Robust Online Portfolio Optimisation

Lukas Jarasunas * **Hrithik Nambiar** *

Abstract

Portfolio optimization is a well-studied optimisation problem in both machine learning and computational finance. The task is to optimally allocate wealth across various assets such that the returns are maximized. In a world where market prices are dynamic and can often be considered adversarial, the assumptions of statistical machine learning fail. Online learning provides a framework for studying such optimization problems. In this work, we compare and contrast different state-of-the-art online learning algorithms using a real-world dataset of different exchange-traded funds. We observe the superior performance of certain classes of algorithms over others during certain market trends. To take advantage of this discrepancy in the performance of different base experts, we propose meta-learning as a solution to combine these base experts. In this work, we present different Nth-degree meta-learners and a particle-swarm-based meta-learner to assign the optimal weights to the decisions of experts. We present both a regret and run-time analysis for these methods. Experimental results on real-world stocks are also presented.

1. Introduction

1.1. Background and Motivation

Portfolio optimization seeks to determine how to allocate wealth to a set of assets to maximize returns and minimize risk. Traditional approaches rely on assumptions of stationary market conditions or known return distributions. However, in real-world financial markets, these assumptions often break down due to the adversarial and dynamic nature of assets' price movements. This necessitates a more robust approach to allocating wealth.

Online learning offers a robust, natural framework for tackling adversarial and dynamic problems. Online learning algorithms make decisions on already observed data to minimize regret compared to the best possible strategy in hindsight. This sequential decision-making framework has been adopted for portfolio optimization with great success: rather than requiring and optimizing over the entire data set up front, we can make more robust strategies without using future data or any assumptions on the distribution of returns. Online learning has created powerful portfolio optimization strategies that tend to be more adaptable or robust than traditional strategies.

However, the numerous online portfolio optimization strategies [OLPSs] perform significantly differently under different market conditions. For example, Follow The Winner [FTW] strategies, where wealth is allocated to assets that have historically performed the best, tend to perform better in trending markets, while Follow The Loser [FTL] strategies, where wealth is allocated according to an asset's price difference from its mean, tend to perform better in sideways moving markets. An optimal strategy would be robust yet highly profitable, no matter what the general market conditions are.

Meta-learning algorithms involve combining the decisions of experts to assign weights to predictions. In portfolio optimization, meta-learners may assign weights to portfolio optimization strategies such that the overall strategy is robust yet highly profitable no matter the general market conditions. Further, a meta-learner of OLPSs is online, keeping the robust and adaptable characteristics of online algorithms.

1.2. Major Contributions

In this work, we present two novel meta-learning algorithms, namely the Nth-degree meta-learners and Particle Swarm Meta-Optimizer. Our algorithms intelligently, adaptively assign weights to online portfolio strategies, essentially assigning weights to assets. We show that this creates more robust portfolios with a sub-linear regret guarantee. We also show the runtime complexity of these algorithms, propose hypotheses, and rigorously test these hypotheses using a train test split.

1.3. Paper Organization

In Section 2, we present an extensive literature review of OLPSs. Section 3 outlines the experimental setup for testing our proposed algorithms. Section 4 presents our algorithms with their definition, motivation, online nature, runtime, hypothesis, results, and analysis. Section 5 finally concludes

the paper and outlines future potential directions of research.

1.4. Goal

Through this paper, we aim to bridge the gap between theoretical advances in OLPSs and their practical application in robust portfolio optimisation by providing scalable and adaptable frameworks for financial markets in the real world.

2. Related Works

We present an extensive literature review on the state-of-the-art algorithms for online portfolio optimization.

2.1. Online Learning

To introduce the concept of online learning, consider a game played over T rounds. In each round t, an adversary secretly chooses a real number $Y_t \in [0, 1]$. You then guess a number $X_t \in [0, 1]$. After your guess, the adversary reveals Y_t, and you incur the loss $\ell_t = (X_t - Y_t)^2$. How can we minimize the cumulative loss over all rounds? Even though Y_t is chosen adversarially and revealed only after X_t is guessed, we can sequentially provide better predictions by adapting to feedback.

We may solve problems like these through regret minimization, which is detailed in the next section. (Wald & Wolfowitz, 1950) first introduced the concept of maximizing utility in statistical decision problems based on zero-sum two-person games. (Savage, 1951) then reframed this problem as minimizing loss. (J.Milnor, 1951) then recharacterized this loss as regret. (Hannan, 1957) focused on minimizing regret in zero-sum repeated games. These papers all built the foundation for online learning, a field that has since been developed and applied to a variety of domains.

2.2. Regret

In statistical machine learning, where the assumption is that the data is sampled from a distribution, the goal is often loss or empirical risk minimization. However, in the case of online learning, where there is no assumption regarding the dataset distribution, the goal is often to minimize the regret against any competitor u. For the game of guessing real numbers in $[0, 1]$, Regret is defined against any competitor $u \in [0, 1]$ as,

$$R_T(\boldsymbol{u}) := \sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{u})$$

In other words, the regret compares the cumulative loss of the player to the cumulative loss of the best action in hindsight. Online learning algorithms aim to have a sub-linear regret. In previous works, this is usually done by making assumptions of strong convexity and smoothness on the loss functions. Having a sub-linear regret guarantees that its performance on average will approach the fixed best strategies as the number of rounds increases.

