Neural Computing and Applications (2020) 32:4187–4211
https://doi.org/10.1007/s00521-019-04231-8

EMERGING TRENDS OF APPLIED NEURAL COMPUTATION - E_TRAINCO

Genetic and deep learning clusters based on neural networks for management decision structures

Will Serrano[1]

Received: 15 September 2018 / Accepted: 9 May 2019 / Published online: 20 May 2019

Abstract

Judgments are taken in a structured way; both human and business management decisions involve a hierarchical process that requires a level of compromise between risk, cost, reward, experience and knowledge. This article proposes a management decision structure that emulates the human brain approach based on genetic and deep learning cluster algorithms and the random neural network. Reinforcement learning takes quick and specific local decisions, deep learning clusters enables identity and memory, and deep learning management clusters make final strategic decisions. The presented genetic algorithm transmits the learned information to future generations in the network weights rather than the neurons. Because the subject's information, a combination of memory, identity and decision data, is never lost but transmitted, the genetic algorithm provides immortality. The management decision structure has been applied and validated in a smart investment Fintech application: an intelligent banker that makes buy and sell asset decisions with an associated market and risk that entirely transmits itself to a future generation. Results are rewarding; the management decision structure with genetics and machine learning based on the random neural network algorithm that emulates the human brain and biology transmits information to future generations and learns autonomously, gradually and continuously while adapting to the environment.

Keywords Genetic learning · Deep learning clusters · Reinforcement learning · Random neural network · Fintech · Smart investment

1 Introduction

The brain takes decisions in a structured way, and the striatum is a fundamental area essential in decision making formed of different sections that have distinct roles: the dorsolateral striatum functions in typical actions, the dorsomedial striatum in goal-directed actions and the ventral striatum in motivation [1]; although these three different regions have independent functions, they coordinate between each other during the different stages of decision-making process based on hierarchical reinforcement learning. Similar structured decision-making approach is also applied in business management, company and institution operations and emergency services that require a level of compromise between risk, cost, reward, experience and knowledge.

In addition to our brain hierarchical decision process, biological organisms autonomously learn in a gradual and continuous approach while adapting to the environment using genetic changes to generate new complex structures in organisms [2]. The current structure of the organisms defines the type and level of the future genetic variation that will provide a better adaption to the environment or increased reward to a goal function. Random genetic changes have more probability to be successful in organisms that change in a systematic and modular manner where the new structures acquire the same set of subgoals in different combinations. This approach enables organisms not only to remember their reward evolution but also to generalize goal functions to successfully adapt to future environments [3]. The adaptations learned from the living organisms affect and guide evolution, even though the

✉ Will Serrano
g.serrano11@imperial.ac.uk

[1] Intelligent Systems and Networks Group, Electrical and Electronic Engineering, Imperial College London, London, UK

characteristics acquired are not transmitted to the genome [4]; however, its gene functions are altered and transmitted to the new generation. This method enables learning organisms to evolve much faster.

Successful machine learning and artificial intelligence models have been based on biology emulating the structures provided by nature during the learning, adaptation and evolution when interacting with the external environment. Neural networks and deep learning are based on brain structure which is formed of dense local clusters of the same neurons. Dense clusters perform different functions which are connected between each other with numerous very short paths and few long distance connections [5]. The brain retrieves a large amount of data obtained from the senses; analyzes the material and finally selects the relevant information [6] where the cluster of neurons specialization occurs due to their adaption when learning tasks.

1.1 Research proposal

This article proposes a management decision structure that emulates the brain functions using reinforcement learning and deep learning clusters based on the random neural network. Information in the presented model is learned through the interaction and adaptation to the environment using reinforcement and deep learning. Decisions are taken in a hierarchical way with different learnings specialized in different stages of the decision process:

- Reinforcement learning [7–9] takes quick and specific local decisions;
- Deep learning clusters [10–12] enable identity and memory;
- Deep learning management clusters [13–16] make final strategic decisions.

In addition, this article presents a genetic learning algorithm based on the genome and evolution applying extreme learning machine methods [17–21]. Information in the proposed genetic algorithm is transmitted to future generations in the network weights through the combinations of four different nodes rather than the value of nodes themselves. The four nodes represent the genome nucleotides (C, G, A or T) that form the double helix of the DNA where the output layer of nodes replicates the input layer as the genome reproduces replicas of organisms. The genetic learning algorithm fixes the output to four neuron values that represent the four different nucleoids, and it also fixes the network weights to generate four different types of neurons rather than random values as proposed by the ELM theory. Genetic algorithm provides immortality: the entire subject's information, defined as the combination of memory, identity and decision data, is never lost but transmitted to future generations.

The proposed management decision structure has been applied and validated in a smart investment application: an intelligent banker that makes buy and sell asset decisions with an associated market and risk that entirely transmits itself to a future generation.

The results presented by this article are rewarding and promising: the intelligent banker takes the right decisions, learns the variable asset price, makes profits on specific markets at minimum risk and finally efficiently transmits the information learned to future generations.

1.2 Research structure

This article presents the research background that consists on the genome, artificial neural networks, machine learning, genetic algorithms and extreme machine learning in Sect. 2. The random neural network with reinforcement learning, deep learning clusters, deep learning management clusters and genetic algorithm are defined in Sect. 3. The management decision structure and its application to smart investment in an intelligent banker are presented in Sect. 4, whereas its implementation is described in Sect. 5. The experimental results are shown in Sect. 6 with a cryptocurrency evaluation in Sect. 7. Finally, conclusions and future work are shared in Sect. 8.

2 Research background and literature review

Artificial neural networks, machine learning and genetic algorithms have been applied in economics and finance to make predictions where extreme machine learning have improved the performance of classic neural network models.

2.1 Genome

The genome is the genetic material of an organism; it consists of 23 pairs of chromosomes (1–22, *X* and *Y*) for a human cell formed of genes (approximately 21,000 in total) that code for a molecule that has a function or instruction to make proteins as presented by Pellegrini et al. [22]. Furthermore, genes are formed of base pairs (approximately 3 billion in total). The DNA is a double helix formed by the combination of only four nucleotides (cytosine [C], guanine [G], adenine [A] or thymine [T]) where each base pair consists of the combination of two nucleoids G-C and A-T. The genetic code is formed of codons; a sequence consisted of three nucleotides or three-letter words. Proteins that have similar combination of base pairs tend to have a related functionality determination of protein functions from genetic sequences following the research of Suzuki [23].

2.2 Artificial neural networks

Artificial neural networks have been applied to make financial predictions and represent financial models. The bankruptcy prediction capability of several neural network architectures based on different training sets and number of iterations was evaluated by Leshno and Spector [24]; the neural networks are trained with data obtained from different firm's financial reports where the prediction capability of the neural network is compared against classical discriminant analysis models. Artificial neural networks for a financial distress prediction model are used by Chen and Du [25]; the back propagation learning algorithm is trained with a dataset obtained from the Taiwan Stock Exchange Corporation where the inputs to the neural network are 37 model factor ratios. An artificial neural network with back propagation gradient descent learning algorithm to predict the direction of stock market index movement for the Istanbul Stock Exchange is applied by Kara et al. [26]; the inputs of the feedforward network correspond to ten technical indicators such as moving average or momentum and the output neuron represents the direction of the index movement.

The effectiveness of neural network models in stock market predictions was evaluated by Guresen et al. [27]; the models analyzed are multilayer perceptron, dynamic artificial neural network and hybrid neural networks where the mean square error and mean absolute deviate metrics are used to compare each model. Different artificial neural networks in bankruptcy prediction against traditional Bayesian classification theory are analyzed by Zhang et al. [28]; the method of cross-validation is applied to examine the sample variation between neural networks. Different ways to use prior knowledge such as newspaper headlines and neural networks to improve multivariate prediction ability is investigated by Kohara et al. [29]; the topics are chief Tokio stock exchange price index, exchange rate dollar-yen, interest rate, crude oil price and New York Dow Jones where the inputs to the neural network are the relative topic difference. Regression, artificial neural networks and support vector machines for predicting the S&P 500 Stock Market Price Index are compared by Sheta et al. [30]; they use 27 potential financial and economic variables that impact the stock movement; these variables are used as the input nodes whereas the output node gives the predicted next week value. Artificial neural networks and fuzzy logic for market predictions are included by Khuat et al. [31] where the input layer contains 30 neurons corresponding to 30 close days and the output node is the close price of the next day.

A feedforward multilayer perceptron to predict a company's stock value is used by Naeini et al. [32]; the network predicts the next day stock value of a company listed in the Tehran Stock Exchange Corporation only based on its stock trade history and without any information of the current market. A hybrid system based on a multiagent architecture to analyze stock market behavior is created by Iuhasz et al. [33] to improve the profitability in a short or medium time period investment; the proposed system compares the results of feed forward and recurrent neural network in terms of accuracy and time performance. The use of neural networks as an alternative to classical statistical techniques for forecasting within the framework of arbitrage pricing theory model for stock ranking is examined by Nicholas et al. [34]; the training and test sets consist of data presented as factors extracted from the balance sheets of the companies in the UK stocks, and the resultant outperformance Y is the output.

A comparative survey of artificial intelligence applications in Finance is presented by Bahrammirzaee [35] that covers artificial neural networks, expert system and hybrid intelligent systems in financial markets, credit evaluation, portfolio management and financial prediction and planning. The use of artificial neural networks in accounting and finance is reviewed by Coakley and Brown [36]; it includes modeling issues and applicability guidelines such as the selection of the learning algorithm, error and transfer functions, architecture and network training. The applications of neural networks in finance are analyzed by Fadlalla and Lin [37]; in particular, the common characteristics of these applications are examined and compared against applications based on statistical and econometrics models. The use of neural networks in finance and economics forecasting is reviewed by Huang et al. [38]; input variables, type of neural network models and performance comparisons are analyzed for the prediction of foreign exchange rates, stock market index and economic growth. Li et al. [39] summarize different applications of artificial intelligence technologies such as neural networks, deep learning and machine leaning in several domains of business administration including finance, retail, manufacturing and management consultancy.

Machine learning has been applied to solve nonlinear models in continuous time in macroeconomics and finance by Duarte [40] where the problem of solving the corresponding nonlinear partial differential equations can be reformulated as a sequence of supervised learning problems. A single variable time-series model that combines two financial volatility metrics to predict and forecast one of them is proposed by Stefani et al. [41]; the method is based on artificial neural networks with a multilayer perceptron in a single-hidden-layer configuration, the k-nearest neighbors as a local nonlinear model used for classification and regression and finally support vector machine in a regression methodology. Deep learning has also been incorporated in long–short-term memory neural networks for financial market predictions by Fischer and Krauss [42] where day returns are calculated for each day at defined stocks. Hasan et al. [43] investigate how to apply hierarchical deep learning models for the problems in finance such as stock market prediction and classification; the deep learning models are based on neural networks, recurrent neural networks with big data finance datasets.

2.3 Genetic algorithms

Genetic algorithms (GA) have been proposed as method to increase learning performance. Overlapping generations (OLG) economies in which agents use genetic algorithms to learn correct decision rules are studied by Arifovic [44]; the results of an OLG model with GA learning against the results of the same model where the agents form expectations via either the sample average of past prices or least squares adaptive algorithms are compared in terms of on equilibrium within inflationary economies. A genetic algorithm to feature discretization in artificial neural networks for the prediction of stock market index was proposed by Kim and Han [45]; the GA is applied to improve the learning algorithm and to reduce the complexity in feature space.

A hybrid model based on genetic algorithm and neural networks to forecast tax collection is applied by Ticona et al. [46]; endogenous and exogenous variables are used as input variables of the neural network for the multistep time series to forecast a hybrid model based on lags of the value of the time series, differences and moving averages. A genetic algorithm-based deep learning method is presented by Hossain and Capi [47]; the GA is used to optimize the deep learning parameters such as the number of hidden units, the number of epochs, learning rates and momentum in learning stage of the hidden layers. A genetic algorithm assisted method for deep learning that improves the performance of a deep autoencoder producing a sparser neural network is presented by David and Greental [48]; the GA population of each chromosome is a set of weights for the autoencoder. The latest deep learning structures and evolutionary algorithms that can be used to train them are reviewed by Tirumala [49]; these include convolutional neural networks, deep belief networks, stacked autoencoders, generative neuroevolution and deep learning using genetic algorithm.

2.4 Extreme learning machine

The learning speed of feedforward neural networks is in general slower than required due to the slow gradient-based learning algorithm that adjusts iteratively the parameters of the networks. A new learning algorithm called extreme learning machine (ELM) for single-hidden-layer feedforward neural networks (SLFNs) which randomly chooses hidden nodes and analytically determines its output weight is proposed by Huang et al. [17]. The output weights linking the hidden layer to the output layer of SLFNs can be analytically determined through simple generalized inverse operation of the hidden layer output matrices; Huang et al. [18] proves that SLFNs work as universal approximators by randomly choose hidden nodes and then only adjusting the output weights linking the hidden layer and the output layer. An ELM architecture for multilayer perceptron is proposed by Tang et al. [19]; the ELM is divided into two main components: self-taught feature extraction followed by supervised feature classification which they are connected by random initialized hidden weights. ELM provides a unified learning platform with extensive methods of feature mappings that can be applied in regression and multiclass classification applications directly, as demonstrated by Huang et al. [20]. An extreme learning machine-based autoencoder is introduced by Kasun et al. [21] which learns feature representations using singular values.

