Implementing Contextual Neural Networks in Distributed Machine Learning Framework

Bartosz Jerzy Janusz(✉) and Krzysztof Wołk

Wroclaw University of Science and Technology, Wroclaw, Poland
{210004,220999}@student.pwr.edu.pl

Abstract. Contextual neural networks are generalization of multilayer neural networks. They possess interesting property of automatic selection of data attributes needed for correct processing of given input vectors. To achieve that they are using neurons with conditional, multistep aggregation functions and error generalized error backpropagation algorithm based on self-consistency paradigm. According to the literature of the subject, currently there are no implementations of those models in high-performance machine learning platforms like Mahout or MLlib. In this paper we present initial results of implementation of contextual neural networks in distributed machine learning framework called H2O. The motivation behind this work is the need to analyze properties of contextual neural networks and conditional multi-step aggregation functions while solving large classification problems.

Keywords: Classification · Self-consistency · Multi-step aggregation functions Generalized error backpropagation

1 Introduction

H2O is open source, distributed, parallel and scalable platform implementing core machine learning methods. It allows to build machine learning (ML) models on big data and has proven its industry-grade reliability [1–4] while solving real life problems [5–8]. H2O includes such algorithms as e.g. deep learning of multilayer neural networks, Gradient Boosting and Random Forest for creation tree-based models, ensemble learning, Principal Component Analysis, and k-Means clustering [9, 10]. It performs batch training of ML models, but can also make online predictions with use of Storm, but cannot train the models online. H2O realizes distributed computation with "distributed fork-join", which parallelizes jobs across many data nodes for efficient in-memory computation with a divide-and-conquer technique, and then combines the results. Final models can be exported as JAVA classes for further use.

H2O can be compared with other state-of-the-art ML frameworks such as Mahout, MLlib, Oryx, SAMOA, SINGA as well as WEKA, and among those is recognized for its usability, speed and extensibility [2, 4]. H2O includes interfaces to Java, R, Python and Scala and easily integrates with Spark. It provides also H2O Flow, an interactive, web-based notebook for manipulating data and building ML models using a hybrid of point-and-click and command-line approach. Basic start of H2O is as simple as:

N. T. Nguyen et al. (Eds.): ACIIDS 2018, LNAI 10752, pp. 212–223, 2018.
https://doi.org/10.1007/978-3-319-75420-8_20

unpacking its actual version to the chosen directory, loading included H2O server with "java –jar h20.jar" command and visiting local address http://localhost:54321 with favorite web browser for accessing H2O Flow web GUI. This allows H2O use on single workstation as well as in multi-node cloud environments, not only by programmers but also by researchers [8].

Although H2O implementations of ML algorithms are highly optimized for maximum performance, such approach causes that developers of H2O are highly concentrated only on basic ML models and methods. In the effect, a number of ML tools available in other frameworks can't be found in H2O. Thus one should also not expect from H2O realization of newest or highly specialized ML methods. This is why in this paper we present first approach to extend H2O Deep Learning functionality with capability of training contextual neural networks (CxNN) with Generalized Error Backpropagation algorithm [11–13].

In the effect, the rest of the paper is organized as follows. In Sect. 2, we briefly present contextual neural networks and the description of the generalized backpropagation algorithm. Then Sect. 3 includes detailed discussion of performed modifications of H2O architecture and shows related changes of H2O Flow web interface. Presented software is next used in Sect. 4 to solve selected UCI ML benchmark classification problems to experimentally test if properties of contextual neural networks built with H2O are as expected. Finally in Sect. 5, we discuss obtained results and possible areas of further development and research related with H2O and contextual neural networks.

