SN Computer Science (2021) 2:111
https://doi.org/10.1007/s42979-021-00524-9

ORIGINAL RESEARCH

seMLP: Self-evolving Multi-layer Perceptron in Stock Trading Decision Making

Seow Wen Jun[1] · Arif Ahmed Sekh[2] · Chai Quek[1] · Dilip K. Prasad[2]

Received: 31 August 2020 / Accepted: 15 February 2021 / Published online: 24 February 2021

Abstract

There is a growing interest in automatic crafting of neural network architectures as opposed to expert tuning to find the best architecture. On the other hand, the problem of stock trading is considered one of the most dynamic systems that heavily depends on complex trends of the individual company. This paper proposes a novel self-evolving neural network system called self-evolving Multi-Layer Perceptron (seMLP) which can abstract the data and produce an optimum neural network architecture without expert tuning. seMLP incorporates the human cognitive ability of concept abstraction into the architecture of the neural network. Genetic algorithm (GA) is used to determine the best neural network architecture that is capable of knowledge abstraction of the data. After determining the architecture of the neural network with the minimum width, seMLP prunes the network to remove the redundant neurons in the network, thus decreasing the density of the network and achieving conciseness. seMLP is evaluated on three stock market data sets. The optimized models obtained from seMLP are compared and benchmarked against state-of-the-art methods. The results show that seMLP can automatically choose best performing models.

Keywords Self evolving neural network · Stock trading neural network · Data adaptation

Introduction

Stock markets are one of the most dynamic systems where a large number of parameters influences the changes in the market [1–3]. Traditionally, two methods are popular for predicting trends of markets, (i) analysis of the financial condition, current trends of economic, several national/international issues, and so on (ii) analysis of historical data such as the movement of the stock prices. Historical data can be analyzed using a variety of indexes such as moving average, golden cross, dead cross, etc. Researchers proposed several modeling approaches such as rule-based [4, 5], trading pattern [6], classifier-based [7], etc. Recent growing nature of Artificial Intelligence (AI) also opened up new opportunities in computational stock trading. Recently, several soft computing and machine learning-based algorithms such as ANN-based [8, 9], Optimized Network-based [10], machine learning fusion [11], have been proposed. Due to the complex nature of the stock market, it is very difficult to design a static intelligent system for stock trading [12].

Artificial neural networks (ANN) have garnered a lot of attention these days due to its accurate predictions in various diverse fields such as medical [13], natural language processing [14] and image recognition [15] among many others. However, it often requires expert tuning of the hyperparameters to be able to generate accurate predictions [16]. Architectures often have to be crafted and subsequently trained and tuned by experts and different permutations have to be tested manually to determine the best topology of the neural network that gives the best results. With a predefined topology, the whole process becomes an adaption of weights to the data and there is no real learning of the topology to best represent knowledge abstraction of the data. In stock trading, if accurate predictions can be obtained using ANNs,

✉ Arif Ahmed Sekh
skarifahmed@gmail.com

Seow Wen Jun
seow0095@e.ntu.edu.sg

Chai Quek
ashcquek@ntu.edu.sg

Dilip K. Prasad
dilipprasad@gmail.com

[1] Nanyang Technological University, Singapore, Singapore

[2] UiT The Arctic University of Norway, Tromsø, Norway

investors will be able to detect trend reversals earlier which will allow them to make more informed decisions to take a long or short position in the market. These days, statistical methods such as the ARIMA method [17] are often used for stock prediction. Since ANNs are a universal function approximator [18], it can be used to model stock data better than such statistical methods which are based on moving averages [19, 20].

Neural network compression is a way to reduce number of parameters without compromising the accuracy (see Fig. 1). With the daily increasing demands for on-device computation, the demand of optimized neural networks are also increasing. Such compression is useful in IoT devices, drones, self driving cars, and mobile devices. In [21] authors used heuristic search based on threshold for compression. Baysian sparsity based weigh estimation is proposed in [22]. Yu et al. [23] proposed to use a weight pruning method for compression. Recently, Ma et al. [24] proposed network comparison using Baysian optimization. Low-rank decomposition based method replace set of parameters by the low-rank approximation for network reduction [25]. In this types of compression Singular Value Decomposition (SVD) [26] and Canonical Polyadic (CP) decomposition [27] are among popular choices. In this paper, we have proposed a self-evolving intelligent framework based on multi-layer perceptron for decision making in stock trading. The proposed method can automatically design the optimum neural network from the data itself.

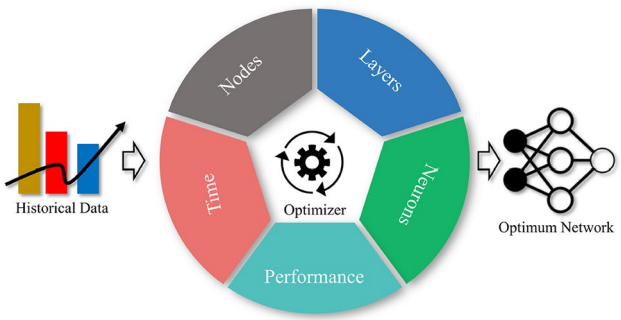


Fig. 1 Optimized neural network by compression

The main contributions of the paper are: (a) A self-evolving ANN is proposed that use the human cognitive ability of concept abstraction and GA for automatic creation of neural network for a given data set. (b) A link pruning module is employed for removing unnecessary links in the network. The module allows automatic design of a compressed network without compromising the performance. (c) The method is applied on three publicly available stock trading data set and achieved state-of-the-art performance.