Reservoir computing consists on an input signal that is fed into a fixed or random dynamical system, or reservoir, where the dynamics of the reservoir map the input to a higher dimension, then a retrieval method is trained to read the state of the reservoir and map it to the desired output [50]; the main benefit is that the reservoir is fixed and training is performed only at the retrieval stage therefore the complexity of the computationally demanding neural networks learning algorithms is reduced. An empirical analysis of deep recurrent neural network architectures with stacked layers develops and improves hierarchical dynamics in deep recurrent architectures within the efficient reservoir computing approach [51]; a deep layering of recurrent models provides an effective diversification of temporal representations in the layers of the hierarchy. Deep Echo State Network models consist of a stack of multiple nonlinear reservoir layers that potentially allow the exploitation of the advantages of a hierarchical temporal feature representation at different levels of abstraction while preserving the training efficiency typical of the reservoir computing methodology [52]; however, adding layers to a deep reservoir architecture affects the regime of network's dynamics toward equally or less stable behaviors. A new architecture, reservoir with random static projections is proposed to improve the performance of Echo State Networks based on the compromise between the amount of nonlinear mapping and short-term memory when applied to time-series data which are highly nonlinear [53]; a similar method is also applied using an ELM whose input is presented through a time delay.

Radial basis function (RBF) networks that have one hidden layer are capable of universal approximation [54]; RBF networks with the same smoothing factor in each kernel node is broad enough for universal approximation. ELMs can be also extended to a RBF network case, which allows the centers and impact widths of RBF kernels to be randomly generated and the output weights to be simply analytically calculated instead of iteratively tuned [55]; the ELM algorithm for RBF networks can complete learning at very fast speed and produce generalization performance very close to support vector machines in many artificial and real benchmarking function approximation and classification applications. The RBF network applies a nonmonotonic transfer function based on the Gaussian density function [56]; while producing robust decision surfaces, the RBF also provides an estimate of how close a test instance is to the original training data, allowing the classifier to identify that a test instance potentially represents a new class while still presenting the most probable classification. Feedforward neural networks using RBF assume that the patterns of the learning environment are separable by hyperspheres [57]; it is demonstrated that their related cost function is local minima free with respect to all the network weights.

3 The random neural network genetic deep learning model

3.1 The random neural network

The random neural network (RNN) [7–9] represents more closely how signals are transmitted in many biological neural networks where they travel as spikes or impulses, rather than as analogue signal levels. The RNN is a spiking recurrent stochastic model for neural networks. Its main analytical properties are the "product form" and the existence of the unique network steady state solution. The random neural network has also been applied in different genetic models [58–67].

The RNN is composed of M neurons each of which receives excitatory (positive) and inhibitory (negative) spike signals from external sources which may be sensory sources or neurons (Fig. 1). These spike signals occur following independent Poisson processes of rates $\lambda^{+}(m)$ for the excitatory spike signal and $\lambda^{-}(m)$ for the inhibitory spike signal, respectively, to cell $m \in \{1, \ldots M\}$.

Neurons interact with each other by interchanging signals in the form of spikes of unit amplitude:

- A positive spike is interpreted as excitation signal because it increases by one unit the potential of the receiving neuron;

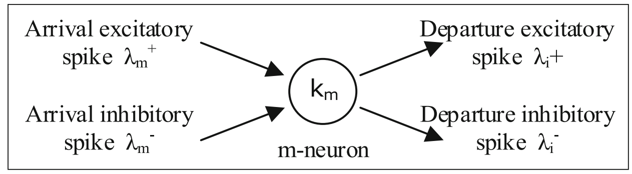


Fig. 1 The random neural network

- A negative spike is interpreted as inhibition signal decreasing by one unit the potential of the receiving neuron or has no effect if the potential is already zero.

Each neuron accumulates signals and it will fire if its potential is positive. Firing will occur at random, and spikes will be sent out at rate $r(i)$ with independent, identically and exponentially distributed inter-spike intervals:

- Positive spikes will go out to neuron j with probability $p^{+}(i, j)$ as excitatory signals;
- Negative spikes with probability $p^{-}(i, j)$ as inhibitory signals.

A neuron may send spikes out of the network with probability $d(i)$. We have:

$$d(i) + \sum_{j=1}^{n} [p^{+}(i,j) + p^{-}(i,j)] = 1 \quad \text{for } 1 \leq i \leq n \tag{1}$$

Neuron potential decreases by one unit when the neuron fires either an excitatory spike or an inhibitory spike. External (or exogenous) excitatory or inhibitory signals to neuron i will arrive at rates $\Lambda(i)$, $\lambda(i)$, respectively, by stationary Poisson processes. The random neural network weight parameters $w^{+}(j, i)$ and $w^{-}(j, i)$ are the nonnegative rate of excitatory and inhibitory spike emission, respectively, from neuron i to neuron j:

$$\begin{aligned} w^{+}(j,i) &= r(i)p^{+}(i,j) \geq 0 \\ w^{-}(j,i) &= r(i)p^{-}(i,j) \geq 0 \end{aligned} \tag{2}$$

Information is transmitted by the rate or frequency at which spikes travel. Each neuron i, if it is excited, behaves as a frequency modulator emitting spikes at rate $w(i, j) = w^{+}(i, j) + w^{-}(i, j)$ to neuron j. Spikes will be emitted at exponentially distributed random intervals. Each neuron acts as a nonlinear frequency demodulator transforming the incoming excitatory and inhibitory spikes into potential.

In this model, each neuron is represented at time $t \geq 0$ by its internal state $k_m(t)$ which is a nonnegative integer. If $k_m(t) \geq 0$, then the arrival of a negative spike to neuron m at time t results in the reduction of the internal state by one unit: $k_m(t^{+}) = k_m(t) - 1$. The arrival of a negative spike to a neuron has no effect if $k_m(t) = 0$. On the other hand, the arrival of an excitatory spike always increases the neuron's internal state by 1; $k_m(t^{+}) = k_m(t) + 1$.

The random neural network defines q_i as the probability a neuron i is excited:

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)} r(i) = \sum_{j=1}^{n} [w^+(i,j) + w^-(i,j)] \quad \text{for } 1 \leq i \leq n \tag{3}$$

where the $\lambda^+(i)$, $\lambda^-(i)$ for $i = 1, \ldots, n$ satisfy the system of nonlinear simultaneous equations:

$$\begin{aligned} \lambda^+(i) &= \sum_{j=1}^{n} \left[q_j r(j) p^+(j,i) \right] + \Lambda(i) \\ \lambda^-(i) &= \sum_{j=1}^{n} \left[q_j r(j) p^-(j,i) \right] + \lambda(i). \end{aligned} \tag{4}$$

3.2 Reinforcement learning algorithm

A random neural network (RNN) [7–9] with at least as many nodes as the number of decisions to be taken is generated where neurons are numbered 1, …, j, …, n; therefore, for any decision i, there is some neuron i. Decisions in this RL algorithm with the RNN are taken by selecting the decision j for which the corresponding neuron is the most excited, the one which has the largest value of q_j.

The state q_j is the probability that it is excited, these quantities satisfy the following system of nonlinear equations:

$$q_j = \frac{\lambda^+(j)}{r(j) + \lambda^-(j)} \tag{5}$$

The reinforcement learning algorithm used in this model is based on the cognitive packet network presented by Gelenbe [68–72]. Given some Goal G that the agent has to achieve as a function to be optimized and reward R as a consequence of the interaction with the environment, successive measured values of the R are denoted by R_l, l = 1, 2, … and these are used to compute a decision threshold:

$$T_l = \alpha T_{l-1} + (1 - \alpha) R_l \tag{6}$$

where α is some constant $0 < \alpha < 1$ that can be statically assigned or dynamically updated based on the external observations.

The agent takes the lth decision which corresponds to neuron j and then the lth reward R_l is measured and its associated T_{l-1} is calculated where the network weighs are updated as follows for all neurons $i \neq j$.

$$\begin{aligned} &\text{if } T_{l-1} \leq R_1: \\ &w^+(i,j) = w^+(i,j) + R_1 \\ &w^-(i,k) = w^-(i,k) + \frac{R_l}{n-2} \quad \text{if } k \neq j \end{aligned} \tag{7}$$

$$\begin{aligned} &\text{else if } R_l < T_{l-1}: \\ &w^+(i,k) = w^+(i,k) + \frac{R_l}{n-2} \quad \text{if } k \neq j \\ &w^-(i,j) = w^-(i,j) + R_l \end{aligned} \tag{8}$$

This research uses reinforcement learning to make binary decisions with only two neurons (Fig. 2).

$$q_0 = \frac{\lambda^+(0)}{r(0) + \lambda^-(0)} \quad q_1 = \frac{\lambda^+(1)}{r(1) + \lambda^-(1)}$$

where

$$\begin{array}{ll} \lambda^+(0) = q_1 w_{10}^+ + \Lambda_0 & \lambda^+(1) = q_0 w_{01}^+ + \Lambda_1 \\ \lambda^-(0) = q_1 w_{10}^- + \lambda_0 & \lambda^-(1) = q_0 w_{01}^- + \lambda_1 \\ r(0) = w_{01}^+ + w_{01}^- & r(1) = w_{10}^+ + w_{10}^-. \end{array} \tag{9}$$

On the above equations, w_{ij}^+ is the rate at which neuron i transmits excitation spikes to neuron j and w_{ij}^- is the rate at which neuron i transmits inhibitory spikes to neuron j in both situations when neuron i is excited. Λ_i and λ_i are the rates of external excitatory and inhibitory signals, respectively.

3.3 The random neural network with multiple clusters

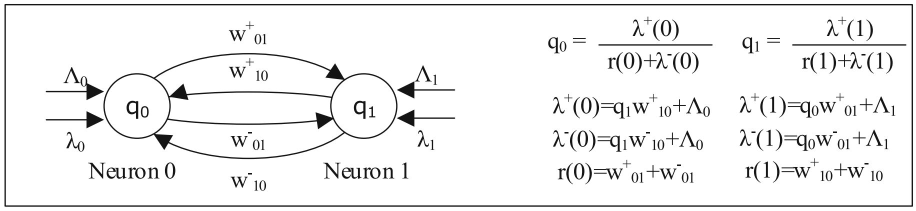
Deep learning with random neural networks is described by Gelenbe and Yin [10–12]. This model is based on the generalized queuing networks with triggered customer movement (G-networks) where customers or tasks are either "positive" or "negative" and customers or tasks can be moved from queues or leave the network. G-networks are introduced by Gelenbe [73, 74]; an extension to this model is developed by Gelenbe et al. [75] where synchronized interactions of two queues could add a customer in a third queue.

The model considers a special network $M(n)$ that contains n identically connected neurons, each which has a firing rate r and external inhibitory and excitatory signals λ^- and λ^+, respectively (Fig. 3). The state of each cell is denoted by q, and it receives an inhibitory input from the state of some cell u which does not belong to $M(n)$; therefore, for any cell $i \in M(n)$, there is an inhibitory weight $w^-(u) \equiv w^-(u,i) > 0$ from u to i.

For any $i,j \in M(n)$, we have $w^+(i,j) = w^-(i,j) = 0$, but all whenever one of the neurons fires, it triggers the firing of the other neurons with the following values. As presented in [14–16], the potential for each identical neuron is calculated as:

$$q = \frac{\Lambda + \frac{rq(n-1)(1-p)}{n-qp(n-1)}}{r + \lambda + q_u w^-(u) + \frac{rqp(n-1)}{n-qp(n-1)}} \tag{10}$$

Fig. 2 Reinforcement learning algorithm



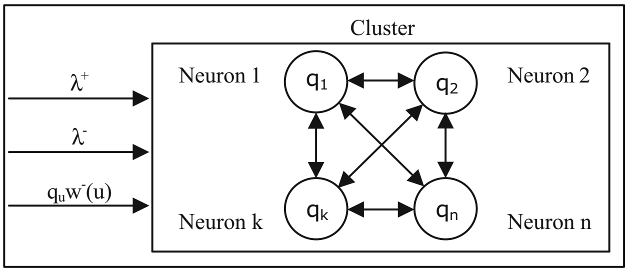


Fig. 3 Clusters of neurons

where (10) is a second-degree polynomial in q:

$$0 = q^2 p(n-1)[\lambda + q_u w^-(u)] - q[np(\Lambda + r) + n(\lambda + q_u w^-(u)) - p(\Lambda + r) + r] + n\Lambda. \tag{11}$$

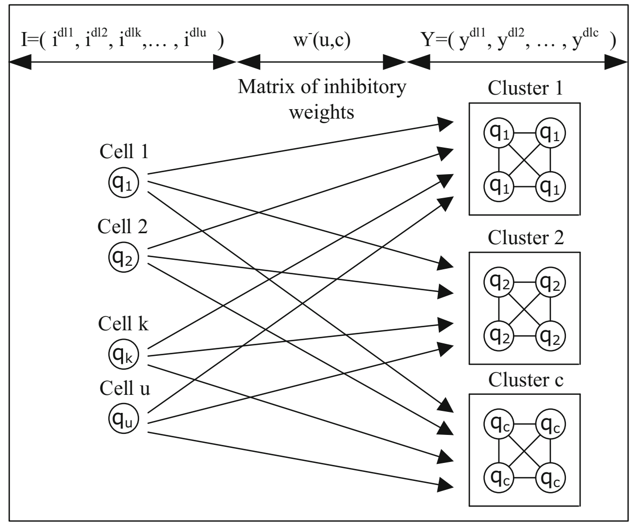


Fig. 4 The random neural network with multiple clusters

3.4 Deep learning clusters

The deep learning architecture presented in [10–12] is composed on C multiple clusters, each of which is made up of a $M(n)$ cluster each with n hidden neurons (Fig. 4). For the cth such cluster, $c = 1, \ldots, C$, the state of each of its identical cells is denoted by q_c. In addition, there are U input cells which do not belong to these C clusters, and the state of the uth cell $u = 1, \ldots, U$ is denoted by $\bar{q}_u$. The cluster network has U input cells and C clusters.

