Guilherme Augusto Schütz

A neural network for online portfolio selection with side information

DISSERTAÇÃO DE MESTRADO

DEPARTAMENTO DE INFORMÁTICA
Programa de Pós-Graduação em Informática

Rio de Janeiro
August 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Guilherme Augusto Schütz

A neural network for online portfolio selection with side information

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Ruy Luiz Milidiú

Rio de Janeiro
August 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Guilherme Augusto Schütz

A neural network for online portfolio selection with side information

Dissertation presented to the Programa de Pós-Graduação em Informática, of PUC-Rio, in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Undersigned Examination Committee.

Prof. Ruy Luiz Milidiú
Advisor
Departamento de Informática — PUC-Rio

Prof. Edward Hermann Haeusler
Departamento de Informática — PUC-Rio

Prof. Marco Serpa Molinaro
Departamento de Informática — PUC-Rio

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico — PUC-Rio

Rio de Janeiro, August 1st, 2018

Guilherme Augusto Schütz

Bachelor's in Economic Science at the *Universidade do Estado de Santa Catarina* — UDESC (2013).

Bibliographic data

Schütz, Guilherme Augusto

A neural network for online portfolio selection with side information / Guilherme Augusto Schütz ; advisor: Ruy Luiz Milidiú. — 2018.
65 f. : il. ; 30 cm

Dissertação (Mestrado em Informática)-Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2018.
Inclui bibliografia

1. Informática – Teses. 2. Aprendizado de Máquina. 3. Redes Neurais. 4. Aprendizado em tempo real. 5. Seleção de portfólio. 6. Finanças computacionais. 7. Otimização convexa. I. Milidiú, Ruy Luiz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

This research project was made possible thanks to funds and support provided by PUC-Rio, CAPES and BTG Pactual.

Abstract

Schütz, Guilherme Augusto; Milidiú, Ruy Luiz (advisor). **A neural network for online portfolio selection with side information**. Rio de Janeiro, 2018. 65p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The financial market is essential in the economy, bringing stability, access to new types of investments, and increasing the ability of companies to access credit. The constant search for reducing the role of human specialists in decision making aims to reduce the risk inherent in the intrinsic emotions of the human being, which the machine does not share. As a consequence, reducing speculative effects in the market, and increasing the precision in the decisions taken. In this paper, we discuss the problem of selecting portfolios online, where a vector of asset allocations is required in each step. The proposed algorithm is the *multilayer perceptron with side information* - MLPi. This algorithm uses neural networks to solve the problem when the investor has access to future information on the price of the assets. To evaluate the use of side information in portfolio selection, we empirically tested MLPi in contrast to two algorithms, a baseline and the state-of-the-art. As a baseline, buy-and-hold is used. The state-of-the-art is the *online moving average mean reversion* algorithm proposed by Li & Hoi (2012). To evaluate the use of side information in the algorithm MLPi a benchmark based on a simple optimal solution using the side information is defined, but without considering the accuracy of the future information. For the experiments, we use minute-level information from the Brazilian stock market, traded on the B3 stock exchange. A price predictor is simulated with 7 different accuracy levels for 200 portfolios. The results show that both the benchmark and MLPi outperform the two algorithms selected, for asset accuracy levels greater than 50%, and on average, MLPi outperforms the benchmark at all levels of simulated accuracy.

Keywords

Machine Learning; Neural Networks; Online Learning; Portfolio Selection; Computational Finance; Convex Optimization;

Resumo

Schütz, Guilherme Augusto; Milidiú, Ruy Luiz. **Uma rede neural para o problema de seleção online de portfólio com informação lateral**. Rio de Janeiro, 2018. 65p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O mercado financeiro é essencial na economia, trazendo estabilidade, acesso a novos tipos de investimentos, e aumentando a capacidade das empresas no acesso ao crédito. A constante busca por reduzir o papel de especialistas humanos na tomada de decisão, visa reduzir o risco inerente as emoções intrínsecas do ser humano, do qual a máquina não compartilha. Como consequência, reduzindo efeitos especulativos no mercado, e aumentando a precisão nas decisões tomadas. Neste trabalho é discutido o problema de seleção de portfólios online, onde um vetor de alocações de ativos é requerido em cada passo. O algoritmo proposto é o *multilayer perceptron with side information* - MLPi. Este algoritmo utiliza redes neurais para a solução do problema quando o investidor tem acesso a informações futuras sobre o preço dos ativos. Para avaliar o uso da informação lateral na seleção de portfolio, testamos empiricamente o MLPi em contraste com dois algoritmos, um baseline e o estado-da-arte. Como baseline é utilizado o *buy-and-hold*. O estado-da-arte é o algoritmo *online moving average mean reversion* proposto por Li & Hoi (2012). Para avaliar a utilização de informação lateral no algoritmo MLPi é definido um benchmark baseado numa solução ótima simples utilizando a informação lateral, mas sem considerar a acurácia da informação futura. Para os experimentos, utilizamos informações a nível de minuto do mercado de ações brasileiro, operados na bolsa de valores B3. É simulado um preditor de preço com 7 níveis de acurácia diferentes para 200 portfólios. Os resultados apontam que tanto o benchmark quanto o MLPi superam os dois algoritmos selecionados, para níveis de acurácia de um ativo maiores que 50%, e na média, o MLPi supera o benchmark em todos os níveis de acurácia simulados.

Palavras-chave

Aprendizado de Máquina; Redes Neurais; Aprendizado em tempo real; Seleção de portfólio; Finanças computacionais; Otimização convexa;

Contents

List of Figures

List of Tables

Summary of notations

AC	anti correlation algorithm
APP	asset price prediction problem
ASTDV	annualized standard deviation
B3	B3 S.A. *Brasil, Bolsa, Balcão*
BBH	best offline buy-and-hold
BCRP	best offline constant rebalancing portfolio
BH	buy-and-hold
BM&FBOVESPA	BM&FBOVESPA S.A. Securities, Commodities & Futures Exchange
CRP	constant rebalancing portfolio
CWMR	confidence weighted mean reversion
EG	exponential gradient
EM	expectation maximisation
FTL	Follow-the-Loser
FTW	Follow-the-Winner
GPSP	general portfolio selection problem
MLP	multilayer perceptron
MLPi	multilayer perceptron with side information
OLMAR	online moving average mean reversion algorithm
ONS	online Newton step algorithm
OPT	optimal offline algorithm
OPTi	optimal online algorithm with side information
PSP	portfolio selection problem
ReLU	Rectified Linear Unit
SGD	stochastic gradient descent
UP	universal portfolio

A good player is always lucky.

Capablanca

1
Introduction

The financial market plays an important role in the economy, increasing companies liquidity and offering more options to economic agents for investing their savings. Portfolio selection (PSP) is a well known problem, both in the finance and machine learning communities. It consists of choosing how much to invest in each one of the available assets in a market. For each time step $t = 1, \ldots, T$, it asks for an allocation $\mathbf{b}(t)$ in m assets, where $\mathbf{b}(t) = (b_1(t), \ldots, b_m(t))$ and

$$b_1(t) + \cdots + b_m(t) = 1 \; .$$

The main finding over recent years is the design of algorithms that offer a rebalancing strategy, changing $\mathbf{b}(t)$ at each time step, with some enhancement guarantees over a fixed strategy. Two types of regret based algorithms have been developed for PSP in the last decades, namely: Follow-the-Winner (FTW) and Follow-the-Loser (FTL). The FTW algorithms (Hazan et al., 2007), choose the best assets in the past to make the portfolio allocation decision. On the other hand, the FTL algorithms buy losers and sell winners. This stands on the *reversion to the mean* hypothesis, where stock prices tend to achieve a historical mean value over time. The surveys by Li & Hoi (2016) and Dochow (2016) summarise these findings.

Some of the most known portfolio selection algorithms follows the seminal work of Cover (1991). Next, we enumerate six of this algorithms: UP, ONS, OLMAR, EG, AC and CWMR. The universal portfolio (UP) is the first *universal* algorithm, i.e., one that presents asymptotically regret bounds over fixed strategies in hindsight. For FTL algorithms, it is difficult to obtain a lower bound regret on the performance in comparison to a benchmark algorithm, since there is no guarantee of the mean reversion hypothesis.

Helmbold et al. (1998) propose the exponential gradient (EG) strategy, which maximises the expected logarithmic return, estimated by the last relative return, and also minimises the deviation from the last allocation. It can be seen as a variant of gradient descent but performing sub-optimal regret. Borodin et al. (2004) propose the anti correlation algorithm (AC), with the underlying assumption that if an asset in a past time window shows a distinctly different performance than other assets, it is more likely indicating a counter-movement of performance in the future. Li et al. (2013) develop the confidence weighted mean reversion (CWMR), with a novel approach that exploits the second order information of the portfolio (not the second order information of the assets returns). It takes advantage of the reversion to the mean property by applying the *confidence-weighted*[1] learning technique. Additionally, it has a regret bound and is a universal strategy.

An additional issue, when defining a portfolio strategy, is the uncertainty of future asset prices. The auxiliary data used to predict future price is called *side information*. As a consequence, several research efforts on stock market focus on the price forecasting task, as described by Patel et al. (2015). Bengio (1997) explores the use of side information as a learning problem. He evaluates his strategy with an experiment in the financial market of Canada. The experiment takes 35 stocks plus the money asset. Its purpose is to show that a financial objective for the learning phase, such as maximising the return, gives better results than minimising the squared error or maximising the likelihood. His initial argument is based on the high noise in the stock market historical prices.

Reinforcement Learning (RL) has been applied to solve PSP. Moody et al. (1998) propose the recurrent reinforcement learning model, applied to the stock market. They also compare it with multiple objective functions, where the differential Sharpe ratio shows the best results. The proposed trading algorithm restricts the allocation to only one asset at each time-step. Martinez et al. (2009) propose a simple rule based decision, just for selecting a portfolio allocation at each time, using the price prediction of a neural network model as input.

Here, we propose multilayer perceptron with side information (MLPi), a novel approach to the online portfolio selection problem (PSP) by incorporating side information. Our focus here is the management of wealth by portfolio reallocation at each time step. For that sake, we define a classification task, where the allocation proportion is the probability of the best classified asset. We propose the MLPi algorithm for solving this task, using future price information of a simulated predictor. For the sake of comparison with MLPi, we select the online Newton step algorithm (ONS) presented by Agarwal et al. (2006). ONS is chosen since it is *universal*, and also the first polynomial time algorithm for PSP. Another algorithm that we implement in this work

[1] The confidence-weighted (CW) learning is proposed by Crammer et al. (2009) as an algorithm that updates both the classifier and the estimate of their parameters confidence.

is the online moving average mean reversion algorithm (OLMAR). This is an algorithm (Li & Hoi, 2012) from the FTL family, that assumes the reversion to the mean hypothesis. It also takes advantage of side information as well, but in a simplified way, that is, by using the moving average of the last w time-steps.

The main contributions of our work are the following:

1. the integration of price prediction models into an algorithm for solving PSP;

2. a neural network based solution to the PSP;

3. a new benchmark algorithm OPTi, using a naive optimal decision, restricted to a finite solution and incorporating side information;

4. an empirical validation of the MLPi and OPTi with historical data from B3 stock exchange;

5. the simulation of an *asset price predictor* for several levels of accuracy, indicating an expected accuracy threshold that a prediction algorithm should perform;

6. the use of the *cosine similarity* as a portfolio validation metric for the training phase.

This work is organised as follows. In Chapter 2, we present a formal statement of the general PSP problem. In Chapter 3, we review some basic concepts of the literature. In Chapter 4, we formalise the selected algorithms, including ONS and OLMAR. In Chapter 5, we discuss and formalise the proposed MLPi algorithm. In Chapter 6, we report our findings on the empirical experiments and comparative analysis of the selected algorithms. In Chapter 7, we outline our main conclusion and propose some future work.

2
Problem Statement

In this chapter, we provide a formal definition for the PSP and its extension, the general portfolio selection problem (GPSP). Next, we define the optimal benchmark with side information.

2.1
General portfolio selection problem

An asset is a financial product. It is commercialised by shares, that represent ownership of asset parts. Here, we assume that shares are infinitely divisible. A portfolio is a selection of m assets. Given a fixed amount of wealth, an allocation strategy is a setting of different proportions of wealth in the assets of a portfolio. A reallocation of wealth can be made on the m assets at the beginning of each time instant with $t = 1, \dots, T$ and $T \geq 1$. All available wealth should be allocated on assets, so it is possible and convenient to have a *money asset*, which is assumed to have constant price for all t. We refer to A as all assets including the money asset, and to A^* as all assets without the money asset. The price is a relative conversion rate between two assets, with infinite available buyers and sellers willing to trade.

Definition 1 *Assume that, for each time instant $t = 1, \dots, T$, we are given m assets, with $m \geq 2$. Then, PSP is the online problem that asks for a sequence $\mathbf{b}(t) = (b_1(t), \dots, b_m(t))$ of asset transactions, taking a financial objective into account.*

The general case of PSP is the GPSP, where the financial objective is to maximise the terminal wealth W_T. Formally, GPSP is given by

$$\underset{b}{\text{maximize}} \qquad W_T = W_o \prod_{t=0}^{T} \sum_{i=1}^{m} b_{ti} r_{ti} \tag{2-1a}$$

$$\text{subject to} \quad \sum_{i=1}^{m} b_{ti} = 1, \qquad \forall\, t = 0, \dots, T, \tag{2-1b}$$

$$b_{ti} \geq 0, \qquad \forall\, i \in A,\ t = 0, \dots, T \tag{2-1c}$$

where r_{ti} is asset i return at time step t, given by the relative change on the *asset value*[1] , that is,

$$r_{ti} = \frac{q_{ti}}{q_{(t-1)i}} \tag{2-2}$$

where q_{ti} is asset i price at time t. Note that if an investor uses an allocation $\mathbf{b}(t)$ on step t, his wealth changes by a factor of $\mathbf{b}(t)\mathbf{r}(t)$.

PSP can be viewed as an online convex optimisation problem (Hazan et al., 2007), where a decision maker takes a sequence of decisions, choosing a sequence of points in a convex set, from a fixed feasible set. At each chosen point, a payoff function is defined. For the GPSP, the payoff is a change in the portfolio wealth after the decision is made.

The objective function (2-1a) returns the wealth W_T made at the end of a given time interval, where W_o is the initial wealth, that we set as $W_o = 1$. The constraint (2-1b) states that we cannot allocate more resources than we have and that all available wealth must be allocated. The constraint (2-1c) guarantees that there is no short-selling [2].

Now, let us introduce the optimal offline algorithm (OPT), the optimal solution to the PSP problem. OPT assumes a prescient investor, that is, an investor with 100% accuracy on its asset return predictions. This is also the case of an offline algorithm where all future information is know in advance. In Table 2.1, we present an illustrative example of this algorithm.

	0	**1**	**2**	**3**	**T = 4**
b_1	0.50	-	1.00	1.00	-
b_2	0.50	1.00	-	-	1.00
q_1	1.00	1.00	1.00	1.00	1.00
q_2	10.00	12.00	9.00	9.00	10.00
r_1	1.00	1.00	1.00	1.00	1.00
r_2	1.00	1.20	0.75	1.00	1.11
W	1.00	1.20	1.20	1.20	1.33

Table 2.1: Offline solution of the GPSP in a hypothetical portfolio.

Observe that the wealth achieved at the end of the time interval is $W_T = 1.33$.

[1] In this work we only analyse the stock market, so q_{ti} is the *close price* of one share of stock i at step t, as t is an interval of 1 minute, the *close price* is the price of the last trading on that minute.

[2] Short-selling is the activity of selling an asset without owing that asset.

The asset b_1 is the money asset. We assume no money inflation during the time interval and define the *best asset* at step t as the asset of step t that gives the best wealth at T.

2.2 Optimal benchmark

Our main benchmark is a modification of OPT, since we don't have the predicted future price only a classification of the predicted future price. We refer to this algorithm as the optimal online algorithm with side information (OPTi), that we detail next.

Consider a model that receives partial information for the return of an asset i at step t. That is, the model receives a label $\rho_i(t+1) \in \{c_1, \ldots, c_k\}$, indicating a relative price change of asset i at time $t+1$, with a probability p. Now, we describe the algorithm OPTi to be used as a benchmark. Suppose that the partial information ρ comes from an external source, like an expert. A naive solution for choosing how much to invest on each asset is to believe 100% on the expert, when the expert has the same p for each asset.

