Soft Computing (2021) 25:10617–10628
https://doi.org/10.1007/s00500-021-05995-9

MATHEMATICAL METHODS IN DATA SCIENCE

A genetic algorithm-assisted deep learning approach for crop yield prediction

Luning Bi[1] · Guiping Hu[1]

Accepted: 20 June 2021 / Published online: 8 July 2021

Abstract
The world population continues to increase which imposes rising demand in agriculture production. How to improve crop breeding to feed the growing population is a significant challenge. The traditional crop breeding is resource-intensive and time-consuming. Predictive modeling on crop yield can speed up the breeding process and make it resource-efficient. In this paper, a genetic algorithm (GA)-assisted deep learning solution method is proposed for the crop yield prediction. The proposed method consists of two phases, i.e., the global search phase and the local search phase. In the global search phase, GA is used to search for the best initial weights of the neural network. In the local search phase, random perturbation is added to avoid the local optimum and vanishing gradient problems. A case study of crop yield prediction is conducted to compare the proposed method and other gradient-based methods. The results show that the proposed method outperforms the gradient-based methods in terms of convergence speed and prediction accuracy.

Keywords Crop yield prediction · Gradient descent · Genetic algorithm · Neural network

1 Introduction

The world population is expected to grow from 7.2 billion to between 9.6 billion by 2050 and 12.3 billion by 2100 (Gerland et al. 2014). This imposes significant challenges for agriculture production due to the increasing demand and limited arable land. One way to alleviate this problem is to improve crop production with efficient crop breeding. Traditional breeding is phenotype-based, which requires selection of the individual after phenotyping. The development of genomic and analytic technologies has reduced the genotyping and sequencing cost significantly. Now there are over 7.4 million plant accessions in gene banks around the world. To effectively leverage these resources and shorten the breeding time, analytic models are essential to predict the phenotype of the crops. This would significantly reduce the phenotyping cost and improve the efficiency of crop breeding.

Crop yield is jointly determined by the genotype and environment of the plants through complex biological and physiologic relationships. Understanding these relationships remains a significant challenge. Many of the existing studies are based on linear models (Des Marais et al. 2013). However, the limitation of the linear models is that the interactive effects of genotype and environment factors cannot be modeled and analyzed effectively. The phenotypes expressed by a genotype under varied environmental conditions can be very different. In other words, the resulting phenotypic variation based on the same genotype can be environment-dependent (Comstock and Moll 1963). Then, linear mixed models based on maximum likelihood were developed to detect the correlation between genotype and environment (Holland 2006). However, this method has limitations on the size of dataset and the prediction accuracy due to the sensitivity of the types of input parameters. The complexity of the G $\times$ E problem and the limitation of the existing prediction model serve as the major motivation for this study to design mathematical models and the solution method that can overcome these shortcomings. In this paper, we proposed a neural network method for the crop yield prediction. Instead of analyzing the G $\times$ E interaction, we predict the crop yield using genotype and environment information through neural networks.

✉ Guiping Hu
gphu@iastate.edu

[1] Iowa State University, Ames, USA

Neural networks are function approximators that have achieved state-of-the-art accuracy in many applications, such as self-driving cars, game playing and face identification (Samek et al. 2016). They especially excel at learning from large datasets with labeled samples (Bau et al. 2020). The most popular methods to train the deep artificial neural networks (DNNs) are gradient-based learning algorithms which is also called first-order methods. The reasons include their ease of implementation, memory efficiency (typically requiring storage on the order of the parameter dimension) and convergence guarantees (Liu et al. 2017). There are many mature gradient-based algorithms, such as stochastic gradient descent (SGD) (Bottou 2010) and improved versions, e.g., Adagrad (Mukkamala and Hein 2017) and Adam (Kingma and Ba 2015).

Despite its popularity as a universal function approximator and easy implementation, the gradient-based algorithms are faced with the inherent drawback of getting trapped in local minima and slow convergence due to random initialization of synaptic weights and biases prior to training a neural network (Ruehle 2017). With every rerun of neural networks during the training phase, the gradient-based algorithms train the neural network based on different initial weights leading to different prediction performance and convergence speed (Chandwani et al. 2015). The other problem is the vanishing/exploding gradient problem caused by the long training process. Many techniques have been designed to reduce the impact of the problem, such as variations on weight initialization strategy, alternative network structures and gradient descent schemes (Olimov 2020; Sherstinsky 2020; Hanin 2018). The use of gradient descent schemes is the most common change since it can keep the gradient in a reasonable range to avoid gradient vanishing/exploding. The commonly adopted gradient descent schemes include rectified linear unit (ReLU) activation function (Yarotsky 2017), gradient clipping (Zhang et al. 2019) and L2 normalization (Wang et al. 2018). However, the drawback is that these methods require more computation resources. In order to minimize the probability of inconsistency and solve the vanishing/exploding gradient problem, it is necessary to develop an effective methodology to improve its prediction accuracy and convergence to global optima. This is major motivation for the algorithm design in this study.

Neuroevolution is a machine learning method that uses evolutionary algorithms to optimize neural networks (Lehman and Miikkulainen 2013). A critical feature of neuroevolution algorithms is that they can adapt to a dynamic environment by their population-based search strategy (Yao 1999). At each iteration, evolution strategies will generate many children solutions, i.e., different neural networks. By comparing the fitness of generated solutions, the best neural network will be selected. Genetic algorithm (GA) is one of the popular evolutionary algorithms due to its ease implementation and good convergence (Whitley et al. 1990). GAs can effectively address the limitations of the traditional gradient-based method through two improvements. First, GA can help avoid the local optimum problem by generating a population that consists of multiple solutions (Salhi et al. 2019). Second, GA can be used as a zeroth-order optimization method using approximate gradients. However, these methods exhibit poor convergence properties when the parameter dimension is large (Kitano 1990).