The deep learning clusters model defines:

- $I = (i_{dl1}, i_{dl2}, \ldots, i_{dlu})$, a U-dimensional vector $I \in [0,1]^U$ that represents the input state $\overline{q_u}$ for the cell u;
- $w^-(u, c)$ is the $U \times C$ matrix of inhibitory weights from the U input cells to the cells in each of the C clusters;
- $Y = (y_{dl1}, y_{dl2}, \ldots, y_{dlc})$, a C-dimensional vector $Y \in [0,1]^C$ that represents the cell state q_c for the cluster c.

Each hidden neuron in the clusters c, with $c \in \{1, \ldots, C\}$ receives an inhibitory input from each of the U input neuron. Thus, for each neuron in the cth cluster, we have inhibitory weights $w^-(u, c) > 0$ from the uth input neuron to each neuron in the cth cluster; the uth input neuron will have a total inhibitory "exit" weight, or total inhibitory firing rate $\overline{r_u}$ to all the clusters which is of value:

$$\overline{r_u} = n \sum_{c=1}^{C} w^-(u, c) \tag{12}$$

As calculated in (10), the potential for each neuron q_c in cluster C is:

$$q_c = \frac{\Lambda_c + \frac{r_c q_c (n-1)(1-p_c)}{n - q_c p_c (n-1)}}{r_c + \lambda_c + \sum_{u=1}^{U} \overline{q_u} w^-(u,c) + \frac{r_c q_c p_c (n-1)}{n - q_c p_c (n-1)}} \tag{13}$$

The activation function of the cth cluster is defined as:

$$\zeta(x_c) = \frac{b_c}{2a_c} - \frac{\sqrt{b_c^2 - 4a_c d_c}}{2a_c} \tag{14}$$

where

$$x_c = \sum_{u=1}^{U} \overline{q_u} w^-(u,c) \tag{15}$$

We have:

$$y_c = \zeta(x_c)$$

The network learns the $U \times C$ weight matrix $w^-(u, c)$ by calculating new values of the network parameters for the input I and output Y using gradient descent learning algorithm which optimizes the network inhibitory weight parameters $w^-(u, c)$ from a set of input–output pairs (i_u, y_c).

3.5 Deep learning management cluster

The deep learning management cluster was proposed by Serrano et al. [13–16]. It takes management decisions based on the inputs from different deep learning clusters (Fig. 5). The deep learning management cluster defines:

- I_{mc}, a C-dimensional vector $I_{mc} \in [0,1]^C$ that represents the input state $\overline{q_c}$ for the cluster c;
- $w^-(c)$ is the C-dimensional vector of inhibitory weights from the C input clusters to the cells in the management cluster mc;
- Y_{mc}, a scalar $Y_{mc} \in [0,1]$, the cell state q_{mc} for the management cluster mc.

The activation function of the management cluster mc is defined as:

$$\zeta(x_{mc}) = \frac{[np(\Lambda_{mc}+r_{mc})+n(\lambda_{mc}+x_{mc})-p(\Lambda_{mc}+r_{mc})+r_{mc}]}{2p_{mc}(n-1)[\lambda_{mc}+x_{mc}]} - \frac{\sqrt{[np(\Lambda_{mc}+r_{mc})+n(\lambda_{mc}+x_{mc})-p(\Lambda_{mc}+r_{mc})+r_{mc}]^2-4p_{mc}(n-1)[\Lambda_{mc}+x_{mc}]n\Lambda_{mc}}}{2p_{mc}(n-1)[\lambda_{mc}+x_{mc}]} \tag{16}$$

where

$$x_{mc} = \sum_{c=1}^{C} \bar{q}_c w^-(c) \tag{17}$$

we have:

$y_{mc} = \zeta(x_{mc})$.

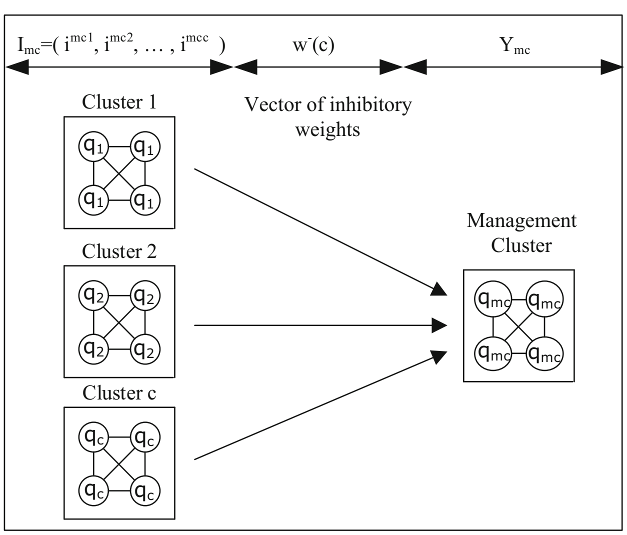


Fig. 5 The random neural network with a management cluster

3.6 Genetic learning algorithm model

The proposed genetic learning algorithm on this article is an autoencoder based on the extreme learning machine (ELM) presented by Huang et al. [17–21] for single-layer feedforward networks (SLFN). For N arbitrary distinct samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots x_{in}]^T \in R^n$ and $t_i = [t_{i1}, t_{i2}, \ldots t_{im}]^T \in R^m$, an standard SLFN with N' hidden nodes and activation function $g(x)$ is mathematically modeled as:

$$f_{N'}(x_j) = \sum_{i=1}^{N'} \beta_i g_i(x_j) = \sum_{i=1}^{N'} \beta_i g_i(w_i \cdot x_j + b_i) = t_j \quad \text{for } j = 1,\ldots N \tag{18}$$

where $w_i = [w_{i1}, w_{i2}, \ldots w_{in}]^T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, b_i is the threshold of the ith hidden node and $g(x)$ activation function of hidden nodes. The above N equations can be written as:

$$h(x)\beta = t_j \qquad H\beta = T \tag{19}$$

where $T = [t_{i1}, t_{i2}, \ldots t_{im}]^T$ are the target outputs and $H = [g_i(w_1 \cdot x_1 + b_1), g_i(w_2 \cdot x_2 + b_2), \ldots g_i(w_{n'} \cdot x_{n'} + b_{n'})]^T$. The output weights β can be calculated by Eq. 6:

$$\beta = H^{\dagger} T \tag{20}$$

where $H^{\dagger}$ is the Moore–Penrose generalized inverse of matrix H.

Extreme learning machine [17–21] proves that the input weights and hidden layer biases of SLFNs can be randomly assigned if the activation functions in the hidden layer are infinitely differentiable. In addition, SLFNs can be considered as a linear system where the output weights can be analytically determined through simple generalized inverse operation of the hidden layer output matrices.

The proposed genetic learning algorithm is based on an ELM auto encoder that models the genome as it codes the replica of the organism that contains it. It consists of two instances of the network described in Sect. 3.4. Network 1 is formed of U input neurons and C clusters, and network 2

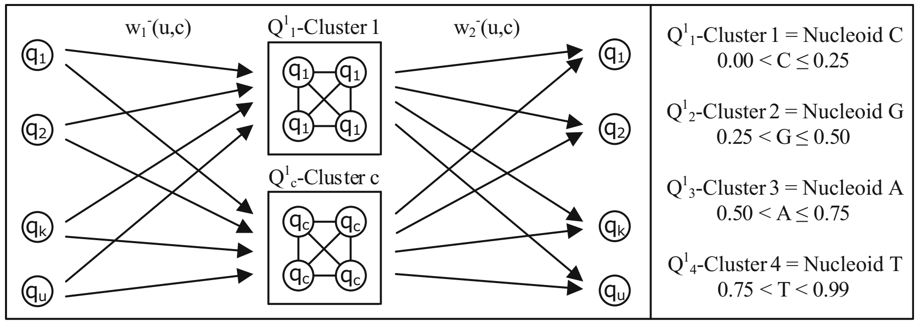


Fig. 6 Genetic learning algorithm

has C input neurons and U clusters (Fig. 6). The organism is represented as a set of data X which is a U vector $X \in [0,1]^U$. The genetic learning algorithm fixes C to 4 neurons that represent the four different nucleoids G, C, A and T and it also fixes W_1 to generate four different types of neurons rather than random values as proposed by the ELM theory. The operational complexity of the proposed algorithm is $O(n^2)$.

Network 1 encodes the organism, and it is defined as:

- $q_1 = (q_1^1, q_2^1, \ldots, q_u^1)$, a U-dimensional vector $q_1 \in [0,1]^U$ that represents the input state q_u for neuron u;
- W_1 is the $U \times C$ matrix of weights $w_1^-(u,c)$ from the U input neurons to the neurons in each of the C clusters;
- $Q^1 = (Q_1^1, Q_2^1, \ldots, Q_c^1)$, a C-dimensional vector $Q^1 \in [0,1]^C$ that represents state q_c for the cluster c where $Q^1 = \zeta(W_1 X)$.

Network 2 decodes the genome, as the pseudoinverse of Network 1, it is defined as:

- $q_2 = (q_1^2, q_2^2, \ldots, q_c^2)$, a C-dimensional vector $q_2 \in [0,1]^C$ that represents the input state q_c for neuron c with the same value as $Q^1 = (Q_1^1, Q_2^1, \ldots, Q_c^1)$;
- W_2 is the $C \times U$ matrix of weights $w_2^-(c,u)$ from the C input neurons to the neurons in each of the U cells;
- $Q^2 = (q_1^2, q_2^2, \ldots, q_u^2)$, a U-dimensional vector $Q^2 \in [0,1]^U$ that represents the state q_u for the cell u where $Q^2 = \zeta(W_2 Q^1)$ or $Q^2 = \zeta(W_2\, \zeta(XW_1))$.

The learning algorithm is the adjustment of W_1 to code the organism X into the four different neurons or nucleoids and then calculate W_2 so that resulting decoded organism Q_2 is the same as the encoded organism X:

$$\min\|X - \zeta(W_2\zeta(XW_1))\| \text{s.t. } W_1 \geq 0 (W_1 \text{positive definite}) \tag{21}$$

Following the extreme learning machine model, W_2 is calculated as:

$$\zeta(XW_1)W_2 = X \tag{22}$$

we have:

$$W_2 = \text{pinv}(\zeta(XW_1))X \tag{23}$$

where pinv is the Moore–Penrose pseudoinverse:

$\text{pinv}(x) = \left(x^T x\right)x^T$.

4 Management decision structure: smart investment

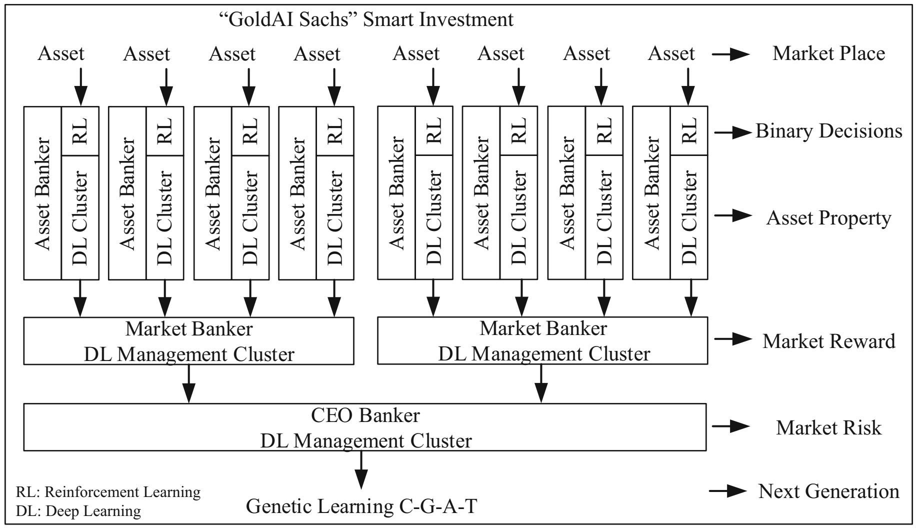
The management decision structure combines in a hierarchal way the four presented different learnings: reinforcement learning, deep learning, deep learning management clusters and genetic learning (Fig. 7). This approach enables structured decisions based on shared information where different learnings are specialized in a decision area. Final decisions are taken collaboratively to achieve a bigger reward that includes the entire decision-making chain.

The smart investment model, named “GoldAI Sachs,” is formed of clusters of intelligent bankers that take local fast binary decisions “buy or sell” on a specific asset based on reinforcement learning (RL) through the interactions and adaptations with the environment where the reward is the profit made. Each asset banker has an associated deep learning (DL) cluster that learns asset identity such as price and reward.

Asset bankers are dynamically clustered to different properties such as investment reward, risk or market type being managed by a market banker deep learning management cluster that selects the best performing asset bankers. The market bankers specialize in learning which asset banker will take the best decision, rather than directly the asset properties.

