ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Ensemble Modeling with a Constrained Linear System of Leave-One-Out Outputs

Yoan Miche[1,2], Emil Eirola[1], Patrick Bas[2], Olli Simula[1], Christian Jutten[2], Amaury Lendasse[1] and Michel Verleysen[3]

1 – Helsinki University of Technology – Dept. of Information and Computer Science
Konemiehentie 2, 02015 TKK – Finland

2 – Institut National Polytechnique de Grenoble – Gipsa-Lab
961 rue de la Houille Blanche, BP46, 38402 Grenoble – France

3 – Université Catholique de Louvain – Machine Learning Group
Place du Levant 3, B-1348-Louvain-la-Neuve, Belgium

Abstract. This paper proposes a method for ensemble models using their Leave-One-Out output and solving a constrained linear system. By the use of the proposed method to create an ensemble of Locally Linear models, results on six different regression data sets are comparable to state-of-the-art methods such as Least-Squares Support Vector Machines and Gaussian Processes, while being orders of magnitude faster.

1 Introduction

One of the typical machine learning paradigms is about finding the model that best fits the given data, in terms of test or validation. Searching for such a model can be very time consuming: finding the model class that best suits the type of data, optimizing the possible hyper-parameters, and finally training the model once all details of the model structure have been selected. This most likely leads to a rather good model, which properly fits the data and avoids the pitfalls of overfitting. Meanwhile, a lot of computation time is consumed, which could be used for building less efficient but much more numerous simpler models. The Boosting idea [1] is using this approach: a "weak" model is built, and the data leading to the largest errors are identified; another model of the same class is then trained with an emphasis on these data, and so on. Hence, the models iteratively overfit some part of the data which is not properly modeled by the previous models. The addressed challenge here – the Ensemble technique – is rather different. All models can be built in parallel. The goal is then to weight each model so that the overall output of a linear combination of models has the best possible error. Several ensemble techniques have been proposed, out of which two kinds can be distinguished [2]: the variable weights approach and the "average" ones. Traditionally, average weights ensemble techniques are used and simply take an average of all the built models. While this obviously has the advantage of having immediately the weights of all models, it yields suboptimal results. The variable weights ensemble techniques try to optimize the weight of each model in the ensemble according to a criterion. Techniques such as the Genetic Algorithm have been recently used for such optimization [3] but are very time consuming. This paper proposes the use of a Leave-One-Out (LOO) output

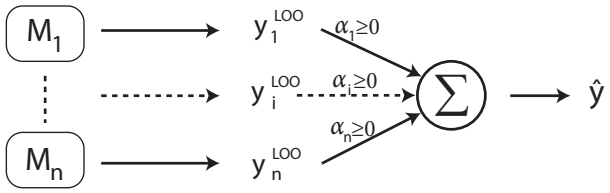

Fig. 1: Illustrative scheme of the Ensemble of models using LOO outputs.

for each model and a Non-Negative constrained Least-Squares problem solving algorithm, leading to an efficient solution coupled with a short computation time. Section 2 details this methodology, with Section 2.1 giving a proof on the applicability of the methodology under some hypotheses. Experiments on six regression data sets and details about the class of models used are given in Section 3, along with a comparison of performances to state-of-the-art methods.

2 Method

The global aim is to find the optimal weights α_i for a given set of models M_i (each providing the prediction $\hat{\boldsymbol{y}}^{(i)} = (y_1^{(i)}, \ldots, y_N^{(i)})$, with N the number of samples) to form an ensemble as a linear combination:

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{n} \alpha_i \hat{\boldsymbol{y}}^{(i)}. \tag{1}$$

In order to not exaggerate the errors of each model in the ensemble, the α_i should usually be set to non-negative. Assuming each model is unbiased, the ensemble model can be made unbiased by having $\sum \alpha_i = 1$. The naive choice is to take a simple average and specify $\alpha_i = \frac{1}{n}$, but it is possible to obtain significantly better performance by optimizing the weights. If the models are known to be entirely independent, a closed-form expression for the optimal weights can be derived as in Section 2.1. On the other hand, if the models contain dependencies which are not easy to quantify, as is often the case, another choice is to solve the weights from a constrained linear system. This scenario is studied in Section 2.2. The overall idea is depicted in Fig. 1.

