Extreme Learning Machines with Regularization for the Classification of Gene Expression Data

Dániel T. Várkonyi, Krisztián Buza

Eötvös Loránd University, Faculty of Informatics, Department of Data Science and Engineering,
Telekom Innovation Laboratories, Budapest, Hungary
{varkonyid,buza}@inf.elte.hu
WWW home page: http://t-labs.elte.hu

Abstract: Extreme learning machine (ELM) is a special single-hidden layer feed-forward neural network (SLFN), with only one hidden layer and randomly chosen weights between the input layer and the hidden layer. The advantage of ELM is that only the weights between hidden layer and output layer need to be trained, therefore, the computational costs are much lower, resulting in moderate training time. In this paper, we compare ELMs with different regularization strategies (no regularization, L1, L2) in context of a binary classification task related to gene expression data. As L1 regularization is known to lead to sparse structures (i.e., many of the learned weights are zero) in case of various models, we examine the distribution of the learned weights and the sparsity of the resulting structure in case of ELM.

Keywords: Extreme Learning Machine, Classification, Logistic Regression, L1 Regularization, Gene Expression

1 Introduction

Recent advances in neural networks lead to breakthroughs in many applications in various domains, such as games, finance, medicine and engineering, see e.g. [6], [17], [21]. In most cases, gradient-based training is used to find appropriate values of the weights of the network. Gradients are usually calculated with back propagation (BP) [16]. However, gradient-based training may be too slow in certain applications.

For the above reason, other training approaches were proposed, such as subset selection [13], [4], second order optimization [7], and global optimization [2], [19], see also [1] for details. All the aforementioned algorithms may stuck into local minima, and suffer from slow convergence.

Extreme Learning Machines (ELM) were introduced by Huang et al. [10], [11] as a special single layer feed-forward neural network. ELMs are general function approximators. ELMs overcome the main disadvantages of feed-forward neural networks (FNN). The training speed of ELM is much faster than that of FNN, since ELM has only one hidden layer, the input weights (i.e., the weights between the input layer and the hidden layer) are initialized once, and not trained iteratively. With a well chosen convex activation function, the issue of stucking into local minima can be avoided.

While neural networks are powerful, due to their complexity, in the lack of appropriate regularization, they tend to overfit the data. In the era of deep learning, L1 regularization became popular due to various reasons: on one hand, sparse structures resemble the brain, on the other hand, they lead to computationally cheap models as the resulting zero-weights correspond to the lack of connections, thus they may be omitted.

Regularized ELMs have been shown to outperform non-regularized ELMs [5], [12], [15], [20]. However, as opposed to our study, none of the aforementioned works focused on the classification of gene expression data and the sparsity of the learned weights.

In our study, we compare various regularization techniques – in particular: L1 and L2 regularization as well as the lack of regularization – in context of classification of gene expression data using ELM.

2 Basic Notation and Problem Formulation

First, we define the classification problem and introduce the basic notation which is used in this paper. We are given a set $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ of training data containing instances $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}) \in \mathbb{R}^n$. For each instance $x^{(i)}$, its label $y^{(i)}$ is also given. The set of labels is denoted by $Y = \{y^{(1)}, y^{(2)}, \ldots, y^{(m)}\}$. Each label $y^{(i)} \in \{0, 1\}$, 0 denotes a negative instance and 1 denotes a positive instance.

We use $x_1, x_2, \ldots x_n$ to denote the input nodes. H is the only hidden layer and the number of units in the hidden layer is denoted by L. We use h_i to denote the ith hidden node. The activation value of ith hidden node for an instance x is $h_i[x] \in \mathbb{R}$, $b_i \in \mathbb{R}$ is the bias of ith hidden node, $a_{i,j} \in \mathbb{R}$ is the randomly initialized weight from x_i to jth hidden node.

The output layer contains only one single unit, $\beta_i \in \mathbb{R}$ is the weight from h_i to the output unit and $b_o \in \mathbb{R}$ is the bias of output node. $ELM(x) \in \mathbb{R}$ is the activation of the output unit for an input instance x. The structure of ELM is shown in Fig. 1.

