Extending Extreme Learning Machine with Combination Layer

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

[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

Abstract. We consider Extreme Learning Machine model for accurate regression estimation and the related problem of selecting appropriate number of neurons for the model. Selection strategies that choose "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. Method proposed in the paper is 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 prediction 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 basing all subsequent inference on that constructed model. Prediction of continuous values, such as daily temperature or stock market prices, is considered a *regression* problem or estimation of regression function.

In the paper we are concerned with regression problem 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, noise is assumed to be homoskedastic with 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 machines, kernel regression, nearest neighbour estimators and fuzzy regression. Each model class is characterized by approximation capabilities and training algorithms with different computational complexities.

The paper focuses on a specific type of neural network that is gaining popularity in recent years, namely Extreme Learning Machine (ELM) [1]. It is shown that a single feedforward hidden layer with input weights randomly assigned has universal approximation capability for any target function, i.e. 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 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, 3, 4] focus on choosing a single model from a candidate set by optimizing some criterion; 2) *ensemble of many ELMs* [5], i.e. the candidate models *all* contribute to the weighted average as the final model. 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 keeping better ones.

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 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 [6]; 2) philosophical issue of ignoring model selection uncertainty by making inference solely on a single model once it is identified [7]. Combination methods have been proposed both in Bayesian statistics, where Bayesian Model Averaging [8] 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 [9]. The proposed method 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 ELM which can be extended to include 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 as selection strategy: *pruning* − starting from a large number of neurons and then removing unnecessary ones [4] and *constructive* − building up from smaller pool of neurons until some condition (usually error) does not improve with additional complexity [2, 3]. Both methods consider several alternative network structures and finally output "the best" model. The idea is to consider them all together and form a weighted average.

Proposed method Extreme Learning Machine-*combination*, denoted ELM(c), is ELM model where selection phase is replaced with *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. Following subsections present the building blocks which constitute 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 $y_i \in \mathbb{R}$. ELM network with M neurons is constructed by first computing hidden matrix $\mathbf{H}$

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

with j-th neuron having activation function g_j, input weights $\mathbf{w}^{\mathrm{i}}_j$, bias b_j and both $\mathbf{w}^{\mathrm{i}}_j$ and b_j are randomly generated. Hidden layer output weights $\boldsymbol{\beta}$ are found by solving linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{y}$, with the Moore-Penrose generalized inverse of the matrix $\mathbf{H}$ and the target values, i.e. $\boldsymbol{\beta} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y}$. Matrix $\mathbf{H}$ is sometimes called feature mapping or feature space of ELM. Any function that is bounded non-constant piecewise continuous function can be taken as activation function in 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 over all validation parts is taken as a measure of generalization ability of the model, and the model with highest measure is taken to be the best. For selection strategy, the model with smallest average validation error is assumed to be most suitable.

The extreme case is $k = N$ or leave-one-out (LOO) CV, where each sample plays the role as a sole sample in validation set. In the case of linear regression, the LOO error can be computed in a simple manner with single fit of the model using PRESS statistic [10]. This 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 leave-one-out residuals for all samples are computed with PRESS formula

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

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

2.3 Jackknife Model Averaging

Jackknife model averaging (JMA) has been recently proposed in [11], a model combining method minimizing LOO-CV error. The authors show that considering LOO residuals of all models, the combination asymptotically achieves 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 [11] 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 linear combination of leave-out-out residuals off all models, and for the purpose of presentation we use similar notation as in [11]. Denote with $\tilde{\mathbf{e}}^m = [\tilde{e}_1^m, \dots, \tilde{e}_N^m]^{\mathrm{T}}$ the leave-out-out residual vector of m-th model in 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 $\tilde{\mathbf{e}} = [\tilde{\mathbf{e}}^1, \dots, \tilde{\mathbf{e}}^M]$ and jackknife estimate of generalization error is given by