The combination of evolutionary algorithms and neural networks has been tried since the 1960s (Bremermann 1962). However, most of them were designed for shallow neural networks which have only one or several hidden layers. For a given degree of function approximation, the number of neurons needed by shallow networks is exponentially larger than that of deep neural networks (Baral et al. 2018). Recently some studies have proposed some neuroevolutionary methods for different deep neural networks, e.g., multilayer neural network (Assunção et al. 2019), deep reinforcement learning (Risi and Stanley 2019).

Although these neuroevolution methods have been extended to different structures of deep neural networks, the drawback of current methods includes two aspects. First, each generated solution means a new neural network. To evaluate the fitness of the generated solution, we need to train it until it converges so that we can determine which solution is better. It requires many computing resources to evaluate every generated solution. Second, although evolution strategies can help jump out the local minimum, it is not always as efficient as the gradient method due to the randomness of evolution strategies.

In light of the research background, the contribution of this paper can be summarized as follows.

1. To overcome the shortcomings of traditional linear models, we proposed a neural network-based approach to predict the crop yield using the environment information and the genotype information.
2. To address the local optimum issue and the gradient vanishing/exploding problem, we proposed a two-phase GA-assisted method for deep neural networks. In the global search phase, we generate multiple sets of parameters for the neural network and use GA to search the solution space. In the local search phase, our proposed method combines the gradient method and the GA method. The GA algorithm will evolve a low-dimensional subspace, i.e., the nodes in one layer, by adding a random perturbation. Then the weights will be updated by the gradient descent method.

The proposed approach has been validated with a case study of the Syngenta crop yield prediction challenge. First, the comparison results between our proposed method and other gradient methods demonstrate that our proposed method can achieve lower prediction error. Second, we observe that in terms of prediction accuracy and algorithm stability, the proposed method also outperforms other methods that are used to address the gradient vanishing problem.

The remainder of this paper is organized as follows. In Sect. 2, the problem description of crop yield prediction is illustrated. In Sect. 3, the application of the neural network to plant traits (e.g., yield) prediction is introduced, and the drawbacks of the gradient-based methods are discussed. The proposed GA-assisted approach is presented in Sect. 4. Section 5 includes the numerical experiments and analysis of the results. Finally, conclusions are drawn in Sect. 6.

2 Crop yield prediction using genotype and environment information

The objective of this study is to predict the phenotype of crop, i.e., yield, with the genotype and environment information.

Genotype (G) refers to the genetic makeup of an individual, which is the nucleotide sequences of DNA (a gene or genes) that are transmitted from parents to offspring (Kang 1997). A gene is defined as a sequence of DNA or RNA that is the basic physical and functional unit of heredity (Davis 2017). Genotype includes one gene locus (AA, Aa, aa), two gene loci (AABB, AaBB, AAbb, etc.) or multiple genes (aabbcc, aabbccddee, AABbCCDDEEFF, etc.). For example, Fig. 1 shows genomic information of 40 genotypes which consists of 100 gene loci.

Environment can be defined as the circumstances surrounding an organism or a group of organisms (Kang 1997). Phenotype (P) refers to an individual's discernible traits, such as yield, height and stalk number. Phenotype is the expression of a genotype in an environment (E). Phenotypes are jointly determined by genotype and environment. Figure 2 illustrates the relationship among phenotype, genotype and environment from three points: (1) The plants of identical genotypes (e.g., Genotype A) growing in different environments may show different phenotypes; (2) when the environment is the same, the plants of different genotypes (e.g., the left endpoints of the three lines) show different phenotypes; and (3) different genotypes react differently when environmental factors change. Assuming the environment axis represents soil moisture and the phenotype axis is the plant yield, when the soil moisture increases in a specific range, plants of Genotype C will have a higher yield while plants of Genotype B will have a lower yield.

Due to the complexity of the G by E as described above, traditional linear models cannot achieve satisfying prediction accuracy. Thus, this paper introduced a neural network-based approach to predict crop yield. The advantage of the neural networks is that they can fit the complex nonlinear relationship without analyzing the G by E interaction specifically.

3 Deep neural network and the drawback of the gradient descent method

Traditional methods used to predict crop yield are linear models. However, the prediction of additive models not satisfying. In recent years, the linear mixed model and regression approaches have become very popular (Schaeffer 2008; Stinchcombe 2012; Robinson 2013). They are more advanced and have achieved better prediction accuracy. The problem is that their performance is not satisfying when dealing with a large dataset that consists of millions of variables and samples. In this case, the complexity of the problem increases significantly. The neural network due to its ability to map complex nonlinear and unknown relationships is a preferred choice among

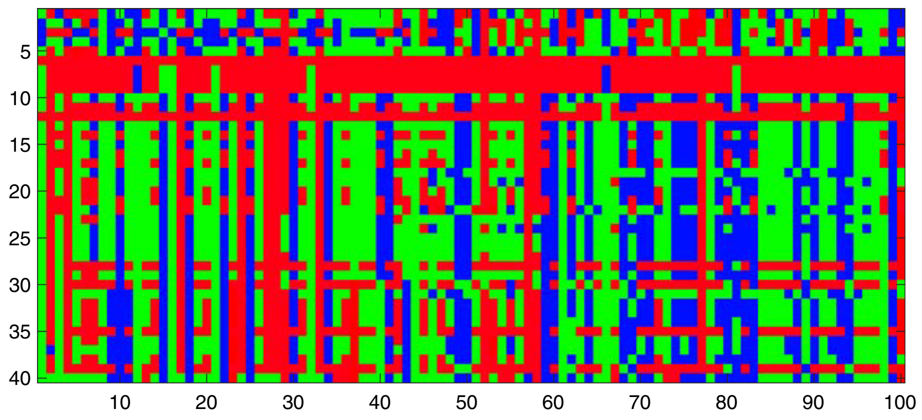


Fig. 1 Genomic information of 40 genotypes (each color represents one gene type)