For a ternary classification, where we have a $\rho_i(t) \in \{-1, 0, 1\}$, a greedy solution for OPTi is given. At each time step t an investor receives information $\rho_i(t+1)$ from an expert for each asset $i \in A$, and must decide how much to invest in each asset i at step t to maximise is terminal wealth. At each step, the investor use the side information $\rho(t+1)$ and allocates all the wealth equally among all assets that will *go up*, i.e. $b_{ti'} = \frac{1}{|A'_t|}$, where $A'_t = \{i' \in A \mid \rho_{i'}(t+1) = 1\}$. When A'_t is empty, the investor must allocate all is wealth in the money asset.

3
Basic Concepts

The seminal work of Markowitz (1952) introduces the well know mean-variance model. This is the first PSP contribution to the financial market. The model aims to minimise the risk of a portfolio expected return. Where the mean of past returns is the expected return, and the variance of the past returns is the measure of risk.

This is clearly part of a long period portfolio selection, not only because of the complexity of the problem at that time but because of investment companies that offer selected portfolios in form of mutual funds as a more secure investment. The main contribution of Markowitz work is that diversification of assets when selecting portfolios can reduce the risk with the same expected return, explained by the negative correlation of assets in that portfolio.

The main strategy for long period investments is the buy-and-hold (BH). For our given investment horizon of $[0, T]$, the investor allocates all the wealth in a portfolio at step 0, and withdraw all amount invested at step T. This is a fixed strategy, there is no rebalancing of wealth along that investment horizon.

On the other hand, some investor may want to perform changes in his portfolio allocation, changing the amount of each asset along the investment horizon. Besides in the real market, there are costs to realise this approach, which in some ways discourages investors to perform these changes, a lot of contributions in the past decades make better returns possible over the fixed strategies.

These strategies are known as *rebalancing portfolios*. A naive strategy of this kind is the constant rebalancing portfolio (CRP). For each asset, an investor defines a constant wealth allocation proportion. Additionally, at each time step $t \in \{1, \ldots, T\}$, the investor readjusts the allocated amount to that proportion. Observe that both CRP and BH uniformly distribute the wealth among the assets in A^*.

A strategy such as the rebalancing portfolios (Kelly, 1956) is called a *Kelly investment*. The metaphor for these kind of investments is a betting game. The gambler's goal is to maximise its expected return, given the probability of winning the bet over a multiperiod time step. It is shown that the gambler should not invest all the available wealth at time zero. In fact, he

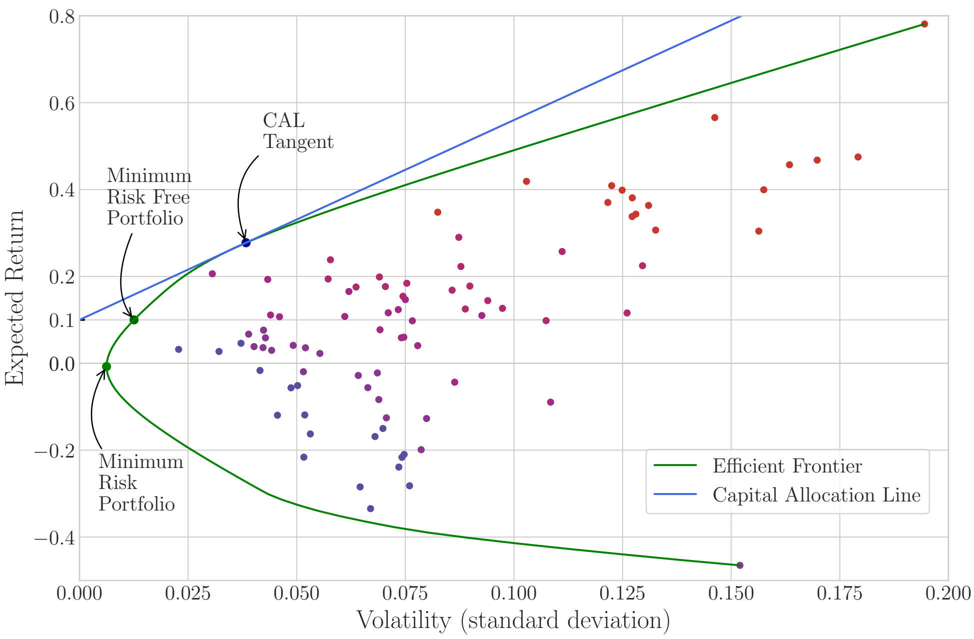

Figure 3.1: Example of the efficient frontier obtained from the Markowitz mean-variance model for a portfolio with $m = 90$ assets from B3; the green line defines the convex set of all possible allocation solutions b over all assets, expressed by the coloured points; the *capital allocation line* in blue express the reward-to-variability ratio, presented by Sharpe (1966), the slope of the line is the Sharpe ratio, when the line tangent the efficient frontier, coming from a risk-free ratio ($r_f = 10\%$), is called *capital market line*, is the maximum reward-to-variability for a given risk-free asset.

should invest along the time steps proportionally to the winning probability. This is due to the exponential return of the game along the time steps. The problem is very close to the *St. Petersburg paradox*, proposed by Nicolas Bernoulli in the eighteen century. In the financial market, this type of game is more present in the form of stock options, where the investor bets on the appreciation/depreciation of an asset in the future. In this case, the investor can lose all the money invested in that stock option. Nevertheless, the only way this can happen is in the rare case of a company bankruptcy.

Other variations of strategies are the *best* CRP (BCRP) and the *best* BH (BBH). BCRP's optimal solution is given by the solution of the objective (3-1)

$$\underset{b}{\text{maximize}} \quad W_T^{\text{BCRP}} = W_o \prod_{t=0}^{T} \sum_{i=1}^{m} b_i^* r_{ti} \tag{3-1}$$

Whereas, BBH's optimal solution is given by the solution of the objective (3-2)

$$\underset{b}{\text{maximize}} \quad W_T^{\text{BBH}} = W_o \sum_{i=1}^{m} b_i^* \prod_{t=0}^{T} r_{ti} \tag{3-2}$$

Problems (3-1) and (3-2) are restricted to the same constraints of GPSP (2-1b) and (2-1c). The solution of BBH is clearly the best asset in A.

3.1 Universality and Regret

The comparison of different online algorithms is made by *competitive analysis* (Koutsoupias & Papadimitriou, 2000). The performance of an online algorithm ALG is compared to the performance of an offline algorithm OPT, that knows the input sequence beforehand. There are three types of worst-case competitiveness, namely: competitive ratio, performance ratio and comparative ratio. Dochow (2016) shows that all these types are equivalent, when considering all problem instances and all online and offline algorithms. A problem instance input $\mathbf{x}$ is a possible market, i.e., a portfolio of m assets and T time steps.

The *competitive ratio* of an ALG can be defined as

$$c = \max_{\mathbf{x}} \frac{Perf(\text{OPT}, \mathbf{x})}{Perf(\text{ALG}, \mathbf{x})} \tag{3-3}$$

where $\mathbf{x}$ ranges over all possible markets, $Perf(\cdot)$ is a performance function like W_T and OPT is the best performing benchmark. The smaller c the more powerful is the online ALG. It follows that an ALG is considered c-competitive if the lower bound for the performance can be formulated as

$$Perf(\text{ALG}, \mathbf{x}) \geq \frac{1}{c} Perf(\text{OPT}, \mathbf{x}) \tag{3-4}$$

which must be valid for any market $\mathbf{x}$ over all possible inputs. The *performance ratio* is obtained when the problem instance is restricted to a finite set $\mathbf{x} \in \mathbf{X}$

$$c(\mathbf{X}) = \max_{\mathbf{x} \in \mathbf{X}} \frac{Perf(\text{OPT}, \mathbf{x})}{Perf(\text{ALG}, \mathbf{x})} \tag{3-5}$$

And the *comparative ratio* when the benchmark algorithms are restricted to $B \in \mathcal{B}$

$$c(ALG, \mathcal{B}) = \max_{B \in \mathcal{B}} \max_{\mathbf{x}} \frac{Perf(\text{B}, \mathbf{x})}{Perf(\text{ALG}, \mathbf{x})} \tag{3-6}$$

therefore, if B consists of all possible online and offline benchmark algorithms, then $c(ALG, \mathcal{B}) = c$. As noted by Koutsoupias & Papadimitriou (2000), these comparisons are unfair since it always gives the worst case for ALG comparing with OPT.

Fujiwara et al. (2011) propose an *average-case performance ratio*,

$$\mathbb{E}[c(\mathbf{X})] = \mathop{\mathbb{E}}_{\mathbf{x}\in\mathbf{X}}\left[\frac{Perf(\text{OPT},\mathbf{x})}{Perf(\text{ALG},\mathbf{x})}\right] \tag{3-7}$$

where the set $\mathbf{X}$ assumes a given distribution. The stock market movements are assumed to perform a geometric Brownian motion.

Cover (1991) introduces another approach to PSP, the UP. Besides proposing an algorithm, the author defines *universality*. An online algorithm ALG that solves the PSP is *universal* when it satisfies

$$\frac{1}{T}\ln W_T^{\text{ALG}} - \frac{1}{T}\ln W_T^{\text{B}} \to 0 \qquad \text{as } T \to \infty \tag{3-8}$$

where B is the best algorithm in a set $\mathcal{B}$ of constant rebalancing algorithms. What can be generalised to

$$\frac{1}{T}\ln\frac{1}{c(\text{ALG},\mathcal{B})} \to 0 \qquad \text{as } T \to \infty \tag{3-9}$$

which quantifies the *extent of universality*, where c is the comparative ratio between ALG and the best algorithm in $\mathcal{B}$. Observe that, when the comparative ratio is exponential, the extent of universality does not converge to zero for increasing T. But it does for a logarithmic and constant comparative ratios.

The performance of an online ALG for PSP is often expressed by the *regret* (Dochow, 2016),

$$regret = -\ln\frac{1}{c(\text{ALG},\mathcal{B})} = \ln c(\text{ALG},\mathcal{B}) \tag{3-10}$$

which an online investor aims to minimise.

Since Cover & Gluss (1986), the class of benchmark algorithms $\mathcal{B}$ is restricted to algorithms that do not change the allocation proportion b over time steps. Therefore, the common practice is to use the best offline constant rebalancing portfolio (BCRP) as a benchmark. Even that BCRP perform sucessive reallocations of wealth among T, the proportions are always the same. For our purpose, that inserts side information of the available markets, the OPTi formulated in Section 2.2 seems more fair.

3.2
Deep learning

The concept of *learning* as a general process for human beings, is very close to the concept of machine learning. The difference is that a machine is requested to perform and learn tasks that a regular human normally does. Machine learning algorithms use computational methods to "learn" information from data without relying on predetermined rules. The algorithms can improve their performance as the number of available samples increases.

A definition on learning is presented by Mitchell (1997, p.2):

> **Definition 2** *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

So, a well defined learning problem should identify these three features: the class of tasks (T); the source of experience, i.e., the available information (E); and the measure of performance (P) to validate the experience taken.

As stated by Shalev-Shwartz & Ben-David (2014), a learning process is called *online*, when it is performed in a sequence of successive rounds. On each round, the learner receives an instance from a suitable domain. After this, the learner is asked to predict a label. At the end of the round, that label is revealed to the learner. With the corrected value in hands, the learner uses this information to improve future predictions.

A classical learning algorithm is the Perceptron (Rosenblatt, 1958; Freund & Schapire, 1999), which performs simple additive updates, for mistakes when classifying an incoming instance. This updates are performed by an activation function, that indicates whether that new weighted information should fire changes or not (in a binary activation), or how much of that weighted input should fire changes on the output. A detailed description of the online Perceptron can be found in Shalev-Shwartz & Ben-David (2014).

Deep learning provides a framework for supervised learning, by adding more layers and neurons connecting an input to an output data. This structure increases the complexity of problems that a network can represent and generalise, but needs more computational resources for solving them.

The multilayer perceptron (MLP) is a sequential composition of single perceptron layers. The backpropagation algorithm (Rumelhart et al., 1986) simplifies the gradient computation for the MLP. Composite activation functions are universal function approximators (Cybenko, 1989), giving MLP more representation power and the ability to learn non-linearly separable problems.

We enumerate four commonly used activation functions: *sign*, *sigmoid*, *ReLU* and *ReLU6*. The *sign* is a simple binary function that indicates when a neuron should be fired or not. The *sigmoid* is given by $y = \frac{1}{1+e^{-x}}$ and is widely used. A problem that can arise with *sigmoid* is when changes of the gradient are small (low/high values of x) the neuron dispatch minor changes in the network reducing the learning capacity, a problem known as 'vanishing gradients'. The *ReLU* – rectified linear units – is expressed as $y = \max(x, 0)$, makes the learning process faster than other functions, and don't present the 'vanishing gradients' problem (Nair & Hinton, 2010). The *ReLU6* is a modification of *ReLU* expressed as $y = \min(\max(x, 0), 6)$, this modification encourages the model to learn sparse features earlier (Krizhevsky, 2010). The negative part of *ReLU* is the 'dying ReLU' problem (Maas et al., 2013), that happens when a large gradient update can inactivate a neuron.

In a supervised learning algorithm, the main goal is to minimise a *loss* function. We want to reduce the loss that results when a wrong output given by $\hat{Y}$ is predicted, instead of the ground truth value given by Y. A common and intuitive loss function is the *mean squared error* (MSE). Bengio (1997) states that, by minimising a financial goal, we would generate better financial results. This is the case, due to the high noise information that is common in this field.

For minimising the *loss* function and to update the parameters of $\hat{Y}$, the stochastic gradient descent (SGD) is a common heuristic approach. To compute these gradients, we use the *backpropagation* algorithm. Kingma & Ba (2014) propose the *Adam* optimiser, which is based on adaptive estimates on lower-order moments. *Adam* updates the parameters by using the first and second raw moment estimates.

A serious problem in deep neural networks is overfitting. This happens when a network fits well the train dataset, but presents low results in the validation phase. The network memorises the data, instead of generalising. The solution to this problem is regularisation, i.e., any modification of the learning algorithm intended to reduce the generalisation error (Goodfellow et al., 2016). A common technique is *dropout*. It consists of randomly dropping some units from the network during the training phase (Srivastava et al., 2014).

By the end of the training procedure, a composite function with fixed parameters is generated. Now, the model is ready to perform predictions for new entries and its prediction quality can be empirically evaluated.

4
Online Portfolio Selection Algorithms

This chapter formalises the two regret based algorithms implemented for the experiments, that is, the online Newton step algorithm (ONS) and the online moving average mean reversion algorithm (OLMAR). It is important to note that OLMAR is the current state-of-the-art algorithm (Dochow, 2016; Li & Hoi, 2016).

4.1
Online Newton Step Algorithm

The ONS (Agarwal et al., 2006) takes the first and second order information into account. So, for a r_t holding period return, giving by

$$r_t = \sum_{i=1}^{m} r_{ti} b_{ti} = \frac{W_t}{W_{(t-1)}} \tag{4-1}$$

it shows the variation of wealth for one trading period. The first order information can be retrieved by

$$\Theta_t^i = \frac{\partial \ln r_t}{\partial b_{ti}} = \frac{r_{ti}}{r_t} \tag{4-2}$$

and describes the price change of A_i in relation to the holding period return. Equation (4-2) gives the intuitive notion that, when $\Theta_t^i > 1$, the asset A_i performs *better* during trading time-step t. Otherwise, it performs worst, when $\Theta_t^i < 1$ for the same time-step.

The second order information, can be extracted by

$$\Theta_t^{ij} = \frac{\partial^2 \ln r_t}{\partial b_{ti} \partial b_{tj}} = -\frac{r_{ti} r_{tj}}{r_t^2} \tag{4-3}$$

and express the combined price change of asset A_i and A_j for $i, j = 1, \ldots, m$ in relation to the quadratic holding period return. A value of $\Theta_t^{ij} < 1$ quantifies that an equally weighted portfolio with only A_i and A_j performs better during time-step t than the current portfolio b_t; such as A_i and A_j perform worst for $\Theta_t^{ij} > 1$ than portfolio b_t (Dochow, 2016).

For each time-step, the matrix $\mathbf{A}_t = [a_t^{ij}]$ is given by

$$\mathbf{A}_t = \begin{bmatrix} 1 - \sum_{\tau=1}^{t} \Theta_\tau^{11} & \dots & -\sum_{\tau=1}^{t} \Theta_\tau^{1m} \\ \vdots & \ddots & \vdots \\ -\sum_{\tau=1}^{t} \Theta_\tau^{m1} & \dots & 1 - \sum_{\tau=1}^{t} \Theta_\tau^{mm} \end{bmatrix} \quad (4\text{-}4)$$

Observe that $a_t^{ij} = 1 - \sum_\tau^t \Theta_\tau^{ij}$ only on the diagonal cells of the matrix $(i = j)$ and $a_t^{ij} = -\sum_\tau^t \Theta_\tau^{ij}$ on all the non-diagonal cells of the matrix $(i \neq j)$. Let $\mathbf{A}_t^{-1}$ denote the inverse of $\mathbf{A}_t$, where the cells are expressed as $\bar{a}_t^{ij}$.

The vector $\mathbf{o}_t$ combines the first and second order information

$$\mathbf{o}_t = \begin{bmatrix} \delta(1 + \frac{1}{\beta}) \sum_{j=1}^{m} \bar{a}_t^{1j} \sum_{\tau=1}^{t} \Theta_\tau^{j} \\ \vdots \\ \delta(1 + \frac{1}{\beta}) \sum_{j=1}^{m} \bar{a}_t^{mj} \sum_{\tau=1}^{t} \Theta_\tau^{j} \end{bmatrix} \quad (4\text{-}5)$$

where one component of $\mathbf{o_t}$ is denoted as o_{it}, with $i = 1, \dots, m$. The allocation for time-step $t + 1$ is defined as

$$\mathbf{b}_{(t+1)}^{\text{ONS}} = \underset{\mathfrak{b} \in \mathfrak{B}_m}{\arg\min} \quad (\mathbf{o}_t - \mathfrak{b})^T \mathbf{A}_t (\mathbf{o}_t - \mathfrak{b}) \quad (4\text{-}6)$$

For solving (4-6), we use the algorithm and Python code provided by Kraft (1994) through the SLSQP package[1].

4.2 Online Moving Average Mean Reversion Algorithm

This algorithm exploits a price prediction by the moving average and choose assets by the mean reversion assumption (Li & Hoi, 2012), buying an asset when its price chance forecast is *low*, and selling when the forecast is *high*.

The moving average for this algorithm can be computed in different ways. Besides that, the *simple moving average* is chosen according to Dochow (2016), that is, given a window size w,

$$\text{MA}_{ti}^{w} = \frac{\sum_{\tau=t-w+1}^{t} q_{\tau i}}{w} \quad (4\text{-}7)$$

and is combined with the current price by the *moving average reversion*, given by

$$\tilde{x}_{ti}^{w} = \frac{\text{MA}_{ti}^{w}}{q_{ti}} \quad (4\text{-}8)$$

[1] `https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html`

at the end of trading period t. The variable $\tilde{x}_{ti}^{w}$ quantifies whether the current price of A_i is greater, with $\tilde{x}_{ti}^{w} < 1$, or lower, when $\tilde{x}_{ti}^{w} > 1$, than the current moving average. The market average of a step t can be obtained by

$$\bar{x}_t^w = \frac{\sum_{i=1}^{m} \tilde{x}_{ti}^{w}}{m}. \tag{4-9}$$

The subsequent step is determine λ_t, i.e. the Lagrangian multiplier,

$$\lambda_t = \max\left\{0, \frac{\epsilon - \sum_{i=1}^{m} b_{ti}\tilde{x}_{ti}^{w}}{\sum_{i=1}^{m}(\tilde{x}_{ti}^{w} - \bar{x}_t^w)^2}\right\} \tag{4-10}$$

where if $\sum_{i=1}^{m}(\tilde{x}_{ti}^{w} - \bar{x}_t^w)^2 = 0$, then $\lambda_t = 0$, and ϵ is the mean reversion level.

The allocation vector is obtained by

$$\mathbf{b}_{t+1} = \mathbf{b}_t + \lambda_t(\tilde{x}_{ti}^{w} - \bar{x}_t^w) \tag{4-11}$$

due to some negative values, the result of Equation (4-11) requires a projection onto the simplex $\mathfrak{B}_m$, the algorithm can be found in Dochow (2016, pag. 88).

5
Online Multilayer Perceptron with Side Information

The portfolio selection process (Markowitz, 1952) can be divided into two subtasks. The first one is price level forecasting, that we call asset price prediction problem (APP). The second subtask is the portfolio selection problem (PSP). At each time step t, APP asks for a prediction $\rho_i(t+1)$, related to the asset i rate of return $r_i(t+1)$. On the other hand, PSP asks for an allocation vector $\mathbf{b}(t) = (b_1(t), \ldots, b_m(t))$ that distributes the available wealth through the assets at the beginning of each trading period t. Hence, $W_0 b_i(1)$ gives the wealth allocated to asset i at the beginning of step 1. The model ensures that all wealth is allocated, since $\sum_{i=1}^{m} b_i(t) = 1$ for all $t \in [1, T]$.

Now, let $\boldsymbol{\rho}(t{+}1) = (\rho_1(t{+}1), \ldots, \rho_m(t{+}1))$ represent the class levels of the rate of return vector $\mathbf{r}(t+1)$. To illustrate, consider 3 class levels, where $\rho_i(t)$ indicates that the price of asset i goes *up* (1), *down* (-1), or stays *equal* (0) at time t. We want that PSP finds an allocation with the given side information $\boldsymbol{\rho}(t+1)$. The idea is to see the problem as a classification task, where an algorithm returns a probability vector $\hat{\boldsymbol{b}}(t)$ of the *best* allocation proportion $\boldsymbol{b}^*(t)$ from a benchmark algorithm. The higher the value of $\hat{b}_i$, the higher the

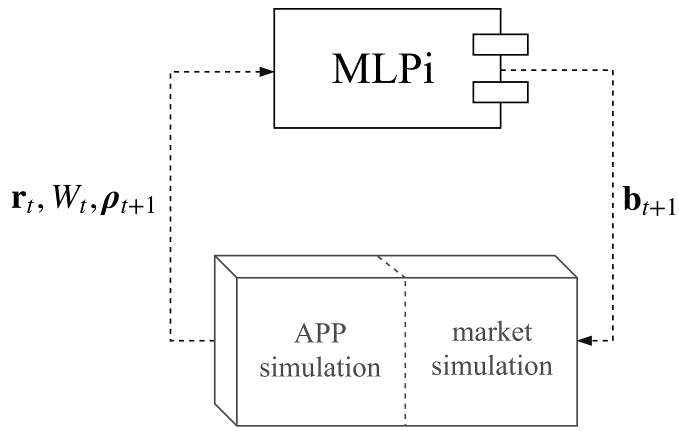

Figure 5.1: Overview of the online multilayer perceptron with side information (MLPi) algorithm. APP is randomly simulated, providing $\boldsymbol{\rho}(t{+}1)$ with 7 levels of accuracy. The market simulation embraces historical data from B3 without transaction costs.

chance that asset i is the *best asset*, for a given time interval.

We propose the online multilayer perceptron with side information (MLPi) for solving the PSP. Basically, we aim to use a future price classifier given by an external expert, which feeds another model with asset price information to make the decision on how much to invest for each portfolio asset. An overview is available in Figure 5.1.

5.1 MLPi Architecture

The architecture of MLPi is a multilayer perceptron (MLP), with 8 layers. The model receives the future price classification vector $\rho(t)$ as input and returns the allocation proportion $\hat{b}_t$. Equations (5-1) identify all the layers of this nonlinear functions,

$$Y_0 = \alpha(\rho w_0 + c_0) \tag{5-1a}$$

$$Y_1 = \alpha(Y_0 w_1 + c_1) \tag{5-1b}$$

$$\dots \tag{5-1c}$$

$$\hat{b} = Y = \sigma(Y_6 w_7 + c_7) \tag{5-1d}$$

therefore, $\hat{b}$ assumes the values returned by the function Y, the output layer; α are the activation functions for the hidden layers, and σ is the softmax function or normalized exponential function, given by Equation (5-2),

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{i=1}^{m} e^{z_i}} \qquad \forall\ j = 1, \dots, m \tag{5-2}$$

so the model can retrieve the probability of corrected classification where $\sum_{j=1}^{m} \sigma(\mathbf{z})_j = 1$ and $\sigma(\mathbf{z})_j \in (0, 1]$.

The α activation function used is Rectified Linear Unit (ReLU) presented by Nair & Hinton (2010) that has the output of $y = \max(x, 0)$, with a little modification as proposed in Krizhevsky (2010), expressed as ReLU6, with the output $y = \min(\max(x, 0), 6)$. Accordingly to the authors this modification encourages the model to learn sparse features earlier. Figure 5.2 has a graph representation of the architecture, identifying that each activation function is performed element-wise. The edges of the graphs represent the weights w of each associated input-output neuron. The size of each weights are listed in Table 5.1. We call the algorithm MLPi, as we are using information from APP for asset allocation decisions, the correct classification ρ^* is only used when it becomes available for updating parameters of Y.

Layer	**Parameter**	**Input**	**Output**	**Size**
0	w_0	6	600	$3,600$
1	w_1	600	400	$240,000$
2	w_2	400	180	$72,000$
3	w_3	180	120	$21,600$
4	w_4	120	80	$9,600$
5	w_5	80	60	$4,800$
6	w_6	60	30	$1,800$
7	w_7	30	6	180
Total				$353,580$

Table 5.1: Parameter size for a 8 layers illustrative network.

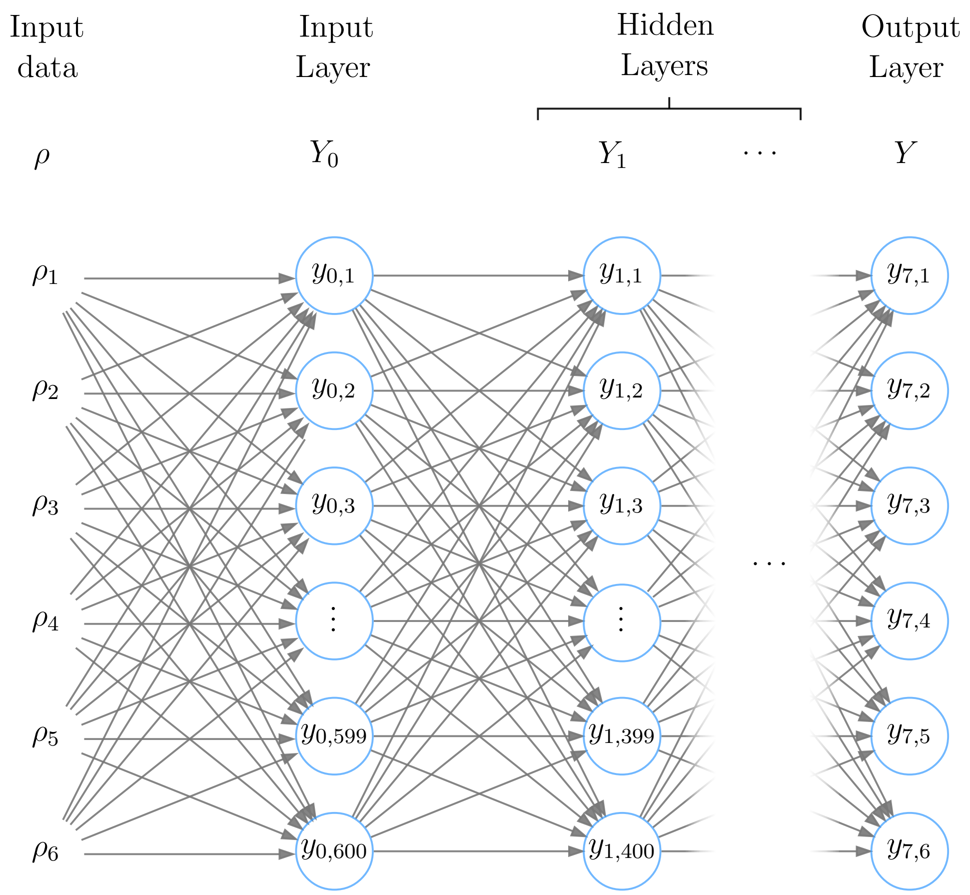

Figure 5.2: Graph representation of the architecture of the proposed MLPi algorithm. Each neuron $y_{k,i}$ is the result of the activation of the weighted input of asset i, of the k-th layer, with $m = 5 + 1$ assets.

5.2 MLPi Loss

One may think that the simple maximisation of terminal wealth would be a sufficient objective of this task. But our model receives a prediction from a ternary classifier, and therefore, it should be safer to use information from a benchmark algorithm.

The objective of a deep learning task is to minimise a *loss* function (Goodfellow et al., 2016). Three main loss functions are initially tested: (i) maximisation of terminal wealth; (ii) minimisation of wealth absolute difference; and (iii) minimisation of regret.

The loss function (i) is the terminal wealth, given by

$$W_\tau = \prod_{t=0}^{\tau} \sum_{i=1}^{m} r_{ti} \hat{b}_{ti} \tag{5-3}$$

where τ is a subset interval, $\tau \subset \{1, \ldots, T\}$. In our initial validation, the network could not learn, and has terminal wealth worst than optimal online algorithm with side information (OPTi).

The loss function (ii) is the average absolute difference between the OPTi and MLPi, based on Cesa-Bianchi et al. (1997).

$$AD_\tau = \frac{\sum_{t=0}^{\tau} \left| \sum_{i=1}^{m} r_{ti} b^* - \sum_{i=1}^{m} r_{ti} \hat{b} \right|}{\tau} \tag{5-4}$$

The Equation (5-4) gives us an interesting approach, the absolute operator makes possible for the learning algorithm to escape from the OPTi in compensation of some positive rewards, when $\sum r_{ti} b^* < \sum r_{ti} \hat{b}$. But when $\sum r_{ti} b^* > \sum r_{ti} \hat{b}$ with a same difference, the minimisation has the same value. Although it seems convenient to inform the network when this difference on wealth is positive and when is negative, i.e. when our algorithm performs better than the OPTi, and when it does not. Indeed, this loss function could learn over increase t, but with results very close to the OPTi.

The loss function (iii) is the regret with respect to OPTi and presents outstanding results for initial validation. So the proposed objective is to minimise the difference of the periodic logarithmic percentage yield given by Equation (5-5). Since the logarithmic and exponential function are monotonic for any function $f : \mathbb{R}_{>0} \to \mathbb{R}$, the *loss* function of Equation (5-5) minimise the wealth difference too.

$$
\begin{aligned}
loss =& \ln\left(\prod_{t=1}^{\tau}\sum_{i=1}^{m} r_{ti}b^*_{ti}\right)^{\frac{1}{\tau}} - \ln\left(\prod_{t=1}^{\tau}\sum_{i=1}^{m} r_{ti}\hat{b}_{ti}\right)^{\frac{1}{\tau}} && \text{(5-5a)}\\
=& \frac{1}{\tau}\sum_{t=1}^{\tau}\ln\left(\sum_{i=1}^{m} r_{ti}b^*_{ti}\right) - \frac{1}{\tau}\sum_{t=1}^{\tau}\ln\left(\sum_{i=1}^{m} r_{ti}\hat{b}_{ti}\right) && \text{(5-5b)}\\
=& \frac{1}{\tau}\sum_{t=1}^{\tau}\ln\left(\frac{\sum_{i=1}^{m} r_{ti}b^*_{ti}}{\sum_{i=1}^{m} r_{ti}\hat{b}_{ti}}\right) && \text{(5-5c)}
\end{aligned}
$$

Equation (5-5) is very close to the Equation (5-4). A main difference stands, Equation (5-4) aims to learn OPTi, and the goal here, is do better than OPTi. Giving the model the freedom to learn some relevant information from the dataset.

To give a visual demonstration, lets consider a hypothetical portfolio Φ with 3 assets, with an allocation vector given by $\mathbf{b}^\Phi = (b_1^\Phi, b_2^\Phi, b_3^\Phi)$. All possible solutions for $\mathbf{b}^\Phi$ are given by Figure 5.3. Where the blue dots gives all the $2^3 - 1$ possible solutions for each step of the OPTi algorithm, the orange area represents all possible solutions for each step of MLPi. With Equation (5-5) the model has an incentive to experiment regions outside the OPTi solution. This raises a comparative measure: the distance between the MLPi solution and its benchmark. As of the solutions b are in the simplex, the *cosine distance* is used, as it will always belong to the interval $[0, 1]$.