2.1 Theoretical Study of Independent Models

Assuming there are n independent models for $\boldsymbol{y}$ – each of the form $\boldsymbol{y}^{(i)} = \boldsymbol{y} + \varepsilon_i$, where the additive noise ε_i has zero mean and variance σ_i^2 – it is possible to directly derive the optimal weights α_i. In the following, Y and $Y^{(i)} = Y + \varepsilon_i$ are to be considered as random variables ($\boldsymbol{y}^{(i)}$ is a vector of realisations of $Y^{(i)}$, etc.). The aim is to minimise the (expected) MSE $\mathrm{E}\left[\left(Y - \sum_i \alpha_i Y^{(i)}\right)^2\right]$, which

can be separated due to the independence of the noise:

$$\mathrm{E}\left[\left(Y - \sum\nolimits_i \alpha_i Y^{(i)}\right)^2\right] = \mathrm{E}\left[\left(Y - \sum\nolimits_i \alpha_i Y - \sum\nolimits_i \alpha_i \epsilon_i\right)^2\right]$$
$$= \mathrm{E}\left[\left(Y - \sum\nolimits_i \alpha_i y\right)^2\right] + \mathrm{E}\left[\left(\sum\nolimits_i \alpha_i \epsilon_i\right)^2\right] = \left(1 - \sum\nolimits_i \alpha_i\right)^2 s^2 + \sum\nolimits_i \alpha_i^2 \sigma_i^2,$$

where $s^2 = \mathrm{E}\left[Y^2\right]$. Differentiating w.r.t. α_k, and setting to zero results in:

$$\frac{d}{d\alpha_k}\left[\left(1 - \sum\nolimits_i \alpha_i\right)^2 s^2 + \sum\nolimits_i \alpha_i^2 \sigma_i^2\right] = -2\left(1 - \sum\nolimits_i \alpha_i\right) s^2 + 2\alpha_k \sigma_k^2 = 0.$$

This leads to the equation

$$\alpha_k \sigma_k^2 = \left(1 - \sum\nolimits_i \alpha_i\right) s^2. \tag{2}$$

Here the right side (call it c), while still dependent on the parameter α_k, is independent of k. Hence it holds that $\alpha_k \propto \sigma_k^{-2}$, with the proportionality coefficient c. Substituting $\alpha_i = c\sigma_i^{-2}$ into Eq. 2, we can solve for c:

$$c = s^2 - \sum\nolimits_i c\sigma_i^{-2} s^2 \quad \Longrightarrow \quad c = \frac{1}{s^{-2} + \sum_i \sigma_i^{-2}}.$$

Finally, the optimal weights are

$$\alpha_k = \frac{\sigma_k^{-2}}{s^{-2} + \sum_i \sigma_i^{-2}}\,, \tag{3}$$

and the resulting MSE with these weights becomes $1/(s^{-2} + \sum_i \sigma_i^{-2})$. This is lower than $\min_k \sigma_k^2$, meaning that the ensemble is more performant than any single constituent model. The error is also lower than the error achieved by the naïve average weighting $\alpha_k = \frac{1}{n}$, which is $\frac{1}{n^2}\sum_i \sigma_i^2$.

There is a trade-off between bias and variance here: minimising the variance introduces a slight bias to the ensemble model. This could be rectified by restricting the weights to $\sum_i \alpha_i = 1$, in which case the optimal weights are

$$\alpha_k = \frac{\sigma_k^{-2}}{\sum_i \sigma_i^{-2}}\,. \tag{4}$$

This can be shown using Lagrange multipliers. The resulting MSE is then slightly larger: $1/\sum_i \sigma_i^{-2}$, however, the difference is practically insignificant. Indeed, if the collections of models includes even a few reasonably accurate ones – that is, $\exists k$, s.t., $\sigma_k \ll s$ – the term s^{-2} is insignificant compared to the sum in the denominator in Eq. 3 for α_k, and the weights (and resulting MSEs) calculated by Eq. 3 or 4 are essentially equivalent.

Some observations concerning Eq. 3 and 4 can be made. First, the weight of a model is inversely proportional to the variance of the error of that model, resulting in good models having large weights, and, correspondingly, poor models having low weights. Second, all the weights are strictly positive. As the assumptions specify that the models are independent, even the poor models can still contribute to the predictive power to some extent.

2.2 Practice

In the previous section, it was assumed that all the models are independent. In practice, this scenario is unlikely. If the particular dependencies are unknown, it is difficult to derive an exact expression for the optimal weights of the ensemble, but in any reasonable case there certainly *exists* a set of weights such that the resulting MSE is notably lower than that of any single model. The alternative is to solve the weights from the linear system in Eq. 5 below, as in this one the dependencies between models can be accounted for. Solving the system is a very aggressive method of fitting and runs the risk of overfitting. To counteract this, the LOO output of the models is used, and the α_i are restricted to be non-negative. Solving the constrained linear system naturally results in higher weights for more accurate models, and low weights for poor models. This inverse relationship between the weight and MSE is in correspondance with Eq. 3. This method is based on the LOO [4] output $\boldsymbol{y}_{\mathrm{LOO}}^{(i)}$ of each of the models M_i, with the coefficients determined using a Non-Negative constrained Least-Squares (NNLS) algorithm. The classical NNLS algorithm in [5] is used to compute this solution. For each model $M_i, 1 \leq i \leq n$, the LOO output $\boldsymbol{y}_{\mathrm{LOO}}^{(i)}$ is computed by omitting the considered point from the training and evaluating the model on that single point. Hence, a set of $\left\{\boldsymbol{y}_{\mathrm{LOO}}^{(i)}\right\}_{1\leq i\leq n}$ outputs are computed, one for each model. The coefficients α_i are then solved from the constrained linear system:

$$\arg\min_{\boldsymbol{\alpha}} \left\| \boldsymbol{y} - \sum_{i=1}^{n} \alpha_i \boldsymbol{y}_{\mathrm{LOO}}^{(i)} \right\|^2 \quad \text{s.t.} \quad \alpha_i \geq 0\,. \tag{5}$$

As mentioned earlier, one advantage of this method is its low computational time, in terms of finding an optimal linear combination. The NNLS algorithm has been widely used and is known to converge in $\frac{1}{2}n$ steps, as noticed in [5]. In the idea of keeping the computational time low, for the whole method to be fast, the class of models used should be such that the LOO output is rapidly computed or eventually approximated. As the Locally Linear models used in the experimental section are not independent, the weights are determined by solving Eq. 5 instead of using Eq. 3.

3 Experiments

3.1 Locally Linear models (LL)

The class of models used in the experiments is known as Locally Linear models [6, 7] (LL). The idea is to perform a linear regression for each sample of the data set, based on its k-nearest neighbors. Each sample's output can then be estimated using the linear regression model. To be able to fit a linear model, k must necessarily be at least the number of dimensions d plus one (this lower bound $k_{\min}$ has to be checked for possible numerical problems might arise). In the experiments, the lower bound $k_{\min}$ has been set to $d + 2$ and the upper bound to $k_{\max} = d + 11$ and all the values in the range have been tested. In this setup, all the possible subsets of variables are investigated and used for the

Table 1: Regression data sets: number of variables and number of samples.

	Abalone	Ailerons	Elevators	CPU	Servo	Bank
Number of Variables	8	5	6	6	4	8
Samples	4177	7129	9517	209	167	4499

Table 2: Computational times (in seconds) on regression data sets.

	Abalone	Ailerons	Elevators	CPU	Servo	Bank
LS-SVM	6.6e+4	4.2e+2	5.8e+2	3.2e+2	1.3e+2	1.6e+3
MLP	2.1e+3	3.5e+3	3.5e+3	5.8e+2	5.2e+2	2.7e+3
GP	9.5e+2	2.9e+3	6.5e+3	3.2	2.2	1.7e+3
E-LL	1.8e+2	4.1e+1	1.0e+2	2.0	4.4e-1	2.0e+2

linear combination of models: there are then $(k_{\max} - k_{\min}) \times (2^d - 1)$ models to combine, using their LOO output and the NNLS algorithm. Irrelevant variables subsets yield suboptimal models and LOO outputs and are therefore not taken in the linear combination. Since all the $2^d - 1$ subsets of variables are examined, and while the LL models and their LOO outputs are fast to compute, it remains necessary to use data sets with a number of variables sufficiently low (no more than 10 in the following experiments). Variable selection or wrapper techniques could be used in order to extend this limit while keeping a reasonable computational time.

3.2 Experiments setup and results

The methodology is applied to six regression data sets from the UCI database [8]. All data sets are processed in the same way: 10 random permutations of the whole set are taken (without repetitions), and each is first divided in training and test sets (two-thirds for training and one third for testing), and then normalized (zero mean and unit standard deviation) both using the mean and standard deviation from the training set. Table 1 gives details about the data sets. In order to make a fair comparison of the proposed method to state-of-the-art techniques, Least-Squares Support Vector Machines [9] (LS-SVM), Multi-Layer Perceptron (MLP) (using the Matlab Neural Network toolbox with Levenberg-Marquardt backpropagation) and Gaussian Processes [10] (GP) are also used on the same data sets. Each method is used on the 10 permutations of the seven data sets, in order to obtain reliable performance and a standard deviation on the results. Computational times are the result of averaging the 10 computational times obtained and are given in Table 2. From Table 3, the proposed methodology performs roughly as well as the best model in each case, and with a smaller standard deviation of the results in most cases. It can be noted that in most cases solving the linear system results in a relatively low number of non-zero coefficients α_i (number of selected models) as shown in Table 3.

Table 3: Normalized MSE Test results in boldface (standard deviations in regular) and number of selected models (in parenthesis) for Ensemble of LL.

	Abalone	Ailerons	Elevators	CPU	Servo	Bank
LS-SVM	**0.44**	**1.43**	**1.09**	**0.31**	**6.9e-1**	**1.18**
	2.6e-2	2.9e-1	1.2e-1	2.4e-1	3.3e-1	3.5e-2
MLP	**0.45**	**2.97**	**0.46**	**0.66**	**2.2e-1**	**0.04**
	5.6e-2	4.8e-2	1.6e-2	8.5e-1	8.1e-2	1.8e-3
GP	**0.44**	**0.30**	**0.35**	**0.31**	**4.8e-1**	**0.04**
	2.3e-2	2.1e-2	8.8e-3	3.1e-1	3.5e-1	2.2e-3
E-LL	**0.46**	**0.30**	**0.39**	**0.14**	**5.6e-1**	**0.04**
	2.5e-2	1.5e-2	1.2e-2	9.4e-2	3.0e-1	1.7e-3
Select.	(23)	(15)	(20)	(12)	(4)	(22)

4 Conclusion

This paper presents a method to build ensembles of models using the LOO outputs, combined using a non-negativity constrained linear system solved in the least-squares sense. It is shown, that under some assumptions on the models, there is always a benefit from the ensemble of models. The application of this method to Locally Linear models proves to yield results comparable to the state-of-the-art models, while keeping a low computational time and a low number of selected models for the ensemble. In future work, this method shall be extended to other classes of models (for which the LOO output is still rapidly computed or approximated). Also, the construction of all the models could be done in parallel, while the proposed results in this paper are for sequentially built models. Computational time could then be highly reduced while keeping the whole methodology identical. Also, variable selection or wrapper techniques can be used to extend the current dimensional limit on the data.

References

[1] R. E. Schapire. Theoretical views of boosting and applications. In *Proceedings of the 10th Int. Conf. on Algorithmic Learning Theory*, pages 13–25, Tokyo, Japan, Dec. 1999.

[2] T. G. Dietterich. *Handbook of brain theory and neural networks.* Cambridge MA: MIT Press, 2nd edition, 2002. Chapter: Articles: Ensemble Learning.

[3] Z. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artif. Intell.*, 137(1-2):239–263, 2002.

[4] A. Lendasse, V. Wertz, and M. Verleysen. Model selection with cross-validations and bootstraps - application to time series prediction with RBFN models. In *ICANN 2003, Istanbul (Turkey)*, volume 2714 of *LNCS*, pages 573–580. Springer, June 26-29 2003.

[5] C. L. Lawson and R. J. Hanson. *Solving least squares problems.* SIAM Classics in Applied Mathematics, 3rd edition, 1995.

[6] M. Birattari, G. Bontempi, and H. Bersini. Local learning for data analysis. In *Proceedings of the 8th Belgian-Dutch Conf. on Machine Learning*, Wageningen, Netherlands, 1998.

[7] C. Atkeson et al. Locally weighted learning. *AI Review*, 11:11–73, 1997.

[8] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[9] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vanderwalle. *Least-Squares Support-Vector Machines.* World Scientific, Singapore, 2002.

[10] C. E. Rasmussen et al. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.