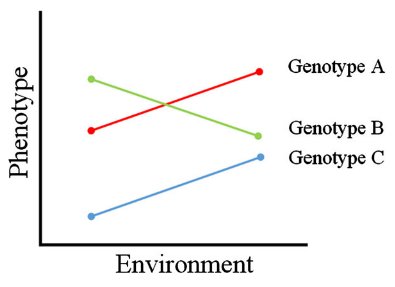


Fig. 2 Genotype and environment interaction (Kang 1997)

researchers for modeling this kind of problems (Schwardt 2009; Affes 2019). Deep neural networks (DNNs) are multilayer feed-forward neural networks that are usually trained using gradient-based methods (Hinton et al. 2012). The gradient-based algorithm is a local search algorithm that employs gradient descent to iteratively update the weights and biases of the neural network, minimizing the loss function commonly measured in terms of a squared error between the actual results and the output of the neural network (Chandwani et al. 2015).

As shown in Fig. 3, a DNN consists of an input layer, hidden layers and a output layer. Each layer consists of multiple nodes. A node multiplies input from the data with a set of coefficients. Then, these products are summed. So far, this model is still linear. However, some problems cannot be solved by a linear model. To solve this problem, a node's so-called activation function is introduced. The widely used activation functions include the sigmoid function, hyperbolic tangent (Tanh) function and ReLU function. The sigmoid function has been successfully applied in prediction problems and classification problems because it is bounded, differentiable and monotonic (Han and Moraga 1995). The Tanh function is a transformed version of the sigmoid function, which has larger gradient values. It performs better than the sigmoid function in the training of multilayer neural networks (Karlik and Olgac 2011). The ReLU function was proposed by Nair and Hinton in 2020 (Nair and Hinton 2010). Since it preserves the properties of linear models, the ReLU function reduces

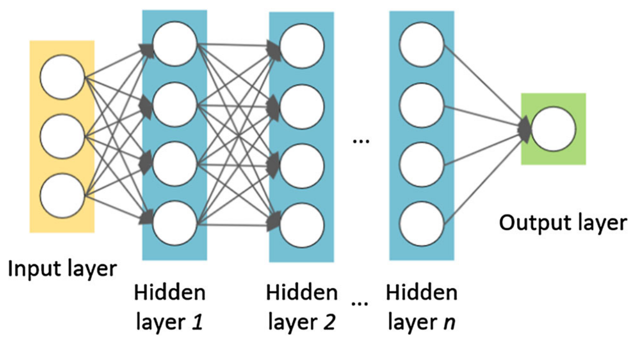


Fig. 3 Structure of deep neural network

the computing complexity of gradient-descent methods (Dahl et al. 2013). However, all these functions suffer from the gradient vanishing/exploding problem.

Given an input vector of dimension d, we consider a neural network with L layers of neurons for prediction. We denote by M_l the number of neurons in the lth layer (note that M_0 = d). We denote the neural activation function by σ. Let $W_l \in R^{M_{l-1} \times M_l}$ denote the weight matrix connecting the $(l-1)$th and lth layer and b_l denote the bias vector for neurons in the lth layer. Let $W_{L+1} \in R^{M_l}$ and $b_L \in R$ denote the weight vector and bias scalar in the output layer, respectively. Therefore, the output of the network f: $R^d \to$ R can be expressed by

$$f(x;\ W) = W_{L+1}^T \sigma(W_L^T \sigma\ (\ldots \sigma\ (W_1^T\ x + b_1) + b_{L-1}) + b_L) + b_{L+1} \tag{1}$$

A standard method to update the W in Eq. (1) is gradient descent (Johnson and Zhang 2013). It can be represented by the following update rule:

$$W^{(t)} = W^{(t-1)} - \eta_t \nabla f(x, W^{(t-1)}) \tag{2}$$

However, this update rule comes along with a problem, i.e., gradient vanishing or gradient exploding problem (Bengio et al. 1994). Take b_1 as an example. The update rule is shown in Eq. (3). Generally, the initial value of w is lower than 1. If the value of σ' is less than 1, $\left|\frac{\partial C}{\partial b_1}\right| \to 0$. This term tends to be very small, which means that the update of the parameter in the first layer is very small. On the other hand, if $\left|\sigma'(z)w\right| > 1$, the term will be very large. The gradient vanishing or gradient exploding problem is essentially caused by the chain multiplication.

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_{L+1}} \sigma'(z_{L+1}) w_{L+1} \sigma'(z_L) w_L \ldots \sigma'(z_2) w_2 \sigma'(z_1) \tag{3}$$

Techniques have been proposed to address this issue, such as Leaky ReLU, gradient clipping and L2 normalization. As shown in Eq. (4), the gradient of the Leaky ReLU activation function is 1 when the unit is active; otherwise, it is a small, positive value. Gradient clipping is to assign a fixed value to the gradient when the gradient is too large or too small. L2 normalization is to normalize the gradient so that the sum of squares will always be up to a fixed value. In this paper, we proposed a GA-assisted method to update the parameters of the neural network.

$$f(x) = \begin{Bmatrix} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{Bmatrix} \tag{4}$$

4 Proposed GA-assisted neuroevolution approach

To overcome the drawback of gradient descent methods motioned above, we proposed a novel zeroth-order method which is named as GA-assisted neuroevolution approach. The advantages of the proposed approach include the ability to avoid the local optimum and gradient vanishing problem, the ease of the computation, the applicability to deep neural networks and the ability to handle large datasets.

