Computational Methods for Investment Portfolio: the Use of Fuzzy Measures and Constraint Programming for Risk Management

Tanja Magoč[1], François Modave[1], Martine Ceberio[1], and Vladik Kreinovich[1]

University of Texas at El Paso
Computer Science Department
500 West University Avenue
El Paso, Texas 79968-0518
tmagoc@miners.utep.edu, fmodave@utep.edu, mceberio@utep.edu,
vladik@utep.edu

Summary. Computational intelligence techniques are very useful tools for solving problems that involve understanding, modeling, and analysis of large data sets. One of the numerous fields where computational intelligence has found an extremely important role is finance. More precisely, optimization issues of one's financial investments, to guarantee a given return, at a minimal risk, have been solved using intelligent techniques such as genetic algorithm, rule-based expert system, neural network, and support-vector machine. Even though these methods provide good and usually fast approximation of the best investment strategy, they suffer some common drawbacks including the neglect of the dependence among among criteria characterizing investment assets (i.e. return, risk, etc.), and the assumption that all available data are precise and certain. To face these weaknesses, we propose a novel approach involving utility-based multi-criteria decision making setting and fuzzy integration over intervals.

1 Introduction

Given the pervasive nature of computer science, virtually all areas have had to deal with enormous amounts of data. These data alone do not provide much information if they cannot be analyzed, understood, and used to extend knowledge. The strength of computational intelligence is to give a wide variety of techniques that can be used to process, model and understand these datasets. One of the fields where computational intelligence has been extremely useful is finance. To simplify, we can consider two types of problems of interest for the computational intelligence community: pricing and portfolio management. The former deals with how to assign a price to a derivative instrument in such a way that arbitrage is not necessary, whereas the latter deals with the optimization issues of one's financial investments, to guarantee a given return, at a minimal risk. Pricing theory is tackled mostly from a stochastic perspective, using models such as Black-Scholes [16]. On the other hand, portfolio management is a natural area of application for computational intelligence. The problem we are interested in is the selection of optimal portfolio–a distribution of wealth over several investment assets in order to diversify risk and obtain a maximal return for the given acceptable level of risk. Typically, the higher the value of the expected return, the higher the value of risk associated with the asset. Besides the return and the risk, other factors, such as time to maturity and transaction cost, influence the decisions of how much money to invest in each asset under the consideration.

Depending on the need of the investor, different goals could be sought. Two most commonly considered problems in portfolio selection are maximization of wealth and minimization of risk to acquire a required level of wealth.

Various investment strategies have been examined to identify an optimal portfolio in different settings including simple return-based strategies that do not consider other characteristics of assets, the strategies that use stochastic processes to model the behavior of assets and portfolios, and strategies that involve intelligent systems. The aim of this chapter is two-fold. First, we present an extensive selection of computational intelligence techniques such as genetic algorithms, neural networks, supports vector machines, and expert-based systems to select optimal portfolio management strategies. Then, we propose a novel approach based on utility-based multi-criteria decision making setting and fuzzy integration over intervals, as a natural framework for portfolio management.

2 Mathematical background

As we have mentioned in the Introduction, the two most commonly considered problems in portfolio selection are maximization of wealth and minimization of risk to acquire a required level of wealth.

The simplest way of representing these problems is as an optimization problem subject to constraints, which maximizes or minimizes a simple objective function

$$\text{maximize (or minimize)} \sum_{i=1}^{m} w_i x_i, \tag{1}$$

subject to constraints such as

$$w_i \geq 0 \ \forall i \in \{1, 2, \ldots, m\} \tag{2}$$

$$\sum_{i=1}^{m} w_i r_i \leq risk \tag{3}$$

$$\sum_{i=1}^{m} w_i R_i \geq return \tag{4}$$

$$\sum_{i=1}^{m} w_i = 1 \tag{5}$$

where x_i is either return (in maximization problems) or risk (in minimization problems), m is the number of investment assets, w_i is the proportion of wealth invested in the asset i, R_i is the return rate of the asset i, r_i is the risk of the asset i, *return* is required level of return, and *risk* is the level of risk acceptable by the investor. As presented above, selecting the optimal portfolio seems to be a straightforward linear programming problem. However, this representation is just the simplest problem that we can face when looking for the optimal portfolio. In general, many more constraints are imposed on the solution. Moreover, the objective function and the constraints are usually much more complex if more information, such as transaction cost, time period, preferable portfolio structure, relationships between characteristics of assets, etc., are taken into consideration. A real-life problem of portfolio selection is most commonly a non-linear optimization problem with constraints that is usually not (easily) solvable using general constraint solving techniques due to their deficiencies to find a global optimum.

Regardless of the setting and its complexity, in the portfolio optimization problem, we aim at finding the vector of weights (i.e. amount of wealth allocated into each asset), $\mathbf{w} = (w_1, w_2, \ldots, w_m)$, given all the other parameters. However, the risk and the return of each asset are predicted rather than certain values, so the uncertainty of the values complicate even more the process of selecting the optimal portfolio.

3 Genetic algorithms

In this section, we first give a general description of genetic algorithms, and then explain how these algorithms work in a portfolio management framework. We also compare how genetic algorithm approaches perform versus other approaches, e.g., greedy algorithms.

3.1 Theoretical background

A genetic algorithm (GA) is an optimization method (see e.g., [11],[14],[24],[25]) that imitates biological process of natural survival of the fittest individuals in a population. Each individual is characterized by a sequence of genes, which constitute a chromosome. The fittest individuals are selected for mating. Through exchange of chromosomal material between selected pairs and through mutations, the new generation is produced. Thus, generating a new population follows a three-step process: selection, crossover–exchange of genetic material between two individuals to produce one or more offsprings, and mutation in genes. Genetic algorithm simulates all three steps of the natural evolution process.

A genetic algorithm starts by defining its optimization variables and the fitness function. Each variable represents a gene, and all genes of an individual represented a chromosome:

Definition 2.1. (Chromosome) *A chromosome representing an individual i is the vector of all genes of this individual*:

$$\text{chromosome}_i = [p_1, \dots, p_N], \tag{6}$$

where N is the number of genes (variables), and p_j $\forall j = 1, \dots, N$ are genes (i.e. values of the variables) of the individual i.

Since the variables could include qualitative as well as quantitative values of different ranges, each of them needs to be encoded into a finite set of distinct values, usually represented using binary digits.

The next step is to define the fitness function used in the algorithm.

Definition 2.2. (Fitness function) *The fitness function*

$$f(\text{chromosome}) = f(p_1, \dots, p_N) \tag{7}$$

represents an optimization criterion that defines the fitness of each individual.

Usually, the fitness is to be maximized, so that the fittest individuals are selected for the next step. However, the fitness function could be defined as a cost function in which case the fittest individuals are the ones with the lowest cost.

After defining the variables and the fitness function, the initial population is generated either by a random number generator or by encoding values of variables for specific individuals. Initializing the population ends the preparation part of the genetic algorithm and denotes the beginning of the iterative steps. The first of three iterative steps is selection. A proportion of the population is selected to proceed to the next step and the remainder of the population is discarded. Most commonly, the generational gap, that is the percentage of the population selected to continue process, is 50%, but any other percentage could be used. The selection process is based on the fitness level of individuals and could be performed mainly in two different ways. The first method ranks all individuals based on their fitness level and selects top ranked individuals. The second method of selection relies on random selection in which a higher probability of selection is given to fitter individuals.

The selected individuals proceed to the crossover step that chooses two individuals for mating in order to produce one or two offsprings. The most commonly used method for mating is the one-point crossover technique that picks a random point r between the first and the last position in a chromosome, a point called crossover point, and produces two offsprings in the following way. The first offspring copies the genes 1 to r from the first parent and the genes $r+1$ to N from the second parent, while the second offspring is produced by changing the order of the parents. The crossover using different parents continues until the number of individuals is increased to the original size of the population. Note however, that there are some variations in how to perform the crossover step.

The crossover step is followed by mutation. A proportion of genes is chosen for mutation. The mutating genes are selected randomly. The selected genes take random values from the domain of the variable. The mutation process is very important since it slows down the quick convergence of the population in a small search area. It also allows the current best solution to jump away from a local optimum that is not a global optimum. However, it is desired that the current best individual is not mutated in order to not lose the current solution, so many GAs apply the elitist strategy to protect the individual with the highest fitness from being mutated.

Finally, the fitness of each individual in the population is calculated again and the convergence criterion is checked. Ideally, at the end, all individuals in the population have the same genes, representing the optimal solution. However, a genetic algorithm is usually stopped after a predefined number of iterations, which results in a set of optimal values rather than just a single solution, a characteristic that suits portfolio selection problems very well.

3.2 Applications to portfolio management

As a computational intelligence technique, GAs have found different applications in portfolio management. In [18], the authors developed a two-stage algorithm to allocate wealth among numerous investment assets to reach an investor's goal. The first step, first described in [32], uses a GA to select the highest performing assets among thousands of available assets, while the second step utilizes another GA to find an optimal wealth distribution among chosen assets.

The choice of the assets to proceed to the second stage of the algorithm is based purely on the return of assets. Each asset is represented as a chromosome containing four genes. Each of the four genes is a representation of one financial indicator used as an input variable. The four variables are:

- Return on capital employed: $\mathrm{ROCE} = \frac{\text{profit}}{\text{shareholder's equity}} \cdot 100\%$.
- Price/Earning Ratio: $\mathrm{P/E} = \frac{\text{profit}}{\text{earnings per share}} \cdot 100\%$.
- Earning per share: $\mathrm{EPC} = \frac{\text{net income}}{\text{the number of ordinary shares}}$.
- Liquidity ratio: $\frac{\text{current assets}}{\text{current liabilities}} \cdot 100\%$.

Each financial indicator is rated and takes one of eight values (0-7), where 0 stands for a poor performance of the asset and 7 represents a very good performance. These values are encoded as binary numbers so that each gene is a three-digit binary number.

Next, "fitness" of each asset is determined. To find the fitness of an asset, all assets are ranked based on the annual price return (APR):

$$APR_n = \frac{ASP_n - ASP_{n-1}}{ASP_{n-1}}, \tag{8}$$

where APR_n is the annual price return for the year n and ASP_n is the annual stock price for the year N. The assets with high APR are considered good assets. Thus, all the assets are ranked from 1 to N where the asset with the highest APR is ranked 1, and the asset with the lowest APR is ranked N. The asset's ranking, r, is then mapped into the range $0-7$ using the linear mapping $R_{\text{actual}} = 7 \cdot \frac{N-r}{N-1}$, where N is the number of assets. Finally, a fitness function, which determines the optimization criterion, is designed. The most commonly used fitness function is the mean square error between the estimated ranking and next year's actual ranking:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(R_{\text{derived}} - R_{\text{actual}})^2}. \tag{9}$$

The goal is to minimize the value of $RMSE$.

After defining the variables and the fitness function, the selection step of GA is performed. Chromosomes are selected randomly for crossover with a higher probability for selection being given to chromosomes with a higher fitness. The one-point crossover technique, which is used to combine two parents to produce two offsprings, picks a position in a chromosome and interchanges the values of two parents at this position. Finally, a random mutation in each gene changes 0 to 1 or vice versa with a probability equal to 0.005.

The generation produced by this method is either accepted as a final population or another iteration of selection, crossover, and mutation is performed. The process stops when one of the following three conditions is satisfied:

- A predefined number of iterations is reached.
- A defined fitness is reached.
- A convergence criterion of the population is reached. In an ideal case, all the chromosomes of the final generation have the same genes, representing the optimal solution.