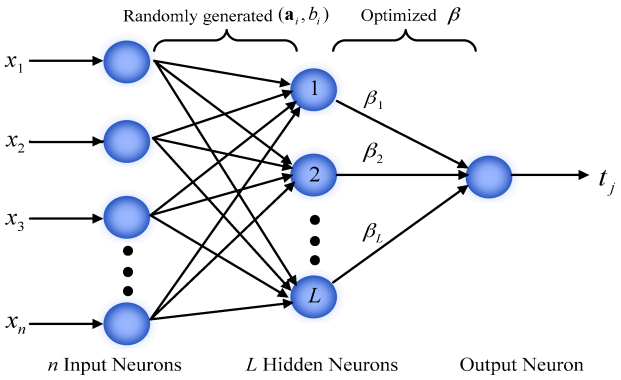

Figure 1: Structure of ELM for binary classification [14]

As described in Section 3.2, training ELM leads to a minimization problem:

$$\min_{\beta_i} J(\beta) \tag{1}$$

where $J(\beta)$ is the cost function of ELM.

3 Methods

In this chapter we will introduce the fundamentals required to understand our work such as ELM, logistic regression and regularization of logistic regression.

3.1 ELM

Extreme Learning Machine is a special kind of single layer feed forwarded network. The network has only one hidden layer. The weights between the input layer and the hidden layer (input weights, for short) are initialized once and not trained, i.e., they remain unchanged. The output weights between the hidden layer and the output layer are trained iteratively. As the input weights remain in their initial state and only the output weights are trained, the training time of an ELM is much lower than that of a comparable single layer feed-forward neural network (SLFN) [11].

The output of an ELM is the value of the activation function applied to the weighted sum of the activation values of hidden nodes:

$$ELM(x) = g(\sum_{j=1}^{L} \beta_j h_j[x] + b_o) \tag{2}$$

where g is the activation function. Since stucking into local minima needs to be avoided, a convex activation, in particular, the sigmoid function was chosen as activation function:

$$g(z) = \frac{1}{1+e^{-z}}. \tag{3}$$

The value of ith hidden node for x input is:

$$h_i[x] = g(\sum_{j=1}^{n} a_{j,i} x_j + b_j). \tag{4}$$

3.2 Logistic Regression in the Output Layer of ELM

Logistic regression (LR) is one of the most often and effectively used binary classification method. In our case, the output layer of ELM implements logistic regression based on the hidden units' activation values. Thus the cost function without regularization is:

$$J(\beta) = \frac{1}{m} \sum_{i=1}^{m} Cost(ELM(x^{(i)}), y^{(i)}) \tag{5}$$

where:

$$Cost(ELM(x), y) = \begin{cases} -log(ELM(x)), & \text{if y=1} \\ -log(1 - ELM(x)), & \text{if y=0.} \end{cases} \tag{6}$$

Using (5) with (6), the cost function can be equivalently written as:

$$\begin{aligned} J(\beta) = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} log(ELM(x^{(i)})) \\ + (1 - y^{(i)}) log(1 - ELM(x^{(i)}))). \end{aligned} \tag{7}$$

The partial derivative of the cost function w.r.t. the kth parameter (β_k) is:

$$\frac{\partial}{\partial \beta_k} J(\beta) = \frac{1}{m} \sum_{i=1}^{m} (ELM(x^{(i)}) - y^{(i)}) h_k[x^{(i)}] \tag{8}$$

Logistic regression can be trained with gradient descent. That is: after initializing the parameters β_k, in every iteration, all β_k-s are updated simultaneously according to the following rule:

$$\beta_k = \beta_k - \alpha \frac{\partial}{\partial \beta_k} J(\beta) \tag{9}$$

where α is the learning rate.

3.3 LASSO and Ridge Regression in the Output Layer of ELM

In logistic regression and generally in all regression models, it is a common goal to keep the model as simple as possible. Regularization punishes a complex model, in particular, a penalty term is added to the cost function. Ridge regression (L2) adds squared magnitude of the coefficients as penalty term to the loss function. LASSO (Least Absolute Shrinkage and Selection Operator) regression adds absolute value of the coefficients as penalty term to the loss function. The key difference between these techniques is that LASSO shrinks the less important features' coefficients to zero, thus, leads to a model with less complex structure.