4.1 Genetic algorithm (GA)

The idea of genetic algorithms is based on one nature mechanism, namely evolution (Davis 1991; Whitley 1994). The workflow of the genetic algorithm is shown in Fig. 4. There are three main operators:

Selection During each successive generation, two individuals of the existing population are selected to generate a new individual. The selection process is usually based on the fitness of individuals. A typical selection method is the roulette in which the individuals of better fitness are preferred. Because there is a good reason that the child solution which combines the component of good parent solutions is more likely to be good. However, this method might be time-consuming when the population is large. There are also some methods that randomly select the parent solutions. In these methods, we do not need to calculate the fitness of each solution.

Crossover After the selection of the parent solutions, the crossover operator is executed. Crossover is also called recombination. It is used to combine the advantageous genes of the parent solution to generate a new individual solution. If the generated solution is better than the parent solutions, replace the old ones with the new ones. The key point of the crossover operator is how to detect the advantageous genes of the parent solutions, especially for the large-scale optimization problem. One solution to this is to shrink the exchange scope of the parent solutions.

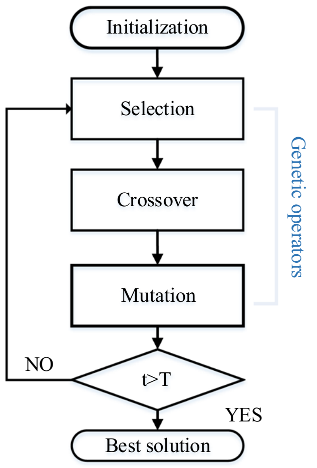


Fig. 4 Workflow of genetic algorithm

Mutation In genetic algorithms, the mutation is a genetic operator used to maintain genetic diversity. In the process of selection and crossover, there is no new gene introduced. On the contrary, some genes might be lost in the process. The consequence of this is that all the individuals in the population will be identical and the algorithms will converge early. To avoid this case, the mutation is used to alter one or more genes randomly. Since mutation operator is a random search strategy, mutation may change the previous solution largely. Therefore, the mutation operator is usually executed according to a small threshold probability.

GA has been successfully applied to artificial neural networks. However, different from shallow neural networks, deep neural networks have millions of weights to be evolved. In other words, it is a large-scale optimization problem (Mei et al. 2014). In a large-scale problem, the search space will expand exponentially due to the increase of variables (Kaveh et al. 2021). The performance of general evolutionary algorithms will deteriorate rapidly due to the "curse of dimensionality" (Kitano 1990; de Oliveira Florentino 2018). A popular solution is the divide-and-conquer strategy. In this paper, a GA-based approach that integrates global search strategy and local search strategy is proposed.

4.2 GA-assisted neuroevolution approach

The workflow of the approach is shown in Fig. 5. There are two phases in the proposed approach, i.e., global search and local search. In the initialization, we randomly generate multiple candidate matrices for each weight and bias. The goal of global search is to search for the best combination. The best individual in the global search phase is used as the base for further update in local search.

Global search phase The parameter initialization of neural networks has a great influence on the algorithm performance. In order to avoid this impact, a novel global search strategy is designed to ensure the quality of the initial point. Instead of generating only one group of parameters, it randomly generates n groups of parameters for each layer. Therefore, there are n candidate weights and biases for each layer. The total number of possible solutions will be n^L. Then, the global search strategy is used to search for the best combinations in this solution space.

a. Initialization. The global population is randomly generated. An individual can be represented by the number of selected weight or bias matrices. For

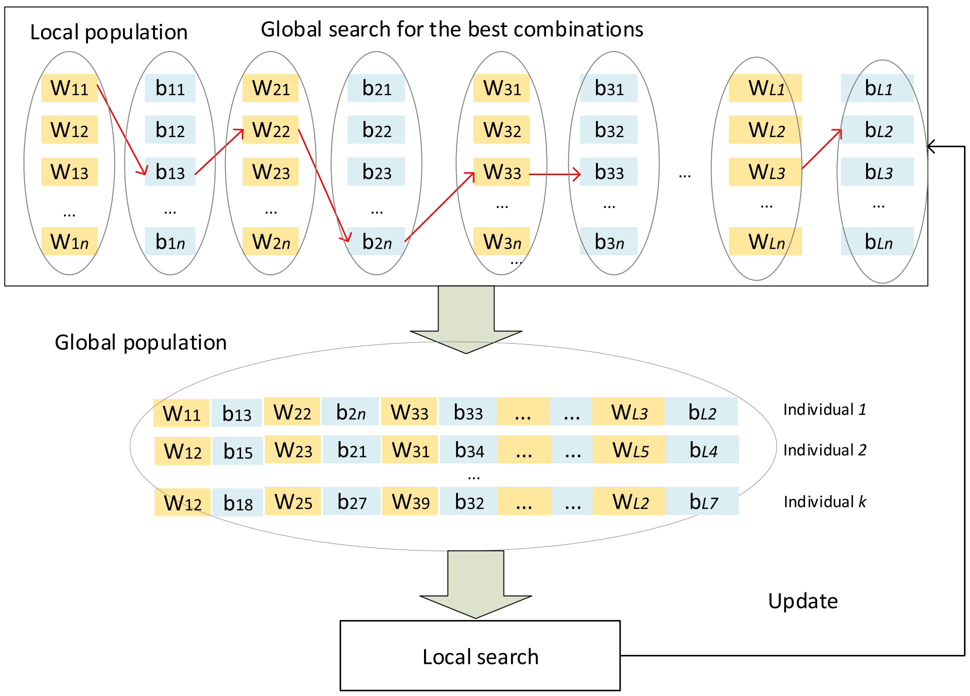


Fig. 5 Workflow of GA-assisted deep learning approach

example, $(W_{11}, b_{13}, W_{22}, b_{2n}, W_{33}, \ldots)$ can be represented as (1, 3, 2, n, 3, …).

