ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Multioutput Regression Neural Network Training via Gradient Boosting*

Seyedsaman Emami[1] and Gonzalo Martínez-Muñoz[1]

Universidad Autónoma de Madrid - Escuela Politéctica Superior
Francisco Tomás y Valiente, 11, 28049 Madrid - Spain

Abstract.
A novel sequential procedure to train the final layers of a multi-output regression neural network (NN) based on Gradient Boosting is proposed, where the NN is an additive expansion of the Gradient Boosting. The method works by training portions of the network in an iterative manner in such a way that each new portion of the NN is learnt to compensate for the errors of the already trained portions, and the final result of the network forms by provided weight and the last hidden layer output. This is in contrast to the standard training of NNs in which the whole network is trained to learn the concept at hand. Extensive experiments show the good performance of the proposed method with respect to NN.

1 Introduction

The focus of the Multi-output regression is to predict multiple continuous targets concomitantly from a common set of input attributes. A straight forward way to tackle this problem is to break it into multiple single output problems and to learn one model for each problem. However, it can be beneficial to share part of the model in between the different targets [1, 2]. For instance, by sharing the main structure of the multiple-output neural networks (NNs).

On the other hand, the Gradient Boosting (GB) framework [3] has demonstrated to be among the best machine learning methods for solving tabular data problems [4]. The main procedure for gradient boosting based method is to built sequentially an ensemble as an additive sequence of learners, such that each new model learns the residual information not learned by previous learners. [3]. These approaches work by combining, generally, decision trees as their base classifiers. Although, the gradient boosting procedure has also been combined with neural networks [5, 6]. In [5], a residual network that uses the functional gradient minimization is proposed. Likewise, in [6] ResNet with the use of Gradient Boosting is reconstructed by augmenting the residual block with a linear binary classifier and continued to train the weights. Nevertheless, these studies are designed for single output classification tasks only.

In this study, we propose the creation of a multi-output neural network that is trained in phases using gradient boosting. An extensive performance analysis is carried out that shows that the proposed method works generally better than standard neural networks.

*The authors acknowledge financial support from PID2019-106827GB-I00/AEI/10.13039/501100011033. And also acknowledge the Centro de Computación Científica — UAM, for the computational resource and time.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

2 Proposed method

In this section, we describe the proposed approach for training a single multi-output regression neural network using gradient boosting. This method is an adaptation for multi target regression of the single output Gradient Boosting Neural Network (GBNN) [7]. GBNN trains the parameters of the neural network in phases and sequentially using gradient boosting, instead of jointly as in any standard back-propagation algorithm.

In order to train a multi target regression network, as the one shown in Fig. 1, at each step only a horizontal portion of it is trained (e.g. the part highlighted with dark lines in Fig. 1). Each new portion of the network is trained using a different set of output targets, as defined by the GB procedure, in such a way that the additive combination of all trained sub-networks given by the output layer units would produce the final prediction.

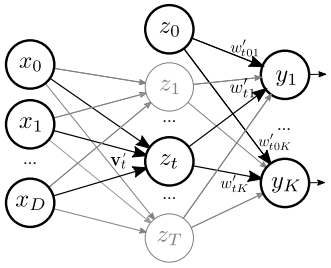

Fig. 1: Multioutput Regression Neural Network Training via Gradient Boosting

In detail, given a multi-output dataset of N instances as $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \mathbb{R}^K$, where D is the number of attributes and K the number of output variables, the objective of a multi-output regression algorithm is to find K functions, $\hat{\mathbf{F}} = \{\hat{F}^{\{k\}}\}_{k=1}^K$, that minimize the empirical loss

$$\sum_{i=1}^{N} L(\mathbf{y}_i, \hat{\mathbf{F}}(\mathbf{x}_i)) \ . \tag{1}$$

In this work, we will only consider the square loss $L\big(\mathbf{y}, \mathbf{F}(\mathbf{x})\big) = \frac{1}{2}\big(\mathbf{y} - \mathbf{F}(\mathbf{x})\big)^2$. A different loss function could be used although a second optimization step would need to be done after each portion of the network is trained like in standard gradient boosting [3].

In the proposed gradient boosting neural network, the set of functions $\mathbf{F}$ are approximated incrementally as an additive sequence $\hat{\mathbf{F}}_t(\mathbf{x}_i) = \hat{\mathbf{F}}_{t-1}(\mathbf{x}_i) + \rho \mathbf{h}_t(\mathbf{x}_i)$, where $\mathbf{h}_t(\mathbf{x})$ is a multi-output neural network for the t^{th} boosting iteration and ρ is the learning rate term, useful to regularize [3]. The construction of each term of the sequence, $\mathbf{h}_t(\mathbf{x})$, should reduce the loss function

$$\mathbf{h}_t(\cdot) = \underset{h}{\operatorname{argmin}} \sum_{i=1}^{N} L(\mathbf{y}_i, \mathbf{F}_{t-1}(\mathbf{x}_i) + \rho \mathbf{h}_t(\mathbf{x}_i)) \ . \tag{2}$$

where the first term of the sequence is initialized with the constant value that minimizes Eq. 1, which for the square loss is the mean value of the targets $\mathbf{F}_0 = \bar{\mathbf{y}}$. The rest of the terms of the sequence, which correspond to portions of the network as shown in Fig. 1, are trained using a standard back-propagation

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

algorithm using as targets the residuals of the previous phase

$$\mathbf{r}_{ti} = -\frac{\partial L(\mathbf{y}_i, \mathbf{F}(\mathbf{x}_i))}{\partial \mathbf{F}(\mathbf{x}_i)}\bigg|_{\mathbf{F}(\mathbf{x})=\mathbf{F}_{t-1}(\mathbf{x})} = \mathbf{y}_i - \mathbf{F}_{t-1}(\mathbf{x}_i)\ . \tag{3}$$

The weights of the network that are trained at each step are disjoint except for the weights from the bias term z_0 to the output layer (see Fig. 1). Hence, in order to obtain the correct output, at the end of the process the bias weights have to be reassigned to the sum of all steps (including step 0)

$$\omega_{0k} = \bar{y}_k + \sum_{t=1}^{T} \rho \omega_{0k}^t \tag{4}$$

In addition, the rest of the weights from the hidden layer to the output layer have to be multiplied by the learning rate ρ. Note that, we are considering only shallow networks. The adaptation to deeper networks is not straightforward. In any case, shallow models can be much better than deeper ones for tabular data [4]. Furthermore, at each step J internal neurons could be trained, and not just one as shown in Fig. 1. Finally, subsampling the data at each step is, like the learning rate, a good way to regularize [8].

3 Experiments

The performance of the proposed method, Multi-output Gradient Boosted Neural Network (GBNN-MO)[1] is investigated on 10 multi-output regression datasets with different numbers of instances, features, and targets. The summary of the analyzed datasets is shown in Table 1. The proposed method is compared with respect to a standard multi-output neural network (NN-MO), single-output Gradient Boosted Neural Network (GBNN-SO) and single-output neural network (NN-SO). The single output methods have to be trained independently once per target. The experimental comparison setup is done using a common three-fold cross-validation for all methods. For each of the three partitions train/test of each dataset the following steps are carried out: (i) the train partition is normalized so that all attributes have zero mean and one variance. The test partition is also normalized using the parameters obtained from the train set; (ii) the optimal values for the hyper-parameters for each method are obtained using a three-fold cross-validation within the training set. The values for the grid search for the Neural Network (NN-MO and NN-SO) are the number of neurons in the hidden layer and set to [1,3,5,7,11,12,17,22,27,32,37,42,47,52,60,70,80,90,100,150,200]. For both GBNN-MO and SO, the grid was set to [0.025,0.1,0.5,1] for learning rate (ρ), [0.5,0.75,1] for the subsample, and [1,2,3,5] for the number of neurons to be learned at each step (J). The total number of internal neurons is set to $T = 200$ for GBNN-MO and SO. All NN and GBNN methods use Adam as the internal optimizer and trained for 200 epochs; and (iii) the four methods were tested on the test set and the average values for the three partitions are reported.