In our case, the L1 regularized cost function is:

$$J(\beta) = \frac{1}{m} \sum_{i=1}^{m} Cost(ELM(x^{(i)}), y^{(i)}) + \frac{\lambda}{m} \sum_{j=1}^{L} |\beta_j|, \tag{10}$$

its partial derivative w.r.t. the kth parameter (β_k) is:

$$\frac{\partial}{\partial \beta_k} J(\beta) = \frac{1}{m} \sum_{i=1}^{m} (ELM(x^{(i)}) - y^{(i)}) h_k[x^{(i)}] + \frac{\lambda}{m} sign(\beta_k). \tag{11}$$

In our case, the L2 regularized cost function is:

$$J(\beta) = \frac{1}{m} \sum_{i=1}^{m} Cost(ELM(x^{(i)}), y^{(i)}) + \frac{\lambda}{m} \sum_{j=1}^{L} \beta_j^2, \tag{12}$$

and its partial derivative w.r.t. the kth parameter (β_k) is:

$$\frac{\partial}{\partial \beta_k} J(\beta) = \frac{1}{m} \sum_{i=1}^{m} (ELM(x^{(i)}) - y^{(i)}) h_k[x^{(i)}] + 2\frac{\lambda}{m} \beta_k \tag{13}$$

where λ is the regularization coefficient which shows the weight of the penalty term in connection with the average cost.

Using the above partial derivatives, similarly to the case of logistic regression, gradient descent can be used to train L1 and L2-regularized ELMs.

3.4 Our Approach

We propose to use ELM for the classification of gene expression data. We train the weights of the output layer of ELM with L1-regularized logistic regression (LASSO).

4 Dataset and Preprocessing

Classification of gene expression data is a challenging task with prominent applications in the medical domain, such as the diagnosis of different subtypes of cancer, see e.g. [3], [18] and the references therein. For this reason, we compare ELMs with different regularization techniques on a publicly-available gene expression dataset, called Mice Protein Expression Data.

The Mice Protein Expression Dataset[1] is available from the UCI repository The main properties of the dataset are summarized in Tab. 1.

Mice Protein Expression Dataset consists of measurements of gene expression levels in mice. In total, the expression levels of 77 genes were measured for 72 mice, out of which 34 were trisomic (trisomy in mice may be seen as a model of Down syndrome in human), while 38 belonged to the control group (i.e., mice that are not affected by the disease). The expression levels of each gene were measured 15-times for each mouse, resulting in a total of $72 \times 15 = 1080$ instances, each of them containing 77 gene expression features, see also [9] for details. For each mouse, its genotype, behavior and treatment are available in the dataset. In our experiment, we used the genotype as class label.

The data contained high number of missing values for some of the gene expression features (in particular for BAD_N, BCL2_N, pCFOS_N, H3AcK18_N, EGR1_N, H3MeK4_N genes). We ignored these features.

Some of the instances of the remaining dataset contained missing values in other features, these instances were also ignored resulting in a dataset of 1047 instances and 71 gene expression features.

We split the data into train and test sets as follows: the test set contains 346 randomly selected instances, while the remaining 701 instances are assigned to the training set.

5 Experimental Settings

We compared three ELMs that differ in terms of the applied regularization technique: in the first model we did not use any regularization at all, in the second and third models we used L1 and L2 regularization, respectively. All three models were initialized with the same parameters.

For the Mice Dataset, the input weights, hidden biases and the output bias b_o were randomly sampled from uniform distributions between -0.1 and 0.1, -0.75 and 0.75, as well as -1 and 1. The initial value of each output weight β_k was set to zero.

Settings of the hyperparameters of ELMs, such as number of hidden nodes, learning rate, regularization coefficient and the number of training iterations are summarized in Tab. 2.

6 Experimental Results

We use the area under receiver-operator characteristic curve (AUC) [8] to assess the accuracy of the examined models. Fig. 2 shows the AUC on the test set as the function of the number of training iterations. As expected, AUC grows with increasing number of iterations. As one can see, the L1-regularized model and the model without regularization outperform the L2-regularized model. The AUC of the model without regularization and with L1 regularization converge to the approximately same value.

