Evolving Neural Networks with Interval Weights by Means of Genetic Algorithm

HIDEHIKO OKADA* Non-member

(Received May 15, 2013, revised June 30, 2013)

Abstract: In this paper, the author proposes an extension of genetic algorithm (GA) for evolving neural networks with interval-valued weights and biases. In the proposed extension, genotype values are not real numbers but intervals. Evolutionary processes in GA are extended so that the processes can handle interval-valued genotypes. Experimental results show that interval neural networks evolved by the proposed method can model target interval functions well despite the fact that no training data is explicitly provided.

Keywords: evolutionary algorithm, neuroevolution, genetic algorithm, neural network, interval.

1. Introduction

A multi-layered feed forward neural network (NN) with interval-valued weights and biases was proposed in literature [1]. A supervised learning method for the interval NN (INN) was also proposed [1] as an extension of the traditional back propagation (BP). The INN approximately models an interval function \( Y = F(x) \), where \( Y \) is an interval number and \( x \) is a real vector, by learning data \((x_q, Y_q), q = 1, 2, \ldots\). The INN can learn the data in which \( \{Y_q\} \) include both of real numbers and interval numbers, because a real number can be specified as an interval number with zero width (i.e., with the same value of upper and lower limits). As the learning method for the INN, the extended back propagation was proposed but no method has been proposed which does not require training data.

Besides, evolutionary algorithms (EAs) have recently been applied to the reinforcement learning of NNs, known as neuroevolution (NE) [2]∼[5]. In NE, weights and biases are tuned by evolutionary operations, not by the BP algorithm. Because NE does not utilize BP, NE does not require errors between NN output values and their target signals. Thus, NE is applicable for problems in which the error function is difficult or impossible to be determined. EAs have been applied to NE of traditional NNs with real-valued weights and biases, where the genotypes (chromosomes) consist of real numbers or bit strings that encode real numbers. The ordinary EAs have not employed interval numbers as their genotype values because their evolutionary operators are designed to handle genotypes with crisp values.

This paper proposes an extension of genetic algorithm (GA) for handling interval-valued genotypes. The extended GA can be applied directly to interval-valued optimization problems by employing interval variables in the optimization problem as genotype values. The author applies the extended GA to the evolution of INNs. Accuracy of evolved INNs in modeling hidden target interval functions is experimentally evaluated. Besides, an interval value can be specified by its lower and upper limits or its center and width. The extended GA can adopt either of the two model, the lower and upper model or the center and width model, to specify interval genotype values. These two models are compared to investigate which one will contribute more for the extended GA to evolve better INNs.

2. Neural Networks with Interval Weights and Biases

The INN employed in our research is the same as in the literature [1], which is a three-layered feed forward NN with interval weights and biases. Figure 1 shows its structure. An INN receives an input real vector \( x \) and calculates its output interval value \( O \) (for the sake of simplicity, the output layer includes a single unit) as follows.

\[
\text{Input Layer:} \quad o_i = x_i. \quad (1)
\]

\[
\text{Hidden Layer:} \quad Net_j = \sum_i W_{ij} o_i + \Theta_j, \quad (2)
\]

\[
O_j = f(Net_j). \quad (3)
\]

\[
\text{Output Layer:} \quad Net = \sum_j W_j O_j + \Theta, \quad (4)
\]

\[
O = f(Net). \quad (5)
\]

In Eqs. (1)∼(5), \( x_i \) and \( o_i \) are real numbers, while \( Net_j, Net, W_{ij}, W_j, \Theta_j, \Theta, O_j \) and \( O \) are interval numbers. \( f(x) \) is...
the unit activation function which is typically the sigmoidal one: \( f(x) = 1/(1 + e^{-x}) \). The feed-forward calculation of the INN is based on the interval arithmetic [6] (for more detail, see the literature [1]). The sigmoidal function maps an input interval number to an output interval number as shown in Fig. 2.

The INN includes \( mn + m \) weights (i.e., \( mn \) weights between \( n \) input units and \( m \) hidden units, and \( m \) weights between \( m \) hidden units and an output unit) and \( m + 1 \) biases (= the total number of units in the hidden and output layers). Thus, the INN includes \( mn + 2m + 1 \) interval variables in total. The interval GA (IGA) handles these interval variables as a genotype \( X = (X_1, X_2, \ldots, X_D) \) where \( X_i \) is an interval number and \( D = mn + 2m + 1 \). Each interval \( X_i \) can be denoted by either of the two parameters: \( X_i = [x_{iL}, x_{iU}] \) or \( X_i = (x_{iL}, x_{iU}) \) where \( x_{iL}, x_{iU}, x_{i}, x_{i} \) and \( y_{i} \) denote the lower, upper, center and width of \( X_i \) respectively. In this paper, the author denotes the former (latter) as LU (CW) model.