b. Evolution of elites. The elites are the best m individuals. The fitness of an elite is good enough that it is not easy to improve it by selection and crossover operators. Therefore, for each elite, two positions are selected randomly. Then the selected positions will be mutated, i.e., replaced by a random selected solution in the candidate pool. For example, as shown in Fig. 6, the W_{22} and b_{33} of the elite are replaced by the W_{25} and b_{37}, respectively.

c. Evolution of worst m individuals. The strategy to improve bad individuals is to combine them with the elites. Since it is safe to make big changes on the current individual, the current individual is used as the father solution. Randomly selected one of the elites as the mother solution. Then the two-point crossover operator is executed by randomly selecting two cutting points and exchange the segment between two cutting points. For example, as shown in Fig. 7, the child solution inherits the W_{11}, b_{13}, W_{22}, W_{L3} and b_{L2} from Parent 1 solution and other parts from Parent 2 solution. There are also some other versions of crossover operators, e.g., the single-point crossover and the uniform crossover. The single-point crossover operator is to select one point and then recombine two individuals. Assuming the number of genes is N, the total number of possible combinations is (N-1) since there are (N-1) possible cutting points. The uniform crossover is to choose each bit with equal possibility to generate a random subset. For uniform crossover, there are 2^N possible combinations because each gene has two states, i.e., selected or not selected. However, the uniform crossover cannot guarantee the exploration ability in 2^N space when N is large. Thus, the two-point crossover, which is able to generate C($N+1$,2) combinations, can achieve a better balance between the exploration ability and the exploitation ability. The two-point crossover also requires less time than the uniform crossover operator in terms of computing efficiency.

d. Evolution of other individuals. The current individual is selected as Parent 1 solution. One elite is randomly selected as Parent 2 solution. Then randomly select a starting point and exchange the next S parameters. For

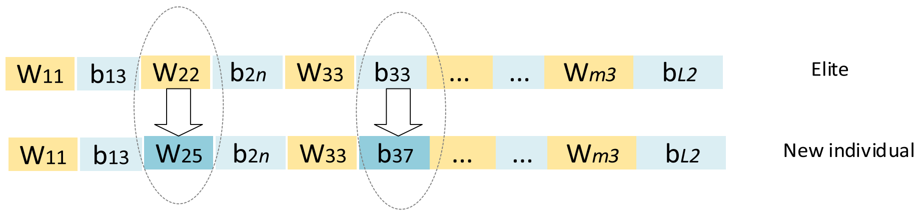

W11 b13 W22 b2n W33 b33 ... ... Wm3 bL2 Elite
W11 b13 W25 b2n W33 b37 ... ... Wm3 bL2 New individual

Fig. 6 Evolution strategy of elites

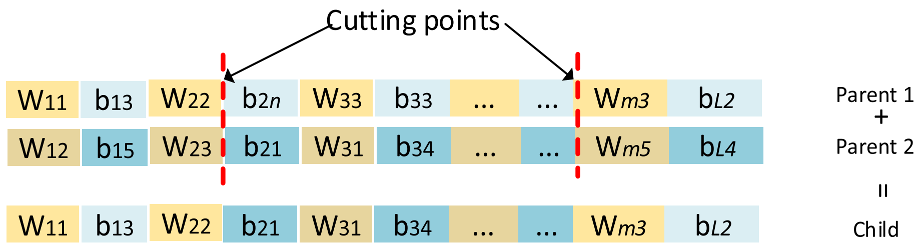

Cutting points
W11 b13 W22 b2n W33 b33 ... ... Wm3 bL2
W12 b15 W23 b21 W31 b34 ... ... Wm5 bL4
W11 b13 W22 b21 W31 b34 ... ... Wm3 bL2
Parent 1
+
Parent 2
=
Child

Fig. 7 Evolution strategy of worst individuals

example, as shown in Fig. 8, a segment consisting of 3 matrices is crossed over.

The pseudo-code of the global search phase is as follows:

```
Global search
Randomly generate the local population, i.e., a $n \times p$ matrix.
Randomly select the candidate solutions for each layer to form the global population.
Calculate the fitness of individuals in the global population
While $iter < n\_iteration$ do
    While $i < n\_global\_pop$:
        For elites:
            Randomly select two points and execute mutation operation
        End for
        For worst individuals:
            Randomly select two cutting points and execute crossover operation with elites
        End for
        For others:
            Randomly select one point and exchange a fixed number of parameters crossover with elites
        End for
        If better, replace the old one.
        $i=i+1$
    End while
    $iter= iter+1$
End while
```

Local search phase In addition to computing the gradient, we add a perturbation to the parameters to help the algorithm get out of local optimum (Colas 2020; Lehman et al. 2018; Ashfahani et al. 2020). At each iteration, only one weight or bias matrix is selected to add a randomly generated perturbation. Each weight or bias matrix is trained batch by batch. We also introduced the dropout strategy to avoid over-fitting. It is realized by the mutation operation which randomly select multiple values of the matrix and set it as zero. If the fitness of the new individual is better, replace the old individual with the new one and break the loop.