Finally, a CEO banker deep learning management cluster manages the different market bankers and takes the final structured investment decisions based on the market reward and associated risk prioritizing markets that generate more reward at a lower risk, as any banker would do. The CEO banker genetic algorithm provides immortality: the entire subject’s information, defined as the combination of memory, identity and decision data, is never lost but transmitted to future generations.

Fig. 7 Management decision structure: smart investment



4.1 Asset banker reinforcement learning

The asset banker reinforcement learning algorithm used in the proposed model takes only two fast binary local asset investment decisions. Each intelligent banker is formed of two interconnected neurons where the investment decision is taken according to the neuron that has the maximum potential.

"GoldAI Sachs" asset banker reinforcement learning is defined as:

- q_0, neuron 0 for a buy decision
- q_1, neuron 1 for a sell decision

The reward R is based on the economic profit that the asset bankers achieve with the decisions they make, successive measured values of the R are denoted by R_l, l = 1,2…; these are used to compute the predicted reward:

$$PR_l = \alpha PR_{l-1} + (1-\alpha)R_l \tag{24}$$

where α represents the investment reward memory that can be statically assigned or dynamically updated based on the external observations.

If the observed measured reward is greater than the associated predicted reward; reinforcement learning rewards the decision taken by increasing the network weight that point to it; otherwise; it penalizes it:

if $R_1 > PR_1$:

$$\begin{aligned} &\text{Reward Buy decision: } w_{10}^{+} = w_{10}^{+} + |R| \\ &\text{or Reward Sell decision: } w_{01}^{+} = w_{01}^{+} + |R| \end{aligned} \tag{25}$$

Otherwise, if $R_1 < PR_1$:

$$\begin{aligned} &\text{Penalise Buy decision: } w_{10}^{-} = w_{10}^{-} + |R| \\ &\text{or Penalise Sell decision: } w_{01}^{-} = w_{01}^{-} + |R| \end{aligned} \tag{26}$$

In addition to the reinforcement leaning, asset bankers make prediction price assets (PPA) based on the previous prediction (PAP) and current price asset (CPA):

$$\text{PPA}_l = \gamma \text{PAP}_{l-1} + (1-\gamma)\text{CPA}_l \tag{27}$$

where γ represents the asset price prediction memory.

4.2 Asset banker deep learning cluster

Deep learning is used to learn key investment values that generate asset identity. The smart investment model assigns a deep learning cluster per asset banker. Each different deep learning cluster learns the asset reward or profit prediction, the asset price and the asset price prediction.

"GoldAI Sachs" groups asset bankers dynamically into market sectors according to their risk, profit or type. The model defines a set of x asset banker deep learning clusters as:

- $I_{\text{Banker-}x} = (i_1^{\text{Banker-}x}, i_2^{\text{Banker-}x}, \ldots, i_u^{\text{Banker-}x})$ a U-dimensional vector where $i_1^{\text{Banker-}x}$, $i_2^{\text{Banker-}x}$, and $i_u^{\text{Banker-}x}$ are the same banker number x;
- $w_{\text{Banker-x}}^{-}(u, c)$ is the $U \times C$ matrix of weights of the deep learning cluster for banker x;
- $Y_{\text{Banker-}x} = (y_1^{\text{Banker-}x}, y_2^{\text{Banker-}x}, \ldots, y_c^{\text{Banker-}x})$ a C-dimensional vector where $y_1^{\text{Banker-}x}$ is the reinforcement learning reward prediction, $y_2^{\text{Banker-}x}$ is the dynamic reward prediction, $y_3^{\text{Banker-}x}$ is the transaction price, and $y_c^{\text{Banker-}x}$ is the price prediction for banker number x.

4.3 Market banker deep learning management cluster

The market banker deep learning management cluster analyzes the predicted reward from its respective asset banker deep learning clusters, prioritizes their values based on local market knowledge and finally reports to the CEO banker deep learning management cluster the total predicted profit that its market can make.

"GoldAI Sachs" model defines a set of x market banker deep learning management clusters as:

- $I_{\text{MarketBanker-}x}$, a C-dimensional vector $I_{\text{MarketBanker}-x} \in [0,1]^C$ with the values of the predicted rewards from asset banker x;
- $w\bar{}_{\text{MarketBanker-}x}$ (c) is the C-dimensional vector of weights that represents the priority of each asset banker x;
- $Y_{\text{MarketBanker-}x}$, a scalar $Y_{\text{MarketBanker}-x} \in [0,1]$ that represents the predicted profit the market banker deep learning management cluster can make.

4.4 CEO banker deep learning management cluster

The CEO banker deep learning management cluster, "AI Morgan" takes the final investment management decision based on the inputs from the market bankers and the associated. The CEO banker selects the markets that can generate better reward at lower risk where the maximum risk is defined as β by the "GoldAI Sachs" board of directors, or other user, where the higher the value the higher risk. β can be statically assigned or dynamically updated based on the external observations.

"GoldAI Sachs" model defines the CEO banker deep learning management cluster:

- $I_{\text{CEO-Banker}}$, a X-dimensional vector $I_{\text{CEO}-\text{Banker}} \in [0,1]^X$ with the values of the set x banker DL management clusters;
- $w\bar{}_{\text{CEO-Banker}}$ (c) is the C-dimensional vector of weights that represents the risk associated to each market;
- $Y_{\text{CEO-Banker}}$, a scalar $Y_{\text{CEO}-\text{Banker}} \in [0,1]$ that represents the final investment decision.

4.5 CEO banker genetic algorithm

Genetic learning transmits entirely the knowledge acquired from the CEO banker, defined as the combination of memory, identity and decision data, to future banker generations when "AI Morgan" considers itself no longer valid due energy limitations or cybersecurity attacks. Because the CEO banker information is never lost but transmitted to future generations, genetic algorithm provides immortality rather than reinforcement learning applied in fast local decisions or deep learning in identity (Fig. 8).

"GoldAI Sachs" model defines genetic learning as an autoencoder where:

- $X_{\text{Genetic}} = (x_1^{\text{Genetic}}, x_2^{\text{Genetic}}, \ldots, x_u^{\text{Gentetic}})$ a U-dimensional vector where x_1^{Genetic}, x_2^{Genetic}, and x_u^{Genetic} are outputs of the x banker deep learning clusters;
- $w\bar{}_{\text{Genetic}-1}$ (u, c) is the $U \times C$ matrix of weights of the genetic encoder;
- $Y_{\text{Nucleoid}} = (y_1^{\text{Nucleoid}}, y_2^{\text{Nucleoid}}, \ldots, y_c^{\text{Nucleoid}})$ a C-dimensional vector where $y_1^{\text{Nucleoid}}, \ldots, y_c^{\text{Nucleoid}}$ is the value of the nucleoid
- $w\bar{}_{\text{Genetic}-2}(c, u)$ is the $C \times U$ matrix of weights of the genetic decoder;
- $Y_{\text{Generation}} = (y_1^{\text{Generation}}, y_2^{\text{Generation}}, \ldots, y_c^{\text{Generation}})$ a C-dimensional vector where $y_1^{\text{Generation}}, \ldots, y_c^{\text{Generation}}$ is the value of the new banker generation.

5 Smart investment implementation

"GoldAI Sachs" is implemented in Java with eight asset bankers clustered in two different markets; bond market is low reward and therefore low risk, whereas risk market is high reward and high risk.

5.1 Asset banker reinforcement learning and deep learning clusters

Each asset banker has a two-node reinforcement learning algorithm to make local fast decisions "buy or sell" trade on different assets. the banker DL cluster learns the asset properties to create asset identity; it has four inputs cells ($u = 4$) and four output clusters ($c = 4$) where the input cell is the normalized value of the x asset banker identification: x ($i_1^{\text{Banker-}x} = 0{\cdot}x$, $i_2^{\text{Banker-}x} = 0{\cdot}x$, $i_3^{\text{Banker-}x} = 0{\cdot}x$, $i_4^{\text{Banker-}x\text{-}} = 0{\cdot}x$) and the output clusters are the normalized asset properties: ($y^{\text{1 Banker-}x}$ = reinforcement learning reward prediction, $y_2^{\text{Banker-}x}$ = dynamic reward prediction, $y_3^{\text{Banker-}x\text{-}}$ = transaction price, $y^{\text{4 Banker-}x}$ = price prediction as shown in Table 1. "GoldAI Sachs" normalizes the DL clusters; the inputs to: $x/10$ and the outputs to (reward or price/100) + 0.5, respectively, to keep the learning algorithm within the stable region.

5.2 Market banker deep learning management clusters

"GoldAI Sachs" has two market banker DL management clusters for the bond and commodities market, respectively. The inputs of the market bankers are the predicted asset banker rewards, whereas the output is the predicted reward the market can make. The network weighs are the asset banker priority; i.e., the best asset banker only to be

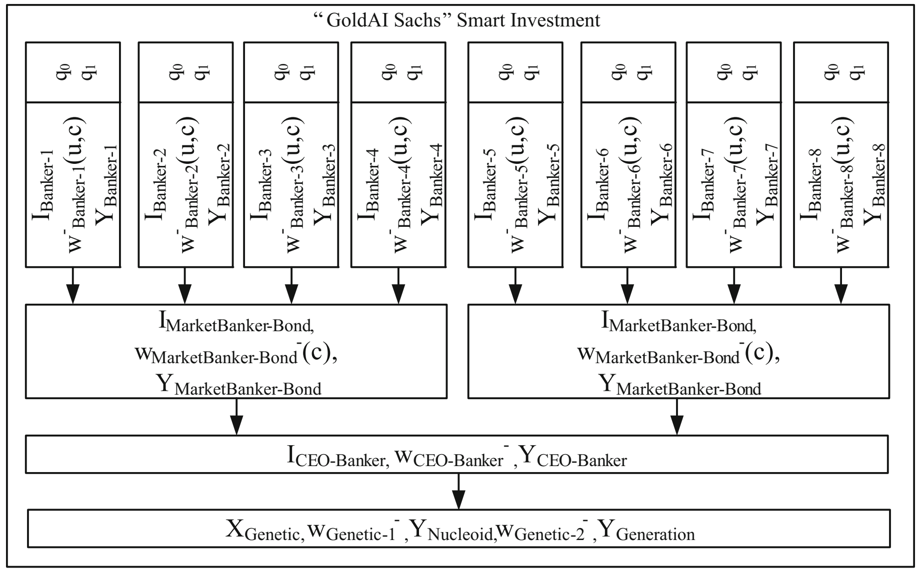


Fig. 8 "GoldAI Sachs" smart investment model definition

Table 1 Asset banker DL cluster implementation

Cluster	Input	Value	Output	Value
Banker 1	$i_1^{\text{Banker}-1}$	0.1	$y_1^{\text{Banker}-1}$	Reinforcement learning reward prediction
Banker 1	$i_2^{\text{Banker}-1}$	0.1	$y_2^{\text{Banker}-1}$	Dynamic reward prediction
Banker 1	$i_3^{\text{Banker}-1}$	0.1	$y_3^{\text{Banker}-1}$	Transaction price
Banker 1	$i_4^{\text{Banker}-1}$	0.1	$y_4^{\text{Banker}-1}$	Price prediction
…	…	…	…	…
Banker 8	$i_1^{\text{Banker}-8}$	0.8	$y_1^{\text{Banker}-8}$	Reinforcement learning reward prediction
Banker 8	$i_1^{\text{Banker}-8}$	0.8	$y_2^{\text{Banker}-8}$	Dynamic reward prediction
Banker 8	$i_1^{\text{Banker}-8}$	0.8	$y_3^{\text{Banker}-8}$	Transaction price
Banker 8	$i_1^{\text{Banker}-8}$	0.8	$y_4^{\text{Banker}-8}$	Price prediction

considered with the maximum banker set at 1.0 and others at 0.0 or the four Bankers with the same priority; with all the network weights set at 0.25 as shown in Table 2.

5.3 CEO banker deep learning management clusters

The input of the CEO banker deep learning management cluster is the market profit provided by the market bankers and its network weight is the risk associated to each market. The smart investment model considers that market risk level is related to the reward the market can generate. The CEO banker, "AI Morgan" starts gradually assessing the reward increasing the risk β from 0.1 up to the maximum risk limit permissible by "GoldAI Sachs" owner or manager. "AI Morgan" then takes the final decision based on a greater predicted reward at the lowest risk β as represented in Table 3.

5.4 CEO banker genetic algorithm

Genetic learning algorithm is an autoencoder where the Market's properties, identity and decisions learned by the banker deep learning clusters are codified into four neurons that represent the 4 nucleoids of the Genome: (cytosine [C], guanine [G], adenine [A] or thymine [T]). These values are transmitted to the next generation in a daily basis.

The input X_{Genetic} to the genetic algorithm corresponds to a 32-dimensional vector where x_1^{Genetic}, x_2^{Genetic}, … x_{32}^{Genetic} correspond to the 4 outputs of the 8 asset bankers $Y_{\text{Banker-}x}$ $Y_{\text{Gene a}}$ is 4-dimensional vector where $y_1^{\text{Nucleoid,}}$ …, y_4^{Nucleoid} is the value of the four different nucleoids. The output of the genetic algorithm $Y_{\text{Generation}}$ corresponds to the input X_{Genetic} as shown in Table 4.

Table 2 Market banker DL management cluster implementation

Cluster	Input $I_{\text{BondMarketBanker}}$	Network weights $w^{-}_{\text{BondMarketBanker}}(c)$	Output $Y_{\text{BondMarketBanker}}$
Bond	Predicted reward banker 1	0.0	Best predicted reward bond banker
	Predicted reward banker 2	1.0	
	Predicted reward banker 3	0.0	
	Predicted reward banker 4	0.0	
Cluster	Input $I_{\text{DerivativeMarketBanker}}$	Network weights $w^{-}_{\text{DerivativeMarketBanker}}(c)$	Output $Y_{\text{DerivativeMarketBanker}}$
Derivative	Predicted reward banker 5	0.0	Best predicted reward derivative banker
	Predicted reward banker 6	1.0	
	Predicted reward banker 7	0.0	
	Predicted reward banker 8	0.0	

Table 3 CEO banker DL management cluster implementation

Cluster	Input $I_{\text{CEO-Banker}}$	Network weight $w^{-}_{\text{CEO-Banker}}$ (c)	Output $Y_{\text{CEO-Banker}}$
CEO banker	$Y_{\text{BondMarketBanker}}$	$1-\beta$	Predicted reward at risk β
	$Y_{\text{DerivativeMarketBanker}}$	β	

6 Smart investment experimental results

"GoldAI Sachs" is evaluated with eight different assets to assess the adaptability and performance of our proposed smart investment solution for 11 days. The assets are split into the bond market with low risk and slow reward and the derivative market with high risk and fast reward (Fig. 9). Experiments are carried out with reinforcement learning first initialized with a buy decision.

6.1 Asset banker reinforcement learning validation

The profit that each asset banker makes when buying or selling 100 assets for 11 days with the maximum profit, the ratio between both the number of winning decisions against the losing ones and the number of buy decisions against the sell, is shown in Tables 5 and 6. The validation covers three different values of investment reward memory α.

The simulation results are almost independent from the value of the investment reward memory α, this is due the reduced complexity in asset price variation. The reinforcement learning algorithm adapts very quickly to variable asset prices where asset 6 shows that the lowest investment memory, $\alpha = 0.1$, is the optimum value. The profit made in assets that start downward such as asset 2, asset 4, asset 6 and asset 8 is worse than the upward ones because the asset bankers are initialized with a buy decision.

6.2 Asset banker deep learning cluster validation

The asset banker deep learning cluster validation for the eight different assets during the 11 different days is shown in Table 7. The learning algorithm error threshold is set at 1.0E−25. The first value is the final iteration number for learning algorithm number, and the second is the normalized error at 1.0E−26.

The learning algorithm of the deep learning clusters is very stable; it achieves a minimum error 1.0E−25 with very reduced iterations.

6.3 Market banker deep learning management cluster validation

The profits generated by the market bankers are shown in Tables 8 and 9 and Figs. 10 and 11. The market bankers take market decisions rather than individual asset decisions form the asset bankers. Market bankers invest 400 assets which is the combination of the four asset bankers purchasing power. The combined profits that the asset bankers can make independently and the profits the bond market manager can obtain are presented; in addition, the maximum values are also shown.

The addition of a specialized market banker deep learning management cluster increases the profits almost to the maximum value. Tables 10 and 11 represent the bond and derivative market bankers deep learning management clusters values.

Table 4 Genetic algorithm implementation

Name	Variable	Value
X_{Genetic}	x_1^{Genetic}	$y_1^{\text{Banker}-1}$
	x_2^{Genetic}	$y_2^{\text{Banker}-1}$
	x_3^{Genetic}	$y_3^{\text{Banker}-1}$
	x_4^{Genetic}	$y_4^{\text{Banker}-1}$
	x_5^{Genetic}	$y_4^{\text{Banker}-2}$
	…	…
	x_8^{Genetic}	$y_4^{\text{Banker}-2}$
	…	…
	x_{32}^{Gentetic}	$y_4^{\text{Banker}-8}$
$w^{-}_{\text{Genetic}-1}(u,c)$	$w^{-}_{\text{Genetic-1}}(1, 1)$ … $w^{-}_{\text{Genetic-1}}(32, 1)$	0.02, … 0.02
	$w^{-}_{\text{Genetic-1}}(1, 2)$ … $w^{-}_{\text{Genetic-1}}(32, 2)$	0.15, … 0.15
	$w^{-}_{\text{Genetic-1}}(1, 3)$ … $w^{-}_{\text{Genetic-1}}(32, 3)$	0.40, … 0.40
	$w^{-}_{\text{Genetic-1}}(1, 4)$ … $w^{-}_{\text{Genetic-1}}(32, 4)$	0.99, … 0.99
Y_{Nucleoid}	y_1^{Nucleoid} $(0.00 < C \leq 0.25$	0.2048
	y_2^{Nucleoid} $0.25 < G \leq 0.50$	0.3900
	y_3^{Nucleoid} $0.50 < A \leq 0.75$	0.6295
	y_4^{Nucleoid} $0.75 < T < 0.99)$	0.9268
$w^{-}_{\text{Genetic-2}}(c,u)$	$w^{-}_{\text{Genetic-2}}(1, 1)$ … $w^{-}_{\text{Genetic-2}}(1, 32)$	$\text{pinv}(Y_{\text{Nucleiod}})X_{\text{Genetic}}$
	$w^{-}_{\text{Genetic-2}}(2, 1)$ … $w^{-}_{\text{Genetic-2}}(2, 32)$	
	$w^{-}_{\text{Genetic-2}}(3, 1)$ … $w^{-}_{\text{Genetic-2}}(3, 32)$	
	$w^{-}_{\text{Genetic-2}}(4, 1)$ … $w^{-}_{\text{Genetic-2}}(4, 32)$	
$Y_{\text{Generation}}$	$y_1^{\text{Generation}}$	x_1^{Genetic}
	$y_2^{\text{Generation}}$	x_2^{Genetic}
	$y_3^{\text{Generation}}$	x_3^{Genetic}
	$y_4^{\text{Generation}}$	x_4^{Genetic}
	$y_5^{\text{Generation}}$	x_5^{Genetic}
	…	…
	$y_8^{\text{Generation}}$	x_8^{Genetic}
	…	…
	$y_{32}^{\text{Generation}}$	x_{32}^{Gentetic}

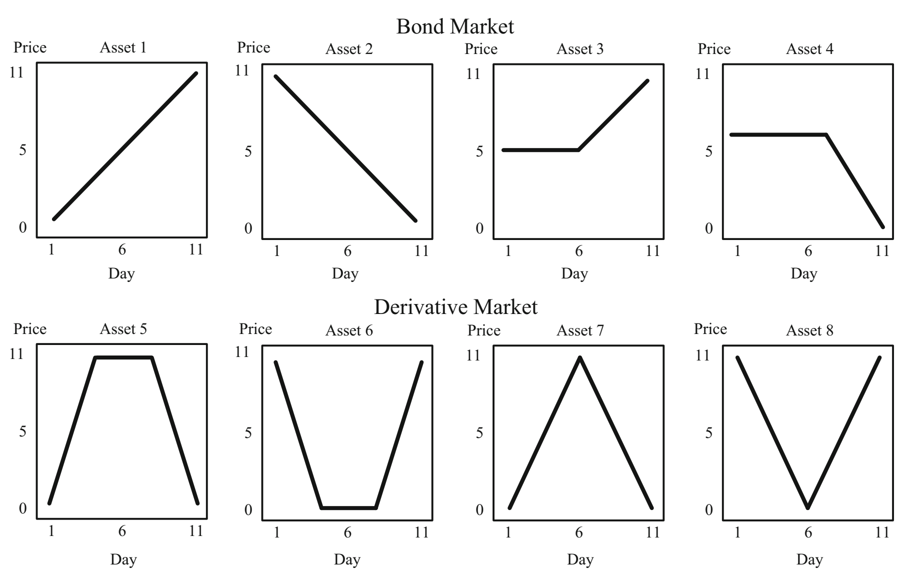


Fig. 9 Smart investment data model

Table 5 Asset banker reinforcement learning validation; α = 0.1

Assets	Profit	Maximum profit	Ratio	Win	Loss	Buy	Sell
1	1000	1000	1.00	10	0	10	0
2	800	1000	0.80	9	1	1	9
3	600	600	1.00	6	0	10	0
4	300	500	0.60	4	1	6	4
5	2000	2000	1.00	8	0	4	6
6	1200	2000	0.60	6	2	4	6
7	1600	2000	0.80	9	1	6	4
8	800	2000	0.40	7	3	4	6

6.4 CEO banker deep learning management cluster validation

The CEO banker, "AI Morgan," profited at different risks ratios with a total of investment of 800 assets is represented in Table 12. A low risk value $\beta = 0.2$ represents 640 assets in the bond market (B) and 160 is the derivative market (D), whereas a high risk value $\beta = 0.8$ is 160 assets in the bond market and 640 in the derivative market, respectively.

The more risk the CEO banker "AI Morgan" takes, the more profit is able to generate as the investment decisions are directed to the derivative market reaching nearly optimum values. Table 13 and Fig. 12 represent the CEO

Table 6 Asset banker reinforcement learning validation; α = 0.5 and α = 0.9

Asset	Profit	Maximum profit	Ratio	Win	Loss	Buy	Sell
1	1000	1000	1.00	10	0	10	0
2	800	1000	0.80	9	1	1	9
3	600	600	1.00	6	0	10	0
4	300	500	0.60	4	1	6	4
5 (α = 0.5)	2000	2000	1.00	8	0	4	6
5 (α = 0.9)	1600	2000	0.80	7	1	7	3
6	600	2000	0.30	5	3	3	7
7	1600	2000	0.80	9	1	6	4
8	800	2000	0.40	7	3	4	6

Table 7 Asset banker deep learning cluster validation

Day	Asset 1	Asset 2	Asset 3	Asset 4	Asset 5	Asset 6	Asset 7	Asset 8
2	It: 1725	It: 983	It: 663	It: 519	It: 389	It: 416	It: 302	It: 337
	E: 9.87	E: 9.70	E: 9.48	E: 9.38	E: 9.51	E: 9.79	E: 6.59	E: 9.92
3	It: 1720	It: 889	It: 663	It: 519	It: 367	It: 324	It: 300	It: 277
	E: 9.73	E: 9.74	E: 9.48	E: 9.38	E: 9.86	E: 8.10	E: 7.90	E: 8.81
4	It: 1720	It: 886	It: 663	It: 519	It: 366	It: 321	It: 299	It: 273
	E: 9.77	E: 9.47	E: 9.48	E: 9.38	E: 9.95	E: 9.00	E: 9.81	E: 9.70
5	It: 1718	It: 884	It: 663	It: 519	It: 386	It: 352	It: 300	It: 273
	E: 9.93	E: 9.69	E: 9.48	E: 9.38	E: 7.83	E: 9.80	E: 7.57	E: 9.53
6	It: 1718	It: 884	It: 628	It: 519	It: 429	It: 373	It: 300	It: 287
	E: 9.97	E: 9.68	E: 9.90	E: 9.38	E: 9.00	E: 8.38	E: 7.48	E: 7.51
7	It: 1720	It: 884	It: 627	It: 547	It: 431	It: 374	It: 370	It: 336
	E: 9.83	E: 9.16	E: 9.10	E: 9.26	E: 9.81	E: 8.78	E: 9.33	E: 9.77
8	It: 1720	It: 884	It: 627	It: 494	It: 388	It: 409	It: 303	It: 335
	E: 9.42	E: 9.49	E: 9.46	E: 8.82	E: 9.89	E: 9.50	E: 9.11	E: 8.75
9	It: 1720	It: 884	It: 627	It: 491	It: 367	It: 324	It: 300	It: 277
	E: 9.41	E: 9.57	E: 9.17	E: 8.95	E: 9.77	E: 8.55	E: 8.38	E: 9.54
10	It: 1719	It: 886	It: 627	It: 491	It: 369	It: 319	It: 299	It: 274
	E: 9.89	E: 9.11	E: 9.12	E: 8.29	E: 9.94	E: 8.34	E: 9.87	E: 7.56
11	It: 1720	It: 896	It: 626	It: 495	It: 407	It: 335	It: 315	It: 273
	E: 9.74	E: 8.21	E: 9.55	E: 9.99	E: 8.76	E: 7.38	E: 8.07	E: 9.83

Table 8 Bond market banker profits; α = 0.1

Day	Total asset banker	Bond market banker	I (%)	Maximum asset banker	Maximum market banker	I (%)
2	0	400	400.00	200	400	100.00
3	200	400	100.00	200	400	100.00
4	200	400	100.00	200	400	100.00
5	200	400	100.00	200	400	100.00
6	300	400	33.33	300	400	33.33
7	200	400	100.00	400	400	0.00
8	400	400	0.00	400	400	0.00
9	400	400	0.00	400	400	0.00
10	400	400	0.00	400	400	0.00
11	400	400	0.00	400	400	0.00
Total	2700	4000	48.15	3100	4000	29.03

Table 9 Derivative market banker profits; α = 0.1

Day	Total asset banker	Derivative market banker	I (%)	Maximum asset banker	Maximum market banker	I (%)
2	0	800.0	400.00	800	800.0	400.00
3	1000	1200	20.00	1000	1200	20.00
4	1000	1200	20.00	1000	1200	20.00
5	800	800	0.00	800	800	0.00
6	400	0	− 100.00	400	0	− 100.00
7	− 400	− 800	100.00	400	800	100.00
8	0	800	800	800	800	0.00
9	1000	1200	20.00	1000	1200	20.00
10	1000	1200	20.00	1000	1200	20.00
11	800	800	0.00	800	800	0.00
Total	5600	7200	28.57	8000	8800	10.00

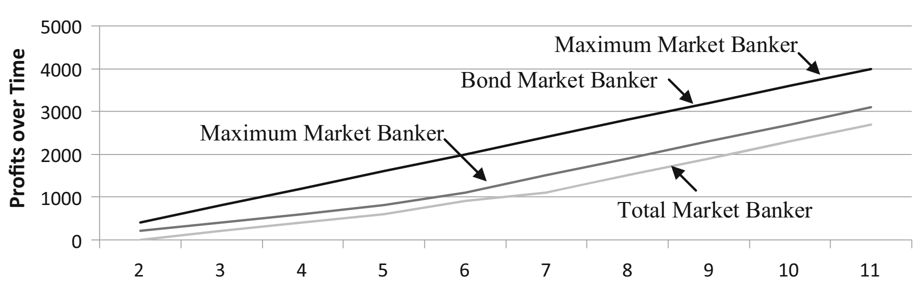


Fig. 10 Bond market banker accumulative profits; α = 0.1

banker deep learning management clusters values at different risks with the final risk decision.

6.5 CEO banker genetic algorithm validation

The genetic algorithm validation for the four different nucleoids (C, G, A, T) during the 10 different days is shown in Tables 14 and 15 with the genetic algorithm error.

The genetic algorithm codifies the CEO banker and transmits this information to the next generations with a residual error with only one iteration at a minimum time.

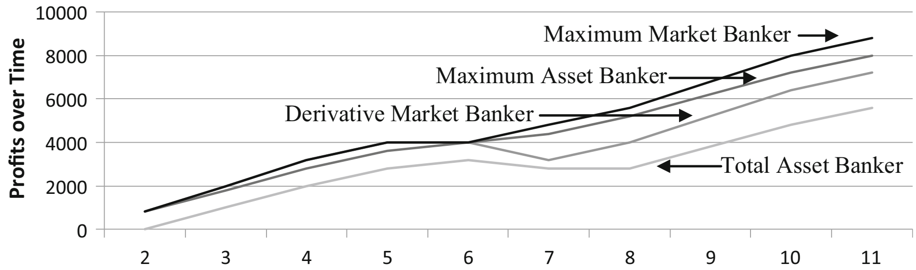


Fig. 11 Derivative market banker accumulative profits; $\alpha = 0.1$

Table 10 Bond market banker deep learning management cluster validation

Day	$I_{\text{BondMarketBanker}}$				$\bar{w}_{\text{BondMarketBanker}}(c)$				$Y_{\text{BondMarketBanker}}$
2	0.0000	0.0000	0.0000	0.0000	1.0	0.0	0.0	0.0	0.9994
3	0.5090	0.4910	0.5000	0.5000	1.0	0.0	0.0	0.0	0.5146
4	0.5100	0.5081	0.5000	0.5000	1.0	0.0	0.0	0.0	0.5142
5	0.5100	0.5098	0.5000	0.5000	1.0	0.0	0.0	0.0	0.5142
6	0.5100	0.5100	0.5000	0.5000	1.0	0.0	0.0	0.0	0.5142
7	0.5100	0.5100	0.5090	0.5000	1.0	0.0	0.0	0.0	0.5142
8	0.5100	0.5100	0.5099	0.4910	1.0	0.0	0.0	0.0	0.5142
9	0.5100	0.5100	0.5100	0.5081	1.0	0.0	0.0	0.0	0.5142
10	0.5100	0.5100	0.5100	0.5098	1.0	0.0	0.0	0.0	0.5142
11	0.5100	0.5100	0.5100	0.5100	1.0	0.0	0.0	0.0	0.5142

Table 11 Derivative market banker deep learning management cluster validation

Day	$I_{\text{DerivativeMarketBanker}}$				$\bar{w}_{\text{DerivativeMarketBanker}}(c)$				$Y_{\text{DerivativeMarketBanker}}$
2	0.0000	0.0000	0.0000	0.0000	1.0	0.0	0.0	0.0	0.9994
3	0.5180	0.4820	0.5180	0.4820	1.0	0.0	0.0	0.0	0.5103
4	0.5288	0.5252	0.5198	0.5162	1.0	0.0	0.0	0.0	0.5051
5	0.5299	0.5295	0.5200	0.5196	1.0	0.0	0.0	0.0	0.5046
6	0.5210	0.5210	0.5200	0.5200	1.0	0.0	0.0	0.0	0.5088
7	0.5021	0.5021	0.5200	0.5200	0.0	0.0	1.0	0.0	0.5093
8	0.5002	0.5002	0.4840	0.4840	1.0	0.0	0.0	0.0	0.5190
9	0.5180	0.4820	0.5164	0.4804	1.0	0.0	0.0	0.0	0.5103
10	0.5288	0.5252	0.5196	0.5160	1.0	0.0	0.0	0.0	0.5051
11	0.5299	0.5295	0.5200	0.5196	1.0	0.0	0.0	0.0	0.5046

Table 12 CEO banker profits; $\alpha = 0.1$

Day	Risk $\beta = 0.2$			Risk $\beta = 0.5$			Risk $\beta = 0.8$			Max profit
	B	*D*	Total	*B*	*D*	Total	*B*	*D*	Total	
2	640	320	960	400	800	1200	160	1280	1440	1440
3	640	480	1120	400	1200	1600	160	1920	2080	2080
4	640	480	1120	400	1200	1600	160	1920	2080	2080
5	640	320	960	400	800	2800	160	1280	1440	1440
6	640	0	640	400	0	400	160	0	160	640
7	640	− 320	320	400	− 800	− 400	160	− 1280	− 1120	320
8	640	320	960	400	800	1200	160	1280	1440	1440
9	640	480	1120	400	1200	1600	160	1920	2080	2080
10	640	480	1120	400	1200	1600	160	1920	2080	2080
11	640	640	1280	400	800	1200	160	1280	1440	1440
Total	6400	3200	9600	4000	7200	12,800	1600	11,520	13,120	15,040

Table 13 CEO banker deep learning management cluster validation

Day	$I_{CEO\text{-}Banker}$		Risk β = 0.2		Risk β = 0.5		Risk β = 0.8	
			$Y_{CEO\text{-}Banker}$	D	$Y_{CEO\text{-}Banker}$	D	$Y_{CEO\text{-}Banker}$	D
2	0.9994	0.9994	0.2127	0.1	0.2127	0.1	0.2127	0.8
3	0.5147	0.5103	0.3444	0.2	0.3450	0.5	0.3456	0.8
4	0.5142	0.5051	0.3450	0.2	0.3462	0.5	0.3475	0.8
5	0.5142	0.5046	0.3451	0.2	0.3464	0.5	0.3477	0.8
6	0.5142	0.5088	0.3447	0.2	0.3454	0.5	0.3461	0.8
7	0.5142	0.5093	0.3447	0.2	0.3453	0.5	0.3460	0.8
8	0.5142	0.5190	0.3441	0.1	0.3441	0.1	0.3441	0.1
9	0.5142	0.5103	0.3446	0.2	0.3451	0.5	0.3456	0.8
10	0.5142	0.5051	0.3451	0.2	0.3463	0.5	0.3475	0.8
11	0.5142	0.5046	0.3451	0.2	0.3464	0.5	0.3477	0.8

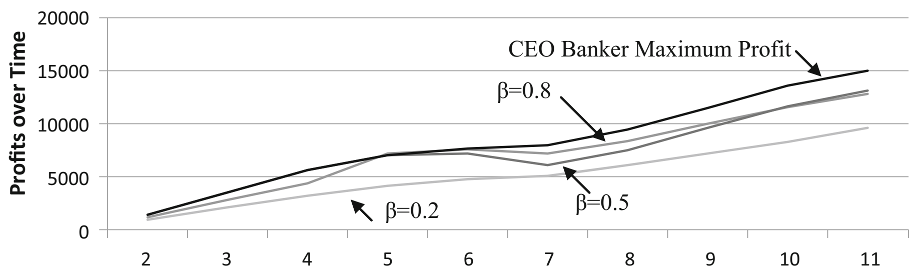


Fig. 12 CEO banker accumulative profits; α = 0.1

Table 14 Genetic algorithm validation

Day	Error	Nucleoid-C	Nucleoid-G	Nucleoid-A	Nucleoid-T
2	3.05E−31	0.2048	0.3893	0.6295	0.9268
3	5.85E−31	0.2026	0.3861	0.6263	0.9259
4	6.78E−32	0.2025	0.3859	0.6262	0.9259
5	1.17E−31	0.2029	0.3865	0.6267	0.9260
6	4.44E−31	0.2033	0.3870	0.6272	0.9262
7	1.29E−31	0.2049	0.3894	0.6296	0.9269
8	3.61E−31	0.2031	0.3868	0.6271	0.9261
9	2.96E−31	0.2021	0.3852	0.6255	0.9257
10	6.90E−31	0.2020	0.3851	0.6254	0.9256
11	1.36E−31	0.2023	0.3856	0.6259	0.9258

Table 15 Genetic algorithm validation

	Error genetic	Iteration	Time (ns)
Value	3.13E−31	1.00	3.17E+05
σ	2.11E−31	0.00	5.75E+04
95% CR	1.31E−31	0.00	3.56E+04

7 Cryptocurrency evaluation

"GoldAI Sachs" is evaluated with seven different assets to assess the adaptability and performance of our proposed smart investment solution for 664 days, from 07/08/2015 to 31/05/2017. The assets are split into the Bitcoin exchange market (BITSTAMP, BTCE, COINBASE, KRAKEN) and the currency market (Bitcoin, Ethereum, Ripple). Reinforcement learning is first initialized with a buy decision.

Cryptocurrency evaluation data has been produced though datasheets obtained from Kraggle; only three different currencies were found (Fig. 13). The values of the different assets within the Bitcoin exchange market are very similar as they trade the same currency, whereas the currency market presents a more disperse set of values.

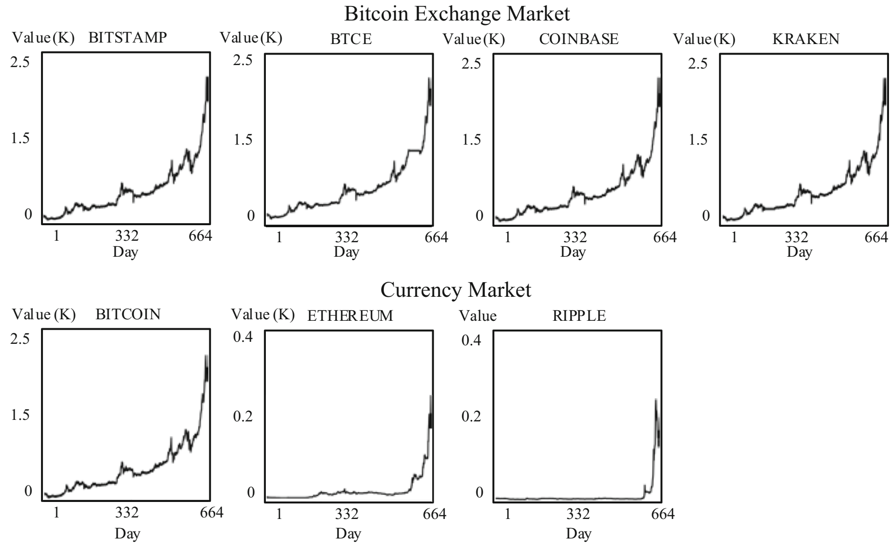


Fig. 13 Cryptocurrency investment data model

7.1 Asset banker reinforcement learning validation

The profit that each asset banker makes when buying or selling 100 assets for 664 days from year 2015 to 2017 with the maximum profit, the ratio between both the number of winning decisions against the losing ones and the number of buy decisions against the sell, is shown in Tables 16, 17 and 18. The validation covers three different values of investment reward memory α.

The RNN reinforcement learning algorithm adapts very quickly to variable asset prices and makes profits; although not to optimum values. The value of the investment reward memory does not have a mayor impact in the overall profit due to the great adaptation of reinforcement learning algorithm though a high value of investment reward memory generates more profits (Fig. 14).

7.2 Asset banker deep learning cluster validation

The asset banker deep learning cluster validation for the seven different assets during the 664 different days from year 2015 to 2017 is shown in Table 19. The learning algorithm error threshold is set at 1.0E−25. The first value is the final iteration number for learning algorithm number, and the second is the normalized error at 1.0E−26.