Consider the game mentioned above. If one assumes that the numbers are from a fixed distribution, they can guess the mean μ in each round and incur a loss of $\sigma^2 T$ in T rounds on average. In such a game, the goal of our online algorithm would be to minimize the regret,

$$\mathbb{E}_Y \left[\sum_{t=1}^{T} (x_t - Y)^2 \right] - \sigma^2 T,$$

2.3. Standard Portfolio Optimisation Methods

Before touching OLPSs, we must first be accustomed to the standard approach of portfolio optimization. Traditional methods focus on allocating wealth across assets to maximize wealth or similar performance measures. Now we introduce the notations used in this work. We assume that we are allocating wealth across $d \geq 2$ assets. Further, let the market gains vector at time t be $\mathbf{w_t} = (w_{t,1}, w_{t,2}, \ldots, w_{t,d})$, where $w_{t,i}$ denotes the gain of asset i in the time period t. A portfolio strategy can be defined as $\mathbf{x}_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,d})$, where $x_{t,i}$ represents the proportion of wealth invested in asset i at time t. The strategy must satisfy $0 \leq x_{t,i} \leq 1$ and $\|\mathbf{x}_t\|_1 = 1$, ensuring that all wealth is constantly invested. Note that one may include the U.S. dollar (or whichever native currency) as an asset, allowing the portfolio strategy to not have all of its wealth invested at all periods. Over T time periods, the objective function is typically maximizing the logarithm of wealth $log(W_T) = log(\prod_{t=1}^{T} w_t \cdot x_t) = \sum_{t=1}^{T} log(w_t \cdot x_t)$ or the Sharpe ratio $\frac{\mathbb{E}[\mathrm{W}_T(\mathbf{x}) - \mathrm{r}]}{\mathrm{STD}[\mathrm{W}_T(\mathbf{x})]}$, where r is the risk-free return.

As an example, the most naive portfolio strategy is buy and hold, where one allocates an initial portfolio and doesn't change it overall T. Another class of strategies is the constant rebalanced portfolio, where one maintains fixed portions of wealth in each asset by rebalancing their portfolio after each timestamp $t \in [0, T]$.

Typically, more complex portfolio strategies will divide available data into train, validation, and test sets. They will then build a model using the training dataset, tune hyperparameters using the validation dataset, and find the profitability using the test dataset. Online portfolio optimization strategies make certain restrictions that can create more robust strategies.

(Kelly, 1956) first proposed maximizing $log(W)$. He assumed market gains are i.i.d. from a known distribution to create a distributional approach to betting and gambling.

2.4. Online Portfolio Optimisation

Unlike standard approaches, an online portfolio optimisation strategy can only utilize information available up to time $t-1$. Say one has data from 2020 until now of certain assets. This means that they should design a strategy such that if run on Jan 1st, 2021, it could only use data from the year 2020. This is fundamentally different from standard methods which tend to use all data points within a train data set. Additionally, no assumptions can be made about the distribution of market returns; this further reinforces the robustness of these methods.

The primary objective of these OLPSs is to minimize the regret in wealth of one's strategy against the best constant rebalanced portfolio [BCRP]—namely, the constant reblanced portfolio with the highest wealth return from the entire set of constant rebalanced portfolios in hindsight: $\mathrm{R}_T(x) = \ln(\mathrm{W}_T(u)) - \ln(\mathrm{W}_T(x))$, where u is the best constant rebalanced portfolio. There are several reasons that constantly rebalanced portfolios are the set of comparators. Firstly, consider a market with market vectors of $(1, 1/2), (1, 2), (1, 1/2), (1, 2), \ldots$. The BCRP here is $(0.5, 0.5)$ which is able to extract $\mathrm{W}_T(u) = (9/8)^{T/2}$, despite the market effectively moving sideways. Also, if the market vectors are i.i.d., the BCRP is asymptotically optimal in hindsight.

Further, the goal of online strategies is to achieve sublinear regret. As a tangible example, exponential gradient [EG], an online strategy covered next, has its wealth graphed against BCRP over an arbitrary set of assets in Figure 1. BCRP outperforms EG in wealth at least. Further, EG's regret over that same period is plotted in Figure 2. Regret is a very theoretical quantity, and current literature mainly proves that the upper bound of a strategy's regret is sublinear. As shown in the plot, these bounds are often quite loose in practice.

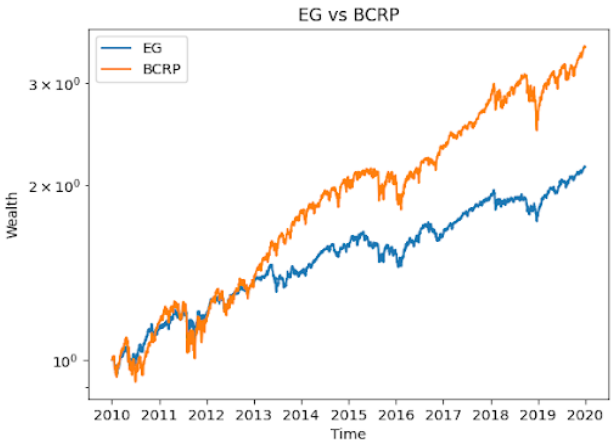

Figure 1. Comparison of wealth generated using EG to BCRP. An example showing BCRP makes a great comparator for regret.