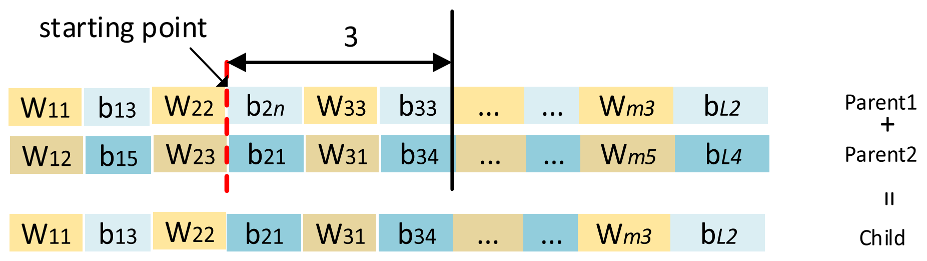
Besides, a simulated annealing algorithm-based strategy has been designed to assist the algorithm to jump out the local optimal (Bertsimas and Tsitsiklis 1993). The simulated annealing algorithm is inspired by the processes which occur during the cooling of physical systems and is a probabilistic technique for approximating the global optimum. If a liquid is cooled slowly, the atoms anneal to increase the size of its crystals. The slow cooling process can be interpreted as a slight decrease in the probability of accepting worse solutions. The algorithm will start with an initial, often randomly generated solution *X* and evolve it to propose an updated solution *X*' according to some evolution strategies. If *X*' has better fitness, the algorithm replaces *X* with *X*'. However, if *X*' is worse than *X*, it will be accepted with probability:

$$P = \exp(-\nabla H/T) \tag{5}$$

where ∇H is the change in fitness and T is a control parameter. Thus, in the local search phase, even if the fitness of the generated individual is worse, it is still possible to be accepted.

By combining the strength of the gradient descent and random search, the convergence and efficiency of the proposed method can be guaranteed. The pseudo-code of the local search phase is as follows:

```
Local search
Select the best one obtained from the above steps for local search
Perturb the large datasets and divide the data into small batches
For the weights of each layer:
    While $i < n\_batches$ do
        While $j <$ n_population do
            Add a random perturbation to the weights of current layer
            Randomly selected one parameter of the matrices and set it as zero.
            If better, replace the old one and break.
            If worse, accept the new solution according to a probability α
            $j=j+1$
        End while
        Update the weights using the gradient descent method.
        $i=i+1$
    End while
End for
```



starting point
3
W11 b13 W22 b2n W33 b33 ... ... Wm3 bL2
W12 b15 W23 b21 W31 b34 ... ... Wm5 bL4
W11 b13 W22 b21 W31 b34 ... ... Wm3 bL2
Parent1
+
Parent2
=
Child

Fig. 8 Evolution strategy of other individuals

In summary, the proposed method is a two-phase algorithm. The global search strategies are utilized to increase the exploration ability while the local search strategies are introduced to increase the exploitation ability. The local search can also help avoid the vanishing gradient and local optimum by adding a random Gaussian perturbation. To increase the robustness and the efficiency of the algorithm, the gradient descent method is used to update the weights of the neural networks.

5 Case study

To validate the proposed solution technique, numerical experiments have been conducted. A dataset from Syngenta Crop Challenge is used as a case study. The target is to predict the crop yield using the genetic variables and environment variables. The codes are implemented in Python. All the experiments are carried out on a computer with 3.1 GHz CPU and NVIDIA GTX1080 GPU.

5.1 Data

The Syngenta Crop Challenge dataset consists of 148,452 samples which were collected from 2009–2016. The dataset consists of two types of data, i.e., genetic data and environment data. The genetic data is represented by three discrete values, i.e., "− 1," "0" and " + 1." The environment data includes soil data and weather data, which are represented by continuous values. The total number of variables is 13,550. The target of the project is to predict the difference between yield and the benchmark result.

Before using the dataset to train the neural network, we should deal with the missing values in genetic data. We dropped the variables whose data are missing for more than 30% sample. Next, we deleted the samples that have more than 30% missing values. For the other missing values, we adopted the imputation method, i.e., filling the value with mean values. Since the number of variables is very large, a linear regression model is used to select the most relevant variables. As a result, one hundred and fifty-four variables have been selected. We used 20% of the samples as the testing dataset.

5.2 Experimental parameters and result analysis

The neural network we used in the experiment consists of 20 layers. The first hidden layer contains 128 neurons. Each of the other hidden layers contains 64 neurons. The activation function is Tanh function.

In the first experiment, the neural network is trained by three gradient-based methods, i.e., SGD, Adam and Adagrad. SGD is widely used due to its ease of implementation and fast computing speed. The stochastic nature of SGD can make it possible to hop out of local optimum. Compared with SGD, Adam usually performs well in terms of convergence since it can get the speed from moving average. Adagrad keeps tracks of the gradient squared and adapts the gradient in different directions, which can improve the robustness effectively. The learning rate is 0.003. The batch size is set to 800. The number of iterations is 1000. As shown in Fig. 9, before 80 iterations, the descent speed of the three methods is very fast. The RMSE is decreased by about 20%, from 20.1 to 15.6. After 80 iterations, the speed slows down and stop updating. In terms of the quality of initial points, Adagrad is 17.74, which is better than 20.14 of gradient descent and 19.93 of Adam. However, they all converge at about 15.6.

In the second experiment, the neural network is trained by using the proposed method. The number of individuals in the local population is set as 10. The number of individuals in the global population is 120. The number of iterations in the global search phase is set as 120 iterations. It should be noted that it does not make any sense to compare the iterations in the two experiments since they are programmed in different ways.

The running process is shown in Fig. 9. Compared with the gradient descent methods, the quality of the initial point of the proposed method is much better. The reason is that, in the global search phase, the RMSE is decreased by about 11%, from 18.163 to 16.250. When the decreasing speed slows down, the local search strategy is performed. In the initial phase of local search, the RMSE is decreased dramatically, from 16.250 to 15.27. Then it decreases in a gentle way until it converges at 14.874.