7.3 Market banker deep learning management cluster validation

The profit the market bankers can make for the 664 days from year 2015 to 2017 is shown in Tables 20, 21 and 22 where the validation includes three different values of investment reward memory α. The market bankers take market decisions rather than individual asset decisions form the asset bankers. Bitcoin exchange market banker invests 400 assets which is the combination of the four asset bankers purchasing power whereas currency market

Table 16 Asset banker reinforcement learning validation; α = 0.1, year 2015–2017

Asset	Profit	Maximum profit	Ratio	Win	Loss	Buy	Sell
BITSTAMP	192,530	957,018	0.20	386	277	597	66
BTCE	177,133	749,307	0.24	355	258	595	68
COINBASE	172,257	985,625	0.17	383	279	503	160
KRAKEN	187,647	977,763	0.19	367	258	619	44
Bitcoin	195,083	952,339	0.20	393	270	634	29
Ethereum	18,626	58,916	0.32	332	322	385	278
Ripple	17	101	0.17	292	370	662	1
Total	943,293	4,681,069	0.20	2508	2034	3995	646

Table 17 Asset banker reinforcement learning validation; α = 0.5, year 2015–2017

Asset	Profit	Maximum profit	Ratio	Win	Loss	Buy	Sell
BITSTAMP	196,226	957,018	0.21	389	274	644	19
BTCE	180,596	749,308	0.24	371	242	659	4
COINBASE	192,951	985,625	0.20	392	270	644	19
KRAKEN	188,704	977,763	0.19	366	259	648	15
Bitcoin	196,059	952,339	0.21	385	278	496	167
Ethereum	19,446	58,916	0.33	355	299	175	488
Ripple	20	101	0.20	293	369	663	0
Total	974,002	4,681,070	0.21	2551	1991	3929	712

Table 18 Asset banker reinforcement learning validation; α = 0.9, year 2015–2017

Asset	Profit	Maximum profit	Ratio	Win	Loss	Buy	Sell
BITSTAMP	196,226	957,018	0.21	389	274	644	19
BTCE	191,875	749,308	0.26	373	240	631	32
COINBASE	192,951	985,625	0.20	392	270	644	19
KRAKEN	187,052	977,763	0.19	359	266	643	20
Bitcoin	202,803	952,339	0.21	394	269	613	50
Ethereum	21,476	58,916	0.36	339	315	432	231
Ripple	20	101	0.20	293	369	663	0
Total	992,403	4,681,070	0.21	2539	2003	4270	371

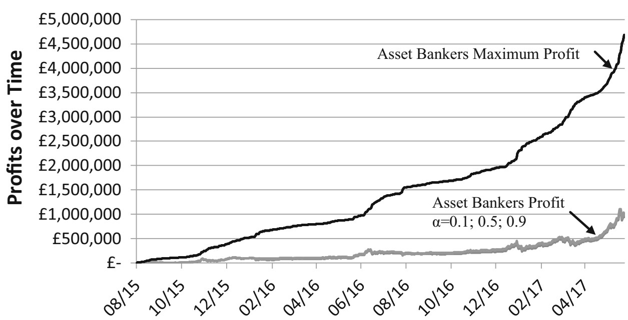


Fig. 14 Asset banker reinforcement learning validation; α = 0.1; α = 0.5; α = 0.9

banker invests 300 assets as there are only three asset bankers.

The Bitcoin exchange market banker does not increase the profits of the market; this is mostly due to the fact that the four Bitcoin exchange asset bankers perform very similarly on almost equal asset conditions therefore with the addition of a market banker, the independent knowledge acquired by each asset banker is lost. However, when the currency asset bankers operate under diverse asset conditions, the addition of a currency market banker increases the market profit due to the right selection of the best performing asset banker. Different values of investment reward memory have an impact on the final profit where a balanced investment reward memory (α = 0.5) between previous and last investment provides optimum results (Figs. 15, 16).

Table 23 represents the bond and derivative market bankers deep learning management clusters average values for the 664 days from year 2015 to 2017.

Table 19 Asset banker deep learning cluster validation

Period	Asset 1	Asset 2	Asset 3	Asset 4	Asset 5	Asset 6	Asset 7
664 days 2015–2017	It: 1854.67 E: 9.60	It: 945.43 E: 9.51	It: 670.24 E: 9.45	It: 524.43 E: 9.34	It: 436.12 E: 9.24	It: 377.75 E: 8.94	It: 336.71 E: 8.71

Table 20 Market banker profits; α = 0.1, year 2015–2017

Market	Total asset banker	Market banker	*I* (%)	Maximum asset banker	Maximum market banker	*I* (%)
Exchange	729,567	541,559	− 25.77	3,669,714	3,731,781	1.69
Currency	213,726	472,323	120.99	1,011,356	2,050,320	102.73

Table 21 Market banker profits; α = 0.5, year 2015–2017

Market	Total asset banker	Market banker	*I* (%)	Maximum asset banker	Maximum market banker	*I* (%)
Exchange	758,477	596,973	− 21.29	3,669,714	3,403,866	− 7.24
Currency	215,525	547,631	154.09	1,011,356	2,003,275	98.08

Table 22 Market banker profits; α = 0.9, year 2015–2017

Market	Total asset banker	Market banker	*I* (%)	Maximum asset banker	Maximum market banker	*I* (%)
Exchange	768,104	427,220	− 44.38	3,669,714	3,335,495	− 9.11
Currency	224,299	531,675	137.04	1,011,356	2,271,457	124.60

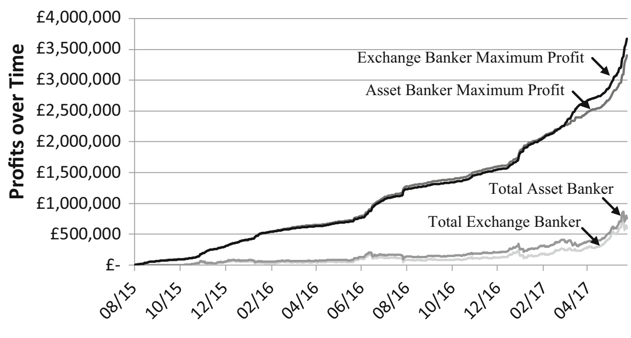


Fig. 15 Exchange market banker profits; α = 0.5

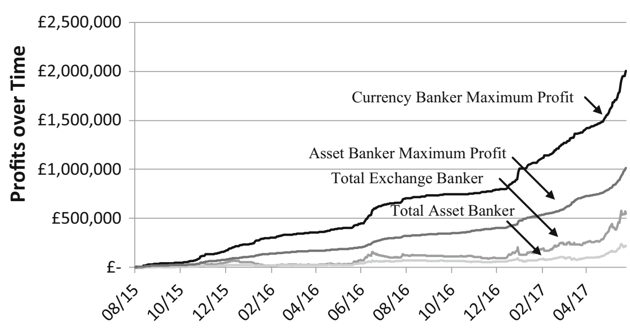


Fig. 16 Currency market banker profits; α = 0.5

7.4 CEO banker deep learning management cluster validation

The CEO banker, "AI Morgan," profits at different risks ratios β with a total of investment of 700 assets for different investment reward memories α is shown in Tables 24, 25 and 26 for the 664 days from year 2015 to 2017. A risk value β = 0.2 represents 560 assets in the exchange market and 140 in the currency market whereas a risk value β = 0.8 is 140 assets in the exchange market and 560 in the currency market, respectively. This research considers the exchange market as low risk and the currency market as high risk.

The results are consisted with the previous validation the best reward is at α = 0.5. The CEO banker takes the right decisions where the profits increase as the risk increases. Table 27 represents the CEO banker deep learning management clusters values at different risk decisions (Figs. 17, 18).

7.5 CEO banker genetic algorithm validation

The genetic algorithm validation for the four different nucleoids (C, G, A, T) average value during the 664

Table 23 Bond and derivative market banker deep learning management cluster validation

Period	$I_{\text{BondMarketBanker}}$				$\bar{w}_{\text{BondMarketBanker}}(c)$				$Y_{\text{BondMarketBanker}}$
Bond market banker									
664 days 2015–2017	0.99	0.99	0.99	0.99	0.21	0.29	0.25	0.25	0.35
Period	$I_{\text{DerivativeMarketBanker}}$				$\bar{w}_{\text{DerivativeMarketBanker}}(c)$				$Y_{\text{DerivativeMarketBanker}}$
Derivative market banker									
664 days 2015–2017	0.99	0.99	0.99	N/A	0.65	0.28	0.07	N/A	0.29

Table 24 CEO banker profits; α = 0.1, year 2015–2017

Risk β = 0.2		Risk β = 0.5		Risk β = 0.8		Max profit
E	*C*	*E*	*C*	*E*	*C*	
758,183	220,418	473,864	551,044	189,546	881,670	6,181,310
Total: 978,600		Total: 1,024,908		Total: 1,071,216		

Table 25 CEO banker profits; α = 0.5, year 2015–2017

Risk β = 0.2		Risk β = 0.5		Risk β = 0.8		Max profit
E	*C*	*E*	*C*	*E*	*C*	
835,763	255,561	522,352	638,903	208,941	1,022,244	5,700,274
Total: 1,091,324		Total: 1,161,254		Total: 1,231,185		

Table 26 CEO banker profits; α = 0.9, year 2015–2017

Risk β = 0.2		Risk β = 0.5		Risk β = 0.8		Max profit
E	*C*	*E*	*C*	*E*	*C*	
598,107	248,115	373,817	620,287	149,527	992,460	5,729,706
Total: 846,222		Total: 994,104		Total: 1,141,986		

Table 27 CEO banker deep learning management cluster validation

Period	$I_{\text{CEO-Banker}}$		Risk β = 0.2		Risk β = 0.5		Risk β = 0.8	
			$Y_{\text{CEO-Banker}}$	*D*	$Y_{\text{CEO-Banker}}$	*D*	$Y_{\text{CEO-Banker}}$	*D*
664 days	0.3530	0.2904	0.4422	0.2	0.4562	0.5	0.4712	0.8

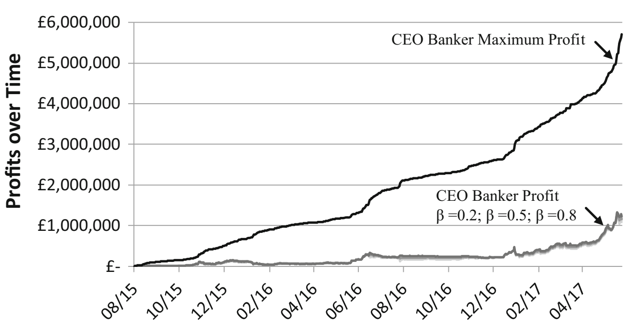


Fig. 17 CEO banker maximum profits; α = 0.5

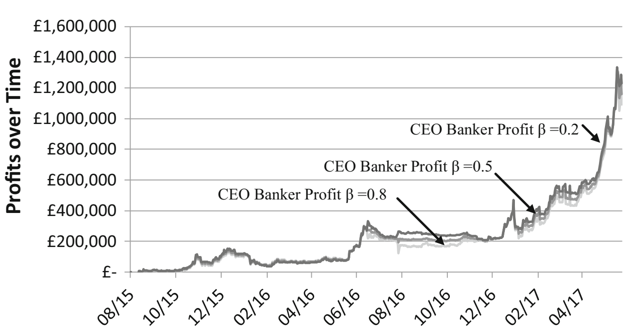


Fig. 18 CEO banker profits; α = 0.5

different days from year 2015 to 2017 is shown in Table 28 with the genetic algorithm error. The genetic algorithm codifies the CEO banker with a residual error.

8 Conclusions

This article has presented a management decision structure based on the human brain with its hierarchical decision process. In addition, this article has defined a new learning genetic algorithm based on the genome where the information is transmitted in the network weights rather than through the neurons. The management decision structure has been implemented in a Fintech application: smart investment model that simulates the human brain with reinforcement learning for fast decisions, deep learning to learn properties to create asset identity, deep learning management clusters to make global decisions and genetic to transmit learned information and decisions to future generations.

In the smart investor model, "GoldAI Sachs" asset banker reinforcement learning algorithm takes the right investment decisions; with great adaptability to asset price changes whereas asset banker deep learning learns asset properties and identity. Market bankers success to increase the profit by selecting the best performing asset bankers and the CEO banker, "AI Morgan," increases the profits considering the

Table 28 Genetic algorithm validation, year 2015–2017

	Error genetic	Iteration	Time (ns)	Nucleoid			
				C	G	A	T
Value	6.76E−31	1.00	1.21E+05	0.1849	0.3594	0.5992	0.9177
σ	7.91E−31	0.00	1.29E+05	0.0064	0.0098	0.0103	0.0033
95% CR	6.02E−32	0.00	9.81E+03	0.0005	0.0007	0.0008	0.0002

associated market risks, prioritizing low risk investment decision at equal profit.

Genetic learning transmits entirely the knowledge acquired from the CEO banker, defined as the combination of memory, identity and decision data, to future banker generations at a minimum error and time. Because the CEO banker information is never lost but transmitted to future generations, genetic algorithm provides immortality. Future work will analyze different methods to improve the performance or increase the profits, of the proposed deep learning cluster structure.