3. Genetic Algorithm with Interval-valued Genotypes

The IGA includes the same processes as those in the ordinary GA (Fig. 3), where initialization of population, fitness evaluation, crossover and mutation are extended so that these processes can handle interval-valued genotypes.

3.1 Initialization of Population

In the initialization process, \( X_1, X_2, \ldots, X_P \) are randomly initialized where \( P \) is the population size. Because the elements in \( X_a \) (i.e., \( X_{a1}, X_{a2}, \ldots, X_{aD} \)) are weights and biases in an INN in this research, smaller absolute values are preferable as initial values for \( X_{ai} \). Thus, the initial values are randomly sampled from the normal distribution \( N(0, \varepsilon) \) or uniformly from an interval \([−\varepsilon, \varepsilon] \) where \( \varepsilon \) is a small positive number. In the case of employing the LU model, two values are sampled per \( X_{ai} \): the smaller (larger) one is set to \( x_{ai}^L (x_{ai}^U) \). In the case of employing the CW model, two values are sampled per \( X_{ai} \): one of the two values is set to \( x_{ai}^C \) and the absolute value of the other is set to \( x_{ai}^W \).

3.2 Fitness Evaluation

To evaluate fitness of an INN as a phenotype instance of the corresponding genotype instance \( X_a = (X_{a1}, X_{a2}, \ldots, X_{aD}) \) where \( X_a \in X_1, X_2, \ldots, X_P \), the INN is supplied with several samples of input real vectors and calculates output values. The input values are sampled within the variable domain of application problem. Fitness of the genotype instance \( X_a \) is evaluated based on the output values. The method for scoring the fitness based on the output values depends on the problem to which the INN is applied. For example, in a case where the INN is applied to controlling an automated system, some performance measure of the system can be used as the fitness score of the genotype instance corresponding to the INN.

3.3 Crossover

Let us denote genotypes of two parents as \( X_a, X_b \) and an offspring genotype as \( X_c \). \( X_a \) and \( X_b \) can be sampled from the population in the same manner as the ordinary GA. In the case of employing the LU model,

\[
X_a = (X_{a1}, X_{a2}, \ldots, X_{aD}), X_{ai} = [x_{ai}^L, x_{ai}^U] \quad (6)
\]

\[
X_b = (X_{b1}, X_{b2}, \ldots, X_{bD}), X_{bi} = [x_{bi}^L, x_{bi}^U] \quad (7)
\]

\[
X_c = (X_{c1}, X_{c2}, \ldots, X_{cD}), X_{ci} = [x_{ci}^L, x_{ci}^U] \quad (8)
\]

Values of \( x_{ci}^L \) and \( x_{ci}^U \) in the offspring \( X_c \) can be determined by applying a crossover operator for the ordinary real GA. Suppose the operator is the blend crossover [7]. In this case, \( x_{ci}^L \) is uniformly randomly sampled from the interval \([\min(x_{ai}^L, x_{bi}^L) − a|x_{ai}^L − x_{bi}^L|, \max(x_{ai}^L, x_{bi}^L) + a|x_{ai}^L − x_{bi}^L|] \), where \( \min(x, y) \) and \( \max(x, y) \) is the smaller and the larger of \( x, y \) respectively. Similarly, \( x_{ci}^U \) is uniformly randomly sampled from the interval \([\min(x_{ai}^U, x_{bi}^U) − a|x_{ai}^L − x_{bi}^L|, \max(x_{ai}^L, x_{bi}^L) + a|x_{ai}^L − x_{bi}^L|] \). Note that \( x_{ci}^L \) must not be
smaller than \( x_{ij}^L \) because \( x_{ij}^L \) and \( x_{ij}^U \) are the lower and upper limits of the interval \( X_{ij} \). If \( x_{ij}^U \) becomes smaller than \( x_{ij}^L \) as the result of applying the blend crossover, then \( x_{ij}^L \) and \( x_{ij}^U \) must be repaired so that \( X_{ij} \) is valid. The repair method can be either of the followings:

- the value of \( x_{ij}^U \) is assigned to \( x_{ij}^L \),
- the value of \( x_{ij}^L \) is assigned to \( x_{ij}^U \),
- the mean value of \( x_{ij}^L \) and \( x_{ij}^U \) is calculated and assigned to both of \( x_{ij}^L \) and \( x_{ij}^U \), or
- the two values for \( x_{ij}^L \) and \( x_{ij}^U \) are switched.

In the case of employing the CW model,

\[
X_a = (X_{a,1}, X_{a,2}, \ldots, X_{a,D}), X_{a,i} = (x_{a,i}^w, x_{a,i}^e), \quad (9)
\]
\[
X_b = (X_{b,1}, X_{b,2}, \ldots, X_{b,D}), X_{b,i} = (x_{b,i}^w, x_{b,i}^e), \quad (10)
\]
\[
X_z = (X_{z,1}, X_{z,2}, \ldots, X_{z,D}), X_{z,i} = (x_{z,i}^e, x_{z,i}^w). \quad (11)
\]

Values of \( x_{z,i}^e \) and \( x_{z,i}^w \) in the offspring \( X_z \) can be determined in the same manner as those for the LU model: \( x_{z,i}^e \) is uniformly randomly sampled from the interval \([\min(x_{a,i}^e, x_{b,i}^e) - \delta, \max(x_{a,i}^e, x_{b,i}^e) + \delta] \). Similarly, \( x_{z,i}^w \) is uniformly randomly sampled from the interval \([\min(x_{a,i}^w, x_{b,i}^w) - \delta, \max(x_{a,i}^w, x_{b,i}^w) + \delta] \). Note again that \( x_{z,i} \) must not be negative because \( x_{z,i} \) is the width of the interval \( X_{z,i} \). If \( x_{z,i}^w \) becomes negative as the result of applying the blend crossover, then \( x_{z,i} \) must be repaired so that \( X_{z,i} \) is valid. The repair method can be either of the followings:

- the value of \( x_{z,i}^w \) is assigned to 0, or
- the absolute value of \( x_{z,i}^w \) is assigned to \( x_{z,i}^w \).

3.4 Mutation Values in the offspring genotypes are mutated under the predetermined mutation probability. In the IGA, each offspring \( X_z \) is a vector \((X_{z,1}, X_{z,2}, \ldots, X_{z,D})\) where \( X_{z,i} \) is an interval number specified by the two real parameters: \( X_{z,i} = [x_{z,i}^L, x_{z,i}^U] \) or \( X_{z,i} = (x_{z,i}^L, x_{z,i}^U) \). The two parameter values of \( X_{z,i} \) which is selected under the probability are mutated by being added (or replaced) with random real numbers to the current values where the random numbers are sampled from the normal distribution \( N(0, \delta) \) or uniformly from an interval \([-\delta, \delta] \) and \( \delta \) is also a small positive number as \( \varepsilon \). After the mutation of \( X_{z,i} \), \( x_{z,i}^L \) may become smaller than \( x_{z,i}^L \) with the LU model or \( x_{z,i}^w \) may become negative with the CW model. Such invalid interval numbers are repaired by the same method applied in the crossover process.

4. Experimental Evaluation

In this section, the author experimentally evaluates the ability of the IGA by applying it to evolution of INNs. The INNs are challenged to accurately model interval functions. The accuracy is evaluated in Subsection 4.1 by using three target functions. Besides, the LU and CW models are compared in Subsection 4.2 to investigate which model better contributes to the IGA in evolving INNs.

4.1 Accuracy in Modeling Interval Functions In the experiment, the author designs three functions and employs them as the modeling targets for INNs. For simplicity, the input \( x \) of the functions is not a real vector but a real scalar (so that the INN includes only a single input unit) and \( 0 \leq x \leq 1 \), as in the literature [1]. The output values of the functions are interval numbers. The three functions \( F_1(x) = [F_1(x)^L, F_1(x)^U] \), \( F_2(x) = [F_2(x)^L, F_2(x)^U] \), and \( F_3(x) = [F_3(x)^L, F_3(x)^U] \) function.
and $F_3(x) = [F_3^L(x), F_3^U(x)]$ are as follows:

$$
F_1(x)^L = 0.2 \sin(2\pi x) - 0.1x^2 + 0.4 \quad (12)
$$
$$
F_1(x)^U = 0.2 \sin(2\pi x) + 0.1x^2 + 0.6 \quad (13)
$$
$$
F_2(x)^L = 0.2 \sin(2\pi x) + 0.2x^2 + 0.2 \quad (14)
$$
$$
F_2(x)^U = 0.2 \sin(2\pi x) - 0.2x^2 + 0.7 \quad (15)
$$
$$
F_3(x)^L = 0.15 \sin(2\pi x) + 0.3 \quad (16)
$$
$$
F_3(x)^U = 0.1 \sin(3\pi x) - 0.1x + 0.7 \quad (17)
$$

Figures 4–6 show these three functions, where $F(x)^L$ and $F(x)^U$ denote $F(x)^L$ and $F(x)^U$, i.e., the lower and upper limits of $F(x)$.

The INN and the IGA are designed as follows.

- **INN:**
  - #units: 1 input, 10 hidden, 1 output.
  - Initial values for $x^L_{w,i}$, $x^U_{w,i}$, $x^L_{o,i}$, and $x^U_{o,i}$ uniformly random within $[-0.01, 0.01]$.
  - Initial values for $x^w_{o,i}$ uniformly random within $[0.0, 0.01]$.
  - $-10.0 \leq x^L_{w,i}, x^U_{w,i}, x^L_{o,i}, x^U_{o,i} \leq 10.0$.
  - $0.0 \leq x^w_{o,i} \leq 10.0$.

- **IGA:**
  - #Total INNs evolved: 1,000,000.
  - Population size and #generation: (100 and 10,000) or (500 and 2,000).
  - $\alpha$ for the blend crossover: 0.5.
  - Mutation probability: 0.01 for each of the elements $X_{c,1}, X_{c,2}, \ldots, X_{c,D}$ in a genotype instance $X_c$.
  - Random values for mutation: $N(0, 1)$ for $x^L_{w,i}, x^U_{w,i}, x^L_{o,i}$, and $N(0, 1)$ for $x^w_{o,i}$.
  - Elitism: best 10 elite genotype instances are copied to the next generation.
  - Tournament size for sampling two parent genotype instances: 5% of the population.

The number of generations is 10,000 (or 2,000) for IGA with 100 (or 500) solutions so that the total number of INNs evolved is consistently 1,000,000.

Genotype instances $X_1, X_2, \ldots, X_k$ are ranked as follows. An INN which corresponds to a genotype instance $X_k$ is supplied with a real input value $x_k$ and calculates its output interval number $O_k$, $x_k$ is sampled within the input domain $[0, 1]$ as $x_k = 0.0, 0.01, 0.02, \ldots, 1.0$. Besides, each value of $x_k$ is supplied to the target function $F(x)$ and the output interval number $F(x_k)$ is obtained. Then, the error $e_r$ for the input $x_k$ is calculated as:

$$
e_r = (o^L_r - f^L_k)^2 + (o^U_r - f^U_k)^2 \quad (18)$$

where,

$I$IAE Journal, Vol.1, No.1, 2013
of updating the weights and biases but only for ranking the genotype instances: any output value of the target functions is completely hidden from the IGA reproduction process.

Figures 7–9 show the results of this experiment. Figure 7 shows the output interval function of the best INN among the total 20,000,000 INNs (= 1,000,000 INNs in each run) evoked by the IGA for modeling $F_1(x)$. Figures 8 and 9 shows those for modeling $F_2(x)$ and $F_3(x)$ respectively in the same manner as Fig. 7. In Figs. 7–9,

- $F(x)_L$ and $F(x)_U$ are the same as those in Figs. 4–6, and
- $INN(x)_L$ and $INN(x)_U$ denote the lower and upper limits of the INN output interval number.

These results shown in Figs. 7–9 reveal that the best INNs evolved by the IGA approximate well to their target functions, despite the fact that no training data is explicitly provided.

### 4.2 Comparison of Two Models for Interval Genotype Values

As described in Subsection 3.3, the constraints for the two real parameters of an interval number (i.e., the lower and upper limits or the center and width) are different, and thus the methods for repairing constraint-violating values are also different between the LU and CW models. This difference may affect the performance of the IGA in searching solutions because the repairs restrict changes of genotype values. In this section, the author compares the two models to investigate which model contributes better for the IGA to find better solutions, based on the result of numerical experiment in the last section.

