A NOVEL PRUNING APPROACH FOR BAGGING ENSEMBLE REGRESSION BASED ON SPARSE REPRESENTATION

Amir Ehsan Khorashadi-Zadeh[⋆] *Massoud Babaie-Zadeh*[⋆] *Christian Jutten*[†]

[⋆]School of Electrical Engineering, Sharif University of Technology, Tehran, Iran
[†]Grenoble INP, CNRS, GIPSA-lab, University Grenoble Alpes, Grenoble, France
Email: a.e.kh.1374@gmail.com, mbzadeh@yahoo.com, christian.jutten@gipsa-lab.grenoble-inp.fr

ABSTRACT

This work aims to propose an approach for pruning a bagging ensemble regression (BER) model based on sparse representation, which we call sparse representation pruning (SRP). Firstly, a BER model with a specific number of subensembles should be trained. Then, the BER model is pruned by our sparse representation idea. For this type of regression problems, pruning means to remove the subensembles that do not have a significant effect on prediction of the output. The pruning problem is casted as a sparse representation problem, which will be solved by orthogonal matching pursuit (OMP) algorithm. Experiments show that the pruned BER with only 20% of the initial subensembles has a better generalization compared to a complete BER.

Index Terms— Bagging Ensemble Regression, Machine Learning, Sparse Representation.

1. INTRODUCTION

Ensemble regression [1, 2] is a method for generating a prediction based on the aggregation of the predictions of several regressors. An important advantage of ensemble models is that the performance of the model is more consistent than a single regressor because of using aggregation of several regressors [2].

Bagging is one of the most effective [3] methods for aggregation of the subensembles in both regression and classification tasks. For regression problems, this aggregation calculates the average of the ouputs of these subensembles [2]. Each subensemble should be independently trained on a different bootstrap samples of original training data [3]. Bootstrap samples have the same number of samples of the original training data, but are obtained by sampling with replacement from original training data. This method is especially useful for high dimensional dataset problems [3].

One of the important disadvantages of ensemble models is that because of using several subensembles, these models need large memories [4]. In addition, it takes too much time for the prediction of a test data [4] because all subensembles should predict this test data and finally compute their averages [4]. A method to tackle this problem is to prune the ensemble model. For this problem, pruning means to remove the subensembles that do not have significant role in final prediction. Not only pruning reduces the size of ensemble models, but also, it improves model generalizations over test dataset [4].

In [5], an algorithm for pruning a neural network ensemble has been proposed. This algorithm, called GASEN, uses genetic algorithm in order to prune the model. Firstly, it trains all subensembles independently. Then it assigns random weights to subensembles and uses genetic algorithm in order to improve assigned weights. Finally it selects the effective subensembles based on their weights.

Another related work is made in [6]. In this paper a pruning method for bagging ensemble classification is introduced, and is called ordering pruning (OP). This algorithm is also used in [7] for bagging ensemble regression (BER) pruning. This method is a greedy algorithm that selects one subensemble in every iteration such that the selected subensembles have the minimum mean square error (MSE) for the training data. Although this algorithm improves model generalization considerably, this pruning method is rather time consuming as will be seen in our experiments.

In this paper, we propose a novel method for pruning the BER model by using sparse representation [8]. This algorithm uses orthogonal matching pusrsuit (OMP) [9] to solve a sparse representation problem. As it will be seen in our simulations, our algorithm is faster than the method of [7], while its generalization on test data is almost in the same order as [7] (and still better than unpruned model). It should also be pointed out that sparse representation has already been used for neural network pruning [10, 11] and for LS-SVM pruning [12].

The paper is organized as follows. In Section 2, the background of the BER model is reviewed. In Section 3, the problem is formulated as a sparse representation, which will be solved by OMP algorithm. Our method for pruning the BER

This work has been partially supported by Research Office of Sharif University of Technology.

model is proposed in Section 4. Finally, the experimental results of our pruning method and the method of [7] are presented in Section 5.

