Multi-Objective Machine Learning Strategies in Investment Portfolio Optimisation

M. Janssen
markjanssen@student.tudelft.nl

X. Savenberg
xandersavenberg@student.tudelft.nl

Abstract

In this paper, we examine various machine learning techniques and their applications on the stock market. We review the general concepts of support vector machines, artificial neural networks and genetic algorithms. We provide an introduction to technical analysis and the problem of portfolio optimisation, and provide arguments in favour of using multi-objective machine learning techniques to help solve this problem, as well as some examples of indicators that could be used for this purpose. We discuss the applications of multi-objective support vector machines, artificial neural networks and genetic algorithms on the stock market. Furthermore, we give recommendations on the direction future research in this field should take, in particular research on the usage of advanced machine learning and distributed computing techniques. Finally, we see opportunities in a more thorough analysis on intraday data, which is a field that provides new challenges in need of a different approach.

1 Introduction

Throughout the entire history of the stock market, investors have always sought to get ahead of their competitors in some way, be it through conventional or creative, risk-free or risky, legal or even illegal means. Since the advent of the twenty-first century, financial markets have increasingly become populated with traders who rely on the potential of computing to improve their results. And while researchers currently disagree on whether this is a good or bad thing, there does seem to be a consensus regarding the fact that so far, the research done has only barely scratched the surface of the possibilities this field has to offer.

On the modern stock market, a large portion of trading has become automated. Investment banks, trading firms, pension funds and many other institutional traders use automated trading in the hopes of minimising risk and maximising profits. A specific kind of automated trading that has gained a lot of popularity recently is High Frequency Trading (HFT), a manner of trading that rapidly trades securities, trading in and out of positions in the blink of an eye. With computing power becoming ever cheaper and more readily available, competition among traders is no longer just about who has the best trading strategies: it is also a matter of who is the fastest in deciding when and how to trade.

Obviously, a human investor is not able to simultaneously make large quantities of decisions in the blink of an eye: for this reason, companies build automated trading systems that take instructions from humans and then perform 'split-second trades' according to pre-defined strategies. This way of trading is not without risks, however: take for example the 2010 'Flash Crash', where the Dow Jones suddenly fell about 1000 points (approximately 9%), only to recover within a few minutes, which was attributed by many to the limitations of automated trading. Malfunctioning systems have also been known to cause traders massive losses. Therefore, it is crucial to not build a system that is robust and can also devise strategies and predict the course of the market as effectively as is possible.

One way of devising trading strategies, which has gained both popularity and credibility in recent years, is Technical Analysis (TA). Through the study of past performance of stocks and other indicators, technical analysts seek to find patterns in the performance of the market and

attempt to use these patterns to their advantage. This way of analysing the market has also drawn a lot of criticism, especially from proponents of the Efficient Market Hypothesis (EMH), who believe that financial markets are 'informationally efficient' and, as a result of this, cannot be consistently predicted. A more thorough explanation of TA and the EMH will be given in Section 4.

Many researchers have already looked into stock market prediction and portfolio optimisation. Though the terms and techniques used may vary, in the end most papers we cite have the same goal: trying to teach a computer to predict the stock market or help make investment decisions as accurately as possible.

Besides the approaches discussed in this paper, there are of course other machine learning methods with their own pros and cons. Huang and Tsai [19] used a self-organising feature map (SOFM) combined with support vector regression (SVR) to predict the Taiwanese futures market, and Afolabi and Olude [2] used a hybrid Kohonen self-organising map (SOM).

There have also been many reviews of various techniques. An example is work by Yoo, Kim and Jan [37], who compared classical statistical techniques, neural networks, support vector machines, and case-based reasoning.

In this paper, we will explore using machine learning techniques for stock market prediction and portfolio optimisation, combining knowledge from the fields of finance and computer science. We will examine how existing Machine Learning techniques such as Neural Networks (NN), Support Vector Machines (SVM) and Single- or Multi-Objective Genetic Algorithms (GA) can be used to assist traders in devising and optimising strategies. In Section 2, we explain the basic concepts of several popular Machine Learning techniques. In Section 3, we introduce the financial concepts to be used in the rest of the paper. In Section 4, we argue that multi-objective techniques are superior for the purposes of stock market prediction and portfolio optimisation for a variety of reasons. In Section 5, we take several multi-objective machine learning techniques and show how they can be used for stock trading purposes. Finally, in Section 6 we review our findings, give our suggestions regarding what areas future research should target, and provide some advice on how it should be carried out.

2 Machine learning

In this section we describe three well-known machine learning techniques: Support Vector Machines, Artificial Neural Networks, and Genetic Algorithms. First we introduce Support Vector Machines (SVM) in Section 2.1. In Section 2.2 we give a brief overview of the Artificial Neural Networks technology. Finally, in Section 2.3 we introduce Genetic Algorithms.

2.1 Support Vector Machines

Support Vector Machines (SVM) find their origin in pattern recognition [13]. The technique is designed as a two-class classifier, which means that a trained SVM is able to distinguish between two classes of input. A common example is the optical recognition of numbers.

SVMs are operated in a two phases. First, the SVM is trained to supplied data which is already classified. Second, the SVM is able to classify other objects according to its model.

SVM is a so-called white-box model: it is possible to see the inner workings of the system after training. This white-box model is opposed to a black-box model, in which the user is not able to see the underlying parameters of a trained model.