2 Contextual Neural Networks and Generalized Error Backpropagation Algorithm

Considered contextual neural networks (CxNN's) were previously applied with success to solve benchmark as well as real-life classification problems [12, 13]. They were used for spectrum prediction in cognitive radio [14] and during research related to measuring awareness of computational systems through context injection [8, 15]. It was also shown, that they can be very good tools for fingerprints detection for crime-related analyses [16], can be helpful for solving such important problems as e.g. rehabilitation and elderly abuse prevention [17, 18]. Moreover it was shown, that they possess three important properties. The first property is that they are not black-boxes. The way how they operate allows to observe which input attributes are more important for classification of given data vectors. This can be very useful for many data-analytics-related tasks. The second property is that their neurons try to solve problems with use of as low number of input signals as possible, separately for each given data vector. This is done both during and after training of the neural network, practically without limiting model classification accuracy – and can considerably decrease costs of its use for data processing (especially time and energy costs). It was reported, that for many problems, above effect can limit connections activity of hidden neurons more than ten times in relation to analogous neural networks which process data with all connections. And the third important property of contextual neural networks is that the described behavior of their neurons during training causes further effects similar to outcomes of using dropout [13, 19–22]. The major difference between

both techniques is that dropout is not related to the data processed by the neural network, nor to the knowledge which the neural network already possess. And finally the decrease of neural network internal activity caused by dropout is not preserved after the training process – what is not the case for contextual neural networks. All the above makes contextual neural networks to be good candidates for implementation within H2O framework.

In detail, contextual neural networks described above are direct generalizations of known neural networks and can have well known architectures (e.g. MLP), but are using neurons with multi-step conditional aggregation functions. As the name suggests, such functions aggregate input signals of the neuron not in one but in multiple steps [12, 13]. Each step of aggregation is used to read-in given subset of inputs and to decide if already accumulated information is enough to calculate the output value of the neuron with needed precision. Typically, the steps of aggregation are realized until the neuron activation, cumulated from groups of inputs processed in previous steps, is lower than some constant aggregation threshold φ^*. Examples of those functions are Sigma-if, CFA and OCFA functions [11]. It is worth to note, that given ordered list of K groups of neuron inputs is sometimes called "scan-path". This is because multi-step conditional aggregation functions are closely related with Starks' scan-path theory [23].

Unlike other neural networks built with Sigma-Pi, Clusteron or Spratling-Hayes neurons [24, 25], contextual models with conditional multi-step aggregation almost do not need separate parameters to describe the composition and priorities of groups of inputs of each neuron [13]. Except two mentioned parameters for defining number of groups of inputs K and aggregation threshold φ^*, which can be common for all neurons within the neural network, all information about scan-paths is stored within connection weights. This improvement can be further exploited by applying self-consistency paradigm to train contextual neural networks with gradient algorithm – Generalized Error Backpropagation (GBP) [12, 13].

The GBP algorithm is a generalization of classical error backpropagation method extended to be able to train contextual neural networks which are using neurons with multi-step conditional aggregation functions [12]. To do that, GBP is applying self-consistency paradigm known from physics to use gradient based method to optimize both output error of the network and non-continuous, non-differentiable scan-paths of aggregation functions of the neurons [26]. The algorithm is maintaining mutual dependency between connection weights of given neuron and virtual, non-continuous parameters defining priorities and grouping of neuron inputs. Those virtual parameters describing scan-paths are calculated from connection weights with Ω function and there is no need to remember their values after the training. But during the GBP scan-paths are updated not more often than once per ω epochs to control the strength of the dependency between connection weights and scan-paths. Thus scan-paths must be preserved between epochs of their calculation in structures separated from connection weights.

For neurons with multi-step aggregation the Ω function consists of two operations: sorting N connection weights of the neuron 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 weights highest among other connections go to the second group, etc. Then the basic scan-path is defined as the list of groups from first to last. Finally scan-path is used by the aggregation function in a way that, groups of inputs are read-in one after

another until given condition is met. Without details of specific aggregation functions, the GBP algorithm can be represented as on the Fig. 1.