The above experiment has been repeated for five times. The result comparison is shown in Table 1. Compared with the traditional gradient descent method, our proposed

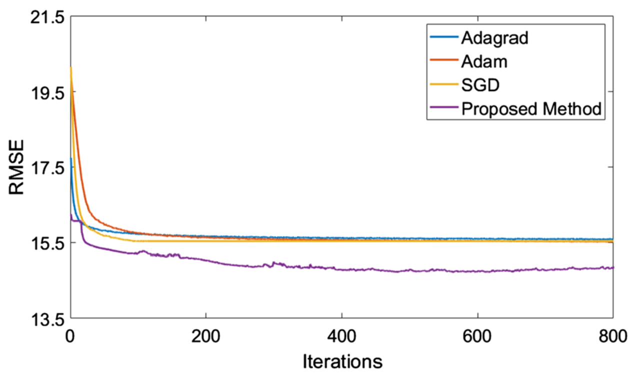


Fig. 9 Training process of gradient-based methods

method improves the training RMSE and testing RMSE by 4% and 5%, respectively. In terms of algorithm stability, the training standard deviation (SD) of the proposed is better than that of Adam and Adagrad. The testing SD of the proposed is comparable to that of other methods. The results proved the effectiveness and the robustness of the proposed method.

The proposed method has been compared with other techniques that aim to improve the quality of gradients, i.e., Leaky ReLU, gradient clipping and L2 normalization. The parameter of Leaky ReLU is set to 0.01, meaning the gradient is equal to 0.002 when the unit is not activated. The gradient clipping value is specified as 0.5, meaning that if a gradient value was less than − 0.5, it is set to − 0.5 and if it is more than 0.5, then it will be set to 0.5. The parameter of L2 normalization is set to 0.9, meaning that the sum of squares of the gradients is up to 0.9. Since SGD performs better than Adagrad and Adam according to Table 1, it is used as the baseline optimizers. The experiment was repeated for five times. As shown in Table 2, both of the training RMSE and the testing RMSE of the proposed method are 3%–5% lower than those of other three methods. It proves that the combination of GA and the gradient descent method can help improve the algorithm performance. Besides, the training SD and the testing SD of the proposed method are also the lowest, which validates the algorithm stability.

6 Conclusions

Improving crop performance through crop breeding is one important path to alleviate the potential problems of feeding the growing population. Efficient and effective crop breeding relies on accurate crop yield prediction, which is one of the most important crop phenotypes. However, the crop phenotype is jointly determined by the genotype and environment factors through complicated interactive relationships. We proposed a GA-assisted deep learning method to predict crop yield. Different from the first-order method, i.e., gradient descent method, the proposed method combines the zeroth-order method and the gradient method to improve efficiency and robustness. Besides, it also can avoid the gradient degradation problem in training deep neural networks. We also add the batch strategy to the algorithm so that it can deal with large datasets. In the case study of Syngenta crop challenge, the experiment results show that the proposed method can reduce the RMSE by about 10%. The proposed method has also been compared with other techniques that are used to improve the gradient quality. The results show that the proposed method has improved prediction accuracy as well as algorithm stability.

On the basis of this paper, plant traits prediction using neuroevolution deep learning methods can be further investigated. However, the current approach is designed

Table 1 Comparison results among SGD, Adagrad, Adam and the proposed method

	Training RMSE	Training SD	Testing RMSE	Testing SD
SGD	15.571	0.073	15.422	0.123
Adagrad	15.620	0.114	15.587	0.074
Adam	15.608	0.116	15.619	0.094
Proposed method	14.944	0.093	14.957	0.094

SD: standard deviation

Table 2 Comparison results among Leaky ReLU, Gradient Clipping, L2 normalization and the proposed method (SD: standard deviation)

	Training RMSE	Training SD	Testing RMSE	Testing SD
SGD + LeakyReLU	15.367	0.117	15.422	0.114
SGD + Gradient Clipping	15.644	0.138	15.664	0.117
SGD + L2 Normalization	15.748	0.100	15.740	0.128
Proposed method	14.944	0.093	14.957	0.094

only for the fully connected deep neural network. Future work will include the investigation of the effect of perturbation on the evolution of the neural network. We are going to apply the proposed method to different deep neural networks, e.g., convolutional neural network.

Declarations

Conflict of interest The authors certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affi liations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

Affes Z, Hentati-Kaffel R (2019) Forecast bankruptcy using a blend of clustering and MARS model: case of US banks. Ann Oper Res 281(1–2):27–64

Ashfahani A, Pratama M, Lughofer E, Ong Y-S (2020) DEVDAN: deep evolving denoising autoencoder. Neurocomputing 390:297–314

Assunção F, Lourenço N, Machado P, Ribeiro B (2019) *Fast denser: efficient deep neuroevolution.* In: Paper presented at the European Conference on Genetic Programming.

Baral C, Fuentes O, Kreinovich V (2018) Why deep neural networks: a possible theoretical explanation. Constraint Programming and Decision Making: Theory and Applications. Springer, New York, pp 1–5

Bau D, Zhu J-Y, Strobelt H, Lapedriza A, Zhou B, Torralba A (2020) Understanding the role of individual units in a deep neural network. Proc Natl Acad Sci 117(48):30071–30078

Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5(2):157–166

Bertsimas D, Tsitsiklis J (1993) Simulated annealing. Stat Sci 8(1):10–15

Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*. Springer pp. 177–186

Bremermann HJ (1962) Optimization through evolution and recombination. Self-Organ Syst 93:106

Chandwani V, Agrawal V, Nagar R (2015) Modeling slump of ready mix concrete using genetic algorithms assisted training of Artificial Neural Networks. Expert Syst Appl 42(2):885–893. https://doi.org/10.1016/j.eswa.2014.08.048

Colas C, Madhavan V, Huizinga J, Clune J (2020) *Scaling map-elites to deep neuroevolution.* In: Paper presented at the Proceedings of the 2020 Genetic and Evolutionary Computation Conference

Comstock R, Moll RH (1963) Genotype-environment interactions. Stat Genet Plant Breed 982:164–196

Dahl GE, Sainath TN, Hinton GE (2013) *Improving deep neural networks for LVCSR using rectified linear units and dropout.* In: Paper presented at the 2013 IEEE international conference on acoustics, speech and signal processing

Davis L (1991) Handbook of genetic algorithms

Davis N (2017) *The selfish gene*: Macat Library.

de Oliveira Florentino H, Irawan C, Jones DF, Cantane DR, Nervis JJ (2018) A multiple objective methodology for sugarcane harvest management with varying maturation periods. Ann Oper Res 267(1–2):153–177

Des Marais DL, Hernandez KM, Juenger TE (2013) Genotype-by-environment interaction and plasticity: exploring genomic responses of plants to the abiotic environment. Annu Rev Ecol Evol Syst 44:5–29

Gerland P, Raftery AE, Ševčíková H, Li N, Gu D, Spoorenberg T, Lalic N (2014) World population stabilization unlikely this century. Science 346(6206):234–237

Han J, Moraga C (1995) *The influence of the sigmoid function parameters on the speed of backpropagation learning.* In: Paper presented at the International Workshop on Artificial Neural Networks

Hanin B (2018) *Which neural net architectures give rise to exploding and vanishing gradients?* In: Paper Presented at the Proceedings of the 32nd international Conference on Neural Information Processing Systems

Hinton G, Deng L, Yu D, Dahl GE, Mohamed A-R, Jaitly N, Sainath TN (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–97

Holland JB (2006) Estimating genotypic correlations and their standard errors using multivariate restricted maximum likelihood estimation with SAS proc mixed. Crop Sci 46:642–654. https://doi.org/10.2135/cropsci2005.0191

Johnson R, Zhang T (2013) *Accelerating stochastic gradient descent using predictive variance reduction.* In: Paper Presented at the Advances in Neural Information Processing Systems

Kang MS (1997) Using genotype-by-environment interaction for crop cultivar development. In: Sparks DL (ed) Advances in Agronomy. Academic Press, Cambridge, pp 199–252

Karlik B, Olgac AV (2011) Performance analysis of various activation functions in generalized MLP architectures of neural networks. Int J Artif Intell Expert Syst 1(4):111–122

Kaveh F, Tavakkoli-Moghaddam R, Triki C et al (2021) A new bi-objective model of the urban public transportation hub network design under uncertainty. Ann Oper Res 296:131–162. https://doi.org/10.1007/s10479-019-03430-9

Kingma DP, Ba J (2015) *Adam: a method for stochastic optimization.* In:Paper presented at the ICLR

Kitano H (1990) Designing neural networks using genetic algorithms with graph generation system. Complex Syst 4(4):461–476

Lehman J, Miikkulainen R (2013) Neuroevolution. Scholarpedia 8(6):30977. https://doi.org/10.4249/scholarpedia.30977

Lehman J, Chen J, Clune J, Stanley KO (2018) *Safe mutations for deep and recurrent neural networks through output gradients.* In: Paper Presented at the Proceedings of the Genetic and Evolutionary Computation Conference

Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26

Mei Y, Li X, Yao X (2014) Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. IEEE Trans Evol Comput 18(3):435–449

Mukkamala MC, Hein M (2017) *Variants of rmsprop and Adagrad with logarithmic regret bounds.* In: Paper presented at the International Conference on Machine Learning

Nair V, Hinton GE (2010) *Rectified linear units improve restricted boltzmann machines.* In: Paper Presented at the Icml

Olimov B, Karshiev S, Jang E, Din S, Paul A, Kim J (2020) Weight initialization based-rectified linear unit activation function to improve the performance of a convolutional neural network model. Concurrency Computat Pract Exper. e6143. https://doi.org/10.1002/cpe.6143

Risi S, Stanley KO (2019) *Deep neuroevolution of recurrent and discrete world models*. In: Paper presented at the Proceedings of the Genetic and Evolutionary Computation Conference

Robinson MR, Beckerman AP (2013) Quantifying multivariate plasticity: genetic variation in resource acquisition drives plasticity in resource allocation to components of life history. Ecol Lett 16(3):281–290

Ruehle F (2017) Evolving neural networks with genetic algorithms to study thestring landscape. J. High Energ. Phys. 38. https://doi.org/10.1007/JHEP08(2017)038

Salhi A, Alsoufi G, Yang X (2019) An evolutionary approach to a combined mixed integer programming model of seaside operations as arise in container ports. Ann Oper Res 272(1–2):69–98

Samek W, Binder A, Montavon G, Lapuschkin S, Müller K-R (2016) Evaluating the visualization of what a deep neural network has learned. IEEE Trans Neural Netw Learn Syst 28(11):2660–2673

Schaeffer L, Jamrozik J (2008) Random regression models: a longitudinal perspective. J Anim Breed Genet 125(3):145–146

Schwardt M, Fischer K (2009) Combined location-routing problems—a neural network approach. Ann Oper Res 167(1):253

Sherstinsky A (2020) Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Phys D Nonlinear Phenom 404:132306

Stinchcombe JR, Kirkpatrick M, Group, F.-v. T. W (2012) Genetics and evolution of function-valued traits: understanding environmentally responsive phenotypes. Trends Ecol Evol 27(11):637–647

Wang D, Oh S-K, Kim E-H (2018) Design of space search-optimized polynomial neural networks with the aid of ranking selection and L2-norm regularization. J Electr Eng Technol 13(4):1724–1731

Whitley D (1994) A genetic algorithm tutorial. Stat Comput 4(2):65–85

Whitley D, Starkweather T, Bogart C (1990) Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Comput 14(3):347–361

Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447

Yarotsky D (2017) Error bounds for approximations with deep ReLU networks. Neural Netw 94:103–114

Zhang J, He T, Sra S, Jadbabaie A (2019) *Why gradient clipping accelerates training: a theoretical justification for adaptivity*. In: Paper Presented at the International Conference on Learning Representations