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

with $\mathbf{S} = \tilde{\mathbf{e}}(\mathbf{w})^{\mathrm{T}}\tilde{\mathbf{e}}(\mathbf{w})/N$. 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 be easily solved with publicly available software packages. All that is needed to solve the minimization problem are 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 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 first m columns of $\mathbf{H}$. That is, keep increasing number of neurons by 1 (starting from 1 until M), and compute new model. During this training phase, gather LOO residual vectors $\tilde{\mathbf{e}}^m$ for each model using Eq. (2). Final step is JMA combining of all M models using $\tilde{\mathbf{e}}$ in minimization problem (Eq. (3)), which gives model weights $\mathbf{w} = [w^1, \dots, w^M]^{\mathrm{T}}$. The function estimate of 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 output $\hat{f}^{i_k}(\mathbf{x}_n)$ for those models whose weights are greater than zero $\mathcal{M} = \{w^m > 0 | 1 \leq m \leq M\} = \{i_1, \ldots, i_K\}$ and output the weighted average over those models $\hat{f}(\mathbf{x}_n) = \sum_{k=1}^{K} w^{i_k} \hat{f}^{i_k}(\mathbf{x}_n)$.

Reason for considering only M models is that there is exponential number of possible models, i.e. all subsets selection problem with M features. The other issue is 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 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 weights $\boldsymbol{\beta}$.

3 TROP-ELM

Tikhonov Regularized Optimally Pruned Extreme Learning Machine (TROP-ELM) [4] brings two adjustments into original ELM. First phase is ranking of neurons with LARS method and selecting the appropriate number of neurons by minimizing LOO error (OP part). The other improvement is L_2 regularization on the weights $\boldsymbol{\beta}$, by introducing slight bias which is reflected on LOO residuals (TR part). The adjusted LOO residuals are computed with new formula where matrix $\mathbf{P}$ is replaced with $\mathbf{P}(\lambda) = \mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^{\mathrm{T}}$, with $\mathbf{I}$ denoting identity matrix. Parameter λ is locally optimized providing solution in couple of steps which is still obtained in reasonable time.

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

4 Experiments

This section shows the comparison between ELM models with selection strategy, and our proposed method with added combination layer. The comparison is done via squared error risk which is estimated as average over 10 Monte-Carlo runs on a test set. Test set consists of one third of data set in consideration, while the other two thirds are part of training set. Training set is normalized to zero mean and unit variance, and the same parameters (mean and variance) are used to scale test set. For each split, a total of 1000 models are trained to account for randomness of the ELM algorithm and averaged over for that split.

Experiments are performed on several data sets from UCI machine learning repository [12] and these include (name, number of samples, number of features): (Abalone, 4177, 8), (Bank_8FM , 4500, 8), (Boston housing, 506, 13), (Breast cancer, 194, 32), (Computer activity, 8192, 12), (Delta ailerons, 7129, 5), (Servo, 167, 4) and (Stocks, 950, 9).

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 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. Total of 50 sigmoid and 50 gaussian kernels are used, which gives $M = 100 + d$ neurons, and the same amount of models to train.

Performance comparison. Table 1 shows the estimated squared error risk for all considered methods, and the improvement obtained when combination is taken into account. ELM* model is ELM(JMA) where combination is replaced with selection, as to provide a fair comparison between two strategies, as opposed to using full model with M neurons. The proposed combination approach always produces an improvement compared to a single model, both for ELM* and TROP-ELM. 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 high number of samples as 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 training phase is expected, and combination alleviates for that increased variability. The only exception is ELM*/ELM(JMA) for Breast cancer data. Increased risk comes from LOO computation of larger models where number of samples for training $(2/3 \cdot 194 = 129)$ and models considered $M = 132$ pose problems for accurate estimation, and usually lead to overfitting. Such extreme cases are potential pitfalls for JMA since models with overconfident LOO estimates are selected during combination phase. This is where TROP-ELM plays important role and provides the JMA with more stable results.