[1] github.com/GAA-UAM/GBNN-MO/

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Dataset	Instances	Features	Targets	Dataset	Instances	Features	Targets
atp1d [1]	337	411	6	rf2 [1]	9,125	576	8
atp7d [1]	296	411	6	scm1d [1]	9,803	280	16
oes10 [1]	403	298	16	scm20d [1]	8,966	61	16
oes97 [1]	334	263	16	sf1 [9]	323	10	3
rf1 [1]	9,125	64	8	sf2 [9]	1,066	10	3

Table 1: The datasets used in the experiments and details

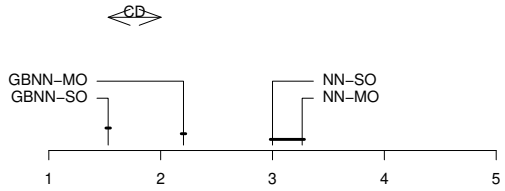

Fig. 2: Demsar diagram for the tested models (more details in the text)

The average performance for each method and target, measured as Root Mean Square Error (RMSE), is shown in Table 2. The table shows the targets in columns and the datasets and methods in rows. The best method result for each target is highlighted in bold face.

The results shown in Table 2 indicate that GBNN-MO is in general superior to the standard multi-output NN training method (NN-MO) in the studied datasets. NN-MO only achieves a better performance than GBNN-MO in: four targets of *oes10*, nine targets of *oes97*, five targets of *rf2* and a two targets in *rf1*. The proposed multi-output method outperforms NN-MO in 78 out of 98 of the tested targets. Comparing the multi-ouput and single-output versions of GBNN, the results are more even. Both methods obtain very similar RMSE values. Although GBNN-SO is better than GBNN-MO in 68 targets and in 30 targets GBNN-MO outperforms GBNN-SO. The same observations can be made for the differences between NN-MO and NN-SO.

In Figure 2, an overall comparison between the performance of the models is illustrated graphically using a Demsăr diagram [10]. The Figure 2 describes the average ranks, where the lower value denotes better performance. Statistical differences between the models is determined with a Nemenyi test. Two methods are statistically different if the difference in average rank is above the critical difference ($CD = 0.47$ for four models and 98 regression targets and p-value = 0.05). In the diagram, models whose performance is statistically different are not linked. As shown in the diagram, both GBNN-MO and SO are better than NN-MO and SO with statistical significance.

Considering the computational performance of the different methods, we have carried out an experiment to measure the training time and the average time to predict an instance for four datasets with a variety in the number of targets. The number of neurons of the hidden layer for the NN methods and the number of neurons of the GBNN-MO and SO are set to 200, the learning rate and

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Dataset	Method	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16
sf1	NN-MO	0.4146	0.4833	0.1638													
	NN-SO	0.4129	0.4999	0.1590													
	GBNN-MO	**0.3929**	0.4573	0.1524													
	GBNN-SO	0.3931	**0.4500**	**0.1445**													
sf2	NN-MO	0.7992	0.3044	0.1004													
	NN-SO	0.7950	0.3020	0.0969													
	GBNN-MO	0.7899	**0.2871**	0.0921													
	GBNN-SO	**0.7892**	0.2880	**0.0915**													
atp1d	NN-MO	99.53	155.92	151.83	144.50	106.50	146.04										
	NN-SO	92.85	157.95	161.04	151.33	100.95	150.40										
	GBNN-MO	52.37	**98.35**	**80.78**	60.84	55.78	65.20										
	GBNN-SO	**46.82**	107.10	84.93	**60.73**	**52.48**	**64.97**										
atp7d	NN-MO	119.23	166.24	174.41	170.83	125.90	171.06										
	NN-SO	115.59	171.00	181.85	179.14	126.35	179.58										
	GBNN-MO	34.46	79.83	67.16	62.19	**39.21**	64.22										
	GBNN-SO	**33.43**	**78.21**	**66.50**	**56.08**	44.06	**62.89**										
rf1	NN-MO	34.06	0.85	35.07	21.68	15.49	4.00	9.64	9.28								
	NN-SO	22.68	0.93	28.55	18.52	9.22	**2.03**	**4.24**	**5.59**								
	GBNN-MO	**18.46**	7.79	20.89	**13.64**	**7.16**	8.45	6.94	7.07								
	GBNN-SO	19.22	**0.78**	**20.28**	14.74	7.52	2.22	5.85	6.37								
rf2	NN-MO	8.53	**0.75**	**8.14**	5.32	6.45	2.42	4.89	7.28								
	NN-SO	8.66	0.90	8.16	5.31	4.75	1.22	3.71	**3.62**								
	GBNN-MO	11.23	5.24	9.61	6.69	**3.86**	5.00	2.86	4.46								
	GBNN-SO	**6.29**	0.75	8.31	**5.02**	5.94	**0.86**	**2.16**	4.64								
scm1d	NN-MO	112.1	123.6	116.1	126.0	121.8	134.2	150.5	164.4	153.5	161.0	121.9	133.3	154.3	167.5	164.7	177.1
	NN-SO	108.3	121.5	117.3	131.3	156.2	174.0	170.5	183.4	108.0	122.1	119.7	129.9	147.2	165.8	157.1	167.4
	GBNN-MO	72.6	78.8	75.6	83.3	**78.1**	**88.7**	**93.7**	**107.3**	99.3	105.4	80.7	88.7	100.5	112.2	113.3	119.4
	GBNN-SO	**62.1**	**73.4**	**72.7**	**78.9**	88.3	101.8	102.1	108.3	**67.5**	**76.6**	**70.7**	**81.6**	**90.6**	**98.7**	**93.4**	**98.5**
scm20d	NN-MO	156.7	173.4	164.4	185.2	177.3	195.4	218.5	235.0	229.8	243.7	174.9	194.4	223.7	241.8	244.5	261.5
	NN-SO	135.5	151.0	156.4	173.7	203.8	221.2	227.2	245.5	141.1	162.7	155.4	175.1	197.5	215.5	212.5	229.1
	GBNN-MO	105.8	113.9	113.2	127.2	**118.1**	**129.7**	**147.9**	161.1	157.2	165.9	115.1	129.7	152.5	162.3	166.5	179.1
	GBNN-SO	**87.1**	**97.4**	**99.6**	**111.8**	133.9	142.8	148.9	**158.2**	**96.2**	**110.2**	**101.0**	**112.8**	**128.6**	**141.1**	**138.7**	**146.7**
oes10	NN-MO	208.8	290.7	509.2	366.4	837.5	380.8	840.7	2323.7	**237.9**	934.6	288.4	780.3	**272.8**	977.5	296.6	223.8
	NN-SO	198.5	266.7	276.8	762.0	243.2	885.4	274.6	222.0	441.6	331.0	730.8	318.9	781.4	1980.9	220.0	866.9
	GBNN-MO	243.1	234.4	495.6	**230.8**	703.9	**239.5**	746.6	3078.7	252.2	788.5	**251.1**	698.8	282.3	**852.2**	296.0	**217.2**
	GBNN-SO	**155.4**	**208.8**	**188.9**	617.1	**171.3**	795.7	**223.9**	**164.4**	417.3	**202.6**	398.4	**159.4**	589.9	1326.2	**163.2**	619.0
oes97	NN-MO	2153.9	478.1	1875.5	665.7	624.0	2240.8	1153.2	1338.3	**180.4**	1033.5	**164.0**	568.1	**438.0**	**333.1**	225.0	**261.5**
	NN-SO	2033.9	444.2	156.9	520.1	426.4	321.9	**231.4**	236.5	1852.0	616.9	567.5	2172.5	1125.2	1309.4	167.5	1010.9
	GBNN-MO	2450.1	449.0	2348.7	558.4	618.3	2372.0	943.8	1244.4	489.0	919.5	412.5	**455.7**	438.2	372.7	433.5	440.1
	GBNN-SO	**1087.6**	**215.5**	**152.0**	**266.6**	**325.6**	**243.7**	237.4	**165.1**	1035.0	**408.3**	481.1	1737.8	799.1	1084.9	**143.3**	896.4

Table 2: RMSE for Single-Output (SO) and Multi-Output (MO) approaches of GBNN and NN

subsample values for GBNN are fixed to 1.0 and 0.5, respectively. The results of the training and average prediction times are shown in Table 3.

As it can be observed from Table 3, the average classification time necessary to classify 10^6 instances is very similar in between the MO models and in between the SO models. This makes perfect sense as both algorithms build the same architecture: one single network for the MO models and one network per target for the SO. The only element that changes is the value of the computed weights, which do not have an impact in the evaluation time. More interestingly, when comparing MO with respect to the SO models, it can be observed that the MO models present faster prediction times. This is due to the fact that in the MO models the input-to-hidden layer is shared in contrast to the SO models that have one input (and output) layer per target. In principle, there is not an important time gain in the hidden-to-output layer as the single output layer of MO is #targets more complex than the individual layers of the #targets SO models. Regarding the training times, it can be observed that the training of a single MO network is also more efficient than training one SO network per output for both NN and GBNN algorithms. On the other hand, the differences

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Method	GBNN-MO		GBNN-SO		NN-MO		NN-SO	
Time	fit	pred	fit	pred	fit	pred	fit	pred
rf1	**24.05**	10.78	100.47	66.40	30.94	**10.33**	163.44	61.10
rf2	172.22	30.88	947.78	244.32	**96.91**	**27.83**	640.84	209.79
scm1d	**72.25**	23.89	375.22	363.79	74.66	**22.71**	1066.25	385.14
scm20d	**16.19**	13.47	158.39	193.57	28.84	**9.25**	373.34	113.39

Table 3: Training time and prediction time for 10^6 instances in seconds

in the training times between NN and GBNN are problem dependent and can be explained by the different number of iterations needed to reach convergence.

4 Conclusion

In this paper, a novel sequential approach to train a single multi-output neural network regressor with gradient boosting is presented. The network is trained in phases. At each stage a portion of the full-network is trained to fit the residuals of the targets not learned by the previous trained portions. The proposed training approach is compared with the standard training procedure on ten multiple target regression tasks with favorable results in general for the proposed methodology. Furthermore, we showed experimentally that the computational complexity of the proposed method is equivalent to a NN trained using the standard procedure.

References

[1] Eleftherios Spyromitros-Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis Vlahavas. Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, 104(1):55–98, 2016.

[2] Donna Xu, Yaxin Shi, Ivor W Tsang, Yew-Soon Ong, Chen Gong, and Xiaobo Shen. Survey on multi-output learning. *IEEE transactions on neural networks and learning systems*, 31(7):2409–2429, 2019.

[3] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[4] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 82:128–150, 2017.

[5] Atsushi Nitanda and Taiji Suzuki. Functional gradient boosting based on residual network perception. In *International Conference on Machine Learning*, 2018.

[6] F. Huang, J. Ash, J. Langford, and R. Schapire. Learning deep resnet blocks sequentially using boosting theory. In *International Conference on Machine Learning*, 2018.

[7] Seyedsaman Emami and Gonzalo Martínez-Muñoz. Sequential training of neural networks with gradient boosting. *arXiv preprint arXiv:1909.12098*, 2019.

[8] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[9] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[10] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.