At the end of the first step of two-stage portfolio optimization algorithm, the assets are ranked based on their return and the best m assets are considered for investment. The second stage of the algorithm determines the wealth distribution among these m assets. It takes into consideration the risk as well as the return with the goal to minimize the risk for the expected level of the return.

This step of the algorithm is based on yet another genetic algorithm. Before applying the second GA, the expected return of each asset and the covariance between each pair of assets are calculated. The expected return of the asset i after n time intervals is calculated as

$$E(R_i) = \sum_{t=1}^{n} \frac{R_{it}}{n}, \tag{10}$$

where

$$R_{it} = \frac{SCP_{it} - SCP_{i(t-1)}}{SCP_{i(t-1)}} \tag{11}$$

is the return of the asset i for time t and SCP_{it} is the closing price for the asset i at time t. The covariance between assets i and j is given by

$$\sigma_{ij} = \frac{1}{n} \sum_{t=1}^{n} (R_{it} - E(R_i)) \cdot (R_{jt} - E(R_j)). \tag{12}$$

The algorithm designs chromosomes using the binary representation of asset's weight, w_i, which is the amount of wealth allocated to the asset i. The weight of the asset i is normalized by

$$x_i = \frac{w_i}{\sum_{j=1}^{m} w_j} \tag{13}$$

to fit into 8-bits allocated for representation of each chromosome. The weights are adjusted through the GA algorithm until the optimal weights are achieved.

Next, a fitness function, defined by

$$Fitness = \sum_{i=1}^{m} \sum_{j=1}^{m} \sigma_{ij} x_i x_j + \left(\sum_{i=1}^{m} E(R_i) x_i - R_p^* \right)^2 \tag{14}$$

is designed to take into consideration the tradeoff between the risk and the return. The optimal solution is obtained by minimizing this function. The first term of the fitness function minimizes the risk, which is defined as volatility of assets included in the portfolio, while the second term minimizes the difference between the portfolio's overall return and the pre-defined required return, R_p^*. The fitness function for each chromosome determines the assets chosen for the selection, crossover, and mutation, which are performed similarly to the processes in the first GA. The results of these processes determine the generation for the next iteration. The final generation determines the distribution of wealth among the chosen assets.

The algorithm was tested on data obtained from Shanghai Stock Exchange during a period ranging from January 2001 to December 2004. The test used monthly and yearly available information. After the first stage of the algorithm, top 10, 20, and 30 stocks were selected for three different experiments. The results showed that the greater the number of stocks selected to be included in the portfolio, the lower the return of the portfolio. The portfolio with 10 stocks produced by the genetic algorithm was also tested against the equally weighted portfolio, which allocates equal amount of money in each of 10 stocks. The investment portfolio that resulted from the GA constantly outperformed the equally weighted portfolio.

A similar two-stage genetic algorithm was build in [19]. The only differences are the details of asset representation and selection, crossover, and mutation processes. An asset is again represented as a chromosome, which is an $n-$dimensional vector consisting of n parameters called genes. If the initial population contains m assets, the selection process picks exactly $[\frac{m}{2}]$ assets with the highest fitness and discards all other assets. For the crossover stage, a random positive integer, $r \leq n$, is selected and two offsprings are produced by the following procedure. The first offspring copies the first r genes from the first parent and the last $n - r$ genes from the second parent, while the second offspring is created by copying the first r genes from the second parent and the last $n - r$ genes from the first parent. Formally, two parents P_1 and P_2 yield two offsprings O_1 and O_2 by following rules:

$$O_1 = \{g_i | g_i \in P_1 \text{ if } i \leq r \text{ else } g_i \in P_2\} \tag{15}$$

and

$$O_2 = \{g_i | g_i \in P_2 \text{ if } i \leq r \text{ else } g_i \in P_1\} \tag{16}$$

where g_i represents the i^{th} gene.

Finally, the mutation is performed by randomly selecting another positive integer r, $r \leq n$. All the genes except the r^{th} gene are copied, and gene r takes a random value that represents a possible mutation.

The algorithm was tested on data obtained from the Australian Stock Exchange. The results were compared against a Greedy algorithm and the comparison showed that the genetic algorithm performed only slightly weaker than Greedy algorithm but ran much faster.

4 Rule-based expert systems

Even though genetic algorithms showed good results when applied to portfolio management, other intelligent systems have been used as well to optimize the distribution of wealth among assets. Rule-based expert system is one of these techniques, so we review the basic of expert systems and then describe their application to portfolio selection.

4.1 Theoretical background

Rule-based expert system simulates the decision making ability of a human expert in a field of interest (see e.g., [23]). The system is designed to allow "communication" between a user and itself in order to obtain some information that is necessary for solving a problem. This is done through a user-interface, which consists of a pseudo-natural language processing component that allows interaction between the user and the system using a limited form of natural language. Another role of the user interface is to display the solution of the problem being considered to the user along with possible explanation for the decision actually made.

The "brain" of the expert based system consists of two parts–the knowledge base and the inference engine. The knowledge base contains the facts and the rules of the subject at hand. The rules are usually the rules of predicate calculus. The inference engine consists of processes that manipulate the knowledge base to make inferences.

The rules are usually directly entered in the system's knowledge base. However, sometimes the rules are inferred through training samples. The process of building an expert system this way usually iterates through many cycles until human experts are satisfied by its performance. The test cases are run on the system to ensure that the system provides the same results as would a human expert in the field.

There are several methods to make inferences from the given rules, but forward chaining and backward chaining are the most commonly used ones [10]. Forward chaining, as the name suggests, starts with the facts and deduces the conclusion by applying rules to the facts. On the other hand, backward chaining involves reasoning in the opposite direction. It starts with the hypothesis and tries to induce the facts to support the hypothesis.

An expert-based system is the simplest example of rule-based systems that has been applied to the selection of optimal portfolio [6]. However, portfolio management involves numerous tasks that, in real life, would not be performed by a single expert. To better simulate the behavior of human experts, a single expert systems have been used as a base for development of multi-agent systems. Multi-agent systems simulate tasks of several experts and combine their expertise to make a final decision [29]. This kind of system allows communication among temporally and spatially separated experts, which is why they have found application in lots of different areas.

4.2 Applications to portfolio management

The first attempt to design an expert system to assist portfolio managers is described in [6]. The basic idea of this system, called *Folio*, is to interview a user and use an expert's knowledge to first determine the user's investment goals and then build the portfolio that best suits the situation. The algorithm consists of three steps: the interview of the investor, the inference of the goals of the investor, and the optimization of distribution of wealth to reach goals of the investor.

The interview contains a set of questions that help the expert to derive the correct goal of the investor. The simple questions determine the user's acceptance of the level of risk, the desired return, and the user's tax bracket among others. Based on the obtained answers, the algorithm infers the rules of a user's goal, and the rules are used to determine the goals of the investor.

Folio recognizes 14 different goals for investment including acquiring a required level of return, reducing risk by investing into different assets, and minimizing risk while attaining the desired return. Each goal is characterized by five parameters: a target value, a penalty for exceeding the target value, a penalty for falling short of the target value, a lower bound under which the penalty becomes infinite, and an upper bound above which the penalty becomes infinite. These parameters are established from the inferred rules. About 50 rules (derived from interview) are used to infer one or more parameters of the goals. Sometimes, a parameter has more than one possible value, in which case a heuristics is used to determine the most certain value.

When the goal and its parameters are specified, *Folio* uses a goal programming algorithm to determine the distribution of wealth among assets that best fits the goal parameters. The goal programming algorithm used by *Folio* is a linear programming solver whose objective function is set to calculate the differences between the user's target values and the obtained values for each of the parameters. The algorithm minimizes the sum of the deviations of obtained values from desired values. The optimal wealth distribution among classes of assets is suggested. The algorithm considers nine classes of assets which include different levels of low-risk to high-risk assets. However, the distribution of wealth among each class is not given by this algorithm for several reasons. First, this method does not require *Folio* to consider thousands of investment assets that exist in the market and therefore, reduces the complexity of the algorithm. Second, *Folio* requires only the aggregate knowledge about the properties of each asset class and not the knowledge of individual assets. Moreover, the aggregate values change less slowly than the individual asset's characteristics, so *Folio* stays current for longer time period. Finally, picking the exact assets for the investment is the responsibility of an investment advisor and not *Folio*.

Even though performance of *Folio* has not been tested on real data, this algorithm is the foundation for the further development of expert based systems which evolved into multi-agent systems for portfolio management. The advantages of multi-agent systems (MAS) over the single-agent systems include [29]:

- The ability to avoid performance bottlenecks due to one stage in the multi-stage process.
- Possibility for interconnection and interoperation of multiple systems.
- Natural distribution of tasks among different agents.
- Possibility to connect spatially and temporally distributed experts.
- Enhancement of overall system performance including reliability, computational efficiency, maintainability, flexibility, and reuse among others.

A multi-agent system for portfolio monitoring, called *Warren*, was developed in [30] and further improved and implemented in [29]. *Warren* was designed to monitor portfolio rather than manage it. Monitoring portfolio is a continuous picture of an existing portfolio, which helps to determine if reallocation of assets is necessary, but does not suggest how to redistribute the wealth. The goal of *Warren* is to provide an overall picture of the existing portfolio based on the numerous available data from different sources.

Warren is composed of several types of agents: interface, task, middle, and information agents. The interface agent, *Warren Interface*, communicates with investor. This type of agent interviews the user and collects all necessary data that determine the goals of the investor. It also displays a comprehensive summary of the user's current portfolio and allows the user to buy and sell assets. The middle agent, *MatchMaker*, helps match agents that request services with agents that provide those services.

The task agents, *RiskCritical* agent and *Comptroller*, perform tasks. The tasks are performed by formulating problem-solving plans and carrying them out in collaboration with other agents. *RiskCritical* agent calculates the risk of the portfolio, while *Comptroller* agent is in charge of buying and selling assets.

The information agents monitor and collect financial information about companies of interest when requested by a middle agent. Warren contains four information agents: *FdsHistory* agent, *iYahooStocks*, *iEdgar*, and *TextMiner*. *FdsHistory* agent provides a historical view of financial data summary, *iYahooStocks* provides stock prices, *iEdgar* provides financial data summaries from SEC's Edgar web site, and *TextMiner* provides financial news analysis. *FdsHistory*, *iYahooStocks*, and *iEdgar* provide quantitative data about companies of interest, while *TextMiner* provides qualitative data available from numerous news agencies.

TextMiner is designed as a text classification agent to sort data available from a high volume of news reports about financial assets since only the useful details should be considered when monitoring portfolio. *TextMiner* sorts the news from Reuters, CNN, Business Wire, etc. by first separating financial from non-financial news in articles. The financial news cover the reports on company's earnings, movements of stock price, revenues, and similar information, while the news about corporate control and legal and regulatory issues are considered non-financial. To separate financial from non-financial data, *TextMiner* was trained on a set of 1,239 news articles, which were labeled manually. The selection process is based on the weighting of commonly used terms (words or phrases) in the following way. First, each news article is represented in a high-dimensional space, where each dimension corresponds to a term. Then, the set of news articles is represented by the term-by-document matrix $M = T \times N$, where T is the number of terms and N is the number of articles. The set of terms $T = \{t_1, \ldots, t_t, \ldots, t_T\}$ is constructed by eliminating the words whose frequency is higher than *frequent threshold* (words that are considered to be just features) and the words whose frequency is lower than *infrequent threshold*. Each term has its weight w_t, which indicates how important the term is for the given document. All the weights are scaled from 0 to 1 with the higher weight being given to terms that appear often in one article but less frequently in other documents. Precisely, the weight of a word is determined by

$$w_t = \frac{(1 + \log(f_{it})) \cdot \log\frac{N}{d_t}}{\sqrt{\sum_{s \neq t}(log(f_{is}) + 1)^2}} \tag{17}$$

where f_{it} is the number of times the term t occurs in the document i, and d_t is the number of documents in which the word t occurs. The weight is normalized by the document's length. After the weights for each term are determined, the article d is compared to one of the classes, $C = \{\text{financial}, \text{non-financial}\}$. A class is determined by the mean vector of all documents in the class,

$$\mathbf{c} = \frac{1}{|c|} \sum_{d \in c} \mathbf{d}, \tag{18}$$

and the calculation of similarity is the measure of the cosine of the angle between the class vector and the document vector

$$s(\mathbf{d}_i, \mathbf{c}_j) = \arg\max_{c_j \in C} \frac{\mathbf{d}_i \cdot \mathbf{c}_j}{||\mathbf{d}_i|| \cdot ||\mathbf{c}_j||}. \tag{19}$$

When financial news are separated from non-financial news, they are classified into one of five groups: good, good–uncertain, neutral, bad–uncertain, bad. Here the 'good' news are the ones clearly showing a company's good financial standing whereas 'bad' news are the ones clearly showing the bad financial standing of a given company. 'Neutral' news mention financial facts but do not give enough information to determine whether the facts indicate the good or the bad financial state of a given company. Two 'uncertain' categories refer to the prediction of future behavior of the company. The classification into one of five classes is performed by co-locating phrases, that is looking for words in the article that are usually in the same order in a sentence but not necessarily next to each other, such as 'earning' and 'increased'. The selection of useful co-located phrases is based on the training set of data.

Finally, a step-by-step description of the performance of *Warren* follows. First, the *MatchMaker* initializes the virtual work-space for agent-naming and resources for *Warren*, and all the other agents register their services with *MatchMaker*. The *Warren Interface* displays the current portfolio of the investor, and allows the user to buy/sell assets. If the investor requests the financial information about a particular company, the interface agent sends the request to the middle agent, and the middle agent invokes information agents to provide requested information. The information agents look for the information on the requested company and provide it to the interface agent. Warren Interface displays the gathered information and the *RiskCritical* agent calculates the risk of new portfolio. Finally, the *Comptroller* agent updates the investor's portfolio if he/she decides to buy/sell an asset.

Even though the entire model has not been tested on real-life data, *TextMiner* showed great results when compared to traditional Bayesian approaches to classify articles. With this in mind, *Warren* gives a promising tool for portfolio monitoring.

5 Neural networks

Neural networks provide yet another approach to identify an optimal portfolio. Thus, we touch the basics of neural networks that are necessary for the understanding of several examples that use this type of computational intelligence techniques in a portfolio selection process.

5.1 Theoretical background

An artificial neural network or just neural network (NN) is designed to imitate the actions of human neural system, which consists of neurons and axons (the links between neurons) (see e.g.,[31]). Similarly, a neural network consists of nodes and directed links between nodes. NN is based on ability to learn from training data sets in order to perform accurately on real data.

Several types of neural networks exist, the simplest one being the perceptron.

Definition 4.1. (Perceptron) *The perceptron is a neural network that consists of two layers of nodes: input and output layers. The input nodes,* $\mathbf{x} = (x_1, \ldots, x_n)$*, represent the input values, and the output nodes,* $\mathbf{y} = (y_1, \ldots, y_m)$*, carry out mathematical calculations and output the results.*

The function used to calculate the outputs is called the activation function. Most commonly used activation function in a perceptron is the *sign* function:

$$\hat{y} = sign\left(\sum_{i=1}^{n} w_i x_i\right) = sign(\mathbf{w} \cdot \mathbf{x}), \tag{20}$$

where $\mathbf{w} = (w_1, \ldots, w_n)$ is the vector of weights assigned to the links from input to output nodes. The weights represent the strength of the connection between the nodes and are determined by a learning process using the training data set for which the expected outcome is known. The weights are updated after each training example by

$$w_j^{(k+1)} = w_j^{(k)} + \lambda\left(y_i - \hat{y}_i^{(k)}\right) x_{ij}, \tag{21}$$

where $w^{(k)}$ is the weight of the i^{th} input link after the k^{th} iteration, x_{ij} is the value of the j^{th} attribute of the training example $\mathbf{x}_i$, and λ is the learning rate that is determined by user. The value of λ belongs to interval $[0, 1]$ and is designed to control the amount of adjustment after each training sample. The learning rate is either a constant that stays small throughout the entire training process to avoid overfitting to a specific training data element or the λ is adaptable in which case it starts with a large value but the size gets smaller during the training process.

The perceptron model is the simplest kind of neural network and is used only for classification purposes. However, more complex multilayer networks are much more powerful and applicable to other types of problems.

Definition 4.2. (Multilayer network) *A multilayer network is a neural network that contains one or more hidden layers of nodes that perform calculations and influence more accurate weight adjustments.*

In a multilayer network, the links between nodes can go either only from a lower layer to a higher layer (input being the lowest layer and output the highest layer), which is the case in feed-forward networks, or the links can connect nodes in the same layer or even be directed towards the previous layers, which is the case in recurrent networks. The multilayer networks can use different activation functions, such as linear, sigmoid, and hyperbolic tangent function among others. These functions allow more complex situations to be modeled by multilayer networks.

A neural network learning algorithm works by minimizing the sum of squared errors:

$$Err(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{22}$$

where $\hat{y}$ depends on $\mathbf{w}$. If $\hat{y}$ is replaced by $\mathbf{w} \cdot \mathbf{x}$, then the error function becomes quadratic in its parameters, and a global minimum can be easily found. However, if a non-linear function is used as an activation function, hidden and output layers produce non-linear outputs, so finding the solution for $\mathbf{w}$ is harder. Usually this problem is solved using a gradient descent method, which basically increases the weights in a direction that reduces the overall error function:

$$w_j^{k+1} = w_j^k - \lambda\frac{\partial Err(\mathbf{w})}{\partial w_j}, \tag{23}$$

where λ is the learning rate. This method can be successfully used to learn weights for the output layer. However, it might not be as easy to perform the computation for hidden layer since it is not possible to know their error term, $\frac{\partial Err}{\partial w_j}$, without knowing what their output values should be. To solve this problem, backpropagation algorithm is used, which forces two phases in each iteration of the algorithm: the forward and the backward phases. In the forward phase, the weights computed in the previous iteration are used to compute the outputs of each node in the network and the computations follow in forward direction. In the backward phase, the weights are updated in the reverse order–the weight update formula is applied to the last layer first, and then for each previous layer one-by-one going towards the first layer, which allows the use of output at the next level to compute the error at the previous layer.

5.2 Applications to portfolio management

Neural networks have found several applications in portfolio management ([2],[4],[20],[33],[34]) ranging from forecasting the behavior of investment assets to optimizing the distribution of wealth among assets.

Lowe [20] used analog NN to find the optimal distribution of wealth among investment assets. The optimal portfolio is constructed by minimizing the risk function defined by

$$\text{risk} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left[y(t) - \sum_{i=1}^{m} w_i x_i(t)\right]^2}, \tag{24}$$

where m is the number of assets, T is the number of iterations, $y(t)$ is the market portfolio return, $x_i(t)$ is the return of the asset i at time t, and w_i is the proportion of wealth invested into the asset i. The risk function is subject to a non-negativity constraint of the weights $w_i \geq 0$ for every asset i and the normalization constraint $\sum\limits_{i=1}^{m} w_i = 1$. This linear constraint optimization problem could be transformed into a nonlinear unconstrained optimization problem that minimizes the cost function

$$E = \frac{1}{T}\sum_{t=1}^{T}\left[y(t) - \sum_{i=1}^{m} w_i x_i(t)\right]^2 + \lambda\left[\sum_{i=1}^{m} w_i - 1\right]^2 + \mu\sum_{i=1}^{m}\frac{1}{1+e^{\beta w_i}}. \tag{25}$$

The first term of this equation corresponds to minimizing the risk; the second term replaces the normalization constraint; and the third term transforms non-negativity constraint into a barrier potential term, which has the form of a logistic used in multilayer perceptron studies. The parameters λ and μ are penalties for breaking constraints, and could be adjusted based on investors preferences.

The minimization of the cost function could be performed by using any standard gradient based method, one of them being Runge-Kutta integration algorithm with a possibility to adapt step size based on the results form previous iteration. The performance of analog neural network in portfolio management was tested on seven stocks in the electricity sector of the UK market for 160 consecutive days starting at 26$^{\text{th}}$ of September 1989.

Another application of neural networks in portfolio management was described by Casas [2] to predict which of three considered classes of assets will outperform the other two. The three classes in consideration are: stocks, bonds, and money markets. The idea is to invest all wealth into one class of assets for a given time interval, and then re-evaluate the performance of the asset classes and make a new decision for the next time interval. This approach does not diversify the portfolio to reduce risk, and is based purely on the return of three classes of assets rather than performance of individual assets.

A neural network, that uses fundamental factors such as earnings, price per earning ratios, interest rates, and inflation, as input values, is trained with backpropagation algorithm to predict behavior of three classes of assets. The relative performance of classes of assets is measured by the risk premium. The risk premium between two asset classes A and B is calculated as

$$\Gamma_{AB} = E(A) - E(B), \tag{26}$$

where $E(x)$ is the expected return of the class x. Assuming that risk premium follows normal distribution, the probability that class A outperforms the class B is given by

$$P(A > B) = CND(\Gamma_{AB}, \mu_{AB}, \sigma^2_{AB}), \tag{27}$$

where CND is cumulative normal distribution function, μ_{AB} is mean risk premium, and σ^2_{AB} is standard deviation of risk premium. The algorithm calculates the probabilities that stocks will outperform bonds, bonds will outperform money markets, and stocks will outperform money markets.

The performance of this algorithm was tested against a buy-and-hold strategy that buys and holds S&P500 Index for the entire time period under consideration, which was 12 months in this case study for the year 1999. The NN approach outperformed the buy-and-hold strategy at the end of 12 months. Moreover, it predicted correctly 92% of the time which asset class would outperform the other two classes.

Another example of forecasting ability of NN was tested in [34]. In this paper, the authors presented a portfolio management algorithm that consists of three parts. The first part uses error correction neural network (ECNN) to forecast the behavior of assets. The second step uses a higher-level feedforward network to compute the excess return of one asset over another asset. Finally, the third part determines the optimal wealth distribution based on the excess returns.

