# Weight Exchange in Distributed Learning

Julian Dorner
HTW Berlin
Berlin, Germany

Samuel Favrichon
Universite de Technologie de Compiegne
Compiegne, France

Arif Selcuk Ogrenci
Kadir Has University
Istanbul, Turkey

***Abstract*—Neural networks may allow different organisations to extract knowledge from the data they collect about a similar problem domain. Moreover learning algorithms usually benefit from being able to use more training instances. But the parties owning the data are not always keen on sharing it. We propose a way to implement distributed learning to improve the performance of neural networks without sharing the actual data among different organisations. This paper deals with the alternative methods of determining the weight exchange mechanisms among nodes. The key is to implement the epochs of learning separately at each node, and then to select the best weight set among the different neural networks and to publish them to each node. The results show that an increase in performance can be achieved by deploying simple methods for weight exchange.**

## I. Introduction

This paper presents the work done to compare the efficiency of different methods to exchange only weights between nodes where each node implements neural networks to perform training on its own data. The data at each node belong to the same problem domain, i.e. each node has the same set of attributes in data. The question addressed is the possibility of achieving a learning accuracy similar to the one obtained in training all of the data in one node. A use case is easy to imagine: different institutions like hospitals, institutes for statistics, insurance companies etc. have data to be analysed with almost identical set of input and output features. In general, they would obtain a better accuracy at predicting some behaviour if they could use all of the data, say for training a neural network. However, they can not (do not want to) share their data because of legal and commercial concerns. The research question is, how to design mechanisms to perform distributed learning for neural networks. The problem is also relevant in the case of massive data that needs to be spread amongst different computers.

Previous work in this area explored theoretical and practical aspects of the problem. Consensus based approaches allow efficient implementation in support vector machines and backpropagation based multilayer perceptron structures [1]-[3]. Another method has been proposed in [4] to use averaging among weights in obtaining the consensus among nodes for random vector functional link neural networks. Furthermore, recent literature also includes increased interest in distributed learning in wireless sensor networks implementing support vector machines [5], evolutionary algorithms [6], and dictionary learning for big data [7]. The purpose of this paper is not to address big data problems (where the data doesn't fit in a single machine) or solve performance issues but rather we offer simple to implement solutions to see how these solutions would indeed improve the efficiency of the distributed learning in case of feedforward networks. In this respect, the methods offered do not discriminate techniques that use a master node or where nodes perform peer-to-peer communication. All the techniques used here can be implemented using both solutions. We present here the effects of different weight exchange mechanisms between distributed nodes of neural networks on the performance. The performance is measured as the accuracy obtained for the classification problem as will be explained later. The remainder of the paper is organised as follows: Section II describes the example problem and the necessary data preprocessing. The methodology will be explained in Section III along with the alternative methods for weight exchange. Next, Section IV displays the results including a comparison of methods, and lastly Section V will include the conclusions and future research directions.

## II. Problem Case

The problem to be investigated is based on the ”census dataset” also known as the ”adult dataset”. It contains various records extracted from the 1994 census database of USA. The samples are individuals of age above 16, and the feature to predict is if the individual makes an annual income over 50.000$ [8]. There are several reasons for picking this particular data set:

- Number of instances: There is a need to have a large enough number of instances. They should be split into multiple data sets without losing the ability to perform learning for predictions based on individual sets to be located in separate nodes.
- Real world problem: The techniques would be useful in a domain where the data owner wouldn't be willing to share their data. Personal data (as we have here) is a good example of that kind of data.
- Easy to learn: The classification problem here is not a hard one. This is important because the aim of this work is to compare methods with limited processing power. It is desired to have meaningful results to be able to show the improvements easily.

Most of the 14 features in this data set are self explanatory, like *age*, or occupation related features but they need to be preprocessed so that the neural network can accept them as numerical inputs in a scaled form for better performance [9].

Firstly, features that do not carry useful information, are removed from the dataset. For instance *fnlwgt*, describes a weight that is state dependent and thus cannot be used to compare individuals across states. Furthermore, *education-num* is just the equivalent of the *education* feature but in a numeric form, thus it doesn't bring any new information. Then, for all the categorical features (*workclass, education, marital-status, occupation, relationship, race, sex, native-country, income*), one-hot encoding is applied in order to turn them into simple numeric attributes. In this way, an attribute is created for each possible value in each feature. This technique increases the number of input features but simplifies their encoding. Numeric attributes *(age, capital-gain, capital-loss, hours-per-week)* are scaled to a distribution having zero mean and unit variance. This prevents an attribute with high values to have too much weight and this also helps the neural network to learn faster. After performing these operations we have a dataset containing 48842 samples with 216 attributes, 2 of them being the income (over 50K or not) as the target feature.

## III. Methodology

In this section, different methods of weight exchange and the tools used in testing these methods are explained. The goal of each method is to improve the learning performance of the neural network to get the best accuracy possible in a fixed number of training epochs. The expected result of this experiment should be that all of the parties agree on a set of weights at the end of training by sharing preferably no data, or by sharing minimum amount of data. In order to get reproducible results, and in order to evaluate our results independently, samples are split. The initial data contain 48842 instances. Firstly, 8842 of the instances are selected randomly which will be used for the final testing, named as the verification set. Then the remaining 40.000 samples are split according to the number of nodes (parties with individual neural network learning systems). In the following simulations, 4 nodes are employed each containing 10000 samples.

### A. Tools

The multilayer perceptron neural network is used as the learning model as it's behaviour is easy to understand. The input layer contains 214 features. The hidden layer contains 100 neurons. A sigmoidal activation function is used for the neurons in the hidden layer, and they are connected to the output layer containing 2 neurons (one for each class of being above or below 50K earning). The output neurons use a softmax activation function. Furthermore, a dropout mechanism with a probability of 0.5 has been added after the hidden layer to prevent overfitting [10].

The training is done by minimising the mean squared error using the Adagrad algorithm [11]. Adagrad gives us an adaptive learning rate that converges faster. The neural network is implemented using the Keras Python library working on top of the Theano platform [12]. The Keras[13] library offers a high level API to interact and to easily prototype neural networks on top of neural network backend platforms such as Theano or TensorFlow. The speed performance is not the concern here that's why high-level languages and simple laptop CPU can be used to perform the tests. Confusion matrices will be used to compare the accuracies obtained by the methods. There are two classes to be discriminated: a sample is classified as A if the sample has a revenue over 50.000$ per year, and it is of class B otherwise. A sample confusion matrix is given in Table I. The confusion matrix will give us actual and predicted classes for both types of samples. The instances on the diagonal from upper left to lower right are the rightly classified instances. The overall accuracy of this particular example is 80%. In the following subsections, the alternative approaches for weight exchange among nodes will be explained in detail.

TABLE I. Confusion Matrix Example

| | class A | class B |
|---|---|---|
| classified as A | 60% | 15% |
| classified as B | 5% | 20 % |

### B. Collage Testing

The first approach for the weight exchange includes some cheating by using a distinct testing set shared by all parties. Each node voluntarily shares a small fraction of its data, 10% in our simulations, that will be used as the collage testing pool. In a loop of training, each node performs usual training on its own data (9000 samples). After each epoch, the performance of each network (node) is measured against that collage testing pool of samples. The weights performing the best on the collage testing set will be exchanged to each of the nodes to be used in the next epoch of training. At the end of the training, that is when the stopping criteria are met, the final winning network is also tested against the verification set (8842 samples) for evaluation. The basic dataflow of collage testing is depicted in Fig. 1.

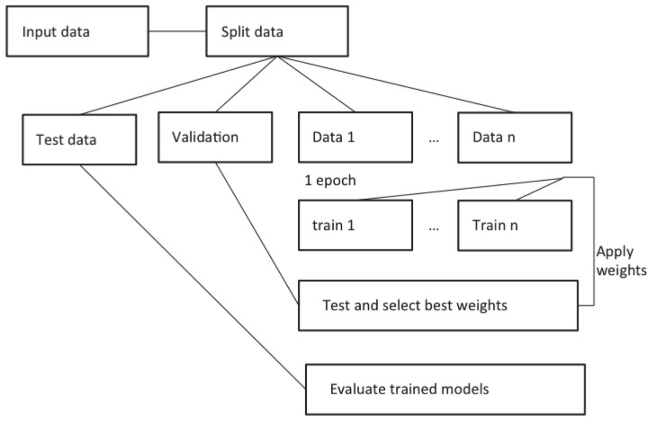

Fig. 1. Dataflow for the collage testing.

### C. Cross Testing

This approach doesn't need a shared testing set. After each epoch, each of the models is tested against all the other nodes' training dataset. This allows the parties involved to not share any data. The verification test set is still used to compare the accuracy of the different models. The dataflow of cross testing is given in Fig. 2. At each iteration of the learning phase, each node publishes its weight set to the other nodes. Then, each node tests its own data with the weights of the other nodes, and publishes the results. An example performance matrix obtained in this process is given in Table II. Here, it is evident that weights obtained in node 2 (column

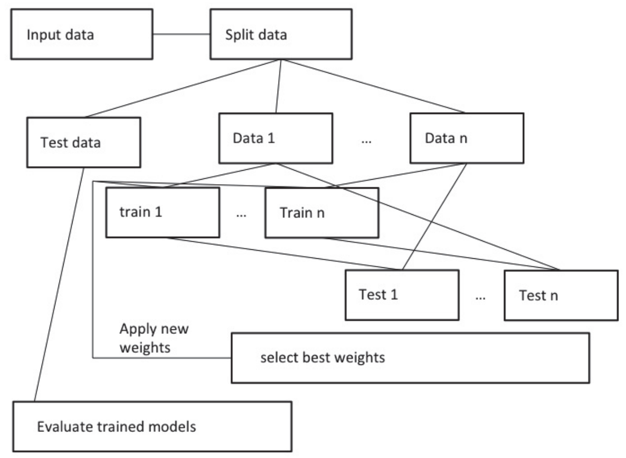

Fig. 2. Dataflow for the cross testing.

2) performs on the average (79%) higher than the other nodes.

The best weights are then used as the weights for all models in the next epoch of training. There are two ways to pick the best weights.

1) Pick the weights of that node that performs highest on the average across all the testing sets (practically all the data of 40.000 samples).
2) Pick the weights of that node that performs the best on the maximum number of test nodes.

## IV. Results

### A. Base Results

Firstly, we need to see how the neural network performs when trained on our whole input data set. For this purpose, a validation split of 0.1 is used in the training carried out with 40.000 samples. The classification obtains 100% accuracy in both the training and verification (8842 samples) sets. Thus, we have obtained a model that has learnt on the data set and it can output accurate predictions.

The next step is to see how each node would perform if they are trained with their local data only. That is, each node has access to its own data for training and to the verification data set for final testing. In that case, if we keep the same number of epochs and the same neural network configuration, the neural networks perform extremely badly on most of the cases. The outputs for the verification set come out to be equivalent to a ZeroR prediction i.e. all the samples are classified as if they belong to the class with the most elements [14]. In some cases the networks are able to learn, but it is mostly random as can be seen in the following outputs. We also have to be careful about the initial distribution in the sub data sets: If the data are not distributed in a balanced manner to the four nodes, i.e. if each of the nodes does not include approximately the same percentage of each category, the network will not be able to train properly due to the unbalance of classes. The results shown in the following tables are based on testing of the independent verification data set containing 8842 instances. Table III and Table IV display testing performances for individually trained nodes without weight exchange for unbalanced and balanced distribution of samples respectively.

