Weights Ordering During Training of Contextual Neural Networks with Generalized Error Backpropagation: Importance and Selection of Sorting Algorithms

Maciej Huk(✉)

Department of Information Systems,
Wroclaw University of Science and Technology, Wroclaw, Poland
maciej.huk@pwr.edu.pl

Abstract. Contextual neural networks which are using neurons with conditional aggregation functions were found to be efficient and useful generalizations of classical multilayer perceptron. They allow to generate neural classification models with good generalization and low activity of connections between neurons in hidden layers. Their properties suggest also that usage of contextual neurons with conditional signals aggregation can cause similar effects as dropout technique in convolutional deep neural networks. The key factor to build such solutions is achieving self-consistency between continuous values of weights of neurons' connections and their mutually related non-continuous aggregation priorities. This allows to optimize neuron inputs aggregation priorities by simultaneous gradient-based optimization of connections' weights with generalized BP algorithm. But such method additionally needs to perform sorting of neuron inputs by its weights after each given number of training epochs. Thus within this text we compare efficiency of training of contextual neural networks with selected sorting algorithms. On this basis we discuss the theoretical properties of analyzed training algorithm which are related not only to characteristics of used weights sorting methods but also to application of self-consistency to selection of neural scan-paths in contextual neural networks.

Keywords: Classification · Self-consistency · Sorting · Aggregation functions

1 Introduction

Contextual neural networks are generalizations of known neural networks architectures which are using neurons with multi-step conditional aggregation functions [1]. Those models were used with success to solve both benchmark as well as real-life classification problems [2, 3]. In [4, 5] they were shown to be very good tools for fingerprints detection for crime-related analyses. They were also successfully used for spectrum prediction in cognitive radio and for research related to measuring awareness of computational systems [6].

N. T. Nguyen et al. (Eds.): ACIIDS 2018, LNAI 10752, pp. 200–211, 2018.
https://doi.org/10.1007/978-3-319-75420-8_19

Contextual neural networks can form classifiers with better generalization properties than their non-contextual versions such as MLP [3]. But what is more important they have also ability to considerably limit activity of connections between neurons without decreasing accuracy of their output values – both during and after training [2]. This can be used to decrease time and energy costs of running trained neural networks, especially in highly constrained applications. Limiting activity of given connections is done adaptively to processed data, thus for different input vectors given neuron can use different subsets of inputs and does not read signals from inputs not needed to calculate output value for given input vector. Finally, this changes the character of the neural network from black-box to grey-box model [3]. This is because by analyzing activity of inputs of neurons in the first hidden layer of contextual neural network one can check which data attributes are needed to calculate output values of the network for each input vector. Finally, with this technique data attributes can be ordered by their importance found by the contextual neural network for solving given problem.

In detail, neurons used to build contextual neural networks aggregate signals not in one but in multiple steps. Each aggregation step is used to read-in given subset of inputs and to decide if already processed information is enough to calculate the output value of the neuron with acceptable accuracy. The composition and order of groups of inputs, adequate for problem solved by the neural network, are selected independently for each neuron during training of the model. Given ordered list of groups of inputs is called a "scan-path", because multi-step conditional aggregation functions are realizations of Starks' scan-path theory [7]. Such functions in following steps aggregate signals from different subsets of inputs until neuron activation cumulated from previous groups of inputs is lower than given constant threshold. Examples of those functions are Sigma-if, CFA, OCFA and RFA functions [1].

It is also worth to notice that contextual neural networks with multi-step conditional aggregation functions in comparison to their non-contextual versions need only a small number of additional parameters, and values of those extra parameters can be easily selected with use of simple rules prior to the training [2]. This is the effect of special construction of aggregation functions of their neurons which allows to use generalized backpropagation algorithm (GBP) and self-consistency paradigm to setup their complex behavior and represent it within values of connection weights [8]. This is considerable improvement in comparison to multi-parameter aggregation functions of other contextual neurons: Clusteron, Sigma-Pi [9] or Spratling-Hayes neuron [10].