Table 1: Characteristic of Mice Protein Expression Dataset

Property	Value
Data Set Characteristics	Multivariate
Number of Instances	1080
Area	Life science
Attribute Characteristics	Real
Number of Attributes	82
Associated Task	Classification
Missing values	Yes

[1] https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression

Table 2: Hyperparameters of ELMs in case of the Mice Gene Expression Dataset

Parameter	Value(s)
Number of hidden nodes (L)	250
Learning rate (α)	0.1
Regularization coefficient (λ)	0.01
Number of training iterations	60M

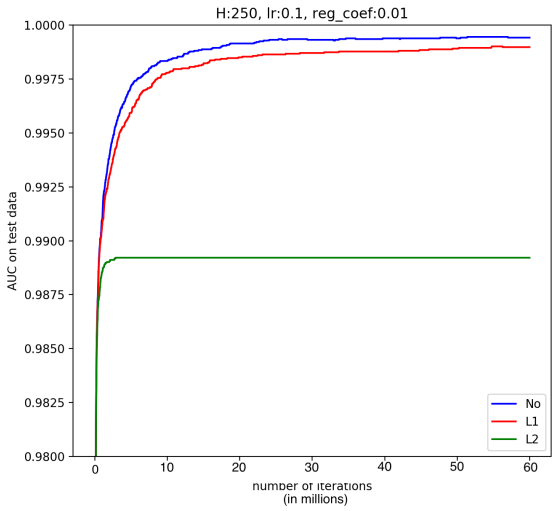

Figure 2: AUC on the test set as function of the number of training iterations

Table 3: AUC values of different methods

Method	Value(s)
ELM without regularization	0.999
ELM with L1 regularization	0.998
ELM with L2 regularization	0.989
SVM	0.960

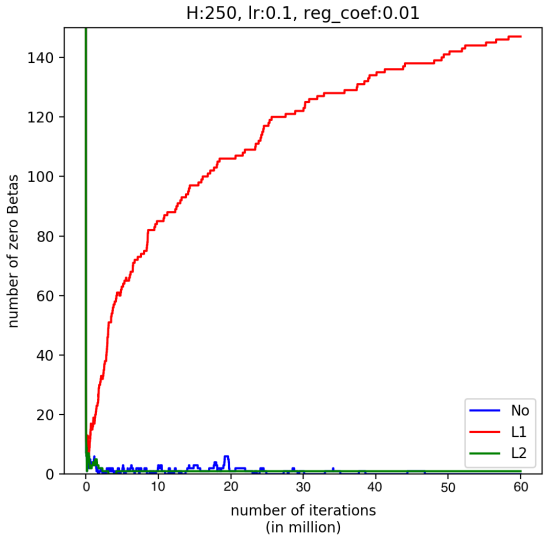

Figure 3: Number of zero β_k-s as the function of the number of iterations

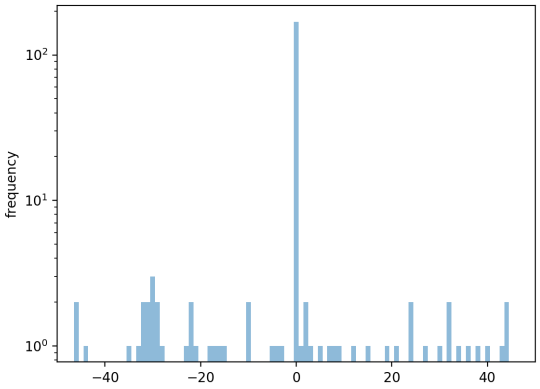

Figure 4: Distribution of β_k-s

L1 regularization is known to lead to sparse structures. Especially in case of highly-correlated features with similar predictive power, L1 regularization tends to prefer the best out of the slightly different features in the sense that a relatively high weight will be assigned to this "best" feature, while zero weights will be assigned to the other ones. In contrast, L2 regularization distributes the weights more "fairly" in the sense that highly correlated features will receive approximately the same weights.

For the above reasons, in case of L1 regularization, we expect many of the β_k weights being approximately zero. In accordance with these expectations, we observed that more than two-third of all the β_k weights were less than the learning rate after 125 million iterations. In this respect, the L1-regularized model is substantially different from the other two models as it can be seen in Fig. 3. The distribution of β_k-s can be seen in Fig. 4.

The sparsity of the L1-regularized model, i.e., the high number of β_k-s being (close to) zero, leads to a computationally simpler model: only the activation values of those hidden units need to be calculated for which the corresponding β_k is different from zero. This makes L1-regularized ELMs better suitable for scenarios in which the computational power is limited, such as embedded systems in case of wearable medical devices or self-driving cars.