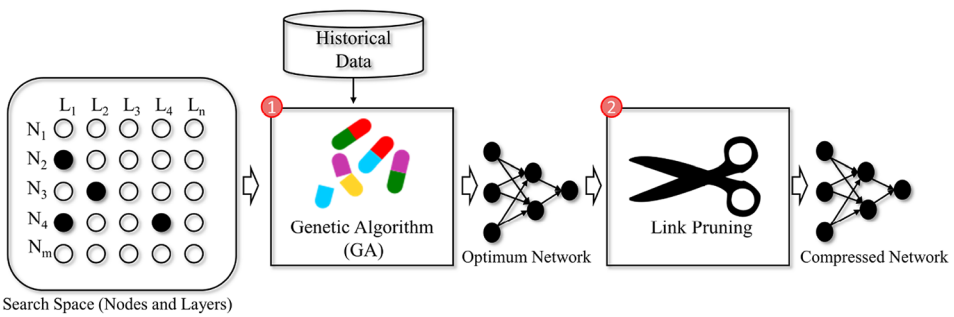
The rest of the paper is organized as follows. Sect. 2 describes the proposed seMLP and the components. Section 3 demonstrates the results and discussion of the results and finally, Sect. 4 concludes the article.

Proposed seMLP

The proposed self-evolving Multi-Layer Perceptron (seMLP) network is a novel neural network that builds its architecture based on the data and hence is capable of adapting its architecture to the data set for better accuracy. seMLP consists of two primary modules. First, it adopts the genetic algorithm (GA) approach to determine the optimal number of hidden layers and neurons per layer to allow the network to better generalize to the data. By allowing the network to discover the best architecture for itself, seMLP can learn the best topology for the data presented without any expert tuning or input. The main goal is to allow the network to be constructed such that the minimum number of hidden layers and neurons are used to derive the most compact and concise model for any data that is fed into the network. Second, a link pruning module is employed to shrink the size of the network by removing links which are redundant and to achieve conciseness and the least density. The framework is depicted in Fig. 2. There are two primary modules (marked with a circle) to produce a compressed neural network from the historical data.

Let $\mathcal{D}$ is a data consists of a set of features (x) and an outcome (y). $\mathcal{D}$ can be divide into train and test ($\mathcal{D}_t, \mathcal{D}_v$) any number of times for cross validation. Let $\mathcal{S}$ is a search space contains nodes and layers that can form a neural network by taking any number of nodes and layers. Performance

Fig. 2 Functional block diagram of the proposed seMLP



of a neural network ($p(\nu)$) is defined by the mean squared error (MSD) and calculated as $\frac{1}{2}\sum_{i=1}^{n}(\hat{y}-y)^2$ and is cross validated over $\mathcal{D}$. Weight of a neural network ($\omega(\nu)$) of the search space $\mathcal{S}$, $\nu \in \mathcal{S}$ is defined by the number of nodes, layers, and links present in the network. In this paper Eq. 1 is solved to find the best network for a given data.

$$\min_{\nu\in\mathcal{S}}(p(\nu),\omega(\nu)). \tag{1}$$

GA-based Network Evaluation

Using a genetic algorithm, seMLP is able to identify an optimal architecture for the network. The parameters to be tuned by the genetic algorithm are: (a) number of hidden layers and (b) number of nodes per hidden layer. A constraint is also set such that a subsequent hidden layer should have a smaller number of neurons for the concept of abstraction and this is realized in the fitness function to force the genetic algorithm to favor networks with such characteristics.

Chromosome Encoding: seMLP employs binary encoding for its chromosome. In the chromosome, it contains a layer cost (δ) and also another variable represents the binary form of layers (λ). This chromosome is the backbone of the individual candidate solution in the population. All the genetic material is contained in this chromosome thus it means the candidate solution is encoded inside the chromosome. From this chromosome, we can derive the architecture of the neural network. For example, the number of hidden layers of a network is equal to the length of δ or λ. The number of neurons (γ) in i^{th} hidden layer is defined as (2). Where the value of λ is the integer value of the binary form of the layer.

$$\gamma_i = \lfloor \text{integer}(\lambda_i) \times \delta_i \rfloor. \tag{2}$$

Population Initialisation: The population is initialized randomly to allow for a random distribution of individuals with different candidate solutions. This gives the genetic algorithm a large heterogeneous pool of genes to choose from to ensure that it is more likely to be able to break out of local optima to get closer to the optimal solution. Second, the population initialized must also be big enough so that there are enough representative candidate solutions contained within the initial population and prevent it from getting too homogenous after a few evolutions.

Selection: The selection operator is used to select individuals to breed for the next generation of network. seMLP has adopted a hybrid approach to the selection where a portion of the fittest candidate solutions from the current generation will be kept and the whole population will also be used for breeding. Thus, the top 10% of the network structure of current generation is always kept and allowed to stay on in the next generation without any modifications. To ensure that the population will get fitter over time, seMLP employs rank weighting. All the individuals in the population is ranked based on the value of their fitness function and fitter with a higher probability. Using rank weighting, it ensures that networks with higher values of fitness function will have a higher probability of being selected for breeding. In contrast, solutions with lower values have a lower chance of being selected. Hence, by selecting fitter individuals for breeding, the next generation will have a higher probability of being fitter than the current generation. The probability of choosing any candidate solution (S) in a population N is calculated as $\frac{N-rank}{\sum_i^n i}$. Where *rank* is the position of S in the population according to fitness.

Crossover: The crossover operation is one of the most important functions as it decides the future of the population. After the selection phase, the crossover operation chooses two parents and breeds two new children by recombining the chromosomes of the parents. After two parents are chosen, there must be checks to make sure that they are not the same parent as it would mean the children will turn out to be similar. Since the crossover operation is used to create a new children chromosome in GA and the chromosome in seMLP consists of the δ and λ, the crossover operation has to be performed on both these values. In seMLP, the single-point crossover is used because it is fast, easy to calculate and it generally gives a good candidate solution as a result.