In this paper we find that elements of the generalized error backpropagation algorithm connected with sorting of data, can be further improved to increase the efficiency of the training of contextual neural networks. Thus the rest of the paper is organized as follows. In Sect. 2, brief description of the generalized backpropagation algorithm is given. Then Sect. 3 presents detailed discussion of possible improvements of the phase of GBP algorithm used to sort neuron's inputs. This is next used in Sect. 4 to experimentally test which sorting algorithms are best suited to be used within GBP while training contextual neural networks to solve selected UCI machine learning benchmark classification problems. Finally in Sect. 5, we discuss how obtained results and analyzed sorting methods can be used to characterize previously not studied theoretical properties of the GBP algorithm and contextual neural networks, especially the influence of self-consistency on neural scan-paths during GBP.

2 Generalized Backpropagation Algorithm

The generalized error backpropagation algorithm (GBP) is a modification of classical error backpropagation method extended to be able to train contextual neural networks which are using neurons with multi-step conditional aggregation functions [1, 2]. This is done by incorporating self-consistency paradigm known from physics [8]. It allows to use gradient based method to optimize simultaneously continuous (connection weights) and non-continuous, non-differentiable parameters of neuron's aggregation functions. The key to such abilities of the GBP method is maintaining mutual dependency of those both groups of parameters. It is realized by defining non-continuous parameters as function Ω of continuous weights. For neurons with multi-step aggregation function the Ω relation consists of two operations: sorting N neuron connection weights and then dividing ordered inputs into list of K equally-sized groups. N/K inputs with highest weights go to the first group, next N/K inputs with highest weights go to the second group, etc. Then the basic scan-path can be defined as the list of groups from first to last. It can be used within aggregation function, where groups of inputs are read-in one after another until given condition is met. Without details of aggregation the GBP algorithm can be represented as on the Fig. 1.

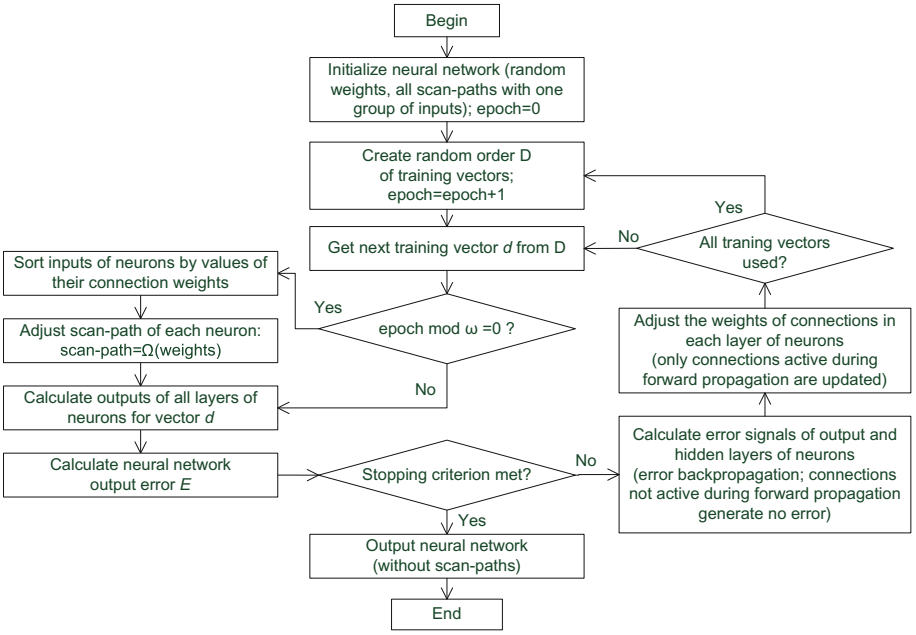

Fig. 1. General flowchart of the generalized error backpropagation algorithm (GBP) for given scan-path creation function Ω. Scan-paths update interval ω controls the strength of self-consistency between connections weights and parameters of neuron aggregation functions. After the training, temporary neuron scan-paths are discarded – they can be re-created later from weights with function Ω.

