Expert Systems With Applications 103 (2018) 184–195

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Combining NeuroEvolution and Principal Component Analysis to trade in the financial markets

João Nadkarni, Rui Ferreira Neves*

Instituto de Telecomunicações, Instituto Superior Técnico, Torre Norte, Av. Rovisco Pais, 1, Lisboa 1049-001, Portugal

ARTICLE INFO

Article history:
Received 10 November 2017
Revised 25 January 2018
Accepted 8 March 2018
Available online 9 March 2018

Keywords:
Financial markets
Trading signal
Technical analysis
Principal Component Analysis (PCA)
NeuroEvolution of Augmenting Topologies (NEAT)

ABSTRACT

When investing in the financial market, determining a trading signal that can fulfill the financial performance demands of an investor is a difficult task and a very popular research topic in the financial investment area. This paper presents an approach combining the principal component analysis (PCA) with the NeuroEvolution of Augmenting Topologies (NEAT) to generate a trading signal capable of achieving high returns and daily profits with low associated risk. The proposed approach is tested with real daily data from four financial markets of different sectors and with very different characteristics. Three different fitness functions are considered in the NEAT algorithm and the most robust results are produced by a fitness function that measures the mean daily profit obtained by the generated trading signal. The results achieved show that this approach outperforms the Buy and Hold (B&H) strategy in the markets tested (in the S&P 500 index this system achieves a rate of return of 18.89% while the B&H achieves 15.71% and in the Brent Crude futures contract this system achieves a rate of return of 37.91% while the B&H achieves −9.94%). Furthermore, it's concluded that the PCA method is vital for the good performance of the proposed approach.

1. Introduction

When studying the stock market, determining the best time to buy or sell stocks is of key importance. However, obtaining a trading signal that finds good entry and exit points in the stock market is difficult due to the stock market's noisy, non-stationary and non-linear characteristics. In fact, stock markets are affected by many highly interrelated factors, such as psychological variables associated with the investors as well as economic and political factors.

The efficient market hypothesis (Fama, 1970) states that it's impossible to gain a competitive advantage over the market because the prices already incorporate and reflect all available information in the market at that moment. On the other hand, some researchers believe that the markets are inefficient, mainly due to the psychological variables associated with the various market participants and the inability of the markets to immediately respond to newly released information and so, there is an opportunity to beat the market and obtain above average returns by using stock market forecasting techniques (Gorgulho, Neves, & Horta, 2011).

In this paper, an approach combining a neuroevolution technique, the NeuroEvolution of Augmenting Topologies (NEAT), with a dimensionality reduction technique, the Principal Component Analysis (PCA), is presented. The PCA technique performs a linear mapping of the high dimensional space input data, consisting of raw financial data with some indicators obtained using technical analysis, to a lower dimensional space such that the variance of the data in the low dimensional representation is maximized. This dimensionality reduction of the data facilitates the identification of patterns by the artificial neural network (ANN) that is generated by NEAT. The neural network generated by NEAT outputs a trading signal that evolves with the help of an adapted genetic algorithm (GA) to a solution that maximizes the returns and daily profits while minimizing the associated risk.

The main contributions of this paper are: the combination of the data dimensionalty reduction technique PCA with the NEAT algorithm to identify the best trading points in the stock market; the use of fitness functions in NEAT that take in consideration not only the returns obtained but also the daily profits, risk and number of days spent with capital invested in the market; the integration of the hyper mutation technique in the NEAT algorithm that adjusts not only the overall mutation rate but also the probability of the mutation to add a new node to the ANN generated by NEAT; the possibility of the ANN originated by NEAT to have different activation functions in the hidden neurons if that helps the evolution process to achieve a better performing solution.

* Corresponding author.
E-mail addresses: rui.neves@tecnico.ulisboa.pt, rui.neves@ist.utl.pt, rfmfn@lx.it.pt (R. Ferreira Neves).

https://doi.org/10.1016/j.eswa.2018.03.012
0957-4174/

Table 1
Results of some studies relevant for this paper.

Ref.	Method	Data	Period	Financial market	Algorithm performance	B&H
Mańdziuk and Jaruszewicz (2009)	ANN with GA for feature selection	Stock price (daily)	07/04/2004–26/08/2004	GSE	5.44% (average)	−5.46%
Chiang, Enke, Wu, and Wang (2016)	ANN with Particle Swarn Optimization to optimize initial weights and Wavelet for denoising	Stock price (daily)	2010	BSESN/ SPY	41.54% / 41.89%	13.24%/ 13.69%
Göçken, Özçalıcı, Boru, and Dosdoğru (2016)	ANN with GA for feature selection	Stock price (daily)	28/05/2013–20/09/2013	BIST100 Index	1.12%	−13.41%
Sandström, Herman, and Ekeberg (2015)	NEAT	Stock price (daily)	2012–2014	Apple / Microsoft / Yahoo	56.81% / 13.19% / 16.69% (Best result, without transaction costs)	NA
Zhonga and Enke (2017)	ANN with PCA for dimensionality reduction	Stock price (daily)	30/11/2011–31/05/2013	SPY	36.1% (Best result, without transaction costs)	30.8%
Qiu et al. (2016)	ANN with GA to optimize initial weights	Stock price (monthly)	Jan 2008–July 2013	Nikkei 225 Index	0.0090 (mean squared error of the month's predicted return against the actual return)	NA
Qiu and Song (2016)	ANN with GA to optimize initial weights	Stock price (daily)	19/10/2012–30/12/2013	Nikkei 225 Index	81.27% (hits on direction of market movement)	NA
Inthachot, Boonjing, and Intakosum (2016)	ANN with GA for feature selection	Stock price (daily)	2009–2014	SET50 Index	63.6% (hits on direction of market movement)	NA

This paper is organized as follows: in Section 2 the related work is discussed. Section 3 presents the architecture and describes the implemented system. In Section 4 the case studies and results are presented and analyzed. Section 5 provides the conclusions obtained by the work developed.

2. Related work

Finding the best time to buy or sell has remained a difficult challenge as there are several factors that may impact stock prices (Chang & Liu, 2008). With the growth of the trading business, investors tried to find methods and tools to accurately predict the share prices, in order to increase their gains and minimize the risk (Khan, Alin, & Hussain, 2011). Methods like fundamental analysis, technical analysis and machine learning have all been used to attempt to find the best entry (buy) and exit (sell) points and gain advantage over the market.

Fundamental analysis looks at the basic or fundamental financial level of a company's business with the goal of forecasting and profit from future price movements as shares of companies with strong fundamentals may rise over time while companies with weak fundamentals may see their stock prices fall. This makes fundamental analysis especially valuable to long-term investors. On the other hand, technical analysis forecasts the future financial price movements based on an investigation of past price movements. This type of analysis doesn't attempt to measure a company's intrinsic value, but involves an analysis of the market's activity, such as price and volume, to identify patterns that can be used as a basis for investment decisions. This makes the technical analysis more suitable for short-term trading, like daily trading as it is the case of the system proposed in this paper. Over the years, numerous technical indicators have been developed by analysts in attempt to accurately forecast future price movements. A more detailed description of technical analysis and technical indicators can be found in Kirkpatrick and Dahlquist (2015) and Murphy (1999).

In recent years, the advances in the field of artificial intelligence have offered unprecedented trading opportunities and the forecasting models based on simple fundamental or technical analysis have been surpassed by models incorporating machine learning and data mining methods. Machine learning algorithms can process a big amount of past financial data (as well as other information that affect the market, as the news for example), that may seem uncorrelated and noisy to the human eye, to detect patterns and predict future outcomes of the market. This approach facilitates a fast reaction to events that affect the market and so, can be a competitive advantage in identifying the ideal entry and exit points.

In Table 1 it's summarized some of the most relevant studies applied to the stock market that were analyzed throughout the development of the approach presented in this paper.

2.1. Principal Component Analysis (PCA)

In machine learning having data of high dimensionality can lead to a high computational cost and overfitting, decreasing the performance of the system. Dimensionality reduction focuses on representing data with a minimum number of dimensions, reducing the underlying complexity in processing the data while retaining its vital proprieties. Principal Component Analysis (PCA) is one of the most popular unsupervised linear techniques for dimensionality reduction (Zhonga & Enke, 2017). PCA is a statistical procedure for identifying a smaller number of uncorrelated features, called principal components, from a large set of features.

PCA uses orthogonal transformation to convert the high dimensional data to the principal components. Because the core of PCA is the rotation of space coordinates (which does not change the data structure), the obtained principal components are a linear combination of the original features that reflect as much as possible the original information (Wang & Wang, 2015). This transformation is such that the first principal component has the largest possible variance and each succeeding component has the highest possible variance under the constraint that it must be orthogonal to the preceding components. The number of principal components is equal to the original data's number of dimensions, but when using PCA for dimensionality reduction only some of those principal components (that capture most of the original data variance) are retained.

An ANN with high dimensional data as input requires a big amount of data to generalize and is prone to overfitting. Thus, PCA is frequently used to reduce the dimensionality of the inputs of a neural network. Zhonga and Enke (2017) studied the effect of three dimensionality reduction techniques to transform the input

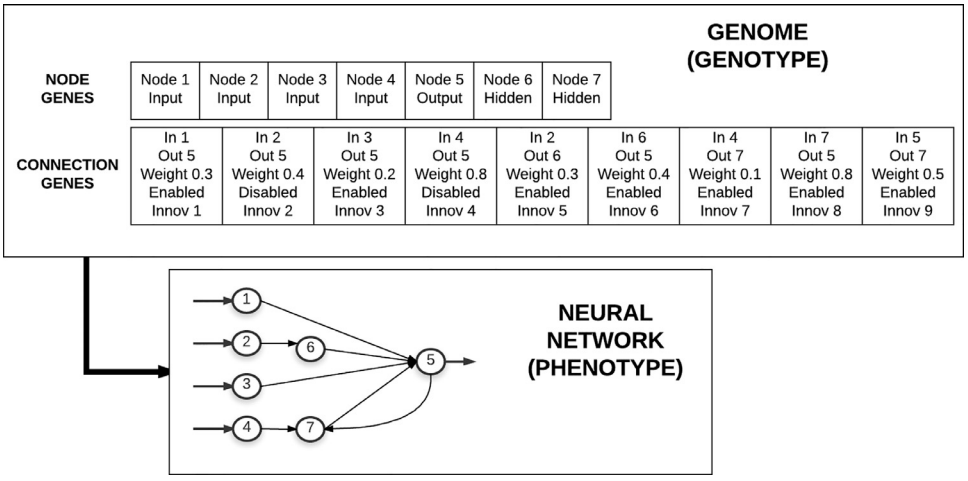

Fig. 1. NEAT genetic encoding.

data of a neural network with the purpose of forecasting the daily direction of future market returns, and concluded that the combination of PCA with the neural network gives a higher classification accuracy than the combination of a neural network with the other two dimensionality reduction techniques studied. Wang and Wang (2015) also investigated the integration of PCA with a neural network, developing a stochastic time effective function neural network in combination with PCA for financial time series prediction and concluded that the incorporation of PCA improved the performance of the stochastic time effective function neural network. Furthermore, Zahedi and Rounaghi (2015) applied a neural network in conjugation with PCA to predict the prices of the Tehran Stock Exchange and concluded that PCA could effectively identify the principal factors in stock prices.

A more extensive and detailed description of the PCA technique can be found in Jolliffe (2002).

2.2. Neuroevolution

Neuroevolution is an evolutionary approach to construct artificial neural networks (ANNs), being inspired from the evolution of biological nervous systems in nature.

An evolutionary algorithm that has been frequently hybridized with ANNs is the genetic algorithm (GA). In 1990, Whitley, Starkweather, and Bogart (1990) began to use the GA to optimize the weighted connections and find a good architecture for the ANN connections. Since then, many studies have used the GA to overcome the drawbacks of the backpropagation approach, obtaining results that support the notion that GAs can enhance the performance of ANN models and reduce the time required for experiments (Chang, Wang, & Tsai, 2005; Chang, Lin, Shieh, & Abbod, 2012; Jadav & Panchal, 2012; Kuo & Chen, 2004).

In recent years, neuroevolution systems have been applied extensively in the financial markets. Perwej and Perwej (2012) employed a GA to choose the optimal topology of the ANN which was used to investigate the daily excess returns of the Bombay Stock Exchange indices over the respective Treasury bill rate returns. Chang, di Wang, and le Zhou (2012) proposed a novel model by evolving partially connected neural networks with a GA to predict the stock price trend using technical indicators as inputs and obtaining a very accurate prediction of the stock price index for most of the data. Yaman (2014) applied a neuroevolution algorithm to produce trader agents with a variety of different trading strategies in order to reduce risk, obtaining an overall performance of the system which demonstrated that the cumulative result of the population of agents created is profitable. More recently, Qiu, Song, and Akagi (2016) applied an ANN using GA to calculate the initial weights of the network in order to predict the return of the Japanese Nikkei 255 index for the next month, concluding that the hybrid ANN approach based on a GA improves the prediction accuracy significantly and outperforms the traditional backpropagation training algorithm.

2.2.1. NeuroEvolution of Augmenting Topologies (NEAT)

One of the most popular neuroevolution approaches to evolve both the weights and topology of an ANN is the NeuroEvolution of Augmenting Topologies (NEAT), that was introduced by Stanley and Miikkulainen (2002). NEAT is based on three main techniques: tracking genes with historical markings, applying speciation and complexifying.

NEAT uses a direct encoding scheme (Fig. 1), in which the neural network architecture is directly encoded into the GA genome (Mitchell, 1996), meaning that the genome specifies every connection and node that will appear in the phenotype. NEAT's genetic encoding is designed to allow corresponding genes to be easily lined up when two genomes crossover.

In NEAT the mutation operation can change both the connection weights and the neural network structure. Connection weights mutate similar to any other neuroevolution system, with each connection having a probability of being perturbed at each generation. Regarding structural mutations, there are two main operands in NEAT: add connection mutation and add node mutation. In the add connection mutation a connection, with random weight, is added between two randomly selected neurons that were previously unconnected, while in the add node mutation a random connection is selected, split and replaced by a new node and two new connections.

Genes that share a historical origin necessarily represent the same structure in a neural network phenotype (although possibly with different connection weights). Tracking the historical origins of the genes is possible in NEAT by assigning an unique innovation number to each new gene that is created through structural mutation. In the crossover process, the genes that have the same innovation number in both parents are the matching genes and the ones that don't match with any gene in the other parent are either disjoint (innovation number in the range of the other parent's innovation numbers) or excess genes (innovation number outside the range of the other parent's innovation numbers).

The genomes are selected to crossover in NEAT based on their fitness score, like in other evolutionary algorithms. During the crossover process, the matching genes of the offspring are selected randomly from either parents and the excess and disjoint genes are

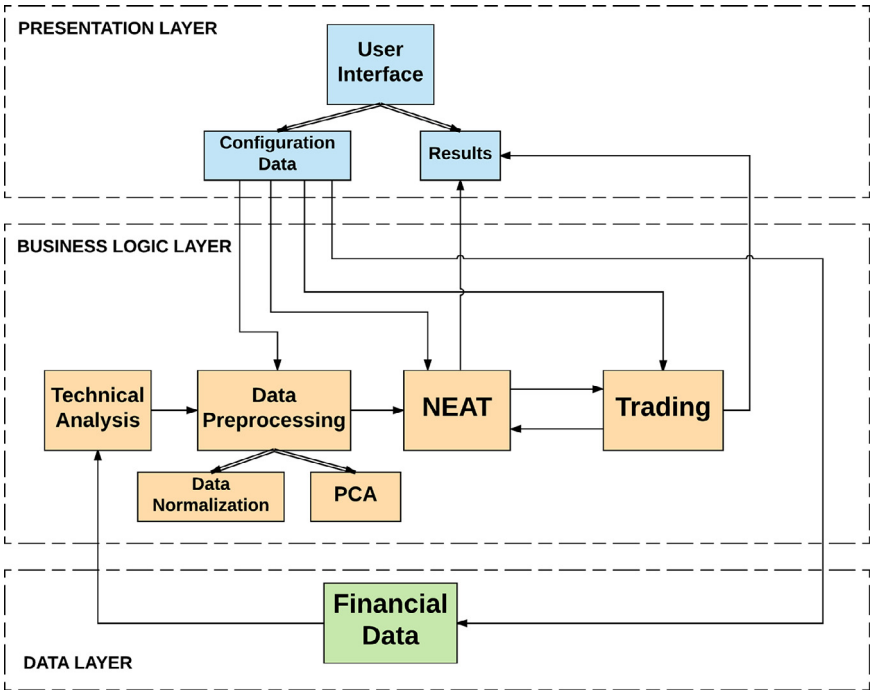

Fig. 2. System architecture.

always selected from the parent with the highest fitness score. The crossover scheme of NEAT helps to mitigate the competing conventions problem as the use of historical markings to align the genomes that perform crossover allows different representations of the same structure to be aligned during crossover and thus, the offspring is likely to be undamaged.