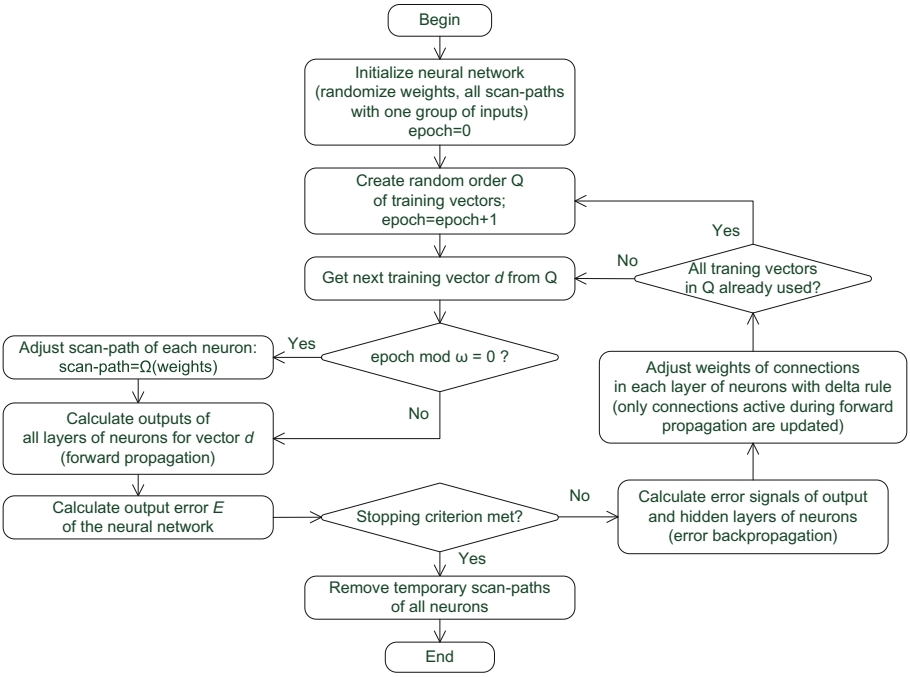

Fig. 1. Diagram of the generalized error backpropagation algorithm for given scan-path creation function Ω. Scan-paths update interval ω controls the strength of self-consistency between connections weights and virtual parameters of neuron aggregation functions. After the training, temporary scan-paths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

At the beginning and during the first ω epochs of GBP all neurons have all inputs assigned to the first group of their scan-paths. This is because at the beginning of training algorithm should not make any assumptions about the importance of the neuron inputs. Then, after each ω epochs scan-paths are updated with Ω function in accordance to actual connection weights.

3 Modification of the H2O.ai Framework

The previous section indicates that for scan-paths update interval $\omega = \infty$ the GBP algorithm behaves exactly as the classical error backpropagation method (BP). By analogy, contextual neural network with neurons using conditional multi-step aggregation for number of groups of inputs $K = 1$ behave exactly like multilayer perceptron network (MLP). This suggests that it should be possible to implement GBP method and contextual neural networks as direct extensions of actual realization of BP and MLP found in H2O. Unfortunately, analysis of highly optimized H2O code for distributed training of MLP

neural networks with BP algorithm extended with deep learning techniques, showed that needed modifications are not straightforward.

3.1 Adding Aggregation Functions and Conditional Backpropagation

Actual, third version of H2O server implements different activation functions of the neurons not as methods of the "Neurons" class but as specializations of "fprop" and "bprop" methods of activation related child classes of the class "Neurons". This creates dedicated neuron classes for each activation function, such as "Tanh" neuron, "Maxout" neuron, etc. Such construction is reasonable under the assumption that low number of different aggregation functions is considered, especially for existing case of only one: linear combinations of weights. But our goal was to extend H2O with several considered aggregation functions (e.g. Sigma-if, CFA, OCFA, Random, etc.). Continuing original approach would then imply creating huge number of specialized neuron classes or sub-classes, such as "Tanh_SigmaIf", "Tanh_CFA", "Tanh_OCFA",…, "Maxout_SigmaIf", etc. This would lead to unnecessary code duplication and problematic maintenance of the source. On the other hand, we wanted to keep our changes of the H2O code structure to be as small and centralized as possible. This lead us to use mixed approach.