It is important to notice how interval ω controls the strength of self-consistency between connections weights and parameters of neuron aggregation functions. At the beginning and during the first ω epochs of GBP algorithm all neurons have all inputs assigned to the first group of their scan-paths. This is because at the beginning of learning we do not want to make any assumptions about the importance of the neurons inputs. Then, after each ω epochs scan-paths are updated with Ω function in accordance to actual connection weights. Thus the selection of interval ω can considerably influence the training process. If $\omega = \infty$ (or number of groups $K = 1$) the GBP algorithm behaves exactly as the classical error backpropagation method, and the contextual neural network with neurons using conditional multi-step aggregation functions behaves like MLP. On the other side, when $K > 1$ and the value of ω is close to one the error space of the neural network is frequently reorganized due to the updates of scan-paths of neurons after each ω epochs. This can make the training process to be ineffective. But it was shown experimentally that when $K > 1$ and $5 < \omega << \infty$, the output error of the neural network can temporarily increase after the update of the scan-path, but during following epochs error can drop down again, together with the activity of neural network connections.

Due to the above, the models built with GBP algorithm can have better generalization properties and lower average connections activity than their non-contextual versions trained with classical error backpropagation. One of the reasons for such effect is the following: cyclic reorganizations of the scan-paths of the neurons during the GBP together with evolving of decision space of each neuron during conditional multi-step signals aggregation, form mechanism analogous to the dropout technique extensively used in deep learning solutions. But there are also fundamental differences. Dropout is not dependent on the processed data, and the decision spaces of neurons of contextual neural network change according to actually processed data vector and during GBP take into account what the model has learned till given epoch. Dropout decreases the internal activity of the network connections only during the training process, and contextual neural networks limit it both during and after the training. This makes conditional multi-step aggregation functions valuable both for basic feedforward neural networks like MLP as well as for convolutional neural networks.

Presented description of the training process of contextual neural networks indicates that frequent updates of scan-paths of the neuron's aggregation functions are connected with high number of sorting of neuron inputs by their weights. For high number of neurons this can form considerable computational cost, thus it is worth to check which sorting algorithm is most suitable for use within the GPB method.

3 Sorting Weights of Neurons' Inputs Within GBP Algorithm

As it was indicated in the previous section, probably the most important element of the GBP algorithm is the update of neuron' scan-path realized with use of inputs sorting phase. This is because it allows to merge gradient-based search done by the error backpropagation algorithm with the self-consistency paradigm. But one can also notice that the selection of the sorting algorithm for GBP method is not straight-forward. This is because at least two factors can influence the efficiency of each sorting of inputs.

The first factor is the number of inputs of given hidden neuron – the size of the data to be sorted can considerably influence efficiency of given sorting method. And the second is the characteristic of the data to be sorted – which is hard to model in the general case of scan-path changes during the GBP process for different problems. This is because, depending on the data to be processed, values of parameters of GBP and on the phase of the training process, connection's weights can change a lot or almost not change between epochs of subsequent scan-paths updates (after each ω epochs). And different sorting algorithms can behave differently for sorted or partially sorted lists of data. In such cases their computational efficiency can be even much lower or much higher from their average efficiency.

To check the influence of the sorting method on the efficiency of the GBP algorithm, it will be helpful to analyze in detail how the sorting method is used by the training algorithm. At the beginning of the GBP the initial, random values are assigned to the connections weights of neuron inputs of given indexes. At the same time all inputs are assigned to the first aggregation group – this is to ensure that all inputs are used during first cycles of error backpropagation algorithm. Let's assume that after such initialization and first omega epochs of the training, connection weights w of neuron inputs of given indexes I as well as their assignment to scan-path groups G are as presented below:

```
W= [0.3,0.9,0.1,0.4,0.2,0.6,0.8] //connection weights
I= [  1,  2,  3,  4,  5,  6,  7] //indexes of inputs
G= [  1,  1,  1,  1,  1,  1,  1] //assignments to groups
```