Forecasting the behavior of each asset in the future is based on the expected return of the asset, which depends on the previous state of the asset, s_t, external influences, u_t, and the difference between the predicted output, y_t, and the observed output, y_t^d, at the previous iteration. Thus,

$$s_{t+1} = f(s_t, u_t, y_t - y_t^d), \tag{28}$$

where $y_t = g(s_t)$ is determined based on the current state. In the suggested model, the expected return is predicted based on error correction neural network, which uses weight matrices of appropriate dimensions, $A, B, C,$ and D, to transform the problem into the following set of equations:

$$s_{t+1} = \tanh(As_t + Bu_t + D\tanh(Cs_t - y_t^d)) \tag{29}$$

$$y_t = Cs_t. \tag{30}$$

The optimization of parameters is obtained by finite unfolding in time using shared weights, which solves

$$\min_{A,B,C,D} \frac{1}{T} \sum_{t=1}^{T} (y_t - y_t^d)^2 \tag{31}$$

After the parameters are established by an ECNN, the expected return is calculated for each asset, f_i. Next, the difference between expected returns of two assets is calculated for each pair of assets, $e_{ij} = f_i - f_j$. Finally, the cumulative excess return of each asset is calculated as weighted sum of excess returns,

$$e_i = \sum_{j=1}^{k} w_{ij} e_{ij}, \tag{32}$$

where $w_{ij} \geq 0$ is the assigned weight to the pair (i, j) of assets. Based on cumulative excess returns, the proportion of wealth that should be invested into the asset i is calculated by

$$a_i = \frac{e^{e_i}}{\sum\limits_{j=1}^{k} e^{e_j}} = a_i(\mathbf{w}, f_1, \ldots, f_k). \tag{33}$$

This form guarantees that exactly all wealth is distributed ($\sum a_i = 1$) and the proportions of investment are non-negative ($a_i \geq 0$).

However, there are sometimes market share constraints given by the asset manager, and they are usually given in the form of an interval with a lower bound and an upper bound. If the mean of the available allocation for asset i is denoted by m_i, and the admissible spread is given by Δ_i, then the proportion a_i should fall into the interval $[m_i - \Delta_i, m_i + \Delta_i]$. The vector of means, $\mathbf{m} = (m_1, \ldots, m_k)$, is used as a benchmark distribution. To comply with the requirements of the manager, the excess return is adjusted by a bias vector $\mathbf{v}$ so that

$$e_i = v_i + \sum_{j=1}^{k} w_{ij}(f_i - f_j), \tag{34}$$

where the vector $\mathbf{v} = (v_1, \ldots, v_k)$ could be determined in advance by setting the excess returns to zero and solving the system of nonlinear equations

$$m_1 = a_1(v_1, \ldots, v_k)$$

$$\vdots$$

$$m_k = a_k(v_1, \ldots, v_k). \tag{35}$$

The non-unique solution of the form

$$v_i = \ln(m_i) + c \tag{36}$$

could be simplified by setting $c = 0$.

Since the interval $[m_i - \Delta_i, m_i + \Delta_i]$ represents a constraint for parameters $w_{i1}, ..., w_{ik}$, the optimal portfolio selection defined as the return maximization problem can be solved by solving a penalized maximization problem

$$\max_{\mathbf{w}} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{k} \left[r_{it} a_i(f_{it}, \ldots, f_{kt}, \mathbf{w}) - \lambda ||a_i - m_i||_{\Delta_i} \right], \tag{37}$$

where r_{it} is the actual return of the asset i at time t and

$$||x||_{\Delta} = \begin{cases} 0 & \text{if } |x| \leq \Delta \\ |x| - \Delta & \text{otherwise} \end{cases} \tag{38}$$

The proposed model was tested on the basis of monthly data of G7 countries markets. The data from September 1979 to June 1993 was used to train network, and based on produced coefficients, the model was tested in the period July 1993 to May 1995. The results showed that the neural network based model outperformed the benchmark model by almost 10%.

The modification of asset allocation step of this algorithm is presented in [33] and shows how to incorporate the risk of investing into selected assets rather than determining the optimal portfolio only based on the return. The authors use an ECNN (developed in [34]) to forecast the return of assets, r_i, which is used to calculate the risk-adjusted expected excess return rather than the expected return that does not consider risk related to the assets. The risk-adjusted excess return is defined by

$$\rho_i = \sum_t \frac{r_{it} - r_f}{|r_{it} - r_{it}^d|}, \tag{39}$$

where r_f is the risk-free asset return and r_{it}^d is the actual return at time t. Based on the risk-adjusted excess returns, the assets are ranked from the highest to the lowest, and all assets whose risk-adjusted excess return is higher than a pre-defined threshold value ρ^* are selected to be included into the portfolio.

If we denote the set of assets included in portfolio by A, the proportion of wealth invested in each of the selected assets is determined by

$$w_i = \frac{\rho_i}{\sum_A \rho_j}. \tag{40}$$

This model was tested on the German stock market by using weekly data from 68 stocks on a period ranging from November 1994 to June 1999, in order to train the neural network considered. The algorithm's performance was tested on data from July 1999 to June 2000. Four portfolios were built with different number of stocks included: 5, 10, 15, and 20 stocks. The results of the algorithm were compared to the performance of the benchmark, which included all 68 stocks whose weights were chosen based on the shares of the stocks in the market. All portfolios produced by the NN algorithm outperformed the benchmark portfolio. Among the four derived portfolios, the portfolio with the smallest number of assets performed better than all the other portfolios.

Finally, [4] shows another application of NN as a forecast model as well as a decision model for wealth allocation. The multilayer perceptron (MLP) with one hidden *tanh* layer (with H hidden units) and a linear output layer is considered. The function represented by MLP is given by

$$f(\mathbf{x}; \theta) = A_2 \tanh(A_1 \mathbf{x} + b_1) + b_2, \tag{41}$$

where $\mathbf{x}$ is the current distribution of wealth among assets, A_1 is an $H \times M$ matrix (with M being the dimension of the input vector $\mathbf{x}$), A_2 is an $N \times H$ matrix (with N being the dimension of the output vector), b_1 is an H-element vector, b_2 is an N-element vector, and $\theta = (A_1, A_2, b_1, b_2)$ is the vector of parameters. The parameters represented by the vector θ are found by training the network to minimize a cost function; the cost function differs for two types of the model–forecast and decision model. The optimization is performed by using a conjugate gradient descent algorithm. The gradient of the parameters with respect to cost function is computed using the backpropagation algorithm for MLP.

In the forecast model, a neural network is used to predict the returns of assets in the next time period, $\mu_{t+1|t}$, given explanatory variables u_t, which belong to the set of the available information, I_t. The network is trained to minimize the prediction error of returns of assets in the next time period by using a quadratic loss function

$$C_F(\theta) = \frac{1}{T} \sum_{t=1}^{T} ||f(u_t; \theta) - r_{t+1}||^2 + C_{WD}(\theta) + C_{ID}(\theta), \tag{42}$$

where $||\cdot||$ is the Euclidian distance, $f(\cdot; \theta)$ is the function computed by MLP, and $C_{WD}(\theta)$ and $C_{ID}(\theta)$ are terms used for regularization purposes. The regularization is needed to prevent overfitting by specifying *a priori* preferences on weights in the neural network. $C_{WD}(\theta)$ is the weight decay. It tries to reduce magnitude of weights in the network by setting a penalty on the squared norm of all network weights. On the other hand, $C_{ID}(\theta)$ is the input decay. It tries to utilize useful inputs to train the network by penalizing the inputs that turn out to be unimportant.

The neural network decision model uses the NN to directly determine the distribution y_t of wealth among assets based on the explanatory variables u_t. NN is trained to minimize the negative of the financial performance evaluation criterion

$$C_D(\theta) = -\frac{1}{T}\sum_{t=1}^{T} W_t + C_{WD}(\theta) + C_{ID}(\theta) + C_{\text{norm}}, \tag{43}$$

where C_{norm} is a preferred norm of the neural network. The preferred norm is important since two vector solutions that differ only by a constant multiple would be considered as different solutions without the use of preferred norm. The result would be that, for each vector θ, there would be a direction with (almost) zero gradient, so there would be no local minimum. The preferred norm variable, which is given by user, re-scales the parameters so that the norm constraint is achieved.

Training MLP for the decision problem is more complex than for the forecast model. It includes a feedback loop, which induces a recurrence by inputting the distribution y_{t-1} to determine the output y_t. Also the backpropagation through time algorithm is used to compute the gradient by going back in time, starting from the last time step until the first one.

Sometimes, the user has an idea of the optimal portfolio or has *a priori* preferences of the portfolio structure (i.e. the proportion of wealth invested into stocks versus the proportion invested into bonds). In this case, instead of the preferred norm, the preferred portfolio is considered. Deviation from the preferred portfolio is penalized by

$$C_{\text{ref.port.}} = \frac{1}{T}\sum_{t=1}^{T} \text{penalty}_{\text{ref.port.}}(y_t), \tag{44}$$

where penalty is calculated as the squared distance between the network output and the reference portfolio.

For testing purposes, Toronto Stock Exchange market data from January 1971 to July 1996 were used, and the results proved to outperform the benchmark algorithms. It was also shown that the decision model is preferred to the forecast model as it relies on fewer assumptions.

6 Support Vector Machines

Finally, we give a general description of support vector machines and their application to portfolio selection.

6.1 Theoretical background

Support vector machine (SVM) is one of the most commonly used classification techniques (see e.g.,[31]). It classifies data into one of two groups by constructing a hyperplane that separates these two groups. The simplest situation is when the data are linearly separable. In this case, usually more than one hyperplane could be constructed to represent the boundary between two classes that will result in a zero error. However, instead of minimizing the empirical error (or error produced by training data), the best hyperplane should minimize the generalization error, that is the error that could result from classifying real data based on the model developed from the training set. To explain how to minimize the generalization error, we first define the margin hyperplanes. We consider a hyperplane b and create two other hyperplanes, b_1 and b_2, such that they are parallel to b and as far as possible from b (going into opposite direction from b) so that they do not touch any training data element. The distance between the hyperplanes b_1 and b_2 is called the margin of hyperplane. Since several non-parallel hyperplanes usually exist in the linearly separable case, we select the pair of parallel hyperplanes that yield the highest margin of hyperplane. The decision boundary is represented by the hyperplane going straight through the middle between two selected margin hyperplanes.

To formally define the best hyperplane, we consider a set of N training examples, each of them denoted by $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$ corresponds to the attribute set for the i^{th} training example and $y_i \in \{-1, 1\}$ is the class label. Given this notation, the decision boundary is given by $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ and b are the parameters of the model which are determined through training. Based on the calculated parameters, the decision for a new data sample $\mathbf{z}$, which is not in the training set, is determined by

$$y = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{z} + b > 0 \\ -1 & \text{if } \mathbf{w} \cdot \mathbf{z} + b < 0 \end{cases} \tag{45}$$

Since the margin hyperplanes are defined as

$$\mathbf{w} \cdot \mathbf{x} + b = \pm 1, \tag{46}$$

each training data sample satisfies the conditions

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1 \tag{47}$$

and

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \text{ if } y_i = -1. \tag{48}$$

These two conditions could be simplified to

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1. \tag{49}$$

Furthermore, we denote the margin hyperplanes by $\mathbf{w} \cdot \mathbf{x} + b = +1$ and $\mathbf{w} \cdot \mathbf{x} + b = -1$, which implies that the margin, d, of the decision hyperplane is $d = \frac{2}{||\mathbf{w}||}$. To simplify calculations necessary to find the best hyperplane,

$||\mathbf{w}||$ is usually replaced by $||\mathbf{w}||^2$. Thus, maximizing the margin is equivalent to minimizing

$$f(\mathbf{w}) = \frac{||\mathbf{w}||^2}{2}. \tag{50}$$

We can formally define the objective of the learning process in SVM training phase as follows:

Definition 5.1. (Linear SVM: separable case): *The learning task in SVM can be formalized as the following constraint optimization problem:*

$$\min_{\mathbf{w}} \frac{||\mathbf{w}||^2}{2} \tag{51}$$

$$\textit{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \;\; \forall i = 1, 2, \ldots, N. \tag{52}$$

This problem of solving for $\mathbf{w}$ and b is a convex optimization problem (since the objective function is quadratic and the constraints are linear) that could be solved by using the standard Lagrange multiplier method, which rewrites the objective function in terms of Lagrangian

$$L_P = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \lambda_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1), \tag{53}$$

where the parameters λ_i are called the Lagrange multipliers. The first term tries to minimize the objective function, while the second term replaces the constraint and must be minimized in order to reduce the penalty of not satisfying the constraint. When solving for the Lagrange multipliers, many of them are equal to zero. However, a few Lagrange multipliers that are non-zero correspond to the training examples that lie exactly on one of the margin hyperplanes and thus represent support vectors, which are used to find the values of $\mathbf{w}$.

The Lagrangian problem could be transformed into a dual problem that involves finding only Lagrange multipliers. The problem maximizes the dual Lagrangian

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j, \tag{54}$$

where the Lagrangian multipliers must be non-negative.

The solution to this problem can be found using numerical techniques such as quadratic programming. The solution for $\mathbf{w}$ is calculated by

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \tag{55}$$

and b is obtained by solving

$$\lambda_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b)] = 0. \tag{56}$$

The decision boundary can be expressed as

$$\left(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x} \right) + b = 0. \tag{57}$$

The previous description to find the optimal decision boundary works well if the training data is linearly separable. However, it is not always the case. Very often, any decision boundary would misclassify some training examples. The problem could be approached by introducing positive slack variables ξ_i that represent the error of the decision boundary for the training sample i [7]. Thus, the new objective function

$$f(\mathbf{w}) = \frac{||\mathbf{w}||^2}{2} + C \left(\sum_{i=1}^{N} \xi_i \right) \tag{58}$$

tends to minimize the error besides minimizing the original objective function. Here C represents the penalty for misclassification, and is determined by user. The new objective function and the inequality constraints

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 - \xi_i \text{ if } y_i = 1,$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq 1 + \xi_i \text{ if } y_i = -1, \tag{59}$$

could be easily transformed into the Lagrangian where each Lagrangian value is bounded above by the value of the parameter C:

$$0 \leq \lambda_i \leq C. \tag{60}$$

This problem could be approached by using quadratic programming.

In some instances, however, a better solution exists than reducing the misclassification. A non-linear decision bound might exist to correctly classify training data that are not separable by linear method. The idea is to transform the original coordinates of the training sample $\mathbf{x}$ into a new space $\Phi(\mathbf{x})$ so that a linear decision bound can be used to correctly separate data in the new space. The problem with this approach is to determine the mapping that will lead to desired results. Now, the problem of learning from training data becomes:

Definition 5.2. (Nonlinear SVM): *The learning task in a non-linear SVM can be formalized as the following constraint optimization problem:*

$$\min_{\mathbf{w}} \frac{||\mathbf{w}||^2}{2} \tag{61}$$

$$\text{subject to } y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1 \;\; \forall i = 1, 2, \ldots, N. \tag{62}$$

The attempt to solve this problem by transforming it into Lagrangian is usually not easy due to need for calculation of the dot product between the new spaces $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$, which might be very complicated. However, since the dot product is a measure of similarity between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$, we can solve this problem by applying the kernel trick, which computes the similarity between two instances in the transformed space by using the original attribute set [1]. The kernel function

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{63}$$

is the function that calculates the similarity of instances $\mathbf{x}_i$ and $\mathbf{x}_j$ by using the attributes in the original space, which simplifies the computation of the dot product. The use of kernel trick also does not require the knowledge of the exact transformation Φ because the kernel function used in non-linear SVM must satisfy Mercer's theorem:

Theorem 5.3. (Mercer's theorem): A kernel function K can be expressed as $K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$ if and only if, for any function $g(x)$ such that $\int g(\mathbf{x})^2 d\mathbf{x}$ is finite, then $\int K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$.

Support vector machine represents the most commonly used classifier that has found implementation in different fields, one of them being the portfolio management.

6.2 Applications to portfolio management

Support Vector Machine (SVM) has found implementation in classification of stocks into one of two classes–the stocks with exceptional high returns (Class+1) and the stocks with unexceptional returns (Class-1) [9]. SVM tries to minimize a bound on the generalization error rather than the empirical error as many other approaches do. It uses several financial indicators to determine the performance of each asset. The n financial indicators of the asset i are represented as a vector $\mathbf{x}_i = (x_1, ..., x_n)$. The expected future return of the stock is a binary dependent variable $y_i = \pm 1$, where $+1$ represents the Class+1 asset and -1 represents the Class-1 asset. Thus, the training set of m companies consists of pairs $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\} \subset R^N \times \pm 1$. The classifier (SVM) tries to learn from the training set, and it behaves as a function that maps the input variables $\mathbf{x}$ into an output value y. The misclassification is reduced by adjusting parameters.

SVM is a classifier that tries to construct an optimal separating hyperplane between two classes by minimizing the bound on the misclassification risk. To solve linearly separable patterns, traditional approach using quadratic programming is utilized to maximize the dual Lagrangian.

In the case of non-separable patterns, different kernel functions could be used. In the test case, the Radial Basis Kernel, $K(\mathbf{x}, \mathbf{y}) = exp(-||\mathbf{x} - \mathbf{y}||^2)$, was used.

The method was tested with Australian Stock Exchange data between 1992 and 2000. The data from three years were used for training and validation of estimated parameters that were then used to predict the performance of stocks in the next year. Eight groups of financial indicators were used to calculate the performance of stocks: Return on Capital, Profitability, Leverage, Investment, Growth, Short term Liquidity, Return on Investment, and Risk. The data for each stock were converted into an eight-element input vector. For the training samples, the output of each stock was determined by its annual actual performance with the top 25% of stocks being selected into Class+1, and the remaining stocks being assigned Class-1. For testing purposes, the stocks selected into Class+1 group were given equal weights in the portfolio. The created portfolio's return outperformed the equally weighted portfolio consisting of all available stocks, which was used as the benchmark portfolio.

Summary of approaches. We have presented several attempts to use intelligence systems in portfolio management. Genetic algorithms, rule-based expert systems, neural networks, and support vector machines have all contributed towards finding the optimal distribution of wealth among available assets. With the exception of genetic algorithm, all other methods are based on the ability to learn from examples and approximation of algorithm's parameters due to training samples. This could lead to overfitting of the parameters to specific type of data or a specific sample, which might not be applicable in other situations.

Moreover, all of the approaches do not consider the relationship between the characteristics of an asset. For example, return and risk are known to usually move in the same direction, that is higher the return of the asset, higher the risk of that asset. However, the presented approaches do not take into consideration this and many other existing relationships.

Furthermore, the return, the risk, and other characteristics of an asset are assumed to be precisely known for each asset in consideration. In reality, this is not always the case as the best we can do is to predict the future return and risk. Sometimes, these predictions are not correct, but all the presented techniques rely on precise knowledge of these values.

To face the drawbacks of the presented approaches, we propose a new approach to portfolio optimization. The novel approach is based on multi-criteria decision making and fuzzy integration over intervals.

7 Fuzzy integration and decision making

Before going into details of portfolio optimization problem, we review basics of fuzzy measures, fuzzy integration, and multi-criteria decision making.

7.1 Theoretical background

A multi-criteria decision making (MCDM) problem seeks the optimal choice among a (finite) set of alternatives. It can formally be defined as a triple $(X, I, (\succeq_i)_{i \in I})$ where

- $X \subset X_1 \times \cdots \times X_n$ is the set of alternatives with each set X_i representing a set of values of the attribute i.
- I is the (finite) set of criteria (or attributes).
- $\forall i \in I$, $\succeq_i$ is a preference relation (a weak order) over X_i.

The next task is to "combine" the preference relations $\succeq_i$ of an alternative into a global value for the alternative such that the final order of the alternatives is in agreement with the decision maker's partial preferences. The natural way to construct a global preference is by using utility function for each attribute to reflect partial preferences of a decision-maker, and then combine these monodimensional utilities into a global utility function using an aggregation operator. The utility functions $u_i : X_i \to \mathbb{R}$ such that for all $x_i, y_i \in X_i$, $u_i(x_i) \geq u_i(y_i)$ if and only if $x_i \succeq_i y_i$, scale the values of all attributes onto a common scale. The existence of monodimensional utility functions is guaranteed under relatively loose hypotheses by the work presented in [17].

Numerous aggregation operators could be used to combine monodimensional utilities into a single number that represents the value of an alternative. Two simple approaches that correspond to optimistic and pessimistic behavior of the decision maker are maximax and maximin strategies, respectively, assuming that the goal of a decision-maker is to maximize the utility. The maximax method compares the utilities of all attributes of an alternative and chooses the highes utility value, $\max\limits_i u_i(x_i)$, to represent the global utility of the alternative $\mathbf{x} = (x_1, ..., x_n)$. This approach reflects the optimistic behavior of the decision-maker since he/she is concerned only with the attribute that has the highest utility for the given alternative. The maximax method tries to maximize the best criterion:

$$\max_{\mathbf{x} \in X} \left(\max_{i \in \{1,\ldots,n\}} u_i(x_i) \right). \tag{64}$$

On the contrary, the maximin method reflects the pessimistic behavior of the decision-maker as the decision-maker is concerned only with the attribute that could result in the worst value. This method compares the utilities of all attributes of an alternative and chooses the lowest utility value, $\min\limits_i u_i(x_i)$, to represent the global utility of the alternative $\mathbf{x}$. The decision-maker tries to maximize the value of the worst case scenario:

$$\max_{\mathbf{x} \in X} \left(\min_{i \in \{1,\ldots,n\}} u_i(x_i) \right). \tag{65}$$

To allow for a position between these extremes when making a decision, a simple combination of maximax and maximin approaches is achieved by a weighted aggregation operator

$$\max_{\text{allalternatives}} (\alpha \max_i u_i(x_i) + \beta \min_i u_i(x_i)), \tag{66}$$

with $\alpha + \beta = 1$ where $\alpha \geq 0$ and $\beta \geq 0$ are weights given by the decision maker. This approach simplifies to the optimistic case if $\alpha = 1$ and to the pessimistic case if $\beta = 1$.

These simple approaches are very tempting to use for quick decisions. However, they focus only on a few criteria and ignore the impact of other characteristics of alternatives, which often does not suit the situation. Thus, we usually need to consider more complex aggregation operators that take into consideration all attributes. The simplest and most natural of them is a weighted sum approach, in which the decision-maker is asked to provide weights, w_i, that reflect the importance of each criterion. Thus, the global utility of alternative $\mathbf{x} = (x_1, ..., x_n) \in X$ is given by

$$u(\mathbf{x}) = \sum_{i=1}^{n} w_i u_i(x_i). \tag{67}$$

The best alternative is the one that maximizes this value. Even though this approach is attractive due to its low complexity, it can be shown that using an additive aggregation operator, such as weighted sum, is equivalent to assuming that all the attributes are independent [22]. In practice, this is usually not realistic and therefore, we need to turn to non-additive approaches, that is to aggregation operators that are not linear combinations of partial preferences.

Before approaching non-additive methods, we give the definition of a non-additive measure, a tool for building non-additive aggregation operators.

Definition 6.1. (Non-additive measure) *Let I be the set of attributes and $\mathcal{P}(I)$ the power set of I. A set function $\mu : \mathcal{P}(I) \to [0, 1]$ is called a non-additive measure (or a fuzzy measure) if it satisfies the following three axioms:*

(1) $\mu(\emptyset) = 0$: the empty set has no importance.
(2) $\mu(I) = 1$: the maximal set has maximal importance.
(3) $\mu(B) \leq \mu(C)$ if $B,\ C \subset \mathcal{P}(I)$ and $B \subset C$: a new criterion added cannot make the importance of a coalition (a set of criteria) diminish.

Of course, any probability measure is also a non-additive measure. Therefore non-additive measure theory is an extension of traditional measure theory. Moreover, a notion of integral can also be defined over such measures.

A non-additive integral, such as the Choquet integral [5], is a type of a general averaging operator that can model the behavior of a decision maker. The decision-maker provides a set of values of importance, this set being the values of the non-additive measure on which the non-additive integral is computed from. Formally, The Choquet integral is defined as follows:

Definition 6.2. (Choquet integral) *Let μ be a non-additive measure on $(I, \mathcal{P}(I))$ and an application $f : I \to \mathbb{R}^+$. The Choquet integral of f w.r.t. μ is defined by:*

$$(C)\int_I f d\mu = \sum_{i=1}^{n} (f(\sigma(i)) - f(\sigma(i-1)))\mu(A_{(i)}),$$

where σ is a permutation of the indices in order to have $f(\sigma(1)) \leq \cdots \leq f(\sigma(n))$, $A_{(i)} = \{\sigma(i), \ldots, \sigma(n)\}$, and $f(\sigma(0)) = 0$, by convention.

It can be shown that many aggregation operators can be represented by Choquet integrals with respect to some fuzzy measure. However, note that there are other non-additive approaches to decision making besides the Choquet integral, one of them being the Sugeno integral [27]:

Definition 6.3. (Sugeno integral) *Let μ be a fuzzy measure on $(I, \mathcal{P}(I))$ and an application $f : I \to [0, +\infty]$. The Sugeno integral of f w.r.t. μ is defined by:*

$$(S)\int f \circ \mu = \bigvee_{i=1}^{n} (f(x_{(i)}) \wedge \mu(A_{(i)})).$$

where $\vee$ is the supremum and $\wedge$ is the infimum.

Even though the Choquet and the Sugeno integrals are structurally similar, their applications are very different. The Choquet integral is generally used in quantitative measurements, while Sugeno integral has found more applications in qualitative approaches. However, we restrict ourselves to quantitative approaches.

Although the Choquet integral is well suited for quantitative measurements, it has a major drawback. The decision maker needs to input a value of importance of each subset of attributes, that is total of 2^n values. More precisely, since the value of the empty set and the entire set are known by the definition of a fuzzy measure, the exact number of values required from the decision-maker is $2^n - 2$. This still leads to an exponential complexity and is therefore intractable. However, we can overcome intractability by using 2-additive measure to limit the complexity to a $O(n^2)$ (as shown in [3]) and still get accurate results.

Before giving the definition of a 2-additive measure, we need to define notion of interaction indices of orders 1 and 2. The importance of an attribute (or the interaction index of degree 1) is best described as the value this attribute brings to each coalition it does not belong to. It is given by the Shapley value [26]:

Definition 6.4. (Shapley value) *Let μ be a non-additive measure over I. The Shapley value of index i is defined by:*

$$v(i) = \sum_{B \subset I \setminus \{i\}} \gamma_I(B)[\mu(B \cup \{i\}) - \mu(B)] \tag{68}$$

with

$$\gamma_I(B) = \frac{(|I| - |B| - 1)! \cdot |B|!}{|I|!} \tag{69}$$

and $|B|$ denoting the cardinal of B.

While the Shapley value gives the importance of a single attribute to the entire set of attributes, the interaction index of degree 2 represents the interaction between two attributes, and is defined by ([8],[13]):

Definition 6.5. (Interaction index of degree 2) *Let μ be a non-additive measure over I. The interaction index between i and j is defined by:*

$$I(i,j) = \textstyle\sum_{B \subset I \setminus \{i,j\}} \left(\xi_I(B) \cdot (\mu(B \cup \{i,j\}) \right.$$

$$\left. -\mu(B \cup \{i\}) - \mu(B \cup \{j\}) + \mu(B)\right)$$

with $\xi_I(B) = \frac{(|I|-|B|-2)! \cdot |B|!}{(|I|-1)!}$.

The interaction indices belong to the interval $[-1, +1]$ and

- $I(i,j) > 0$ if the attributes i and j are complementary;
- $I(i,j) < 0$ if the attributes i and j are redundant;
- $I(i,j) = 0$ if the attributes i and j are independent.

Even though we can define interaction indices of any order, defining the importance of attributes and the interaction indices between two attributes is generally enough in MCDM problems. Thus, 2-additive measures constitute a feasible and accurate tool in this setting. The formal definition of 2-additive measure follows [8]:

Definition 6.6. (2-additive measure) *A non-additive measure μ is called 2-additive if all its interaction indices of order equal to or larger than 3 are null and at least one interaction index of degree two is not null.*

We can also show [12] that the Shapley values and the interaction indices of order two offer us an elegant way to represent a Choquet integral. Therefore, in a decision-making problem, we can ask the decision maker to give the Shapley values, I_i, and the interaction indices, I_{ij}, and then use the Choquet integral w.r.t. to a 2-additive measure, μ, to obtain the aggregation operator:

$$(C)\int_I f d\mu = \sum_{I_{ij}>0} (f(i) \wedge f(j)) I_{ij} + \sum_{I_{ij}<0} (f(i) \vee f(j)) |I_{ij}| + \sum_{i=1}^{n} f(i)(I_i - \frac{1}{2}\sum_{j \neq i} |I_{ij}|).$$

This form of the Choquet integral is accurate and practical approach to many situation, one of them being portfolio management.

7.2 Application to Portfolio Management

We propose two different algorithms that make use of multi-criteria decision making approach to find the optimal portfolio allocation. A two-stage algorithm uses a multi-criteria decision making setting to rank all asset. Based on the rank, good assets are selected among thousands of assets that exist in market and wealth is invested in these selected assets only. The second step of the algorithm utilizes another MCDM setting to determine the exact wealth allocation among the assets to best suit the goals of the investor.

The second algorithm utilizes similar multi-criteria decision making settings by starts by clustering all assets into three groups based on their risk. Based on the investor's acceptable level of risk, distribution of wealth among three groups of assets is determined and MCDM setting is created to determine the exact allocation of wealth within each cluster.

We first define a multi-criteria decision making problem by considering the set of all asset as the set of alternatives. We determine a finite set of criteria that characterize investment assets–return (R), risk (r), time to maturity (t), transaction cost (c), etc., and define the utility functions for each of them. The simplest method to choose rational utility functions is to provide mappings from the values of an alternative onto the interval $[0, 1]$, $f : X_i \to [0, 1]$. For the return of an asset, this could mean that the highest realistic return is mapped into 1, the lowest return to 0, and the other returns are proportionally mapped into values between 0 and 1. The utility of risk could be defined in a similar fashion, but taking the reciprocal of the risk since a high value of risk is less desired than a low value of risk. Similar arguments hold for time to maturity and transaction cost. Once the utility function for each criterion is defined, we proceed to calculation of the value of each asset.

If the decision maker (the investor) is concerned only with the return or only with the level of risk, then the maximax strategy could be used to rank all the assets with a high importance given to return in the first case and to the risk in the second case, and low importance given to all the other attributes. However, usually an investor wants to maximize the return for a given level of risk or minimize the risk while attaining the required return level in a certain time period. Thus, all the criteria have some influence on the decision. The decision-maker is asked to input the Shapley value of each criterion, that is the importance of each criterion relative to other criteria. Since the attributes are mutually dependent (e.g., higher return usually implies higher risk, longer time to maturity usually means higher return, etc.), the weighted sum approach does not promise to give accurate results. However, we can approximate the interaction indices for each pair of attributes by estimating the correlation between their values, and use the Choquet integral with respect to a 2-additive measure, defined by Shapley values and interaction indices of order 2, to calculate the global value of an asset. The Choquet integral values are used to order the assets giving higher rank to the assets with the higher value of the Choquet integral.

Top n assets are chosen to proceed to the second stage of the algorithm. The number n is either pre-defined by the investor, or all the assets with the Choquet value above a threshold specified by the investor are selected. We denote the set of all assets that are used to create portfolio by A. The second stage of the algorithm tends to find the optimal distribution of wealth among n selected assets, $\mathbf{w} = (w_1, \dots, w_n)$, by considering another multi-criteria decision making setting. The set of alternatives is defined as the set of all possible portfolios using only the assets selected based on their rank. The set of criteria is unchanged from the first stage of the algorithm. However, the values of the criteria are defined in terms of their values for each asset in the portfolio as follows:

- The return of the portfolio is

$$R(\mathbf{w}) = \sum_{i=1}^{n} R_i w_i. \tag{70}$$

- The risk of the portfolio is

$$r(\mathbf{w}) = \sum_{i=1}^{n} r_i w_i. \tag{71}$$

- Time to maturity of the portfolio, however, is not the weighted sum of the individual assets' maturity times. It is the maximum time to maturity of all assets included in the portfolio:

$$t(\mathbf{w}) = \underset{j}{max}\, t_j, \text{where } j \text{ is such that } x_j \in A. \tag{72}$$

  Note that if all assets are included in every portfolio, then the time to maturity will be same for all portfolios.
- The transaction cost of the portfolio is

$$c(\mathbf{w}) = \sum_{i=1}^{n} c_i v_i, \tag{73}$$

where $v_i = w_i$ if the transaction cost of the asset i is a proportion of wealth invested into the asset, and $v_i =$ constant s if the transaction cost of the asset j is equal to s for any amount invested.
- Similarly, the values of other attributes characterizing a portfolio could be defined in terms of values of individual assets included into the portfolio.

Keeping the same Shapley values for all attributes and interaction indices of degree 2 for each pair of attributes as given in the first step of the algorithm, we maximize the Choquet integral of the alternatives. Thus, this stage of the algorithm reduces to an optimization constraint programming problem that finds the vector $\mathbf{w} = (w_1, \ldots, w_n)$ that maximizes the objective function

$$\max_{\mathbf{w}} \sum_{I_{ij}>0} I_{ij}[u_i(x_i) \wedge u_j(x_j)] + \sum_{I_{ij}<0} |I_{ij}| \cdot [u_i(x_i) \vee u_j(x_j)] + \sum_{i=1}^{n} \left(u_i(x_i) - \frac{1}{2} \sum_{i \neq j}^{n} I_{ij} \right). \tag{74}$$

Here, x_i and x_j represent criteria of the portfolio (e.g., risk, return, time to maturity, etc.), which are defined in terms of w_i, and one of the characteristics of portfolio (r_i, R_i, t_i, c_i, or others).
The maximization problem is subject to the following constraints:

$$\sum_{i=1}^{n} w_i R_i \geq R \text{or} \sum_{i=1}^{n} w_i r_i \leq r \text{ (portfolio satisfies the main goal of the investor);} \tag{75}$$

$$\sum_{i=1}^{n} w_i = 1 \text{ (exactly all wealth is invested);} \tag{76}$$

$$w_i \geq 0 \;\; \forall i = 1, \ldots, n \text{ (money can not be borrowed to be invested in an asset).} \tag{77}$$

This problem involving constraints could be solved using standard optimization techniques. Since all constraints are linear, the choice of the optimization technique depends on the form of the objective function. Using the simple utility functions described in this section, the objective function is linear as well, which allows us to use of the simplex method to determine the optimal solution. However, if some complex utility functions are used to execute the multi-criteria decision process, the objective function might not be linear and other methods, such as Karush-Kuhn-Tucker and Fritz-John, must be used to find the solution. Since both methods guarantee to find a local optimum but not the global one, we can iterate the algorithm several times with different starting points to find a better solution.

To reduce the complexity of the algorithm, we developed another algorithm that utilizes MCDM setting in portfolio selection. It starts by ordering all assets based only on their risk, and using this ranking it clusters all the assets into three groups: high, middle, and low risk assets. The clustering is performed such that the third of assets with the highest risk constitutes group 1, group 2 contains the middle risk assets, and group 3 is the third of assets with the lowest risk. Next, we calculate the Choquet integral of each asset following the same MCDM setting as in the first algorithm. We select top $n_1 > 0$, $n_2 > 0$, and $n_3 > 0$ assets respectively from high, middle, and low risk clusters to be included into the portfolio. The values of n_1, n_2, and n_3 are either all equal and predetermined, or they are such that the values of assets selected from each cluster are higher than a predefined threshold value.

Based on the investor's level of risk aversion, the proportion of wealth invested in each cluster is determined and denoted by p_1, p_2, and p_3 respectively for groups 1, 2, and 3. If the decision-maker is highly risk-averse, p_1 will be much smaller than p_2 and p_3, while for a risk-prone individual, p_3 will be smaller than p_1 and p_2. However, none of the numbers will be equal to zero in order to diversify portfolio, which is necessary to reduce unsystemic risk, the risk that depends on the company.

Finally, the wealth allocated to each cluster is distributed among the assets that belong to the group, so that the optimal portfolio is selected. Each cluster is considered separately from the other two and the best distribution of wealth is determined by maximizing the Choquet integral of the portfolios built by selected assets in each group

$$\max_{\mathbf{w}} \sum_{I_{ij}>0} I_{ij}[u_i(x_i) \wedge u_j(x_j)] + \sum_{I_{ij}<0} |I_{ij}|[u_i(x_i) \vee u_j(x_j)] + \sum_{i=1}^{n} (u_i(x_i) - \frac{1}{2} \sum_{i \neq j}^{n} I_{ij}), \tag{78}$$

subject to

$$\sum_{i=1}^{n} w_i = 1 \text{ (exactly all wealth is invested);} \tag{79}$$

$$w_i \geq 0 \;\; \forall i = 1, \ldots, n \text{ (money can not be borrowed to be invested in an asset).} \tag{80}$$

Note that this algorithm does not explicitly require satisfaction of the main goal of the investor (e.g., required return level, maximum risk rate, etc.), but this requirement is implicitly accounted for in the distribution of wealth among three clusters. We can again apply one of the standard optimization techniques to solve this problem.

Even though the utility based multi-criteria decision making setting and its solution by use of Choquet integral with respect to 2-additive measure is a feasible and accurate solution for values given by the decision maker, this approach faces another problem. We cannot expect a decision maker to give precise values for the importance and interaction indices. In order to overcome this hurdle, it was shown [3] that the use of intervals provides a nice solution in MCDM problem.

7.3 Intervals

Interval Arithmetic (IA) is an arithmetic over sets of real numbers called *intervals*. It had started the development in fifties in order to model uncertainty, and to tackle rounding errors of numerical computations. For a complete presentation of interval arithmetic, we refer the reader to [15].

Definition 6.7. (Interval) *A closed real interval is a closed and connected set of real numbers. The set of closed real intervals is denoted by* $\mathbb{IR}$. *Every* $\boldsymbol{x} \in \mathbb{IR}$ *is denoted by*

$$[\underline{x}, \overline{x}], \tag{81}$$

where its bounds are defined by $\underline{x} = \inf \boldsymbol{x}$ *and* $\overline{x} = \sup \boldsymbol{x}$.
For every $a \in \mathbb{R}$, *the interval point* $[a, a]$ *is also denoted by* a.

In the following, elements of $\mathbb{IR}$ are simply called *real intervals* or *intervals*.

The *width* of a real interval $\boldsymbol{x}$ is the real number $w(\boldsymbol{x}) = \overline{x} - \underline{x}$. Given two real intervals $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{x}$ is said to be *tighter than* $\boldsymbol{y}$ if $w(\boldsymbol{x}) \leq w(\boldsymbol{y})$.

Interval Arithmetic operations are set theoretic extensions of the corresponding real operations. Due to properties of monotonicity, these operations can be implemented by real computations over the bounds of intervals. Given two intervals $\boldsymbol{x} = [a, b]$ and $\boldsymbol{y} = [c, d]$, we have for instance:

$$\begin{cases} \boldsymbol{x} + \boldsymbol{y} = [a + c, b + d] \\ \boldsymbol{x} - \boldsymbol{y} = [a - d, b - c] \\ \boldsymbol{x} \times \boldsymbol{y} = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \\ \boldsymbol{x}^n = \begin{cases} [a^n, b^n] & \text{if } n \text{ is an odd natural number} \\ [0, \max\{|a|, |b|\}^n] & \text{if } n \text{ is even and } 0 \in [a, b] \\ [\min\{|a|, |b|\}^n, \max\{|a|, |b|\}^n] & \text{if } n \text{ is even and } 0 \notin [a, b] \end{cases} \end{cases}$$

The associative law and the commutative law are preserved over $\mathbb{IR}$. However, the distributive law does not hold. In general, only a weaker law is verified, called semi-distributivity. For all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{IR}$, we have:

associativity: $(\boldsymbol{x} + \boldsymbol{y}) + \boldsymbol{z} = \boldsymbol{x} + (\boldsymbol{y} + \boldsymbol{z})$
$(\boldsymbol{x}\boldsymbol{y})\boldsymbol{z} = \boldsymbol{x}(\boldsymbol{y}\boldsymbol{z})$

commutativity: $(\boldsymbol{x} + \boldsymbol{y}) = (\boldsymbol{y} + \boldsymbol{x})$
$\boldsymbol{x}\boldsymbol{y} = \boldsymbol{y}\boldsymbol{x}$

sub-distributivity: $\boldsymbol{x} \times (\boldsymbol{y} + \boldsymbol{z}) \subseteq \boldsymbol{x} \times \boldsymbol{y} + \boldsymbol{x} \times \boldsymbol{z}$

Intervals of preferences

As mentioned earlier, to define preferences over multi-dimensional alternatives, the user is required to provide importance and interaction indices, but is more likely to provide intervals of values $\boldsymbol{I_i}$ and $\boldsymbol{I_{ij}}$, $\forall i, j \in \{1, \ldots, n\}$, which leads to evaluation of a Choquet integral over intervals using IA:

$$\begin{aligned}(C_{\mathbb{I}}) \int_I f d\mu = & \sum_{\boldsymbol{I_{ij}}>0} \left((f(i) \wedge f(j)) - \frac{1}{2}(f(i) + f(j)) \right) \boldsymbol{I_{ij}} \\ & + \sum_{\boldsymbol{I_{ij}}<0} \left((f(i) \vee f(j)) - \frac{1}{2}(f(i) + f(j)) \right) |\boldsymbol{I_{ij}}| \\ & + \sum_{i=1}^{n} f(i) \boldsymbol{I_i}\end{aligned}$$

where the annotation $(C_{\mathbb{I}})$ means that the interpretation of this formula is performed using IA. As a consequence, the value of the integral is an interval.

Back to the Portfolio Optimization Problem

Intervals allow the problem of portfolio management to be presented more realistically as the investor is asked to provide the ranges of values of the importance and interaction indices of order 2 instead of the exact values. It is reasonable to believe that an investor can determine whether, for example, minimization of risk is more important than the return from an asset, or whether the time period in which an amount can be obtained is more important than risk. However, it is more realistic that the investor can give the interval of how much one criterion is more important than the other criterion rather than giving the exact values of the relative importance among criteria. Thus, the intervals provide a rational way to solve the portfolio optimization problem by following the same procedures as the non-interval based methods and evaluating the Choquet integral over intervals and extending the optimization techniques to intervals as well. However, a new issue arises when using intervals to evaluate alternatives: the result of the Choquet integral evaluated over intervals is an interval, and intervals are not as easy to compare as real numbers.

Strategies of preference

When comparing intervals, the ideal case is when the intervals of preferences do not intersect. In this case, if alternative I is evaluated with values that are all better than those of alternative J, the preference is clearly given to the alternative I.

However, the above case is very specific and unfortunately does not happen often. More common is that two intervals intersect and we need to choose a better of two overlapping intervals. The strategies to make decisions in such cases are described below.

A simple naive strategy offers a straightforward solution that compares only the upper bounds and gives the preference to the interval with the highest upper bound (which corresponds to an optimistic behavior of a decision-maker as he/she is only interested in the highest potential values rather than all the values that could be reached), or compares the lower bounds and gives preference to the highest lower bound (which corresponds to a pessimistic behavior).

However, many alternatives between the very optimistic case and the very pessimistic case are possible. They require us to look simultaneously at the upper and lower bounds as well as the width of the intervals, which highlights the degree of uncertainty of the alternative's value. To combine these variables, a degree of preference was introduced [3]. A degree of preference, $d(I, J)$, intended to express the extent to which a better value of the Choquet integral is likely to lie in interval I, rather than in interval J.

It is defined as a function $d: \mathbb{I}^2 \rightarrow [0, 1]$, where:

$$d(I,J) = \begin{cases} \frac{\overline{I}-\overline{J}}{|(\overline{I}-\overline{J})+(\underline{J}-\underline{I})|} & \text{if } \overline{I} > \overline{J} \text{ and } \underline{J} \geq \underline{I} \\ 1 & \text{if } \overline{I} = \overline{J} \text{ and } \underline{I} > \underline{J} \\ & \text{or: if } \overline{I} > \overline{J} \text{ and } \underline{I} \geq \underline{J} \\ & \text{or: if } \overline{I} = \overline{J} \text{ and } \underline{I} = \underline{J} \\ 1 - d(J,I) & \text{otherwise} \end{cases} \tag{82}$$

The higher the value of the degree of preference, the greater the chance that the optimal interval is the interval I, while lower value of the degree of preference implies that the interval J would more likely contain higher value of Choquet integral.

The degree of preference, as described above, assumes that a decision-maker is risk-neutral, that is the person is not willing to undergo an extreme risk nor he/she believes that there is a reason to be too careful. However, sometimes, a person exhibits a risk-prone attitude and leans towards optimistic behavior, or on the other hand, the decision-maker could be more risk-averse especially if there is a reason to expect pessimistic results. If a decision-maker could provide the level of risk that he/she is willing to take in order to maximize the utility, then we can modify the degree of preference to include this fact.

Let us assume that the level of risk a person wants to take is expressed by a real value in the interval $[0, 1]$, where naturally, values close to 0 represent pessimistic situations, and values closer to 1 mean more optimistic expectations. Now, we can tighten the considered interval to better suit this level of risk. The shrinking of the interval $[\underline{X}, \overline{X}]$ based on the risk level $r \in [0, 1]$ is done in the following way [21]:

- First, calculate the proportion of the interval that is considered important by the decision-maker:

$$p = 2 \cdot \min\{r - 0, 1 - r\}. \tag{83}$$

- Next, calculate the size of the interval that corresponds to the given proportion:

$$\text{size} = p \cdot (\overline{X} - \underline{X}). \tag{84}$$

- Finally, calculate the interval of importance, $[\underline{N}, \overline{N}]$:

$$[\underline{N}, \overline{N}] = \begin{cases} [\underline{X}, \underline{X} + size] \text{ if } r \leq 0.5 \\ [\overline{X} - size, \overline{X}] \text{ otherwise} \end{cases} \tag{85}$$

This approach clearly returns a single point instead of an interval in cases when the level of risk is at extreme points, *i.e.,* the interval of importance is the upper bound of the original interval when the risk level is 1, and the lower bound when the risk level is 0. In both cases, the problem is reduced to comparison of single (extreme) points rather than intervals, the situation that corresponds to the naive strategy.

Once we have tightened the intervals to reflect the level of risk the decision-maker is willing to take, we apply equation (82) to new intervals of importance to calculate the degree of preference, which determines the better of two intervals.

The presented approach to determine the better of two intervals given the level of risk works well if the decision-maker can provide the exact degree of risk he/she is willing to take. However, in reality it is hard to describe the level of risk by a single number [21]. More probable is that a person could define the level of risk by an interval $r = [\underline{r}, \overline{r}]$. In this case, the calculation of the interval of importance that encounters for the optimism/pessimism of a person is a bit more complicated. Instead of a precise interval, the result is an interval whose bounds are themselves intervals (2nd order interval), and therefore, the degree of preference would result in an interval, $d(I, J) = [\underline{d}, \overline{d}]$ rather than a single number. Three situations could occur:

- $\overline{d} < 0.5$ in which case the preferable choice is interval J.
- $\underline{d} > 0.5$ in which case the preferable choice is interval I.
- $0.5 \in [\underline{d}, \overline{d}]$ in which case the preferable choice is

  (1) interval I if $(\overline{d} - 0.5) \geq (0.5 - \underline{d})$

  (2) interval J otherwise.

All of the above rankings of intervals suppose uniform probability distribution, which is a reasonable assumption if no additional information is available. However, sometimes more information is accessible and more accurate probability distribution over an interval could be considered. Typically, if the width of interval is not limited, it is common that a decision-maker would give an interval bigger than what he/she really believes the interval should be to cover any possible extreme value even though the extreme values would very rarely happen. Thus, it is not uncommon that the values within an interval would not follow uniform distribution, but rather a form of Gaussian distribution (possibly screwed). In this situation, it is reasonable to assume that the interval Choquet integral would also not follow uniform distribution but would rather have higher probability of values in the interior of the interval than those close to bounds.

In the case when more information is available about probability distribution over an interval, we can slightly modify the approach of calculating the degree of preference [21]. As before, we start by tightening the given interval based on the level of risk, r, that a person is willing to take. Thus, we need to determine the value, s, within the given interval $[\underline{X}, \overline{X}]$ such that

$$s = \begin{cases} P(x \leq 2r) & \text{if } r < 0.5 \\ P(x \geq (2r-1)) & \text{if } r > 0.5 \end{cases} \tag{86}$$

So the interval of importance is:

$$[\underline{N}, \overline{N}] = \begin{cases} [\underline{X}, \underline{X} + s] & \text{if } r \leq 0.5 \\ [\overline{X} - s, \overline{X}] & \text{otherwise} \end{cases} \tag{87}$$

Note that the above formula when applied to the uniform distribution leads exactly to the equation (85), with s replacing the variable *size*.

The next step is to calculate the degree of preference between two intervals given their new bounds. Encountering the probability distribution, the degree of preference is given by:

$$d(I, J) = \begin{cases} \frac{P(\overline{J} \leq x \leq \overline{I})}{P(\overline{J} \leq x \leq \overline{I}) + P(\underline{I} \leq x \leq \underline{J})} & \text{if } \overline{I} > \overline{J} \text{ and } \underline{J} \geq \underline{I} \\ 1 & \text{if } \overline{I} = \overline{J} \text{ and } \underline{I} > \underline{J} \\ & \text{or: if } \overline{I} > \overline{J} \text{ and } \underline{I} \geq \underline{J} \\ & \text{or: if } \overline{I} = \overline{J} \text{ and } \underline{I} = \underline{J} \\ 1 - d(J, I) & \text{otherwise} \end{cases} \tag{88}$$

When applied to uniform distribution, this equation simplifies to equation (82).

Back to the Portfolio Optimization Problem

Even though the presented utility-based approach to decision making has not yet been tested on real data, the theoretical framework is sound and general enough to be applied successfully to real finance data sets. Its performance could be tested against the performance of benchmark portfolios, and there is no doubt that the approach would perform superiorly compared to these techniques as fuzzy integration over intervals adjusts the drawbacks of other commonly used intelligent techniques, which outperform benchmark portfolios themselves. Thus, this method offers a natural and logical framework to optimize portfolio selection.

8 Conclusion

Computational intelligence techniques are very useful for solving problems involving the understanding, modeling and analysis of large data sets. Very often, trying to take all variables into considerations, along with variable dependencies is not practical, as this approach rapidly becomes untractable, even if using distributed computing techniques. On the other hand, humans are very efficient at identifying what matters, and discarding what does not matter in a given situation. Computational intelligence techniques are precisely replicating this process of eliminating the 'noise' and focusing on the data that matter.

We have seen that finance is an area that is well-suited for computational intelligent approaches. We have presented a state of the art on computational techniques for portfolio management, that is, how to optimize a portfolio selection process. More specifically, we have shown that genetic algorithms, rule-based systems, neural networks, and support vector machines offer some advantages to benchmark portfolio management schemes, be it in complexity or in accuracy. We then proceeded to show that a utility-based approach to decision making offers a natural and logical framework to optimize portfolio selection, and have shown how the Choquet integral (which generalizes a large class of aggregation operators in multi-criteria decision making), constraint programming, and interval computation can be used to solve such problems, and allow us to deal with both uncertain and imprecise data. This approach has not been tested yet on real data, however, the theoretical framework is sound and general enough to be applied successfully to real finance data sets. Moreover, it addresses some of the issues pertaining to other computational technique approaches, such as overfitting of parameters, description of the dependencies between characteristics of an asset, imprecise data, etc.

Last, although we have focused on portfolio management problems, it is quite possible to use similar computational technique approaches to pricing problems, as an alternative to the more traditional stochastic differential equations and stochastic integration approaches.

References

1. Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992) A training algorithm for optimal classifiers. In: D. Haussler (Editor) 5^{th} Annual ACM Workshop on COLT, 144-152, ACM Press, Pittsburgh, PA.
2. Casas, C. A. (2001) Tactical asset allocation: an artificial neural network based model. Proceedings of International Joint Conference on Neural Networks, 3:1811-1816.
3. Ceberio, M. and Modave, F. (2006) Interval-based multicriteria decision making. In: B. Bouchon-Meunier, G. Coletti, and R. R. Yager (Editors) Modern Information Processing: from Theory to Applications. Elsevier ed.
4. Chapados, N. and Bengio, Y. (2001) Cost functions and model combination for VaR-bsed asset allocation using neural networks. IEEE Transactions on Neural Networks, 12:890-906.
5. Choquet, G. (1953) Theory of capacities. Annales de l'Institut Fourier, 5.
6. Cohen, P. R. and Lieberman, M. D. (1990) A report on FOLIO: an expert assistant for portfolio managers. Investment Management: Decision Support and Expert Systems, 135-139.
7. Cortes, C. and Vapnik, V. (1995) Support-Vector Networks. Machine Learning, 20.
8. Denneberg, D. and Grabisch, M. (1996) Shapley value and interaction index. Mathematics of interaction index.
9. Fan, A. and Palaniswami, M. (2001) Stock selection using support vector machines. Proceedings of International Joint Conference on Neural Networks, 3:1793-1798.
10. Giarratano, J. and Riley, G. (1994) Expert systems: principles and programming. PWS Publishing Company, Boston, MA.
11. Goldberg, D. E. (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, MA.
12. Grabisch, M. (2000) The interaction and Mobius representation of fuzzy measures on finite spaces, *k*-additive measures: a survey. In: M. Grabisch, T. Murofushi, and M. Sugeno (Editors) Fuzzy measures and integrals: Theory and applications. Physica Verlag.
13. Grabisch, M. and Roubens, M. (2000) Application of the Choquet integral in multicriteria decision making. In: M. Garbisch, T. Murofushi, and M. Sugeno (Editors) Fuzzy Measures and Integrals: Theory and Applications. Physica Verlag.
14. Haupt, R. L. and Haupt, S. E. (2004) Practical Genetic Algorithms (2^{nd}*edition*). Wiley, New York.
15. Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001) Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics. Springer-Verlag, London.
16. Joshi, M. (2003) The concepts and practice of mathematical finance. Cambridge.
17. Krantz, D., Luce, R., Suppes, P., and Tverski, A. (1971) Foundations of measurement. Academic Press.
18. Lai, K. K., Yu, L., Wang, S., and Zhou, C. (2006) A double-stage genetic optimization algorithm for portfolio selection. I. King et al. (Editors): ICONIP 2006, part III, LNCS 4234:928-937. Springer-Verlag Berlin Heidelberg.

19. Lin, L., Cao, L., Wang, J., and Zhang, C. (2004) The applications of genetic algorithms in stock market data mining optimization. In: A. Zanasi, N. F. F. Ebecken, and C. A. (Editors) Data Mining V, WIT Press.
20. Lowe, D. (1994) Novel exploitation of neural network methods in financial markets. IEEE International Conference on Neural Networks, 6:3623-3628.
21. Magoc, T., Ceberio, M., and Modave, F. (2008) Interval-based multi-criteria decision making: strategies to order intervals. Proceedings of North American Fuzzy Information Processing Spciety.
22. Modave, F. and Grabisch, M. (1997) Preferential independence and the Choquet integral. 8^{th} International Conference on the Foundations and Applications of Decision Under Risk and Uncertainty, Mons, Belgium.
23. Nilsson, N. J. (1998) Artificial intelligence: a new synthesis. Morgan Kaufman.
24. Prugel-Bennett, A. and Shapiro, J. L. (1994) An analysis of genetic algorithms using statistical mechanics. Physycal Review letters, 72(9) 1305-1309.
25. Reeves, C. R. and Rowe, J. E. (2003) Genetic algorithms-Principles and Perspective: A Guide to GA Theory. Kluwer Academic Publisher, London.
26. Shapley, L. S. (1953) A value for n-person games. In: H. W. Kuhn and A. W. Tucker (Editors) Contributions to the Theory of Games, 2:307-317. Princeton University Press.
27. Sugeno, M. (1974) Theory of fuzzy integrals and its applications. PhD Thesis, Tokyo Institute of Technology.
28. Schlkopf, B., Burges, C., and Vapnik, V. (1995) Extracting support data for a given task. Advances in Neural Information Processing Systems.
29. Seo, Y., Giampapa, J., and Sycara, K. (2004) Financial news analysis for intelligent portfolio management. Tech. report CMU-RI-TR-04-03, Robotics Institute, Carnegie Mellon University.
30. Sycara, K., Decker, K., and Zeng, D. (1998) Intelligent agents in portfolio management. In: N. Jennings and M. Wooldridge (Editors) Agent Technology: Foundations, Applications, and Markets. Springer.
31. Tan, P., Steinbach, M., and Kumar, V. (2006) Introduction to data mining. Addison Wesley.
32. Zhou, C., Yu, L., Huang, T., Wang, S., and Lai, K. K. (2006) Selecting valuable stocks using genetic algorithm. T.-D. Wang et al. (Editors): SEAL 2006, LNCS 4247, pp. 688-694. Springer-Verlag Berlin Heidelberg.
33. Zimmermann, H. G. and Grthmann, R. (2005) Optimal asset allocation for a large number of investment opportunities. Intelligent systems in Accounting, Finance, and Management, 13:33-40.
34. Zimmermann, H. G., Neuneier, R, and Grthmann, R. (2002) Active portfolio management based on error correction neural networks. Proceedings of Neural Network Information Processing systemns, Vancouver, Canada.