SVM in general

The first phase, the training, is based upon a regression of a line. If we consider a simple example, a dataset with points in a two dimensional plane with classification. This dataset is represented as two groups of dots, as shown in Figure 1. SVM uses a maximisation problem to draw a line between the two groups of points. The distance from the line to each class of points has to be

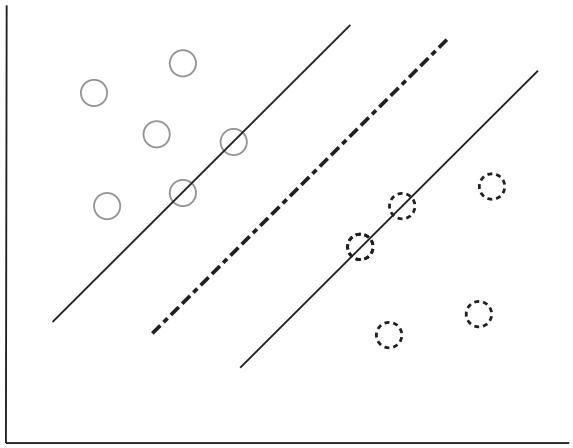

Figure 1: This is a simple example of a 2-dimensional classification problem. The dotted line is the line that separates the two classes of nodes. The Support vectors are indicated by a continuous line.

maximised. This maximisation is done using Lagrange multipliers. The distance from the line to each of the two classes of dots has to be maximised, subject to that the line has to be in between the two group of dots.

From this optimisation, two support vectors follow. These two vectors indicate the decision line between the two groups of dots. These two vectors are as far apart from each other as possible, due to the optimisation. These vectors are tangents of the two classes of dots.

The input is given in a feature space. Several features can determine the place of an object in the space. The features are characteristics of objects. For some feature spaces the objects are not in such way that an easy classification can be given. The solution to this problem is to map the objects from the feature space to a higher order feature space. This is done using a kernel function. In this higher order feature space a SVM classification can be made. A new object can be classified by mapping it to the higher order from the normal feature space and than its class can be estimated.

For some problems it is not possible to give a perfect classification based upon the training set. The solution to this is to allow some objects to be out of the classification line. This means that some objects are not in the correct corresponding side of their class. This is named the soft margin [13]. The soft margin determines how many objects are 'allowed' to lay outside of the classification.

SVM and Time Series

In finance several examples exist of the usage of SVM for time-series forecasting. Tay [31] used a five day Relative Price Difference (5-RDP) to do forecasting. The method used here is to train the SVM on 5-RDP values from the past. Kim [22] did research on the number of dimensions the feature space should have. A higher order feature space gives more possibilities to do accurate classification on. The problem of over fitting can be overcome to choose an order of dimension which is not too high [31].

2.2 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are a model of computing that performs pattern recognition tasks in a manner that is inspired by the structure and performance of a biological neural network, such as the one found in the human brain. ANNs are extensively described in books by Duda et al. [17] and Yegnanarayana [36].

Attractive features of a biological neural network include robustness and fault tolerance, flexibility, the ability to deal with a variety of data situations and collective (parallel) computation.

The performance of ANNs will probably never come close to that of a biological neural network, if not only for that we do not fully understand the operation of biological neurons. Furthermore the number of interconnections and asynchronous nature make it impossible to simulate; the human brain, for example, has about 85 billion neurons with 10^{14} to 10^{15} interconnections.

However, we can utilise some features of a biological neural network by constructing a network consisting of basic interconnected processing units. The general model of a processing unit consists of a function part followed by an output part. The function part receives N input values and applies an *activation function* to derive the so-called the *activation value*. For example, a very simple activation function could sum all inputs, making the activation value the sum of all inputs. The output part produces a signal from this activation value.

Several processing units are interconnected according to a predefined topology to accomplish a specific task. The inputs to a processing unit may come from outputs of other processing units and/or from external sources. The amount of output of one unit received by another unit is influenced by the strength of the connection, reflected in a weight value associated to the link. Each of the N units in a given ANN has, at any given instance of time, a unique activation value and a unique output value. The set of N activation values of the networks defines the *activation state* at that instant. A schematic diagram of a simple neural network can be found in Figure 2.

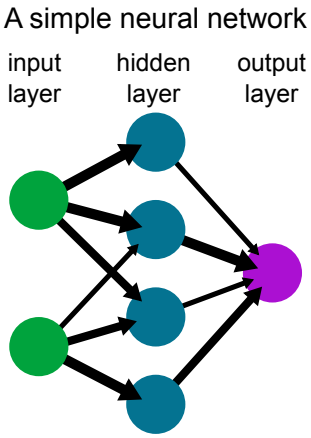

Figure 2: Schematic diagram of a simple neural network. [35]

To update the output states of all units, multiple strategies are possible: the output could be updated asynchronously, changing all activation values at the same time, or sequentially, taking the output state of the network into account each time. For each unit, the output state can be determined from the activation value either *deterministically* or *stochastically*. ANN models use more complex equations to govern activation dynamics according to task to be handled.

ANNs and Time Series

ANNs are normally used for pattern recognition tasks. To make them suitable for time series prediction, a multilayer perceptron (MLP) or radial basis function (RBF) topology is used, with a set of N-tuples as inputs and a single output as the target value of the network. These techniques are extensively described in papers by Connor [12] and Frank et al. [18].

A MLP topology consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Each node (except for input nodes) is a neuron with a nonlinear activation function. MLP utilises backpropagation to train the network, i.e. for a desired output, the network learns from many possible inputs.

In a RBF network, radial basis functions are used as activation functions. Radial basis functions are a means to approximate multivariate functions by linear combinations of terms based on a single univariate function. A full introduction to radial basis functions can be found in a book by Buhmann [8]. Radial basis functions of inputs and neuron parameters are combined into a linear combination which then forms the output of the network. RBF networks are typically trained by a two-step algorithm, with backpropagation as an optional third step to fine-tune the network.

2.3 Genetic Algorithms (GA)

A Genetic Algorithm (GA) is a search heuristic that arrives at its solution in a way derived from natural selection and evolution, or 'survival of the fittest' - the 'fittest' in this case being the most optimal solutions to the problem the algorithm is trying to solve. Based on these principles, GAs use principles of evolution such as population, selection, reproduction and mutation, GAs evolve a sample of candidate solutions to a problem over 'generations', improving upon the best solutions so far while still leaving room for deviation from the norm. To illustrate the workings of GAs, we will take you through the process step by step.

Step 1: Express the solution domain genetically
Solutions to the problem should be encoded in a way that allows for the GA to easily perform necessary operations such as crossover and mutation. Usually, this is done by encoding the solution as an array of bits, though other options are also available, such as using arrays of other types or data structures such as trees.

Step 2: Define a fitness function
A fitness function is a function that, when provided with an encoded solution, expresses how 'fit' that solution is to solve the problem in a fitness score. A higher score indicates a better solution.

Step 3: Initial population
Having found a way to encode solutions, an initial population A of n possible solutions is randomly generated.

Step 4: Calculate fitness of initial population
Using the fitness function, the fitness of every member of A is calculated.

Step 5: Make selection from current population
Now that we know the fitness of every member of A, we want to create a new generation from the fittest individuals the current population has to offer. A way to select suitable individuals is through roulette wheel selection: we randomly select individuals, with individuals with a higher fitness score having a higher chance to make it into the selection: This can be compared to a roulette wheel where some slots are bigger than others, hence the name.

Step 6: Crossover and mutation
With this selection, crossover and mutation are used to randomly generate the 'offspring' of these solutions. When using bitstrings to represent solutions, a crossover operation would combine parts of two solutions from the selection into a new bitstring, and a mutation would take an individual from the selection and randomly alter one or more of its bits. Depending on how much the population should change between generations, the odds of crossover and mutation occurring can be increased or decreased.

Step 7: Insert offspring into new population
The resulting offspring of the previous step, along with some individuals from the selection, is added to a new population B for the next generation of solutions.

Step 8: Repeat Steps 5-7 until both populations have the same size
Keep generating offspring until the new population is ready to replace our old one.

Step 9: Replace old population by the new one
Having replaced A with B, there is a new set of individuals whose fitness can be evaluated.

Step 10: Repeat steps 4 through 9 until stopping conditions are reached
The algorithm typically terminates after a certain amount of generations, or after the population has reached a certain level of fitness.

Because a solution with a higher fitness score has a higher chance to 'survive', the average fitness over the population will increase over time, and because of the differentiating effect of crossover and mutation, solutions are (somewhat) protected against clustering around a suboptimal local peak. In this way, Genetic Algorithms, when given enough time and having variables such as crossover rate, mutation rate and population assigned appropriately, will be able to arrive at a solution to a wide variety of problems, and they are indeed employed today in many fields, ranging from bioinformatics and chemistry to economics and finance.

A specialisation of GAs that is often used in stock market prediction is Genetic Programming (GP). The distinguishing feature of GP is that its population is built up of computer programs. By measuring each individual program's performance, GP tries to find a program that is best suited to performing a certain computational task and providing the desired output. In Section 5 we will elaborate further on how this approach, or GA in general, can be used in predicting the stock market and making investment decisions.

GAs and Time Series

Using GAs with time series can be done in several ways. Two examples are predicting future stock price or trying to find a profitable strategy for a market, given historical data. A way to encode solutions in these two situations is in the form of polynomial functions, which allows for the easy application of mutation and crossover operations. These could be built up as a tree structure, with mutation changing branches and crossover switching branches around between trees. Fitness can be measured in the first case by seeing how close the solution gets to correctly predicting stock price or market trend (up or down), or, in the second case, by measuring how much profit (or loss) a generated strategy would have made over the given time period. A good example of this approach was given by Neely et al. in their 1997 paper. [28]

3 Financial Background

In this section, we will give a brief overview of some concepts used throughout the paper. In section 3.1 an introduction to Technical Analysis is given. This principle relates to the forecasting of stocks. In section 3.2 multiple indicators are shown, which can be used in Multi-Objective Learning for Finance, this directly relates to the next section, section 4. These indicators are indicators an investor would directly use.

3.1 Technical Analysis (TA)

Technical analysis has been around for quite some time now, but this part of the financial analysis is still often referred as pseudo-science, and its relation to finance is considered by some to be the same as alchemy is to chemistry. The goal of technical analysis is to forecast the value of assets, without any knowledge about the company itself. Instead, TA bases its predictions on the historic price path of the asset. The three main assumptions of technical analysis are as follows:

- **The market discounts everything:** TA assumes that, at any given time, a stock's price reflects all aspects affecting the company, and as such there is no need to account for fundamental factors, broad economic factors or market psychology in a strategy. It follows that the investor then only needs to consider the supply and demand of a stock in the market, which can be derived from the movement of the stock's price.

- **Prices move in trends:** The second of the basic assumptions of TA is that prices trend directionally: up, down, or sideways or some combination. If a trend can be found in a stock's past performance, the analyst can use this to predict the stock's future behaviour, as it is likely to keep following its current trend.
- **History tends to repeat itself:** In terms of price movement, history is assumed to repeat itself. This is believed to be caused mainly by market psychology: because investor behaviour repeats itself, recognisable and thus predictable price patterns should emerge over time.

The counterpart to TA is the more traditional method of fundamental analysis. Fundamental analysis is based on the efficient market hypothesis (EMH), which states that the attributes of a company, such as profit, size, etc., are immediately reflected in the price of the asset. The efficient market hypothesis exists in several forms. The weak efficient market hypothesis states that all available knowledge from the past is already reflected in the price. The semi-strong efficient market hypothesis holds that prices reflect all publicly available information and change instantly when this information changes. The strong efficient market hypothesis states that inside knowledge is also directly reflected in the price and that therefore inside knowledge does not have any value.

The goal of TA is to forecast the price paths of assets. This has been done by optical pattern recognition - the most basic example being holding a ruler over a printed graph and extrapolating along the ruler. There are a lot of theories on how price paths behave. As an example we consider the work of Osler [29]. Osler considers two observation about trends: first, that trends tend to reverse course at predictable support and resistance levels, and second, that trends tend to be unusually rapid after rates cross such levels. These two observations about trends are just some of the many aspects considered in TA.

3.2 Indicators in Multi-Objective Learning for Finance

The usage of multi-objective machine learning in finance is based upon indicators. These indicators are measures of a stock's various attributes, and can be weighed by an algorithm to decide on a trading strategy.

A classification can be made upon the indicators for machine learning. The most fundamental one is the historic price path. Others are the fundamental indices and the macroeconomic indices as proposed by Tsai [33]. Fundamental indices include elements used in fundamental analysis e.g. net income growth, cash flow ratio, and return on assets. Macroeconomic indices include for example the import and export values. In this paper we will not consider these macroeconomic indices.

When making investment decisions, the trader is confronted by a very large pool of available indicators to use in devising their strategy. An overview of indicators is given in Table 1.

4 Multi-Objective Optimisation

In this section, we will give an introduction to technical analysis and the basic principles of multi-objective optimisation. We will explain why an investor would want to take a multi-objective approach to portfolio optimisation.

4.1 Basic Principles of Multi-Objective Optimisation

The basic concepts of multi-objective optimisation are elegantly described by Castillo-Tapia et al. [10], as follows.

Without loss of generality, we will assume only minimisation problems and are interested in solving problems of the type:

$$\text{minimise } \vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \tag{1}$$

Feature	Description	Formula	Refs.
%K	Stochastic %K. It compares where a security‘s price closed relative to its price range over a given time period.	$\frac{C_t - LL_{t-n}}{HH_{t-n} - LL_{t-n} \times 100}$, where LL_t and HH_t mean lowest low and highest high in the last t, respectively.	[22, 1]
%D	Stochastic %D. Moving average of %K.	$\frac{\sum_{i=0}^{n-1} \%K_{t-i}}{n}$	[22, 1]
Slow %D	Stochastic %D. Moving average of %K.	$\frac{\sum_{i=0}^{n-1} \%D_{t-i}}{n}$	[22]
Momentum	It measures the amount that a security‘s price has changed over a given time span.	$C_t - C_{t-4}$	[22]
ROC	Price rate-of-change. It displays the difference between the current price and the price n days ago.	$\frac{C_t}{C_{t-n}} \times 100$	[22]
Williams’ %R	Larry William’s %R. It is a momentum indicator that measures overbought/oversold levels.	$\frac{H_n - C_t}{H_n - L_n} \times 100$	[22, 1]
A/D Oscillator	Accumulation/distribution oscillator. It is a momentum indicator that associates changes in price.	$\frac{H_t - C_{t-1}}{H_t - L_t}$	[22]
Disparity5	5-day disparity. It means the distance of current price and the moving average of 5 days.	$\frac{C_t}{MA_5} \times 100$	[22]
Disparity10	10-day disparity.	$\frac{C_t}{MA_10} \times 100$	[22]
OSCP	Price oscillator. It displays the difference between two moving average of a security’s price.	$\frac{MA_5 - MA_10}{MA_5}$	[22, 1]
CCI	Commodity channel index. It measures the variation of a security’s price from its statistical mean.	$\frac{(M_t - SM_t)}{(0.015 D_t)}$ where $M_t = (H_t + L_t + C_t)/3$, $SM_t = \frac{\sum_{i=1}^{n} M_{t-i+1}}{n}$, and $D_t = \frac{\sum_{i=1}^{n} \lvert M_{t-i+1} - SM_t \rvert}{n}$.	[22, 1]
RSI	Relative strength index. It is a price following an oscillator that ranges from 0 to 100.	$100 - \frac{100}{1 + (\sum_{i=0}^{n-1} Up_{t-1}/n)/(\sum_{i=0}^{n-1} Dw_{t-1}/n)}$ where Up_t means upward-price-change and Dw_t means downward-price-change at t.	[22, 1, 34]
MACD	Moving average convergence divergence. It shows the relationship between two moving averages of prices.	$MACD(n)_{t-1} + n/2 + 1 \times (DIFF_t - MACD(n)_{t-1})$, $DIFF_t = EMA(12)_t - EMA(26)_t$,	[21]

C_t is the closing price at time t, L_t the low price at time t, H_t the high price at time t, MA_t the moving average of t days, and EMA the exponential moving average.

Table 1: Indicators

subject to:

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{2}$$

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables, $f_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, k$ are the objective functions and $g_i, h_j : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m,\ j = 1, \ldots, p$ are the constraint functions of the problem.

We are interested in the concept of multi-objective optimality, which can be more formally described through the following definitions:

Definition 1. Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^k$, we say that $\vec{x} \leq \vec{y}$ if $x_i \leq y_i$ for $i = 1, \ldots, k$, and that $\vec{x}$ **dominates** $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x} \leq \vec{y}$ and $\vec{x} \neq \vec{y}$

Definition 2. We say that a vector of decision variables $\vec{x} \in \mathcal{X} \subset \mathbb{R}^n$ is **non-dominated** with respect to $\mathcal{X}$, if there does not exist another $\vec{x}' \in \mathcal{X}$ such that $\vec{f}(\vec{x}') \prec \vec{f}(\vec{x})$.

Definition 3. We say that a vector of decision variables $\vec{x}^* \in \mathcal{F} \subset \mathbb{R}^n$ ($\mathcal{F}$ is the feasible region) is **Pareto-optimal** if it is non-dominated with respect to $\mathcal{F}$.

Definition 4. The **Pareto Optimal Set** $\mathcal{P}^*$ is defined by:

$$\mathcal{P}^* = \{\vec{x} \in \mathcal{F} \mid \vec{x} \text{ is Pareto-optimal}\}$$

Definition 5. The **Pareto-front** $\mathcal{PF}^*$ is defined by:

$$\mathcal{PF}^* = \{\vec{f}(\vec{x}) \in \mathbb{R}^k \mid \vec{x} \in \mathcal{P}^*\}$$

The image of the Pareto optimal set forms the Pareto front which is normally displayed in a graphical form in order to determine the most desirable combination of objectives. We thus wish to determine the Pareto optimal set from the set $\mathcal{F}$ of all the decision variable vectors that satisfy equations 2 and 3. When dealing with real-world problems, in practice, not the entire Pareto optimal set is normally desirable (e.g. it may not be desirable to have different solutions that map to the same values in objective function space) or achievable.

4.2 Reasons for Multi-Objective Optimisation

In the financial world, the performance of traders is not only evaluated based on the profit they make. Profit is certainly an important aspect of performance, but metrics related to risk are also important indicators of how a trader is doing. A trader who implements a trading rule which maximises profit regardless of the risks involved is, so to speak, 'asking for trouble'.

The general idea of portfolio management is to have a portfolio which maximises profit under as little risk as possible. Modern portfolio theory is based on theories formulated by Markowitz [26]. Markowitz believes that as an investor, an optimal portfolio does not exist, because there is an inherent trade-off between risk and profit.

This trade-off is a key argument to justify using multi-objective machine learning. A trader does not just want to maximise profit, but wants to find an optimal point between profit and risk. In technical analysis, the objectives are composed of multiple fitness functions.

Closer to our research question is the work of Sagi et al. [7], who proposed four objectives. The objectives are: maximising profit, minimisation of transaction cost, minimisation of risk trend, and minimisation of risk correlated to an index.

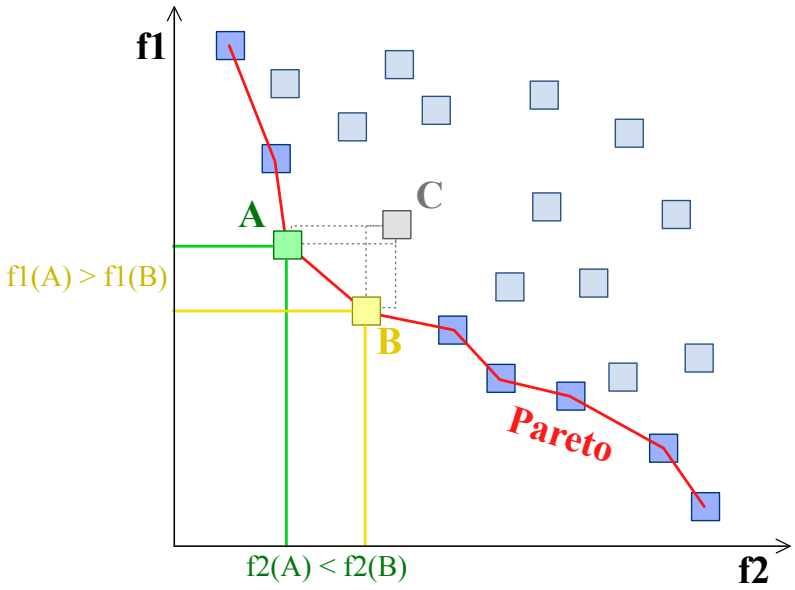

Figure 3: Graphical representation of a Pareto frontier. [16]

5 Techniques for multi-objective optimisation

In the Section 2, we examined several Machine Learning techniques and textbfsomewhere else we outlined the investor's dilemma of balancing many priorities. Moreover we explained why multi-objective machine learning is a good solution to the problem of predicting the behaviour of the stock market and assisting traders in devising effective trading strategies. We now turn our attention to the application of support vector machines, artificial neural networks, and finally multi-objective genetic algorithms to this problem.

5.1 Support Vector Machines (SVM)

Earlier work on SVM mainly focused on pattern recognition of images. The application to finance was made later. A study by Mukherjee [27] applied SVM to a chaotic time series. The experiments showed that SVM is able to forecast a time series better than other techniques as neural networks. Time series can be used to represent financial data, but the time series used here was generated by the researchers themselves. Therefore, it is uncertain whether their conclusions can actually be applied to 'real' financial time series. This is due to the fact that in this work time series were considered of a different origin. It is possible that time series of finance behave different in these models.

Work by Kim [22] concluded that it is indeed possible to forecast financial time series. The time series were taken from futures prices on the Chicago Mercantile Market: the results were that SVM outperformed back-propagation neural networks. By employing technical analysis indicators, Kim was able to successfully optimise his strategy using SVM.

Most studies we found only used a small input size for their SVM or other machine learning techniques, and only a small amount of features. Some studies, however, consider a large number of input features [9]. The twenty input features used in this study is more than in most studies. The features used in this study are again technical analysis indicators, such as the RSI and the MASD. Experiments were performed on the S&P 500 index, attempting to predict the closing price of the next day. The usage of the twenty features on the whole index led to a forecasting result with an accuracy of 86% .

Another study has shown that a trading system using SVMs can outperform other trading

strategies [24]. This study compared a trading strategy using SVMs to a Technical Analysis Trader, a Random Trader and a few other traders. They have shown that the SVM trader outperformed the other traders.

SVM in a hybrid setting

In general, SVM are not very effective when they are used in multi-objective settings. It is possible to approximate multi-objective behaviour with a good fitness function that optimises several parameters, but, as explained previously, this is not a very effective solution.

SVM can be used, however, in a multi-objective GA setting, as shown by Ayhin, Karakose and Akin[5], by using SVM as an intermediate step. SVM is applied to each objective, so multiple SVMs exist per iteration. This technique has been described as 'parallel learning' [6]. Although it has been proven to work for only two other cases, the results so far seem promising. The experiments in this work were performed on a three-phase engine, but we believe that this technique of having multiple objectives, optimising them individually, and then using GA to combine them can also be applied to finance.

This same kind of optimisation has also been performed in other fields, as shown in work by Igel [20]. Experiments were performed using a common medical benchmark dataset, and the results were modelled on a Pareto frontier. Their results showed that SVMs can help achieve better results in multi-objective optimisation.

A work more related to the traditional time series analysis combined ARIMA with SVMs [30]. The ARIMA (Autoregressive integrated moving average) analysis has been widely used in time-series analysis. The ARIMA model is an extension of the moving average model; in ARIMA the volatility also moves over time. This work has shown evidence that SVM can help better forecast time series. The forecasting errors were reduced. Optimal parameter selection is not achieved – this is still left for future research. Although ARIMA is not part of the 'technical analysis school', it can be used to forecast stocks.

5.2 Artificial Neural Networks

Artificial Neural Networks are not directly suited for multi-objective optimisation problems. However, they have been successfully applied for financial optimisation problems in a single-objective setup. Additionally ANNs can be combined with other techniques to form hybrid machine learning strategies. We will review several papers that researched ANNs in this area.

Chen et al. [11] model and predict the direction of return on the market index of the Taiwan Stock Exchange. A probabilistic neural network (PNN) is used to forecast the direction of index return after it is trained by historical data. These forecasts are compared with generalised methods of moments (GMM) with a Kalman filter. In addition, forecasts are applied to various index trading strategies, and performances are compared with those generated by a 'buy-and-hold' strategy: that is, simply buying stocks and holding on to them for a long period of time before selling, as opposed to trading frequently. Performance is also compared to investment strategies guided by forecasts estimated by the random walk model and parametric GMM models. Empirical results of their study show that the PNN-based investment strategies obtain higher returns than other examined investment strategies.

Lam [23] investigated the usage of neural networks, specifically the backpropagation algorithm, to integrate fundamental and technical analysis for financial performance prediction. She compares the performance of neural networks with a minimum and maximum benchmark. As the maximum benchmark the average return from the top one-third returns in the market that approximates the return from perfect information is used. The minimum benchmark is the overall market average return that approximates the return from highly diversified portfolios. This study's experimental results indicate that neural networks using one year's or multiple years' financial data consistently and significantly outperform the minimum benchmark, but not the maximum benchmark. Neural networks with both financial and macroeconomic predictors do not outperform the minimum nor maximum benchmark in this study. Moreover, the study has a demonstration of rule extraction

as a post-processing technique to improve prediction accuracy using expert knowledge and for explaining prediction logic to financial decision makers.

Tsai et al. [32] used ANNs in combination with a decision tree (DT) to form a hybrid machine learning technique for stock price forecasting. Their experimental result shows that the combined model has a 77% accuracy, which is higher than single ANN and DT models. However, they do not compare against other algorithms such as GA or SVN. They also indicate that a GA could be used to pre-process the data for better performance.

In a newer paper, Tsai et al.[33] propose a hybrid algorithm which is based upon several machine learning techniques to select the right indicators. The basic technique to select good indicators is Principle Component Analysis or PCA, a procedure that transforms a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables (called principal components). Other methods include machine learning techniques such as a back-propagation neural network, genetic algorithms, and decision trees (CART). Experimental results show that the intersection between PCA and GA and the multi-intersection of PCA, GA, and CART perform the best. Their experiments show that the 85 original variables could be reduced to the 14 and 17 most important features respectively.

5.3 Genetic Algorithms (GA)

Early explorations into portfolio optimisation using GAs yielded varying, but mostly positive results. Only a single, simple objective played a role: making as much profit as possible. This was usually measured by the excess return the algorithm's strategy would have provided over a standard buy-and-hold strategy.

However, genetic algorithms can also be used in a multi-objective setup as introduced in the previous section. In this use case they are usually referred to as 'multi-objective evolutionary algorithms' or MOEAs.

Anione et al. [4] were (one of) the first to use MOEAs in the area of investment portfolio optimisation. They adopt a GA with a weighted linear aggregating function as the basis of their research. As an objective they use a minimisation of risk and a maximisation of (expected) return, introducing a trade-off coefficient between the two factors. To achieve this using GA several different populations are to be evolved for a number of trade-off coefficient values. These different populations form different portions of the Pareto frontier. Of course, the higher the number of populations, the more granular the Pareto frontier will be. Loraschi et al. [25] pursued this research and found that a distributed GA approach outperformed the sequential version of the algorithm.

Neely et al. [28] used Genetic Programming (a certain 'flavour' of GA) to find technical trading rules for several Foreign Exchange (FX) markets. The rules had a tree structure, which made them suitable for crossover and mutation operations. Trees were made up of arithmetic operations, Boolean operations, conditional operations, numerical constants and Boolean constants. An example trading rule generated by this approach could be: $avg(15) > avg(250)$, which means: go long if the 15-day moving average exceeds the 250-day moving average. Neely et al. were very positive about their results, even though they had not put any effort into optimising the genetic process.

Allen and Karjalainen [3] used GA's to construct trading rules in a way similar to the work by Neely et al., but they were not as optimistic about the results: they concluded that their simulation did not earn significant excess returns over a simple buy-and-hold strategy, and that their work provided little evidence for the existence of economically significant technical trading rules.

Furthermore, Dempster and Jones [14], using a similar genetic programming strategy and building on their own previous work, concluded that though the majority of indicator-based rules were loss-making individually, it was possible to profit from technical trading rules when they are constructed by combining several different indicators instead of using individual ones: in this case, they found it was possible to make a moderate, but significant profit.

Finally, Dempster et al. [15] conducted further research to compare (combinations of) several machine learning techniques in the area of intraday FX trading. While most research focusses on optimisation using closing prices, i.e. one data point per day, many trading firms instead leverage intraday high-frequency data to be able to trade during the day. The aim here is to end the day with a net open position of zero, instead of using much longer horizons as is customary when trading using closing prices. In their research they consider several strategies that use a collection of technical indicators as input and seek a profitable trading rule defined in terms of them. The strategies are approached using reinforcement learning and a GA, and compared to an appropriate Markov decision problem and a simple heuristic. They find the GA approach to deliver superior results at nonzero transaction costs, although they do mention that none of the evaluated algorithms produce 'significant profits' at more realistic transaction costs.

A study from Bodas-Sagi [7] performs experiments using MOEAs. This system uses the simple trading role to buy and hold a asset for a few days. The forecasts are given for a day, a week and a month. Algorithms are rated as a fraction of the forecast they have correctly forecasted. An algorithm only forecasts if the price will be lower or higher as it is now. The system reviews several combination of systems. The objectives which are maximised to are RSI and MACD, these terms are explained in table 1. The best result was achieved using a MIX of objectives. This means combining all objectives together yields the best results. Other methods that were compared to were single-objective optimisations and multi-objective optimisations based upon specific components. Experiments were performed using the S&P 500 index. This paper shows evidence that multi-objective machine learning can play a role in forecasting asset price paths.

Yet another study (Becker et al. [6]) performed experiments in the MOEA field. The researchers used three fitness functions that are measures of the fitness of a stock in a portfolio. The question at hand was which stocks should be selected in a portfolio. Experiments were performed using the S&P 500, excluding some companies. A test was performed to compare three evolutionary techniques, based on the fitness functions for optimisation: a parallel algorithm, a sequential algorithm and a constrained fitness function. The parallel algorithm did outperform their benchmark and the constrained fitness function produced results close to the parallel algorithm.

6 Conclusions and Future Research

In this paper, we examined the application of various machine learning techniques on the area of investment portfolio optimisation. After investigating the advantages and disadvantages of artificial neural networks, support vector machines and genetic algorithms, we came to the conclusion that genetic algorithms are most suited for a multi-objective optimisation approach. We examined some of the theory behind multi-objective investment and technical analysis, and the arguments for and against the latter. Furthermore, we also described in detail the application of multi-objective genetic algorithms to portfolio optimisation.

Though we have found evidence in favour of theory, we have found that technical analysis is a field that is still very much lacking in recognition amongst researchers, mostly because there is a lack of evidence to support the theory. In many of the studies we came across in writing this paper, the research scope was often limited, or experiments were carried out in such a way that the results were either vague or not reliable enough. Often, little to no effort was made to optimise their machine learning process at all: perhaps this was due to a lack of expertise in the relevant computer science areas. The papers we found commonly suggest that further research could or should explore the effects of optimisation. The case could be made that in the past, researchers might not have had access to sufficient computing power or the raw data required to perform thorough research, but it seems apparent that these issues have become less relevant over the years.

In order to tackle the problem of inadequate amounts of computing power, we feel that more research could be done in the area of utilising distributed computing strategies in order to more effectively solve machine learning problems. In the case of genetic algorithms, distributed computing could, for example, help achieve higher population sizes. In the more general case, the

use of larger or more granular training data sets and an increased number of indicators (decision variables) used in a multi-objective approach could produce markedly different results. Even if no significant increase in accuracy is found this way, a distributed computing approach with a noteworthy improvement in speed (run time) could still open the door to more real-world applications of machine learning in finance. Rapid evaluation and improvement of strategies could change the way that trading is performed. Trading is extremely time-sensitive: deploying a strategy without taking the time to properly test and evaluate it carries with it a risk of large losses, but taking too long carries with it the risk of missing out on opportunities for profit. Improved computing speed would allow strategies to be more properly tested and optimised before they are put into use in a real-world trading platform.

Moreover, we feel more thorough studies could definitely help decide the debate about technical analysis. At the moment, many studies have been published, but few to none have the scale required to help change the general opinion on technical analysis in the financial sector. We propose that a large-scale study be carried out, comparing a large number of technical indicators implemented in many different strategies, incorporating the previous suggestion of utilising distributed computing power, will certainly shed new light on technical analysis. Whether these results can provide a definitive breakthrough on the application of technical analysis is to be seen, considering the fair amount of skepticism in this area.

An area of research with relatively little work is usage of intraday data, even though its widespread use in the financial world in the form of high-frequency trading. We think that the application of machine learning techniques to intraday data certainly forms an interesting research direction. Most current experiments are only performed on closing prices of data, probably due to the common availability of this data. Intraday data, especially tick data, is very valuable for more elaborate strategies. A system to evaluate and improve strategies based on technical analysis using machine learning could help a trader make even better decisions in a system using intraday data.