NEAT applies speciation to deal with the topological innovation problem. Speciation is the division of the population into species such that similar topologies are in the same species. This way, the neural networks compete only with neural networks of similar topologies, giving time for the structural innovations to be optimized and, therefore, protecting topological innovations. Genomes are grouped into species by their compatibility distance (δ). The compatibility distance is a simple linear combination of the number excess genes (E), disjoint genes (D) and the average weight differences of matching genes ($\bar{W}$):

$$\delta = \frac{c1 * E}{N} + \frac{c2 * D}{N} + c3 * \bar{W} \quad (1)$$

In Eq. (1), N is the number of genes in the larger genome of the population and the coefficients $c1$, $c2$ and $c3$ allow to adjust the relative importance of the three factors (E, D and $\bar{W}$). After the compatibility distance of each genome is determined, the genomes are placed into the first species (no genome is in more than one species) in which its distance to a randomly chosen member of the species is less than a defined compatibility threshold (δ_t). If a genome is not compatible with any existing species then a new species is created.

In order to maintain topological diversity the reproduction mechanism of NEAT uses fitness sharing, where each genome i of the population is assigned an adjusted fitness f_i', according to its original fitness score f_i and the number of genomes present in its species ($\sum_{j=1}^{n} sh(\delta(i,j))$). The formula for the calculation of the adjusted fitness of each genome i is presented in Eq. (2).

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))} \quad (2)$$

Therefore, the adjusted fitness of each genome is simply its normal fitness divided by the total number of genomes in its species. In NEAT's reproduction system, each species is allowed to spawn a number of offsprings proportional to the sum of the adjusted fitness of its member genomes. Thus, it's difficult for a species to become too big and take over the entire population even if the majority of its genomes perform well, preserving the topological diversity in the population.

Many neuroevolution systems that evolve the topology of the ANN start with a random collection of nodes and connections. This doesn't lead to find minimal solutions since there may exist many unnecessary nodes and connections already present in the initial population. Therefore, NEAT starts with an uniform population of neural networks with input and output nodes (no hidden nodes) and the connections that link the inputs to the outputs, differing only in the initial random weights of the connections. Structural mutations introduce hidden nodes incrementally throughout evolution and thus, NEAT starts with minimal structure allowing the solution to be searched in a low dimensional space which considerably improves performance. This process of searching for the optimal topology by incrementally producing more complex structures constitutes NEAT's complexification process.

3. Proposed approach

3.1. System architecture

This paper presents a trading system that uses the NEAT algorithm combined with the PCA technique to detect the best entry and exit points in the market, with the goal of maximizing the returns and daily profits while taking in consideration the risk and the number of days with capital invested in the market. As illustrated in Fig. 2, the system's architecture is structured on a traditional layered architecture composed by three distinct layers: presentation layer, business logic layer and data layer. The system was developed in Python programming language.

The user starts by inputting the configuration data to the system, where it can be specified: the desired market financial data to be used (input to the financial data module, indicating the file containing the raw financial data used), if PCA is used (input to the

Table 2
List of all 31 variables outputted to the data preprocessing module.

Tecnhical indicators	Financial data
SMA20,SMA50,SMA100	Open
EMA20,EMA50,EMA100	High
Upper, Middle and Lower Bollinger Bands	Low
PSAR	Adj. Close
ATR	Volume
MACD Line, Signal line and MACD Histogram	
PPO	
RSI	
ADX	
CCI	
Momentum	
Stochastic %D,%K	
Williams %R	
ROC	
OBV	
MFI	
Chaikin Oscillator	

data preprocessing module), NEAT parameters (input to the NEAT module, whose default values are presented in Table 4) and trading parameters (input to the trading module).

3.2. Technical analysis

The technical analysis module receives the raw financial data (daily volume and open, high, low and adjusted close prices) and applies several technical indicators, outputting the raw data with the technical indicators to be preprocessed by the data preprocessing module.

In this system, a set of 26 technical indicator features is used in addition to the set of 5 raw financial data features to achieve the set of 31 features (Table 2) that is outputted by this module to the data preprocessing module. These technical indicators were computed with the help of the TA-Lib python library (Fortier, 2007).

3.3. Data preprocessing

Real-world raw data is highly susceptible to noise, missing values and inconsistency. The quality of data affects the data mining results and so, in order to help improve the quality of the data and, consequently, of the results the raw data needs to be preprocessed.

In the system presented by this thesis, the data preprocessing module first starts by dividing the input data provided by the technical analysis module into two distinct sets: the training data set (in-sample data) used to train the model and the testing data set (out-of-sample data) used to test the performance of the model. Then, data normalization and PCA is applied to this data. The Scikit-learn python library (Pedregosa et al., 2011) was used to develop the data preprocessing in this system.

3.3.1. Data normalization

Data normalization is important before applying the PCA since PCA is a variance maximizing exercise. This way, PCA projects the original data onto directions which maximize the variance and if the data features are not measured in the same scale the principal components can be dominated by a single feature with values of bigger magnitude.

In the system implemented, the 31 features of the input data (presented in table 2), provided by the technical analysis module, are all normalized before the PCA method is applied. The data normalizing technique applied to the data is the Min-Max normalization (Eq. (3)), which scales the data to a fixed range [0,1].

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3}$$

Table 3
Ten first principal components of the 31 features obtained from the daily S&P 500 index data (period of 27/03/2006 to 29/01/2015).

Principal component	Variance (%)	Cumulative variance (%)
1	58.4652359	58.4652359
2	22.2981750	80.7634109
3	6.2864476	87.0498585
4	5.0857207	92.1355792
5	2.4078661	94.5434453
6	1.7744661	96.3179114
7	1.1456564	97.4635678
8	0.6629277	98.1264955
9	0.5727300	98.6992255
10	0.3656172	99.0648426

Table 4
Parameters of the implemented system.

Parameter	Value
Sigmoid probability	1/3
TanH probability	1/3
ReLU probability	1/3
Population size	512
c1 (Eq. (1))	1.0
c2 (Eq. (1))	1.0
c3 (Eq. (1))	0.4
δ_t	3.0
δ_t modifier	0.3
Min species	5
Max species	40
Young species fitness boost	1.1
Elitism fraction	0.01
Mutate only probability	0.25
Overall mutation rate	0.2
Node mutation probability	0.03
Connection mutation probability	0.3
Weights mutation probability	0.8
Crossover probability	0.75
Inter-species crossover probability	0.001

3.3.2. PCA

It's the PCA tecnhique, applied in the data preprocessing module, that transforms the high dimensionality input data provided by the technical analysis module to a low dimensionality input data prepared to be fed to the NEAT module.

PCA first fits the model with the normalized training data set, generating the learning model parameters needed for the PCA technique. After the PCA model is fit to the normalized training data set, the 31 principal components are ordered by the amount of variance they explain and the first principal components whose variance captured adds up to at least 95% are preserved. The data is then projected on the preserved principal components and this transformed low dimensional data is fed to the NEAT module.

In practice, only a small percentage of components are significant in terms of variance explained and so, the number of features can be significantly reduced while retaining most of the variance of the data. In Table 3, an example with the ten first principal components, originated from applying the PCA method to the 31 features extracted from S&P500 index data, is presented. In the example of Table 3, it can be observed that the first six principal components explain more than 95% of the data (96.3179114%) and so, in this example, the input features fed to NEAT are reduced (with PCA) from the original 31 features to 6 features.

3.4. NEAT

The NEAT module receives the preprocessed data from the data preprocessing module, applies the NEAT algorithm and outputs the trading signal to the trading module. This module is the core of the system, where the preprocessed input financial data patterns

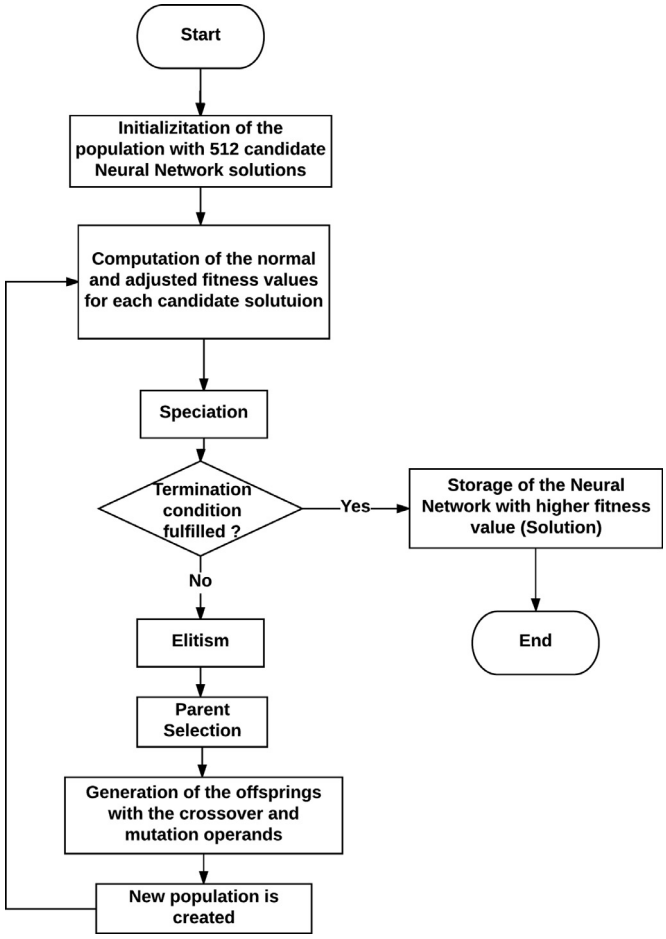

Fig. 3. Flowchart of the NEAT algorithm.

are discovered and an ANN capable of generalizing the information learned from the training data set is built through evolution with the help of an adapted GA. The flowchart of the implemented NEAT algorithm can be observed in Fig. 3 and in Table 4 the values used in this system for some of the most important NEAT parameters are presented. Note that the population size of NEAT was set to 512 individuals through experimental search, taking in consideration the performance and computational cost (the powers of 2 where tested, stopping when the increase in population size didn't correspond to a considerable increase in performance). Python's multiNEAT library (Chervenski & Ryan, 2012) for performing neuroevolution was used as a basis for the implementation of the NEAT algorithm.

3.4.1. ANN

The ANN evolved by NEAT has the inputs provided by the data preprocessing module and a single output value between 0 and 1 that corresponds to the predicted trading signal, indicating if the system should adopt a long or neutral position in the next trading day. Since the output value is in the range [0,1], the activation function defined for the output node is the sigmoid function. The ANN of the NEAT algorithm starts as a feedforward neural network but the algorithm let's the network evolve to a recurrent neural network if that type of network is more adjusted to the problem.

The activation function is the only aspect of the hidden nodes that has to be defined before the start of the NEAT algorithm. The same activation function could be set to all the hidden nodes of the network but that would limit the optimization process due to its single mathematical functionality (Zhang, 2015). Recent work has shown that using different activation functions in the hidden neurons can perform better than using a single activation function (Maul, Bargiela, Chong, & Adamu, 2014; Miikkulainen et al., 2017; Zhang, 2015) and so, in the implemented system the activation function of each of the hidden neuron also evolves with the GA. Therefore, when a new hidden node is added to the network by a structural mutation it has an equal probability of having a Sigmoid, Hyperbolic Tangent (TanH) or ReLU activation function.

3.4.2. Speciation

Regarding the speciation operand of NEAT, the choice of the compatibility threshold is important as it affects the number and constitution of the species, which affects the performance of the NEAT algorithm (Nodine, 2010). In this system, the problem of choosing the best value for the compatibility threshold (δ_t) is avoided by making it dynamic. This way, the compatibility threshold is changed when the system breaks some defined maximum or minimum number of species boundary to make the number of species stabilize between those defined boundaries.

In this system it's also attributed a fitness boost to young species (considered in this system to be species that exist for less than 15 generations) of 1.1 such that each of the species' members has its adjusted fitness value multiplied by 1.1, further mitigating the topological innovations problem and giving the necessary time for new promising structural innovations to evolve.

3.4.3. Fitness function

The definition of the fitness function is crucial to the performance of the system and so, different fitness functions were tested and analyzed. The primary goal of the system is to maximize the returns in the market, but concepts like daily profits, risk and time spent with capital invested in the market are also taken in consideration when devising and testing the different fitness functions. The following fitness functions are proposed:

- A fitness function that measures the profit obtained by day in the market (Equations 4). This way, this fitness function just divides the rate of return (ROR) obtained by the number of days spent in a long position (with capital invested in the market).

$$ROR = \frac{FinalCapital - InitialCapital}{InitialCapital} * 100\,, \tag{4a}$$

$$ROR/day = \frac{ROR}{Number\ of\ Days\ in\ Market}\,. \tag{4b}$$

- A fitness function that measures the Risk Return Ratio (RRR). The set of Equations 5 present the calculations necessary to obtain the RRR, where DD and MDD represent the drawdown and maximum drawdown, respectively. Low values of max drawdown (MDD) are desirable as they represent investments with low losses and consequently, with less risk and a higher risk-return ratio (RRR).

$$DD_t = Max_{i\in(Start,t)}(ROR_i) - ROR_t\,, \tag{5a}$$

$$MDD = Max_{t\in(Start,End)}(DD_t)\,, \tag{5b}$$

$$RRR = \frac{ROR}{MDD}\,. \tag{5c}$$

- A fitness function that measures the mean of the daily profit. The set of Equations 6 present the calculation necessary to obtain the Mean Daily Profit, where TC, DC, MDP and DP represent the transaction costs, daily return, mean daily profit and daily profit, respectively.
Note that the daily profit (Eq. (6b)) measures the daily return obtained in every trading day and so, the daily profit is zero

when the system is out of the market (neutral position, with 0 stocks). On the other hand, when the system is in the market (in a long position) the daily profit is the change (in percentage) in the daily price of the stock, including the transaction costs. The inclusion of the transaction costs simulates that, in every day spent in a long position, the system buys one stock and sells it the next day. This way, the mean daily profit criteria measures if the daily returns obtained in a long position can overcome daily transaction costs and so, even if the ROR of the system is positive this criteria may yield a negative value if the system spends a considerable number of days in a long position where the daily profit isn't even big enough to pay daily transaction costs.

$$DR_t = \frac{Open_t - Open_{t-1} - TC}{Open_{t-1}} * 100 \tag{6a}$$

$$DP_t = \begin{cases} DR_t & \text{if Long} \\ 0 & \text{if Neutral} \end{cases}, \tag{6b}$$

$$MDP = Mean_{t \in (Start,End)}(DP_t). \tag{6c}$$

3.4.4. Mutation

The overall mutation rate is set to 0.2, which means that each new candidate solution generated by the crossover operand has a 20% probability of suffering a mutation. On the other hand, there is a 25% probability that the offsprings are generated from mutation only, without the intervention of the crossover operand.

In the implemented system, only one of type of mutation (weights mutation, add node mutation or add connection mutation) can be applied to each candidate solution in one generation. The roulette wheel selection method is used to choose the type of mutation that is applied to a candidate solution that suffers a mutation, with each type of mutation having a probability of being chosen proportional to its probability value (presented in Table 4).

3.4.5. Hypermutation

The core idea of hypermutation is to adjust the mutation rate based on the problem characteristics and search process state. In this system, the overall mutation rate and the add node mutation (structural mutation) probability are adjusted during the evolution process in order to help the algorithm jump out of a local optimum when the evolution process is beginning to stagnate. This way, the hypermutation technique in this system adjusts the mutation operand of NEAT as it follows:

- If the highest fitness value in the population doesn't increase in 4 generations then the overall mutation rate is increased 0.1.
- If the highest fitness value in the population doesn't increase in 6 generations then both the overall mutation rate and the add node mutation probability are increased 0.05. If it still hasn't increased in 8 generations then the same increases are again applied.
- If the highest fitness value in the population increases then the overall mutation rate and the add node mutation probability are assigned their original values of 0.2 and 0.03, respectively.

3.4.6. Crossover

The crossover operand has a 75% probability of generating the offsprings in this system. Besides the regular crossover operand of NEAT presented in the previous section, this system also allows (with a very small probability) inter-species crossover which can further increase the diversity of the population. In the inter-species crossover all the excess and disjoint genes from both parents are copied to the offspring, instead of being copied only from the most fit parent like in the intra-species crossover. This happens because it is not fair to compare the fitness of topologically different individuals, as already explained. This way, all structural developments from both individuals are present in the offspring of an inter-species crossover.

3.4.7. Termination condition

The termination condition of NEAT can affect the quality and speed of the search and so, it should prevent premature termination and avoid needless computations. In this system one of the following two conditions has to be fulfilled for the NEAT algorithm to terminate: the algorithm reaches 200 generations or there are no improvements in the highest fitness score of the population for 10 generations.

3.5. Trading

The trading module is responsible for simulating the trading experience in a real world market environment, receiving as input the trading signal outputted by the NEAT module and acting in the market accordingly. In Fig. 4 the trading module of the system is exemplified using the trading signal generated by the proposed system in the S&P 500 index for the period of 06/02/2015 to 27/02/2015. The trading module starts with a user defined initial capital for investment and invests that capital according to the received trading signal as it follows:

- A value above 0.5 represents a buy signal. If the system is not holding any stocks the buying signal makes the system to buy, at the opening price of the corresponding trading day, as many stocks as possible with the current capital. On the other hand, if the system is already holding stocks the buying signal tells the systems to keep the stocks and don't sell.
- A value equal or below 0.5 represents a sell signal. If the system is not holding any stocks the sell signal acts as neutral signal and the system just remains without stocks and out of the market. On the other hand, if the system is holding stocks then the sell signal makes the system to sell all his stocks at the opening price of the corresponding trading day.

Note that, when trading futures, leverage isn't considered. This way, it's considered that all the money is invested when buying the future contract (like it happens with a stock) and so the example provided in Fig. 4 is also valid for the trading of futures presented in the results of this paper.

To simulate the real trading environment, in each transaction (buy or sell) it's payed a transaction cost. These values can be charged as a percentage of the total value of the transaction or as an absolute cost per stock, depending on the market and broker used for the transaction.

4. Results

The financial data used to train and test the implemented system are the daily volume and prices (open, high, low and adjusted close) of different markets over the period of 27/03/2006 to 13/04/2017, using 80% of the data to train the system and 20% to test it. To test the robustness of the system to different markets, the experiments are performed in the following markets: S&P 500 index, Brent Crude futures contract, Exxon Mobil Corporation stocks and Home Depot Inc. stocks. The S&P 500 index was chosen as a test market because it is the index with the biggest trading volume, gives a good representation of the United States stock market and sets the trend for the United States economy. On the other hand, Brent Crude futures contract was chosen as a tested market because it has one of the highest volume of trades in the commodities market. Finally, the stocks of the Home Depot Inc. (consumer discretionary sector) and Exxon Mobil corporation (energy

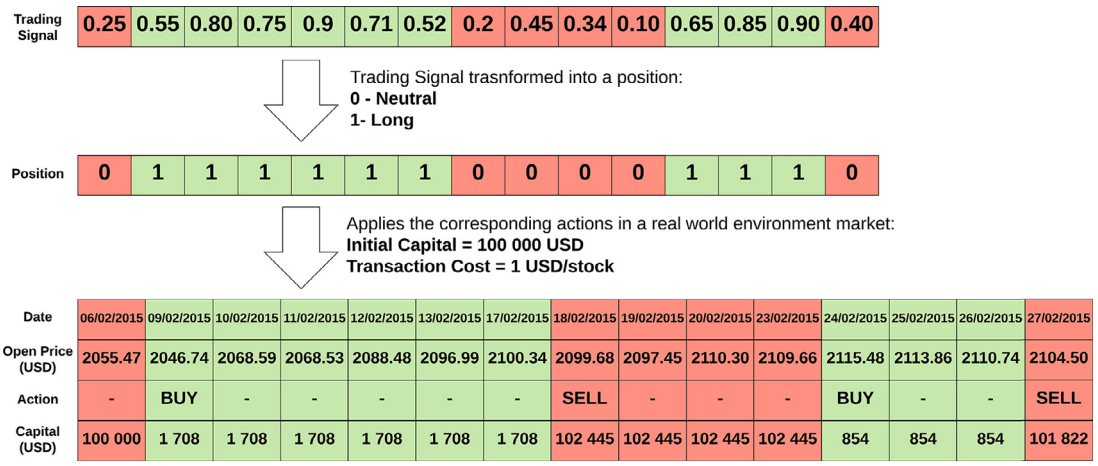

Fig. 4. Example of the trading module execution with S&P 500 data.

sector) were also chosen because of their high trading volume in their respective sector.

The PCA and NEAT models are trained during the train period (first 80% of the data) and the trained model is used to test the performance of the system in the test period (last 20% of the data). During the test period, the PCA and NEAT models remain unchanged (without updating the models with the already processed days of the test period) because this allows the output of the system (action to be undertaken in the next trading period) to be obtained almost instantly after receiving the necessary input data. This way, the described system can be easily adapted to be incorporated in the trading of securities by minute, second or even tick instead of by day (that is the time period used to test the implemented system presented in this paper), which would be impossible to do if the PCA and NEAT models were updated with the past test days as this update doesn't allow the output of the system to be obtained instantly (limiting the real-life use of the system).

The financial data was acquired from Yahoo Finance (S&P 500, Exxon Mobil and Home Depot) and Quandl (Brent Crude) platforms. Note that, due to missing values in the acquired data, not all markets have the same number of train and test trading days.

The transaction costs were applied according to the Interactive Brokers (largest electronic brokerage firm in the United States) low-cost online platform transaction costs: 1 USD per stock transacted in S&P 500 index (considered trading the S&P 500 index with CFDs); 0.1% of the transaction value in Brent Crude futures contract; 0.005 USD per stock transacted in Exxon Mobil and Home Depot stocks.

As the proposed approach uses NEAT, that has a probabilistic and directed random nature, each experiment was repeated twenty times in order to obtain an average of the system's performance and it's this average that is used in the results presented next.

The performance of the experiments is measured not only by the returns (ROR) obtained but also by the risk-return ratio (RRR), mean daily profits (MDP) and returns obtained by day in the market (ROR/day). The number of days with capital invested in the market (Market Days) and maximum drawdown (MDD) of the returns are also taken in consideration.

4.1. Case study I – fitness functions

The first experiment performed was devised to analyze the quality of results obtained by each of the fitness functions proposed in the previous section. The average results obtained by each fitness function on the test data of the financial markets used are presented in Table 5, together with results of the Buy and Hold strategy for comparison.

Table 5
Results of the B&H and the different fitness functions (average).

	B&H	ROR/day	RRR	MDP
Brent				
Transactions	2	69	39	67
ROR (%)	−9.94	45.56	1.95	37.91
Market Days	556	105	418	129
ROR/day (%)	−0.018	0.46	−0.0092	0.36
MDD (%)	64.24	23.22	48.29	24.22
RRR	−0.16	2.12	0.014	1.69
MDP (%)	−0.19	0.048	−0.10	0.029
Exxon Mobil				
Transactions	2	30	25	36
ROR (%)	−5.10	3.53	2.63	4.60
Market Days	555	100	130	249
ROR/day (%)	−0.0092	0.024	0.098	0.063
MDD (%)	28.4	12.00	12.82	19.94
RRR	−0.18	0.30	0.29	0.37
MDP (%)	−0.013	0.0058	0.0038	0.0072
Home Depot				
Transactions	2	42	17	22
ROR (%)	37.65	23.09	35.62	39.97
Market Days	555	179	514	503
ROR/day (%)	0.068	0.12	0.071	0.081
MDD (%)	22.80	11.54	20.25	21.85
RRR	1.65	1.86	1.85	1.85
MDP (%)	0.057	0.037	0.054	0.060
S&P 500				
Transactions	2	24	33	71
ROR (%)	15.71	7.93	11.61	18.89
Market Days	555	56	238	172
ROR/day (%)	0.028	0.094	0.091	0.12
MDD (%)	14.55	6.32	7.77	9.01
RRR	1.08	0.99	1.95	2.11
MDP (%)	−0.065	0.0040	−0.018	0.0080

Analyzing the results obtained, the following observations can be made:

- The RRR fitness function clearly obtains the worse overall results of the three tested fitness functions, only outperforming the Buy and Hold strategy returns in the Brent futures contract and Exxon Mobil stocks.
- The ROR/day fitness function achieves the best results in the Brent futures contract (where it produces exceptional results) but is also the fitness function that performs worse in the S&P 500 index.

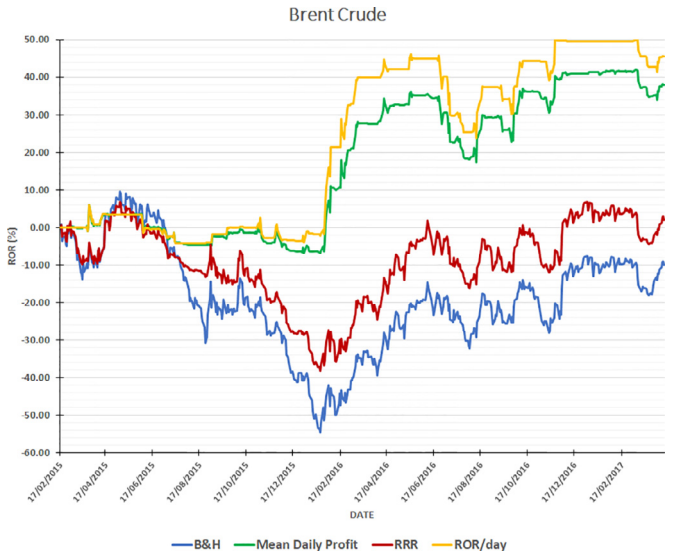

Fig. 5. Returns obtained by the system with the different fitness functions and the Buy and Hold in the Brent Crude futures contract.

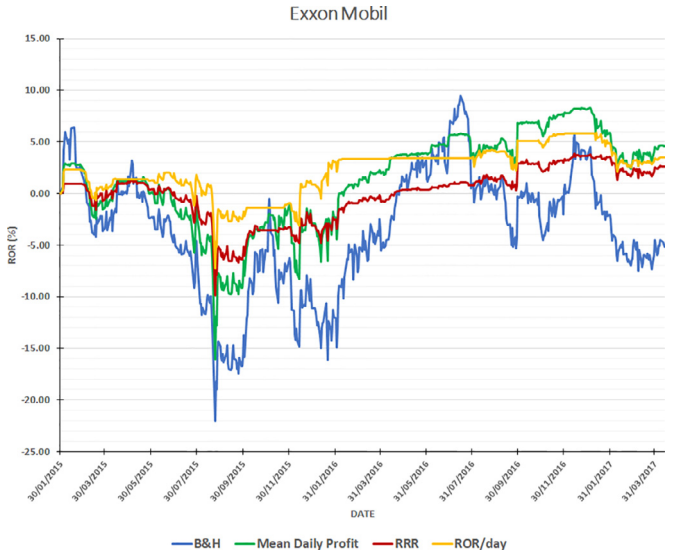

Fig. 6. Returns obtained by the system with the different fitness functions and the Buy and Hold in the Exxon Mobil stocks.

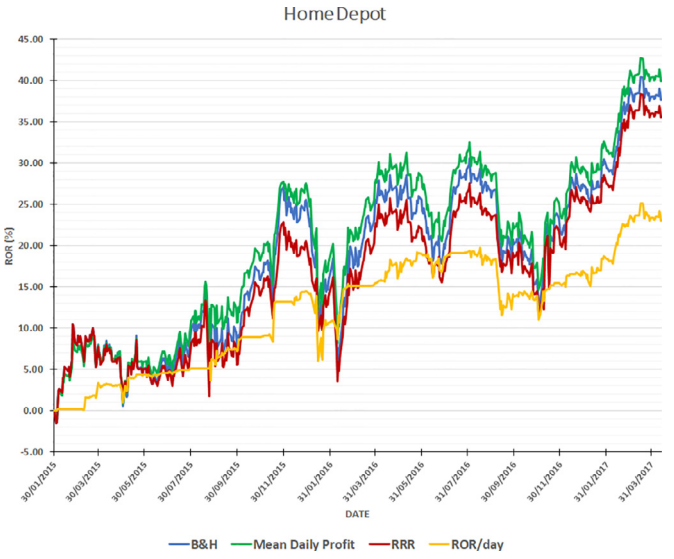

Fig. 7. Returns obtained by the system with the different fitness functions and the Buy and Hold in the Home Depot stocks.

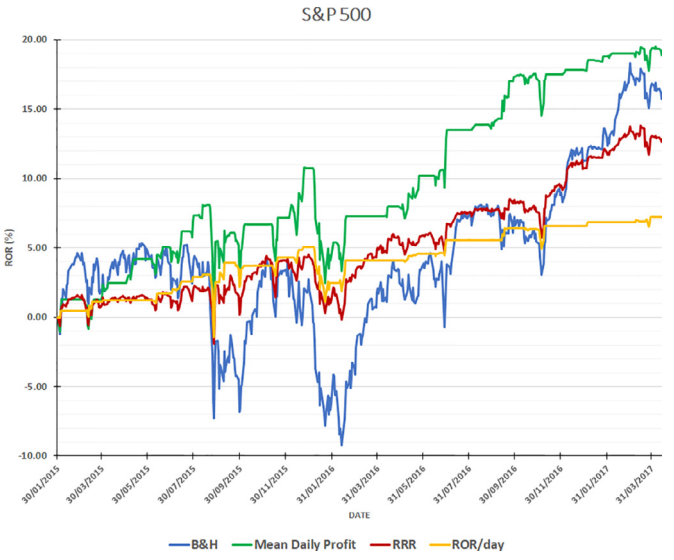

Fig. 8. Returns obtained by the system with the different fitness functions and the Buy and Hold in the S&P 500 index.

- The system with the Mean Daily Profit fitness function can outperform the Buy and Hold strategy in all markets. This fitness function, contrarily to the two other tested fitness functions, is not clearly outperformed by the Buy and Hold strategy in any market, achieving acceptable results even in the markets where it performs the worse.

This way, the Mean Daily Profit fitness function achieves better and more consistent overall results in maximizing the returns and daily profits while minimizing the risk and the number of days spent with capital held up in the market. In particular, of the tested fitness functions, the Mean Daily Profit fitness function is the only one that can outperform the Buy and Hold performance in all of the tested markets.

In Figs. 5–8 it's presented the evolution of the returns obtained by the Buy and Hold strategy and the average returns obtained by the system with the different fitness functions in the different markets during the period of test. Analyzing for example the performance in the S&P 500 index (Fig. 8), it can be observed that the system with both RRR and ROR/day fitness functions can avoid the loss in returns of the Buy and Hold strategy in the first half of the tested period but don't follow closely the big upward trend of the second half of the tested period and get outperformed by the Buy and Hold strategy in terms of returns, despite clearly having a less associated risk (the returns don't oscillate as much as in the Buy and Hold strategy). On the other hand, the system with the Mean Daily Profit fitness function can profit from the sideways periods of the first half of the tested period in the S&P 500 index and is able to follow the upward trend of the second half of the tested period, outperforming the returns of the Buy and Hold strategy (with less associated risk). It should be noted however, that the system could still improve its performance in the S&P 500 index as it's not able to predict the last big increase in prices (from 31/01/2017 to 28/02/2017).

4.2. Case study II

Having determined the fitness function more robust and suitable to the application of this paper, in the case study presented

Table 6
Results of the B&H and the system with and without PCA (average).

	B&H	System without PCA	System with PCA
Brent			
Transactions	2	63	67
ROR (%)	−9.94	6.93	37.91
Market Days	556	147	129
ROR/day (%)	−0.018	0.012	0.36
MDD (%)	64.24	50.44	24.22
RRR	−0.16	0.25	1.69
MDP (%)	−0.19	−0.0029	0.029
Exxon Mobil			
Transactions	2	38	36
ROR (%)	−5.10	2.97	4.60
Market Days	555	398	249
ROR/day (%)	−0.0092	0.017	0.063
MDD (%)	28.4	19.67	19.94
RRR	−0.18	0.19	0.37
MDP (%)	−0.013	0.0042	0.0072
Home Depot			
Transactions	2	22	22
ROR (%)	37.65	28.73	39.97
Market Days	555	355	503
ROR/day (%)	0.068	0.11	0.081
MDD (%)	22.80	17.46	21.85
RRR	1.65	1.63	1.85
MDP (%)	0.057	0.045	0.060
S&P 500			
Transactions	2	29	71
ROR (%)	15.71	5.49	18.89
Market Days	555	48	172
ROR/day (%)	0.028	0.085	0.12
MDD (%)	14.55	4.28	9.01
RRR	1.08	0.76	2.11
MDP (%)	−0.065	0.0042	0.0080

next only that fitness function (Mean Daily Profit) is used. In the second case study of this paper the importance of the PCA and hypermutation methods in the performance of the system is tested and analyzed.

Note that the system was also tested with both PCA and hypermutation methods removed from the system, but the results obtained from that test aren't presented next as they don't add further insight to the results obtained by individually removing the PCA and hypermutation methods.

4.2.1. Case study II.a – influence of the PCA method

In Table 6 the results of the system with and without performing the PCA method are presented with the Buy and Hold results for comparison. Analyzing the results obtained, it can be observed that the importance of the PCA method in the performance of the system is considerable, achieving solutions that produce higher returns in all the markets tested when compared to the system without the PCA method. In particular, without the PCA method the system produces solutions that can only outperform the Buy and Hold in the markets where the Buy and Hold has negative returns in the period tested (Brent Crude futures contract and Exxon Mobil stocks). Furthermore, the PCA method allows the system not only to achieve solutions with higher returns but also solutions that achieve those returns with less risk (higher RRR), higher daily profits (higher MDP) and with a better relation of returns and days spent with the capital invested in the market (higher ROR/day).

In Fig. 9 it's presented the evolution of the returns obtained by the Buy and Hold strategy and the average returns obtained by the system with and without the PCA method in the S&P 500 index during the period of test. It can be observed that, as mentioned above, the system without PCA gets considerably outperformed by the system with PCA.

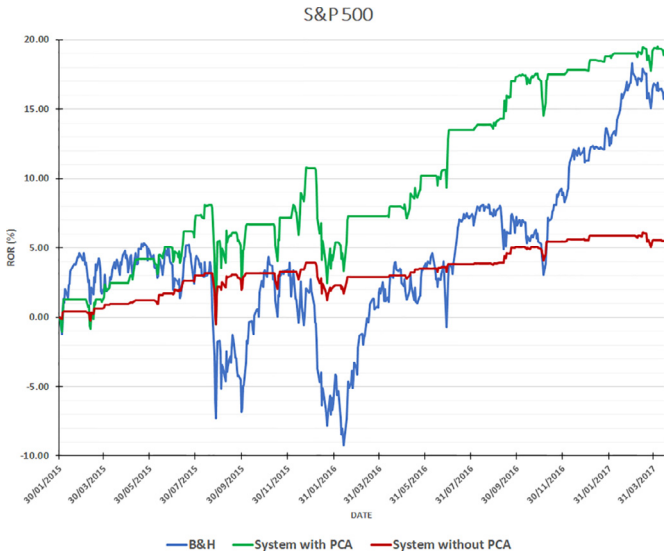

Fig. 9. Returns obtained by the system with and without the PCA method and the Buy and Hold in the S&P 500 index.

Table 7
Results of the B&H and the system with and without hypermutation (average).

	B&H	System without hyper-mutation	System with hyper-mutation
Brent			
Transactions	2	81	67
ROR (%)	−9.94	36.58	37.91
Market Days	556	196	129
ROR/day (%)	−0.018	0.24	0.36
MDD (%)	64.24	30.38	24.22
RRR	−0.16	1.27	1.69
MDP (%)	−0.19	0.0026	0.029
Exxon Mobil			
Transactions	2	38	36
ROR (%)	−5.10	4.35	4.60
Market Days	555	294	249
ROR/day (%)	−0.0092	0.073	0.063
MDD (%)	28.4	20.69	19.94
RRR	−0.18	0.38	0.37
MDP (%)	−0.013	0.0061	0.0072
Home Depot			
Transactions	2	25	22
ROR (%)	37.65	40.25	39.97
Market Days	555	485	503
ROR/day (%)	0.068	0.083	0.081
MDD (%)	22.80	20.83	21.85
RRR	1.65	1.83	1.85
MDP (%)	0.057	0.063	0.060
S&P 500			
Transactions	2	73	71
ROR (%)	15.71	17.45	18.89
Market Days	555	159	172
ROR/day (%)	0.028	0.12	0.12
MDD (%)	14.55	8.89	9.01
RRR	1.08	1.94	2.11
MDP (%)	−0.065	0.0087	0.0080

4.2.2. Case study II.b – influence of the hypermutation technique

In Table 7 the results of the system with and without performing the hypermutation technique are presented with the Buy and Hold results for comparison. Analyzing the results obtained, it can

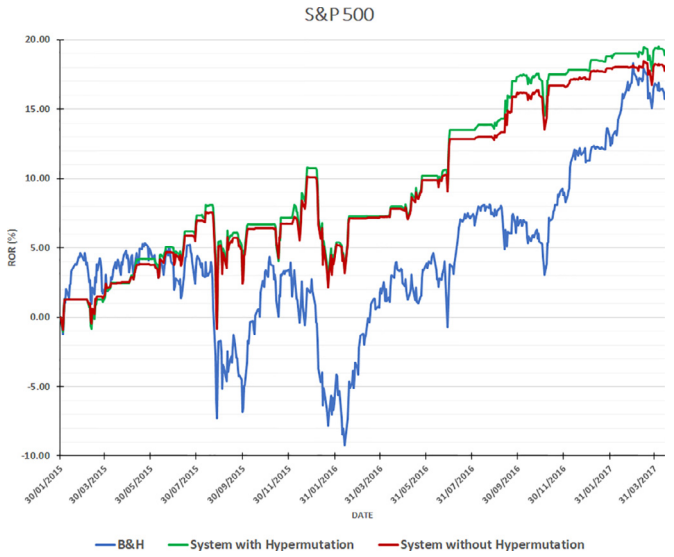

Fig. 10. Returns obtained by the system with and without the hypermutation technique and the Buy and Hold in the S&P 500 index.

be observed that the hypermutation technique doesn't have the crucial importance to the system that the PCA method has, but can still slightly increase the overall performance of the system in the tested markets. Therefore, the increase in the overall mutation rate and add node mutation probability when the system is stagnating still adds value to the system as it can generate solutions with a slightly higher overall performance when investing in the financial markets.

In Fig. 10 it's presented the evolution of the returns obtained by the Buy and Hold strategy and the average returns obtained by the system with and without the hypermutation technique in the S&P 500 index during the tested period. It can be observed that the return lines are very similar in the system with and without hypermutation indicating that they produce similar investment decisions but, as mentioned above, the system with hypermutation slightly outperforms the system without hypermutation.

4.3. Case study III – Sectors of the S&P 500 index

To further evaluate the robustness of the proposed system, the best performing version of the system (with the hypermutation and PCA methods and with the mean daily profit fitness function in the NEAT algorithm) was tested with one representative stock (chosen by trading volume) of each of the eight sectors with the most significant market cap present in the S&P 500 index (Table 8 and 9). Besides the Exxon Mobil (energy) and Home Depot (consumer discretionary) stocks, already presented in the previous case study, the system was also tested with the Apple Inc. (information technology), Berkshire Hathaway Inc. (financial), Monsanto Company (materials), Pfizer Inc. (health care), Procter & Gamble Company (consumer staples), and Union Pacific Corporation (industrial) stocks.

Analyzing the results obtained, the following observations can be made:

- The implemented trading system achieves a positive return in all the tested stocks, even in the three company stocks in which the Buy and Hold strategy outputs negative returns. Furthermore, the implemented trading system can achieve a higher return than the Buy and Hold in six of the eight company stocks tested.
- When investing in Apple stocks, the Buy and Hold achieves a slightly superior return, daily profits and return by day spent

Table 8
Results of the B&H and the proposed system in stocks representative of the different sectors present in the S&P 500 index – Part I.

	B&H	This system
Apple		
Transactions	2	31
ROR (%)	19.83	19.73
Market Days	555	426
ROR/day (%)	0.036	0.033
MDD (%)	37.52	31.28
RRR	0.53	0.62
MDP (%)	0.040	0.033
Berkshire		
Transactions	2	56
ROR (%)	13.10	6.52
Market Days	555	133
ROR/day (%)	0.024	0.21
MDD (%)	17.74	7.76
RRR	0.74	1.48
MDP (%)	−0.17	−0.020
Exxon Mobil		
Transactions	2	36
ROR (%)	−5.10	4.60
Market Days	555	249
ROR/day (%)	−0.0092	0.063
MDD (%)	28.4	19.94
RRR	−0.18	0.37
MDP (%)	−0.013	0.0072
Home Depot		
Transactions	2	22
ROR (%)	37.65	39.97
Market Days	555	503
ROR/day (%)	0.068	0.081
MDD (%)	22.80	21.85
RRR	1.65	1.85
MDP (%)	0.057	0.060

Table 9
Results of the B&H and the proposed system in stocks representative of the different sectors present in the S&P 500 index – Part II.

	B&H	This system
Monsanto		
Transactions	2	10
ROR (%)	−1.72	6.54
Market Days	555	15
ROR/day (%)	−0.0031	1.31
MDD (%)	35.87	6.42
RRR	−0.048	1.32
MDP (%)	−0.19	0.0083
Pfizer		
Transactions	2	39
ROR (%)	6.76	11.04
Market Days	555	112
ROR/day (%)	0.012	0.18
MDD (%)	24.64	10.80
RRR	0.27	1.22
MDP (%)	−0.18	−0.014
Procter & Gamble		
Transactions	2	62
ROR (%)	5.62	14.83
Market Days	555	273
ROR/day (%)	0.010	0.010
MDD (%)	21.74	12.93
RRR	0.26	1.18
MDP (%)	−0.18	−0.037
Union Pacific		
Transactions	2	35
ROR (%)	−11.67	2.91
Market Days	555	87
ROR/day (%)	−0.021	0.021
MDD (%)	45.58	10.45
RRR	−0.26	0.27
MDP (%)	−0.21	−0.018

in the market, but the implemented trading system can achieve a slightly superior risk-return ratio. Thus, the performance of implemented trading system and of the Buy and Hold strategy in the Apple stocks is similar.
- Only when investing in the Berkshire Hathaway stocks (of the financial sector) can the Buy and Hold achieve a considerably higher return than the implemented trading system but, even there, the implemented trading system still achieves a considerably superior risk-return ratio, return by day spent in the market and daily profits.
- The implemented trading system achieves a higher risk-return ratio than the Buy and Hold in all of the eight tested stocks.
- With the exception of the Apple stocks, this trading system achieves a higher mean of the daily profits than the Buy and Hold in all the tested stocks.
- With the exception of the Apple and Procter & Gamble stocks, the implemented trading achieves a higher return by day than the Buy and Hold in all the tested stocks.

5. Conclusions

This work proposes an approach combining the PCA method for dimensionality reduction with the NEAT algorithm for evolving an ANN with an adapted GA to generate a trading signal capable of achieving high returns and daily profit with low associated risk. This approach, with a fitness function in the NEAT algorithm measuring the mean daily profits, shows robust results over a wide variety of financial markets. One of the major conclusions that can be drawn from the results obtained is that the dimensionality reduction provided by the PCA is crucial to the good performance of the system. The hypermutation technique, despite not having the importance of the PCA, also adds value to the system slightly increasing the performance.

To further evaluate the robustness of the implemented system, eight stocks representative of the eight sectors with the highest market cap present in the S&P 500 index were tested and, with the results obtained, it can be concluded that the implemented system is robust as it can outperform the Buy and Hold in the majority of the tested stocks.

Some ideas for future work are: add the short investment position to the system to open up an opportunity to increase the returns by profiting from bear periods; test other fitness functions involving other financial concepts (like the sharpe ratio); test other dimensionality reduction techniques with NEAT or adapt the GA of NEAT to work also as a feature selector; join the backpropagtion method to the NEAT implementation so that the backpropagation method can further optimize the weights of the ANN while NEAT focuses more on optimizing the topology.

Acknowledgement

This work was supported in part by Fundação para a Ciência e a Tecnologia (Project UID/EEA/50008/2013).

References

Chang, P.-C., & Liu, C.-H. (2008). A TSK type fuzzy rule based system for stock price prediction. *Expert Systems with Applications, 34*(1), 135–144.
Chang, P.-C., di Wang, D., & le Zhou, C. (2012). A novel model by evolving partially connected neural network for stock price trend forecasting. *Expert Systems with Applications, 39*(1), 611–620.
Chang, P.-C., Wang, Y.-W., & Tsai, C.-Y. (2005). Evolving neural network for printed circuit board sales forecasting. *Expert Systems with Applications, 29*(1), 83–92.
Chang, Y.-T., Lin, J., Shieh, J.-S., & Abbod, M. F. (2012). Optimization the initial weights of artificial neural networks via genetic algorithm applied to hip bone fracture prediction. *Advances in Fuzzy Systems, 2012*, 1–9.
Chervenski, P., & Ryan, S. (2012). MultiNEAT : Portable software library for performing neuroevolution. http://multineat.com/index.html. [Online; accessed 20-March-2017].
Chiang, W.-C., Enke, D., Wu, T., & Wang, R. (2016). An adaptive stock index trading decision support system. *Expert Systems with Applications, 59*, 195–207.
Fama, E. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance, 25*(2), 383–417.
Fortier, M. (2007). TA-Lib : Technical analysis library. http://www.ta-lib.org/. [Online; accessed 20-April-2017].
Gorgulho, A., Neves, R., & Horta, N. (2011). Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition. *Expert Systems with Applications, 38*(11), 14072–14085.
Göçken, M., Özçalıcı, M., Boru, A., & Dosdoğru, A. T. (2016). Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Systems with Applications, 44*, 320–331.
Inthachot, M., Boonjing, V., & Intakosum, S. (2016). Artificial neural network and genetic algorithm hybrid intelligence for predicting thai stock price index trend,. 2016, 1–8.
Jadav, K., & Panchal, M. (2012). Optimizing weights of artificial neural networks using genetic algorithms. *International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE), 1*(10), 47–51.
Jolliffe, I. T. (2002). *Principal component analysis* (2nd ed.). Springer. ISBN:978-0387954424.
Khan, Z. H., Alin, T. S., & Hussain, M. A. (2011). Price prediction of share market using artificial neural network (ann). *International Journal of Computer Applications, 22*(2), 0975–8887.
Kirkpatrick, C. D., & Dahlquist, J. A. (2015). *Technical analysis: The complete resource for financial market technicians* (3rd ed.). FT Press. ISBN:978-0134137049.
Kuo, R., & Chen, J. (2004). A decision support system for order selection in electronic commerce based on fuzzy neural network supported by real-coded genetic algorithm. *Expert Systems with Applications, 26*(2), 141–154.
Mańdziuk, J., & Jaruszewicz, M. (2009). "Dead" chromosomes and their elimination in the neuro-genetic stock index prediction system. In *Proceedings of the 16th international conference on neural information processing: Part II*. In *ICONIP '09* (pp. 601–610). Springer-Verlag.
Maul, T. H., Bargiela, A., Chong, S.-Y., & Adamu, A. S. (2014). Towards evolutionary deep neural networks. *28th European conference on modelling and simulation*. ECMS.
Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., et al. (2017). Evolving deep neural networks. *CoRR*.
Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press. ISBN:978-0262133166.
Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. New York Institute of Finance. ISBN:978-0735200661
Nodine, T. (2010). Speciation in NEAT. *Undergraduate Honors Thesis, HR-10-06*. Department of Computer Science, The University of Texas at Austin.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12*, 2825–2830.
Perwej, Y., & Perwej, A. (2012). Prediction of the bombay stock exchange (BSE) market returns using artificial neural network and genetic algorithm. *Journal of Intelligent Learning Systems and Applications, 4*(2), 108–119.
Qiu, M., & Song, Y. (2016). Predicting the direction of stock market index movement using an optimized artificial neural network model. *PloS ONE, 11*(5).
Qiu, M., Song, Y., & Akagi, F. (2016). Application of artificial neural network for the prediction of stock market returns: The case of the japanese stock market. *Chaos, Solitons & Fractals, 85*, 1–7.
Sandström, C., Herman, P., & Ekeberg, Ö. (2015). *An evolutionary approach to time series forecasting with artificial neural networks*.
Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation, 10*(2), 99–127.
Wang, J., & Wang, J. (2015). Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks. *Neurocomputing, 156*(C), 68–78.
Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing, 14*(3), 347–361.
Yaman, A. (2014). Neuroevolution and an application of an agent based model for financial market. Master's thesis. CUNY City College, USA.
Zahedi, J., & Rounaghi, M. M. (2015). Application of artificial neural network models and principal component analysis method in predicting stock prices on tehran stock exchange. *Physica A: Statistical Mechanics and its Applications, 438*, 178–187.
Zhang, L. M. (2015). Genetic deep neural networks using different activation functions for financial data mining. In *International conference on big data* (pp. 2849–2851). IEEE.
Zhonga, X., & Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications, 67*(C), 126–139.