Having in mind the above observation that specialized neuron classes of H2O, named as activation functions, in fact realize whole neuron transfer functions, we extended their set with single "Ext" neuron class. Inside we encapsulated possibility to use both different activation and aggregation functions, leaving the original H2O code intact. Additionally, within this class we exchanged the basic error backpropagation method "bprop" with implementation of generalized error backpropagation named "conditionalBprop". The structures and methods related to the latter are added within new "NeuronConnectionGroups" and "LayerConnectionGroups" classes. Finally, it is worth to note, that we equipped "Ext" class not only with new aggregation and activation methods (Sigma-if, CFA, OCFA, Bipolar Sigmoid, Leaky Rectifier) but also with their forms existing in original H2O code (weighted linear aggregation, hyperbolic tangent, etc.). This allows us to check how they work with other added activation and aggregation functions and/or with generalized backpropagation algorithm. It is also worth to note that after defining required aggregation methods within "Ext" class the H2O framework automatically takes care about distributing calculations among available nodes, because this is done at the level where internal mechanisms of the trained model are not important.

3.2 Measuring Connections Activity in Distributed, Multi-threaded Environment

Special attention we had to pay during implementation of structures and methods for analyzing connections activity. This is due two reasons. First, because such kind of analysis is characteristic for contextual neural networks and it was completely absent within H2O. And the second reason is that H2O is by default distributed, multi-threaded environment. Even when run on single multi-core processor, it automatically adapts number of threads and their load to maximize efficiency of its tasks. In the effect, the measurements of activity of hidden connections of contextual neural network must be realized in a Map/Reduce manner. Additionally, each thread in each computational node

within H2O cluster can train different neural networks during cross-validation procedure, as well as can process different portions of the training data. This is why it was decided that in presented version of the software, the possibility of hidden connections activity analysis will be limited to cases when the H2O cluster processes (trains or uses) neural network for one data set. Thus we assume that measurements of connections activity are not being done when cross-validation training is executed.

It is also worth to notice, that even without cross-validation, when multiple nodes and threads are available, H2O trains neural network in distributed way. In such case H2O independently maintains several local copies of the model and uses scheduled synchronization of their structures to create single, so called, shared model. At the end of the training process, shared model is returned as the result of the training process. Thus during the training we measure activity of connections only of the local neural networks – once after each epoch. In detail, when shared model is populated to all computational nodes, all threads are using this model to process disjoint portions of the given set of data vectors. For each data vector neurons of the local neural networks calculate their outputs and during this we are counting active inputs of the neurons. By summing up activities of all neurons of given neural network we get its number of active connections for given data vector. Finally, by summing up the activities for data vectors processed by all nodes and dividing this by the number of data vectors we get the final result – average activity of hidden connections of given model for given data set. This can be further presented also as the average percentage of active hidden connections of the local neural network (designated as avg_hca). By analogy, after the training we measure activity of hidden connections of the final, shared model of the neural network.

We have implemented the structures for aggregating data about activities of hidden connections of neural network within added "LayerConnectionsGroups" class. Measurement for given data vector is done within the call of forward propagation method "fprop". For simplicity we have modified the "fprop" function to return value being the number of active connections for processed data vector. Results of all "fprop" calls are then aggregated in 64 bit counters and averaged for all data vectors. Then such results for each epoch of training are saved to text file dedicated for given training. Such approach requires to perform measurements of activity of hidden connections with H2O parameter "train_samples_per_iteration" set to zero. This guarantees that the given set of data vectors will be processed by the neural network exactly one time during given epoch, regardless of the number of computing nodes.

3.3 Modifying the H2O Flow Web-GUI

Finally we have modified the original H2O Flow web application to be able to use new functionalities added to the H2O server. Its changes were limited to original DeepLearningModel.java and DeepLearningV3.java class files and were related to defining new parameters of H2O API as well as showing or hiding needed options within the web-GUI interface. Especially, we defined new "DeepLearningParameters.ExtActivation" parameter connected with the selection of our "Ext" transfer function computation with the GBP algorithm. We also decided to not change the name of original "activation"

parameter to keep compatibility of modified API with the original one. In the effect the example screenshot of the modified H2O Flow application can look as on the Fig. 2.

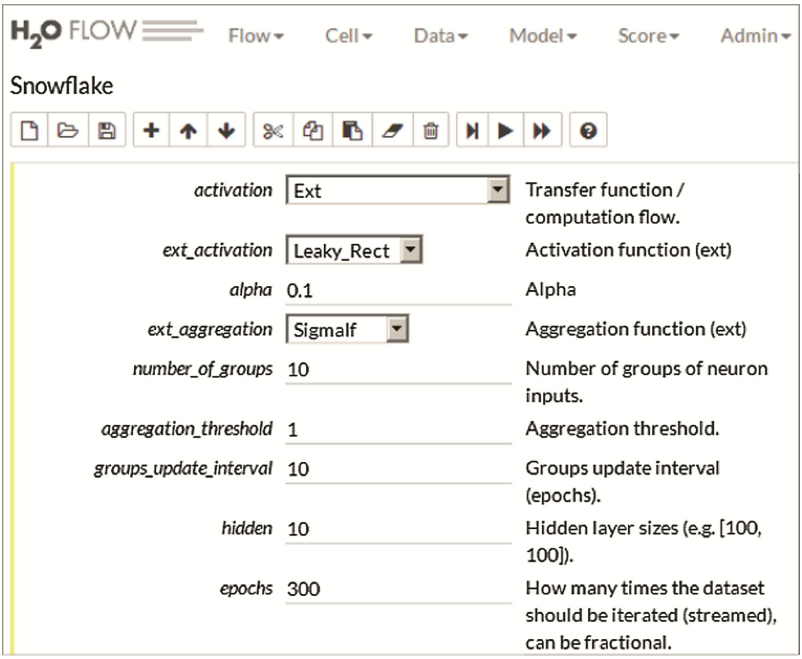

Fig. 2. Example fragment of the modified H2O Flow application with settings of characteristic parameters of contextual neural network prior to its training

One can notice, that to signal the true meaning of the "activation" parameter we have changed its description to "Transfer function/computation flow". After selecting its new "Ext" value, additional parameters are shown. Especially, the activation and aggregation functions can be selected by changing values of "ext_activation" and "ext_aggregation" parameters, respectively. Fields "number_of_groups", "aggregation_threshold" and "groups_update_interval" are related to parameters K, $\varphi*$ and ω, which in our implementation of GBP are common for all neurons within the neural network.

4 Results of Experiments

To verify correctness and analyze the properties of the presented modification of the H2O software we have run several tests. Here we present the most important results obtained for selected classification benchmark problems from UCI Machine Learning repository: Sonar, Crx and KDDCup (1999) [27]. Sonar and Crx are small data sets but are popular within the ML related literature. On the other hand, the KDDCup is one of the biggest UCI ML data sets (over 700 MB of training data). Analyzed properties of contextual neural networks were average classification accuracy as well as average

activity of hidden connections (avg_hca). The software was modified version of the H2O Server 3.11 and H2O Flow 0.52. The detailed experiments setup was as follows.

In all experiments we have used default settings of the H2O Flow, except the following changes. Cross-validation and adaptive rate of training step α were not used. Constant training step α was 0.1. Parameters score_training_samples and number of training samples per MapReduce iteration (train_samples_per_iteration) were set to zero, what means "all vectors in the training data set". After initial tests, during described experiments for Sonar and Crx problems, score_interval, maximal number of training epochs, aggregation threshold and interval of groups update were 0.1 s, 600, 0.6 and 25, respectively. For KDDCup10 (around 10% of full KDDCup data) and KDDCup data sets values of the same parameters were 1 s, 30, 0.001 and 3 respectively. Values of the other parameters, related to the processed data sets 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	Number of connections between neurons	Number of groups of neuron inputs (*K*)	Number of classes (network outputs)	Number of data vectors
Crx	60 (9)	10	400	10	5	690
Sonar	60 (0)	10	620	10	2	208
KDDCup10	119 (0)	100	14200	14	23	494022
KDDCup	124 (2)	100	14700	14	23	4898431