Table 1. Estimated risk (average test mean-squared error) for all models and the improvement in percentage of combination layer for both ELM* and TROP-ELM.

model	Abalone	Bank	BostHou	BreastC	CompAct	DeltaAil	Servo	Stocks
ELM*	12.1	1.085e-3	18.0	1.19e+3	35.8	2.81e-8	0.729	0.831
ELM(JMA)	9.14	1.044e-3	15.2	1.43e+3	31.1	2.74e-8	0.614	0.716
(%)	24.77	3.79	15.92	-20.59	12.97	2.57	15.76	13.85
TROP-ELM	6.39	1.081e-3	18.7	1.31e+3	33.6	2.75e-8	0.748	0.926
TROP-(JMA)	5.95	1.057e-3	15.9	1.19e+3	30.5	2.71e-8	0.652	0.781
(%)	6.97	2.23	15.20	8.82	9.07	1.46	12.90	15.68

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

Run times. Great advantage of ELM is its really low computational cost. The question is whether proposed combination procedure takes too much time compared to the original ELM. Table 2 summarizes the execution time for ELM* and ELM(JMA) models. Execution time for ELM* is computed as finding solution for 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 small, while for data sets with 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.

Running times are computed using Matlab environment and carried on Intel Xeon processor (E3-1200 family; 3.20GHz) using only one core. Each model is trained independently of other models, and quadratic programming for JMA is solved with Matlab's *quadprog* function.

Table 2. Running time in *seconds* and percentage of increase with respect to a selection strategy.

model	Abalone	Bank	BostHou	BreastC	CompAct	DeltaAil	Servo	Stocks
ELM*	0.85	0.98	0.13	0.08	1.98	1.39	0.05	0.21
ELM(JMA)	0.89	1.00	0.15	0.12	2.02	1.41	0.07	0.23
(%)	4.48	1.82	22.42	42.17	2.05	1.25	37.82	13.34

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

5 Conclusions

The paper addresses the issue of selection strategy for Extreme Learning Machine and its variants. As explicated, this approach neglects the issue of model selection uncertainty, which can be harmful for accurate prediction. 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 LOO residuals shows better results than a single "best" model based on same LOO errors. Extension to TROP-ELM(JMA) shows that the proposed method can be easily adapted to other ELM variants that are based on selection strategy. The extra computational cost is quite low and only depends on the number of models considered.

Notation ELM(c) also allows other criteria and combination methods to be used instead of leave-out-out cross-validated error, such as Bayesian Information Criterion (BIC) where model weights can be easily derived from BIC scores.

In the experiments, we have used fixed starting number of neurons, but the question remains on whether that number can be automatically selected based on combination weights. One approach would be to start with some small number, say $M_b = 10$, train models and perform 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.

Other question worth examining is the *pruning/screening step* or removal of poor models from candidate set prior to combination. From the theory in [11] the answer is negative, but in practice due to finite sample size of data there are difficulties in stable estimation of parameters of larger models which is a potential pitfall for model combining.

Nevertheless, the success of the proposed combination strategy suggests that practice of selecting number of neurons from a candidate set leads to less accurate inference, and that some form of weighted average is required.

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] 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)
[4] 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)
[5] 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: ICANN 2009, Part II. LNCS, vol. 5769, pp. 305–314 (2009)
[6] Breiman, L.: Stacked regressions. Machine Learning 24(1), 49–64 (1996)
[7] Draper, D.: Assessment and Propagation of Model Uncertainty (with discussion). Journal of the Royal Statistical Society: Series B 57(1), 45–97 (1995)
[8] Hoeting, J.A., Madigan, D., Raftery, A.E., Volinsky, C.T.: Bayesian model averaging : A tutorial. Statistical Science 14(4), 382–417 (1999)
[9] Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
[10] Allen, D.M.: The relationship between variable selection and data augmentation and a method for prediction. Techometrics 16(1), 125–127 (1974)
[11] Hansen, B.E., Racine, J.S.: Jackknife model averaging. Journal of Econometrics 167(1), 38–46 (2012)
[12] Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml