Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Sciences 170 (2005) 3–33
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

A hybrid genetic-neural architecture for stock indexes forecasting

G. Armano *, M. Marchesi, A. Murru

DIEE, University of Cagliari, Piazza d'Armi, I-09123, Cagliari, Italy

Received 10 March 2001; accepted 18 March 2003

Abstract

In this paper, a new approach for time series forecasting is presented. The forecasting activity results from the interaction of a population of experts, each integrating genetic and neural technologies. An expert of this kind embodies a genetic classifier designed to control the activation of a feedforward artificial neural network for performing a locally scoped forecasting activity. Genetic and neural components are supplied with different information: The former deal with inputs encoding information retrieved from technical analysis, whereas the latter process other relevant inputs, in particular past stock prices. To investigate the performance of the proposed approach in response to real data, a stock market forecasting system has been implemented and tested on two stock market indexes, allowing for account realistic trading commissions. The results pointed to the good forecasting capability of the approach, which repeatedly outperformed the "Buy and Hold" strategy.

Keywords: Stock market forecasting; Time series prediction; Genetic algorithms; eXtended classifier systems; Artificial neural networks

1. Introduction

It is widely acknowledged that financial time series modeling and forecasting is an arduous task. These time series behave very much like a random walk

* Corresponding author.
E-mail address: armano@Diee.unica.it (G. Armano).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2003.03.023

process and several studies have concluded that their serial correlation is economically and statistically insignificant [11]. The same studies seem to confirm the efficient market hypothesis (EMH) [8], which maintains that the current market price of a stock fully reflects—at any time—the available information assimilated by traders. As new information enters the system, the imbalance is immediately detected and promptly redressed by a counteracting change in market price. Depending on the type of information examined, three forms of EMH exist: weak, semi-strong, and strong. We are particularly concerned with the weak EMH, which only takes into account past stock price data. In this case, the underlying assumption is that no predictions can be made based on stock price data alone, as they follow a random walk in which successive changes have zero correlation. This hypothesis implies that future changes in stock market prices cannot be predicted from information about past prices. Notwithstanding these difficulties, most stock market investors seem convinced that they can statistically predict price trends and make a profit. This is done by exploiting technical or fundamental analysis rules, as well as "momentum strategies" (i.e., buying when the market is bullish and selling when it is bearish). For these reasons, many attempts have been made to model and forecast financial markets, using all the computational tools available for studying time series and complex systems: linear auto-regressive models, principal component analysis, artificial neural networks (ANNs), genetic algorithms (GAs), and others (for an interesting review see [12]). In this paper, we present a hybrid approach to stock market forecasting that integrates both GAs [13] and ANNs [29] and cooperatively exploits them to forecast the next-day price of stock market indexes. In particular, we use an extended classifier system (XCS) [36], which relies on technical-analysis indicators (see, for example, [1]) to determine the current market status, in conjunction with feedforward ANNs explicitly designed for financial time series prediction. To our knowledge, no previous work has been done on hybrid systems that integrate XCSs with feedforward ANNs. We have called the proposed approach NXCS, standing for neural XCS, and customized it for financial time series prediction. A forecasting system based on an NXCS module has been tested on financial time series showing the trend of some major stock market indexes on a fairly large observational window. In particular, about 9 years of data of the COMIT [1] and S&P500 stock market indexes have been used to train and test the system, being very careful to avoid any form of data snooping. In both cases, the first 1000 data were used to train and tune the NXCS module, and the resulting system was tested on the subsequent 1000 data, leaving its overall configuration unchanged. We compared the system's forecasting capabilities with the "Buy and Hold" (B&H) strategy,

[1] Comit is the acronym for "Banca Commerciale Italiana", a major Italian bank.

considering realistic transaction costs. Further comparisons with a system based on neural networks technology have also been performed. The results are encouraging, demonstrating the validity of the approach. The remainder of the paper is organized as follows: in Section 2 a short introduction to the state-of-the-art in financial time series forecasting with artificial intelligence (AI) techniques is given; in Section 3 the novel approach is described, first from a general perspective and then with all the customizations designed to handle the problem of financial time series forecasting; in Section 4, after briefly outlining the overall architecture of the proposed system, experimental results are discussed; and lastly the conclusions are drawn in Section 5.

2. AI techniques for financial time series forecasting

In recent years, advances in both analytical and computational methods have led to a number of interesting new approaches to financial time series forecasting, based on non-linear and non-stationary models. In the following, we focus the review of previous work on GAs and ANNs applied to stock market prediction, as our proposal is based on the integration of such techniques. In addition, as the resulting framework is also an implementation of the general concepts known as mixture of experts, the related work on this topic will be briefly described.

2.1. GAs for financial time series forecasting

GAs are a family of computational models inspired by natural evolution [10,13]. In a broader usage of the term, a genetic algorithm is any population-based model that uses selection and recombination operators to generate new sample points in a search space. An implementation of a GA deals with a population of "chromosomes", each representing a potential solution of a target problem, usually in form of binary strings. In particular, for classification tasks, a chromosome can be represented by a condition–action pair, where the condition is a string of values in {0,1,#} and the action is a label that denotes a class. Thus, each chromosome is specialized on a subset of the input space; i.e., some inputs activate the chromosome and other inputs exclude it from the decision process. Usually, these chromosomes are randomly created, and undergo reproductive opportunities in such a way that better solutions are given more chances to reproduce than poorer ones. Although GAs have been adopted in a multitude of different tasks, in this paper we focus on proposals that address only the problem of financial time series forecasting. Noever and Baskaran [26] investigated the task of predicting trends and prices in financial time series, making experiments on the S&P500 stock market. Mahfoud and Mani [23] addressed the general problem of predicting future performances of

individual stocks. Their work is particularly relevant, as they make a comparison between GAs and ANNs applied to financial forecasting. According to their experiments—repeated on several stock markets—both approaches outperform the B&H strategy. A combined approach, obtained by averaging out GAs and ANNs outputs, is also experimented with positive results. GAs have also been used in a variety of hybrid approaches to financial time series prediction. For example, Muhammad and King [25] devised evolutionary fuzzy networks to forecast the foreign exchange market, whereas Kai and Wenhua [16] exploited GAs to train ANNs for predicting a stock price index.

2.2. ANNs for financial time series forecasting

ANNs appear to be particularly suited for financial time series forecasting, as they can learn highly non-linear models, have effective learning algorithms, can handle noisy data, and can use inputs of different kinds (see [28] for a survey). Furthermore, complex non-linear models based on exponential GARCH processes [2] show similar results (in terms of out-of-sample prediction performance) to those obtained by much simpler ANNs based on multilayer perceptron (MLP) architectures [3]. A major weakness of MLPs, from a time-series forecasting perspective, is the absence of an internal state, making it difficult to capture the dynamics of the given data series. Even if standard MLPs can still be used [5], due to their simplicity and flexibility, a variety of more complex architectures with some form of internal memory has been proposed (see [4] for a survey). In particular, Recurrent ANNs (RANNs) have proved capable of outperforming stateless architectures in financial time series forecasting [9]. Nevertheless, it is well known [22] that out-of-sample results have a strong dependence on the time period used to train the network. In other words, ANNs trained on data belonging to a specific period perform reasonably well only if the test period is relatively similar to the one used for training. Furthermore, the similarity between two periods of time is guaranteed by sharing some kind of economic condition (e.g., bullish or bearish period), instead of being caused by temporal contiguity. This problem is related to the presence of the so-called regime-shifting phenomenon, which occurs when the underlying process can be thought of as being multistationary. In this case, several periods of stationarity (i.e., regimes) hold, separated by usually rapid transitions. Under this hypothesis, obtaining a single model that holds for different regimes can be extremely difficult.

2.3. Multiple experts for financial time series forecasting

As previously pointed out, GAs deal with a population of chromosomes, each specialized on a subset of the input space. Hence, nothing prevents us from considering them as an implementation of the multiple experts concept,

which received a great deal of attention—though in forms substantially different from the GA proposal—from both the connectionist and the time series communities. As we believe that a multiple experts centered perspective could be useful for dealing with the problem of regime-shifting, below we also briefly recall some related work in this specific area. Note that, in this case, the problem of how to combine experts outputs in order to obtain a response from the overall system becomes a crucial aspect. In the time series community, Weigend et al.—starting from Jacobs and Jordan's mixtures of experts [14]—devised non-linear gated experts [33,34], applying them to several time series domains, including financial ones. The key elements characterizing these experts are the non-linearity of local experts and gates, the experts' capability of separately adapting themselves to noisy data, and a soft-partitioning mechanism (i.e., softmax [24]) for blending outputs. Moreover, Shi and Weigend [31] proposed a statistical method for experts selection based on Hidden Markov models.

3. A hybrid approach for dealing with stock market forecasting

In this section, the hybrid approach previously summarized is described with more detail, from both a conceptual and a technical perspective. As for conceptual issues, a novel kind of model identification is introduced, originated by the need of dealing with multistationary processes. In addition, the idea of partitioning the input space starting from suitable technical-analysis domain knowledge is illustrated and framed in a multiple-experts perspective. To this end, the underlying framework is briefly introduced, to give the reader the ability to perceive the general characteristics of the proposed approach. As for technical issues, some basic notions about the particular kind of GAs (i.e., XCSs) adopted for evolving a population of hybrid experts is given. Finally, the particular kind of ANN that has been designed and adopted is described, with all customizations devised to deal with the task of financial time series forecasting.

