Adv Syst Sci Appl 2017; **3**:42–48
Published online at http://ijassa.ipu.ru/ojs/ijassa/article/view/503

Interval-valued Data Forecasting Using Dual-parametric Neural Networks

Yuliya. E. Polozova[1], Pavel. V. Saraev[1]

[1]*Lipetsk State Technical University, Lipetsk, Russia*

Abstract: This study is focused on the approach to modelling and forecasting interval-valued data using a dual-parametric neural network (DPNN). This concept was proposed as a subclass of interval neural networks that contains two types of parameters: real and interval ones. This approach makes it possible to get guaranteed inclusion of an exact (single value) solution into interval calculation results. In this paper we intend to give a theoretical overview of previous research on the subject and description of the new developed methods and algorithms for learning DPNN. The experiments demonstrate that interval calculation results obtained by using the proposed approach include of exact solution at least in 60% of cases.

Keywords: interval neural network, forecasting, dual-parametric neural network, interval-valued data, parametric identification

1. INTRODUCTION

Today neural networks are successfully applied to many different problems, particularly time series data forecasting. These applications usually use sets of single point values as input and output data. But in many applications it is more natural to use the input values and the predicted results in the form of intervals. An interval neural network (INN), which contains interval arithmetic, is used to calculate such interval-valued data.

One of the most important features of interval analysis is the guaranteed inclusion of the exact solution into interval calculation results. Previous studies show that this feature is not achieved in the case of INN. Trying to eliminate this drawback the authors introduced dual-parametric neural network (DPNN) as a new subclass of INN. Models of this subclass contain two types of parameters: real and interval ones.

In this paper, we generalize previously developed and new learning methods and algorithms of DPNN and apply them to forecasting interval-valued time series.

2. THEORETICAL OVERVIEW

Interval neural network (INN) is a neural network that contains at least one interval parameter, namely input, output, or weight. The INNs can be used for the following reasons:

- initial data are sometimes presented in the form of interval values rather than single-point ones;
- training data sets are reduced in size due to the use of clustering analysis;
- INNs allow for making reliable accuracy estimates of calculation results.

*Corresponding author: julipolozova@yandex.ru

The existing training methods for interval neural networks are based on the backpropagation (BP) algorithm for interval data [1–5] with training error calculated as the quadratic function of quality:

$$J(w) = \frac{1}{2}\sum_{i=1}^{k} Q_i(w) = \frac{1}{2}\sum_{i=1}^{k}\Big\{(\underline{y_i(w)} - \underline{\tilde{y}_i})^2 + (\overline{y_i(w)} - \overline{\tilde{y}_i})^2\Big\} \tag{2.1}$$

where k is the number of examples in the training data set, w is the weight vector, $Q_i(w)$ is the training error on example i of the training data set, $\underline{y_i(w)}$ is the lower bound of the network‘s output interval on example i, $\underline{\tilde{y}_i}$ is the lower bound of the training output interval on example i, $\overline{y_i(w)}$ is the upper bound of the network‘s output interval on example i, $\overline{\tilde{y}_i}$ is the upper bound of the training output interval on example i.

The bounds of the output interval for neuron i (Index i is already used for training samples. It would be more clear to use another letter, say, k or j for neurons) of layer l are computed as follows:

$$\underline{y^{(l,i)}} = \sigma(\underline{net^{(l,i)}}) = \sigma\Big(\sum_{j=0,w_j\geqslant 0}^{N_{l-1}} w_j^{(l,i)}\underline{y^{(l-1,j)}} + \sum_{j=0,w_j<0}^{N_{l-1}} w_j^{(l,i)}\overline{y^{(l-1,j)}}\Big) \tag{2.2}$$

$$\overline{y^{(l,i)}} = \sigma(\overline{net^{(l,i)}}) = \sigma\Big(\sum_{j=0,w_j\geqslant 0}^{N_{l-1}} w_j^{(l,i)}\overline{y^{(l-1,j)}} + \sum_{j=0,w_j<0}^{N_{l-1}} w_j^{(l,i)}\underline{y^{(l-1,j)}}\Big) \tag{2.3}$$

where $\underline{y^{(l,i)}}$ is the lower bound of the output interval for neuron i of layer l, $\sigma(\cdot)$ is the activation function for hidden layer neurons, $\underline{net^{(l,i)}}$ the lower bound of the activation level for neuron i of layer l, N_{l-1} is the number of neurons in layer $l-1$, $w_j^{(l,i)}$ is weight j for neuron i of layer l, $\underline{y^{(l-1,j)}}$ is the lower output bound for neuron j of layer $l-1$, $\overline{y^{(l-1,j)}}$ is the upper output bound for neuron j of layer $l-1$, $\overline{y^{(l,i)}}$ is the upper output bound for neuron i of layer l, $\overline{net^{(l,i)}}$ is the upper bound of the activation level for neuron i of layer l.

The gradient of the INN training quality function is calculated as follows:

$$\frac{\partial Q_k(w)}{\partial w_j^{(l,i)}} = \begin{cases} \underline{s^{(l,i)}}\sigma'_{net}(\underline{net^{(l,i)}})\underline{\underline{y^{(l-1,j)}}} + \overline{s^{(l,i)}}\sigma'_{net}(\overline{net^{(l,i)}})\overline{y^{(l-1,j)}}, & w_j^{(l,i)} \geqslant 0, \\ \underline{s^{(l,i)}}\sigma'_{net}(\underline{net^{(l,i)}})\overline{y^{(l-1,j)}} + \overline{s^{(l,i)}}\sigma'_{net}(\overline{net^{(l,i)}})\underline{y^{(l-1,j)}}, & w_j^{(l,i)} < 0. \end{cases} \tag{2.4}$$

where $\underline{s^{(l,i)}}$, $\overline{s^{(l,i)}}$ are the lower and upper bounds of the factor determined by the recurrent procedure

$$\underline{s^{(l,i)}} = \Big(\sum_{j=1,w_j\geqslant 0}^{N_{m+1}} \underline{s^{(l+1,j)}}\sigma'_{net}(\underline{net^{(l+1,j)}})w_i^{(l+1,j)} + \sum_{j=1,w_j<0}^{N_{m+1}} \overline{s^{(l+1,j)}}\sigma'_{net}(\overline{net^{(l+1,j)}})w_i^{(l+1,j)}\Big)$$

$$\overline{s^{(l,i)}} = \Big(\sum_{j=1,w_j\geqslant 0}^{N_{m+1}} \overline{s^{(l+1,j)}}\sigma'_{net}(\overline{net^{(l+1,j)}})w_i^{(l+1,j)} + \sum_{j=1,w_j<0}^{N_{m+1}} \underline{s^{(l+1,j)}}\sigma'_{net}(\underline{net^{(l+1,j)}})w_i^{(l+1,j)}\Big) \tag{2.5}$$

with the initial condition $\underline{s^{(L,1)}} = \underline{y(w)} - \underline{\tilde{y}}$, $\overline{s^{(L,1)}} = \overline{y(w)} - \overline{\tilde{y}}$, where L is the number of layers in the neural network model.

3. PROPOSED APPROACH

3.1. Modification of error function

One of the most important advantages of interval analysis methods is the possibility to get guaranteed estimates of single-point solution in the output interval. However, function (2.1) does not guarantee that output intervals of the training examples will be fully included in the output intervals of the network. To solve this problem, a training quality function for an INN model with one output was proposed in previous research:

$$J([w]) = \sum_{i=1}^{k} Q_i([w]) = \sum_{i=1}^{k} p_i b_i, \tag{3.6}$$

where $b_i = \begin{bmatrix} b_{i1} \\ b_{i2} \end{bmatrix} = \begin{bmatrix} \underline{\tilde{y}_i} - \underline{y_i([w])} \\ \overline{y_i([w])} - \overline{\tilde{y}_i} \end{bmatrix}; p_i = [p_{i1} \;\; p_{i2}], p_{iq} = \begin{cases} 1, & b_{iq} \geq 0, \\ -\frac{b_{iq}}{N}, & b_{iq} < 0; \end{cases} \quad q = \overline{1,2}.$
Here k is the size of the training data set, $[w]$ is the interval weight vector, $Q_i([w])$ is the training error on example i , p_i is the row vector of weight coefficients for the deviation of the network's output interval bounds from the bounds of the example i of the training data set, b_i is the vector of deviation between the network's output interval bounds and the bounds of example i, $\underline{\tilde{y}_i}$ is the lower bound of the output interval on training example i, $\underline{y_i([w])}$ is the lower bound of the network's output interval on sample i, $\overline{y_i([w])}$ is the upper bound of the network's output interval on example i, $\overline{\tilde{y}_i}$ is the upper bound of output interval of example i, p_{iq} is element q of row vector p_i, b_{iq} is is element q of row vector b_i, q is the number of the vector element, N is the tolerance level showing the admissible sequence of deviations of an interval bound (e.g. 0.1, 0.01, 0.001).

To train the INN model, it was offered to use the following interval adaptive algorithm of global function optimization based on weight vector bisection [6] instead of interval BP algorithm.

Algorithm 1
Input: the minimum width $\delta > 0$ of the bar, required training error $\varepsilon > 0$.
Output: $[w]$ is the weight vector, which is a minimum of function (3.6); J^* is the minimum value of function (3.6).

```
1. Initialization of the initial weights of $[w]$.
2. Calculation of INN output $y^*$ and quality function $J^*$.
3. Initialization of list $L = [w], J^*$.
4. Initialization of bisection coordinate $l = 0$ and iteration counter $c = 0$ (which counts
   iterations needed for exiting the cycle).
5. Cycle: as long as $\min wid_{i=1}^{n}[w_i] > \delta$ and $J^* > \varepsilon$.
    (a) Calculation of bisection coordinate $l = l + 1$. If $l$ exceeds the weight vector, then
        $l = l$.
    (b) Bisection of $[w]$ in coordinate $l$ into $[w']$ and $[w'']$.
    (c) Calculation of INN outputs $y^*$ and quality functions $J'$ and $J''$.
    (d) If $J' > J^*$ and $J'' > J^*$, then go to Step 5e. Otherwise, go to Step 5f.
    (e) Assignment $c = c + 1$. If $c$ is not equal to the weight vector value, then go to Step
        5a. Otherwise, exit the cycle.
    (f) Deletion of element $([w], J^*)$ from $L$.
    (g) Addition of $([w'], J')$ and $([w''], J'')$ to list $L$.
    (h) Arrangement of list $L$ in ascending order by the value of the second field.
    (i) The first record is denoted by $([w], J^*)$.
```

Thus, usage of the proposed error function and learning algorithm makes it possible to guarantee that output interval contains exact solution.

It is known that a result of calculating a function in interval analysis depends on the way the variable is represented in the formal expression. To obtain a more accurate value the number of variables contained in the formal expression should be minimized. In the above formulation of the problem it means that the number of layers should be minimized too. This conclusion is confirmed by experimental results of previous research. Thus, minimizing ofNN layers improves accuracy of training. But a too small number of layers reduces ability of INN to approximation. This conflict was overcome by introduction of the new subclass of INN.

3.2. Dual-parametric neural network

Dual-parametric neuron network (DPNN) is a subclass of INN that contains two types of parameters: real and interval ones. A DPNN model with a single interval output contains:

- n interval output neurons;
- m hidden layers;
- real neuron weights in all layers (from layer 1 to layer $m-1$);
- interval neuron weights in layer m.

Besides, input and output values are also interval. In the case when input or output values should be real it is possible to represent them as intervals whose left and right bounds are equal.

Real weights make it possible to get high quality of learning of the neural network model. Interval weights guarantees that the intervals of the training examples will be fully included in the output intervals of the network. Thus, it is possible to guarantee that output intervals contains single-point solutions.

For the purpose of training the DPNN, it was proposed in previous research to use an algorithm combining the interval BP method and the global optimization algorithm for interval values.

1. At the first stage, the real weights are trained using the interval BP algorithm.
2. At the second stage, the resulting weights are only used to initialize the model, followed by the training of interval weights by means of the interval global optimization algorithm.

Besides, a structural identification algorithm was developed for DPNN models with one hidden layer. It allows for selecting the optimal number of neurons, which are consecutively added to the hidden layer until the training error is reduced.

3.3. Learning algorithm based on the intervals extension procedure

As we note earlier to train the second stage model of DPNN we use the interval global optimization algorithm based on bisection of weight vector. This approach has several drawbacks. One of them is that the initial intervals of the weight vector need to have enough width. This width should be enough to include training data outputs into the model outputs for all examples in a training set when the signal passes through the network at the first time. Thus, there is a difficulty in generating initial weights to train the model at the second stage of DPNN learning algorithm.

The second drawback is related to the first one. According to the traditional interval global optimization algorithm an efficient choise for the bisection is cut along a coordinate with the maximum width [7]. But if the widths of all intervals is the same there is no reason to consider any of them as the most appropriate. It is possible to generate random initial intervals with different widths. But the question remains how to achieve inclusion of training data outputs into model outputs at the first iteration if this is not satisfied.

To eliminate these drawbacks we introduce a learning algorithm based on the intervals extension procedure instead of the bisection.

Algorithm 2

Step 1. We initialize a model with one interval output, n input interval neurons, m hidden layers and weights assigned in real values to the neurons of all layers.

Step 2. We train the model using the BP algorithm.

Step 3. We initialize a model with one interval output, n input interval neurons, m hidden layers and weights assigned in real values to the neurons of all layers (from layer 1 to layer $m-1$). The neuron weights in hidden layer m are assigned in interval values.

Step 4. We assign real values to the weights in the model trained at Step 2, and the ones in the model defined at Step 3 for all hidden layers.

Step 5. We train the model formed at Step 4 in accordance with the algorithm of training INNs with interval weights. The weights of neurons in all hidden layers (from layer 1 to layer $m-1$) are regarded as constant values. Only the weights of hidden layer m are subject to change.

Step 5.1. We select the increment value M of an interval.

Step 5.2. A loop starts along the coordinates of the weight vector. With each iteration M substracts from the lower bound of the weight vector interval. At each iteration, the network output and the value of quality function are calculated. If the received training error value is less than the error obtained in the previous step, the weights vector and training error are fixed as minimum ones.

Step 5.3. Step 5.2 is repeated while the training error on each loop iteration is reducing the minimum one.

Step 5.4. Steps 5.2 and 5.3 are executed in the same way for the upper bound of weight vector intervals. In this case, M is added to the upper bound of the interval at each loop iteration.

4. NUMERICAL EXPERIMENTS

The goal of our experiments is to show a difference between forecasting results obtained using INN with the interval BP algorithm and the proposed approach with DPNN. In our experiments, for the learning of a DPNN we generate interval-valued time series:

$$x^{(i)} = 0.0, 0.1, 0.2, ...;$$

$$\underline{y^{(i)}} = 0.2\sin(2\pi x^{(i)}) + 0.1\big(x^{(i)}\big)^2 + 0.2 + 0,005rnd[-1,1];$$

$$\overline{y^{(i)}} = 0.2\sin(2\pi x^{(i)}) + 0.2\big(x^{(i)}\big)^2 + 0.3 + 0,005rnd[-1,1].$$

Here $rnd[-1,1]$ is a random real number in the closed interval $[-1,1]$. Training set includes 35 pairs $\big(\underline{y^{(i)}}, \overline{y^{(i)}}\big)$. The next 5 pairs are used as a basis for projections.

We use the proposed model of DPNN with a single output neuron, 9 input neurons and one hidden layer with 3 neurons in it. To compare proposed approach and interval BP algorithm we make projections using both of them. Results are shown in Fig. 4.1, 4.2 and in Table 4.1.

Table 4.1. Experimental results

No. of the experiment	1	2
Training time, sec	28	302
Training error	0.07	4.13
Average relative deviation of projected value bounds from actual ones	0.15	0.12

The experiments demonstrate that projected values obtained by using the proposed approach (experiment 2) include of actual values at least 3 of 5 values. Thus, in 60% of cases we can guarantee that projected value includes the exact solution. Projected results interval covers just a part of actual values interval because the forecasting error accumulates with

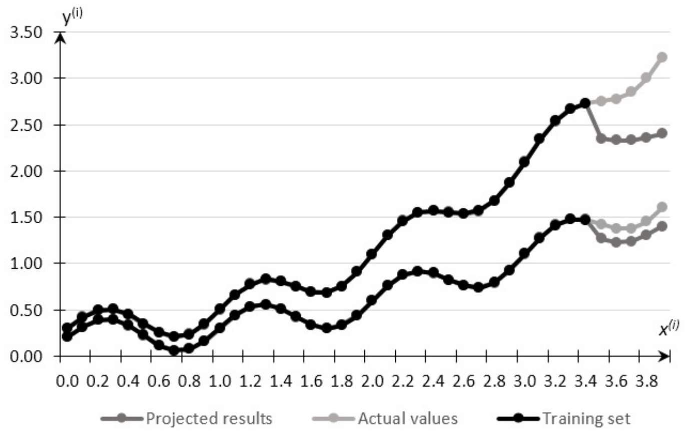

Fig. 4.1. Forecasting using only interval BP algorithm (experiment 1).

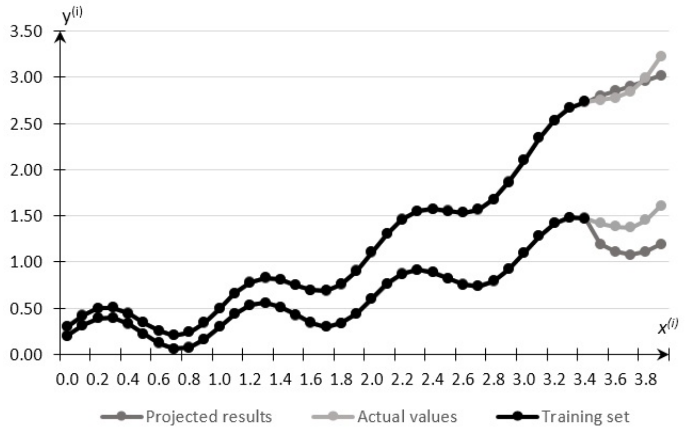

Fig. 4.2. Forecasting using the proposed approach with the learning algorithm based on the intervals extension procedure (experiment 2).

each subsequent projected interval. For the same reason actual values lies out of the projected interval bounds in the last values and not in the first ones. But, nevertheless, projected values obtained in experiment 2 seem to be more informative for practical use than projected results in experiment 1.

Training time in Table 4.1 is so different, because BP algorithm used in experiment 1 is a part of learning algorithm of experiment 2. Training errors is so different, because algorithms in experiment 1 and 2 uses different error functions – (2.1) and (3.6) respectively. Average relative deviation of projected value bounds from actual ones is calculated as follows:

$$\delta = \frac{1}{2n} \sum_{i=1}^{n} \left(\frac{|\underline{y_i} - \underline{\tilde{y}_i}|}{|\underline{y_i}|} + \frac{|\overline{y_i} - \overline{\tilde{y}_i}|}{|\overline{y_i}|} \right),$$

where n is the number of projected values; $\underline{y_i}, \overline{y_i}$ are the lower and the upper projected value bounds; $\underline{\tilde{y}_i}, \overline{\tilde{y}_i}$ are the lower and the upper actual value bounds.

5. CONCLUSION

This study was centered on the approach to modelling and forecasting
addedof interval-valued data using DPNN models. In the article we provided a theoretical overview of the technology we use, and described the proposed approach that makes it possible to get guaranteed inclusion of exact (single value) solution into interval forecasting results. Our experiments confirmed the efficiency of DPNN models in predicting interval-valued data as at least in 60% of cases we can guarantee that projected values include the exact solutions.

As a prospective investigation our research will focus on training of INN and DPNN models. We take the following training data set of interval values as the initial data: $\{[\tilde{x}_i], [\tilde{y}_i]\}_{i=1}^{k}$. The INN (DPNN) will be used to compute interval functions for interval arguments in the following manner: $[y] = f([w], [x])$, where $[w]$ is the vector of interval parameters (weights) of the network. Besides, we are going to consider the problem of computing such INN (DPNN) weight values that allow the model outputs to include all interval outputs comprised by the training data set. One more problem to be considered is the optimization of the training quality function $Q([w]) = \max d([y_i], [\tilde{y}_i]) \to min$ with the distance between intervals d calculated as follows:

$$d([y_i], [\tilde{y}_i]) = \begin{cases} +\infty, & \text{if } \overline{y_i} < \overline{\tilde{y}_i} \text{ or } \underline{y_i} < \underline{\tilde{y}_i}; \\ \max\left\{|\overline{y_i} - \overline{\tilde{y}_i}|, |\underline{y_i} - \underline{\tilde{y}_i}|\right\}, & \text{otherwise.} \end{cases}$$

REFERENCES

1. Patil R.B. (1995) Interval Neural Networks. *APIC'95, El Paso, Extended Abstracts, A Supplement to the International Journal of Reliable Computing*, El Paso, TX, 164.
2. Belohlavek R. (1997) Backpropagation for Interval Patterns. *Neural Network World*, **7**(3), 335–346.
3. Garczarczyk Z.A. (2000) Interval neural networks. *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353)*, **3**, 567–570.
4. Kim H.J. & Ryu T.-W. (2005) Time Series Prediction Using an Interval Arithmetic FIR Network. *Neural Information Processing – Letters and Reviews*, **8** (3), 39–47.
5. Yang D., Li Z. & Wu W. (2016) Extreme learning machine for interval neural networks. *Neural Computing and Applications*, **27**(1), 3–8.
6. Sharyi S.P. (2008) Randomized algorithms in interval global optimization *Numerical Analysis and Applications*, **1**(4), 376389. https://doi.org/10.1134/S1995423908040083
7. Jaulin L., Kieffer M., Didrit O. & Walter E. (2001) *Applied Interval Analysis.* London, UK: Springer.