Then given sorting method creates ordered list of connection weights, reordering accordingly also indexes of neuron inputs - without changing of the initial relation of given weight of its neuron input. After sorting of weights and reordering of connection indexes, neuron inputs are divided into given number of groups K. If possible, all groups have the same number of elements. But when the number of neuron inputs N can't be divided into K groups of equal size, the eventual rest of connections are added to the group of highest number. Then, during following epoch, neuron first aggregates signals from inputs belonging to group 1. Next, accordingly to details of its aggregation function, it conditionally aggregates signals from the remaining groups.

```
W= [0.9,0.8,0.6,0.4,0.3,0.2,0.1] //connection weights
I= [  2,  7,  6,  4,  1,  5,  3] //indexes of inputs
G= [  1,  1,  2,  2,  3,  3,  3] //assignment to 3 groups
```

Such weights sorting procedure and reassignment of neuron's input connections to aggregation groups is performed after every omega epochs of training, to keep mutual relation between values of weights and aggregation priorities after changes of weights values done in the effect of presentation of training vectors.

It is important to note, that for every sorting algorithm, given initial neural network is transformed exactly into the same trained neural model. This is because each sorting algorithm for the same input data outputs identical sorted list of values. But depending

on the number of inputs of network' hidden neurons, different sorting methods will present different efficiency. However, the reader can notice that average sorting efficiency not necessarily must apply to data sorted during the training with generalized backpropagation. This is because GBP training procedure changes weights and their assignments to aggregation groups in non-random way. If the changes of values of weights between epochs are very small, then subsequent runs of sorting algorithm for given neuron will process list of data which is almost perfectly ordered. In such case superior efficiency would be achieved by algorithms that are optimal for sorted data – e.g. insertion-sort algorithm. In the opposite case, if the changes of values of weights between epochs are big enough to considerably change initial assignments to aggregation groups, then in most cases sorting algorithm for given neuron will process data which is not ordered. In such case better efficiency would be achieved by method such as quick-sort algorithm.

Finally, in this paper we want to check experimentally the efficiency of different sorting algorithms under conditions occurring during training of contextual neural networks with GBP algorithm. It will help to formulate guidelines how one should select the sorting algorithm for given contextual neural network. But it is more important, that it will help us to answer important question about characteristic of GBP method: how much the changes of weights between subsequent sorts influence the aggregation priorities of hidden neuron inputs? It can be, that after each sorting of weights the aggregation priorities of neuron inputs and the related error space of neural network change drastically. This would make the training more like random process. In the opposite case, the sorting would not change the initial aggregation priorities at all, what could mean that usage of the self-consistency paradigm has no influence on the training results. Is any of the above dominating the training? Does it depend on the frequency of scan-paths updates? Answering to the above questions will greatly increase our understanding of properties of the generalized backpropagation algorithm.

4 Results of Experiments

To answer the questions stated in the previous section, a set of experiments was performed for six different serial versions of sorting algorithms such as: Quick-sort (recursive), Merge-sort (recursive), Insertion-sort, Bubble-sort, Optimized Bubble-sort and Two-Way Optimized Bubble-sort [11–13]. Quick-sort, Insertion-sort and Merge-sort were used in their basic versions, without any improvements. In particular this means that Quick-sort was using middle element as a pivot and no specialized mechanism was used for sorting short sub-partitions [12]. By analogy the Insertion-sort was not performing initial placement of the smallest element at the beginning nor shifting larger elements prior to the insertion instead of carrying out many exchanges [11]. Decision to not use abovementioned optimizations was done because they were designed for the average case of unsorted data and we expect that this can be not valid in the case of partially sorted, GBP related data. Thus using simple algorithms should ease the interpretation of the results and help formulating possible sorting optimizations dedicated for use with GBP algorithm.

During the experiments contextual neural networks were trained with generalized backpropagation algorithm to solve example UCI ML benchmark problems such as Iris, Sonar, Soybean, Lung Cancer and Wine. All neurons within the neural network were using bipolar sigmoid or leaky rectifier activation function and Sigma-if aggregation function [2, 3]. Aggregation functions of all hidden neurons were dividing their inputs into K groups and were using aggregation threshold $\varphi^* = 0.6$. Continuous data attributes were normalized to range <0, 1>, and nominal attributes were represented with single binary input for every value (one-hot encoding). Resulting architectures of neural networks used during the experiments are presented in Table 1.