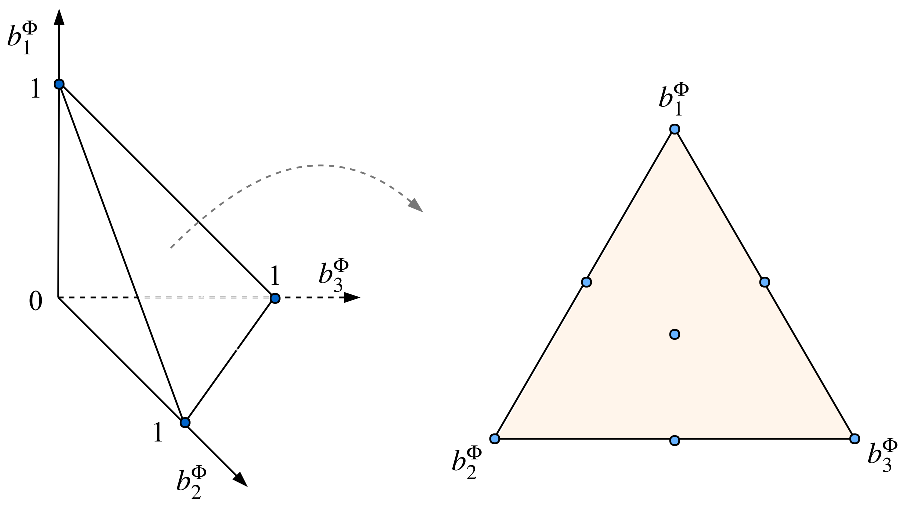

Figure 5.3: Hypothetical Portfolio Φ, with 3 assets, solutions are represented by a standard 2-simplex. Where the standard n-simplex is given by $\Delta^n = \{(b_0, \ldots, b_n) \in \mathbb{R}^{n+1} \mid \sum_{i=0}^{n} b_i = 1 \text{ and } b_i \geq 0 \text{ for all } i\}$.

5.3
MLPi Learning

Two algorithms are provided for MLPi. One that performs an initial training with historical data, and output $\hat{Y}$ with the updated parameters. The other uses the fresh information from the market to perform the predicted allocation proportion $\hat{\mathbf{b}}(t+1) = \hat{Y}(\boldsymbol{\rho}(t+1))$ After the model receives the corrected information, it executes the first algorithm to update the parameters.

The Algorithm 5.1 gives the estimated function $\hat{Y}(\rho)$ based on the training dataset to retrieve the allocation proportion $\hat{\mathbf{b}}(t+1)$ for a given $\hat{\boldsymbol{\rho}}(t+1)$. This algorithm can be executed online, so the model Y can be updated online, to absorb the fresh information of the market.

Algorithm 5.1: Multi-layer perceptron algorithm (training)

```
input : r_train, a matrix T × m with assets returns
        μ, learning rate
        epochs, number of loops over the dataset
        τ, the batch size
        φ, regularisation probability of ρ_train
output: Ŷ(ρ), the estimated function for portfolio selection
Initialisation
  | Y ← last layer from Equation (5-1)
  | w_l ~ N(0, 0.1) initial weights parameters of Y for all layer l
  | c_l ← 0 initial bias parameters of Y for all layer l
  | loss ← (1/τ) Σ_{t=1}^{τ} ln( Σ_{i=1}^{m} r_ti b*_ti / Σ_{i=1}^{m} r_ti b̂_ti ), Equation (5-5)
  | ρ_train ← simulate a prediction over r_train with probability φ
end
begin
  | for each epoch in epochs do
  |   | ρ_τ ← get next τ elements from ρ_train
  |   | b* = run OPTi with ρ_τ
  |   | b̂ ← solution of loss minimisation with parameters ρ_τ, b*
  |   |   using AdamOptimiser(μ)
  |   | update all weights and biases of Y with respect to loss
  | end
end
```

where $loss \leftarrow \frac{1}{\tau}\sum_{t=1}^{\tau} \ln\left(\frac{\sum_{i=1}^{m} r_{ti} b^*_{ti}}{\sum_{i=1}^{m} r_{ti}\hat{b}_{ti}}\right)$.

For minimising the *loss* function and update the parameters of Y, given by the Equation (5-1), a stochastic optimisation algorithm is needed. We chose the *Adam* optimiser (Kingma & Ba, 2014). The authors experiments show better results than the stochastic gradient descent (SGD), and for our initial validations too. A prediction for an allocation b can be obtained by the Algorithm 5.2.

Algorithm 5.2: Online multi-layer perceptron algorithm for portfolio selection

---

input : r_t, a vector of assets returns at the end of current step t
$\hat{\rho}_{t+1}$, a vector of predicted classification price for each asset
$\hat{Y}(\rho)$, the estimated function from Algorithm 5.1

output: $\hat{b}_t$, a vector with size m of the estimated allocation proportion to be executed at the end of step t

Initialisation
  $loss \leftarrow$ loss function from Equation (5-5)
end
begin
  run Algorithm 5.1 with new vector of assets returns r_t
  $b_t \leftarrow \hat{Y}(\rho_{t+1})$
end

5.4 Auxiliary Systems

In this work, we assume APP is given by an external source, like an expert. In the next section we propose a simulation environment for evaluation, where we use an expert with different accuracy levels. For an example, suppose that an expert receives some information at t and make a prediction for step $t+1$, returning a vector $\boldsymbol{\rho}(t+1)$ identifying the direction of the future price of each asset. The PSP receives this information and returns a vector $\mathbf{b}(t+1)$, the allocation strategy to be executed at the beginning of step $t+1$. The market simulation just execute that decision returning the total wealth achieved by the end of step $t+1$. This algorithm can be applied in an online execution, at each time-step $t \in [1, T]$.

If we could find a *price predictor* that returns with 100% of accuracy the next price label for each asset, it's clear that the best solution is either invest 100% on the best asset, or split the wealth equally among the best assets for each step. Because it is very unlikely to have that level of accuracy, we can infer that for any other accuracy, we can still invest more resource on the predicted best assets, but to prevent errors from the APP model we should invest some part of the resource in the others assets too.

That is what we expect achieving using a learning algorithm: it can learn from the dataset assets that are more likely to win in the next step, so instead of splitting the amount in each *best asset*, the model can invest slightly more in one of the *best assets*. It's like the model learns to predict the asset price too.

5.4.1
Asset Price Predictor

The purpose of this simulation is to validate our model using a predictor with a given *accuracy* p, where p is defined as the *accuracy score* or the *Jaccard similarity coefficient*,

$$accuracy(\rho, \hat{\rho}) = \frac{1}{n} \sum_{k=1}^{n} \mathbf{1}(\rho_k = \hat{\rho}_k) \tag{5-6}$$

it is the proportion of true positives, over all n samples. And ρ is the true classification of asset prices defined in Section 2.2.

For this experiment, the *accuracy* p will be constant for all assets in all portfolios simulations. This means that for any asset on any simulation, for $n = 1000$ and $p = 0.4$, we have 400 correct price classifications, and 600 incorrect classifications. All incorrect classifications are randomly selected by a uniform distribution of non-true classifications.

This approach simplifies the model, it is clear that if assets have different accuracy values the OPTi could be improved using this information, and have a better solution. Solutions for this problem can be found using dynamic stochastic programming, for example, see Samuelson (1969), Dumas & Luciano (1991), Mulvey & Vladimirou (1992) and Le Ny (2009). The reinforcement learning algorithms identified in Chapter 1 have foundations based on the dynamic stochastic programming theory.

5.4.2
Market Simulation

To evaluate our model, we use a simulated environment with real market data. A *bootstrap* simulation approach, proposed by Efron & Tibshirani (1986) is used to validate the predictors, and give the possibility of estimating some statistics without knowing the distribution.

Our empirical experiments are detailed in the follow steps:

Step 1: Choose N random portfolios with replacement, each one with m assets available for each algorithm in the trading period. For each portfolio, the samples are taken *without* replacement. So, one asset can appear in multiple portfolios. Make sure that each portfolio is unique and that for all portfolios each asset is unique.

Step 2: Generate B predictors, each one with a fixed accuracy p.

Step 3: Train the proposed algorithm MLPi on the training dataset with the Algorithm 5.1.

Step 4: Compute the wealth for each time-step t over all B *bootstrap* simulations for each algorithm with side information (OPTi and MLPi) with the validation dataset. Take the *average* wealth simulation for evaluation metrics.

Step 5: Compute the wealth for each time-step t over all other algorithms (online moving average mean reversion algorithm (OLMAR), online Newton step algorithm (ONS), constant rebalancing portfolio (CRP), best offline constant rebalancing portfolio (BCRP) and best offline buy-and-hold (BBH)) using the validation dataset.

Step 6: For each different accuracy p, go back to Step 2.

The results presented here were obtained by the following parameters: $N = 200$ random portfolios, with $m = 5 + 1$ assets and with $B = 100$ predictors simulations. The experiments were made with 7 accuracy levels, $[0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75]$.

As described in Efron & Tibshirani (1986), the bootstrap confidence interval can be computed over the bootstrap simulations.

Take $\bar{x}_t$ as the sample mean of step t of all the B simulations. We want to know how much the distribution of $\bar{x}_t$ varies around μ_t, i.e. we want to know the distribution of $\delta = \bar{x}_t - \mu_t$. Since we don't have the value of μ_t, we can use the bootstrap simulations

$$\delta_t^* = \bar{x}_t^* - \bar{x}_t \qquad (5\text{-}7)$$

where $\bar{x}_t^*$ denote the mean obtained by a bootstrap simulation. Therefore, by the law of large numbers the distribution of δ_t^* can be estimated with high precision.

To compute δ_t^* take the $\bar{x}_t^*$ obtained by each simulation and sort them in ascending order. For a 95% of confidence interval, take the 2.5th percentile, that is, the 2.5th element of our 100 simulations; and for the 97.5th percentile, take the 97.5th element. As the desired percentile lies between two elements, a linear interpolation is used, $\delta_{t.025}^* = \delta_{t.02}^* + 0.5(\delta_{t.03}^* - \delta_{t.02}^*)$ And the confidence interval is giving by $[\bar{x}_t - \delta_{t.025}^*, \bar{x}_t - \delta_{t.975}^*]$.

A sample of a bootstrap simulation of the predictors for one portfolio is plotted in Figure 5.4.

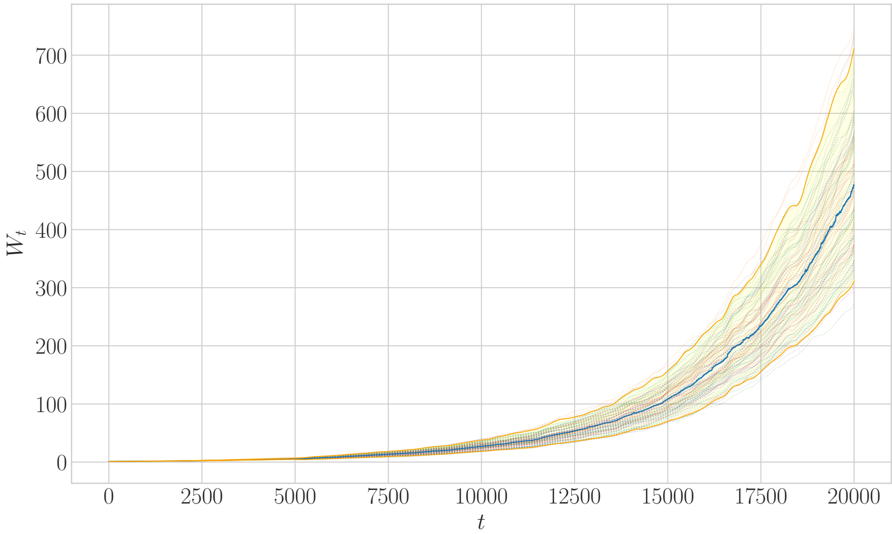

Figure 5.4: Wealth simulation of MLPi for all predictors ($B = 100$) with accuracy $p = 0.50$ for portfolio 144, with assets CSNA3, SBSP3, POMO4, PETR3, VALE3 and \$. The solid blue line is the mean, the solid yellow lines are upper and lower bounds of bootstrap confidence intervals (yellow filled area) with significance level of 95%.

5.4.3 Metric Evaluation

To evaluate a trained portfolio the cosine similarity is chosen as an evaluation metric. The objective here is to define a sufficient value that verifies if the trained MLPi, with any portfolio Φ, is *good enough* to be accepted for an allocation decision. The idea is to decide when the MLPi should be chosen for an allocation decision.

The proposed algorithm to choose the cosine similarity limit is the expectation maximisation (EM), proposed by Dempster et al. (1977), it has a wide range of applications, but the use case here is for clustering the portfolios by their respective cosine similarity values.

EM as a clustering algorithm is an iterative method, similar to *k-means*, that aims to separate the data set into k groups represented by distinct normal distributions, each with parameters (μ_k, Σ_k). With this algorithm is possible to identify groups in elliptical formats, where an ellipsis can take precedence over other when they collide based on their weights (an example is specified in the Figure 6.3).

The algorithm has two steps which are executed by a number of iterations. The *expectation* stage assigns points to its closest representatives (distributions). The distance between a point and a representative is equal to 1 minus the probability of the point belonging to this representative (defined by the normal distribution). The *maximisation* step recalculates the (μ_k, Σ_k) parameters of each representative.

Assuming there is a dataset $D = \{x_1, \ldots, x_n\}$, where x_i is a d-dimensional vector. In addition, it is also assumed that the points are randomly generated i.i.d. from a density function $p(x)$.

Each group is then defined by the average of its elements and the covariance matrix $\{\mu_k, \sum_k\}$. Therefore, the probability of x belonging to a given group k can be obtained by:

$$P(x_k|\theta_k) = \frac{1}{(2\pi)^{\frac{d}{2}}|\sum_k|^{\frac{1}{2}}} e^{\frac{-1}{2}(x-\mu_k)^t(x-\mu_k)} \tag{5-8}$$

where, $\theta_k = \{\mu_k, \sum_k\}$.

The algorithm is divided into two steps:

E-Step: Calculate the values of the matrix $Q_{n \times k}$ for all points x and all clusters k. Note that for each given point x_i it is valid that: $\sum_{k=1}^{K} w_{ik} = 1$, i.e, each row of the matrix Q of dimension $N \times K$ has a sum equal to 1.

M-Step: Given the probabilities calculated in the previous step, we can now calculate the new parameters: mean and covariance matrix for each group.

$$\mu_k^{new} = \frac{1}{\sum_{i=1}^{N} w_{ik}} \sum_{i=1}^{N} w_{ik} x_i \qquad 1 \leq k \leq K \tag{5-9}$$

The new covariance matrix is given by:

$$\Sigma_k^{new} = \frac{1}{\sum_{i=1}^{N} w_{ik}} \sum_{i=1}^{N} w_{ik}(x_i - \mu_k^{new})^t(x_i - \mu_k^{new}) \qquad 1 \leq k \leq K \tag{5-10}$$

The initialisation of the algorithm can be performed in two ways: with an initial set of parameters and then driven to an **E-Step**; or with an initial set of probabilities of each point belong to each group and then do a first **M-Step**. Initialisation can also be set using some heuristics, such as using the output of *k-means* as input to the method (which is applied in this case).

6
Experiments

To better evaluate MLPi we simulate the APP model, as an expert, so we can verify when a good prediction accuracy is enough to retrieve better wealth against a predefined benchmark.

6.1
B3 Dataset

To empirically validate MLPi, we run a series of stochastic simulations. The first simulation, is to select the assets that will be available in our market simulator. For this, we need an information source that gives the desired granularity of the dataset and a sufficient historical data.

The data is collected from B3, the fusion of BM&FBOVESPA S.A. Securities, Commodities & Futures Exchange (BM&FBOVESPA) with Cetip S.A. Organised Markets. BM&FBOVESPA is the only stock exchange in Brazil. The source of data is from Thomson Reuters[1] and the step size is in minutes, where the price is adjusted for any split, and all dividends and bonus paid are reinvested. The data is collected from January 2, 2008, to May 4, 2018. The total of stocks available to the market are the 30 most negotiated of the last 3 years. Because of the short trading period (one minute), is imperative to select a list of assets that have enough tradings. For all assets to have data for the same time interval, when no trading occurs in one minute, the last minute price is used. If a stock has low tradings, this means too much *neutral* ($\rho = 0$) price change classification in the dataset. All considered symbols, the *market simulator assets* are listed in Table 6.1.

