised learning with several economic features to perform predictions about stocks' next price moviments. However, supervised learning methods have additional logic layers and lack correspondence between training and trade targets, making them less effective than RL in investment. The author of [5] presents an RL approach where the agent observes stock prices and decides on portfolio weight, using Deep Reinforcement Learning. Another promising approach has been developd in [7], where the author used adversarial neural networks to find the stochastic factor of discount, meaning the tangent portfolio.

The usage of risk factors for portfolio allocations is not completely new. [8] and [9] show factor-based risk parity approaches for asset allocation. Although a literature that combines risk factors with RL to portfolio allocation is not know by the authors at this point of time, [10] presents the math fundamentals to use risk exposures as proxy of risk and dimensionality reduction when computing portfolio risk and covariance structure. In this work, the author uses different metrics and methodologies to evaluate the covariance estimates by using exposure to risk factors and find favorable results for the usage of such method.

III. Methodology

The problem of portfolio management consists in dynamically changing the allocations in a set of financial instruments with the purpose of providing a good wealth over time. For sure, each investor will have a definition of what is a "good wealth", and the quantitative investors actually should have an objective metric to evaluate how good is the wealth evolution over time.

The present work deals with the portfolio management problem as a Markov decision process. In this case, the agent interacts via backtest with a financial environment with the purpose of maximizing its profits as well as controlling its risk. The core part of the general framework to train the agent is reinforcement learning. In this model, the agent is modeled with a pair of neural networks, which are trained interacting with a trading environment. The trading environment is a software that uses historical data to simulate the enrollment of the financial market.

A. Financial Market Variables

The dividends and splits adjusted closing price for the stock i at the timestamp t we call $p_{i,t}$. The stock financial return from $t-1$ to t is written as $r_{i,t} = 1 - p_{i,t-1}/p_{i,t}$.

The estimated volatility for the stock i at timestamp t is calculated as the standard deviation of returns using a window of size 20, calculated using returns $r_{i,t-20}$ to $r_{i,t-1}$. The window size of 20 days was chosen to capture fast changes in the stock volatility.

The estimated correlation between the stock i and the market factor at timestamp t, $\rho_{i,t}$, is the Pearson correlation of the stock and market factor returns from $t-60$ to $t-1$. The window size of 60 days was chosen because the correlation metric tends to be more stable over time.

The covariance matrix for the stocks is calculated based on risk factors. The variances are just the square of the volatilities, while the covariances are calculated as

$$cov_{i,j,t} = \rho_{i,t} * \rho_{j,t} * \sigma_{i,t} * \sigma_{j,t} \tag{1}$$

assuming that the market idiosyncratic returns [1] of the stocks i and j is zero, the covariance formula in equation 1 can be proved assuming that the idiosyncratic returns of two given assets is zero.

Given the weight applied in the stock i at time t, $w_{i,t}$, and the vector of weights at time t, w_t, the portfolio volatility at the beginning of day t is calculated as

$$\sigma_t = w_t^T \mathcal{M} w_t, \tag{2}$$

where $\mathcal{M}$ is the covariance matrix.

[1]Idiosyncratic returns, or residual returns are the residuals of the linear regression between the stock and market factors returns. Saying that the idiosyncratic returns of two assets have zero correlation means that their correlation is lead exclusively from the exposure to the common factor.

The correlation between the portfolio and the market factor at the beginning of day t:

$$\rho_t = \sum_i w_{i,t} * \rho_{i,t} * \sigma_{i,t} / \sigma_t$$

B. Risk decomposition

For each stock, at each day, there are calculated two variables. The first variable is the estimated stock volatility, and the second the estimated correlation between the stock and the market factor.

Although the following calculation is not made directly, it is important to understand the choice of the two variables. First, the systematic risk of the stock is calculated with the product of the former two variables:

$$systematic\_risk = \rho * \sigma$$

Second, the idiosyncratic risk of the stock is the leftover risk:

$$\sigma^2 = systematic\_risk^2 + idiosyncratic\_risk^2 \qquad (3)$$

The sum in equation 3 describes the risk decomposition into systematic and idiosyncratic risks for only one common factor of risk (in our case the SPY as proxy of market factor).