7 Conclusion and Outlook

In this paper, we compared regularization approaches in context of classification of gene expression data with extreme learning machines. We observed that L1 regularization leads to sparse models that are computationally simpler than the comparable models without regularization or with L2 regularization. Therefore, L1-regularized models may be better suitable for embedded systems.

We plan to perform similar experiments on further gene expression datasets as part of our future work.

Acknowledgement

This work was supported by the project no. 20460-3/2018/FEKUTSTRAT within the Institutional Excellence Program in Higher Education of the Hungarian Ministry of Human Capacities. This work was also supported by Telekom Innovation Laboratories (T-Labs), the Research and Development unit of Deutsche Telekom.

References

[1] Albadr, M., Tiuna, S.: Extreme Learning Machine: A Review. International Journal of Applied Engineering Research **12** (2017) 4610–4623

[2] Branke, J.: Evolutionary algorithms for neural network design and training. In Proceedings of the First Nordic Workshop on Genetic Algorithms and its Applications. (1995) 1–21

[3] Buza, K.: Classification of gene expression data: a hubness-aware semi-supervised approach. Computer methods and programs in biomedicine **127** (2016) 105–113

[4] Chen, S.,Cowan, C. F., Grant, P. M.: Orthogonal least squares learning algorithm for radial basis function networks. IEEE Transactions on neural networks **2** (1991) 302–309.

[5] Wanyu, D.,Qinghua, Z., Lin, C.: Regularized extreme learning machine. IEEE symposium on computational intelligence and data mining (CIDM2009) (2009) 389–395.

[6] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542** (2017) 115–118

[7] Hagan, M. T., Menhaj M. B.: Training feedforward networks with the Marquardt algorithm. IEEE transactions on Neural Networks **5** (1994) 989–993.

[8] Hanley, J. A., McNeil B. J.: A method of comparing the areas under receiver operating characteristic curves derived from the same cases. Radiology **148**(3) (1983) 839–843

[9] Higuera, C., Gardiner, K. J., J. Cios, K. J.: Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome. PLoS ONE **10**(6) (2015) e0129126

[10] Huang, G., Zhu, Q., Siew, CK.: Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. Proceedings of the International Joint Conference on Neural Networks **2** (2004) 985–990

[11] Huang, G., Zhu, Q., Siew, CK.: Extreme learning machine:theory and applications. Neurocomputing **70** (2006) 489–501.

[12] Iosifidis, A., Tefas, A., Pitas, I.: Extreme learning machine for large-scale media content analysis. Procedia Computer Science **53** (2015) 420–427.

[13] Li, K.,Peng, J.-X.,Irwin, G. W.: A fast nonlinear model identification method. IEEE Transactions on Automatic Control **50** (2005) 1211–1216.

[14] Li, Y., Zhang, S., Yin, Y., Xiao, W., Zhang, J.: A Novel Online Sequential Extreme Learning Machine for Gas Utilization Ratio Prediction in Blast Furnaces. Sensors **17** (2017) 1847.

[15] Martínez-Martínez, J., Escandel-Montero, P., Soria-Olivas, E., Martín-Guerrero, J., Magdalena-Benedito, R., Gómez-Sanchis J.: Regularized extreme learning machine for regression problems. Neurocomputing **74** (2011) 3716–3721.

[16] Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning representations by back-propagating errors. Nature **323** (1986) 533–536.

[17] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L.,van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis D.: Mastering the game of Go with deep neural networks and tree search. Nature **529** (2016) 484–489

[18] Sotiriou, C., Soek-Ying, N.,McShane, L. M.,Korn, E. L., Long, P. M.,Jazaeri, A., Martiat, P., Fox, S. B.,Harris, A. L., Liu E. T.: Breast cancer classification and prognosis based on gene expression profiles from a population-based study. Proceedings of the National Academy of Sciences **100** (2003) 10393—10398.

[19] Yao, X.:A review of evolutionary artificial neural networks. International Journal of Intelligent Systems **8** (1993) 539–567.

[20] Yu, Q., Miche, Y., Eirola, E., van Heeswijk, M., Séverin, E., Lendasse A.: Regularized extreme learning machine for regression with Missing Data. Neurocomputing **102** (2013) 45–51.

[21] Yuchi, T.,Kexin, P., Suman, J., Baishakhi, R.: DeepTest: automated testing of deep-neural-network-driven autonomous cars. Proceedings of the 40th International Conference on Software Engineering (ICSE'2018) 303–314