It is worth to notice that H2O by default sets the neural network architecture with use of extended "one-hot encoding" of inputs. Continuous attributes are represented by single input, and nominal attributes are represented with single binary input for every value. In addition to this standard method H2O performs also analysis of interactions between data attributes and in the effect extends the list of network inputs with additional elements. Counts of those extra inputs for considered data sets are given in Table 1 in round brackets. How interactions-based inputs are created is described in DataInfo class within H2O source code.

The results of experiments for Sonar and Crx data sets are presented on Figs. 3 and 4. They show how for both problems the average classification error decreases with subsequent epochs of GBP training of contextual neural networks. But it can be also noticed that in both cases with gradual decrease of classification error also the average activity of hidden connections semi-logarithmically drops down from 100% to 14% for Sonar and to 20% for the Crx problem.

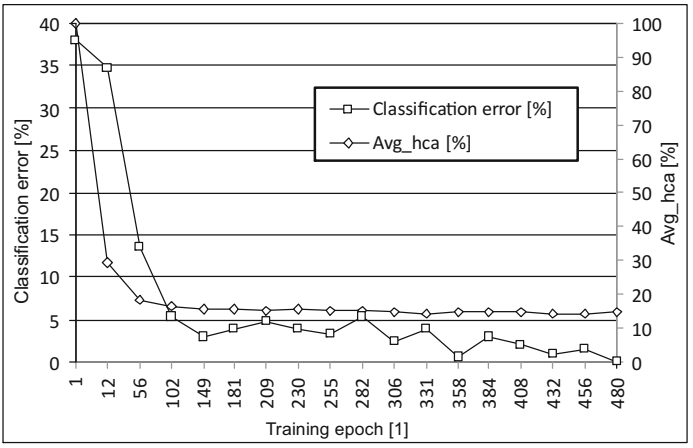

Fig. 3. Average classification error and hidden connections activity of contextual neural network solving Sonar problem, during its training with GBP algorithm implemented in H2O framework. Number of hidden neurons = 10, number of neuron inputs groups $K = 10$, aggregation threshold $\varphi^* = 0.6$, interval of groups update $\omega = 25$.

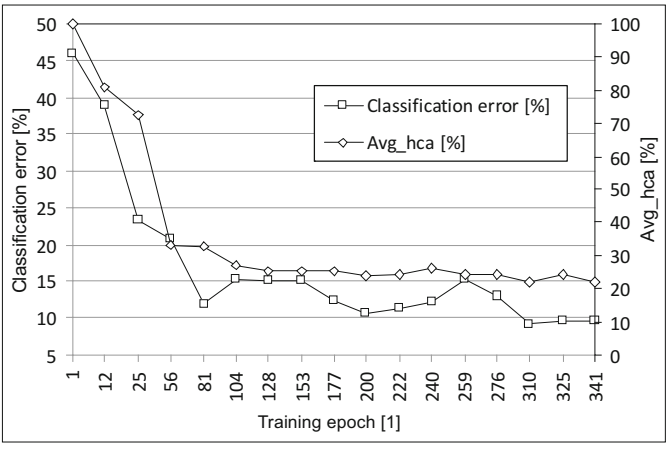

Fig. 4. Average classification error and hidden connections activity of contextual neural network solving Crx problem, during its training with GBP algorithm implemented in H2O framework. Number of hidden neurons = 10, number of neuron inputs groups $K = 10$, aggregation threshold $\varphi^* = 0.6$, interval of groups update $\omega = 25$.

The constant 100% activity level is the maximum of possible avg_hca results – e.g. for MLP neural networks or CxNN with number of neuron inputs groups K set to 1. During performed experiments at the beginning of GBP algorithm each CxNN had all inputs of all neurons assigned to the same, first group thus such models were behaving like the MLP. This caused that the initial avg_hca in both cases is 100%.