TABLE II. Performance (% Correct) Matrix Example

| Data of / Weights from node | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 82 | 72 | 62 | 52 |
| 2 | 80 | 90 | 70 | 60 |
| 3 | 69 | 79 | 89 | 59 |
| 4 | 65 | 75 | 85 | 55 |
| Average | 74 | 79 | 76.5 | 56.5 |

TABLE III. Unbalanced Data Set Classification Performance for 4 Nodes

| | | | |
|---|---|---|---|
| 0 | 24 | 23,9 | 0,1 |
| 0 | 76 | 0 | 76 |
| 0 | 24 | 0 | 24 |
| 0 | 76 | 0 | 76 |

It is evident that only one of the nodes is able to perform correct classification whereas the other three nodes classify all samples as class B.

TABLE IV. Balanced Data Set Classification Performance for 4 Nodes

| | | | |
|---|---|---|---|
| 18.5 | 5.2 | 14.6 | 9.2 |
| 0 | 76.2 | 0 | 76.2 |
| 23.5 | 0.2 | 22.6 | 1.1 |
| 0 | 76.2 | 0 | 76.2 |

The verification (test) results for a balanced distribution are not bad but they are worse than what we can achieve with all the data combined. Two of the nodes display false classification ratios of 5.2% and 9.2% respectively. These are the results that can be improved by exchanging the weights between the parties.

### B. Collage Results

The classification performance (in %) of the nodes for a collage testing is given in Table V for random splitting of the training data to instances. It can be stated that collage testing allows almost full performance.

TABLE V. Collage Results for Random Splitting into 4 Nodes

| | | | |
|---|---|---|---|
| 23.7 | 0.01 | 23.7 | 0.01 |
| 0 | 76.2 | 0 | 76.2 |
| 23.7 | 0.01 | 23.7 | 0.01 |
| 0 | 76.2 | 0 | 76.2 |

### C. Cross Testing Results

The classification performance (in %) of the nodes with the cross testing are given in Table VI and Table VII for the two methods of picking the weights to be exchanged respectively: based on the best average performance in all nodes and based on the best performance in the majority of nodes. In both cases, the training data are randomly divided to instances. Keeping in mind that those performance values are for the independent verification (test) set of 8842 samples, it can be stated that weight exchange method based on the best average performance, allows almost full performance without sharing any local data with other nodes.

TABLE VI. Cross Testing Results: Best Average Performance, Random Splitting

| | | | |
|---|---|---|---|
| 23.7 | 0.1 | 23.7 | 0.1 |
| 0 | 76.2 | 0 | 76.2 |

| | | | |
|---|---|---|---|
| 23.7 | 0.1 | 23.7 | 0.1 |
| 0 | 76.2 | 0 | 76.2 |

TABLE VII. Cross Testing Results: Best Performance in Majority, Random Splitting

| | | | |
|---|---|---|---|
| 20.4 | 3.3 | 20.8 | 2.9 |
| 0 | 76.2 | 0 | 76.2 |

| | | | |
|---|---|---|---|
| 21 | 2.7 | 20.9 | 2.8 |
| 0 | 76.2 | 0 | 76.2 |

## D. Comparing Results

In Table VIII, a comparison of test performances of different methods is given.

TABLE VIII. Comparison of Verification Set Performance (%)

| Method Used | Average Accuracy |
|---|---|
| Random Splitting | 96.08 |
| Collage Testing | 99.9 |
| Cross Testing - average performance | 99.9 |
| Cross Testing - majority | 97.08 |

It can be noticed that using those weight exchange methods has led to an improvement in the accuracy obtained. The collage and cross testing achieve similar accuracy on this dataset, so there is no obvious way to choose. However, collage testing requires the parties to share a small amount of their data, and is thus less optimal.

## V. Conclusion

The major contribution of the paper can be stated as follows: In a neural network based learning task of distributed data, it is possible to obtain an accuracy almost as good as the one obtained on the complete data set, by training multiple models on sub data sets and by only exchanging the weights in a smart way. Some of the ways to exchange those weights, by using simple to understand techniques, have been investigated for a real life case problem. It is evident that this problem can have multiple solutions. The solution chosen should be the one that is the easiest to implement given the data available and given the algorithm being used. Some other issues arise when implementing this solution to a real life problem. The parties must first decide on the implementation of the algorithm, and on common conventions regarding the data. The data processing questions are not addressed here. Some characteristics of the dataset need to be looked into before using it for the analysis. For instance the repartition of the classes or hidden relationship between variables can lead the impossibility to build a correct predicting model.

There are several issues to be investigated further: Larger data sets with more complex learning requirements have to be tested in order to generalise the results obtained. For instance, it would be interesting to carry out research about how these techniques could be applied to a regression problem. The uneven distribution of data samples to nodes, both in terms of size and character, may have striking effects on the performance of weight exchange methods. It could also be interesting to see how different algorithms could benefit from parameter sharing techniques beyond weight exchanges. Another future research topic can be to look at meta classifiers as described in [15]. Lastly, computational performance issues have to be addressed adequately, especially for big data problems.

## Acknowledgment

The authors would like to thank to Kadir Has University and to the Erasmus Program as this work has been carried out as two of the authors were Erasmus exchange students at Kadir Has University.

## References

[1] P.A. Forero, A. Cano,and G.B. Giannakis, ”Consensus-based distributed support vector machines,” Journal of Machine Learning Research, vol. 11, pp. 1663-1707, 2010.

[2] O. Dekel, R. Gilad-Bachrach, O. Shamir,and L. Xiao, ”Optimal distributed online prediction using mini-batches,” Journal of Machine Learning Research, vol. 13, pp. 165202, 2012.

[3] L. Georgopoulos and M. Hasler, ”Distributed machine learning in networks by consensus,” Neurocomputing, vol. 124, pp. 2-12, 2014.

[4] S. Scardapane, D. Wang, M. Panella,and Aurelio Uncini, ”Distributed learning for random vector functional-link networks,” Information Sciences, vol. 301, pp. 271284, 2015.

[5] W. Kim, M. L. Stankovic, K. H. Johansson, and H. J. Kim, ”A distributed support vector machine learning over wireless sensor networks,” IEEE Transactions on Cybernetics, vol. 45, pp. 2599-2611, Nov. 2015.

[6] Yue-Jiao Gonga, Wei-Neng Chena, Zhi-Hui Zhana, Jun Zhanga, Yun Lid, Qingfu Zhange, and Jing-Jing Li, ”Distributed evolutionary algorithms and their models: A survey of the state-of-the-art,” Applied Soft Computing, vol. 34, pp. 286-300, 2015.

[7] H. Raja and W. U. Bajwa, ”Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data,” IEEE Transactions on Signal Processing, vol. 64, pp. 173-188, Jan. 2016.

[8] M. Lichman, UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/Census+Income, Irvine, CA: University of California, School of Information and Computer Science, 2013.

[9] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Mller, ”Efficient backprop,” Neural Networks: Tricks of the Trade Book Series: Lecture Notes in Computer Science, vol. 1524, pp. 9-50, 1998.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ”Dropout: A simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.

[11] J. Duchi, E. Hazan, and Y. Singer, ”Adaptive subgradient methods for online learning and stochastic optimization,” Journal of Machine Learning Research, vol. 12, pp. 2121-2159, 2011.

[12] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio, ”Theano: new features and speed improvements,” Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[13] F. Chollet, ”Deep learning library for Python,” https://github.com/fchollet/keras, 2015.

[14] I. H. Witten and E. Frank, Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2005.

[15] A. Prodromidis, P. Chan, and S. Stolfo, Meta-Learning in Distributed Data Mining Systems: Issues and Approaches, in Advances in Distributed Data Mining, AAAI Press, 2000.