2.5. Benchmarks/Algorithms

This section will cover a variety of popular portfolio optimization algorithms. We classify these benchmark algorithms into three main classes based on the strategy they follow: Follow-The-Winner [FTW], Follow-The-Loser [FTL], and Meta-Learning Algorithms.

(Cover & Ordentlich, 1996) proposed the first portfolio optimization algorithm which would be considered online. It contained minimax regret and no assumptions over market gains. Future papers focused on specific OLPSs and proving sublinear regret or similar properties of these algorithms.

2.5.1. Follow The Winner

FTW strategies tend to allocate wealth to assets that have performed well in the past. Looking at online strategies, the simplest method is the Follow-the-Leader [FTL1] strategy, which selects the portfolio at time t to maximize cumulative past log-wealth: $x_t = \arg\max_x \sum_{i=1}^{t-1} \ln(\mathrm{W}_i(x))$. FTL1 can lead to unstable allocations even though it is conceptually straightforward. Follow-The-Regularized-Leader [FTRL] aims to solve this by adding a regularization term $r(\mathbf{x})$ to encourage diversification: $x_t = \arg\max_x \left[r(x) + \eta_t \sum_{i=1}^{t-1} \ln(\mathrm{W}_i(x)) \right]$, where η_t is a learning rate to control the balance between regularization and historical performance. A specific example of FTRL is the Exponential Gradient Algorithm [EG](Helmbold et al., 1998) which uses Shannon's entropy as the regularization term: $r(\mathbf{x}_t) = -\sum_{j=1}^{d} x_{t,j} \ln(x_{t,j})$. Shannon's entropy is highest with an equal allocation of wealth between assets, making a perfect regularization term. Similar strategies allocate wealth to the best-performing assets, making FTW strategies best in trending markets where the best-performing assets stay as the best-performing assets.

2.5.2. Follow-The-Loser

Unlike FTW strategies, FTL strategies transfer wealth from the winners to the losers. The main idea behind this strategy is that of mean reversion (Poterba & Summers, 1988), which suggests that the good stocks which are poorly performing at a certain period would perform well in the following periods. Hence, this strategy suggests to transfer wealth to the "losers". (Borodin et al., 2003) proposed an algorithm called Anticor, which calculates the correlations of assets across a time window, and finds the anti-correlated assets for transferring wealth to grab the advantage of the potential mean reversal behaviour of the market. (Li et al., 2012) presents a Passive-Aggressive-Mean-Reversion [PAMR] algorithm to adjust portfolio weights iteratively while minimizing a hinge

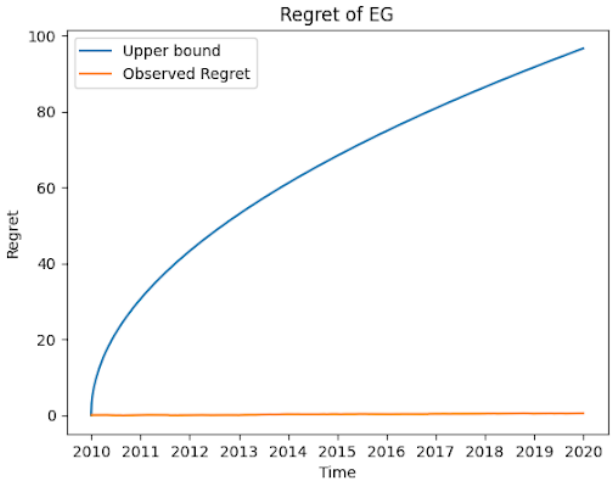

Figure 2. Comparison of the computed regret of EG against the theoretical upper bound.

loss which ensures a minimal deviation from the current portfolio while also adhering to the mean-reversal principle. (Li et al., 2013) proposed Confidence-Weighted-Mean-Reversal [CWMR] which modeled the portfolio vector as a Gaussian, which helps incorporate an uncertainty signal in the mean reversion signal. This helps performance by allowing for cautious updates during volatile periods and aggressive updates in stable markets. Online Moving Average Reversion [OLMAR] (Li & Hoi, 2012) maintained a moving average of the relative prices of assets, to model multi-period mean reversion, unlike PAMR and CWMR which implicitly predicts prices at $t+1$ as that of $t-1$. Robust Median Reversion [RMR] (Huang et al., 2016) presented a more robust algorithm to exploit mean reversion via a robust L1-median estimator. Similar strategies allocate wealth to the worst performing assets, making FTL strategies best in sideways moving markets where assets fluctuate around their means.