Integer Crossover for δ: The values of δ are integers in the chromosome. Hence, the integer crossover [28] is used. A random weight (β) between [0, 1] is chosen and multiplied with the δ. The δ of the two children are re-combined values of the parents. The crossover is defined as:

$$\delta_{c1} = \beta\delta_{p1} + (1-\beta)\delta_{p2}, \tag{3}$$

$$\delta_{c2} = \beta\delta_{p2} + (1-\beta)\delta_{p1}, \tag{4}$$

where $p1$, $p2$ are parent and $c1$, $c2$ are children.

Single Point Crossover for λ: The λ in the chromosome is binary encoded. Thus, we can use the single point crossover for binary encoded values. A random point, k, is chosen as the crossover point in the string of binary digits. This ensures that the child carries genetic information from both parents. The single point crossover are defined as:

$$\lambda_{c1} = \lambda_{p1}[1..k] + \lambda_{p2}[k..\text{end}], \tag{5}$$

$$\lambda_{c2} = \lambda_{p2}[1..k] + \lambda_{p1}[k..\text{end}], \tag{6}$$

where k is the maximum length of the layers.

Choosing Layers for Crossover: If the number of layers of the chosen parents differ, there is a need to choose exactly which layers are to be involved in the crossover because there is positional significance to the layers. For example, if the first parent has five layers and the second parent has only two layers, there might potentially be information loss if the fourth layer of $p1$ is crossed with the first layer of $p2$. This is because the first layer of $p2$ is likely to be a more generalized layer as it is the first hidden layer while the fourth layer of $p1$ is likely to be a more concrete layer. Hence, during crossover, the position of the layers selected for crossover should be kept relatively similar between the two chosen parents thus preserving the positional significance of the layer. Therefore, for cases where the number of layers differ between the two parents, we propose a structure for choosing which layers are to be chosen for crossover. First, for the two resultant children, the number of layers for them will be the same as the parents. For example, if parent 1 has 5 layers and parent 2 has 3 layers, child 1 and child 2 will then have five and three layers too. The structure for the crossover is defined by a partition point (p). p is defined in 7. Provided parent $p1$ always have more than or an equal number of layers compared to parent $p2$:

$$p = \left\lfloor \frac{length(\lambda_{p1})}{length(\lambda_{p2})}. \right\rfloor \tag{7}$$

Subsequently, for each layer in p2, the layers in p1 that could be selected for crossover with the layer in p2 is restricted to the range $\{\lambda_i, \lambda_i + p\}$. Within that range, the layer to be crossovered with p2 will be randomly selected with equal probability. The child with the same number of layers as p1 will then inherit the rest of the missing layers from p1. If the crossover produces any layer with zero nodes or $\delta = 0$ then the process is repeated.

Mutation: The purpose of the mutation operator is to ensure that the population does not get too homogenous which might result in candidate solutions getting stuck in local minima which decreases the possibility of getting a more optimal solution. In seMLP, the mutation is achieved by a parameter called mutation probability (μ). A higher probability will result in more mutations which will introduce more new genetic material back into the population at a higher rate while a lower probability will introduce new genes at a slower rate. seMLP does not mutate individuals who are in the top 10% of the population in terms of fitness and these are the fittest individuals who are selected to be kept without any changes. As there are two components to the chromosome of seMLP which is the δ and λ, the mutation rate is also defined for both of them (μ_δ, μ_λ) and these are in the range [0,1]. In each mutation μ_δ, μ_λ is chosen randomly and must be less or equal to the restriction of the mutation probability.

Fitness Function: The fitness function is the most important parameter for the success of any genetic algorithm. It affects not just the convergence of the genetic algorithm but also the quality of the final solution; whether it is near the global optimum. The fitness function must thus be crafted according to what the desired outcome of the genetic algorithm is. For seMLP it is to discover the best architecture for the input data. Hence, there will be three aspects to the fitness function:

(i) The network must hold the property of knowledge abstraction i.e. proceeding hidden layers must have lower or equal number of nodes,
(ii) The network must be minimized in terms of number of layers and nodes, and
(iii) The network performance must be higher in terms of RMSE or R^2. Therefore, these goals have to be implicitly encoded into the fitness function for the genetic algorithm to discover the best solution.

First, we define abstraction penalty (α) to ensure the knowledge abstraction. α is defined in (8).

$$\alpha_i = \begin{cases} 0, & \text{if length(layer}_i) > \text{length(layer}_{i+1}) = 1 \\ 1, & \text{otherwise} \end{cases}. \tag{8}$$

Next, a hidden layer penalty (ρ) is introduced to ensure minimum network depth. ρ is calculated as $n \times c_h$, where n is number of hidden layer and c_h is hidden layer cost, we have taken $c_h = 0.1$ empirically and found suitable in our case. The normalized constant (c) is used to normalized penalty. It is defined in Eq. (9), where γ is the lower limit of the credit in a generation.

$$c = (\gamma + c_h)n - \gamma. \tag{9}$$

Credit (τ) is the cost of a network, higher cost represents a complex network. For a given network τ is calculated as $\sum_{i=1}^{n} \alpha_i$, where n is the number of layers present in the network. Finally, a credit ratio (ϕ) is calculated as Eq. (10). ϕ is normalized between 0 and 1 over a generation.

$$\phi = \frac{\tau - \rho + c}{(n-1)}. \tag{10}$$

Outliers' Penalty: Outlier values for the quality metric can be defined as the values which are very far from the

distribution of the other values. Mathematically, if the population is fairly normally distributed, it can be defined as values which lie outside more than three standard deviations away from the mean. As seMLP employs feature scaling, any outliers might skew the distribution in favor of the outliers which means that all the other non-outlier values might have a very small range after scaling. This is not ideal as the majority of the values are "squashed" and might affect the accuracy of the fitness evaluation as the credit ratio (ϕ) might outweigh the quality metric. Therefore, seMLP employs an outlier labeling-based technique using the box-and-whisker plot or more commonly known as the boxplot. The advantage of using a boxplot is that it is computationally fast, does not require any tuning parameters and also uses robust summary statistics that are located at actual data points [29]. Furthermore, it does not make any distributional assumptions and does not depend on any standard deviation or mean. Hence, we can apply the boxplot to detect and remove outliers. Tukey et al. [30] defined the inter-quartile range (IQR) as the interval between the lower quartile (q_1) and upper quartile (q_3). The lower quartile (q_1) is the 25th percentile while the upper quartile (q_3) is the 75th percentile. The IQR can then be used to identify outliers according to the illustration below:

$$iqr = q_3 - q_1 \tag{11}$$

$$\text{Lower Fence} = q_1 - (1.5 \times \text{iqr}), \tag{12}$$

$$\text{Upper Fence} = q_3 - (1.5 \times \text{iqr}), \tag{13}$$

$$\text{Lower Extreme} = q_1 - (3 \times \text{iqr}), \tag{14}$$

$$\text{Upper Extreme} = q_3 - (3 \times \text{iqr}). \tag{15}$$

In seMLP, there is only a need to detect outliers at one of the extreme points. For example, if RMSE is the quality metric used, seMLP will only utilize the boxplot method to catch outliers above the upper extreme as the lower the RMSE, the better the quality of the candidate solution. After the outliers are found, they will be removed from the normalization of the quality metric as they will skew the normalization as described above. Although these outliers represent a very bad candidate solution, they are not eliminated from the population immediately by forceful removal. Instead, they are allowed to remain in the population in the last few positions in terms of ranking to maintain the variety in the gene pool. Thus, their fitness function will be calculated in a slightly different manner from the rest of the population. The difference will be in the calculation of the quality metric as the outliers only affect the calculation of that. Hence, the calculation of the quality metric of the outliers must give a worse result compared to the non-outliers, such that the outliers will remain the last few in the population.

First, the minimum fitness of the entire population based on the non-outliers has to be calculated as such based on for RMSE and R^2 respectively. Next, the fitness of the outlier is calculated and an outlier penalty (Ω) is introduced as in Eq. (16).

$$\Omega = \min(\text{fitness}_{\text{non outliers}}) - \text{fitness}_{\text{outliers}}, \tag{16}$$

Ω is used to normalize the fitness score. High value of Ω puts lower weight of outliers when we normalize the fitness score. Finally, the fitness score (ψ) is calculated using

$$\psi = \begin{cases} \frac{\phi}{\text{RMSE}_{\text{normalized}}}, & \text{if RMSE} \\ \phi \times R^2_{\text{normalized}}, & \text{if } R^2 \end{cases}. \tag{17}$$

Link Pruning

After the best architecture of the neural network has been found by the genetic algorithm, seMLP will proceed to prune the links/weights in the network to remove the largest number of links possible without affecting the accuracy of the MLP. In seMLP, pruning of links is done based on the threshold principle. If links that have weights above a certain threshold, they are kept as they are deemed to be important while links at or below the threshold are removed by setting them to zero. To find the threshold, a binary search is employed. The threshold will be found using a binary search based on the RMSE error returned after the threshold is applied. The thresholding is based on percentile ranking so the threshold is set in the range from (0, 100) non-inclusive. For example, if the threshold set is 60%, any links below the 60th percentile of all weights for that layer will be removed. Linear interpolation is also used when the desired percentile lies between two data points (i, j) and $i < j$. Binary search works for the pruning of links because there is a direct correlation between the RMSE of the neural network against the links. If the threshold is set too high, the RMSE of the neural network increases as too many useful links have been pruned. On the other hand, if the threshold is set too low, the RMSE of the neural network does not change because no useful links have been pruned and thus the accuracy of the network has not been affected. Therefore, this fits the requirements of using a binary search algorithm where the threshold responds proportionally to its performance. This can be illustrated according to the Fig. 3.

After the thresholds for each layer has been found by binary search, the whole network can be pruned. The links between each layer can be removed according to the respective thresholds found. The last layer to the output layer is fully connected and not pruned because it would affect

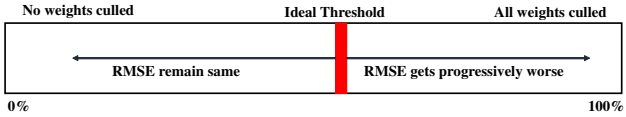


Fig. 3 Illustration of the ideal threshold for pruning of links in neural network

Table 1 Data set used for the experiments

Dataset	Duration	#of data points
S&P 500	11 March 1999 to 14 Nov 2018	4954
NASDAQ	7 Oct 1998 to 22 Jan 2019	5104
KOSPI	15 Mar 1998 to 15 Mar 2019	4932

the error too drastically. This is similar to the CNN architecture [31], where the output layer will always be fully connected to ensure that all the information flow and the knowledge learned in the network will reach the output thus giving a more accurate result. Any links removed from the preceding layer will not affect the subsequent layer because those links are below the threshold to make any meaningful contribution to the node that it is connected to. Hence, removing it will not affect the performance of the network and the whole network can be pruned.

Results and Discussion

This section, the performance of seMLP for optimizing the network and also performance in stock trading is analyzed. The experiments are carried out on three different unique stock indexes: (a) S&P 500 Index, (b) NASDAQ Composite, and (c) Korea Composite Stock Price Index (KOSPI). The first two are chosen as they are two of the most heavily followed stock indices in the US stock market and most investors around the world will be concerned about its movement and trends. The KOSPI is the representative stock market index of South Korea and it is chosen because it one of the more popular stock market index in Asia and to give some contrast to the S&P500 and NASDAQ Composite which are stock market indices based in the western world. The details of the data sets are given in Table 1.

Experimental Set-up and Hyperparameter

For the experimental setup of the genetic algorithm portion of seMLP, it requires a definition of the parameters for both the genetic algorithm portion and also for the neural network training in the genetic algorithm. The genetic algorithm portion of seMLP requires certain parameters to be set such as the limits for the number of hidden layers and the number of nodes per hidden layer. These are defined in Table 2.

Table 2 Hyperparameters of GA

Hyperparameter	Value
Number of evolutions	25
Number of models per generation (including during population initialization)	100
Lower limit of number of hidden layers	2
Upper limit of number of hidden layers	6
Quality metric in fitness function	R^2
Lower limit of number of nodes per hidden layer	1
Upper limit of number of nodes per hidden layer	255
Mutation rate of nodes	0.1%
Mutation rate of delta	0.3%
Crossover rate	1

Table 3 Hyperparameters of neural network

Hyperparameter	Value
Number of input features	39
Number of output features	1
Epochs	300
Activation function	ReLU
Batch size	128
Learning rate	0.001
Regularizer	10^{-4}

As the genetic algorithm portion of seMLP takes in neural networks as candidate solutions, there is a need to keep the hyperparameters of the neural networks constant across the whole population and generations. The hyperparameters of the neural network are defined in Table 3.

We have taken input feature length 39 empirically. First, to construct the input vector, the difference between the prices of consecutive days have been calculated using (18), where $y(t)$ is stock price at t.

$$\delta y(t) = y(t) - y(t-1). \tag{18}$$

A sliding window of size 39 is then used over δy with a stride of 1 to determine the range of input vectors for the neural network. Thus, each input vector will now contain the delta changes in prices between consecutive days. Second, after preparing the input vectors, each input vectors have to be normalized. This is one of the most important preprocessing steps such that the neural network will be able to learn the trend and changes in prices. Each input vector and corresponding output will be normalized within their own window.

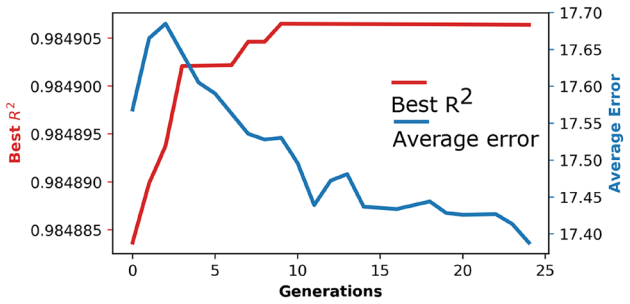


Fig. 4 Best R^2 and average error against the generation in S&P 500 data set

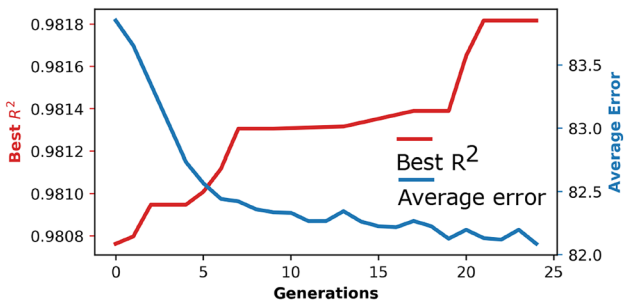


Fig. 5 Best R^2 and average error against generation in NASDAQ data set

All the experiments are carried out in Intel(R) Xeon(R) Gold 6154 CPU with 128 GB of RAM and NVIDIA Quadro RTX 600 GPU of capacity 64 GB. The data set is divided into two parts. First 80% data is used for training and rest for test and performance evaluation.

Results and Analysis of GA-Based Evolving

This section we present the evaluation of the networks and their performance against generation. We have analyzed the best R^2 and average error in each generation. The average error is defined as the mean of the error between the predicted and correct prices over all individuals in the population of that generation. Figs. 4, 5, and 6 presents such results on three public data sets. It is observed that the average R^2 is gained as 0.00223%, 0.108%, and 0.0720% respectively in S&P, NASDAQ, and KOSPI data set.

It can be seen that for all the three data sets, the best R^2 value improves from generation 0 before converging to its final value. Before it converges, it is often able to break out of local optima and to continue increasing. This demonstrates the ability of the genetic algorithm to find better candidate solutions through mutation and crossovers. The average error for each generation is also decreasing which confirms that the R^2 value is reporting the correct trend of

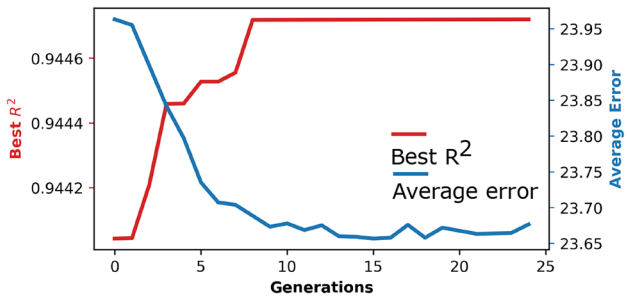


Fig. 6 Best R^2 and average error against the generation in KOSPI data set

Table 4 Improvement in R2 values from first to last generation (Optimum Model)

Data set	Best R2 at first generation	Best R2 at last generation
S&P500	0.984884	0.984906
NASDAQ	0.98076	0.98182
KOSPI	0.94404	0.94472

improvement where the predicted prices are getting more accurate.