The Table 6.2 have a sample of the aggregated raw dataset of trades by minute. Using the *close price* as the trading price of our market simulator, the Table 6.3 represents a sample of a portfolio dataset composed by the assets

[1] The Thomson Reuters dataset is available as a paid service by DataScope Select (DSS), therefore all dataset content have restricted access. Samples information available in Table 6.2 and 6.3 are extracted from a BM&FBOVESPA service, that keep the historical information daily updated from the past 2 years, available at `ftp://ftp.bmf.com.br/MarketData/Bovespa-Vista/` accessed in May 6, 2018 6 a.m. GMT, although this dataset is not recommended for price historical analysis, because there is no price normalisation (for splits, bonus and dividends).

PETR4	BBAS3	ITSA4	ITUB4	GGBR4	BBDC4
ABEV3	CMIG4	KROT3	CIEL3	JBSS3	USIM5
PETR3	BBSE3	CSNA3	GOAU4	CCRO3	BRFS3
VALE3	LAME4	FIBR3	ECOR3	BRML3	EMBR3
MRVE3	HYPE3	LREN3	KLBN11	BRAP4	VIVT4

Table 6.1: Assets available for the market simulator

PETR4, VALE3, ABEV3.[2]

time-step	**symbol**	**open**	**close**	**min**	**max**	**trades**	**quantity**	**volume**
2015-01-02 10:07	PETR4	9.91	9.93	9.91	9.94	40	38,600	383,147
2015-01-02 10:08	PETR4	9.93	9.93	9.92	9.94	21	19,700	195,543
2015-01-02 10:09	PETR4	9.92	9.91	9.91	9.93	107	67,500	669,595
2015-01-02 10:10	PETR4	9.91	9.89	9.88	9.92	106	103,600	1,025,180
2015-01-02 10:11	PETR4	9.88	9.89	9.88	9.90	33	84,500	835,901

Table 6.2: Sample aggregate raw dataset.

time-step	**PETR4**	**VALE3**	**ABEV3**
2015-01-02 10:07	9.93	21.69	16.18
2015-01-02 10:08	9.93	21.74	16.16
2015-01-02 10:09	9.91	21.69	16.20
2015-01-02 10:10	9.89	21.69	16.18
2015-01-02 10:11	9.89	21.70	16.20

Table 6.3: Sample portfolio dataset

There are some technical details of a stock exchange that withhold an asset to perform trades over a period, like auction, or circuit break and also happens on some specific intervals of the day. The current stock market trading period *opens* at 10:00 a.m. and *closes* at 16:55 p.m., the after-market is despised. But since the dataset have data from more than 10 years, things were different in the past, for this, the *open price* are defined as the first hour of any trade occurred for an asset, and the *close price* are the hour and minute of the last trade. To make all asset prices available between all minutes of the trading period and all collected data, 3 actions are taken:

[2] Stocks of the respectively companies, PETROBRAS S.A., VALE S.A. e AMBEV S.A.

(i) when there is no trading on a minute time-step, the price of the last available minute is repeated;

(ii) when the first price of any day is not available, the last price of the previous day is used;

(iii) the first day is removed from the dataset.

This gives a final dataset of 1.048.161 minutes of close price for 30 stocks.

6.2 Training and Validation

Our initial results are from the experiments of the training and validation procedure of the proposed algorithm MLPi. We split the dataset in 3 parts: *train* dataset, with 830.953 minutes; *validation* dataset, with 108.604; and *test* dataset, with 108.604. These numbers were chosen to keep the validation and test datasets in equal-size intervals. This means the *train* occurs between January 2, 2008 and April 29, 2016; the *validation* between May 2, 2016 and April 28, 2017; and the *test* between May 1, 2017 and May 3, 2018.

The train process is executed first with the *train* dataset with Algorithm 5.1. The adjustments of hyperparameters are made using the *validation* dataset. Because of the size of the dataset and the quantity of portfolios and simulations, the tuning phase is limited by time. The adjustments gives us the following parameters: a batch size of $\tau = 5.503$; a predictor accuracy $\phi = 0.98$; a learning rate of 0.003; and a dropout regularisation of 0.9.

For evaluation of algorithms, two measures are taken comparing the allocation proportions given by the MLPi and OPTi. The cosine similarity $S_c(b^{OPTi}, b^{MLPi})$, is given by Equation (6-1),

$$S_c(p,q) = \frac{\sum_{i=1}^{m} p_i q_i}{\sqrt{\sum_{i=1}^{m} p_i^2}\sqrt{\sum_{i=1}^{m} q_i^2}} \tag{6-1}$$

and the cross entropy $H(b^{OPTi}, b^{MLPi})$, by Equation (6-2).

$$H(p,q) = -\sum_{i=1}^{m} p_i \log q_i \tag{6-2}$$

both measures are very common in classification problems.

Comparing both plots presented in Figure 6.1 and 6.2, it seems that the cosine similarity has a better fit-to-purpose, that is, find a evaluation metric that can reject a portfolio for running with MLPi. Therefore, we argue that the cosine similarity is a more stable metric for evaluating our proposed model.

We are now facing a problem of when we should choose our proposed model over the naive OPTi solution. So, we want a limit value of the cosine similarity to make the decision of not using our proposed model if that limit in the train phase is lower.

In a practical environment, an investor may wish to not invest in those portfolios with lower performance. But even that, we want a strategy for choosing those best portfolios. So, in our test environment, beside selecting all 200 portfolios, we run 2 investors alternatives for MLPi:

(i) that uses the best performed portfolio in our train dataset;

(ii) select portfolios above a *cosine similarity limit*.

For the second strategy, we want the x-axis *limit* that split Figure 6.2 at least in two groups, the left, and the right side of *limit*. Take a look at Figure 6.2 it appear to have a 3, maybe 4 clusters of points. The idea is split those points in 3 classes, and take the cosine similarity limits of the right group. Three main justifications we have for selecting only this group, the points are not so dispersed; it contains the best portfolios, in terms of terminal wealth; they have more similarity with the OPTi solution.

One may think that would be useful to remove portfolios from the right side too, adding a maximum limit for cosine similarity. But a cosine similarity that approaches to 1 identifies that the model MLPi fit almost equally to OPTi. Because of this, is expected that the mean wealth increases when we move away from the maximum similarity, until it reaches a maximum value and start to decrease.

In Section 5.4.3 we presented the EM heuristic for defining this limit to disapprove a trained portfolio. The Figure 6.3 has the result of the EM for the *train* and *validation* set. The estimation of clusters were made with only the cosine similarity (x-axis), the centroids of each group is represented by "$\times$".

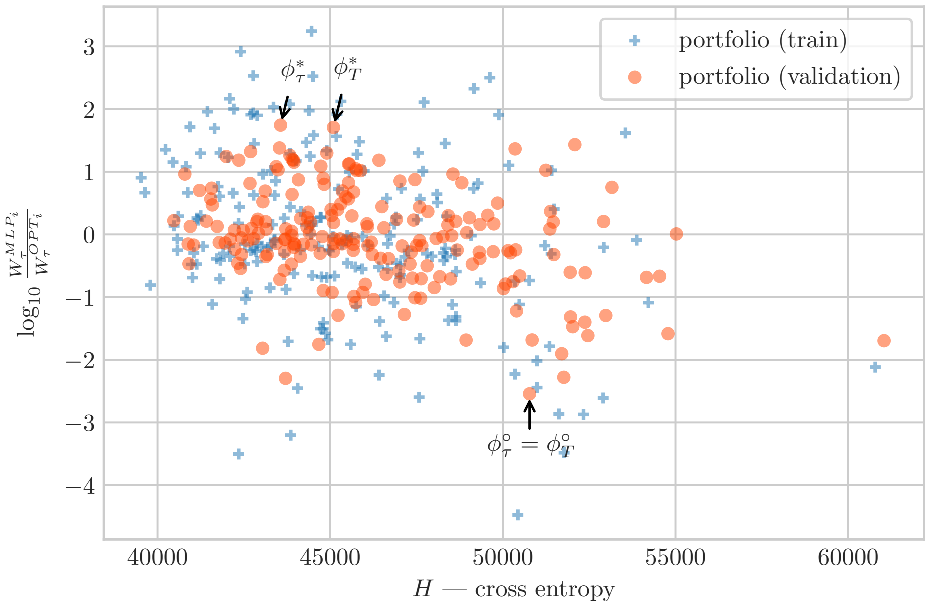

Figure 6.1: Cross-entropy compared with the logarithmic terminal wealth difference achieved by MLPi over the benchmark OPTi, the performance measure, $\log_1 0 \left(\frac{W_T^{MLPi}}{W_T^{OPTi}}\right)$ during training and validation datasets of all $N = 200$ random portfolios and $p = 0.75$ in comparison with the *cross entropy*. Since is a logarithmic difference, is visible that the minimum performance is bigger in magnitude than the maximum performance of MLPi. The Φ^* and Φ° are

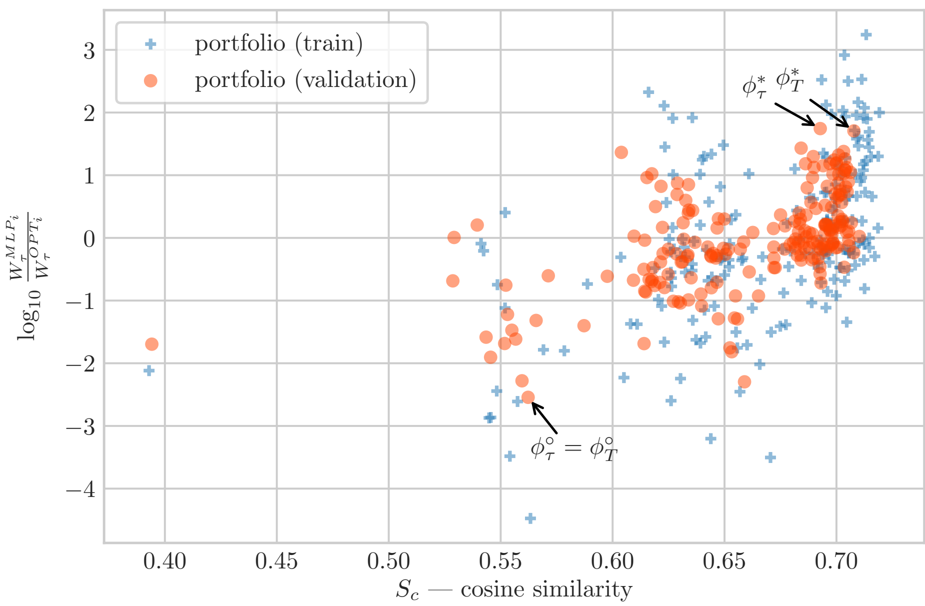

Figure 6.2: Cosine similarity compared with the logarithmic terminal wealth difference achieved by MLPi over the benchmark OPTi, the performance measure in comparison with the *cosine similarity*.

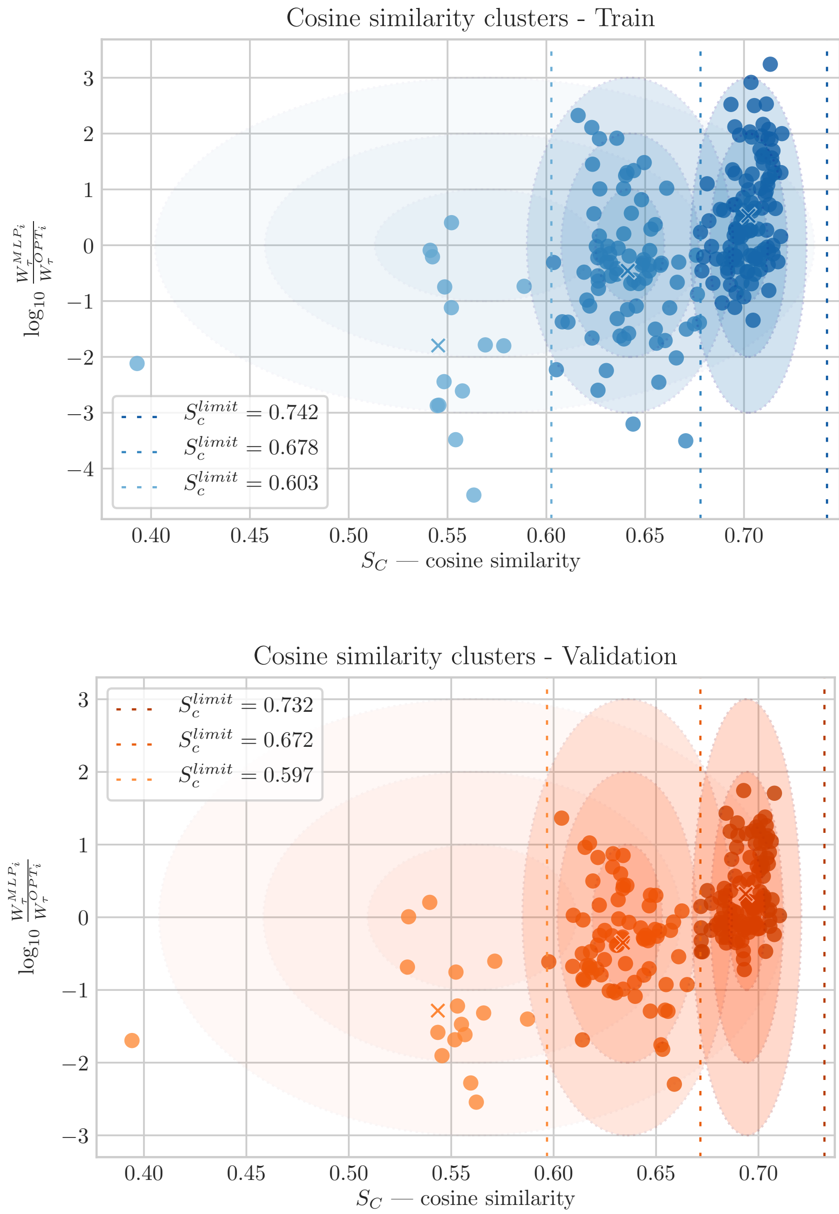

Figure 6.3: The EM algorithm, with 3 clusters selected of the cosine similarity. The information on the test dataset is just for comparison, and validation of this heuristic. The limits to be used is the limit of the *train* phase, in blue, where the selected limits, are the vertical lines splitting the rightmost cluster. With the EM algorithm is possible to find a right limit too, this is, any point at the right side of the rightmost vertical line, would be classified as part of the leftmost cluster.

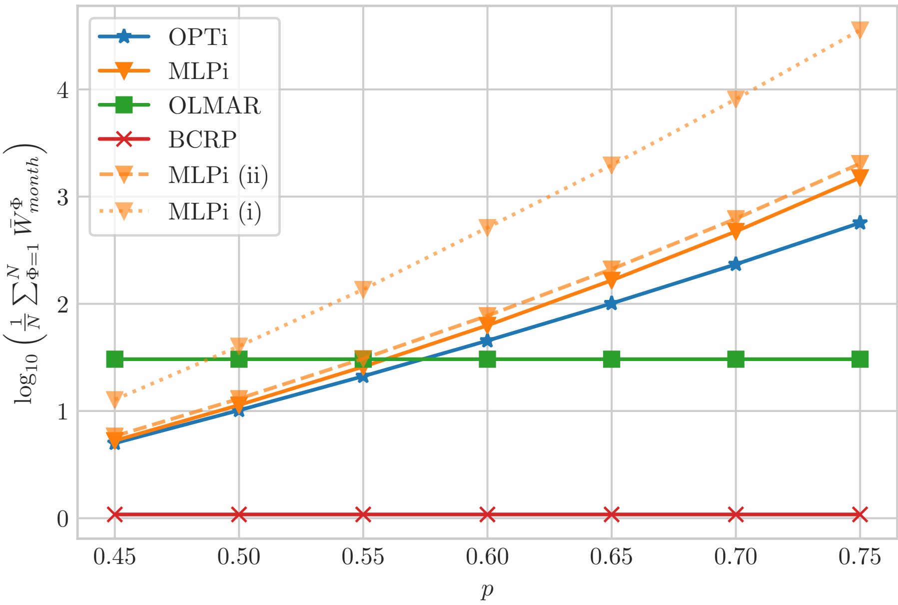

Figure 6.4: Terminal wealth achieved by all algorithms in the *validation* dataset, with initial wealth $W_o = 1$. The MLPi (i) is the best portfolio of the *train* phase. The MLPi (ii) runs with portfolios filtered by the cosine similarity limit of the *train* phase.

6.3 Results

Now that the MLPi is already adjusted and defined, we perform the market simulation with the *test dataset*. For testing, the *train* and *validation* are combined and used as the new *train* dataset, getting a total of 9 years of trading period (939.557 time steps). All metrics defined in Section 3 are presented here. Some algorithms presented in the literature review are omitted, due to poor results.

The Figure 6.5 shows the results in a boxplot format. It is clear that the variance of portfolios wealth comparison increases as the predictor accuracy increases. The portfolios bellow the first quartile (identified by the bottom of each box) are almost the same for all accuracy levels p, 94.3% of equally portfolios. The relation of all these portfolios are available in Table A.3.

The objective of the PSP is the total wealth achieved at the end of step T. To give more samples of a terminal step, the wealth $\bar{W}$ computed is the average wealth achieved at the end of each month (as the test dataset has 1 year, it gives us 12 terminal wealth samples). The Figure 6.6 shows the average wealth of the $N = 200$ selected portfolios, achieved by each portfolio for all considered accuracy levels of the simulated predictor. It is clear that the proposed algorithm beat the benchmark OPTi, and for a prediction accuracy $p \geq 0.50$, the algorithm have better results in comparison with the state of the art literature OLMAR.

Even that MLPi was not tested with a real predictor, a lot of contributions already exist in literature, like Patel et al. (2015), and Chen et al. (2015). That uses side information from news to predict the future classification of the stock price. Besides both works use a binary classification (*up* or *down*), the best accuracy of the predictors is very high, Patel et al. (2015) presents a naive-Bayes (Gaussian process) that perform with 73.3% accuracy, and a naive-Bayes (Multivariate Bernoulli Process) with 90.19%. With this, it is safe to state that our proposed algorithms for portfolio selection can have o good performance with a real stock price prediction model.

The Figure 6.6, risk versus reward gives us the relation of risk and return for each portfolio with the selected algorithms. At a first look the MLPi algorithm could be consider more *risky*, but since the y-axis is in logarithmic scale, the difference of the wealth scale very high. Even that, as an investor recommendation, the metric to compare should be the Sharpe ratio. The Table A.4 gives all log Sharpe ratios comparison of both algorithms. Selecting only portfolios with a Sharpe ratio positive difference, gives us that 70.5% of the total 200 have a higher Sharpe ratio. If only the *approved* portfolios are selected, gives us an approximate 93% of the total 114 approved portfolios.

The execution of the whole experiment is very slow, because the market is simulated for 200 portfolios, with 7 levels of accuracy for each algorithm. The train phase, gives us 1.400 executions of the MLPi (Algorithm 5.1 with 20 epochs) with a total runtime of 48 hours. The test phase, with 100 bootstrap simulations of APP for each portfolio, gives us 140.000 executions of the MLPi (Algorithm 5.2) for each $t = 1, \ldots T$. The execution of the experiment had a total runtime around 65 hours for the test phase. The Table 6.4 shows the average runtime for each algorithm. The MLPi is fast in comparison to the OLMAR and ONS.

	Average Runtime (s)
BCRP	0.5193
OLMAR	44.7463
ONS	152.6816
OPTi	2.6183
MLPi	10.0744

Table 6.4: Comparative on the average runtime in seconds of each algorithm for 1 portfolio (and the average for 100 bootstrap simulations for MLPi and OPTi) with the test dataset.

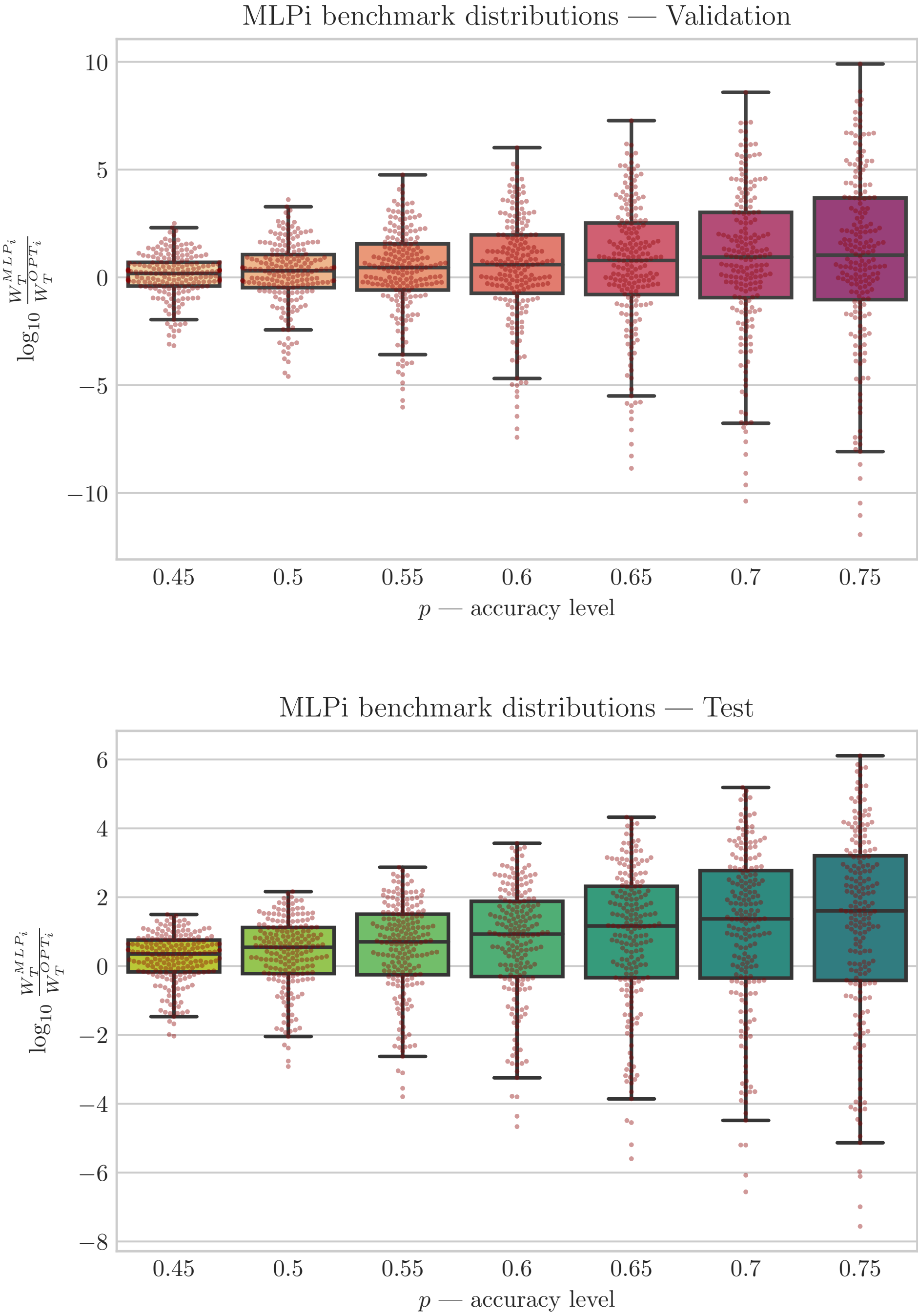

Figure 6.5: Box plot of benchmark comparison (MLPi *vs* OPTi). The middle line of the box is the median, the limits of the box are the first and third quartile, and the extremes (whiskers) are the confidence intervals for a 95% of significance. Each dot represent a portfolio. The dots outside the box plot, are consider *outliers*.

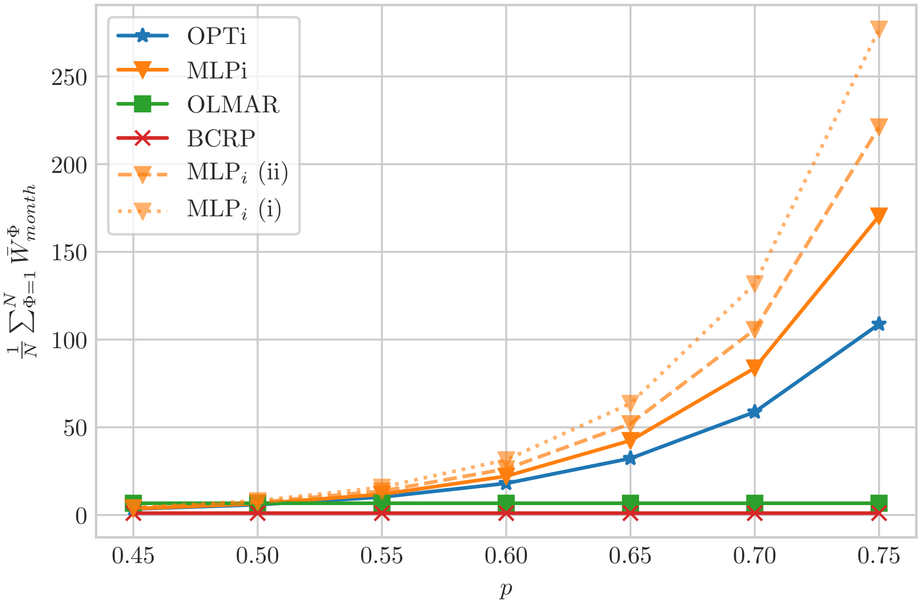

Figure 6.6: Terminal wealth achieved by the algorithms in the *test* dataset, with initial wealth $W_o = 1$. The MLPi (i) is the best portfolio of the *train* phase. The MLPi (ii) the filtered portfolios by the cosine similarity limit of the *train* phase.

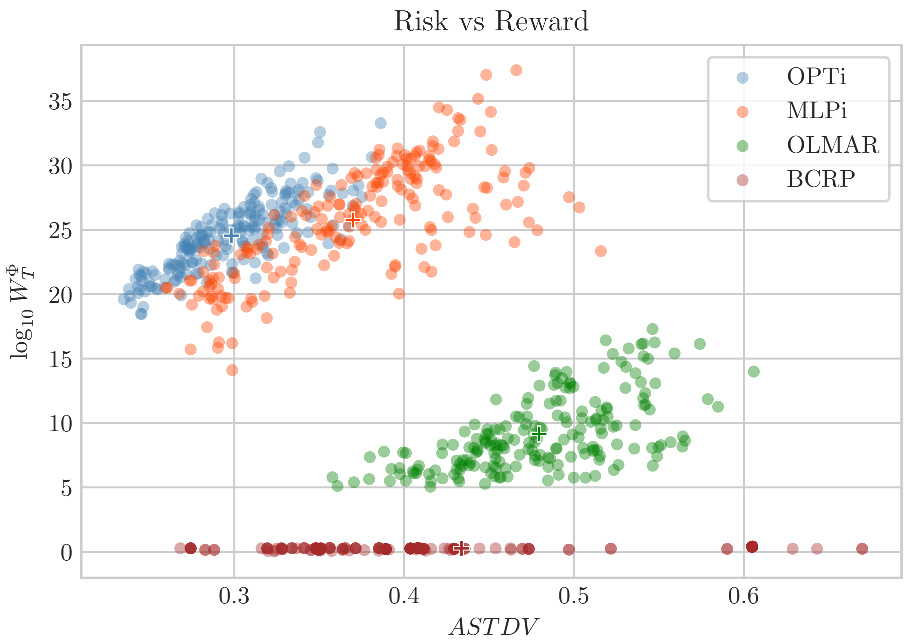

Figure 6.7: Plot of the ASTDV for each portfolio against the logarithm wealth achieved at the end of the test dataset. The cross marker indicates the centroid for each algorithm.

7
Conclusions

In this work, we explored the portfolio selection problem (PSP) with side information using a neural network approach. The main contribution of this work is the algorithm multilayer perceptron with side information (MLPi). The MLPi is compared to the state of the art algorithm of the machine learning community, however, for a predictor validation in different accuracy levels, we run the experiments with a simulated predictor. Because of this, the selected benchmark for the comparative metrics is a naive optimal solution to the problem. Also, we presented an empirical validation in a market simulation with 10 years of historical data. The experiments show that the MLPi can beat the online moving average mean reversion algorithm (OLMAR) – the state-of-the-art – for a predictor with 50% of accuracy. Also, the MLPi outperform the proposed benchmark optimal online algorithm with side information (OPTi) on average, in all tested accuracy levels.

Our recommendation as future works continues in the enhancement of the proposed network and the empirical validation with a real price predictor (Qian & Rasheed, 2007; Bollen et al., 2011; Patel et al., 2015). Other types of deep learning models can be tested. Bring more features to the input data, like the quantity of negotiated shares per interval, or even sentimental analysis of the market (Bollen et al., 2011). Improve the use of predicted information with online algorithms, some contributions are found in Shalev-Shwartz & Ben-David (2014). The redesign of other algorithms, like OLMAR, that already uses side information (the moving average) to make decisions.

In addition, there is room for improvement of the proposed benchmark using the predictor's accuracy with stochastic dynamic programming.

Bibliography

Agarwal, A., Hazan, E., Kale, S. & Schapire, R. E. (2006), Algorithms for portfolio management based on the Newton method, *in* 'Proceedings of the 23rd international conference on Machine learning - ICML '06', ACM Press, New York, New York, USA, pp. 9–16. DOI 10.1145/1143844.1143846.

Bengio, Y. (1997), 'Using a Financial Training Criterion Rather than a Prediction Criterion', *International Journal of Neural Systems* **08**(04), 433–443. DOI 10.1142/S0129065797000422. ISSN 0129-0657.

Bollen, J., Mao, H. & Zeng, X. (2011), 'Twitter mood predicts the stock market', *Journal of Computational Science* **2**(1), 1–8. DOI 10.1016/J.JOCS.2010.12.007. ISSN 1877-7503.

Borodin, A., El-Yaniv, R. & Gogan, V. (2004), 'Can we learn to beat the best stock', *Journal of Artificial Intelligence Research* **21**(1), 579–594.

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E. & Warmuth, M. K. (1997), 'How to use expert advice', *Journal of the ACM* **44**(3), 427–485. DOI 10.1145/258128.258179. ISSN 00045411.

Chen, K., Zhou, Y. & Dai, F. (2015), A LSTM-based method for stock returns prediction: A case study of China stock market, *in* 'International Conference on Big Data, Big Data', IEEE, pp. 2823–2824. DOI 10.1109/BigData.2015.7364089.

Cover, T. M. (1991), 'Universal Portfolios', *Mathematical Finance* **1**(1), 1–29. DOI 10.1111/j.1467-9965.1991.tb00002.x. ISSN 0960-1627.

Cover, T. M. & Gluss, D. H. (1986), 'Empirical Bayes stock market portfolios', *Advances in Applied Mathematics* **7**(2), 170–181. DOI 10.1016/0196-8858(86)90029-1. ISSN 0196-8858.

Crammer, K., Dredze, M. & Pereira, F. (2009), Exact Convex Confidence-Weighted Learning, *in* D. Koller, D. Schuurmans, Y. Bengio & L. Bottou, eds, 'Advances in Neural Information Processing Systems 21', Curran Associates, Inc., pp. 345–352.

Cybenko, G. (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals, and Systems* **2**(4), 303–314. DOI 10.1007/BF02551274. ISSN 0932-4194.

Dempster, A. A. P., Laird, N. M., Rubin, D. B., Journal, S., Statistical, R. & Series, S. (1977), 'Maximum Likelihood from Incomplete Data via the EM Algorithm', *Journal of the Royal Statistical Society* **39**(1), 1–38.

Dochow, R. (2016), *Online Algorithms for the Portfolio Selection Problem*, Springer Fachmedien Wiesbaden, Wiesbaden.

Dumas, B. & Luciano, E. (1991), 'An Exact Solution to a Dynamic Portfolio Choice Problem under Transactions Costs', *The Journal of Finance* **46**(2), 577. DOI 10.2307/2328837. ISSN 00221082.

Efron, B. & Tibshirani, R. (1986), 'Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy', *Statistical Science* **1**(1), 54–75. DOI 10.1214/ss/1177013815. ISSN 0883-4237.

Freund, Y. & Schapire, R. E. (1999), 'Large Margin Classification Using the Perceptron Algorithm', *Machine Learning* **37**(3), 277–296. DOI 10.1023/A:1007662407062. ISSN 08856125.

Fujiwara, H., Iwama, K. & Sekiguchi, Y. (2011), 'Average-case competitive analyses for one-way trading', *Journal of Combinatorial Optimization* **21**(1), 83–107. DOI 10.1007/s10878-009-9239-4. ISSN 1382-6905.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press.

Hazan, E., Agarwal, A. & Kale, S. (2007), Logarithmic regret algorithms for online convex optimization, *in* 'Machine Learning', Vol. 69, pp. 169–192. DOI 10.1007/s10994-007-5016-8. ISSN 08856125.

Helmbold, D. P., Schapire, R. E., Singer, Y. & Warmuth, M. K. (1998), 'On-Line Portfolio Selection Using Multiplicative Updates', *Mathematical Finance* **8**(4), 325–347. DOI 10.1111/1467-9965.00058. ISSN 0960-1627.

Kelly, J. L. (1956), 'A New Interpretation of Information Rate', *Bell System Technical Journal* **35**(4), 917–926. DOI 10.1002/j.1538-7305.1956.tb03809.x. ISSN 00058580.

Kingma, D. P. & Ba, J. (2014), 'Adam: A Method for Stochastic Optimization', *3rd International Conference for Learning Representations* pp. 434–435. DOI 10.1109/ICCE.2017.7889386. ISSN 09252312.

Koutsoupias, E. & Papadimitriou, C. H. (2000), 'Beyond Competitive Analysis', *SIAM Journal on Computing* **30**(1), 300–317. DOI 10.1137/S0097539796299540. ISSN 0097-5397.

Kraft, D. (1994), 'Algorithm 733: TOMP-Fortran modules for optimal control calculations', *ACM Transactions on Mathematical Software* **20**(3), 262–281. DOI 10.1145/192115.192124. ISSN 00983500.

Krizhevsky, A. (2010), 'Convolutional deep belief networks on cifar-10', *Unpublished manuscript* pp. 1–9.

Le Ny, J. (2009), 'Introduction to Stochastic Approximation Algirithms', *Lecture notes on Dynamic Programming and Stochastic Control* pp. 129–142.

Li, B. & Hoi, S. C. H. (2012), 'On-Line Portfolio Selection with Moving Average Reversion', *Proceedings of the 29th International Conference on Machine Learning (ICML-12)* pp. 273–280. DOI Doi 10.1007/S10994-012-5281-Z.

Li, B. & Hoi, S. C. H. (2016), *Online portfolio selection: principles and algorithms*, CRC Press.

Li, B., Hoi, S. C. H., Zhao, P. & Gopalkrishnan, V. (2013), 'Confidence Weighted Mean Reversion Strategy for Online Portfolio Selection', *ACM Transactions on Knowledge Discovery from Data* **7**(1), 1–38. DOI 10.1145/2435209.2435213. ISSN 15564681.

Maas, A. L., Hannun, A. Y. & Ng, A. Y. (2013), Rectifier Nonlinearities Improve Neural Network Acoustic Models, *in* 'ICML'.

Markowitz, H. (1952), 'Portfolio selection', *The Journal of Finance* **7**(1), 77–91. DOI 10.1111/j.1540-6261.1952.tb01525.x. ISSN 00221082.

Martinez, L. C., Da Hora, D. N., Palotti, J. R., Meira, W. & Pappa, G. L. (2009), From an artificial neural network to a stock market day-trading system: A case study on the BMF BOVESPA, *in* 'Proceedings of the International Joint Conference on Neural Networks', IEEE, pp. 2006–2013. DOI 10.1109/IJCNN.2009.5179050. ISSN 1098-7576.

Mitchell, T. M. (1997), *Machine Learning*, McGraw-Hill International Editions, McGraw-Hill.

Moody, J., Wu, L., Liao, Y. & Saffell, M. (1998), 'Performance functions and reinforcement learning for trading systems and portfolios', *Journal of Forecasting* **17**(5-6), 441–470. DOI 10.1002/(SICI)1099-131X(1998090)17:5/6<441::AID-FOR707>3.3.CO;2-R. ISSN 02776693.

Mulvey, J. M. & Vladimirou, H. (1992), 'Stochastic Network Programming for Financial Planning Problems', *Management Science* **38**(11), 1642–1664. DOI 10.1287/mnsc.38.11.1642. ISSN 0025-1909.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'ICML'10 Proceedings of the 27th International Conference on International Conference on Machine Learning', Association for Computing Machinery, p. 170.

Patel, J., Shah, S., Thakkar, P. & Kotecha, K. (2015), 'Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques', *Expert Systems with Applications* **42**(1), 259–268. DOI 10.1016/J.ESWA.2014.07.040. ISSN 0957-4174.

Qian, B. & Rasheed, K. (2007), 'Stock market prediction with multiple classifiers', *Applied Intelligence* **26**(1), 25–33. DOI 10.1007/s10489-006-0001-7. ISSN 0924-669X.

Rosenblatt, F. (1958), 'THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN', *Psychological Review* **65**(6), 19–8.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), Learning internal representations by error propagation, *in* 'Parallel distributed processing: explorations in the microstructure of cognition, vol. 1', MIT Press, pp. 318–362.

Samuelson, P. A. (1969), 'Lifetime Portfolio Selection by Dynamic Stochastic Programming', *Review of Economics and Statistics* **51**(3), 239–246. DOI 10.2307/1926559. ISSN 00346535.

Shalev-Shwartz, S. & Ben-David, S. (2014), *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, New York, NY, USA.

Sharpe, W. F. (1966), 'Mutual Fund Performance', *The Journal of Business* **39**, 119–138. DOI 10.2307/2351741.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', *Journal of Machine Learning Research* **15**, 1929–1958.

A
Appendix

This appendix contains data information for elucidating any presumption taken on the main work, as for verification of the presented charts.

Φ			assets		
0	EMBR3	CCRO3	BRAP4	BRKM5	BBDC4
1	SBSP3	ITSA4	MRFG3	MRVE3	CYRE3
2	BRML3	CSAN3	PETR4	EMBR3	PCAR4
3	RENT3	WEGE3	GGBR4	USIM5	POMO4
4	CMIG4	BRKM5	CCRO3	BRML3	LAME4
5	MRFG3	GGBR4	BBAS3	ELET3	PETR3
6	JBSS3	GGBR4	CYRE3	ELET3	LAME4
7	RENT3	MRFG3	BRAP4	GGBR4	LAME4
8	CSAN3	USIM5	SBSP3	RENT3	LREN3
9	WEGE3	CYRE3	BRML3	GOAU4	RENT3
10	ENBR3	BBAS3	USIM5	VALE3	GOAU4
11	PCAR4	ITSA4	CYRE3	CSNA3	BBAS3
12	SBSP3	BRAP4	CSNA3	JBSS3	ELET3
13	CMIG4	ELET3	EMBR3	LAME4	CSNA3
14	LAME4	EMBR3	CMIG4	CSAN3	GGBR4
15	GGBR4	CMIG4	BBAS3	MRVE3	BRKM5
16	GGBR4	RENT3	VALE3	CYRE3	BRKM5
17	CCRO3	LAME4	VALE3	SBSP3	POMO4
18	VALE3	MRFG3	EMBR3	ELET3	WEGE3
19	PCAR4	ELET3	BBDC4	RENT3	BRML3
20	EMBR3	MRFG3	VALE3	USIM5	PETR4
21	BRAP4	ELET3	CSNA3	JBSS3	GOAU4
22	BRML3	MRVE3	USIM5	LAME4	ENBR3
23	BBDC4	EMBR3	PETR4	CSNA3	ELET3
24	PCAR4	WEGE3	BRML3	CCRO3	PETR3
25	PETR3	BBAS3	MRFG3	POMO4	BRKM5
26	RENT3	CCRO3	PETR4	POMO4	EMBR3

Continued on next page

Φ	assets				
27	USIM5	WEGE3	CYRE3	BRAP4	BRML3
28	SBSP3	BRML3	RENT3	BBAS3	VALE3
29	VALE3	JBSS3	EMBR3	PETR4	MRVE3
30	PETR3	BBDC4	ENBR3	BBAS3	SBSP3
31	WEGE3	BRAP4	CSNA3	CMIG4	ELET3
32	GGBR4	PCAR4	WEGE3	BBAS3	PETR4
33	JBSS3	BBDC4	USIM5	ENBR3	CMIG4
34	JBSS3	ENBR3	CCRO3	BRML3	CMIG4
35	GGBR4	ELET3	BBAS3	RENT3	CCRO3
36	WEGE3	BRML3	EMBR3	PCAR4	BBAS3
37	MRVE3	CSAN3	LAME4	MRFG3	PETR4
38	MRVE3	CYRE3	CSAN3	GGBR4	GOAU4
39	BBAS3	LAME4	CSAN3	BRML3	ITSA4
40	RENT3	CYRE3	SBSP3	MRVE3	JBSS3
41	LAME4	MRFG3	PETR4	MRVE3	ENBR3
42	EMBR3	JBSS3	VALE3	PETR3	BRAP4
43	BRKM5	EMBR3	ITSA4	BRML3	MRVE3
44	PETR4	GOAU4	CCRO3	POMO4	LAME4
45	EMBR3	CYRE3	LREN3	MRFG3	ENBR3
46	CYRE3	PETR4	CMIG4	MRFG3	CSNA3
47	CCRO3	PETR4	WEGE3	MRFG3	MRVE3
48	POMO4	MRFG3	LAME4	ITSA4	BRAP4
49	BBDC4	ENBR3	POMO4	BRML3	JBSS3
50	LAME4	BRKM5	ELET3	JBSS3	VALE3
51	PETR3	POMO4	BRAP4	USIM5	GGBR4
52	MRFG3	PETR4	VALE3	BRML3	ELET3
53	PETR4	CMIG4	BRML3	EMBR3	SBSP3
54	RENT3	USIM5	PCAR4	PETR3	MRVE3
55	MRVE3	RENT3	LREN3	GOAU4	PETR3
56	ENBR3	CCRO3	USIM5	MRVE3	BRAP4
57	CCRO3	CSAN3	RENT3	USIM5	BBAS3
58	MRVE3	LAME4	USIM5	WEGE3	GOAU4
59	GOAU4	ENBR3	MRFG3	PETR4	CSNA3
60	MRVE3	JBSS3	LAME4	MRFG3	BBDC4
61	PCAR4	EMBR3	POMO4	ITSA4	RENT3
62	LREN3	CMIG4	GOAU4	WEGE3	CSAN3

Continued on next page

Φ	assets				
63	PETR4	CMIG4	PETR3	BRKM5	MRFG3
64	USIM5	MRFG3	BBDC4	WEGE3	CSNA3
65	PETR3	CSAN3	CCRO3	BBAS3	BBDC4
66	WEGE3	LAME4	GOAU4	CSAN3	RENT3
67	JBSS3	GOAU4	ITSA4	BRAP4	PETR3
68	USIM5	CMIG4	MRVE3	CYRE3	BRAP4
69	EMBR3	BRKM5	USIM5	CSNA3	BBDC4
70	PCAR4	BRML3	POMO4	BRAP4	BBDC4
71	GGBR4	CYRE3	LREN3	PETR3	JBSS3
72	POMO4	BRKM5	ELET3	JBSS3	CSNA3
73	ELET3	CCRO3	EMBR3	GGBR4	USIM5
74	SBSP3	LAME4	BBDC4	POMO4	CYRE3
75	USIM5	CSAN3	PETR4	GOAU4	LAME4
76	CMIG4	VALE3	GOAU4	EMBR3	BBDC4
77	CSNA3	GGBR4	JBSS3	GOAU4	ITSA4
78	BRML3	ITSA4	JBSS3	GOAU4	EMBR3
79	POMO4	BRAP4	PCAR4	CSNA3	GGBR4
80	BBAS3	VALE3	ENBR3	WEGE3	CCRO3
81	ITSA4	RENT3	ELET3	CYRE3	CSNA3
82	CSAN3	WEGE3	CSNA3	BBAS3	PETR4
83	WEGE3	MRFG3	RENT3	USIM5	CYRE3
84	BRAP4	CYRE3	PETR3	PETR4	LAME4
85	EMBR3	ENBR3	BRML3	MRVE3	BRKM5
86	SBSP3	PETR3	ITSA4	CCRO3	BRML3
87	ITSA4	LREN3	EMBR3	USIM5	CMIG4
88	MRFG3	WEGE3	CMIG4	JBSS3	BBDC4
89	CCRO3	VALE3	PCAR4	CSNA3	RENT3
90	JBSS3	ELET3	ENBR3	CMIG4	PETR3
91	EMBR3	ENBR3	BBDC4	CSNA3	BBAS3
92	GOAU4	RENT3	LREN3	BBAS3	JBSS3
93	LREN3	RENT3	WEGE3	POMO4	ELET3
94	CYRE3	PCAR4	MRVE3	SBSP3	RENT3
95	VALE3	EMBR3	CSNA3	SBSP3	LREN3
96	BRAP4	VALE3	POMO4	USIM5	BBDC4
97	MRFG3	POMO4	PCAR4	WEGE3	CMIG4
98	BRAP4	PCAR4	WEGE3	USIM5	ELET3

Continued on next page

Φ	assets				
99	ELET3	BRML3	USIM5	BRKM5	GOAU4
100	PCAR4	GOAU4	BRKM5	JBSS3	USIM5
101	BBAS3	CCRO3	LAME4	BRML3	ELET3
102	PETR4	BRML3	ELET3	ENBR3	GOAU4
103	BRML3	BRKM5	GGBR4	USIM5	POMO4
104	RENT3	BBDC4	VALE3	JBSS3	ELET3
105	BRAP4	BRKM5	LREN3	BRML3	CSAN3
106	ITSA4	JBSS3	BRAP4	CSNA3	BRML3
107	BBAS3	BBDC4	EMBR3	POMO4	SBSP3
108	ENBR3	SBSP3	USIM5	MRVE3	VALE3
109	BBDC4	ELET3	BRKM5	CYRE3	PETR3
110	SBSP3	CMIG4	WEGE3	PCAR4	USIM5
111	PCAR4	SBSP3	BRAP4	EMBR3	LAME4
112	LAME4	LREN3	PCAR4	CYRE3	BRAP4
113	CSNA3	CCRO3	PETR4	ITSA4	PCAR4
114	USIM5	PETR3	GGBR4	EMBR3	LREN3
115	GGBR4	ELET3	CCRO3	VALE3	GOAU4
116	CMIG4	VALE3	BRML3	BBAS3	GOAU4
117	ITSA4	LREN3	WEGE3	ELET3	BBDC4
118	ENBR3	USIM5	PETR3	BRML3	CCRO3
119	CSAN3	ITSA4	USIM5	CSNA3	PETR3
120	MRVE3	BBAS3	CSNA3	PCAR4	LAME4
121	ELET3	PETR3	CSNA3	BRAP4	LREN3
122	CCRO3	WEGE3	CMIG4	GGBR4	GOAU4
123	MRVE3	CSAN3	PCAR4	BRAP4	BBAS3
124	CSNA3	JBSS3	SBSP3	BBDC4	GGBR4
125	MRVE3	CSAN3	PETR4	BBAS3	GGBR4
126	MRFG3	ENBR3	CSAN3	VALE3	PETR3
127	USIM5	VALE3	POMO4	PCAR4	CSNA3
128	POMO4	CCRO3	CYRE3	PETR4	ENBR3
129	LAME4	CCRO3	ELET3	CYRE3	MRFG3
130	SBSP3	CSNA3	CMIG4	PETR3	GOAU4
131	BRML3	PETR3	PCAR4	BBAS3	VALE3
132	PETR4	LAME4	BBDC4	ENBR3	BRML3
133	JBSS3	ELET3	CSNA3	SBSP3	ENBR3
134	SBSP3	ELET3	USIM5	VALE3	WEGE3

Continued on next page

Φ	assets				
135	CSNA3	EMBR3	SBSP3	ITSA4	GOAU4
136	GGBR4	CMIG4	BRKM5	CSNA3	LAME4
137	ENBR3	BRAP4	LAME4	ITSA4	POMO4
138	CMIG4	USIM5	WEGE3	BBDC4	SBSP3
139	SBSP3	CYRE3	MRVE3	PETR4	BBDC4
140	RENT3	PCAR4	POMO4	EMBR3	LREN3
141	ENBR3	BBDC4	BBAS3	JBSS3	CCRO3
142	LREN3	EMBR3	WEGE3	GGBR4	ITSA4
143	BRML3	SBSP3	VALE3	BRKM5	CYRE3
144	CSNA3	SBSP3	POMO4	PETR3	VALE3
145	CMIG4	PETR3	BBDC4	SBSP3	BBAS3
146	BRML3	LREN3	RENT3	CSNA3	CCRO3
147	MRVE3	GOAU4	BRAP4	BRKM5	LREN3
148	ITSA4	GOAU4	USIM5	BRML3	CYRE3
149	BRAP4	LAME4	ITSA4	LREN3	RENT3
150	WEGE3	LREN3	CYRE3	JBSS3	CSNA3
151	RENT3	BBAS3	LAME4	BRML3	SBSP3
152	CCRO3	CSAN3	USIM5	PETR3	BBAS3
153	CSNA3	VALE3	LREN3	RENT3	CCRO3
154	GOAU4	BRML3	JBSS3	POMO4	BRKM5
155	CMIG4	CYRE3	ITSA4	POMO4	USIM5
156	CCRO3	SBSP3	BBDC4	BRAP4	USIM5
157	LREN3	POMO4	EMBR3	PCAR4	CCRO3
158	USIM5	BRKM5	ELET3	CSAN3	MRVE3
159	CSAN3	CSNA3	CCRO3	PCAR4	CYRE3
160	ITSA4	CSAN3	BBAS3	CYRE3	SBSP3
161	GOAU4	SBSP3	BRAP4	MRFG3	ENBR3
162	LREN3	USIM5	CCRO3	BRKM5	ENBR3
163	MRVE3	BRML3	ENBR3	RENT3	JBSS3
164	CMIG4	BBDC4	WEGE3	CSNA3	CSAN3
165	CCRO3	PETR4	ENBR3	BRKM5	GOAU4
166	POMO4	CSAN3	WEGE3	VALE3	USIM5
167	GGBR4	MRFG3	ELET3	PETR4	CYRE3
168	PCAR4	RENT3	MRVE3	BRML3	PETR3
169	MRFG3	ITSA4	CSAN3	LAME4	BRKM5
170	VALE3	RENT3	CMIG4	GOAU4	ITSA4

Continued on next page

Φ	assets				
171	PETR3	ITSA4	ENBR3	PETR4	VALE3
172	EMBR3	GOAU4	MRVE3	GGBR4	LREN3
173	BRKM5	GOAU4	BRML3	LREN3	RENT3
174	CSNA3	CCRO3	PETR4	JBSS3	BRAP4
175	MRFG3	MRVE3	WEGE3	ITSA4	BRAP4
176	MRVE3	RENT3	LAME4	LREN3	BBAS3
177	VALE3	CMIG4	CCRO3	PETR3	PCAR4
178	BBAS3	CSNA3	BRKM5	LREN3	ITSA4
179	LREN3	PCAR4	BRKM5	CCRO3	VALE3
180	CMIG4	RENT3	BRAP4	GGBR4	CSAN3
181	LAME4	JBSS3	BRKM5	BBDC4	ENBR3
182	CCRO3	VALE3	BRML3	SBSP3	RENT3
183	PETR3	ITSA4	MRFG3	PCAR4	MRVE3
184	LAME4	ENBR3	BRML3	VALE3	USIM5
185	WEGE3	BBAS3	SBSP3	ITSA4	ELET3
186	BRAP4	MRFG3	BBDC4	WEGE3	SBSP3
187	ENBR3	USIM5	VALE3	MRFG3	BRKM5
188	GGBR4	POMO4	PETR4	PCAR4	ENBR3
189	JBSS3	MRVE3	ENBR3	VALE3	ELET3
190	BBAS3	VALE3	CMIG4	SBSP3	LREN3
191	CMIG4	EMBR3	BRKM5	USIM5	CSNA3
192	LREN3	MRVE3	PETR4	BBAS3	CMIG4
193	GOAU4	EMBR3	MRVE3	CSNA3	RENT3
194	GGBR4	PETR3	CMIG4	BRAP4	BRKM5
195	BRML3	CMIG4	BRKM5	PETR4	CCRO3
196	BBAS3	RENT3	EMBR3	CYRE3	BRKM5
197	MRVE3	VALE3	BRAP4	BRKM5	LREN3
198	MRVE3	POMO4	VALE3	GGBR4	BBAS3
199	GOAU4	LAME4	BBAS3	JBSS3	MRFG3

Table A.1: The list off all random portfolios generated with the available assets listed in Table 6.1.

Φ \ p	0.45	0.50	0.55	0.60	0.65	0.70	0.75
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
9	9	9	9	9	9	9	9
12	12	12	12	12	12	12	12
15	15	15	15	15	15	15	15
16	16	16	16	16	16	16	16
24	24	24	24	24	24	24	24
27	27	27	27	27	27	27	27
28	28	28	28	28	28	28	28
32	32	32	32	32	32	32	32
34	34	34	34	34	34	34	34
39	39	39	39	39	39	39	39
50	50	50	50	50	50	50	50
51	51	51	51	51	51	51	51
53	53	53	53	53	53	53	53
58	58	58	58	58	58	58	58
60	60	60	60	60	60	60	60
62	62	62	62	62	62	62	62
65	65	65	65	65	65	65	65
71	71	71	71	71	71	71	71
75	-	-	-	-	-	-	75
80	80	80	80	80	80	80	80
90	90	90	90	90	90	90	90
94	94	94	94	94	94	94	-
98	98	98	98	98	98	98	98
99	99	99	99	99	99	99	99
105	105	105	105	105	105	105	105
106	106	106	106	106	106	106	106
110	110	110	110	110	-	-	-
116	116	116	116	116	116	116	116
117	117	117	117	117	117	117	117
119	119	119	119	119	119	119	119
122	122	122	122	122	122	122	122
131	131	131	131	131	131	131	131
132	132	132	132	132	132	132	132
133	133	133	133	133	133	133	133
136	136	136	136	136	136	136	136
142	142	142	142	142	142	142	142
146	-	-	-	-	146	146	146
149	149	149	149	149	149	149	149
162	162	162	162	162	162	162	162
165	165	165	165	165	165	165	165
171	171	171	171	171	171	171	171
172	172	172	172	172	172	172	-
173	173	-	-	-	-	-	-
174	-	-	-	-	-	-	174
177	177	177	177	177	177	177	177
179	179	179	179	179	179	179	179
181	181	181	181	181	181	181	181
193	193	193	193	193	193	193	193
194	-	194	194	194	194	194	194
195	195	195	195	195	195	195	195
197	197	197	197	197	197	197	197

Table A.2: Portfolios outliers (54) for the first quartile of the *validation* phase, 92.6% of equally portfolios.

Φ \ p	0.45	0.50	0.55	0.60	0.65	0.70	0.75
8	8	8	8	8	8	8	8
10	10	10	10	10	10	10	10
12	12	12	12	12	12	12	12
13	13	13	13	13	13	13	13
17	17	17	17	17	17	-	-
20	20	20	20	20	20	20	20
22	22	22	22	22	22	22	22
30	30	30	30	30	30	30	30
35	35	35	35	35	35	35	35
36	36	36	36	36	36	36	36
46	46	46	46	46	46	46	46
47	47	47	47	47	47	47	47
53	-	53	53	-	-	-	-
54	54	54	54	54	54	54	54
55	55	-	55	55	55	55	55
56	56	56	56	56	56	56	56
57	57	57	57	57	57	57	57
64	64	64	64	64	64	64	64
81	81	81	81	81	81	81	81
86	86	86	86	86	86	86	86
88	88	88	88	88	88	88	88
90	90	90	90	90	90	90	90
92	92	-	-	-	-	-	-
93	93	93	93	93	93	93	93
94	94	94	94	94	94	94	94
98	98	98	98	98	98	98	98
99	99	99	99	99	99	99	99
102	102	102	102	102	102	102	102
103	103	103	103	103	103	103	103
105	105	105	105	105	105	105	105
109	-	109	-	109	109	109	109
110	110	110	110	110	110	110	110
117	117	117	117	117	117	117	117
120	120	120	120	120	120	120	120
123	123	123	123	123	123	123	123
132	132	132	132	132	132	132	132
133	133	133	133	133	133	133	133
139	139	139	139	139	139	139	139
146	146	146	146	146	146	146	146
149	-	-	-	-	-	149	149
152	152	152	152	152	152	152	152
155	155	155	155	155	155	155	155
164	164	164	164	164	164	164	164
168	168	168	168	168	168	168	168
171	171	171	171	171	171	171	171
176	176	176	176	176	176	176	176
177	177	177	177	177	177	177	177
178	178	178	178	178	178	178	178
180	180	180	180	180	180	180	180
190	190	190	190	190	190	190	190
191	191	191	191	191	191	191	191
197	197	197	197	197	197	197	197
199	199	199	199	199	199	199	199

Table A.3: Portfolios outliers (53) for the first quartile of the *test* phase, 94.3% of equally portfolios.

Φ	MLPi	OPTi	Φ	MLPi	OPTi	Φ	MLPi	OPTi	Φ	MLPi	OPTi
0	21.3764	21.4660	50	29.8810	25.9807	100	31.5325	29.0645	150	27.8264	26.8797
1	26.2459	24.7897	51	34.4951	29.4589	101	26.2772	23.7046	151	21.3231	21.4952
2	22.6930	22.1606	52	26.9981	25.7423	102	25.8946	27.3953	152	17.9808	22.9325
3	35.5196	30.4083	53	23.8865	24.2843	103	25.0811	31.0637	153	25.3126	22.5631
4	25.6483	24.4783	54	20.3889	24.3765	104	28.5932	24.3862	154	37.3740	33.0406
5	28.6469	25.7906	55	24.0229	24.5569	105	19.9040	22.3714	155	30.2019	32.2274
6	30.6572	28.8537	56	24.1198	25.0810	106	29.2894	26.9311	156	25.6772	23.0260
7	27.1987	25.3064	57	22.5440	23.2174	107	25.1828	24.1721	157	30.4631	25.7547
8	21.9739	23.8259	58	30.9257	27.8918	108	25.4081	23.9237	158	28.4499	27.1921
9	29.4588	26.3910	59	31.6506	28.3854	109	22.9855	23.5032	159	28.0996	24.2417
10	22.2361	24.5319	60	28.7948	25.5181	110	22.4953	25.6093	160	21.0847	21.1855
11	24.9076	22.8587	61	30.4712	26.2005	111	22.1412	22.1098	161	28.1356	26.7655
12	23.6228	28.3732	62	29.6063	25.9314	112	23.7761	22.0300	162	27.0175	24.1860
13	27.0295	28.4800	63	26.6767	24.7199	113	24.0160	22.9460	163	27.3192	25.3172
14	26.6493	25.1782	64	22.6712	26.9037	114	29.1410	25.0808	164	20.4403	24.1587
15	24.1698	24.0829	65	19.5795	19.0937	115	31.4019	26.4320	165	27.4120	24.5028
16	24.6036	23.0059	66	27.9879	24.9531	116	26.8350	24.7382	166	32.9810	28.2640
17	25.3558	25.7544	67	30.3418	26.7086	117	14.6261	22.1863	167	30.1227	27.9677
18	28.7570	26.0799	68	29.0682	27.4473	118	27.4507	25.1058	168	20.1391	22.5156
19	24.3715	22.9543	69	29.0755	25.2235	119	29.8684	25.5742	169	26.2966	24.0218
20	18.6312	25.6146	70	29.3504	25.3955	120	22.3449	22.8777	170	28.9092	25.3443
21	32.2445	31.0529	71	27.6225	25.6309	121	29.3454	25.2626	171	16.2698	19.6160
22	20.7647	25.9041	72	37.7185	33.6888	122	29.7067	27.2814	172	28.7527	25.8965
23	27.8320	24.7009	73	29.0078	28.2722	123	16.7191	20.9663	173	25.7843	24.9687
24	21.8039	22.0890	74	31.6288	25.9801	124	29.8764	25.4680	174	29.7989	25.8514
25	31.8781	27.4305	75	30.0586	26.6511	125	23.8796	21.7440	175	26.4183	24.6893
26	30.3256	26.6499	76	29.8321	24.3590	126	25.5166	22.8767	176	19.3190	20.8160
27	28.7854	26.1979	77	30.9798	29.2802	127	34.8707	29.1099	177	21.0936	22.0965
28	21.6063	20.6922	78	31.7203	28.4432	128	31.2040	27.1571	178	19.7403	22.1318
29	27.2708	24.0239	79	30.6977	28.4837	129	29.7601	27.2974	179	21.6168	20.4804
30	16.3715	19.0553	80	20.6121	20.2308	130	31.1336	27.6335	180	21.7604	24.5609
31	29.3962	27.7882	81	25.2815	27.0766	131	21.1384	19.9468	181	24.5649	23.1792
32	22.5623	21.0432	82	25.2713	22.2295	132	20.4298	21.0204	182	21.6168	21.7939
33	31.1218	27.3114	83	30.8348	27.6296	133	27.4946	28.5703	183	24.4197	23.7276
34	26.4200	26.6802	84	22.8447	21.8884	134	28.1065	25.7139	184	26.4966	24.4354
35	20.1561	24.0747	85	23.5896	23.5610	135	31.2295	26.7934	185	24.9511	22.7998
36	20.5876	21.5507	86	20.3548	22.2476	136	26.5760	26.5376	186	25.5460	22.7922
37	24.3729	24.1161	87	28.5120	26.2051	137	30.2893	26.8234	187	29.8217	25.7627
38	26.9276	26.5382	88	26.1744	26.7806	138	27.2673	24.6547	188	29.5522	26.7898
39	22.2736	21.2154	89	23.4299	22.4949	139	20.2309	21.2187	189	28.0617	26.2682
40	26.6563	25.1835	90	27.2915	28.1622	140	30.3182	25.6841	190	19.5489	21.0924
41	24.6135	24.2926	91	26.4298	21.7294	141	24.8807	21.6712	191	26.3572	28.4894
42	26.0126	23.2078	92	25.5824	25.2916	142	23.6458	23.0033	192	24.2927	22.0446
43	23.6551	23.5301	93	22.1250	28.3337	143	23.7465	22.6368	193	31.1274	27.6182
44	34.6697	29.4642	94	20.3584	22.3631	144	33.0308	27.0595	194	24.3310	24.2938
45	26.9076	24.7112	95	24.3108	22.8580	145	21.8346	20.4750	195	25.1469	24.0723
46	27.3938	28.3707	96	31.5391	26.0379	146	20.2299	24.0991	196	21.7757	22.2202
47	20.1984	24.2442	97	32.2249	30.0158	147	28.9297	24.7452	197	16.8034	20.9560
48	33.9050	29.1408	98	24.1700	26.2884	148	30.0253	28.7320	198	30.9263	25.8202
49	34.0305	28.4243	99	25.7414	30.1945	149	20.9589	21.6054	199	23.9117	28.1967

Table A.4: The Sharpe ration, computed by $sharpe = \frac{(R_\Phi - R_f)}{\sigma_\Phi}$, where R_Φ is the annual return of portfolio Φ, and R_f is a free risk return, here defined at 10%, and the σ_Φ is the annual standard deviation. The Sharpe ratio is better suited for portfolios comparison, portfolios with higher Sharpe ratios are preferable.

p	OPTi	MLPi	OLMAR	BCRP	MLPi (ii)	MLPi (i)
0.4500	3.4259	3.6158	6.7063	1.0584	3.9153	4.2275
0.5000	5.8808	6.4560	6.7063	1.0584	7.2189	8.1389
0.5500	10.2273	11.8045	6.7063	1.0584	13.6246	15.8112
0.6000	18.0388	22.1050	6.7063	1.0584	26.2877	31.4901
0.6500	32.2858	42.4561	6.7063	1.0584	52.0406	63.6138
0.7000	58.7342	83.8137	6.7063	1.0584	105.7120	131.7604
0.7500	108.7999	170.4091	6.7063	1.0584	221.2186	276.9419

Table A.5: The monthly average wealth achieved by the algorithms for each accuracy level.