Table 1. Architectures of contextual neural networks used during the experiments for selected benchmark problems from UCI ML repository.

Training data set	Number of inputs of neural network	Number of hidden neurons (N)	Number of classes	Number of connections between neurons	Number of groups of inputs of neurons (K)
Iris	4	10	3	70	3
Wine	13	10	3	160	3
Hypo	35	10	5	400	14
Sonar	60	10	2	620	10
Soybean	134	20	19	3060	22
LungC	244	3	3	741	3

The numbers of neurons in hidden layer and the number of groups K were based on results of previous experiments and selected to achieve suboptimal classification accuracies of resulting models with considerable reduction of connections activity between neurons.

The values of parameters of the GBP training algorithm were as follows: constant training step $\alpha = 0.1$ (for networks with sigmoidal activation function) or $\alpha = 0.01$ (for networks with leaky rectifier activation function), interval of aggregation groups update $\omega = 25$, stopping criterion: perfect classification of training data or no classification accuracy improvement for more than 300 or 1300 subsequent training epochs. No momentum was used.

For each set of values of above parameters, training was performed within Borland Delphi 7 environment for set of ten different values of seed of built in pseudo-random generator used to initialize neural networks and to select sequences of training vectors during the training. The seed values were: 0, 1013420422, 550932796, 729462840, 923512745, 327401735, 423664106, 834385610, 245071639, 619168307. Additionally each training was repeated ten times to ensure that resulting neural models for given seed value and given set of values of other parameters are identical, and to increase precision of measurements of computational cost. Cross-validation was not used because analysis of the abilities of generalized backpropagation algorithm to build contextual neural networks of good classification properties was not the goal of

performed experiments. Thus each training for given problem was done with use of the whole given training data set.

During each experiment following properties were measured:

- cumulated cost of execution of sorting method during the training,
- number of training epochs before the stopping criterion was met,
- average classification error of the resulting model,
- average, minimal and maximal activity of internal connections of the neural network.

Activity of the connections was measured for the neural network with the highest average classification accuracy observed during the GBP process. Computational cost of each sorting was measured with the use of main processor' integrated Time-Stamp-Counter (TSC) 64 bit register. Cumulated cost of execution of sorting method during given training was calculated by summing up numbers of ticks of the processor clock counted by TSC counter during each sorting (for each hidden neuron and each epoch). Then, the average cost of single sorting was calculated by dividing the cumulated cost of sorting by the number of calls of the sorting method. To minimize influence of the operating system on the measurement precision, TSC register was used with highest priority execution mode of the simulation program.

Basic results of experiments designed as described above are presented in Table 2. They include average classification errors, average number of epochs of training and average connections activity of contextual neural networks. Stopping criterion of the GBP was: network error equal 0 or no decrease of the error during last 1300 epochs.

Table 2. Average values of classification error, epochs of training as well as minimal, maximal and average activities of hidden neurons connections of contextual neural networks solving example UCI benchmark classification problems. Neurons with Sigma-if aggregation function as well as bipolar sigmoid (BS) and Leaky Rectifier (LR) activation functions were used.

Training data set	Activation function	Classification error [%]	Training epochs [1]	Activity (avg.) [%]	Activity (max) [%]	Activity (min) [%]
Iris	BS	3.9	149.1	86.0	87.5	85.7
Wine	BS	4.3	194.6	91.7	91.7	91.7
Hypo	BS	1.3	1763.3	25.7	73.1	17.5
Sonar	BS	0.8	2037.8	15.4	35.6	9.7
Soybean	BS	2.0	2286.8	22.9	36.0	15.9
LungC	BS	0.3	326.5	26.4	40.2	24.2
Iris	LR	5.0	748.7	57.1	57.1	57.1
Wine	LR	3.4	497.8	66.1	68.2	65.6
Hypo	LR	2.4	2285.7	19.8	58.9	17.5
Sonar	LR	6.9	725.7	19.7	39.2	11.5
Soybean	LR	2.1	2465.2	18.5	30.3	15.2
LungC	LR	2.2	1703.2	26.9	41.7	24.2

Achieved considerable decrease of connections activity proves contextual behavior of neural networks trained with GBP. It can be also seen that contextual neural networks of bigger structures (for Hypo, Sonar, Soybean and LungC problems) can achieve lower activity of connections than smaller networks (here for Iris and Wine problems). This was also expected because was observed in earlier experiments [3]. Thus performed simulations can be used to search for potential dependencies between properties of contextual neural networks trained with the GBP algorithm and computational efficiency of selected sorting methods used for scan-paths updates in GBP.

Within Table 3 we present measured relative average computational cost of six selected sorting methods during their use for neurons' scan-path update by GBP algorithm. Measures were taken by repeating GBP training for each of the sorting methods for the same set of benchmark problems, neural networks architectures and related parameters (including seed of pseudo-random generator, initial values of connections weights, etc.). For given set of initial values of parameters the result of the training was model identical for each of the sorting methods. This is because all used sorting methods give the same output for the same input data, and switching between different sorting methods can't change the result of the GBP. But each of the sorting methods can need different number of operations to sort given input data. This causes differences between computational costs of considered sorting methods during the GBP. To ease the analysis of results all obtained values of computational costs are presented in relation to the cost measured for the Insertion-sort method. For the same reason, within Table 3 the length of scan-path of hidden neurons is additionally presented. Scan-path length is defined here as the number of values which have to be sorted during single update of the scan-path of given neuron by the GBP algorithm.

Within presented results it can be noticed that the factor with strongest influence on the efficiency of scan-path updates is the length of scan-paths. This causes that for neural networks with hidden neurons having low number of inputs (data for Iris and Wine benchmarks) the most efficient sorting method is Insertion-sort. With the increase of the length of the scan-path above certain length the most efficient sorting method becomes Quick-sort. Based on the results from Table 3 this bound can be estimated by interpolation but it can be also presented graphically as on the Fig. 2. This shows that for considered contextual neural networks Quick-sort within GBP should be preferred for scan-paths longer than 35 elements, irrespective of the type of activation function, and Insertion-sort should be used for shorter scan-paths.

The above observation of effectiveness of sorting algorithms is considerably different than when Quick-sort is compared with Insertion-sort for random distribution of values to be sorted. In [12] as well as in well-known programming libraries the threshold, below which Insertion-sort is used instead of Quick-sort, is much lower than 35. For example, Bentley and McIlroy set this threshold to 7 [12]. GNU C Library (glibc), GNU ISO C++ Library and C++ Standard Library (STL) as this threshold use value 5. And in FORTRAN 90/95 one can find it to be equal 10. Such low values of this threshold are used because mentioned solutions are implemented to maximize average sorting efficiency for all possible sets of input data. Thus within conducted experiments Insertion-sort presents computational cost lower than average and closer to its best case of computational complexity $\Theta(n)$, where n is the length of scan-path. In the same case cost of Quick-sort is higher than average and closer to its worst case complexity of $\Theta(n^2)$.

Table 3. Average relative cost of connections sorting per neuron per epoch during training of contextual neural networks with GBP for different sorting algorithms. Sigma-if aggregation and example UCI problems were used for two activation functions: bipolar sigmoid (BS) and Leaky Rectifier (LR). Each result is average of 100 measurements. Scan-path length is the number of inputs of hidden neuron. Stopping: 300 epochs without improvement of classification accuracy.

Training data set	Activation function	Scanpath length	Merge sort [%]	Quick sort [%]	Insert sort [%]	Bub. sort two-way [%]	Bub. sort optimized [%]	Bubble sort [%]
Iris	BS	4	153.4	131.2	100	112.8	111.0	97.2
Wine	BS	13	180.8	137.1	100	172.4	188.4	242.1
Hypo	BS	35	154.0	98.1	100	265.9	271.3	280.6
Sonar	BS	60	103.1	83.1	100	250.7	272.9	297.7
Soybean	BS	134	51.2	44.0	100	242.8	252.6	293.0
LungC	BS	244	50.7	35.9	100	270.2	285.6	284.6
Iris	LR	4	186.3	135.8	100	119.8	117.4	101.6
Wine	LR	13	184.4	125.1	100	153.8	166.7	169.2
Hypo	LR	35	149.5	101.6	100	220.9	228.3	253.3
Sonar	LR	60	98.3	78.7	100	243.8	268.1	282.4
Soybean	LR	134	53.9	44.0	100	244.2	254.2	260.3
LungC	LR	244	42.0	47.15	100	225.2	237.9	236.0

