Extending Extreme Learning Machine with Combination Layer

Dušan Sovilj[1], Amaury Lendasse[1,2,3], and Olli Simula[1]

[1] Aalto University School of Science, Espoo, Finland
[2] IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
[3] University of The Basque Country, Donostia-San Sebastián, Spain
{dusan.sovilj, amaury.lendasse, olli.simula}@aalto.fi

Abstract. We consider the Extreme Learning Machine model for accurate regression estimation and the related problem of selecting the appropriate number of neurons for the model. Selection strategies that choose "the best" model from a set of candidate network structures neglect the issues of model selection uncertainty. To alleviate the problem, we propose to remove this selection phase with a combination layer that takes into account all considered models. The proposed method in this paper is the Extreme Learning Machine(Jackknife Model Averaging), where Jackknife Model Averaging is a combination method based on leave-one-out residuals of linear models. The combination approach is shown to have better predictive performance on several real-world data sets.

1 Introduction

Accurate predictions of future instances are becoming a recurring problem in scientific research. The problem is addressed by forming a model and then making all subsequent inferences on that constructed model. Prediction of continuous values, such as daily temperature or stock market prices, is considered a *regression* problem or estimation of a regression function.

In the paper, we are concerned with regression problems of the form

$$y_i = f(\mathbf{x}_i) + \epsilon_i \tag{1}$$

where $\{(\mathbf{x}_i, y_i) \,|\, 1 \leq i \leq N\}$ are data samples with $\mathbf{x}_i$ consisting of several explanatory features or variables and y_i the target variable, while ϵ_i is the noise term. Usually, the noise is assumed to be homoskedastic with a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ with known variance. The problem is finding a model $\hat{f}$ that can best approximate the target function f. This is a general setting, and many such models exist, including, but not limited to, linear regression, neural networks, support vector regression, kernel regression, nearest neighbour estimators and fuzzy regression. Each model class is characterized by the approximation capabilities and the training algorithms with different computational complexities.

This paper focuses on a specific type of neural network that is gaining popularity in recent years, namely the Extreme Learning Machine (ELM). It is

shown in [1] that a single feedforward hidden layer with input weights randomly assigned has universal approximation capability for any target function, that is, the estimation can be made as small as possible considering standard squared error loss function. The advantage of ELM is its very fast training time, since the output weights are found in a simple linear setting between the hidden feature space and the target variable.

One drawback concerning the ELM model is the selection of the starting number of neurons to adequately capture the overall variations in data. To solve this issue, two approaches have been proposed: 1) *selection* methods [2–6] focus on choosing a single model from a candidate set by optimizing some criterion; 2) *ensemble of many ELMs* [7], i.e., the candidate models *all* contribute to the weighted average as the final model. The second strategy tries to avoid the problem by considering much larger candidate set, and focusing on finding appropriate model weights, aiming to assign zero weights for poor models and non-zero weights for better models.

We propose a mixture between the two mentioned strategies. That is, when constructing a *single* ELM that considers several competing models, instead of picking only one model, use all models and find appropriate weights to separate the models based on their *generalization ability*. Two reasons for such an approach are: 1) empirical success of combination methods over the selection ones when it comes to prediction accuracy [8]; 2) philosophical issue of ignoring model selection uncertainty by making inference solely on a single model once it is identified [9]. Combination methods have been proposed both in Bayesian statistics, where Bayesian Model Averaging [10] is considered a natural approach to model selection uncertainty by considering models as another nuisance parameter, and in a frequentist spirit with weights computed based on bootstrapping or perturbation of data [11]. The proposed idea is to *remove* the procedure "selection of the best model" and consider the ELM as a set of models altogether.

The paper is organized as follows. Section 2 describes the proposed method alongside its main parts: Extreme Learning Machine, Jackknife Model Averaging and Leave-one-out Cross-validation. Section 3 shows one variant of the ELM which can be extended to include the proposed strategy. In Section 4, we show results on several UCI Machine Learning Repository data sets. Conclusions are summarized in Section 5.