3.1. Context-based identification of multistationary models

The basic idea that lies behind the proposed approach is simple: rather than trying to identify a global model, an attempt to identify different local models is performed, according to the hypothesis that financial time series are multistationary. Let us point out in advance that, no matter whether the task is prediction or classification, our approach to model identification differs very much from the ones based on a state description — e.g., Hidden Markov models [27] and RANNs. To stress the difference with existing approaches and to give a flavor of the underlying assumptions, we decided to call

"context-based" our approach to model identification. In fact, it applies the divide-and-conquer strategy by first identifying the current context and then using one or more local experts acknowledged as able to deal with it. Here, different sources of information are entrusted with different tasks, i.e., performing the identification of the current context vs. performing the intended classification or prediction. As a particular case of context-based model identification, let us consider a population of experts, each embedding two components: a context selector and a classifier/predictor. The selector is aimed at controlling the activation of the associated classifier/predictor, depending on the current input. In the next session, we briefly outline the corresponding framework, named guarded experts.

3.2. The guarded experts framework

Let us assume that an input and an output space exist (i.e., I and O, respectively), and that an oracle Λ holds that, for each $x \in I$, maps x to a value in O. Given an input vector, a globally scoped expert $\widetilde{H}$ is a total function that approximates Λ on the whole input space, whereas a guarded expert $\widetilde{\Gamma}$ is a partial function that approximates Λ on a proper subset of the input space. In its simplest form, $\widetilde{\Gamma}$ can be represented by a pair $\langle g, \tilde{h} \rangle$, where g is a context-selector (i.e., a guard) devoted to accept or reject any given input x as belonging to the domain of the function $\tilde{h}$. Without loss of generality, let us separate the inputs used for performing context selection from the ones used for making a prediction or classification. Thus, $I \equiv I_g I_h$, where $g : I_g \rightarrow B$ maps a non-empty subset of input features to $B = \{\text{false}, \text{true}\}$, and $\tilde{h} : I_h \rightarrow O$ approximates the oracle on the subset of inputs for which the property g holds. Applying a guarded expert $\widetilde{\Gamma} = \langle g, \tilde{h} \rangle$ to an input $x \equiv x_g x_h$ returns a value that is described by the following, semi-formal, semantics:

$$\widetilde{\Gamma}(x) = \text{if } g(x_g) \text{ then } \tilde{h}(x_h) \text{ else } \perp \tag{1}$$

namely, given a guarded expert $\widetilde{\Gamma} = \langle g, \tilde{h} \rangle$, the classifier/predictor $\tilde{h}$ is activated by an input $x \equiv x_g x_h$ only if $g(x_g) = \text{true}$ (i.e., if the input x matches the guard g). The subset of inputs covered by a guarded expert $\widetilde{\Gamma} = \langle g, \tilde{h} \rangle$ strictly depends on its guard g, according to the definition:

$$\text{Covers}(\widetilde{\Gamma}) = \left\{ x_g x_h \in I | g(x_g) = \text{true} \right\} \tag{2}$$

Let us now briefly depict the characteristics of a population Ω of guarded experts, starting from the following definition:

$$\Omega = \left\{ \widetilde{\Gamma}_i | \widetilde{\Gamma}_i = \langle g_i, \tilde{h}_i \rangle, \ i = 0, 1, \ldots, n \right\} \tag{3}$$

The set of inputs covered by Ω can be defined on top of Eq. (2) as follows:

$$\text{Covers}(\Omega) = \bigcup_{\widetilde{\Gamma}_i \in \Omega} \text{Covers}(\widetilde{\Gamma}_i) \tag{4}$$

Furthermore, given an input $x \equiv x_g x_h$ and a population of guarded experts Ω, the function $\text{Select}(\Omega|x)$ is responsible for the *match-set* formation; in other words, it returns the set of guarded experts whose guards are matched by x. In symbols:

$$\text{Select}(\Omega|x) = \text{Select}(\Omega|x_g x_h) = \left\{ \Gamma = \langle g, \tilde{h} \rangle \in \Omega | g(x_g) = \text{true} \right\} \tag{5}$$

We are interested in investigating a particular class of populations of experts, able to cover the whole input space. We say that a population Ω of guarded experts covers the whole input space I when the following constraint holds:

$$\text{Covers}(\Omega) \equiv I \tag{6}$$

This property allows us to guarantee that a population of guarded experts can substitute a globally scoped expert on the whole input space. In the following, we assume that such a property holds; i.e. that a non-empty match set can be formed for any input $x \in I$:

$$\forall x \in I : |\text{Select}(\Omega|x)| > 0 \tag{7}$$

Note that the definitions given by Eqs. (6) and (7) do not rule out the case of multiple experts being selected for dealing with a given input x. Without going into unnecessary details, let us assume that a voting policy or an outputs-blending mechanism (for classification and prediction tasks, respectively) should be supplied for evaluating the overall response of the experts enabled by the current input. As for classification tasks, the most widely acknowledged voting policy is the so-called *majority rule*, which consists of adopting the response that reaches the highest score among the selected experts. As for prediction tasks, the most widely acknowledged outputs-blending mechanism is the *average rule*, which averages out the outputs of the selected experts. A "weighted" version of both the majority and average rule can be enforced by taking into account each single expert according to a different degree of reliability (e.g., its degree of adaptation to the given environment).

3.3. Neural XCS

Bearing in mind that here the term GAs is used in a broad sense, denoting systems where a population of classifiers evolve according to Darwinian selection, we decided to implement the guarded experts framework by hybridizing Wilson's XCSs with feedforward ANNs explicitly designed for financial time series prediction. The resulting approach has been called NXCS,

standing for Neural XCS. As for XCSs (briefly outlined in Appendix A), they have been devised by Wilson based on Holland's Learning Classifier Systems [13]. Since then, further related work has provided a deep theoretical understanding of the XCS approach [17–19,37], and many extensions have been proposed (see, for example, [20,21,38,39]). For the sake of simplicity, in the following we restrict our attention to the prediction task only, being interested in investigating how NXCS experts can be customized for financial time series forecasting. To relate NXCS with the general framework of guarded experts, let us point out that XCS classifiers play the role of guards, whereas ANNs are used for implementing predictors. In particular, given an input vector $x \in I$, let us separate some of its binary (x_b) features from the rest of the input vector (x_r), yielding $x \equiv x_b x_r$. Thus, the guard of an NXCS expert $\widetilde{\Gamma} = \langle g, \tilde{h} \rangle$ maps any binary input to {false, true}, whereas its predictor $\tilde{h}$ is an ANN trained and activated only on a subset of the input space (according to the selection performed by the guard g). Furthermore, let us assume that an NXCS expert can have multiple outputs. It is worth pointing out that an NXCS expert can be described from both an evolutionary and a neural perspective. According to the former interpretation, it can be considered an extension of an XCS classifier, whose action part has been replaced with a suitable ANN. According to the latter interpretation, it can be considered an extension of a neural predictor, equipped with an additional guard that acts as a selector, thus preventing the neural predictor from working outside the context defined by the guard itself. In any case, NXCSs are able to perform a synergistic integration between two sources of information (i.e., binary and numerical), adopting a suitable technology for each one. Fig. 1 shows the structure of a generic NXCS expert, pointing to its guard and the corresponding predictor (the capability of the genetic guard to enable the neural predictor is also put into evidence).

3.4. Handling a population of NXCS experts

An NXCS is an evolutionary system where a population of NXCS experts, each characterized by a genetic guard and the corresponding neural predictor, is raised in a typical XCS-like environment. Each NXCS expert has a set of

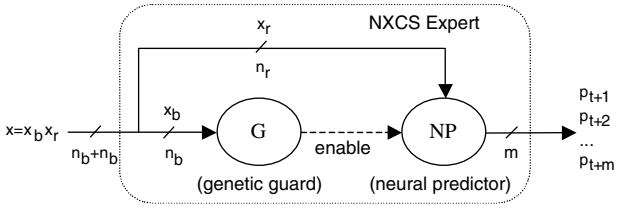

Fig. 1. The structure of an NXCS expert, which embeds a genetic guard and a neural predictor.

associated parameters, to be used and updated while evaluating, step-by-step, the evolution of the population. In particular, for each NXCS expert $\widetilde{\Gamma}$, let us recall the following parameters:

- the fitness (f), i.e., the degree of adaptation of $\widetilde{\Gamma}$ to the environmental constraints;
- the prediction or strength (p), i.e., the expected reward to the system for using $\widetilde{\Gamma}$.

Given a population of NXCS experts, its dynamics is basically conformant to the one that characterizes XCS classifier systems. Of course, some differences exist, due to the fact that the action part of an XCS classifier has been replaced with an ANN. The remainder of this subsection is devoted to illustrate the major differences that hold between an XCS and an NXCS. In particular, the mechanisms for (i) generating and maintaining experts, as well as for (ii) performing experts' selection and outputs blending will be briefly described.

3.4.1. Generating and maintaining NXCS experts

Given an NXCS expert, its genetic guard supports covering, crossover, and mutation operations, whereas its neural predictor is trained once, immediately after being created, and left unaltered while the system is running. The fitness f of an NXCS expert is updated according to the default XCS policy. On the other hand, the prediction p is dealt with in a non-standard way: Having to cope with prediction tasks, we let p coincide with the primary ANN output. [2]

Furthermore, when a new NXCS expert has to be inserted into the population, its predictor is generated with initial random weights and then trained using a set of examples obtained by selecting the inputs that match its guard on a fixed-length window of past prices. In this way, each predictor is trained on a different set of examples, according to the current time and to the corresponding guard.

3.4.2. NXCS mechanisms for experts selection and outputs blending

In an XCS classifier system, experts' selection and outputs combination occur as follows (see also Appendix A). Given a message, first the match set (M) is created gathering all XCS classifiers whose condition is matched by the current input, then the adopted voting policy is applied. The default voting policy is the so-called fitness-weighted majority rule. Being $M_i \subseteq M$ the set of

[2] The concept of primary output stems from the consideration that ANNs may have multiple outputs, in which case an output is arbitrarily selected as being primary. For time series forecasting, this primary output coincides with the next predicted value of the time series.

XCS classifiers that support the action i, the reward the system can achieve for choosing it (i.e., P_i) is estimated according to the formula:

$$P_i = \frac{\sum_{c \in \mathbf{M}_i} p_c \cdot f_c}{\sum_{c \in \mathbf{M}_i} f_c} \tag{8}$$

where p_c and f_c are the prediction and the fitness of each XCS classifier $c \in M_i$. Note that each supporting classifier contributes to strengthen the hypothesis on the corresponding action according to its current fitness. The winning action (i^*) is the one that maximizes the estimated reward. In an NXCS used for prediction tasks, we decided to leave unaltered the selection mechanism and to define an outputs-blending mechanism suited to match the characteristics of the given task. In particular, to forecast the next value of a time series, a fitness-weighted average rule is enforced by blending the outputs of the experts that belong to the match set according to their fitness. In this way, the more reliable is an expert the more its ANN output contributes to the overall prediction. In particular, given a match set M, the overall primary output, say $p(t+1)$, is evaluated according to the formula:

$$p(t+1) = \frac{\sum_{c \in M} p_c \cdot f_c}{\sum_{c \in M} f_c} \tag{9}$$

where p_c and f_c are the prediction—which coincides with the primary ANN output—and the fitness of each NXCS expert that belongs to the match set M.

3.5. Customizing NXCS experts for stock market forecasting

According to the general choice made for NXCSs, guards and the neural predictors have been supplied with different information, to better exploit their capabilities of dealing with binary and real inputs, respectively. As technical-analysis indicators are commonly used by financial traders to predict transitions between different market regimes, we assumed they could be sensible inputs for the partitioning system, whereas the forecasting of future prices clearly requires knowledge about past prices. In agreement with these informative concepts, NXCS experts have been tailored to deal with stock market forecasting as follows: (i) some technical-analysis domain knowledge has been embodied into the guard of each expert; (ii) a novel kind of feedforward ANN, able to deal with short-term dynamics on a weekly base (i.e., 5-days), has been defined and adopted to perform the prediction task.

3.5.1. Embodying technical-analysis domain knowledge into NXCS guards

As a kind of relevant domain knowledge, some technical analysis *indicators* have been considered. They are reported in Table 1, under the assumption that

Table 1
Definitions of technical-analysis indicators

Indicator	Definition
Difference of averages	$\mathrm{DOA}_{N1,N2}(t) = \mathrm{MA}_{N1}[q](t) - \mathrm{MA}_{N2}[q](t)$
Rate of change	$\mathrm{ROC}_N(t) = \frac{q(t)-q(t-N)}{q(t-N)}$
Relative strength index	$\mathrm{RSI}_N(t) = \left(1 + \frac{\mathrm{MA}^+_N[\mathrm{neg}](t)}{\mathrm{MA}^+_N[\mathrm{pos}](t)}\right)^{-1}$ where pos and neg are defined as follows: $\mathrm{pos}(t) = \max\left(0, \frac{q(t)-q(t-1)}{q(t)}\right)$ $\mathrm{neg}(t) = \max\left(0, \frac{q(t-1)-q(t)}{q(t)}\right)$ MA^+ differs from MA in the fact that only positive values in the pos and neg sequences are taken into account
Convexity	$\mathrm{CX}_N(t) = \sum_{i=1}^{N} q(t-N+i) - r(t-N+i)$ where r is the line connecting $q(t-N)$ and $q(t)$
Up trend	$\mathrm{UP}_N = 1$ if local minima in $[t-N,t]$ form an increasing sequence; 0 otherwise. Being $q_k = q(t-k)$, a local minimum occurs if $q_k = \min(q_{k-1}, q_k, q_{k+1})$ for $0 < k < N$. Note that $q(t)$ and $q(t-N)$ are considered minima
Down trend	$\mathrm{DW}_N = 1$ if local maxima in $[t-N,t]$ form a decreasing sequence; 0 otherwise. Being $q_k = q(t-k)$, a local maximum occurs if $q_k = \max(q_{k-1}, q_k, q_{k+1})$ for $0 < k < N$. Note that $q(t)$ and $q(t-N)$ are considered maxima

$q(t)$ is the stock price at day t, and that MA (i.e., Mobile Average) is a generic "low-pass" filtering operator defined as:

$$\mathrm{MA}_N[x](t) = \frac{1}{N} \cdot \sum_{i=0}^{N-1} x(t-i) \tag{10}$$

The input of a NXCS guard is a vector of eight binary features, i.e., $x_b = (b_1, b_2, \ldots, b_8)$. As shown in Table 2, the meaning of each input feature directly stems from a technical-analysis indicator. In particular, in its basic form, a feature has been obtained by adding a relational constraint (e.g., b_3, b_4). More complex features have been obtained composing basic features (e.g., b_1, b_2).

Accordingly, an NXCS guard consists of a vector of eight values in {0,1,#}. It can either take into account or disregard a given input feature by suitably setting the corresponding matching condition. In particular, a matching condition set to either "0" or "1" enables a feature, whereas a matching condition set to "#" (= *don't care*) allows to disregard it. The overall matching between an input vector and an NXCS guard occurs when all "non-#" values coincide. Each resulting feature has a very simple interpretation, together with a default action to be undertaken in absence of other relevant information. As an example, let us consider the feature b_3, obtained by adding the relational

Table 2
Binary inputs, to be matched by the guard of an NXCS expert

x_b	Binary input	Note
b_1	$\mathrm{DOA}_{1,30}(t) > 0$ and $\mathrm{DOA}_{1,30}(t-1) < 0$	We just crossed a valley
b_2	$\mathrm{DOA}_{1,30}(t) < 0$ and $\mathrm{DOA}_{1,30}(t-1) > 0$	We just surmounted a peak
b_3	$\mathrm{RSI}_{15}(t) < 0.3$	Too many sales vs. purchases
b_4	$\mathrm{RSI}_{15}(t) > 0.7$	Too many purchases vs. sales
b_5	$\mathrm{UP}_5(t) = 1$	Bullish period
b_6	$\mathrm{DW}_5(t) = 1$	Bearish period
b_7	$\mathrm{UP}_5(t-1) = 1$ and $\mathrm{UP}_5(t) \neq 1$	Current bullish period has just finished
b_8	$\mathrm{DW}_5(t-1) = 1$ and $\mathrm{DW}_5(t) \neq 1$	Current bearish period has just finished

constraint "<0.3" to the indicator "relative strength index", i.e., $\mathrm{RSI}_N(t)$. Such an indicator basically accounts for the number of purchases vs. the number of sales in a given observational window of length N. Hence, the relational expression $\mathrm{RSI}_{15}(t) < 0.3$ tries to capture the informal semantics "heavy price falls". In this case, the associated default action (i.e., the action to be undertaken in absence of further information) would be "take a long position". Thus, the feature b_3 could be the precondition of a technical-analysis rule that—although controversial—can be summarized as "when there is a heavy price fall, then take a long position". Unfortunately, due to the fact that an NXCS guard usually takes into account several features together, it is not common the case that all involved features suggest taking the same position. In fact, it is far more common the case of contradictory default suggestions about the action to be undertaken. That is why, in the proposed system, no particular emphasis is given on extracting technical-analysis rules embedded within genetic guards.

Note that indicators used to prepare inputs for the genetic guards are evaluated on windows that range from 2 to 6 weeks (i.e., usually 10–30 days of actual trading), depending on the selected indicator. The reason why there is no common observational window depends only loosely on the experiments performed while tuning the system. In fact, the decision about which window has to be used for a given indicator results from a priori, thus subjective, appraisals made on the underlying dynamics of each indicator. Notwithstanding this arbitrary degree of freedom, such a choice should not be considered as a drawback, since "carving up" indicators with the aim of optimizing the behavior of the system might introduce an unwanted correlation between the system's parameters and the time series used for tuning it. In addition, given the great number of indicators that have been exploited, an attempt to find the best window for each indicator would end up to a long tuning activity.