2. NOTATIONS AND BACKGROUND

In this section, the background of the BER model is very briefly reviewed. Consider the dataset $\{\mathbf{x}^i, y^i\}$ for $i \in \{1, 2, ..., N\}$. The goal of a regression problem is to find a function that can model the relation between the inputs $\{\mathbf{x}^i\}$ and the outputs $\{y^i\}$ for $i \in \{1, 2, ..., N\}$. There are already many regression methods in the literature [13]. Ensemble regression [2] is an approach for combining the results of several regressors (subensembles), where the regression methods of the subensembles are chosen arbitrarily.

In order to evaluate a regression model, the dataset is usually divided into training and test data. Now, consider a regression problem in which we want to train an ensemble regression model that can predict $y^i \in \mathbb{R}$ based on $\mathbf{x}^i \in \mathbb{R}^p$ for $i \in \{1, 2, .., N_{train}\}$. Moreover, suppose that our ensemble model consists of L subensembles, each of which provides its own prediction for y^i. So, every subensemble is independently trained on different bootstrap samples of the training data. After training all subensembles, a method is deployed in order to generate a final output based on the outputs of all subensembles. Bagging method calculates the average of the outputs of all subensembles in order to provide a good estimate for y^i. This model is called bagging ensemble regression (BER).

3. SPARSE REPRESENTATION

In this section, sparse representation [8] is very briefly reviewed. In sparse representation, a signal $\mathbf{y} \in \mathbb{R}^n$ is to be represented as a linear combination of several signals $\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_m$ where $m > n$, and $\forall i, \mathbf{d}_i \in \mathbb{R}^n$. After [14], signals $\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_m$ are called atoms and their collection is called dictionary, which is denoted by $\mathbf{D} \in \mathbb{R}^{n \times m}$. Sparse representation tries to find the minimum number of atoms that can approximately represent $\mathbf{y}$. In other words, sparse representation aims to solve

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_0 \\ \text{s.t.} \quad & \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 < \epsilon, \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^m$ and $\|\mathbf{x}\|_0$ stands for the 'l_0 norm' of $\mathbf{x}$, that is, the number of nonzero entries of $\mathbf{x}$. For the case $\epsilon \neq 0$, the above problem is better to be called 'sparse approximation' rather than 'sparse representation'. Moreover, for $\epsilon \neq 0$, the condition $m > n$ is not necessary, and the problem can be expressed equally also for the case $n \geq m$.

The optimization problem (1) has been solved by OMP in [9]. OMP is a greedy algorithm that in each step, selects an atom from the dictionary $\mathbf{D}$ that is most similar to the error. Although OMP has been proposed in [9] for $m > n$, this algorithm also can be used where $n \geq m$. The algorithm stops when a constraint is fulfilled. The constraint can be either ϵ (as in (1)), or a maximum of selected atoms (sparsity size). So OMP gets $\mathbf{D}$, $\mathbf{y}$ and sparsity size (k) or ϵ as inputs and returns a sparse vector $\mathbf{x}$ as output.

4. SPARSE REPRESENTATION PRUNING ALGORITHM

In this section, our method for pruning the BER model is presented. Consider a BER model that consists of L subensembles. Suppose that all subensembles have been trained on the training data $\{\mathbf{x}^i, y^i\}$, $i \in \{1, 2, ..., N_{train}\}$. So, for one train sample $\mathbf{x}^i \in \mathbb{R}^p$, we have L predictions $\{y_1^i, y_2^i, ..., y_L^i\}$. Now, we put these L values in a vector and call this vector $\mathbf{d}^i \in \mathbb{R}^L$. Then, all these vectors for all training samples are put in a matrix $\mathbf{D} \in \mathbb{R}^{N_{train} \times L}$. So $\mathbf{D}$ contains the predictions of all subensembles on all N_{train} samples. According to this notation in BER, the final prediction $\mathbf{y}^* = (y^{*1}, y^{*2}, ..., y^{*N_{train}})^T$ is obtained from

$$\mathbf{y}^* = \mathbf{D}\mathbf{v}, \tag{2}$$

where $\mathbf{v} = \frac{1}{L}\mathbf{1}$, in which $\mathbf{1}$ is the all-one vector. As it can be observed, the number of subensembles is a parameter that should be determined before training. This paper aims to choose the subensembles that play the most important role in generating the desired outputs.

It is crucial that the BER model has an acceptable performance on the test data (not only on the training data). If it works well just on training data and not on the test data, it means that the final regressor has a weak generalization. Lack of generalization happens especially when the model has too many parameters. So, the BER is pruned to avoid poor generalization.

Our pruning method aims to remove as many as subensembles such that the MSE on the training data does not change too much compared to the unpruned model. Here, we cast this problem as a sparse representation problem. Suppose that the estimated $\mathbf{y}^*$ on complete and selected subensembles are called respectively as $\mathbf{y}_{bp}^*$ and $\mathbf{y}_{ap}^*$ ('bp' and 'ap' stand for 'before pruning' and 'after pruning'). Our method tries to select the subensembles such that $\mathbf{y}_{ap}^*$ and $\mathbf{y}_{bp}^*$ are nearly equal. Mathematically, we minimize the number of nonzero indices in $\mathbf{v}$ in order to $\|\mathbf{y}_{bp}^* - \mathbf{y}_{ap}^*\|_2^2$ is close to zero:

$$\begin{aligned} \min \quad & \|\mathbf{v}^*\|_0 \\ \text{s.t.} \quad & \|\mathbf{y}_{bp}^* - \mathbf{D}\mathbf{v}^*\|_2^2 < \epsilon. \end{aligned} \tag{3}$$

We solve the optimization problem (3) by using OMP. As stated in Section 3, the constraint of OMP algorithm can be expressed either based on ϵ or sparsity size (k). Hereafter, we use sparsity size (k) for the stopping criterion. So we determine the number of atoms (subensembles) that should be

selected by OMP and then OMP tries to find the k most important subensembles.

Note that after solving (3), for each selected subensemble, there will be also an associated weight, meaning that the final prediction would be a weighted sum of the predictions of the selected subensembles. We call this pruning method as weighted sparse representation pruning (Weighted SRP). However, as it will be seen in our simulations, if we forget their weights, and provide the final estimation by just an unweighted averaging of the selected subensembles, the model results in better predictions than weighted model (especially for the test data).

The final unweighted pruning algorithm is shown in Algorithm 1, and is called sparse representation pruning (SRP).

Algorithm 1 SRP method

Input: $\{\mathbf{x}^i, y^i\}_{i=1}^{N_{train}}$, k

Output: $\mathbf{S}$ (the set of indexes of the selected subensembles)

1: Train all subensembles with training data $\{\mathbf{x}^i, y^i\}_{i=1}^{N_{train}}$
2: Put predictions of all subensembles for all training data in $\mathbf{D}$
3: $\mathbf{y}_{bp}^* = \mathbf{D}\mathbf{v}$ where $\mathbf{v} = \frac{1}{L}\mathbf{1}$, in which $\mathbf{1}$ is the all-one vector.
4: $\mathbf{v}^* = \mathrm{OMP}(\mathbf{D}, \mathbf{y}_{bp}^*, k)$
5: $\mathbf{S} = \{i | v_i^* \neq 0\}$

As it is said before, by pruning the BER model, it is expected that the model generalization is enhanced and MSE on test data will be reduced. It is also obvious that by pruning the BER model, the model performance on test data will become faster than the unpruned model.

5. EXPERIMENTAL RESULTS

In order to assess the performance of SRP, some experiments have been done over 5 popular datasets, all from [15]. These datasets have been selected from real world problems in order to evaluate SRP in a challenging context. Table 1 demonstrates the number of samples and attributes for each dataset. In order to report more reliable information, 10-fold cross-validation is repeated 5 times randomly and the average of all MSE's is reported. In addition, we deploy decision tree regressor [16] as our subensembles. All trees have depth 2 and have been independently trained on different bootstrap samples of the training data. The BER model was trained with Scikit-learn Python package [17]. All the computations have been implemented on Python programming language. For implementing SRP, we used OMP that also exists in Scikit-learn Python package [17].

Now we describe our experiments in more details. At first, for each dataset, we repeat for 5 times a 10-fold cross-validation and compute the average of their MSE's in order to report a good approximation of the error. Each 10-fold cross-validation is selected from the dataset randomly. To sum up, the steps that are done for experiments are as follows:

Table 1. Properties of the Datasets used in Experiments

Dataset	Instances	Attributes
Diabetes	442	10
Fetch California Housing	20640	8
Boston	506	13
Airfoil self noise	1503	5
Abalone	4417	9

1. Shuffle the dataset,
2. Generate 10 partitions for 10-fold cross-validation,
3. For each division, train L subensembles independently on different bootstrap of the training data,
4. Use our SRP pruning method in order to select the k subensembles that produce smaller errors,
5. For each division, compute the test error on the pruned BER model,
6. Compute the average of these 10 test errors,
7. Repeat these steps 5 times and finally compute the average of these 5 test errors and report as final MSE.

In addition, the experiments are done for $L = 100, 200$ in order to analyze the effect of initial number of subensembles on SRP.

Now, we compare SRP with ordering pruning (OP) [7] algorithm, in both testing error and pruning speed. In our experiments, the number of selected subensembles (pruning size) is set to 20% of the initial subensembles (although different pruning sizes can be used in both SRP and OP algorithms).

Table 2 shows train MSE of the BER model in three cases: before pruning, BER pruned by SRP, BER pruned by OP, and test MSE in four cases: before pruning, BER pruned by Weighted SRP, BER pruned by SRP, BER pruned by OP, all with $L = 100$ initial subensembles on different datasets. The first column specifies the datasets that have been used. Three next columns contain the MSE of the training data over three models. The next four columns contain the MSE of the test data over four models. Note that for each dataset, the bold value indicates the best test MSE, and the underlined value shows the second best test MSE. Table 3 has the same information of Table 2 but for $L = 200$ initial subensembles.

As it can be seen from Tables 2 and 3, both SRP and OP algorithms improve MSE on training data. It is because, these algorithms remove the subensembles that do not have a significant role in reducing error. However, the main performance criterion is MSE on test data, not training data. It is clear from Tables 2 and 3 that both SRP and OP algorithms reduce the

Table 2. Average train and test MSE for before pruning, SRP and OP [7] with $L = 100$ initial subensembles. Bold and underlined values show, respectively, the best and the second best MSE's on the test data.

	Train			Test			
Dataset	**Before pruning**	**SRP**	**OP [7]**	**Before pruning**	**Weighted SRP**	**SRP**	**OP [7]**
Diabetes	2971.92	2950.14	2843.53	3409.18	3407.96	**3364.59**	<u>3366.14</u>
Fetch California Housing	0.7137	0.7089	0.7012	0.7195	0.7196	<u>0.7141</u>	**0.7070**
Boston	19.55	18.49	17.25	22.92	23.01	<u>21.87</u>	**21.02**
Airfoil self noise	26.00	25.41	24.89	27.15	28.94	<u>26.43</u>	**25.98**
Abalone	6.20	6.17	6.12	6.33	6.34	<u>6.28</u>	**6.24**

Table 3. Average train and test MSE for before pruning, SRP and OP [7] with $L = 200$ initial subensembles. Bold and underlined values show, respectively, the best and the second best MSE's on the test data.

	Train			Test			
Dataset	**Before pruning**	**SRP**	**OP [7]**	**Before pruning**	**Weighted SRP**	**SRP**	**OP [7]**
Diabetes	2966.11	2947.33	2827.70	3421.54	3425.36	**3378.04**	<u>3388.50</u>
Fetch California Housing	0.7137	0.7093	0.7009	0.7189	0.7189	<u>0.7141</u>	**0.7062**
Boston	19.55	18.49	17.09	22.91	23.00	<u>21.67</u>	**20.87**
Airfoil self noise	25.97	25.35	24.80	27.02	28.33	<u>26.20</u>	**25.81**
Abalone	6.20	6.17	6.11	6.32	6.32	<u>6.28</u>	**6.23**

Table 4. Pruning time (in seconds) for SRP and OP for $L = 200$ initial subensembles

Dataset	**SRP**	**OP [7]**
Diabetes	0.0045	0.3465
Fetch California Housing	0.0496	0.3659
Boston	0.0040	0.3307
Airfoil self noise	0.0078	0.3528
Abalone	0.0131	0.3270

MSE on test data as well. This means these two algorithms improve the generalization of the BER model. Although in some datasets, the test data MSE of the SRP is better than OP, the OP algorithm works better in some other datasets. As well, as it can be seen, test MSE on Weighted SRP is more than SRP. So SRP is much more beneficial than Weighted SRP. To sum up, both these SRP and OP algorithms have near performances in MSE sense.

Table 4 indicates pruning time for both SRP and OP algorithms. For this experiment, the initial number of subensembles is set to $L = 200$. It is clear that, the SRP exhibits extremely better performance than OP in speed sense. The efficiency of SRP comes from using OMP algorithm for pruning the BER model. Our simulations were performed in Python 3.7.3 environment on a system with 3.6 GHz CPU, and 16 GB RAM, under Microsoft Windows 10 64-bit operating system.

6. CONCLUSION

In this paper, a novel idea for pruning the BER model by using sparse representation is proposed. By increasing the number of subensembles in a BER model, it needs a large memory to predict a test sample. In addition, prediction procedure will become time consuming. Model generalization also decreases by increasing the number of subensembles. In order to tackle these problems in this paper, the SRP method was used in order to select the subensembles that have the most significant role in correct prediction. Our SRP method uses OMP in order to solve a sparse representation problem. The experiments showed that SRP improves model generalization for the test data. In addition, because of using OMP, SRP prunes the BER model faster than OP.

7. REFERENCES

[1] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, Geoffrey E Hinton, et al., "Adaptive mixtures of local experts.," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[2] Leo Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[3] Peter Bühlmann, Bin Yu, et al., "Analyzing bagging," *The Annals of Statistics*, vol. 30, no. 4, pp. 927–961, 2002.

[4] Gonzalo Martínez-Muñoz and Alberto Suárez, "Pruning in ordered bagging ensembles," in *Proceedings of*

the 23rd international conference on Machine learning. ACM, 2006, pp. 609–616.

[5] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang, "Ensembling neural networks: many could be better than all," *Artificial intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[6] Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez, "An analysis of ensemble pruning techniques based on ordered aggregation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 245–259, 2008.

[7] Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz, and Alberto Suárez, "Pruning in ordered regression bagging ensembles," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006, pp. 1266–1273.

[8] Michael Elad, *Sparse and redundant representations: from theory to applications in signal and image processing*, Springer Science & Business Media, 2010.

[9] Joel A Tropp and Anna C Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[10] Jie Yang, Abdesselam Bouzerdoum, and Son Lam Phung, "A neural network pruning approach based on compressive sampling," in *2009 International Joint Conference on Neural Networks*. IEEE, 2009, pp. 3428–3435.

[11] Jie Yang and Jun Ma, "Feed-forward neural network training using sparse representation," *Expert Systems with Applications*, vol. 116, pp. 255–264, 2019.

[12] Jie Yang, Abdesselam Bouzerdoum, and Son Lam Phung, "A training algorithm for sparse ls-svm using compressive sampling," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, pp. 2054–2057.

[13] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[14] Stéphane G Mallat and Zhifeng Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[15] Catherine L Blake and Christopher J Merz, "Uci repository of machine learning databases, 1998," 1998.

[16] Leo Breiman, *Classification and regression trees*, Routledge, 2017.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.