2 Combining Extreme Learning Machine(s)

Two approaches have been adopted for selection strategy: *pruning* − starting from a large number of neurons and then removing unnecessary ones [5, 6] and *constructive* − building up from smaller pool of neurons until some condition (usually error) does not improve with the additional complexity [2–4]. Both approaches consider several alternative network structures and finally output "the best" model. The idea is to consider them all together and form a weighted average.

The proposed method Extreme Learning Machine-*combination*, denoted by ELM(c), is an ELM model where the selection phase is replaced with *a combination layer*, i.e., additional layer of hierarchy. The parentheses denote that combination is taking place, while c denotes the method used to produce model weights. The following subsections present the building blocks which constitute the ELM(Jackknife Model Averaging), or ELM(JMA) for short.

2.1 Extreme Learning Machine Overview

Extreme Learning Machine (ELM) network presents a new way of building a neural structure. The idea is in random initialization of input weights and biases for a single hidden layer, which leads to removal of any kind of iterative training algorithm. As shown in [1], with this randomization and under certain constraints on transfer functions, the output weights of a hidden layer can be computed with simple linear regression and the model has universal approximation capabilities.

Consider a data set $\{(\mathbf{x}_i, y_i) \,|\, 1 \leq i \leq N\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. The ELM network with M neurons is constructed by first computing the hidden matrix $\mathbf{H}$

$$\mathbf{H} = \begin{bmatrix} g_1(\mathbf{w}_1^{\mathrm{i}}\mathbf{x}_1 + b_1) & \cdots & g_M(\mathbf{w}_M^{\mathrm{i}}\mathbf{x}_1 + b_M) \\ \vdots & \ddots & \vdots \\ g_1(\mathbf{w}_1^{\mathrm{i}}\mathbf{x}_N + b_1) & \cdots & g_M(\mathbf{w}_M^{\mathrm{i}}\mathbf{x}_N + b_M) \end{bmatrix}$$

with j-th neuron having activation function g_j, input weights $\mathbf{w}_j^{\mathrm{i}}$, bias b_j and both $\mathbf{w}_j^{\mathrm{i}}$ and b_j are randomly generated. Hidden layer output weights $\boldsymbol{\beta}$ are found by solving the linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{y}$, with the Moore-Penrose generalized inverse of the matrix $\mathbf{H}$ and the target values – $\boldsymbol{\beta} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y}$. The matrix $\mathbf{H}$ is sometimes called feature mapping or feature space of the ELM. Any function that is a bounded non-constant piecewise continuous function can be taken as activation function in the ELM.

2.2 Leave-one-out Cross-validation

Cross-validation (CV) is one of the most used strategies for evaluating regression models, and provides immediate comparison between a wide range of different model classes. k-fold CV splits the data set into k parts, and each part plays the validation role once the model is trained on the remaining $k-1$ parts. The average error of all k parts is taken as a measure of generalization ability of the model. For selection strategy, the model with the smallest average validation error is assumed to be the most suitable for the given data set.

The extreme case is $k = N$ or leave-one-out (LOO) CV, where each data sample is taken to be a sole sample in the validation set. In the case of least squares linear regression, the LOO error can be computed with a single fit of the model using the PRESS statistic [12], which removes the computational burden of training N separate models. If we denote with $\mathbf{P} = \mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}$, then the leave-one-out residuals for all samples are computed with the PRESS formula

$$\tilde{\mathbf{e}}^{\mathrm{LOO}} = \frac{\mathbf{y} - \mathbf{P}\mathbf{y}}{\mathbf{1} - \mathrm{diag}(\mathbf{P})}. \tag{2}$$

where $\mathrm{diag}(\mathbf{P})$ denotes the main diagonal of $\mathbf{P}$, $\mathbf{1}$ a vector of ones and division is performed element by element.

2.3 Jackknife Model Averaging

Jackknife model averaging (JMA) is a model combining method minimizing the LOO-CV error which has recently been proposed in [13]. The authors show that considering the LOO residuals of all models, the combination asymptotically achieves the lowest possible expected squared error out of any model considered. That is, the combination is conditioned on the candidate set, and can only be better than those in the set itself. The theory in [13] is restricted to linear models and random samples, but it allows heteroskedastic noise term ϵ_i in Eq. (1) and *unbounded* number of models to be present in the candidate set.

JMA is a linear combination of leave-one-out residuals off all models, and for the purpose of presentation we use similar notation as in [13]. Denote with $\tilde{\mathbf{e}}^m = [\tilde{e}_1^m, \ldots, \tilde{e}_N^m]^{\mathrm{T}}$ the leave-one-out residual vector of the m-th model in the candidate set. Then, the jackknife averaging residual vector is

$$\tilde{\mathbf{e}}(\mathbf{w}) = \sum_{m=1}^{M} w^m \tilde{\mathbf{e}}^m = \tilde{\mathbf{e}}\mathbf{w}$$

where $\tilde{\mathbf{e}} = [\tilde{\mathbf{e}}^1, \ldots, \tilde{\mathbf{e}}^M]$ and jackknife estimate of the generalization error is given by

$$\mathrm{CV}(\mathbf{w}) = \frac{1}{N}\tilde{\mathbf{e}}(\mathbf{w})^{\mathrm{T}}\tilde{\mathbf{e}}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{S}\mathbf{w} \tag{3}$$

where $\mathbf{S} = \tilde{\mathbf{e}}^{\mathrm{T}}\tilde{\mathbf{e}}/N$ is an $M \times M$ matrix. The choice of $\mathbf{w}$ is the one that minimizes the cross-validation criterion defined in Eq. (3) with the weights constrained on a unit simplex $\mathcal{H} = \{\mathbf{w} \in \mathbb{R}^M | w^m \geq 0, \sum_{m=1}^{M} w^m = 1\}$. This is a quadratic programming problem with respect to $\mathbf{w}$ and can easily be solved with publicly available software packages. All that is needed to solve the minimization problem are the LOO residuals of every model in the set.

2.4 ELM(JMA)

The ELM(JMA) model is constructed as follows. Start with a fixed number of neurons, say M, and compute the matrix $\mathbf{H}$. Then, train M models, where the output weights for each model are computed with $\boldsymbol{\beta}^m = (\mathbf{H}_m^{\mathrm{T}}\mathbf{H}_m)^{-1}\mathbf{H}_m^{\mathrm{T}}\mathbf{y}$, $1 \leq m \leq M$, and $\mathbf{H}_m$ consists of the first m columns of $\mathbf{H}$. That is, keep increasing the number of neurons/columns by 1 (starting from 1 until M) and compute the new model. During this training phase, gather LOO residual vectors $\tilde{\mathbf{e}}^m$ for each model using Eq. (2). The final step is JMA combining of all M

models using $\tilde{\mathbf{e}}$ in the minimization problem defined by Eq. (3) which gives model weights $\mathbf{w} = [w^1, \ldots, w^M]^{\mathrm{T}}$. The function estimate of the target variable $\mathbf{y}$ is then $\sum_{m=1}^{M} w^m \mathbf{H}_m \boldsymbol{\beta}^m$. Prediction for a fresh sample $\mathbf{x}_n$ is straightforward: compute the output $\hat{f}^{i_l}(\mathbf{x}_n)$ for those models whose weights are greater than zero $\mathcal{M} = \{w^m > 0 | 1 \leq m \leq M\} = \{i_1, \ldots, i_L\}$ and output the weighted average of those models $\hat{f}(\mathbf{x}_n) = \sum_{l=1}^{L} w^{i_l} \hat{f}^{i_l}(\mathbf{x}_n)$.