What is also interesting, the observed final classification accuracies of trained contextual models for Sonar and Crx problems measured for training data are comparable to the generalization (accuracy measured for test data) reported for MLP of analogous architectures. Most probably this is caused by the dynamic changes of contextual neural network architecture during the training which can prevent overfitting of constructed models.

After the above experiments two additional tests were performed for KDDCup 1999 and its 10% version dataset (designated in this paper as KDDCup10) to check the described solution against largest (in terms of training examples) problem served by UCI ML repository.

Then the results were compared with outcomes of training of two MLP neural networks of the same architecture – one originally implemented in H2O 3.10.0.3 software (MLP1), and the second one constructed as contextual neural network with number of groups $K = 1$ (MLP2). The results are given in Table 2.

Table 2. Hidden connections activities (avg_hca) and classification accuracies of the training data (train_acc) of contextual neural networks solving KDDCup 1999 benchmark problem from UCI ML repository. MLP1- original H2O implementation of Multi-Layer Perceptron, MLP2 – contextual neural network with $K = 1$, CxNN - contextual neural network for $K = 14$.

Training data set	MLP 1 avg_hca [%]	MLP 2 avg_hca [%]	CxNN avg_hca [%]	MLP 1 train_acc [10^{-3}%]	MLP 2 train_acc [10^{-3}%]	CxNN train_acc [10^{-3}%]
KDDCup10	100 ± 0	100 ± 0	22.6 ± 0.3	1.55 ± 0.04	1.60 ± 0.07	1.5 ± 0.08
KDDCup	100 ± 0	100 ± 0	22.5 ± 0.2	0.8 ± 0.4	0.9 ± 0.4	1.1 ± 0.3

Obtained measurements show that implemented contextual neural network works as intended. As expected, for both data sets the activity of hidden connections of CxNN for $K = 1$ is the same as for MLP model and equal 100%. Moreover, when the number of groups of neuron inputs is increased to $K = 14$ the avg_hca decreases almost five times in comparison to MLP. This is achieved without considerable decrease of classification accuracy. Such behavior is characteristic for CxNN models and proves correctness of GBP implementation. It can be also noticed that for both data sets the avg_hca of CxNN models is the same. This is also correct because both considered data sets represent the same classification problem. In such case, for given set of values of parameters, independently from the number of training vectors both CxNN need similar average number of active hidden connections to solve the problem.

5 Conclusions

The above text describes the first approach to implement contextual neural networks with multi-step aggregation functions in scalable, distributed ML framework. Presented results for selected classification problems from UCI ML repository show that the mechanisms added to H2O software are working as intended, including Generalized Error Backpropagation method and CxNN model which is direct generalization of the

MLP neural network. The value of constructed software lies in the fact that it can be used for further research on contextual neural networks and conditional multi-step aggregation functions. It allows to perform experiments with large data sets and easily distribute calculations among many nodes within available H2O clusters.

Further research in the presented area can include detailed analysis of computational efficiency and optimization of the modified H2O.ai code. It would be also valuable to test CxNN behavior with different aggregation functions and other large classification benchmark data sets, such as Poker Hand, Susy and HIGGS. But especially valuable would be to try CxNN to solve "URL Reputation" classification problem – which has not only large number of training vectors but also over three millions of attributes. This would be interesting test of abilities of contextual neural networks to dynamically select attributes needed for correct data processing.