Optimum Model Selection: The main motivation of the uses of GA is to find the best network structure for the prediction of the various stock market indices. The best architectures for the three different stock market is chosen based on the performance of the network in terms of R^2 score. The best results of prediction in first and final generation is reported in Table 4. The models producing best R^2 are selected. It is observed that a network with only two hidden layers is sufficient for all the cases but the number of nodes in each layer is different. The best network found by GA as $\{41, 34\}$ nodes (S&P 500), $\{8, 3\}$ (NASDAQ), and $\{20, 12\}$ number of nodes (KOSPI). Thus, drawing conclusions from these genetic algorithm experiments, it can be concluded that in general for stock market indices data sets, only two hidden layers are needed for the depth of the network while the nodes per hidden layer can be kept below 50. This should be sufficient to model most stock market indices and give a reasonably good prediction.

Pruning of Links

After the genetic algorithm in seMLP finds the most compact architecture with the minimum width for the data set involved, seMLP can prune the links in the network so that the network will be the most concise. As all the neural networks built in the genetic algorithm are fully-connected

multi-layer perceptrons, they are the densest possible and this section aims to cull the redundant links in the network. Only the links from the input layer to the last hidden layer is retrieved. The links from the last hidden layer to the output layer is not retrieved and not pruned because experimentally, we have found that culling the links from the last layer substantially decreases the accuracy and thus that last layer should stay fully connected. We have used three neural networks of the same architecture for each data set as three neural networks to predict prices at $t+1$, $t+2$ and $t+3$. We have observed that the links of the networks are not similar during pruning. The results of such pruning are depicted in Fig. 7.

From the pruning results, typically about 17–94% of links can be pruned between consecutive layers in a neural network. For a complete neural network with all the layers, 14.29–84.90% of total weights can be pruned from the network. This shows that there are actually a lot of redundant links in a neural network as these links are too low in weight to affect the final output of the network. Hence, they can be deleted from the network without affecting the accuracy thus saving space and computation complexity. It is also observed that the S&P500 data set has the highest number of redundant links at 80.35%, followed by the KOSPI at 43.40% and lastly the NASDAQ Composite at 28.77%. This could be because the neural network for the S&P500 data set is the most complex and thus have the largest number of links, therefore, increasing the chances of redundancy. On the other hand, the neural network for the NASDAQ Composite is much less complex with only eight nodes in hidden layer 1 and three nodes in hidden layer 2. Thus, the links in the network will be significantly more important as there are much lesser links in the network. Hence, each link is more likely to play a significant role in the output so the threshold for pruning will be lower.

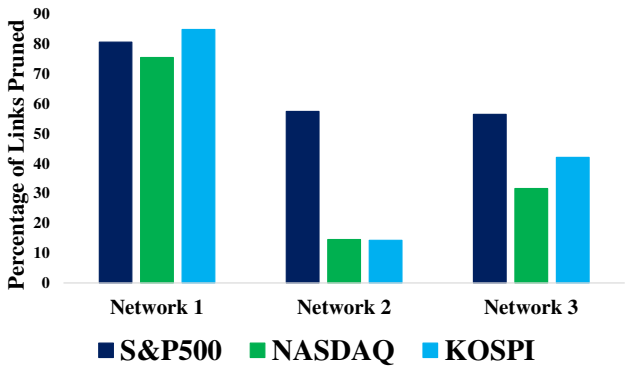


Fig. 7 Results of link pruning of three networks in a different data set

Benchmark Experiment

To evaluate the results of the genetic algorithm with other neural networks, benchmark experiments were performed using the S&P 500 data set. For this benchmark experiment, the genetic algorithm was used to determine the best topology for the S&P 500 data set. The seMLP neural network generated by the genetic algorithm uses the same hyperparameters as described earlier. Two other multi-layer perceptrons (MLP) with different hidden layer sizes are also designed to use as benchmark comparisons with the seMLP network. The accuracy metrics Non-Dimensional Error Index (NDEI) is used and defined as $NDEI = \frac{RMSE}{\sigma}$, σ is the standard deviation of the correct output values. The benchmark experiments have been carried out on S&P 500 data set using evolving Mamdani fuzzy inference system (eMFIS) [32], Self-reorganizing Fuzzy Associative Machine (SeroFAM) [33], Evolving Fuzzy Neural Network (EFuNN) [34], SeroFAM using An extended Takagi-Sugeno-Kang inference system (SeroFAM-TSK) [35] and dynamic evolving neural-fuzzy inference system (DENFIS) [36] are fuzzy neural network (FNN) models which are split into Mamdani and TSK models. All these FNN models and the two ANN models are used as benchmarks against the seMLP model. An increase in NDEI represents an increase in error hence it means the accuracy is worse. The results are summarized in Table 5.

For the two ANN models, the NDEI is the average over ten different trainings selection to ensure an accurate representation of the error. The results indicate that seMLP compares favorably with fuzzy neural network models. It outperformed Mamdani type models which are focused on interpretability but it could not match the TSK models which are the most accurate type of fuzzy neural networks. These results show that the genetic algorithm portion of seMLP is strong and is able to determine a good topology for an MLP network that can hold its own against another more advanced type of neural network architectures such as FNNs. Therefore, since the genetic algorithm is modular, it can potentially be used on other architectures such as on fuzzy

Table 5 Results of benchmark experiments on S&P500

Method	Type	NDEI
ANN [9] {41, 41} Layers	MLP	0.0272
ANN [9] {34, 41} Layers	MLP	0.0280
eMFIS [32]	Mamdani	0.0241
SeroFAM [33]	Mamdani	0.0272
EFuNN [34]	Mamdani	0.1544
SeroFAM-TSK [35]	TSK	0.0199
DENFIS [36]	TSK	0.0200
seMLP {41, 34} Layers	MLP	**0.0168**

neural networks to find the best topology for the data set. This would give a better result for those networks. From the results, it can also be seen that the architecture derived from seMLP outperformed the other artificial neural networks (ANNs) that do not have abstraction built into its topology. The seMLP network has hidden layers of size 13 and 2 for the first and second layers respectively which shows that the hidden layers decrease in size for knowledge abstraction to happen in the network. Benchmarking this with two other ANNs that has the same hidden layer size and another with an increasing hidden layer size, the results show that a multi-layer perceptron (MLP) network with abstraction outperforms other MLP networks that are not constrained for abstraction as evident in the lower NDEI value for the seMLP network. Therefore, it can be concluded that for an MLP network to achieve the best accuracy, knowledge abstraction should happen in the neural network thus designing an architecture with decreasing hidden layer size will aid this abstraction of knowledge and give predictions with higher accuracy.

Adaptability of seMLP

Instead of using fully connected multi-layer perceptrons, seMLP can also be adapted to use other forms of neural networks such as LSTMs. For seMLP, it is a very modular algorithm as it is derived from the genetic algorithm. Hence, any form of the neural network can be fed into the genetic algorithm portion of seMLP and it will be able to find the best architecture of that particular network. If an LSTM is used instead of an MLP, the genetic algorithm will be able to find the best architecture for the number of LSTM cells and best number of layers within each LSTM cell to give the highest accuracy of prediction from the LSTM network. The link pruning module is not applicable here. We have also experimented with an LSTM based model selection applicable to S&P 00 data set. The parameters of the GA is presented in Table 6. The parameters of LSTM based training is reported in Table 7.

The optimizer ends with a LSTM containing three layers with [200, 184, 125] nodes in the corresponding layers (excluding the input and output layers). The model achieved best R^2 of 0.9882254, and NDEI of 0.0144.

Ablation study

Here, we present the ablation study of the proposed seMLP. We have used S&P500 data set and evaluation metric discussed in Sect. 3.4. The results are summarized in Table 8. We have considered four experimental conditions. First, we use an ANN with two layers and ten nodes in each layer. Next, we use maximum possible layers (6) with maximum possible nodes (255) according to our previous experimental setup. We note that both networks perform poorly. In the third case, we use seMLP produced network without pruning the links. We note that the optimized network is compressed with high accuracy (lower NDEI). Finally, we show that the network can be further compressed without compromising accuracy.

Table 6 Hyperparameters of GA used in LSTM-based model selection

Hyperparameter	Value
Number of evolutions	25
Number of models per generation (including during population initialization)	100
Lower limit of number of layers	2
Upper limit of number layers	10
Quality metric in fitness function	R^2
Lower limit of number of nodes per layer	10
Upper limit of number of nodes per layer	256
Mutation rate of nodes	0.1%
Mutation rate of delta	0.3%
Crossover rate	1

Table 7 Hyperparameters of the LSTM

Hyperparameter	Value
Number of input features	39
Number of output features	1
Epochs	300
Optimizer	Adm
Batch size	128
Learning rate	0.001
Regularizer	10^{-4}

Conclusion

Due to the increasing growth and popularity of artificial neural networks for applications in many industries today, there is a need for neural networks that can adapt its architecture to fit the data thus giving more accurate predictions. Many ANNs today require expert tuning and are unable to represent an abstraction in the data. To address these issues, a novel self-evolving Multi-Layer Perceptron (seMLP) system is proposed that consists of a genetic algorithm that is capable of learning the best architecture for the data to best represent an abstraction of the data. The genetic algorithm in seMLP prioritizes the selection of models with better knowledge abstraction thus it is able to build architectures that have an inherent ability to abstract knowledge from the data. This genetic algorithm proposed is also able to work

Table 8 Results of ablation study on S&P500

Method	NDEI	Remarks
1. ANN 2 layers with 10 nodes	0.1231	ANN with smaller width and depth
2. ANN 6 layers with 255 nodes	0.2102	ANN with maximum width and depth
3. seMLP {41, 34} (optimized)	0.0168	Optimized layers and nodes
4. seMLP {41, 34} + Link pruning	0.0168	Contains ∼ 70% less link than 3

with deep neural networks with multiple hidden layers and no pre-determined architecture is required at the start. After finding the best architecture, seMLP then proceeds to prune the network using a thresholding function that prunes the redundant links to produce a final architecture that is the most compact and concise. The results have been encouraging where the network discovered by seMLP outperformed other neural networks without abstraction. Furthermore, this network has been pruned to significantly to leave only the important links. Lastly, the seMLP network is applied to stock trading. The results have shown that using the proposed seMLP a data specific artificial neural network can be generated using GA and also beneficial for a highly dynamic system such as stock trading.

Funding Open access funding provided by UiT The Arctic University of Norway (incl University Hospital of North Norway). This study is not Funded from anywhere.

Compliance with Ethical Standards

Conflict of Interest The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed Consent Informed consent was obtained from all individual participants included in the study.

References

1. Pang X, Zhou Y, Wang P, Lin W, Chang V. An innovative neural network approach for stock market prediction. J Supercomput. 2020;76(3):2098–118.
2. Hiransha M, Ab Gopalakrishnan E, Krishna Menon V, Soman KP. Nse stock market prediction using deep-learning models. Procedia Comput Sci. 2018;132:1351–62.
3. Ashish S, Dinesh B, Upendra S. Survey of stock market prediction using machine learning approach. In: 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), IEEE, 2017;2:506–509.
4. Keng Ang K, Quek C. Stock trading using rspop: a novel rough set-based neuro-fuzzy approach. IEEE Trans Neural Networks. 2006;17(5):1301–15.
5. Arévalo R, García J, Guijarro F, Peris A. A dynamic trading rule based on filtered flag pattern recognition for stock market price forecasting. Expert Syst Appl. 2017;81:177–92.
6. Liheng Z, Charu A, Guo-Jun Q. Stock price prediction via discovering multi-frequency trading patterns. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, 2017;2141–2149.
7. Ballings M, Van den Poel D, Hespeels N, Gryp R. Evaluating multiple classifiers for stock price direction prediction. Expert Syst Appl. 2015;42(20):7046–56.
8. Hedayati Moghaddam A, Hedayati Moghaddam M, Esfandyari M. Stock market index prediction using artificial neural network. J Econ Finance Adm Sci. 2016;21(41):89–93.
9. Selvamuthu D, Kumar V, Mishra A. Indian stock market prediction using artificial neural networks on tick data. Financial Innov. 2019;5(1):16.
10. Qiu M, Song Yu. Predicting the direction of stock market index movement using an optimized artificial neural network model. PLoS One. 2016;11(5):e0155133.
11. Patel J, Shah S, Thakkar P, Kotecha K. Predicting stock market index using fusion of machine learning techniques. Expert Syst Appl. 2015;42(4):2162–72.
12. Soni S. Applications of anns in stock market prediction: a survey. Int J Comput Sci Eng Technol. 2011;2(3):71–83.
13. Ting DSW, Liu Y, Burlina P, Xu X, Bressler NM, Wong TY. Ai for medical imaging goes deep. Nat Med. 2018;24(5):539.
14. Goldberg Y. A primer on neural network models for natural language processing. J Artif Intell Res. 2016;57:345–420.
15. Barret Z, Vijay V, Jonathon S, Quoc VL. Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2018;8697–8710.
16. Hector M, Aaron K, Matthias F, Jost Tobias S, Matthias U, Michael B, Maximilian D, Marius L, Frank H. Towards automatically-tuned deep neural networks. In: Automated Machine Learning. Springer, 2019;135–149.
17. Ariyo AA, Adewumi AO, Ayo CK. Stock price prediction using the arima model. In: 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation. IEEE, 2014;106–112.
18. Barron AR. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans Inf Theory. 1993;39(3):930–45.
19. Weng B, Ahmed MA, Megahed FM. Stock market one-day ahead movement prediction using disparate data sources. Expert Syst Appl. 2017;79:153–63.

20. Dash R, Kishore Dash P. A hybrid stock trading framework integrating technical analysis with machine learning techniques. J Finance Data Sci. 2016;2(1):42–57.
21. Han S, Pool J, Tran J, Dally W. Learning both weights and connections for efficient neural network. In: Advances in neural information processing systems, 2015;1135–1143.
22. Louizos C, Ullrich K, Welling M. Bayesian compression for deep learning. In Advances in neural information processing systems, 2017;3288–3298.
23. Yu R, Li A, Chen C-F, Lai J-H, Morariu VI, Han X, Gao M, Lin, C-Y, Davis LS. Nisp: Pruning networks using neuron importance score propagation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018;9194–9203.
24. Ma X, Triki AR, Berman M, Sagonas C, Cali J, Blaschko MB. A bayesian optimization framework for neural network compression. In: Proceedings of the IEEE International Conference on Computer Vision, 2019;10274–10283.
25. Jaderberg M, Vedaldi A, Zisserman A. Speeding up convolutional neural networks with low rank expansions. In: Proceedings of the British Machine Vision Conference. BMVA Press, 2014.
26. Denton EL, Zaremba W, Bruna J, LeCun Y, Fergus R. Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems, 2014;1269–1277.
27. Tai C, Xiao T, Zhang Y, Wang X, Weinan E. Convolutional neural networks with low-rank regularization. In: 4th International Conference on Learning Representations, ICLR 2016, 2016.
28. Cheong F, Lai R. Constraining the optimization of a fuzzy logic controller using an enhanced genetic algorithm. IEEE Trans Syst Man Cybernet Part B (Cybernetics). 2000;30:31–46.
29. Wickham H, Stryjewski L. 40 years of boxplots. Am Statistician. 2011.
30. Friedman JH, Tukey JW. A projection pursuit algorithm for exploratory data analysis. IEEE Trans Comput. 1974;100(9):881–90.
31. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, 2012;1097–1105.
32. Chai Y, Jia L, Zhang Z. Mamdani model based adaptive neural fuzzy inference system and its application. Int J Comput Intell. 2009;5(1):22–9.
33. Tan J, Zhou WJ, Quek C. Trading model: self reorganizing fuzzy associative machine-forecasted macd-histogram (serofam-fmacdh). In: 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, 2015;1–8.
34. Rezaei-Ravari M, Sattari-Naeini V. Reliable congestion-aware path prediction mechanism in 2d nocs based on efunn. J Supercomput. 2018;74(11):6102–25.
35. Jacob BJ, Cheu EY, Tan J, Quek C. Self-reorganizing tsk fuzzy inference system with bcm theory of meta-plasticity. In: The 2012 International Joint Conference on Neural Networks (IJCNN). IEEE, 2012;1–8.
36. Jiang H, Kwong CK, Okudan Kremer GE, Park W-Y. Dynamic modelling of customer preferences for product design using denfis and opinion mining. Adv Eng Inform. 2019;42:100969.