Reason for considering only M models is that there is an exponential number of possible models, i.e., all subsets selection problem with M variables. The other issue is the ordering of neurons with different activation functions. In our approach, we randomly permute the order to prevent only one type of function from dominating the network structure and to allow more variability.

It should be noted that a linear combination of linear models can be seen as *one* linear model by summation of appropriate weight vectors. The proposed method then returns a single model, but the model where selection uncertainty has been accounted for. In this view, ELM(c) introduces another form of regularization on the weights $\boldsymbol{\beta}$.

3 TROP-ELM

Tikhonov Regularized Optimally Pruned Extreme Learning Machine (TROP-ELM) [6] brings two adjustments to the original ELM. First phase is ranking of neurons by LARS and selecting the appropriate number of neurons by minimizing the LOO error (OP part). The other improvement is L_2 regularization on the weights $\boldsymbol{\beta}$, by introducing a slight bias which is reflected on the LOO residuals (TR part). The adjusted LOO residuals are computed with a new formula where matrix $\mathbf{P}$ is replaced by $\mathbf{P}(\lambda) = \mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^{\mathrm{T}}$, where $\mathbf{I}$ denotes the identity matrix. The parameter λ is locally optimized, providing the solution in a few iterations, with slight increase in computational time.

Inclusion of the TROP-ELM in experiments is two-fold. First, to demonstrate that the model combination can easily be applied to other variants of the ELM model. Second, the TROP-ELM uses a ranking algorithm to produce a different ordering than our suggested random permutation, and as such can give insight whether a heuristic approach to ordering is better than a random one.

4 Experiments

This section shows the comparison between the two groups of ELM models: one group employs the selection strategy (ELM* and TROP-ELM) while the other consists of our proposed method with the added combination layer (ELM(JMA) and TROP-ELM(JMA)). The comparison is done via squared error risk which is estimated as an average of 10 Monte-Carlo runs on a test set. The test set consists of one third of the data set in consideration, while the other two thirds constitute the training set. The training set is standardized to zero mean and unit variance, and the same parameters (mean and variance) are used to scale

Table 1. Data sets used in the experiments. N indicates the total number of samples in the data set without division into the training and test parts and d denotes the number of features.

Name	N	d	Name	N	d
Abalone	4177	8	Computer activity	8192	12
Bank_8FM	4500	8	Delta ailerons	7129	5
Boston housing	506	13	Servo	167	4
Breast cancer	194	32	Stocks	950	9

the test set. For each run, a total of 1000 models are trained to account for randomness of the ELM algorithm and the average of 1000 test errors is taken to be the estimated error for that run.

Experiments are performed on eight data sets from two repositories: the UCI machine learning repository [14] and the LIACC regression repository [15]. The data sets are listed in Table 1.

Three types of activation functions g are used: sigmoid, Gaussian and linear. For Gaussian functions, the centres of kernels are taken randomly from data points $\mathbf{x}_i$, while linear activations functions are simply identity functions for each feature, i.e., $g_{j_k}(\mathbf{x}_i) = x_i^k$, $k \in \{1, \ldots, d\}$, the k-th feature of sample i. A total of 50 sigmoid and 50 Gaussian kernels are used, which gives $M = 100 + d$ neurons, and the same number of models to train.

4.1 Performance Comparison

Tables 2 and 3 show the estimated squared error risk for the ELM and TROP-ELM variants respectively, and the improvement obtained when the combination is taken into account. The ELM* method selects the model that has the smallest LOO error in the candidate set (the best model) and is used instead of the full model with M neurons in order to provide a fair comparison between two strategies. The proposed combination approach always produces an improvement compared to a single model for both ELM variants. The reduced risk can range from small values of $1-2\%$ to much larger improvement of around 25% (Abalone). Smaller achievements are expected in data sets with a high number of samples as a single ELM is able to capture all complexity present in the data. Larger jumps are seen for data with moderate sample sizes, where more variability in the training phase is expected, and the combination alleviates that increased variability. The only exception is ELM*/ELM(JMA) for Breast cancer data. In this case, the increased risk estimate comes from LOO computation of larger models where the number of samples for training ($2/3 \cdot 194 \approx 129$) and models considered $M = 132$ pose problems for accurate estimation, and may lead to overfitting. Such extreme cases are potential pitfalls for JMA, since the models with overconfident LOO estimates are selected during the combination phase. This is where TROP-ELM plays an important role and provides the JMA with more stable results.

Table 2. Estimated risk (average test mean-squared error) for ELM* and ELM(JMA) models and the improvement (in percent) of the combination approach over the selection strategy.

Data set	ELM*	ELM(JMA)	(%)
Abalone	12.1	9.14	24.77
Bank_8FM	1.085e−3	1.044e−3	3.79
Boston housing	18.0	15.2	15.92
Breast cancer	1.19e+3	1.43e+3	−20.59
Computer activity	35.8	31.1	12.97
Delta ailerons	2.81e−8	2.74e−8	2.57
Servo	0.729	0.614	15.76
Stocks	0.831	0.716	13.85

Table 3. Estimated risk (average test mean-squared error) for TROP-ELM and TROP-ELM(JMA) models and the improvement (in percent) of the combination approach over the selection strategy.

Data set	TROP-ELM	TROP-ELM(JMA)	(%)
Abalone	6.39	5.95	6.97
Bank_8FM	1.081e−3	1.057e−3	2.23
Boston housing	18.7	15.9	15.20
Breast cancer	1.31e+3	1.19e+3	8.82
Computer activity	33.6	30.5	9.07
Delta ailerons	2.75e−8	2.71e−8	1.46
Servo	0.748	0.652	12.90
Stocks	0.926	0.781	15.68

The issue of ordering of neurons is less obvious. The ELM*/ELM(JMA) pair outperforms the TROP versions in some data sets (Boston housing, partially Breast cancer, Servo and Stocks), while it is inferior for the other data sets. The only noticeable difference is for Abalone, where ordering improves performance dramatically, while for the other data sets the increase (Bank and Delta ailerons) is quite small. This suggest that the ordering of neurons might not be so critical for accurate prediction.

4.2 Run Times

A great advantage of ELM is its low computational cost. The question is whether proposed combination procedure takes too much time compared to the original ELM. Table 4 summarizes the execution time for the ELM* and ELM(JMA) models. The execution time for the ELM* is computed as the amount of time required to train all M linear systems. The computational increase mostly depends on the sample size of the data set, and for larger ones the increase is quite

Table 4. Run times in *seconds* for ELM* and ELM(JMA) models and the increase in computation time (in percent) for the combination strategy with respect to the selection strategy.

Data set	ELM*	ELM(JMA)	(%)
Abalone	0.85	0.89	4.48
Bank_8FM	0.98	1.00	1.82
Boston housing	0.13	0.15	22.42
Breast cancer	0.08	0.12	42.17
Computer activity	1.98	2.02	2.05
Delta ailerons	1.39	1.41	1.25
Servo	0.05	0.07	37.82
Stocks	0.21	0.23	13.34

small, while for data sets with a couple of hundred samples there is substantial extra cost (around 40%), but such cost is still affordable and on a scale of less than one second.

Run times are computed in the Matlab environment and carried out on an Intel Xeon processor (E3-1200 family; 3.20GHz) using a single core. Each model is trained independently of the other models, and quadratic programming for JMA is solved with the *quadprog* function from the Optimization Toolbox.

It should be stressed that quadratic programming is only dependent on the number of models considered M, i.e., independent of the sample size N. This means that the combination phase takes almost the same amount of time for all data sets in our case. The extra cost is even more negligible for the TROP-ELM since there is additional optimization for the λ parameter. Solving quadratic problems for even larger cases when $M \approx 1000$ is still fast, but the actual bottleneck becomes the training phase for all 1000 models.

5 Conclusions

This paper addresses the issue of selection strategy for the Extreme Learning Machine and its variants. As explicated, this approach neglects the issue of model selection uncertainty which leads to degraded prediction accuracy. Instead, a combination procedure taking into account all available models must be considered to combat the problem. The proposed approach ELM(JMA) with Jackknife Model Averaging as the combination method of the LOO residuals shows better results than a single "best" model based on the same LOO errors. Extension to TROP-ELM(JMA) shows that the method can be easily adapted to the other ELM variants that are based on the selection strategy. The extra computational cost is quite low and only depends on the number of models considered.

Notation ELM(c) signifies that other criteria and combination methods can be paired instead of the leave-one-out residuals and Jackknife Model Averaging,

such as the Bayesian Information Criterion (BIC) where the model weights can easily be derived from the BIC scores.

In the experiments, we have used fixed starting number of neurons, but the question remains whether that number can automatically be selected based on the combination weights. One approach would be to start with some small number, say $M_b = 10$, train models and perform the combination, and check if the full model (or the most complex from this set) is included in the combination, i.e., $w^{M_b} > 0$. If it is, then add new batch of neurons and repeat the procedure until the most complex model is not present in the combination or if all newly added models are excluded.

Another question worth examining is the *pruning/screening step* or removal of poor models from the candidate set prior to combination. From a theoretical perspective this issue is addressed via weight assignment, but in practice due to finite sample size of the data there are difficulties in stable estimation of the parameters of larger models which is a potential pitfall for model combining.

Nevertheless, the success of the proposed combination strategy suggests that the practice of selecting one model from a candidate set leads to less accurate inference, and that some form of weighted average is required even in the case when building a single ELM network.

References

1. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing 70(1-3), 489–501 (2006)
2. Huang, G.B., Chen, L., Siew, C.K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks 17(4), 879–892 (2006)
3. Feng, G., Huang, G.b., Lin, Q., Gay, R.: Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Transactions on Neural Networks 20(8), 1352–1357 (2009)
4. Lan, Y., Soh, Y.C., Huang, G.B.: Constructive hidden nodes selection of extreme learning machine for regression. Neurocomputing 73(16–18), 3191–3199 (2010)
5. Rong, H.J., Ong, Y.S., Tan, A.H., Zhu, Z.: A fast pruned-extreme learning machine for classification problem. Neurocomputing 72(1–3), 359–366 (2008)
6. Miche, Y., van Heeswijk, M., Bas, P., Simula, O., Lendasse, A.: TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization. Neurocomputing 74(16), 2413–2421 (2011)
7. van Heeswijk, M., Miche, Y., Lindh-Knuutila, T., Hilbers, P., Honkela, T., Oja, E., Lendasse, A.: Adaptive ensemble models of extreme learning machines for time series prediction. In: Alippi, C., Polycarpou, M.M., Panayiotou, C.G., Ellinas, G. (eds.) ICANN 2009, Part II. LNCS, vol. 5769, pp. 305–314. Springer (2009)
8. Breiman, L.: Stacked regressions. Machine Learning 24(1), 49–64 (1996)
9. Draper, D.: Assessment and propagation of model uncertainty (with discussion). Journal of the Royal Statistical Society: Series B 57(1), 45–97 (1995)
10. Hoeting, J.A., Madigan, D., Raftery, A.E., Volinsky, C.T.: Bayesian model averaging : A tutorial. Statistical Science 14(4), 382–417 (1999)
11. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)

12. Allen, D.M.: The relationship between variable selection and data augmentation and a method for prediction. Techometrics 16(1), 125–127 (1974)
13. Hansen, B.E., Racine, J.S.: Jackknife model averaging. Journal of Econometrics 167(1), 38–46 (2012)
14. Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml
15. Torgo, L.: LIACC regression data sets, http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html