The two kinds of risk sum up in different ways when building a portfolio; when we average idiosyncratic risks of two assets it is computed as a quadratic average, whereas when averaging systematic risks it is computed linearly. For that reason the systematic risk is frequently called non-diversifiable risk, whereas the idiosyncratic risk is called diversifiable risk.

C. Forecasting

The bet timing and accuracy are strongly related with the forecasting activity. The RL agent could use the raw forecasts to be accurate in its bets. Those raw forecasts are in general price action features or fundamentals [6]. However, we chose to deliver to the agent processed forecasts, in a way to reduce both the input dimensionality and the training time.

As a forecasting function, it was chosen to be a LSTM model, which is a recurrent neural network, see [11]. Here are used 60 points of past financial returns to predict 1 step ahead return. Thus, in time-step t, the returns $r_{i,t-60}, \ldots, r_{i,t-1}$ were used as features to predict $r_{i,t+1}$. The architecture of the LSTM is composed of ReLu gate functions, 20 hidden units and was not subjected to any tuning process. For more information on the LSTM implementation refer to [12]. Although the forecasting function is based on a recent machine learning model, it is not the core of this work. As mentioned previously in this section, theoretically, the RL agent could receive as input raw forecasts, without the need of any exogenous model.

D. Reinforcement Learning

The RL agent in the defined problem is a portfolio manager. It receives data from the financial environment and decides what actions to take. The data it receives consists of information about specific stocks, as well as portfolio information. The action it takes is simply the amount of capital to be allocated in a specific stock at a specific point in time.

Normally, the RL literature of portfolio management defines as the agent's action a vector of weights:

$$a_t = w_t$$

It assumes that for each point of time, the agent processes information about all assets and performs allocations for all at once.

For the present work, the step is not only divided by days, but also by investment. On each day, the agent chooses the investment stock by stock. That implies that the input dimension of the agent can be drastically reduced. With that, while the temporal tracked time-step is designated with the index t, the RL time-step is designated with the index (i, t), i designating i-th the stock of day t.

In order to properly define a reinforcement learning environment, it is necessary to describe its primordial elements: The spaces of states, actions and rewards and the transition dynamics. Those elements are described next.

1) State Space: The state is a vector of dimension 7, being composed by the following features:

1) The estimated portfolio volatility at time t, σ_t
2) The estimated volatility for stock i at time t, $\sigma_{i,t}$
3) The stock prediction (from the LSTM algorithm) for time $t+1$, $\hat{y}_{i,t}$
4) The maximum weight allocation that is allowed, $availableWeight_t$
5) The weight currently allocated at the stock i at time t, $w^-_{i,t}$. The "-" superscript states that this is the weight before the agent make the action of the timestamp (i, t)
6) The estimated correlation between the stock i and the market factor at time t, $\rho_{i,t}$
7) The estimated correlation between the portfolio and the market factor at time t, ρ_t

2) Action Space: Given that at each RL time-step the agent has to deal with only one stock, the action that it may take is a real number, which relates with the weight that will be added to the current stock at the current point in time. The action values range from -1 to 1. More details on how the action will relate with the stock's allocation are described on section III-D4

3) Reward Space: The reward is a real number that relates with the profit or loss and with the current portfolio estimated volatility. Thus, at the RL time-step (i, t), the reward $reward_{i,t}$ is calculated as

$$reward_{i,t} = \alpha * w_{i,t} * r_{i,t+1} + \gamma * max(0, \sigma_t - volThreshold)$$

where α and γ are normalization parameters and $volThreshold$ is the maximum desired volatility of the portfolio. The three parameters of the reward function will not be subjected to a tuning process.

There are two things very important to note in the reward function. First, at each RL time-step, the profit part of the reward is dependent only on the performance of the specific stock and allocation performed at timestamp (i,t); and second, the volatility penalization is dependent on the allocations on all the stocks, including the stock being nogotiated at timestamp (i,t). That means that this penalization would be completely explained by both the RL state and the agent's action of time-step (i,t).

4) Transition Dynamics: Each sample of the state space is indexed by a specific stock at a specific point in time. Thus, while the temporal tracked time-step is designated with the index t, the RL time-step is designated with the index (i,t).

The RL time increases following the temporal sequence, being that at each temporal time-step, the RL environment sweeps all the stocks available to negotiate, one at the time. Thus, each stock gives a sample state, and requires a sample action. The sequence in which the stocks are passed is defined by the allocated weight $w_{i,t}^-$ (here the superscript "-" is used to indicate that the variable was not updated by the agent's action yet at time t for the i-th stock). At the beginning of each time-step t, the stocks are ordered with decreasing absolute weight and are swept by that order. This sorting was made so that the agent could reduce its larger positions at the first steps of the day if it believes it is necessary.

The action taken by the agent at time-step (i,t) will be first normalized (multiplied by 0.1)[2] and then added to the allocated stock weight. Besides that, the updated weight will be bounded by the variable $availableWeight_{i,t}$. The bounding function works as below.

- The $availableWeight$ starts as 1 for the first day and first negotiated stock.
- At each new allocation, the available weight is reduced by the absolute value of the allocation (agent action multiplied by 0.1) if it increases the absolute weight on the stock being negotiated.
- The sum of all absolute values of the weights and the $availableWeight$ is always equal to zero. Note that $availableWeight$ may sometimes be negative due to large price movements.
- At the beginning of each day, the weights allocated in each asset are updated by the assets corresponding price movements. The $availableWeight$ is then updated so the previous item remains valid (overall sum remains zero). This may cause the $availableWeight$ to be lower than zero.
- If the $availableWeight$ is already lower or equal to zero, any action that would increase a stock's weight in absolute value will generate an allocation of zero. In this case, an allocation would take place only if it has an opposite signal relative to the current weight in the traded stock.
- If the $availableWeight$ is higher the zero, the absolute value of the action will be bounded so that the $availableWeight$ doesn't become negative.

[2] The choice for the normalization of the action was made because of the issue of sparse rewards, see [13]. During the training, if the agent is allowed to make really big allocations at once, it will almost all the time be penalized by the volatility and allocated in two or three stocks, what can really slow down the training process.

Whenever there is a new RL time-step the agent deals with a new asset, thus, the state variables 2, 3, 5 and 6 (subsection III-D1), that are stock specific variables, change accordingly with the new stock. The portfolio variables σ_t, ρ_t and $availableWeight_{i,t}$ will change due to the previous RL action. But it is important to remark that if the new RL time-step also represents a new day, then the portfolio variables σ_t, ρ_t and $availableWeight_{i,t}$ will be updated. This will happen because when a new day arrive, the estimate of each stock volatility and correlation will change, causing changes in the portfolio variables as well, also $w_{i,t+1}^- = w_{i,t} * (1 + r_{i,t+1})$. Besides that, when there is a transition of day, the stocks are reordered to be swept by decreasing absolute weight.

It is an important feature of this framework that the RL timestep is incremented by stock. This feature, in union with the risk decomposition, has an interesting property. Whenever a new stock/allocation arrives, it is not necessary to use equation 2 on the updated set of stocks in the portfolio. Instead of that, the variance change in the portfolio can be calculated in constant time, using variables of the portfolio before the new allocation and variables of the new stock. The variables used to update the portfolio variance and market correlation are: The variance and correlation of the portfolio (before the new allocation), the variance and correlation of the negotiated stock and finally the allocation itself. Note that whenever we say "correlation" we mean the correlation between a stock or the portfolio with the market factor. The portfolio variables updates are described in equations 4 and 5, which can be demonstrated supposing the residual returns have zero correlation and a bit of algebra.

$$\Delta\sigma^2 = 2 * \Delta w_i \rho_i \sigma_i (\rho^- \sigma^- - w_i^- \rho_i \sigma_i) + (\Delta w_i^2 + 2\Delta w_i w_i^-)\sigma_i^2 \tag{4}$$

$$\rho_P = \rho_P^- \sigma_P^- / \sigma_P + \rho_i \sigma_i (w_i - w_i^-)/\sigma_P \tag{5}$$