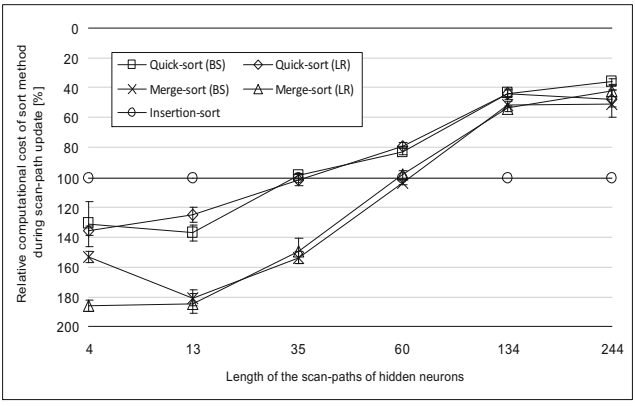

Fig. 2. Average computational cost of sorting during updates of scan-paths of contextual neural network by the GBP algorithm as a function of scan-paths length for selected sort algorithms and two activation functions of neurons: bipolar sigmoid (BS) and leaky rectifier (LR). Results are presented in relation to the measurements for Insertion-sort method. Subsequent, growing lengths of scan-paths correspond to the following UCI ML benchmark data sets: Iris, Wine, Hypothyroid, Sonar, Soybean and Lung Cancer. Scan-paths update interval was equal 25.

The interpretation of obtained results is straightforward - GBP algorithm can keep weights of inputs of given hidden neuron partially ordered between subsequent updates of its scan-path. Thus the scan-paths do not change in a random manner. Moreover, the

average percentage of the scan-path which is changed during scan-paths update can be different for different activation functions of the neurons. We observe that it is higher for bipolar sigmoid and training step $\alpha = 0.1$. than for leaky rectifier function and $\alpha = 0.01$. Thus on average the scan-paths of neurons are not constant during the training of considered contextual neural networks with the GBP algorithm. Finally, if the scan-paths are changing during the training and those changes are not random, then self-consistency effect has considerable influence on the training. This is important, both practical and theoretical, result.

Using the outcomes of performed experiments it can be also checked how the frequency of scan-paths updates influences the efficiency of sorting methods. For Wine problem it is visualized on Fig. 3. For scan-paths interval ω from 5 to 60 epochs mutual relations of efficiency between sorting methods do not change. Thus, excluding cases when ω is higher than number of epochs of given training, earlier conclusions on functioning of self-consistency and GBP method stay valid for different ω values.

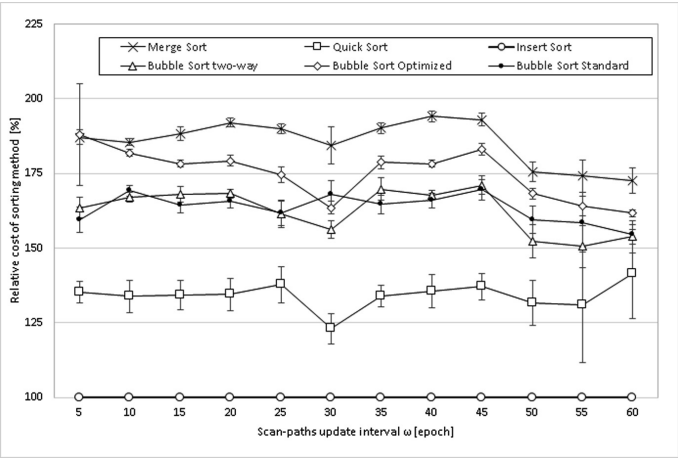

Fig. 3. Average computational cost of sorting during updates of scan-paths of contextual neural network by the GBP algorithm for Wine data set as a function of scan-paths update interval ω. Results are presented in relation to the measurements for Insertion-sort method. Activation function: bipolar sigmoid. Stopping: 300 epochs without error decrease. Training step $\alpha = 0.1$.

5 Conclusions

Measurements and analyses presented in this paper are unique in their nature because basic BP algorithm is not related with sorting methods. We have shown that self-consistency within GBP has considerable influence on the evolution of scan-paths of contextual neurons during training. In the effect, for scan-paths shorter than 35 elements Insertion-sort makes GBP method less computationally expensive than other considered sorting algorithms. Above this threshold Quick-sort was better choice and such result was noted for wide range of scan-paths update interval ω.

Moreover, measured values of this Insertion-sort threshold are considerably higher than analogous values used by hybrid sorting methods implemented within standard programming libraries. This is because those implementations of sorting algorithms are optimized for achieving maximal average efficiency – and the scan-path related data of neurons is preserved partially sorted during most of the epochs of their training with GBP. Thus in cases when computational efficiency of GBP would be important, one should consider using dedicated sorting method for scan-paths updates.

Presented experiments open new, previously not explored valley in the field of research on artificial neural networks. Thus possible further work includes analyzing relations between computational efficiency of GBP and such factors as different sorting methods (e.g. Shell-sort, partial interval sorting), various multi-step aggregation functions as well as parameters of neural networks and training methods. We expect that such research will allow to train contextual neural networks more efficiently and that it will extend our actual understanding of the nature of those models.

References

1. Huk, M.: Notes on the generalized backpropagation algorithm for contextual neural net-works with conditional aggregation functions. J. Intell. Fuzzy Syst. **32**, 1365–1376 (2017)
2. Huk, M.: Backpropagation generalized delta rule for the selective attention Sigma-if artificial neural network. Int. J. Appl. Math. Comput. Sci. **22**, 449–459 (2012)
3. Huk, M.: Learning distributed selective attention strategies with the Sigma-if neural network. In: Akbar, M., Hussain, D. (eds.) Advances in Computer Science and IT, pp. 209–232. InTech, Vukovar (2009)
4. Szczepanik, M., Jóźwiak, I.: Data management for fingerprint recognition algorithm based on characteristic points' groups. In: Pechenizkiy, M., Wojciechowski, M. (eds.) New Trends in Databases and Information Systems. Advances in Intelligent Systems and Computing, vol. 185, pp. 425–432. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-32518-2_40
5. Szczepanik, M., Jóźwiak, I.: Reliability and error probability for multimodal biometric system. In: Korbicz, J., Kowal, M. (eds.) Intelligent Systems in Technical and Medical Diagnostics. Advances in Intelligent Systems and Computing, vol. 230, pp. 325–332. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-39881-0_27
6. Huk, M.: Measuring the effectiveness of hidden context usage by machine learning methods under conditions of increased entropy of noise. In: 3rd IEEE International Conference on Cybernetics, pp. 1–6. IEEE Press (2017)
7. Privitera, C.M., Azzariti, M., Stark, L.W.: Locating regions-of-interest for the Mars Rover expedition. Int. J. Remote Sens. **21**, 3327–3347 (2000)
8. Raczkowski, D., Canning, A.: Thomas-Fermi charge mixing for obtaining self-consistency in density functional calculations. Phys. Rev. B **64**, 121101–121105 (2001)
9. Mel, B.W.: The Clusteron: toward a simple abstraction for a complex neuron. In: Advances in Neural Information Processing Systems, vol. 4, pp. 35–42. Morgan Kaufmann (1992)
10. Spratling, M.W., Hayes, G.: Learning synaptic clusters for nonlinear dendritic processing. Neural Process. Lett. **11**, 17–27 (2000)
11. Knuth, D.: The Art of Computer Programming: Sorting and Searching, vol. 3. Addison Wesley, Boston (1998)
12. Bentley, J.L., McIlroy, M.D.: Engineering a sort function. Softw.: Pract. Exp. **23**, 1249–1265 (1993)
13. Astrachan, O.: Bubble sort: an archaeological algorithmic analysis. SIGCSE Bull. **35**, 1–5 (2003)