3.5.2. Devising a feedforward ANN for stock market forecasting

Two different requirements influenced the design and implementation of the feedforward ANN used to perform predictions on a local basis; i.e., limiting the overfitting phenomenon and the influence of noise. Overfitting training data while performing poorly on test data is one of the major problems of learning algorithms, in particular when noisy data are involved. This problem is particularly relevant for economic time series, which are very noisy and require complex learning algorithms. Overfitting is strongly related to non-stationarity: data may appear to be noisy when using inadequate models. On the other hand, partitioning the input space reduces the number of learning examples for each model, thus increasing the risk of overfitting each smaller training set. Two approaches are commonly adopted to reduce overfitting while using ANNs: stopping the learning process when the network has attained good performance on a separate validation set, and adopting some form of weight pruning (e.g., [6,15]) as a means of reducing the network's ability to learn the least significant patterns. Furthermore, it is common practice to pre-process data for reducing the effect of outliers—often a result of exogenous shocks—and to compress the dynamic range, for instance using logarithms as in [9]. One of the major problems that arises when using a single-output MLP to forecast time series is that the only information used by the backpropagation (BP) algorithm is the derivative of the cost function—usually the squared error between desired and actual output. With noisy data, the small amount of information contained in a single output is typically inadequate to distinguish non-linearities from noise. A possible solution could be to predict more than one output, to provide more information aimed at characterizing the output dynamics. However, as pointed out by Weigend and Zimmermann [35], this approach only results in a slight performance increase over single-output networks. [3] For this reason, the authors propose a multilayer architecture explicitly designed to utilize all available information and to minimize the risk of overfitting. The network forecasts dynamic variables, such as derivatives and curvatures of prices on different time spans, instead of directly forecasting future prices. These variables are subsequently combined in an interaction layer (with constant weights), which outputs several estimates that, properly averaged, give the final prediction.

In this paper, we agree with the conjecture of Weigend and Zimmermann, aimed at increasing the information flow by predicting multiple interacting outputs. However, the approach adopted here is much simpler, as it does not rely on numerous layers or on explicit forecast of derivatives and curvatures of the price. The feedforward ANN defined and used by NXCS to tackle the

[3] In fact, the BP algorithm on a standard MLP keeps the information relative to each point separate, so that the derivatives of the cost function relatively to each output are independent, and outputs can interact only through the hidden units.

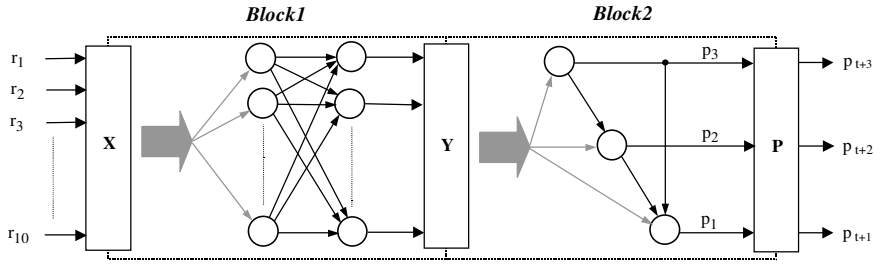

Fig. 2. A feedforward ANN for stock market forecasting, with 10 numerical inputs and 3 outputs.

problem of stock market forecasting is shown in Fig. 2. The first block (*Block1*) of this architecture is a standard MLP, whereas the second block (*Block2*), whose structure is similar to the one adopted on cascade correlation architectures [7], has been designed to enforce the desired outputs interaction. Note that the resulting ANN is still a feedforward network, so that backpropagation can be used. The input layer of *Block1* is denoted by X, whereas the input layer of *Block2* (which corresponds to the outputs of *Block1*) is denoted by Y. The overall ANN is used to predict the prices for the next three days, [4] represented by the output vector $P = (p_{t+1}, p_{t+2}, p_{t+3})$. The primary output is associated with the prediction of the next-day price of the stock index (i.e., p_{t+1}). Note that p_{t+2} is an input for p_{t+1}, and that p_{t+3} is an input for both p_{t+1} and p_{t+2}; hence, the primary output takes as input every other forecasted price. In this way, derivatives of the standard Mean Square Error cost function are not independent with respect to each output, thus helping to reduce the influence of noise—according to the cited Weigend and Zimmermann's conjecture.

The input $X \equiv x_r = (r_1, r_2, \ldots, r_{10})$ is a vector of 10 numerical features (see Table 3). In particular, the input of a single ANN is represented by the value of 5 technical-analysis indicators, together with the prices of the last 5 days, suitably filtered to reduce the effect of outliers. The following equation has been used for pre-processing prices:

$$dl_N(t) = \operatorname{sign}[q(t) - q(t-N)] \cdot \ln\left(\left|\frac{q(t) - q(t-N)}{q(t-N)}\right| + 1\right) \tag{11}$$

where $q(t)$ and $q(t-N)$ are the prices at day t and $t-N$, respectively, whereas $dl_N(t)$ is the corresponding input to the ANN. The equation used for pre-processing is similar to the one used by Giles et al. [9], and to the one proposed in [32], the most notable difference being that we set $N = 5$ instead of $N = 1$.

[4] Without loss of generality, the percent variation of the current price is actually predicted, with a range in [−1,1].

Table 3
Inputs to the ANN

x_r	Numerical input	Note
r_1	$\frac{\mathrm{DOA}_{1,30}(t)}{\mathrm{MA}_{30}[q](t)}$	Difference of averages (on 30 days, and normalized with MA_{30})
r_2	$\mathrm{MA}_3[\mathrm{ROC}_{15}](t)$	Three weeks rate of change (averaged with MA_3)
r_3	$\mathrm{CX}_{10}(t)$	Two weeks convexity
r_4	$\mathrm{RSI}_{15}(t)$	Three weeks relative strength index
r_5	$\mathrm{MA}_{10}[\sigma_{30}](t)$	30 days standard deviation of the relative price variations (averaged with MA_{10})
r_6	$dl_5(t)$	Last prices ($k = 0, 1, \ldots, m$), pre-processed by Eq. (11), to reduce the effects of outliers ($m = N = 5$).
r_7	$dl_5(t-1)$	
r_8	$dl_5(t-2)$	
r_9	$dl_5(t-3)$	
r_{10}	$dl_5(t-4)$	

Note that setting $N = 5$ enforces a non-linear transformation of the current week's prices, which allows to embody within *dl* inputs also information about the previous week. The quantity $\sigma_{30}(t)$ represents the standard deviation of the relative price variations, computed for the last 30 days.

4. Experimental results

Assessing the performance of a forecasting system is not an easy task for a number of reasons. First, as data are non-stationary, the significance of the results obtained in a test period is not easy to quantify. Furthermore, variables traditionally handled by learning algorithms, such as mean square error or percent of correct classifications, do not have a direct economic relevance. Finally, stock markets have a number of constraints and costs that cannot be overlooked. To test our approach, we implemented a system that makes it easier to interpret the results from an economically biased perspective. To let the reader better understand the characteristics of the system that has been devised for making experiments, a preliminary description of its architecture is given. Afterwards, experimental results are discussed. As already pointed out, tests have been performed on two relevant stock market indexes, also taking into account transition costs. The results have been compared with the ones obtained by simply adopting the B&H strategy. A comparison with results obtained by experimenting a system based on RANNs is also briefly reported. Finally, the statistical significance of experimental results is discussed, along with the overall characteristics of the NXCS-based approach (in particular, its ability of dealing with bullish, bearish, and quiet periods).

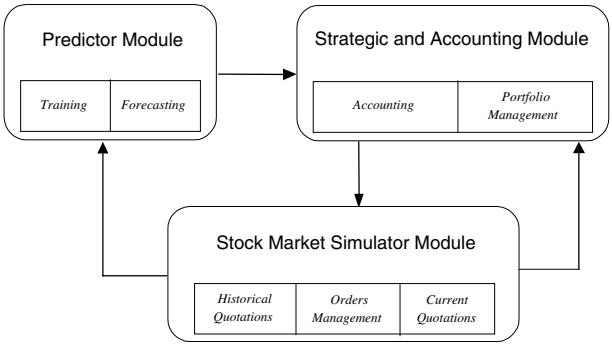

Fig. 3. Overall system architecture.

4.1. System architecture

The overall system is shown in Fig. 3. It is composed of three interacting subsystems: (i) an NXCS module, handling a population of NXCS experts, (ii) a stock market simulator, and (iii) a strategic and accounting module. Let us now concentrate on the second and third component of the system, having thoroughly discussed the characteristics of the NXCS module in the previous sections.

4.1.1. The stock market simulator

The stock market simulator is a simple virtual market, maintaining the daily prices of the stock index under investigation. We considered an index in place of a single stock to avoid (or average out) the impact of company-specific news on the prediction of only one stock, given that the prediction is performed by taking into account past prices only. Furthermore, a global index can be considered as a good benchmark for assessing the EMH, although some studies claim its validity also for individual stock prices (e.g., [3]). In the simulated market set up for experimenting the proposed methodology, a virtual trader can buy or sell derivatives on the stock index concerned, and both short and long positions can be taken over the index. So that our virtual investor can gain from both a fall and rise of the stock index price, we defined two different kinds of derivative assets—to be exploited alternatively, depending on the decision taken by the strategic and accounting module. The first should be used when a long position has to be taken, as it produces exactly the same returns as the stock index price. The second should be used when a short position has to be taken, as it produces the returns of the stock index price, with inverted sign. [5]

[5] In a real market we can buy or sell futures on the index to obtain a similar result. For the sake of simplicity, in this paper we do not consider the peculiarities of such advanced financial instruments.