5) PPO: Proximal Policy Optimization (PPO) is a policy gradient method to train deep reinforcement learning models. There are two neural networks composing a PPO model, the actor network and the critic. The actor network computes the policy, while the critic computes estimates of the value function. This method introduces the clippage of the training objective function, which prevents the policy update steps too large, using the specialized KL divergence metric. It is an on-policy algorithm commonly used in continuous action and state spaces. Recently it has shown stable results, so we chose it. For more information about this method refer to [14]. As for the software, we used the stable-baselines 3 implementation of PPO, see [15]. We did not perform any hyperparameter tuning on the actor-critic networks. The discounting rate was set to one. Besides that, the RL hyperparameters were set as its default values.

TABLE I
SELECTED STOCKS

Stock	Market Capitalization (bi USD)
AAPL	583
GOOG	528
MSFT	443
BRKb	325
XOM	324
AMZN	316
GE	314
FB	295
JNJ	284

E. Hypotheses

We assumed some hypotheses to preserve the simplicity of analysis. Such hypotheses are described next.

1) There are no costs of transaction;
2) There are no slippages or market impact of the orders;
3) The orders are executed with the desired amount of money at the closing time, thus being possible to trade fractions of stocks.

We must say that although hypothesis 3, the order is created with information strictly prior to the closing time. About hypothesis 1, although the transaction costs are a simple case of reward engineering for RL, we chose to not use any. This choice was made to preserve the simplicity of the analysis.

IV. EXPERIMENTS AND RESULTS

We performed experiments with the top 10 stocks based on their market capitalization of the end of 2015. The data consists of daily data for US listed stocks from 2004 to 2022. The market capitalization selection criteria was chosen to assure the liquidity of the instruments. The stocks and its market capitalization values are displayed on table I. Besides the stocks data, we use the SPY index as a proxy for the market factor.

Data is divided into tree sets, whose we will call A, B and C. Set A ranges from 2004-01-01 to 2009-12-31, set B from 2010-01-01 to 2015-12-31 and set C from 2016-01-01 to 2022-06-13. The LSTM will be trained on set A and evaluated on sets B and C. The reinforcement learning agent is trained on set B and evaluated on the set C. Thus, the set C is our final out-of-sample data for evaluations.

Two benchmark strategies are evaluated:

- EW. Equal weight for all stocks
- MV. Minimum variance, but daily controlling for the ex-ante portfolio volatility remain in 8% annual. The minimum variance allocation is the literature default, see [16], however using the covariances estimates as in equation 1

Two deep reinforcement learning strategies were evaluated:

- DRL-1. Deep reinforcement learning without volatility penalization. Parameters $\alpha = 100$ and $\gamma = 0$. The parameter α is set to 100 so the reward is numerically the profits (or losses) in percentage.

TABLE II
OBSERVED PERFORMANCES

Strategy	Annual Return	Annual Volatility	Sharpe Ratio
DRL-2	4.83%	6.50%	0.74
DRL-1	29.39%	38.07%	0.77
EW	17.88%	21.20%	0.84
MV	8.87%	8.95%	0.99

- DRL-2. Deep reinforcement learning with volatility penalization. Parameters $\alpha = 100$, $\gamma = -0.1$ and $volThreshold = 0.5\%$. The daily volatility of 0.5% corresponds to an annual volatility of 7.94%. The parameter γ is set to -0.1 so that the profits reward and the risk penalization remain in the same scale.

There are two main comparisons that we aim to do. First we must check if the covariances estimates are aligned with their realizations. To do that, we should check if the realized variance of the MV benchmark is too distant from 8% annual. Second, we must compare the RL agents performance with the benchmarks.

We use as main performance measures the annual return of the strategy, the annual volatility and the annualized sharpe ratio. For the former metric, we consider the risk-free interest rate as zero.

The results for the five strategies are presented on table II.

The minimum variance benchmark presents one of the best performances, besides the fact that the estimated volatility is close to the realized volatility (target of 8% annual), which is an indicative that the method of estimation of the covariances that we adopt may be a good one. The fact that the realized volatility is actually higher than 8% indicates that the market residuals returns between the stocks actually present a positive bias. That makes sense, once there are specific "per sector" price movements that may not be fully captured by the market factor.

The DRL-1 presents the highest cumulative return and DRL-2 presents the lowest volatility among all the strategies. The figure 1 shows that the DRL-1 is able to get higher rates of return, however with a perceptively higher level of volatility. As we can see in figure 2, the sum of the weights of the strategy DRL-1 goes significantly above one. Since from the environment dynamics it is not possible for the DRL algorithm to "force" weights summing more than one, we can conclude that the DRL-1 learns to acquire long positions at the beginning of the trading period and then to stay passive. Although a strategy like that really worked for the large US equities, it is not our intention either that the strategy becomes passive nor that the strategy gets more risk over time because it is holding positions for a really long time.

The figure 3 shows how the minimum variance strategy with the volatility controlled in daily basis outperforms the DRL-2 strategy designed to have low variance. However, from figure 2 it is possible to realize a really interesting property of the DRL-2 strategy: The summation of the portfolio weights oscillates between positive and negative values, what makes this strategy

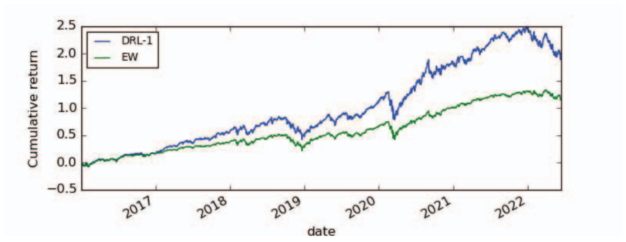

Fig. 1. Cumulative returns of two strategies. Deep reinforcement learning without volatility penalization (DRL-1) vs equal weight (EW).

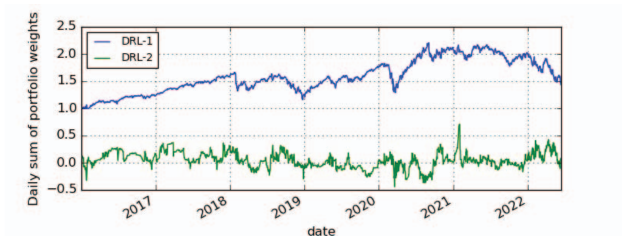

Fig. 2. Daily summation of the portfolio weights. Comparing the behavior of two deep reinforcement learning strategies.

really different from the other 3. This property has two major consequences: First the DRL-2 algorithm learned that a strong positive (or negative) bias in the portfolio weights would sum up the systematic risk; second, the performance of the DRL-2 does not completely rely on the US market performance, instead it learned to be profitable without a strong systematic risk, what approximates this strategy of a *pure alpha* one, see [17].

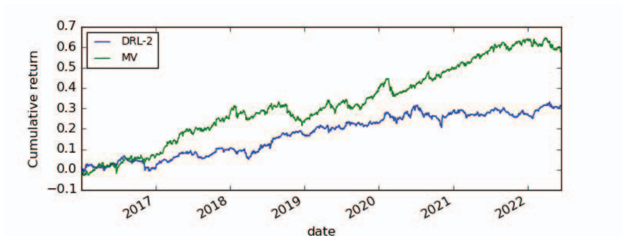

Fig. 3. Cumulative returns of two strategies. Deep reinforcement learning with volatility penalization vs minimum variance strategy.

V. Conclusions and Future Works

The purpose of this paper is to present a framework to use reinforcement learning to perform asset allocation, controlling the volatility, aiming to maximize the profits, using a low dimensional approach. Although the framework doesn't present impressive out-of-sample results, the strategies are able to control the portfolio volatility, getting away from systematic risk and still being profitable. Besides that, the minimum variance test shows that the realized volatility does not change too much from the target volatility using our low dimensional estimates of the covariance matrix. The strategy without volatility control acquires a strong positive bias in a way that its profits are the highest, however the risk increases disproportional.

We believe we show that it is possible to adopt a low dimensional approach to the portfolio allocation problem. For future works, we aim to develop better predictors with or without SL, we aim to use a larger universe of stocks, such as the complete SP 500 index at each point in time and finally we aim to explore more the RL algorithms and its hyperparameters.