2.5.3. Online Meta-Learning

Meta-learning in the context of online learning is also referred to as expert learning. The idea in this learning paradigm is to learn to assign weights to the decisions of a pool of base expert algorithms. These weights are updated in an online fashion to accommodate the dynamic nature of the problem at hand. The advantages of meta-algorithms are attributed to the fact that one does not know which algorithm is the optimal expert before the decision is made (Das & Banerjee, 2011). When both heuristics and universal strategies are present in the pool of base experts, the meta-learning algorithm conserves the universality property. (Vovk & Watkins, 1998) developed an Aggregating Algorithm which suggested iteratively updating P_t the weights assigned to each base expert at time t as $P_{t+1}(A) = \int_A \beta^{\ell(\mathbf{x}_t, \gamma_t(\theta))} P_t(d\theta)$, where $\beta^{\ell(\mathbf{x}_t, \gamma_t(\theta))}$ represents the loss incurred by assigning weight P_t at the previous time step. The most influential work in the area was (Das & Banerjee, 2011), which proposed two meta-learning algorithms, namely, Online Gradient Update (OGU) and Online Newton Update (ONU). OGU updates the weights P_t assigned to each of the k base expert as $P_{t+1}(h) = P_t(h) \exp\left(-\eta \ell_t(h)\right) / Z_t$ where $\eta > 0$ and Z_t is the partition function. Similarly, ONU updates the weights as, $w_{t+1,h} = \prod_{\Delta_k}^{A_t} \left(w_t - \frac{2}{\beta} A_t^{-1} \nabla f_t \right)$. Here, A_t is defined as $A_t = \sum_{\tau=1}^{t} \nabla f_t \nabla f_t^T + \epsilon \mathbb{I}$. Theoretically, OGU and ONU can achieve the growth rate as the optimal convex combination of the underlying experts. Most recently, (Zhang et al., 2024) has proposed combing the OGU and ONU meta-learning algorithms with a peak price prediction model.

3. Experimental Setup

3.1. Dataset

Our proposed algorithms will be tested on the longest-running ETFs across six distinct asset classes: VTI, EFA, EEM, TLT, TIP, and VNQ. These ETFs ensure board market representation and diversification while also avoiding survivor-ship bias. The daily closing prices of these ETFs from 2010-2024 are easily accessible through Yahoo Finance.

Further, the returns of OLPSs are part of the dataset, as our proposed meta-learners will be based on these returns. We use several types of base strategies: Follow-The-Winner (Universal Portfolio [UP], EG, ONS), Follow-The-Loser (PAMR, OLMAR, RMR, CWMR), and correlation-based (Correlation-driven nonparametric learning approach [CORN]). To verify the claim that certain portfolio strategies perform worse than others in given periods, each OLPS has been the worst-performing strategy in at least one month of the available dataset in figure 3.

3.2. Testing Framework

Even though our proposed algorithms are online, we need a testing framework. Namely, we need a train set for hyperparameter selection and a test set to estimate actual returns. The strategies will be trained on closing prices from 2010–2019. Hyperparameters will be chosen according to performance in this time period. Finally, strategies with optimal hyperparameters will be tested on data from 2020 to the present, ensuring the methods generalize to unseen market conditions. Although our algorithms are online in nature, we use this test dataset to test the robustness of our algorithms without any intervention.

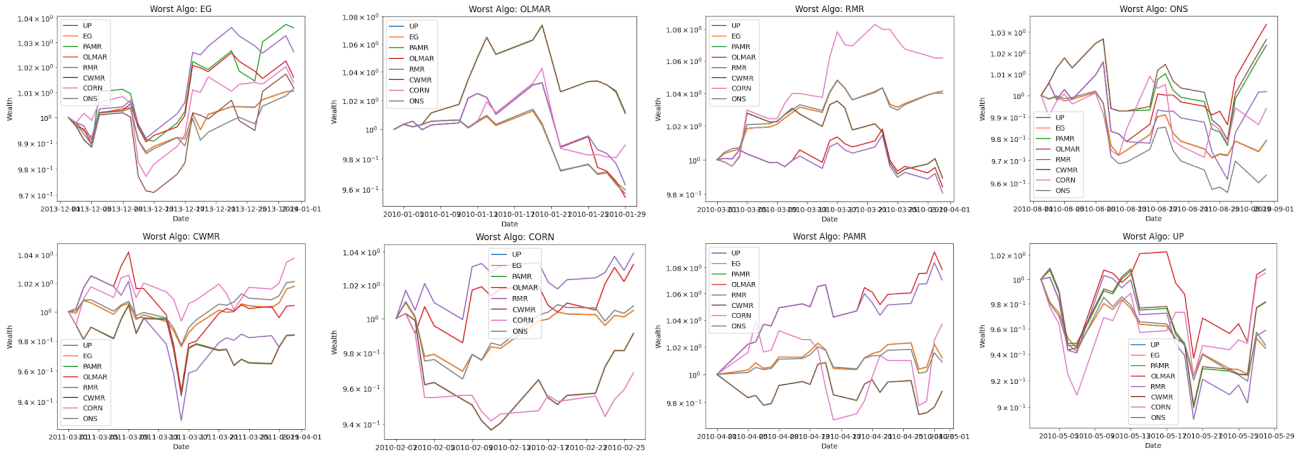

Figure 3. This figure shows that each OLPS performs the worst in atleast one month. This motivates us to develop meta-learning algorithms.

4. Proposed Algorithms

Given this background, we may propose and test novel meta-learning online portfolio optimisation algorithms and provide theoretical guarantees for them.

4.1. Nth-Degree Meta-Learners

Firstly, define a 0th-degree strategy as directly choosing the weights of the assets. In this context, these strategies are UP, EG, ONS, etc. For example, a 0th-degree UP strategy is simply UP. Then, a 1st-degree strategy is a meta-learner that selects the weights of multiple 0th-degree strategies. For example, a 1st degree EG strategy would apply EG to the UP, EG, ONS, etc ... strategies. Similarly, a 2nd-degree strategy allocates weights to 1st-degree strategies. For example, a 2nd-degree EG strategy would apply EG to the 1st-degree UP, 1st-degree EG, 1st-degree ONS, etc... strategies. This hierarchy extends to an Nth-degree strategy, which chooses weights for $N-1$th-degree strategies. The intuition is that meta-learners [of the first degree] generally perform better than individual assets. We will analyze exactly how these strategies scale at increasing degrees of meta-learning.

4.1.1. Sublinear Regret

First, though, we prove that these strategies have sub-linear regret:

Proof. Let S be a set of OLPSs, where $R_T(s) < O(T)\ \forall s \in S$ [i.e. sublinear regret], $|S| = n$. Finally, denote $S_i^{(d)}$ as the dth-degree strategy of S_i.
We want to show that $R_T(s_i^{(d)}) < O(T) \forall i \in [1, N], d \in \mathbb{Z}^+$, or that every OLPS strategy has a sublinear regret at every possible degree.

We may use induction:
Base case (d = 0):
$R_T(s_i^{(0)}) = R_T(s_i) < O(T)$ by definition.
Inductive case:
Assume $R_T(s_i^{(d)}) < O(T)$.
Let w_i denote the weight assigned to each d-1 degree strategy by $s_i^{(d)}$, where $\sum_{i=1}^{N} w_i = 1$.Then,
$R_T(s_i^{(d+1)}) = ln(W_T(c)) - ln(W_T(s_i^{(d+1)}))$
$= ln(\sum_{i=1}^{n} w_i W_T(c)) - ln(\sum_{i=1}^{N} w_i W_T(s_i^{(d)}))$
$= \sum_{i=1}^{n} ln(w_i W_T(c)) - ln(w_i W_T(s_i^{(d)}))$
$= \sum_{i=1}^{n} ln(W_T(c)) - ln(W_T(s_i^{(d)})) + ln(w_i) - ln(w_i)$
$< \sum_{i=1}^{n} O(T) = nO(T) = O(T)$
$\therefore R_T(s_i^{(d+1)}) < O(T)$ □

So, nth-degree strategies are online and have sublinear regret.

4.1.2. Runtime

Let's further understand the runtime of Nth-degree strategies. Let $O(w) = max(\{O(s_i)\, \forall i \in [1, m]\})$ or the worst runtime of a 0th degree strategy. Clearly, it takes $O(n * w)$ time to compute all 0th-degree strategies. Given that the 0th-degree strategies are precomputed, it will take another $O(n * w)$ time to compute all 1st-degree strategies or $O(n * w) + O(n * w) = 2O(n * w)$ time in total to compute the all 1st-degree strategies. Following this pattern of pre-computation, it will take $d * O(n * w)$ time to compute all $d - 1$ degree strategies. Given this precomputation, it will take $O(w)$ time to compute the dth-degree strategy or $d * O(n * w) + O(w) = O(n * w * d)$ time to compute a dth-degree strategy. Using dynamic programming, we can make the runtime of a dth-degree strategy linear in terms

of d. This circumvents the recursive nature of a dth-degree strategy, which when implemented poorly could create a very inefficient algorithm.

4.1.3. HYPOTHESIS

We hypothesize that high-degree OLPSs are able to outperform their base strategy and likely their 1st-degree strategy [common definition of meta-learners]. The optimal degree will depend on the assets traded, base strategies, and time period. This degree may be determined by the performance of various degrees of the same base strategy on the train set and can be tested on the test set. The exact degree is likely difficult to estimate and better saved for computational results on the train and test sets.

4.1.4. TRAIN RESULTS

Now we are able to train these strategies from 2010-2020. Figure 4 displays the wealth of a given strategy at varying degrees.

Firstly, we see that FTW strategies accumulate more wealth at higher degrees, while FTL and correlation strategies accumulate the most wealth around the 1st or 2nd degree. We can analyze the Sharpe ratios and volatility [standard deviation of returns] to make sure that we are getting the full picture.

Table 1. Sharpe Ratio on Train Set

Degree	UP	EG	PAMR	OLMAR	RMR	CORN	ONS
Degree_0	0.626187	0.623242	1.006900	0.834050	0.793279	0.928044	0.497848
Degree_1	0.940998	0.938118	0.845562	1.058154	0.998033	0.777952	0.737514
Degree_2	0.985156	0.984917	0.759415	0.927993	0.930661	0.856015	0.938137
Degree_3	0.946288	0.946797	0.912035	0.838745	0.864952	0.875373	0.954222
Degree_4	0.921406	0.921790	0.945669	0.953328	0.950013	0.813847	0.942634

Table 2. Volatility of Train Set

Degree	UP	EG	PAMR	OLMAR	RMR	CORN	ONS
Degree_0	0.115754	0.115837	0.187018	0.193949	0.194657	0.173505	0.143930
Degree_1	0.135353	0.134898	0.173591	0.176660	0.177851	0.175709	0.121770
Degree_2	0.144311	0.143981	0.162422	0.164771	0.166155	0.172341	0.136122
Degree_3	0.149439	0.149392	0.161294	0.161413	0.160829	0.161717	0.147167
Degree_4	0.152908	0.152864	0.156684	0.157316	0.157995	0.157208	0.151525

Here, FTW strategies have the highest Sharpe ratio around degree 2 and FTL strategies have the highest Sharpe ratio around degree 1. Further, FTW strategies have increasing volatility [most of which is likely 'upwards volatility'] for higher degrees, while the FTL strategies have decreasing volatility at higher degrees. Overall, these results imply that 2nd-degree strategies are likely optimal for FTW strategies and 1st-degree strategies are optimal for FTL strategies.

4.2. Test Results

Now, we may test these strategies to make sure that our results hold from 2020-2024, data which hasn't been used at all so far. The same graphs and tables on the test set are provided in Figure 5.

These results do match our hypothesis from the train results. Namely, we can achieve higher wealth than the base strategy using 2nd-degree FTW strategies and 1st-degree FTL strategies.

4.2.1. ANALYSIS

Overall Nth-degree strategies can combine returns from lower-degree strategies to give higher returns. Nth-degree strategies have been shown to perform better with FTW than FTL strategies. Our initial intuition for this is that it makes more sense to use FTL strategies on assets more than on strategies—there's a higher probability that a strategy that's underperforming is a 'bad' strategy than an asset that's underperforming is a 'bad' asset, especially when the assets tested have been hand chosen to be 'good' assets. Therefore, it would make sense for FTW strategies to be better meta-learners than FTL strategies—there are plenty of strategies which are very bad but only a handful that a good. Regardless, the nth-degree strategies can take advantage of each other, and it makes intuitive sense that the best degree of a FTW strategy is one degree larger than a FTL strategy [i.e. the FTW strategy can take perfect advantage of the best FTL strategy before that strategy diminishes]. Further, at high degrees, Nth-degree meta-learners seem to simply average out the returns of all base strategies. Intuitively this makes sense as meta-learning is a sort of averaging.

Overall, the initial results of Nth-degree meta-learners are very promising. At certain degrees, they are reliably, by the train test split setup, able to provide higher returns with a higher Sharpe ratio. The optimal degree is very dependent on the data and base strategy. Although clear patterns seem to emerge which are dependent on the class of portfolio optimization strategy.

4.3. Particle-Swarm-Optimizer Meta-Learner

Particle Swarm Optimization (Kennedy & Eberhart, 2002) is a meta-heuristic stochastic optimization algorithm which is biologically inspired by the movement of a flock of birds or a school of fishes. It involves initializing particles, randomly across the search space and iteratively improving their positions based on the value of the "fitness function" that is optimized. Individually the particles contain no intelligence but by interacting with other particles they can explore the search space efficiently.

Every particle is a solution to the optimization problem. In our work, the position of each particle represents a weight vector $w = [w1, w2, ..., wk]$, which is the weight assigned to the decisions of k base algorithms or experts. In this formulation, $\sum w_i = 1$ and $0 \leq w_i \leq 1$. In essence, each particle is its meta-learner. Each particle has an associated

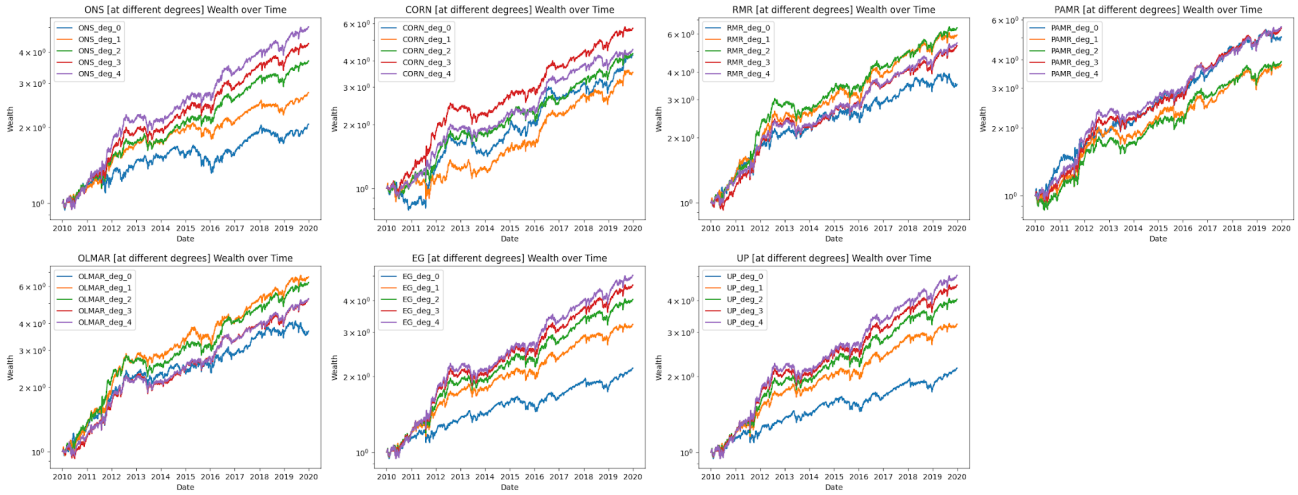

Figure 4. This figure shows the wealth generated by different Nth-degree Meta-Learners from 2010-2020.

velocity component. The basic idea lies in accelerating each particle towards its personal best position (i.e. solution) achieved so far and also towards the best solution attained so far by any particle in the swarm. The update equations for the swarm are given as,

$$v^{t+1} = m * v^t + c_1 * r_1 * (P^t_{best} - x^t) + c_2 * r_2 * (G^t_{best} - x^t) \quad (1)$$

$$w^{t+1} = w^t + v^{t+1} \quad (2)$$

where $c1 = 1$ and $c2 = 2$ are constants called the cognitive and social factors of the swarm respectively. $r1, r2$ are random numbers from $Unif(0, 1)$. P^t_{best} and G^t_{best} is the best position of individual particles and the swarm as a whole until time t.The goal of the swarm is to optimize the fitness function. In our implementation, the fitness function was the logarithm of wealth until time t. The particles explore the search space to maximize this fitness function. A good property of PSO is that the particles have an associated momentum component controlled by a hyper-parameter m, which aids in exploring the search space and not getting stuck in local optima.

4.3.1. HYPOTHESIS

We hypothesize that the momentum of the particles will help the swarm to discover and adapt to better solutions in the dynamic market where we do not know which base expert performs better (Fig 3). We hypothesize that combining the decisions of multiple base experts would help in generating higher wealth in some periods in a dynamic market with minimal compromise in run-time. This idea is tangential to boosting algorithms in machine learning, which is also an ensemble meta-heuristic for increasing robustness.

4.3.2. ONLINE PSO

We train the PSO on data until 2010 until convergence to find the optimal weights for each base expert (or strategy), similar to 6. This strategy weight is updated using PSO every 30 days by running just a few iterations on the data collected in the elapsed 30 days. This strategy weight is then utilised to generate wealth during the next 30 days. This is different from just following the best base expert since we observe that PSO can weigh and utilise the decisions of multiple base experts to generate higher wealth than any of the individual base experts in this period.

4.3.3. RESULTS

Firstly, we run our PSO algorithm on the benchmark algorithms using data from 2010-2020. We notice that PSO outperforms all base experts by using a weighted combination of PAMR and CORN algorithms (Fig 6). This is consistent with our training results in section 4.1.4, which shows that PAMR and CORN generates the highest wealth in 2010-2020. But PSO leverages the best of both these algorithms to outperform them.

We implemented an online version of the PSO algorithm. We run the PSO algorithm every 30 days to find the optimal weights to be assigned to the base experts. This weight is then used for the next 30 days.

We observe similar gains in wealth (4x) in the period 2020-2024 which we refer to as the "test dataset" in this paper.

4.3.4. ANALYSIS

We see that during certain periods the PSO algorithm can give equally high weights to multiple algorithms thereby

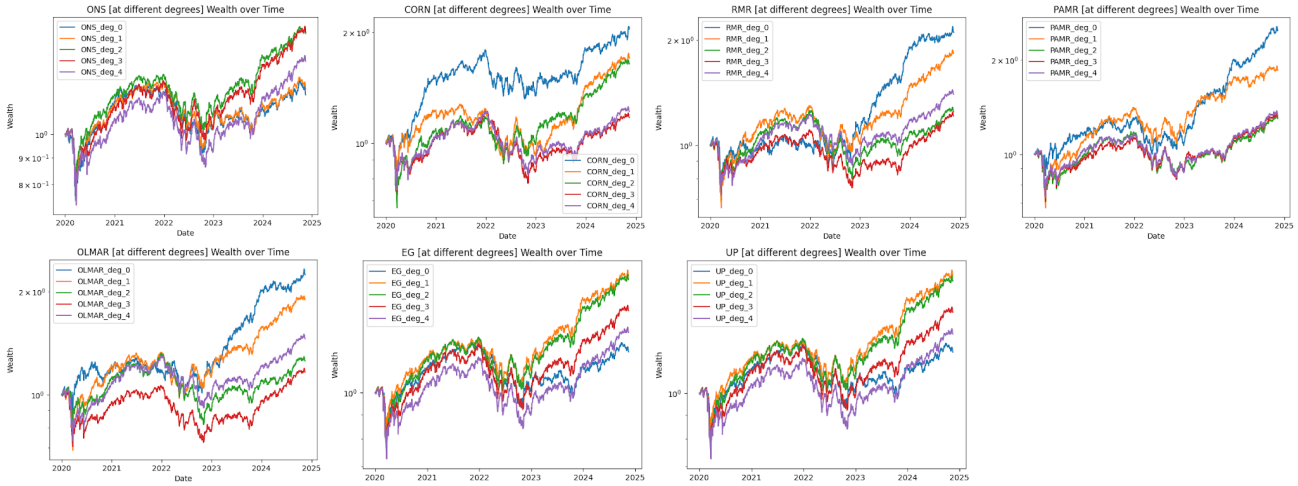

Figure 5. This figure shows the wealth generated by different Nth-degree Meta-Learners from 2020-2024, which we refer to as the testing period.

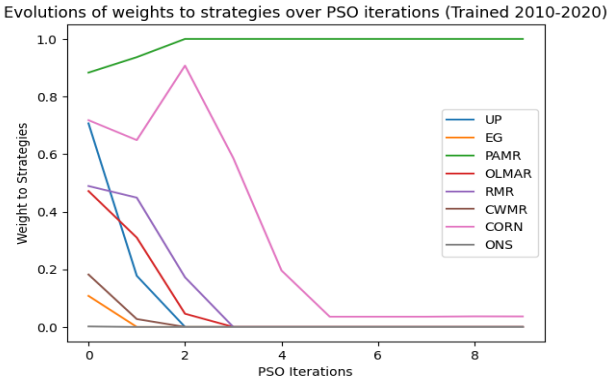

Figure 6. PSO converges to weighing strategy that gives a very high weight to PAMR and small weight to CORN, when trained on data from 2010-2020. This strategy of weighing PAMR and CORN generates higher wealth than all base experts.

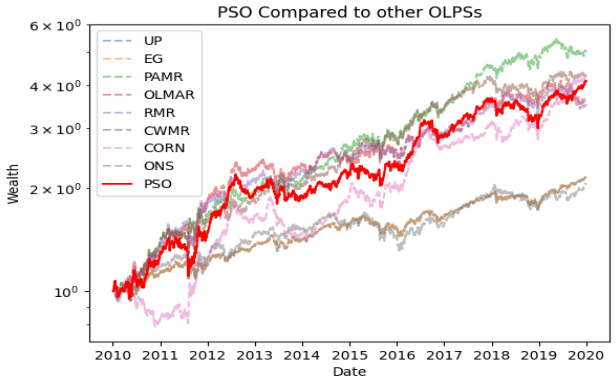

Figure 7. Comparison of the wealth generated using the proposed online PSO meta-learning algorithm against other benchmarks.

outperforming these algorithms in that period. This is in line with our hypothesis.

However, when used in the fashion suggested in the previous section, we notice that a few base experts outperform the PSO algorithm in the period 2010-2020 7. Initial analysis attributes this to the fact that we only run the PSO every 30 days. Therefore, the algorithm is currently unable to exploit the market dynamics in this period. A way to circumvent this pitfall is to track the market volatility and adaptively run PSO to learn the weights for the strategies. We leave this for future work.

Regardless, this PSO algorithm is a valid meta-learning algorithm in performing significantly better than the worst OLPS and a portfolio consisting of even weighting of OLPSs. In this way, choosing this PSO algorithm is better than arbitrarily choosing any OLPS.

5. Conclusion and Future Work

This work presents two promised meta-learning algorithms for Robust Online-Portfolio Optimization. The Nth-Degree Meta-Learners have been shown to have sub-linear regret, time complexity linear in n, and higher returns/Sharpe ratio for $n > 0$. Future work includes testing these meta-learners on different sets of data, including more online portfolio strategies to choose from, and adding a native currency as a potential asset at all degrees. Further combining Nth-Degree

Meta-Learners such that an Nth-degree strategy could utilize some subset of all $d < N$th-degree strategies would also be very interesting. The particle swarm optimization algorithm is an extremely efficient algorithm for learning weights for these online optimization algorithms. We show that combining the decisions of multiple base experts helps in superior performance in some periods in a dynamic market with minimal compromise in run-time. We would like to draw parallels between our approaches and boosting algorithms in machine learning, which is also an ensemble meta-heuristic for increasing robustness. The PSO approach has multiple ideas that we defer for future work. Firstly, improvements to the PSO approach which seem very promising include adaptively running PSO by tracking the market volatility and also studying the effects of the number of particles and better ways to re-initialize particles in the search space. We would also like to study the effect of transaction cost on our algorithms. In a real-world setting, transferring wealth between assets attracts a fee which must be included as a constraint to the optimization problem.

Acknowledgements

We would like to thank Professor Francesco Orabona for his book (Orabona, 2019), which introduced us to the field of online learning this semester. We recommend readers refer to the proofs for sub-linear regret for the benchmark online learning algorithms used in this work from this book.

References

Borodin, A., El-Yaniv, R., and Gogan, V. Can we learn to beat the best stock. *Advances in Neural Information Processing Systems*, 16, 2003.

Cover, T. M. and Ordentlich, E. Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2):348–363, 1996.

Das, P. and Banerjee, A. Meta optimization and its application to portfolio selection. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1163–1171, 2011.

Hannan, J. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3(2):97–139, 1957.

Helmbold, D. P., Schapire, R. E., Singer, Y., and Warmuth, M. K. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347, 1998.

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

J.Milnor. *Games against nature*. Technical Report RM-679, RAND PROJECT AIR FORCE, 1951.

Kelly, J. L. A new interpretation of information rate. *the bell system technical journal*, 35(4):917–926, 1956.

Kennedy, J. and Eberhart, R. Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4:1942–1948 vol.4, 2002. URL https://api.semanticscholar.org/CorpusID:3114196.

Li, B. and Hoi, S. C. On-line portfolio selection with moving average reversion. *arXiv preprint arXiv:1206.4626*, 2012.

Li, B., Zhao, P., Hoi, S. C., and Gopalkrishnan, V. Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine learning*, 87:221–258, 2012.

Li, B., Hoi, S. C., Zhao, P., and Gopalkrishnan, V. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(1):1–38, 2013.

Orabona, F. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

Poterba, J. M. and Summers, L. H. Mean reversion in stock prices: Evidence and implications. *Journal of financial economics*, 22(1):27–59, 1988.

Savage, L. J. The theory of statistical decision. *Journal of the American Statistical association*, 46(253):55–67, 1951.

Vovk, V. and Watkins, C. Universal portfolio selection. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pp. 12–23, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 1581130570. doi: 10.1145/279943.279947. URL https://doi.org/10.1145/279943.279947.

Wald, A. and Wolfowitz, J. Bayes solutions of sequential decision problems. *The Annals of Mathematical Statistics*, pp. 82–99, 1950.

Zhang, Y., Lin, H., Li, J., and Yang, X. Combined peak price tracking strategies for online portfolio selection based on the meta-algorithm. *Journal of the Operational Research Society*, 75(10):2032–2051, 2024.