We feel that the use of advanced machine learning techniques could be an enormous asset to research in the general financial domain, and the pursuit of future research will certainly contribute to shaping real-world applications in this area.

Acknowledgements

We would like to express our great appreciation to our supervisor, Mathijs de Weerdt, for his guidance and feedback in writing this paper.

References

[1] Steven B Achelis. *Technical Analysis from A to Z*. McGraw Hill New York, 2001.

[2] Mark O Afolabi and Olatoyosi Olude. Predicting stock prices using a hybrid kohonen self organizing map (som). In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 48–48. IEEE, 2007.

[3] Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of financial Economics*, 51(2):245–271, 1999.

[4] Salvatore Anione, Andrea Loraschi, and Andrea Tettamanzi. A genetic approach to portfolio selection. *Neural Network World*, 6(93):597–604, 1993.

[5] Ilhan Aydin, Mehmet Karakose, and Erhan Akin. A multi-objective artificial immune algorithm for parameter optimization in support vector machine. *Applied Soft Computing*, 11(1):120–129, 2011.

[6] Ying L Becker, Harold Fox, and Peng Fei. An empirical study of multi-objective algorithms for stock ranking. *Genetic Programming Theory and Practice V*, pages 239–259, 2008.

[7] DJ Bodas Sagi, Francisco J Soltero, J Ignacio Hidalgo, Pablo Fernandez, and F Fernandez. A technique for the optimization of the parameters of technical indicators with multi-objective evolutionary algorithms. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[8] Martin D Buhmann. *Radial basis functions: theory and implementations*, volume 12. Cambridge university press, 2003.

[9] Xianggao Cai, Su Hu, and Xiaola Lin. Feature extraction using restricted boltzmann machine for stock price prediction. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, volume 3, pages 80–83. IEEE, 2012.

[10] MG Castillo Tapia and Carlos A Coello Coello. Applications of multi-objective evolutionary algorithms in economics and finance: A survey. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 532–539. IEEE, 2007.

[11] An-Sing Chen, Mark T Leung, and Hazem Daouk. Application of neural networks to an emerging financial market: forecasting and trading the taiwan stock index. *Computers & Operations Research*, 30(6):901–923, 2003.

[12] Jerome T Connor, R Douglas Martin, and LE Atlas. Recurrent neural networks and robust time series prediction. *Neural Networks, IEEE Transactions on*, 5(2):240–254, 1994.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[14] M.A.H. Dempster and C.M. Jones. A real-time adaptive trading system using genetic programming. *Quantitative Finance*, 1(4):397–413, 2001.

[15] MAH Dempster, Tom W Payne, Yazann Romahi, and GWP Thompson. Computational learning techniques for intraday fx trading using popular technical indicators. *IEEE Transactions on neural networks*, 12(4):744–754, 2001.

[16] Johann Dréo. Pareto front [online]. 05 2006. URL: `http://en.wikipedia.org/wiki/File:Front_pareto.svg` [cited 2013-03-12].

[17] Richard O Duda, Peter E Hart, and David G Stork. *Pattern Classification and Scene Analysis 2nd ed.* 1995.

[18] RJ Frank, Neil Davey, and SP Hunt. Time series prediction and neural networks. *Journal of Intelligent & Robotic Systems*, 31(1):91–103, 2001.

[19] Cheng-Lung Huang and Cheng-Yi Tsai. A hybrid sofm-svr with a filter-based feature selection for stock market forecasting. *Expert Systems with Applications*, 36(2):1529–1539, 2009.

[20] Christian Igel. Multi-objective model selection for support vector machines. In *Evolutionary Multi-Criterion Optimization*, pages 534–546. Springer, 2005.

[21] Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, 38(5):5311–5319, 2011.

[22] Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307–319, 2003.

[23] Monica Lam. Neural network techniques for financial performance prediction: integrating fundamental and technical analysis. *Decision Support Systems*, 37(4):567–581, 2004.

[24] Marco Lippi, Lorenzo Menconi, and Marco Gori. Balancing recall and precision in stock market predictors using support vector machines. In *Neural Nets and Surroundings*, pages 51–58. Springer, 2013.

[25] A Loraschi, A Tettamanzi, Marco Tomassini, and Paolo Verda. Distributed genetic algorithms with an application to portfolio selection. *Artificial Neural Networks and Genetic Algorithms*, 1995.

[26] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 2012.

[27] Sayan Mukherjee, Edgar Osuna, and Frederico Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 511–520. IEEE, 1997.

[28] Christopher Neely, Paul Weller, and Robert Dittmar. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of Financial and Quantitative Analysis*, 32(4), 1997.

[29] Carol L Osler. Currency orders and exchange rate dynamics: an explanation for the predictive success of technical analysis. *The Journal of Finance*, 58(5):1791–1820, 2003.

[30] Ping-Feng Pai and Chih-Sheng Lin. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005.

[31] Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, 2001.

[32] CF Tsai and SP Wang. Stock price forecasting by hybrid machine learning techniques. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, page 60, 2009.

[33] Chih-Fong Tsai and Yu-Chieh Hsiao. Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, 50(1):258–269, 2010.

[34] Ray Tsaih, Yenshan Hsu, and Charles C Lai. Forecasting s&p 500 stock index futures with a hybrid ai system. *Decision Support Systems*, 23(2):161–174, 1998.

[35] User:Wiso [online]. 10 2008. URL: `http://en.wikipedia.org/wiki/File:Neural_network_example.svg` [cited 2013-03-26].

[36] B Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2004.

[37] Paul D Yoo, Maria H Kim, and Tony Jan. Machine learning techniques and use of event information for stock market prediction: A survey and evaluation. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 835–841. IEEE, 2005.