References

[1] P. A. L. Castro, Annoni Junior, R. Schiman, and Jaime Simão, “Autonomous Investment Analysis: A Discrete Probabilistic Approach,” *Revista de Informática Teorica e Aplicada-RITA-ISSN*, vol. 2175, pp. 2745.

[2] H. M. Markowitz, “Portfolio Selection,” *The Journal of Finance*, vol. 7, no. 1, year 1952.

[3] L. Conegundes and A. C. Machado Pereira, “Beating the Stock Market with a Deep Reinforcement Learning Day Trading System,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8, 2020, doi: 10.1109/IJCNN48605.2020.9206938.

[4] R. A. de Oliveira, H. S. Ramos, D. H. Dalip, and A. C. Machado Pereira, “A Tabular SARSA-based Stock Market Agent,” in *Proceedings of the First ACM International Conference on AI in Finance*, pp. 1-8, 2020.

[5] Z. Jiang, D. Xu, and J. Liang, “A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem,” *arXiv preprint arXiv:1706.10059*, 2017.

[6] B. Kelly, S. Gu, and D. Xiu, “Empirical Asset Pricing via Machine Learning,” *The Review of Financial Studies*, vol. 33, no. 5, pp. 2223-2273, 2020.

[7] L. Chen, M. Pelger, and J. Zhu, “Deep Learning in Asset Pricing,” *arXiv preprint arXiv:1904.00745*, 2019.

[8] Bhansali, Vineer, Josh Davis, Graham Rennison, Jason Hsu, and Feifei Li. ”The risk in risk parity: A factor-based analysis of asset-based risk parity.” *The Journal of Investing* 21, no. 3 (2012): 102–110.

[9] Roncalli, T. and Weisang, G., “Risk parity portfolios with risk factors,” Quantitative Finance, vol. 16, no. 3, pp. 377-388, 2016.

[10] L.K.C. Chan, J. Karceski, and J. Lakonishok, “On portfolio optimization: Forecasting covariances and choosing the risk model,” The Review of Financial Studies, vol. 12, no. 5, pp. 937–974, 1999.

[11] Chen, Kai and Zhou, Yi and Dai, Fangyan, “A LSTM-based method for stock returns prediction: A case study of China stock market,” in *2015 IEEE International Conference on Big Data (Big Data)*, pp. 2823-2824, 2015.

[12] Chollet, Francois and others, ”Keras”, 2015

[13] Hare, Joshua. ”Dealing with sparse rewards in reinforcement learning.” arXiv preprint arXiv:1910.09281 (2019).

[14] Schulman, John and Wolski, Filip and Dhariwal, Prafulla and Radford, Alec and Klimov, Oleg. ”Proximal policy optimization algorithms.” *arXiv preprint arXiv:1707.06347* (2017).

[15] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. ”Stable-Baselines3: Reliable Reinforcement Learning Implementations.” Journal of Machine Learning Research 22, no. 268 (2021): 1-8. http://jmlr.org/papers/v22/20-1364.html.

[16] Chaves, Denis and Hsu, Jason and Li, Feifei and Shakernia, Omid. ”Risk parity portfolio vs. other asset allocation heuristic portfolios”. *The Journal of Investing*, vol. 20, no. 1, pp. 108–118, 2011, Institutional Investor Journals Umbrella.

[17] R. Grinold and R. Kahn, *Active Portfolio Management A Quantitative Approach for Providing Superior Returns and Controlling Risk*, McGraw-Hill Companies, 2000. (Refer to the forecasting sections.)

Appendix

This appendix is dedicated to demonstrate how to calculate the variation in the portfolio volatility when there is a change in the weight applied in only one of the assets in the portfolio.

Consider there is a portfolio defined by the weights w of a universe of stocks. Consider also that you are willing to change the weight of a specific stock i, applying an allocation Δw_i. For that stock, the weight before the new allocation will be designated as w_i^-, and after the allocation $w_i = w_i^- + \Delta w_i$. For any other stock j, with $j \neq i$, the weight won't change and is designated as w_j. There are no constraints about w other than being a finite value. The other variables will be defined as follows.

- σ_P^- the volatility of the portfolio before the new allocation.
- σ_P the volatility of the portfolio after the new allocation.
- ρ_P^- the portfolio correlation with the market factor before the allocation.
- ρ_P the portfolio correlation with the market factor after the allocation.
- σ_j the volatility of the stock j.
- ρ_j the correlation between the market factor and the stock j.

A. Correlation of the portfolio formula

- β_P is the beta of the portfolio with relation to the market factor.
- β_j is the beta of the stock j with relation to the market factor.
- σ_{mkt} the volatility of the market factor

Theorem 1 (will not prove here):

$$\beta_P = \sum_j w_j \beta_j$$

Theorem 2 (will not prove here):

$$\beta = \rho\sigma/\sigma_{mkt}$$

Substituting the beta formula from theorem 1 into the theorem 2, we have:

$$\rho_P \sigma_P/\sigma_{mkt} = \sum_j w_j \rho_j \sigma_j/\sigma_{mkt}$$

and thus:

$$\rho_P = \sum_j w_j \rho_j \sigma_j/\sigma_P \tag{6}$$

B. Covariance Formula

Theorem 3 (see [10]):

Supposing that the correlation between the idiosyncratic returns of two given assets is zero, we have:

$$cov_{j,k} = \beta_j \beta_k \sigma_{mkt}^2$$

Conclusions:

With theorems 2 and 3, we have

$$cov_{j,k} = \rho_j \sigma_j \rho_k \sigma_k \tag{7}$$

C. Variance Change

Defining $cov_{j,k}$ as in equation 7 for $j \neq k$ and $cov_{j,j} = \sigma_j^2$ we have:

$$\sigma_p^2 = \sum_j \sum_k w_j w_k cov_{j,k}$$

Decomposing the equation above:

$$\sigma_P^2 = 2\sum_{j\neq i} w_i w_j cov_{i,j} + w_i^2\sigma_i^2 + \sum_{j\neq i}\sum_{k\neq i,j} w_j w_k cov_{j,k} + \sum_{j\neq i} w_j^2 \sigma_j^2$$

The decomposition above is important because the first two parts of the sum are dependent on the stock i and the last two are not dependent. We can use that to compute the variation in the portfolio variance $\Delta\sigma_P^2 = \sigma_P^2 - (\sigma_P^-)^2$ due to the allocation Δw_i in stock i:

$$\Delta\sigma_P^2 = (w_i - w_i^-)2\sum_{j\neq i} w_j cov_{i,j} + (w_i^2 - (w_i^-)^2)\sigma_i^2 \tag{8}$$

Substituting the covariance estimate from equation 7, we have

$$\Delta\sigma_P^2 = \Delta w_i 2\sum_{j\neq i} w_j \rho_i \sigma_i \rho_j \sigma_j + (\Delta w_i^2 + 2\Delta w_i w_i^-)\sigma_i^2$$

The equation 6 may be rewritten as:

$$\rho_P\sigma_P - w_i\rho_i\sigma_i = \sum_{j\neq i} w_j \rho_j \sigma_j$$

Substituting in the equation 8 we have:

$$\Delta\sigma_P^2 = \Delta w_i 2\rho_i\sigma_i(\rho_P^-\sigma_P^- - w_i^-\rho_i\sigma_i) + (\Delta w_i^2 + 2\Delta w_i w_i^-)\sigma_i^2 \tag{9}$$

D. Portfolio Correlation Change

In the case of an allocation Δw_i, From eq 6, we have:

$$\rho_P\sigma_P - w_i\rho_i\sigma_i = \sum_{j\neq i} w_j\rho_j\sigma_j = \rho_P^-\sigma_P^- - w_i^-\rho_i\sigma_i$$

And thus:

$$\rho_P = \rho_P^-\sigma_P^-/\sigma_P + \rho_i\sigma_i(w_i - w_i^-)/\sigma_P \tag{10}$$

* We must know that when σ_P is equal to zero, the equation 10 is not well defined, but in this case ρ_P will be set to zero.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

This work was supported by the grant #2022/01524-2, São Paulo Research Foundation (FAPESP)