These derivative assets will be denoted appending a "+" or "−" to the name of the stock index (e.g., S&P500+ and S&P500−). The stock market simulator is also responsible for applying transaction costs that have a fixed part (10 €) and a variable part of 0.1% on the value of the stock option bought or sold, representing the spread between the value of the index and the actual buy or sell price. Switching from a short to a long position and vice versa is a two-step operation, so that—as in real markets—both fixed and variable commissions are doubled. We do not explicitly considered taxes, since tax systems heavily depend on the specific country. Nevertheless, let us note that in most countries the net gain might be lower than the computed value, due to the presence of taxes. Furthermore, only one daily operation is allowed in this market, and stock prices do not change as a consequence of this operation. Of course, this is a strong restriction; nevertheless, it appears to be compatible with the amount of money (i.e., 100,000 €) invested by the system. In other words, we deem that the perturbation due to the operations managed by the system is irrelevant with respect to the overall trading activity that actually took place in the real market where the time series data come from.

4.1.2. The strategic and accounting module

Here, the strategic and accounting module plays a marginal role, due to the presence of a single stock in the market. Presently, its main responsibility is to turn any prediction made by the NXCS module into an actual trading activity. To illustrate the simple mechanism that has been implemented, let us first recall that the overall prediction of the NXCS module ranges in [−1,1]. That is why, the decision of taking a long or short position is derived by simply thresholding the NXCS-module prediction with zero (i.e., a positive value is associated with a long position and vice versa). The strategic and accounting module is able to enforce two different trading strategies, called "defensive" and "aggressive". In the defensive strategy, the system either has all its assets in cash or invested in the index. If the predictor suggests taking a long position, the system invests all its capital in the stock index (e.g., S&P500+). If it has already invested in the index, it keeps its position. On the other hand, if the strategic module suggests taking a short position, the system sells all its stock. If it has already only cash assets, it keeps its position. This strategy cannot obtain returns during market falls, but for the days the system stays out of the market, an interest rate of 4% (yearly) is guaranteed to the trader. The aggressive strategy behaves like the defensive one for taking a long position. To take a short position, the system invests all its capital in the derivative asset that guarantees returns with inverted sign (e.g., S&P500−). In this way, the system never has cash assets (except at the very beginning of the simulation), and can obtain higher returns by correctly forecasting falls in the stock price. When the system's assets switch from the index to the derivative, or vice versa, the percent commission is computed twice. The strategic and accounting module is also responsible for

not to change position on the stock when the reliability of the NXCS predictor, evaluated on recent predictions, is too low. In this way, transaction costs are saved. When the predictor's reliability exceeds a given threshold, trading operations are restarted.

4.2. Tests

The system described in the previous sections has been tested over a long time span. Both training and testing have been performed on a personal computer equipped with a 600 MHz processor and 128 MB of RAM. We used 2000 daily prices for the COMIT and for the S&P500 index. [6] In both cases, the first 1000 prices were used to find the best configuration for the NXCS. This configuration includes: (i) the parameters derived from XCS (e.g., population size limit, learning rate, GA frequency), and (ii) the parameters that characterize the ANN architecture (e.g., number and position of neurons, learning rate, momentum). We performed several tuning sessions, assessing different ANN architectures and XCS parameters. To evaluate the performance of each configuration, we used economic results obtained in the first 800 trading days as training set, and the last 200 as validation set. On the average, a population of about 1200 experts has been handled during the training, validation and testing phase. For any selected configuration, the time required to train and validate the system was about four hours, whereas the time spent for testing it was about two hours. It is worth pointing out that, during a simulation, the evolutionary mechanism continuously generates new experts and updates their fitness –although experts' generation occurs mostly during the training phase, since a stable population of experts tends to take place as the simulation goes on. This is the reason why the time spent for testing is considerably lower than the time spent for training. On the other hand, the time spent for making a next-day prediction is negligible (i.e., 5–15 s), thus allowing the system to be experimented on real trading activities. Each simulation started with 100,000 € cash (the absolute value of the investment is relevant, due to the presence of a fixed part in transaction costs) and has been assessed using a *merit factor* (ω), defined for each trading strategy s, according to the formula:

$$\omega(s) = \frac{CV_s}{CV_{\mathrm{B\&H}}} \cdot \frac{\sigma_{\mathrm{B\&H}}}{\sigma_s} \tag{12}$$

where CV is the cumulative value obtained by enforcing a given trading strategy (i.e., defensive, aggressive, and B&H), whereas σ is the standard deviation of the daily returns series, representing the risk. According to the

[6] From May 21, 1992 to April 20, 2000 for the COMIT index. From June 2, 1993 to May 3, 2001 for the S&P500 index.

Table 4
Major parameters' values used for the NXCS subsystem

	Parameter	COMIT	S&P500
XCS	Maximal population size	1000	1000
	Learning rate	0.0165	0.0083
	Accuracy fall-off (α)	0.0007	0.08
	Accuracy threshold (ε_0)	0.93	0.60
ANN	Input layer	10	10
	Hidden layer 1	8	8
	Hidden layer 2	3	3
	Output layer	3	3
	Learning rate	0.001	0.001
	Momentum	0.1	0.1

given definition, ω measures a cumulative value normalized with the one obtained by adopting the B&H strategy (i.e., $CV_s/CV_{B\&H}$), in a risk-adjusted perspective (i.e., $\sigma_{B\&H}/\sigma_s$).

Given a stock index, the best configuration is the one that maximized ω for the selected defensive or aggressive strategy on the validation set. The most significant system parameters adopted for predicting the COMIT and S&P500 stock indexes are reported in Table 4. Once the best configuration was found, we tested the system on the subsequent 1000 prices. To improve the reliability of the results, we performed several tests for different values of randomly generated variables (e.g., ANN initial weights, mutated and randomly generated bits in binary rules). As expected, only small variations have been recorded throughout the runs; nevertheless, results have been averaged to improve the statistical significance of the results.

Table 5 shows the economic results obtained on the COMIT and the S&P500 indexes using different trading strategies (i.e., defensive and aggressive), against

Table 5
Comparing economic results for the COMIT and S&P500 stock market indexes

Stock index	TS	CV	AR%	CP	IC	σ	S	ω
COMIT	B&H	290,974	30.95%	110	0	1.404	0.0831	1.0000
	Def.ve	355,720	37.68%	33,596	7853	1.114	0.1193	1.5383
	Agg.ve	381,999	39.85%	72,257	0	1.400	0.1019	1.3022
S&P500	B&H	148,077	10.42%	110	0	1.302	0.0367	1.0000
	Def.ve	199,293	19.02%	28,350	2616	1.215	0.0628	1.4418
	Agg.ve	255,022	26.66%	59,786	0	1.297	0.0787	1.7284

TS = trading strategy, CV = cumulative value (in euro), AR% = percent of annualized returns, CP = commission paid (in euro), IC = interest collected (in euro), σ = standard deviation, S = sharpe ratio, and ω = merit factor.

the B&H strategy. Note that the last two columns of the table (which report the Sharpe ratio [30] and the merit factor) give information about the total return in a risk-adjusted perspective. Let us recall that the Sharpe ratio (S) is a direct measure of reward-to-risk, commonly defined as

$$S(x) = \frac{r_x - r_b}{\sigma_x} \tag{13}$$

where x is some investment, r_x is the average rate of return of x, r_b is the best available rate of return of a benchmark portfolio or security, [7] and σ_x is the standard deviation of r_x. As reported, both defensive and aggressive strategies have an advantage over the B&H strategy, even though they pay considerable commissions. Note that the aggressive strategy shows a slight advantage in terms of risk, evaluated by the standard deviation of relative price variations over the entire length of the simulation. Resulting Sharpe ratio and merit factor show, in a risk-adjusted perspective, the superiority of both strategies over the B&H and the relative advantage of the defensive strategy. Despite these impressive results, Figs. 4 and 5 show that, for both indexes, the system is able to take advantage of bearish periods, whereas in bullish or quiet periods it is not able to obtain a significant gain. In our opinion, this is mainly due to the fact that it is very difficult to perform better than the B&H as long as stock prices keep rising up. It is still reasonable that the system performances are low also during quiet periods, although—at least in principle—a gain could be obtained even in presence of small index variations. In addition, it is worth noting that the advantage for the COMIT index over the S&P500 index originates mainly from the 1998 stock market crash (about Autumn 1998, as put into evidence by the plot of Fig. 4), while in bullish or quiet times, returns and risks are similar.

Table 6 shows the predictor performance in terms of percentage of correct classifications (for both the COMIT and the S&P500 indexes), compared with the B&H strategy and with results obtained by adopting a locally recurrent ANN (LRANN) [4] equipped with a hidden layer of 20 neurons. [8] The inputs given to the network are the same used for the NXCS approach, whereas the characteristics of the hidden-layer neurons result from a long tuning activity in which several configurations have been experimented.

[7] The Sharpe ratio is evaluated on the daily returns time series. The benchmark rate of return has been set to 4% (annualized).

[8] The paper of Campolucci illustrates in great detail the characteristics of such a kind of neurons. For the sake of brevity, let us only recall that a hidden-layer neuron is locally recurrent as only its output is used to feedback the neuron. Furthermore, its inputs are processed by autoregressive mobile average (ARMA) filters, able to take into account the past history of inputs within a given window length, to be considered as a parameter of the hidden-layer neuron.

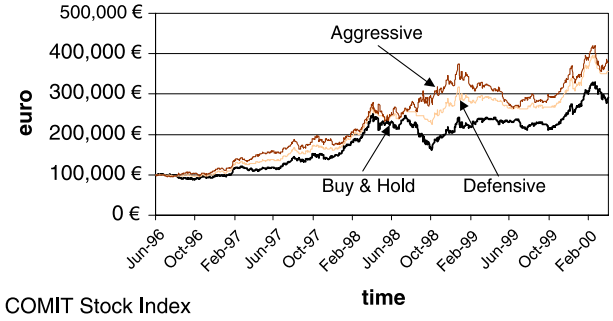

Fig. 4. Economic results on the COMIT stock market index (For color see online version).

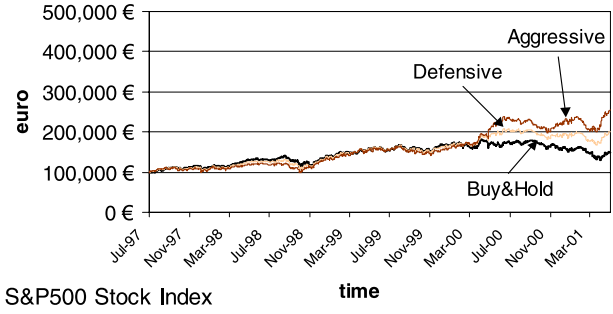

Fig. 5. Economic results on the S&P500 stock market index (For color see online version).

Our results point out that the system equipped with an NXCS module performs better that the one equipped with an LRANN module. As for the B&H strategy, it was observed to be only slightly superior, in terms of correct classifications, [9] to the NXCS. To investigate this aspect, in apparent contrast with the above economic results (see Table 5), we assumed that the difference could derive from less significant variations, and we computed new results using just the days with a percent index variation >1%. As conjectured, disregarding these minimal variations, both defensive and aggressive NXCS strategies perform better than the B&H one. We also defined a further indicator, called ideal profit ($ip\%$), as

$$ip\% = 100 \cdot \frac{\sum \operatorname{sign}\left[\mathrm{d}(t) \cdot \hat{\mathrm{d}}(t)\right] \cdot |\mathrm{d}(t)|}{\sum |\mathrm{d}(t)|} \tag{14}$$

[9] Let us point out that our concept of "correct classification" is economically biased, in the sense that the proposed position for a given day is considered correct whenever it produces a positive return (i.e., it accounts also for transition costs).

Table 6
Comparing predictive capabilities of LRANNs- and NXCSs-based systems

Stock index	PT	CC	CC > 1%	*ip*%	*L*/*S*
COMIT	B&H	55.500%	59.845%	11.204%	–
	LRANN	53.600%	55.959%	12.240%	432/568
	NXCS	55.167%	61.140%	17.121%	729/271
S&P500	B&H	54.700%	54.447%	4.904%	–
	LRANN	52.350%	56.019%	11.370%	889/111
	NXCS	55.500%	59.030%	14.126%	883/117

PT = prediction technique, CC = percent of correct classifications, *ip*% = ideal profit, L/S = long vs. short signal statistics).

where $\mathrm{d}(t)$ represents a real index percent variation and $\hat{\mathrm{d}}(t)$ the estimation. This indicator measures the performance of the overall predictor focusing on the opportunities to make a gain that have been unattended by the system, and vice versa. As reported in Table 6, the ip% indicator confers a significant advantage to the NXCS predictor. To give the reader some basic information about trading statistics, the overall number of long and short signals has also been reported—for both the cases of a system equipped with an NXCS module and with an LRANN module.

4.3. Statistical significance of the results

Perhaps the most important question that arises while examining experimental results is whether or not the defensive or aggressive NXCS strategy is significantly superior to the B&H one. To investigate this issue, let us make some final considerations. As reported in Table 6, the NXCS-based approach is compared with the B&H strategy according to two different perspectives: (i) percent of correct classifications, with minimal variations included or excluded, and (ii) ideal profit, which measures the ability of exploiting gain opportunities. According to the former perspective, NXCS performances are very similar (or even worst, for the COMIT index) to the ones obtained by adopting a B&H strategy. At a first glance, one may conclude that, on the average, there is no significant advantage of the NXCS-based approach with respect to the B&H strategy. This is certainly true when considering the whole trading history, whereas it may be at least controversial after disregarding minimal index variations (i.e., $\leqslant 1\%$), whose economic relevance is negligible. To look over carefully this issue, let us evaluate the statistical significance of such results. When a stationary regime holds, we can assume that the number m of correct classifications, in a test set of n days, follows the Binomial distribution:

$$\wp(m) = \binom{n}{m} p^m (1-p)^{n-m} \tag{15}$$

where p is the (unknown) probability of taking the correct classification for each single day. Unfortunately, as previously pointed out, financial time series are—at least—multistationary. Notwithstanding this difficulty, any results obtained starting from Eq. (15) might still have a statistical significance under the hypothesis that a multistationary process holds. In fact, since the overall population of experts has been devised to perform regime identification, everything goes as if the overall system is composed by a set of subsystems, each adapted to deal with a particular, stationary, regime. Thus, after setting $p = m/n$ (with n large enough), we can estimate the interval, centered in p, into which m/n falls (for an arbitrary test set) with a given confidence. For example, with 95% of confidence:

$$p - 1.96 \cdot \sqrt{\frac{p \cdot (1-p)}{n}} \leqslant \frac{m}{n} \leqslant p + 1.96 \cdot \sqrt{\frac{p \cdot (1-p)}{n}} \tag{16}$$

Using $n = 1000$ (i.e., the number of days actually tested) and $p \cong 0.595$ or $p \cong 0.544$ (for the COMIT and the S&P500 index, respectively), we obtain that the confidence interval is about $\pm 3\%$ in both cases. This means that the results obtained on the COMIT index, as expected, do not carry out a statistical significance, being the percent of correct classifications inside the confidence interval (i.e., $|61.1\% - 59.8\%| < 3\%$). As for the S&P500 index, the NXCS-based approach is just outside the confidence interval (i.e., $|59\% - 54.4\%| > 3\%$). Note that the difference between the NXCS-based approach and the B&H strategy for the COMIT index is—not surprisingly—worse than the one obtained on the S&P500 index: The more the B&H strategy performs well, the harder is to outperform it. According to the adopted criterion (i.e., disregarding minimal variations), notwithstanding the length of the testing periods, one cannot conclude that results are statistically significant, as opposite conclusions could be drawn by taking into account the COMIT and the S&P500 index.

Fortunately, experimental results evaluated on the mere percent of correct classifications do not take into account the economic relevance of each trading failure. In our opinion, it is most important to measure *when* a predictor succeeds or fails rather than comparing its overall performance with the one obtained by adopting the B&H strategy. In fact, it is crucial for a predictor to succeed when the stock index has a not-negligible variation, and vice versa. This is the main reason why we defined the *ip*% indicator, which accounts for hits and failures according to the corresponding variation of the stock index. As reported in Table 6, these results are quite impressive and should be considered favored with respect to the one evaluated on the percent of correct classification. The *ip*% indicator basically accounts for normalized returns, without considering any trading commission. To assess a more concrete version of this indicator able to include also trading commissions, for each stock

market index we derived from the corresponding returns time series another time series, devised to give information about trading results—with a lower granularity. For example, assuming that the time series $C = \{r_k | k = 1, 2, \ldots, 1000\}$ represents the returns obtained trading on the COMIT index, the derived time series represents the rate of change of such returns, defined as

$$C_N = \left\{ r'_k | r'_k = \frac{r_k - r_{k-N}}{r_{k-N}},\ k = N+1, N+2, \ldots, 1000 \right\} \tag{17}$$

where r_k is the return at day k, and $N \ll 1000$ is the rate-of-change window. For each trading strategy (i.e., B&H, NXCS defensive, and NXCS aggressive) and for each stock market index (i.e., COMIT and S&P500), two time series ($N = 30$ and 100) have been derived from the corresponding returns time series, thus giving rise to twelve time series. Afterwards, some relevant information has been extracted from them, reported in Table 7 (for $N = 30$) and in Table 8 (for $N = 100$). In particular, we put into evidence the ability of the NXCS defensive and aggressive strategies to perform better than the B&H strategy on both the percent of gains and losses. Each loss or gain column reports the number of days where the selected strategy has satisfied the associated constraint; for example, the column "Loss $\geqslant 5\%$" in Table 7 reports the number of days where the selected strategy had a loss of 5% or more. In almost all cases, for both $N = 30$ and 100, the NXCS defensive and aggressive strategies (the aggressive one first) perform better than the B&H strategy.

As a concluding remark, let us note that several metrics give their contribution to the hypothesis that NXCS strategies perform better than the B&H one. The annualized returns, the Sharpe ratio, and the merit factor (all reported in Table 5), point out the superiority of the NXCS-based approach, from both an economically biased and a risk-adjusted perspective. In addition, the "rate of change" series discussed above, give further indications about the superiority of the NXCS-based approach vs. the B&H strategy.

Table 7
Gain and loss statistics, evaluated on a mobile window of 30 trading days

Thirty days trading statistics				Loss			Gain		
Stock index	TS	AR%	σ_{30}	$\geqslant 5\%$	$\geqslant 10\%$	$\geqslant 15\%$	$\geqslant 10\%$	$\geqslant 15\%$	$\geqslant 20\%$
COMIT	B&H	3.76%	3.08	160	44	29	228	144	56
	Def.ve	4.18%	2.80	85	10	0	203	119	41
	Agg.ve	4.40%	2.90	134	39	3	242	112	34
S&P500	B&H	0.94%	2.40	124	32	5	40	7	0
	Def.ve	1.90%	2.39	104	25	3	59	12	0
	Agg.ve	2.83%	2.66	124	31	5	121	54	11

TS = trading strategy, AR% = percent of annualized returns, σ = standard deviation.

Table 8
Gain and loss statistics, evaluated on a mobile window of 100 trading days

100 Days trading statistics				Loss			Gain		
Stock index	TS	AR%	σ_{30}	$\geqslant$5%	$\geqslant$10%	$\geqslant$15%	$\geqslant$10%	$\geqslant$15%	$\geqslant$20%
COMIT	B&H	10.13%	3.08%	160	44	29	228	144	56
	Def.ve	12.10%	2.80%	85	10	0	203	119	41
	Agg.ve	12.32%	2.90%	134	39	3	242	112	34
S&P500	B&H	4.12%	2.40%	124	32	5	40	7	0
	Def.ve	6.74%	2.39%	104	25	3	59	12	0
	Agg.ve	9.24%	2.66%	124	31	5	121	54	11

TS = trading strategy, AR% = percent of annualized returns, σ = standard deviation.

4.4. Overall characteristics of the NXCS-based approach

According to the results reported in Figs. 4 and 5, one may conclude that—during bullish or quiet periods—the NXCS-based approach shows results that are very similar to the ones obtained by adopting the B&H strategy. On the other hand it is able to gain a significant advantage over the B&H strategy during bearish periods–at least from an economically biased perspective. This is not surprising for a number of reasons. As for bullish periods, one must admit that it is difficult to outperform the B&H strategy when 60% (or more) of the days reserved for actual trading show a positive index variation. As for quiet periods, let us observe that they are mostly characterized by minimal index variations, which makes it difficult to properly exploit them. In fact, let us recall that our genetic guards embody technical analysis indicators, which give useful information only when not negligible variations occur. Fortunately, the above weaknesses are more than balanced by the behavior of the NXCS approach in correspondence with bearish periods, in which the NXCS-based approach is able to gain an advantage over the B&H strategy. Summarizing the results from a trader-oriented perspective, we can regard the overall system as an investor able to exploit positive trends, but especially able to recognize and make a profit out of market falls (thus reducing the overall risk), whereas in quiet times the minimal variations of the index do not allow to make a discernible profit. This ''trading identikit'' is particularly important, since it is very difficult for real traders to outperform the B&H strategy for long time periods like the ones used for making experiments with the proposed system.

5. Conclusions and future work

In this paper, a novel approach to perform stock market forecasting has been presented and described from the conceptual and technical perspectives.

Conceptually, predictions are obtained by following a divide-and-conquer strategy, based on the idea that partitioning the input space according to a significant amount of domain knowledge would facilitate the prediction task. To let the reader better understand the motivations of this choice, the non-stationary characteristics of financial time series have been put into evidence while assuming that a particular kind of non-stationarity holds, i.e., multis-tationarity, characterized by the existence of several locally stationary regimes (which occur along the time series), separated by rapid transition events. Under this hypothesis, regime identification becomes a crucial task for a system aimed at dealing with financial time series forecasting. Our solution to this problem exploits a population of local experts, each being acknowledged to participate to the process of making predictions only on a subset of the input sequence. To support this perspective, an hybrid approach has been defined that results from the interaction of a population of experts, each embedding and integrating the genetic and neural technologies. Being aimed at making it clearer the characteristics of the proposed approach, the underlying framework (i.e., guarded experts) has been briefly illustrated. The framework is centered on a population of local experts, each containing a selector (i.e., a guard) and a classifier or a predictor, depending on the kind of tasks that must be tackled. Given an input, each guard controls the activation of the corresponding classifier/predictor, thus preventing it from taking part of a decision process related to an input that does not match the guard. Technically, the neural XCS approach has been described in detail, as an implementation of the guarded experts framework, explicitly taylored for prediction tasks. Any NXCS expert embodies an XCS classifier, devised to partition the input space by means of technical-analysis indicators, together with a feedforward ANN that takes into account past prices, among other relevant inputs. The way a population of NXCS experts must be handled has been illustrated from an XCS perspective; in particular, the fitness-updating and outputs-blending mechanisms have been described, together with all relevant adjustments required to perform the task of predicting the next-day price of a stock index. To investigate the performances of the proposed approach on real data, a system for stock market forecasting has been implemented and tested on two stock market indexes, also taking into account realistic trading commissions. The system has proved to perform very well in the selected application task, obtaining results superior to the B&H strategy for large test sets. Being aimed at making comparisons with other relevant AI techniques devised for predicting financial time series, we reported also some results—in terms of percent of correct classification—obtained by adopting a locally recurrent ANN in place of the NXCS predictor. As for the future work, we are investigating the conceptual relevance of what we call "context-based" model identification as opposed to other approaches built around the concept of state or inputs history. Furthermore, we are willing to assess the impact of other selection and outputs-blending mechanisms on the

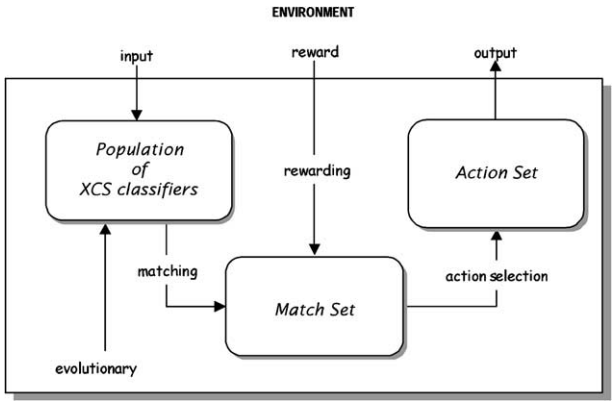

Fig. 6. The basic scheme of an XCS system.

performances of the overall NXCS-based system, in a multiple-experts perspective. An extension of the system, able to deal with the case of predicting the price of multiple stocks, is also being investigated.

Appendix A. A short introduction to XCS classifier systems

An XCS is an evolutionary learning system consisting of a performance, reinforcement, and discovery component. In XCS terminology, inputs—represented by bit strings of fixed length—are named messages, and outputs actions. The system handles a population of classifiers, each represented by a condition–action pair, together with some parameters that characterize the internal state of each classifier. Conditions are represented by a ternary string of bits (i.e., "0"/false, "1"/true, "#"/do not-care), and have the same length as incoming messages. Only when a message matches a classifier's condition, the corresponding action is selected, thus giving rise to a context-dependent behavior. The overall functional architecture of an XCS is shown in Fig. 6.

Given a classifier c, its most important parameters are: [10] p_c = prediction (or strength), expected reward to the system for using c; f_c = fitness, degree of adaptation of c to the environmental constraints; ε_c = prediction error, difference between expected and actual reward to the system for using c. As for the performance component, each time the system receives a message from the

[10] In absence of ambiguity, we will write p, ε, and f to denote p_c, ε_c, and f_c, respectively. This appendix has been included to help the reader concentrating on the relevant aspects of Wilson's XCSs, and is not meant to substitute the existing bibliography.

Table 9
Widrow–Hoff rules used for updating the most important parameters of an XCS classifier (the parameter β controls the "inertial" behavior of the update)

Operation	Formula	Where...
Prediction update	$p_{t+1} = p_t + \beta \cdot (r - p_t)$	r is the overall actual reward obtained by the system from the environment
Prediction error update	$\varepsilon_{t+1} = \varepsilon_t + \beta \cdot (\lvert p_{t+1} - r\rvert - \varepsilon_t)$	(an immediate reward is assumed, after each decision)
Fitness update	$f_{t+1} = f_t + \beta \cdot (k' - f_t)$	k' is the relative accuracy of a classifier, evaluated by normalizing its accuracy (k) over the action set corresponding to the selected action i^*

environment, a match set (M) is created containing all classifiers whose conditions are matched by the incoming message. Then the system has to select what action to take, according to its voting policy. The most common voting policy is the fitness-weighted majority rule, evaluated using a prediction array (P). Let M_i be the subset of M containing the classifiers whose action is i; the *i-th* element of the prediction array, say P_i, is calculated by means of the formula:

Note that P_i is an estimate of the reward the system can achieve for choosing the corresponding action. As for the reinforcement component, once the best action $i*$ is selected, and a reward r has been obtained by the system from the environment, all the classifiers in M_i* (the action set) are updated according to the Widrow–Hoff rules reported in Table 9. Note that the accuracy and the relative accuracy (k and k', respectively) of a classifier in $\mathbf{M}_{i*}$ can be evaluated as follows: [11]

Accuracy

$$k = \exp\left[\ln(\alpha) \cdot \frac{\varepsilon - \varepsilon_0}{\varepsilon_0}\right] \quad \begin{array}{l}\text{when } \varepsilon > \varepsilon_0,\\ \text{otherwise } k = 1\end{array}$$

Relative accuracy

$$k' = \frac{k}{\sum_{c \in \mathbf{M}_{i*}} k_c}$$

As for the discovery component, new classifiers can be generated in two different ways: (i) When no classifier exists that covers the incoming message (or the classifiers covering it have low fitness), a new classifier able to cover the message is created, its parameters being computed averaging over the whole

[11] The parameters a and ε_0 (i.e., accuracy fall-off and accuracy threshold, respectively) are set once, before running an XCS experiment. Note that $a = \alpha$ when the prediction error $\varepsilon = 2 \cdot \varepsilon_0$.

population of classifiers; (ii) Given a match set M, when the average time elapsed from the last mating (evaluated on the classifiers in M) exceeds a predefined threshold, the system selects two classifiers in M, with a probability proportional to each classifier's fitness. Then, crossover and mutation operators are applied to the selected classifiers, thus yielding two offspring, whose parameters are computed as the average of their parents.

When the maximum permissible size for the given population has been attained, a classifier must be selected for deletion. The probability of being deleted increases for classifiers with low fitness, and is proportional to the average cardinality of the match sets they have been involved with in the past. In this way, classifiers that frequently occur in large match sets have a higher probability of being deleted. It is worth pointing out that the reward resulting from the environment is shared among the classifiers belonging to the action set. In this way, separate niches tend to emerge, each of them hosting some (at least one) "specialized" classifiers. Furthermore, the fitness-sharing mechanism that distributes rewards among classifiers in each niche, together with the deletion criterion, tend to limit the number of classifiers that can survive in a niche.

References

[1] S.B. Achelis, Technical Analysis from A to Z, second ed., Irwin Professional Publishing, Chicago, 1995.
[2] T. Bollerslev, Generalized autoregressive conditional heteroskedaticicty, Journal of Econometrics 52 (1986) 307–327.
[3] J.Y. Campbell, A.W. Lo, A.C. MacKinlay, The Econometrics of Financial Markets, Princeton University Press, Princeton, New Jersey, 1997.
[4] P. Campolucci, A. Uncini, F. Piazza, B.D. Rao, On-line learning algorithms for locally recurrent neural networks, IEEE Transactions on Neural Networks 10 (2) (1999) 253–271.
[5] F. Castiglione, Forecasting price increments using an artificial neural network, Advances in Complex Systems 4 (1) (2001) 45–56.
[6] R.E. Dorsey, R.S. Sexton, The use of parsimonious neural networks for forecasting financial time series, Journal of Computational Intelligence in Finance 6 (1) (1998) 24–31.
[7] S.E. Fahlmann, C. Lebiere, The cascade-correlation learning architecture, Technical Report CMU-CS-90-100, Carnegie Mellon University, 1990.
[8] E.F. Fama, The behavior of stock market prices, The Journal of Business 38 (1965) 34–105.
[9] C.L. Giles, S. Lawrence, A. Chung Tsoi, Rule inference for financial prediction using recurrent neural networks, in: Proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering (CIFE), 1997, pp. 253–259.
[10] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, 1989.
[11] G. Hawawini, D.B. Keim, On the predictability of common stock returns: world-wide evidence, in: R.A. Jarrow, V. Maksimovic, W.T. Ziemba (Eds.), Handbook in Operations Research and Management Science, vol. 9, Finance, North-Holland, Amsterdam, 1995, pp. 497–544, Chapter 17.
[12] T. Hellström, K. Holmström, Predicting the stock market, Technical Report IMa-TOM-1997-07, Department of Mathematics and Physics, Malardalen University, Sweden, 1997.

[13] J.H. Holland, Adaptation, in: R. Rosen, F.M. Snell (Eds.), Progress in Theoretical Biology, vol. 4, Academic Press, New York, 1976, pp. 263–293.
[14] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Computation 3 (1991) 79–87.
[15] P.J.B. Hancock, Pruning neural nets by genetic algorithm, in: International Conference on Artificial Neural Networks, Elsevier, 1992, pp. 991–994.
[16] F. Kai, X, Wenhua, Training neural network with genetic algorithms for forecasting the stock price index, in: Proceedings of the 1997 IEEE International Conference on Intelligent Processing Systems, Beijing, China, October 1997, vol. 1, pp. 401–403.
[17] T. Kovacs, Evolving optimal populations with XCS classifier systems, MSc. Dissertation, University of Birmingham, UK, 1996.
[18] T. Kovacs, Strength or accuracy? A comparison of two approaches to fitness, in: A.S. Wu (Ed.), Second International Workshop on Learning Classifier Systems during GECCO99, Orlando, Florida, 1999, pp. 258–265.
[19] P.L. Lanzi, A study of the generalization capabilities of XCS, in: Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97), Morgan Kaufmann, San Francisco, CA, 1997, pp. 418–425.
[20] P.L. Lanzi, Adding memory to XCS, in: Proceedings of the IEEE Conference on Evolutionary Computation (ICEC98), 1998, pp. 609–614.
[21] P.L. Lanzi, A. Perrucci, Extending the representation of classifier conditions part II: from messy coding to S-expressions, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99), vol. 1, 1999, pp. 345–353.
[22] B. LeBaron, A.S. Weigend, A boostrap evaluation of the effect of data splitting on financial time series, IEEE Transactions on Neural Networks 9 (1997) 213–220.
[23] S. Mahfoud, G. Mani, Financial forecasting using genetic algorithms, Applied Artificial Intelligence 10 (6) (1996) 543–565.
[24] P. McCullagh, J.A. Nelder, Generalized Linear Models, Chapman and Hall, London, 1989.
[25] A. Muhammad, G.A. King, Foreign exchange market forecasting using evolutionary fuzzy networks, in: Proceedings of the IEEE /IAFE 1997 Computational Intelligence for Financial Engineering, 1997, pp. 213–219.
[26] D. Noever, S. Baskaran, Genetic algorithms trading on the S&P500, The Magazine of Artificial Intelligence in Finance 1 (3) (1994) 41–50.
[27] L.R. Rabiner, B.H. Juang, An introduction to Hidden Markov models, IEEE ASSP Magazine 3 (1) (1986) 4–16.
[28] A.P.N. Refenes, A.N. Burgess, Y. Bentz, Neural networks in financial engineering: a study in methodology, IEEE Transactions on Neural Networks 8 (6) (1997) 1222–1267.
[29] D.E. Rumelhart, J.L. McLelland, in: Parallel Distributed Processing, Explorations in the Microstructure of Cognition, vol. 1, Foundations, MIT Press, 1986.
[30] W.F. Sharpe, Adjusting for risk in portfolio performance measurement, Journal of Portfolio Management (1975).
[31] S. Shi, A.S. Weigend, Taking time seriously: Hidden Markov experts applied to financial engineering, in: Proceedings of the 1997 IEEE/IAFE Conference on Computational Intelligence for Financial Engineering, 1997, pp. 244–252.
[32] A.S. Weigend, B.A. Huberman, D.E. Rumelhart, Predicting sunspots and exchange rates with connectionist networks, in: Proceedings of the 1990 NATO Workshop on Nonlinear Modeling and Forecasting, Addison Wesley, Santa Fe, NM, USA, 1991.
[33] A.S. Weigend, M. Mangeas, A.N. Srivastava, Nonlinear gated experts for time series: discovering regimes and avoiding overfitting, International Journal of Neural Systems 6 (1995) 373–399.
[34] A.S. Weigend, Time series analysis and prediction using gated experts with application to energy demand forecast, Applied Artificial Intelligence 10 (1996) 583–624.

[35] A.S. Weigend, H.G. Zimmermann, Exploiting local relations as soft constraints to improve forecasting, Journal of Computational Intelligence in Finance 6 (1998) 14–23.
[36] S.W. Wilson, Classifier fitness based on accuracy, Evolutionary Computation 3 (2) (1995) 149–175.
[37] S.W. Wilson, Generalization in the XCS classifier System, in: Proceedings of the Third annual Genetic Programming Conference, Morgan Kaufmann, San Francisco, CA, 1998, pp. 665–674.
[38] S.W. Wilson, State of XCS classifier system research, Second International Workshop on Learning Classifier Systems during GECCO99, 1999.
[39] S.W. Wilson, Get Real! XCS with Continuous-Valued Inputs, in: Festschrift in Honor of John H. Holland, L.Booker, S. Forrest, M. Mitchell, and R. Riolo (Eds.), Center of Study of Complex Systems, May 15–18, The University of Michigan, ANN Arbor, MI, 1999, pp. 111–121.