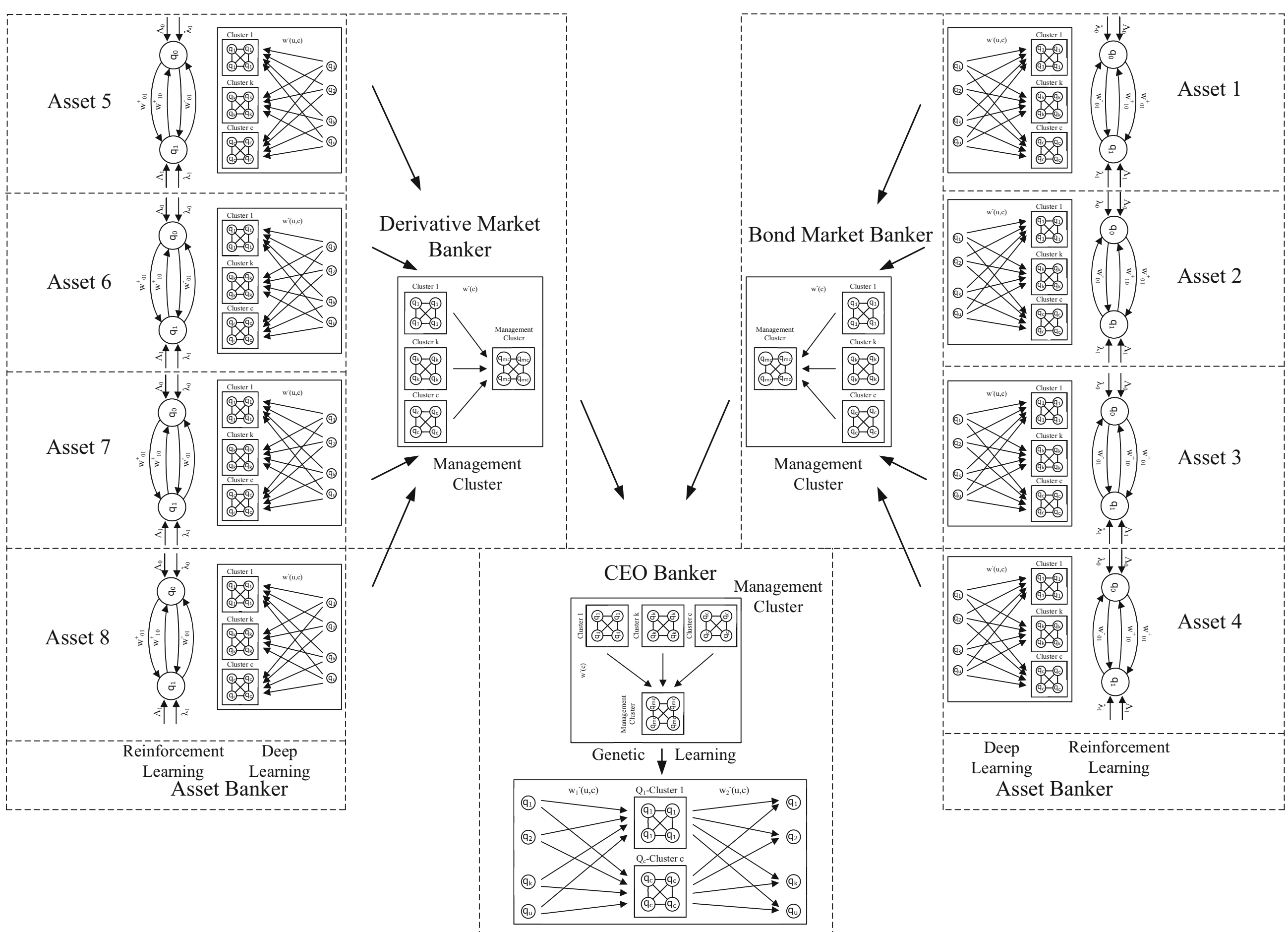
Funding This article was not funded or received any grants, except the academic support of Imperial College London.

Compliance with ethical standards

Conflict of interest The author declares that he has no conflict of interest against any company or institution.

Ethical standards This research has not involved human participants and/or animals, except for the author contribution.

Appendix: Smart investment model—neural schematic



References

1. Ito M, Doya K (2015) Distinct neural representation in the dorsolateral, dorsomedial, and ventral parts of the striatum during fixed- and free-choice tasks. J Neurosci 35(8):3499–3514
2. Kirschner M, Gerhart J (2005) The plausibility of life resolving Darwin's dilemma. Yale University Press, New Haven, pp 1–336
3. Parter M, Kashtan N, Alon U (2008) Facilitated variation: how evolution learns from past environments to generalize to new environments. Public library of science. Comput Biol 4(11):1–15
4. Hinton G, Nowlan S (1996) How learning can guide evolution. Complex Syst 1:495–502
5. Smith D, Bullmore E (2007) Small-world brain networks. Neuroscientist 12:512–523
6. Sporns O, Chialvo D, Kaiser M, Hilgetag C (2004) Organization, development and function of complex brain networks. Trends Cogn Sci 8(9):418–425
7. Gelenbe E (1989) Random neural networks with negative and positive signals and product form solution. Neural Comput 1:502–510
8. Gelenbe E (1993) Learning in the recurrent random neural network. Neural Comput 5:154–164
9. Gelenbe E (1993) G-networks with triggered customer movement. J Appl Probab 30:742–748
10. Gelenbe E, Yin Y (2016). Deep learning with random neural networks. In: International joint conference on neural networks, pp 1633–1638
11. Yin Y, Gelenbe E (2016) Deep learning in multi-layer architectures of dense nuclei, pp 1–10. CoRR arXiv: https://arxiv.org/abs/1609.07160
12. Yin Y, Gelenbe E (2017) Single-cell based random neural network for deep learning. In: International joint conference on neural networks, pp 86–93
13. Serrano W, Gelenbe E (2016) An intelligent internet search assistant based on the random neural network. In: Artificial intelligence applications and innovations, pp 141–153
14. Serrano W (2016) A big data intelligent search assistant based on the random neural network. In: International neural network society conference on big data, pp 254–261
15. Serrano W, Gelenbe E (2017) Intelligent search with deep learning clusters. In: Intelligent systems conference, pp 254–267
16. Serrano W, Gelenbe E (2017) The deep learning random neural network with a management cluster. In: International conference on intelligent decision technologies, pp 185–195
17. Huang G, Zhu Q, Siew C (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
18. Huang G, Chen L, Siew C (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
19. Tang J, Deng C, Huang G (2016) Extreme learning machine for multilayer perceptron. IEEE Trans Neural Netw Learn Syst 27(4):1–13
20. Huang G, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern 42(2):513–529
21. Kasun L, Zhou H, Huang G (2013) Representational learning with extreme learning machine for big data. IEEE Intell Syst 28(6):31–34
22. Pellegrini M, Marcotte E, Thompson M, Eisenberg D, Yeates T (1999) Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. Proc Natl Acad Sci USA 96:4285–4288
23. Suzuki M (1994) A framework for the DNA protein recognition code of the probe helix in transcription factors: the chemical and stereo chemical rules. Structure 2(4):317–326
24. Leshno M, Spector Y (1996) Neural network prediction analysis: the bankruptcy case. Neurocomputing 10(2):125–147
25. Chen W, Du Y (2009) Using neural networks and data mining techniques for the financial distress prediction model. Expert Syst Appl 36:4075–4086
26. Kara Y, Acar M, Kaan Ö (2011) Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul Stock Exchange. Expert Syst Appl 38:5311–5319
27. Guresen E, Kayakutlu G, Daim T (2011) Using artificial neural network models in stock market index prediction. Expert Syst Appl 38:10389–10397
28. Zhang G, Hu M, Patuwo B, Indro D (1999) Artificial neural networks in bankruptcy prediction: general framework and cross-validation analysis. Eur J Oper Res 116:16–32
29. Kohara K, Ishikawa T, Fukuhara Y, Nakamura Y (1997) Stock price prediction using prior knowledge and neural networks. Intell Syst Account Finance Manag 6:11–22
30. Sheta A, Ahmed S, Faris H (2015) A comparison between regression, artificial neural networks and support vector machines for predicting stock market index. Int J Adv Res Artif Intell 4(7):55–63
31. Khuat T, Le M (2017) An application of artificial neural networks and fuzzy logic on the stock price prediction problem. Int J Inf Visual 1(2):40–49
32. Naeini M, Taremian H, Hashemi H (2010) Stock market value prediction using neural networks. In: International conference on computer information systems and industrial management applications, pp 132–136
33. Iuhasz G, Tirea M, Negru V (2012) Neural network predictions of stock price fluctuations. In: International symposium on symbolic and numeric algorithms for scientific computing, pp 505–512
34. Nicholas A, Zapranis A, Francis G (1994) Stock performance modeling using neural networks: a comparative study with regression models. Neural Netw 7(2):375–388
35. Bahrammirzaee A (2010) A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. Neural Comput Appl 19:1165–1195
36. Coakley J, Brown C (2000) Artificial neural networks in accounting and finance: modeling issues. Int J Intell Syst Account Finance Manag 9:119–144
37. Fadlalla A, Lin C (2001) An analysis of the applications of neural networks in finance. Interfaces 31(4):112–122
38. Huang W, Lai K, Nakamori Y, Wang S, Yu L (2007) Neural networks in finance and economics forecasting. Int J Inf Technol Decis Mak 6(1):113–140
39. Li Y, Jiang W, Yang L, Wu T (2018) On neural networks and learning systems for business computing. Neurocomputing. 275:1150–1159
40. Duarte V (2017) Macro, finance, and macro finance: solving nonlinear models in continuous time with machine learning. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, pp 1–27
41. Stefani J, Caelen O, Hattab D, Bontempi G (2017) Machine learning for multi-step ahead forecasting of volatility proxies. In: Workshop on mining data for financial applications, pp 1–12
42. Fischer T, Krauss C (2017) Deep learning with long short-term memory networks for financial market predictions. In: Florida Atlantic University discussion Papers in Economics, vol 11, pp 1–32
43. Hasan A, Kalıpsız O, Akyokuş S (2017) Predicting financial market in big data: deep learning. In: International conference on computer science and engineering, pp 510–515
44. Arifovic J (1995) Genetic algorithms and inflationary economies. J Monet Econ 36:219–243

45. Kim K, Han I (2000) Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert Syst Appl 19:125–132
46. Ticona W, Figueiredo K, Vellasco M (2017) Hybrid model based on genetic algorithms and neural networks to forecast tax collection: application using endogenous and exogenous variables. In: International conference on electronics, electrical engineering and computing, pp 1–4
47. Hossain D, Capi G (2017) Genetic algorithm based deep learning parameters tuning for robot object recognition and grasping. Int Sch Sci Res Innov 11(3):629–633
48. David O, Greental I (2014) Genetic algorithms for evolving deep neural networks. In: ACM genetic and evolutionary computation conference, pp 1451–1452
49. Tirumala S (2014) Implementation of evolutionary algorithms for deep architectures. In: Artificial intelligence and cognition, pp 164–171
50. Schrauwen B, Verstraeten D, Campenhout J (2007) An overview of reservoir computing: theory, applications and implementations. In: Proceedings of the European symposium on artificial neural networks, pp 471–482
51. Gallicchio C, Micheli A, Pedrelli L (2017) Deep reservoir computing: a critical experimental analysis. Neurocomputing. 268:87–99
52. Gallicchio C, Micheli A (2017) Echo state property of deep reservoir computing networks. Cogn Comput 9(3):337–350
53. Butcher JB, Verstraeten D, Schrauwen B, Day CR, Haycock PW (2013) Reservoir computing and extreme learning machines for non-linear time-series data analysis. Neural Netw 38:76–89
54. Park J, Sandberg IW (1991) Universal approximation using radial-basis-function networks. Neural Comput 3(2):246–257
55. Huang G, Siew C (2004) Extreme learning machine: RBF network case. In: IEEE control, automation, robotics and vision conference, pp 1–8
56. Leonard JA, Kramer MA (1991) Radial basis function networks for classifying process faults. IEEE Control Syst Mag 11(3):31–38
57. Bianchini M, Frasconi P, Gori M (1995) Learning without local minima in radial basis function networks. IEEE Trans Neural Netw 6(3):749–756
58. Gelenbe E (1997) A class of genetic algorithms with analytical solution. Robot Auton Syst 22(1):59–64
59. Gelenbe E (2007) Steady-state solution of probabilistic gene regulatory networks. Phys Rev 76(1):031903
60. Gelenbe E, Liu P, Laine J (2006) Genetic algorithms for route discovery. IEEE Trans Syst Man Cybern 36(6):1247–1254
61. Gelenbe E (2007) Dealing with software viruses: a biological paradigm. Inf Secur Tech Rep 12:242–250
62. Gelenbe E (2008) Network of interacting synthetic molecules in equilibrium. Proc R Soc A Math Phys Sci 464:2219–2228
63. Kim H, Gelenbe E (2009) Anomaly detection in gene expression via stochastic models of gene regulatory networks. Comput Biol 10(3):S26
64. Kim H, Gelenbe E (2012) Stochastic gene expression modeling with hill function for switch-like gene responses. IEEE/ACM Trans Comput Biol Bioinf 9(4):973–979
65. Kim H, Gelenbe E (2012) Reconstruction of large-scale gene regulatory networks using bayesian model averaging. IEEE Trans Nanobiosci 11(3):259–265
66. Gelenbe E (2012) Natural computation. Comput J 55(7):848–851
67. Kim H, Park T, Gelenbe E (2014) Identifying disease candidate genes via large-scale gene network analysis. Int J Data Min Bioinform 10(2):175–188
68. Gelenbe E (2004) Cognitive packet network. Patent US 6804201 B1
69. Gelenbe E, Xu Z, Seref E (1999) Cognitive packet networks. In: International conference on tools with artificial intelligence, pp 47–54
70. Gelenbe E, Lent R, Xu Z (200) Networks with cognitive packets. In: IEEE international symposium on the modeling, analysis, and simulation of computer and telecommunication systems, pp 3–10
71. Gelenbe E, Lent R, Xu Z (2001) Measurement and performance of a cognitive packet network. Comput Netw 37(6):691–701
72. Gelenbe E, Lent R, Montuori A, Xu Z (2002) Cognitive packet networks: QoS and performance. In: IEEE international symposium on the modeling, analysis, and simulation of computer and telecommunication systems, pp 3–9
73. Gelenbe E (1993) G-networks: a unifying model for neural nets and queueing networks. In: Modelling analysis and simulation of computer and telecommunications systems, pp 3–8
74. Fourneau J, Gelenbe E, Suros R (1994) G-networks with multiple class negative and positive customers. In: Modelling analysis and simulation of computer and telecommunications systems, pp 30–34
75. Gelenbe E, Timotheou S (2008) Random neural networks with synchronized interactions. Neural Comput 20(9):2308–2324