Figure 10 shows the error values of the best INN for $F_1(x)$ among each number of INNs evolved (e.g., 500,000 INNs are evolved in total at the 5,000th generation with the population size of 100). In this figure, “LU (100)” denotes the result with the LU model and the population size of 100. “LU (500)” , “CW (100)” and “CW (500)” denote their results in the same manner as “LU (100)”. The error values are the averaged ones over the five runs. Figure 11 and 12 show the error values for $F_2(x)$ and $F_3(x)$ respectively, in the same manner as Fig. 10.

Figures 10–12 reveal that, for all of the three target functions, the CW model contributed better than the LU model did with the population sizes of both 100 and 500, i.e., after the 1,000,000 INNs were evolved, the dotted curves for the CW model went below the solid curves for the LU model. This finding suggests that the CW model is better for the IGA to employ as the model for specifying interval genotype values.

To investigate the reason why the IGA could evolve better INNs with the CW model, the author counts the number of repairs for the invalid genotype values. As described in Subsections 3.3, $x_{2,1}^L$, $x_{2,1}^U$, or $x_{3,1}^w$ violate the constraints then the new values are repaired. Such repairs may interfere with the evolution of INNs because the repairs restrict modification of weights and biases. Thus, a smaller number of repairs will be better in evolving INNs. Table 1 shows the numbers of repairs where the values in the table are the averaged ones over the five runs under each condition. For example, the IGA with the LU model and the population size of 100 required $1.64E + 6(1.64 \times 10^6)$ repairs for $F_1(x)$, while the IGA with the CW model and the population size of 100 required $3.65E + 5$ repairs for $F_1(x)$. Table 1 clearly shows...
that the CW model required less repairs than the LU model did, which will be a reason for the fact that the CW model could contribute to evolve better INNs.

5. Conclusion

In this paper, the author proposed the interval-valued GA (IGA), an extension of the GA, and applied it to the neuroevolution of interval-valued neural networks. In the proposed extension, values in the genotypes are not real numbers but interval numbers. To handle the interval-valued genotypes, the IGA extends its processes of initialization of populations, crossover and mutation.

The IGA was challenged to evolve INNs which modeled hidden interval functions. The experimental results showed that the neural networks evolved by the IGA approximated the target functions well, despite the fact that no training data was explicitly provided. In addition, the results revealed that the CW model contributed slightly better to the IGA than the LU model did. The reason would be because the IGA with the CW model required less number of repairs for invalid genotype values than the IGA with the LU model did.

In the future work, the author will further evaluate the ability of the IGA by applying it to problems other than neuroevolution.

References

[1] H. Ishibuchi, H. Tanaka and H. Okada, “An architecture of neural networks with interval weights and its application to fuzzy regression analysis”, Fuzzy Sets and Systems, vol.57, no.1, pp.27-39, 1993.

[2] D.B. Fogel, L.J. Fogel and V.W. Porto, “Evolving neural networks”, Biological Cybernetics, vol.63, issue 6, pp.487-493, 1990.

[3] X. Yao, “Evolving artificial neural networks”, Proc. of the IEEE, vol.87, no.9, pp.1423-1447, 1999.

[4] K.O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies”, Evolutionary Computation, vol.10, no.2, pp.99-127, 2002.

[5] D. Floreano, P. Durr and C. Mattiussi, “Neuroevolution: from architectures to learning”, Evolutionary Intelligence, vol.1, no.1, pp.47-62, 2008.

[6] G. Alefeld and J. Herzberger, “Introduction to Interval Computation”, Academic Press, 1983.

[7] L.J. Eshelman and J.D. Schaffer, “Real-coded genetic algorithms and interval-schemata”, Foundation of Genetic Algorithms, pp.187-202, 1992.

Acknowledgment

This research was supported by Kyoto Sangyo University Research Grant.

Hidehiko Okada (Non-member) is currently a Professor with the Department of Computer Science and Engineering, Kyoto Sangyo University, Kyoto, Japan. He received the B.S. degree in industrial engineering and the Ph.D. degree in engineering from Osaka Prefecture University in 1992 and 2003, respectively. He had been a researcher with NEC Corporation from 1992 to 2003, and since 2004 he has been with the university. His current research interests include computational intelligence and human-computer interaction. He is a member of Information Processing Society of Japan, Institute of Electronics, Information and Communication Engineers, Society of Instrument and Control Engineers, Japanese Society for Artificial Intelligence, Japan Society for Fuzzy Theory and Intelligent Informatics and Human Interface Society. He received the best paper award in the 1st International Conference on Industrial Application Engineering 2013.