References

1. Grolinger, K., Capretz, M.A.M., Seewald, L.: Energy consumption prediction with big data: balancing prediction accuracy and computational resources. In: 2016 IEEE International Congress on Big Data (BigData Congress), pp. 1–8 (2016)
2. Ng, S.S.Y., Zhu, W., Tang, W.W.S., Wan, L.C.H., Wat, A.Y.W.: An independent study of two deep learning platforms - H2O and SINGA. In: 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp 1–5. IEEE Press, Bali (2016)
3. Niu, F., Recht, B., Christopher, R., Wright, S.J.: HOGWILD!: a lock-free approach to parallelizing stochastic gradient descent. In: Advances in Neural Information Processing Systems, pp. 693–701 (2011)
4. Richter, A.N., Khoshgoftaar, T.M., Landset, S., Hasanin, T.: A multi-dimensional comparison of toolkits for machine learning with big data. In: 2015 IEEE International Conference on Information Reuse and Integration, pp. 1–8. IEEE, San Francisco (2015)
5. Suleiman, D., Al-Naymat, G.: SMS spam detection using H2O framework. Procedia Comput. Sci. **113**, 154–161 (2017)
6. Domingos, S.L., Carvalho, R.N., Carvalho, R.S., Ramos, G.N.: Identifying IT purchases anomalies in the Brazilian government procurement system using deep learning. In: 15th IEEE International Conference on Machine Learning and Applications (ICMLA) (2016)
7. Al Najada, H., Mahgoub, I.: Big vehicular traffic data mining: towards accident and congestion prevention. In: International Wireless Communications and Mobile Computing Conference, pp. 256–261 (2016)
8. 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)
9. Liang, M., Trejo, C., Muthu, L., Ngo, L.B., Luckow, A., Apon, A.W.: Evaluating R-based big data analytic frameworks. In: 2015 IEEE International Conference on Cluster Computing (CLUSTER), pp. 1–2. IEEE, Chicago (2015)
10. Cook, D.: Practical Machine Learning with H2O Powerful, Scalable Techniques for Deep Learning and AI. O'Reilly Media, Newton (2016)
11. 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)

12. 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)
13. 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)
14. Huk, M., Pietraszko, J.: Contextual neural-network based spectrum prediction for cognitive radio. In: 4th International Conference on Future Generation Communication Technology (FGCT 2015), pp. 1–5. IEEE Computer Society, London (2015)
15. Huk, M.: Context injection as a tool for measuring context usage in machine learning. In: Nguyen, N.T., Tojo, S., Nguyen, L.M., Trawiński, B. (eds.) ACIIDS 2017. LNCS (LNAI), vol. 10191, pp. 697–708. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54472-4_65
16. 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
17. Huk, M.: Using context-aware environment for elderly abuse prevention. In: Nguyen, N.T., Trawiński, B., Fujita, H., Hong, T.-P. (eds.) ACIIDS 2016. LNCS (LNAI), vol. 9622, pp. 567–574. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49390-8_55
18. Huk, M.: Context-related data processing with artificial neural networks for higher reliability of telerehabilitation systems. In: 17th International Conference on E-health Networking, Application & Services (HealthCom), pp. 217–221. IEEE Computer Society, Boston (2015)
19. Huk, M., Kwasnicka, H.: The concept and properties of sigma-if neural network. In: Ribeiro, B., Albrecht, R.F., Dobnikar, A., Pearson, D.W., Steele, N.C. (eds.) Adaptive and Natural Computing Algorithms, pp. 13–17. Springer, Vienna (2005). https://doi.org/10.1007/3-211-27389-1_4
20. Huk, M.: Sigma-if neural network as the use of selective attention technique in classification and knowledge discovery problems solving. Ann. UMCS Sectio AI – Inf. **4**(2), 121–131 (2006)
21. Huk, M.: Manifestation of selective attention in Sigma-if neural network. In: 2nd International Symposium Advances in Artificial Intelligence and Applications, International Multiconference on Computer Science and Information Technology IMCSIT/AAIA 2007, vol. 2, pp. 225–236 (2007)
22. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
23. 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)
24. 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)
25. Spratling, M.W., Hayes, G.: Learning synaptic clusters for nonlinear dendritic processing. Neural Process. Lett. **11**, 17–27 (2000)
26. Raczkowski, D., Canning, A.: Thomas-Fermi charge mixing for obtaining self-consistency in density functional calculations. Phys. Rev. B **64**, 121101–121105 (2001)
27. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml