Neurocomputing 149 (2015) 187–197

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Binary/ternary extreme learning machines

Mark van Heeswijk*, Yoan Miche

Aalto University School of Science, Department of Information and Computer Science, P.O. Box 15400, FI-00076 Aalto, Finland

ARTICLE INFO

Article history:
Received 28 August 2013
Received in revised form
17 December 2013
Accepted 19 January 2014
Available online 28 September 2014

Keywords:
Extreme learning machine
Hidden layer initialization
Intrinsic plasticity
Random projection
Binary features
Ternary features

ABSTRACT

In this paper, a new hidden layer construction method for Extreme Learning Machines (ELMs) is investigated, aimed at generating a diverse set of weights. The paper proposes two new ELM variants: Binary ELM, with a weight initialization scheme based on $\{0,1\}$–weights; and Ternary ELM, with a weight initialization scheme based on $\{-1,0,1\}$–weights. The motivation behind this approach is that these features will be from very different subspaces and therefore each neuron extracts more diverse information from the inputs than neurons with completely random features traditionally used in ELM. Therefore, ideally it should lead to better ELMs. Experiments show that indeed ELMs with ternary weights generally achieve lower test error. Furthermore, the experiments show that the Binary and Ternary ELMs are more robust to irrelevant and noisy variables and are in fact performing implicit variable selection. Finally, since only the weight generation scheme is adapted, the computational time of the ELM is unaffected, and the improved accuracy, added robustness and the implicit variable selection of Binary ELM and Ternary ELM come for free.

1. Introduction

The core idea of the Extreme Learning Machine (ELM) [1,2] is that it creates a single-layer feedforward neural network (SLFN) consisting of a randomly initialized hidden layer which randomly projects the inputs into a high-dimensional space. These randomly projected inputs are then transformed in a nonlinear way using some (often) nonlinear transfer function like tanh. Finally, the training of the ELM consists of solving the linear system formed by these nonlinearly transformed outputs of the hidden layer, and their corresponding target values [1,2].

The fact that the hidden layer is not touched after initialization and training consists of solving a linear system, makes the ELM very fast compared to other learning methods based on for example back-propagation or gradient-descent [1,2]. However, an aspect of the ELM that has not received much attention so far is how to exactly initialize the hidden layer. Typically, some heuristics are used and the random layer weights and biases are drawn from a uniform distribution in interval $[-5,5]$ (assuming that the data is normalized to be zero mean and unit variance) [3], or from another probability distribution like the Gaussian distribution [4]. However, heuristics like these are not necessarily optimal for any given data set and it is possible to improve the hidden layer initialization by adapting it to the problem at hand.

One approach for adapting the hidden layer to the context is the mechanism of batch intrinsic plasticity (BIP) [5–7]. The idea of BIP is that it adapts the slope and bias of the hidden layer neurons such that their outputs are approximately exponentially distributed. Given that the exponential distribution is the maximum entropy distribution, the information transmission of the neurons is maximized, resulting in a better model [8].

However, given that a transfer function typically looks like $f(\mathbf{w}^T\mathbf{x}+b)$, and $\mathbf{w}^T\mathbf{x}$, the inner product between weight vector $\mathbf{w}$ and input $\mathbf{x}$, can be rewritten as $\mathbf{w}^T\mathbf{x} = |\mathbf{w}\|\mathbf{x}| \cos\theta$, where θ is the angle between vectors $\mathbf{w}$ and $\mathbf{x}$, it can be seen that the diversity of neuronal inputs is mostly affected by the diversity of the norms of vectors $\mathbf{w}$ and $\mathbf{x}$ and their angle θ. Although BIP adapts the scaling of the input weights (and with that, the expected value of $|\mathbf{w}\|\mathbf{x}|$) such that the neuron operates in a useful regime, BIP does not optimize the weight generation scheme itself. This suggests that in order to further improve the diversity of the information extracted by the hidden layer, the diversity of the angle θ between the weight vectors and the inputs could be optimized. In this paper, this is achieved by using a binary $\{0,1\}-$weight scheme, or a ternary $\{-1,0,1\}-$weight scheme. By using a weight scheme like this, each neuron in the hidden layer focuses on a particular subspace of the variables, and the diversity of the extracted information is improved. Furthermore, the binary and ternary weight schemes allow the ELM to perform implicit variable selection, because neurons that incorporate useful variables extract more useful information and receive higher weight, while neurons that incorporate bad variables extract less useful information and are given lower weight.

* Corresponding author.
E-mail address: mark.van.heeswijk@aalto.fi (M. van Heeswijk).

http://dx.doi.org/10.1016/j.neucom.2014.01.072
0925-2312/© 2014 Elsevier B.V. All rights reserved.

Experiments show that especially the ternary weight scheme can generally improve the achieved test error. Furthermore, it is shown that the Binary ELM and the Ternary ELM are more robust against irrelevant and noisy variables and are in fact performing implicit variable selection. These advantages come at no increase in computational cost in comparison to drawing the random weights from e.g. a uniform or Gaussian distribution, since only the weight generation scheme is adapted.

The rest of the paper is organized as follows. Section 2 of the paper discusses the background and theory of ELM, and gives a short overview of ELM variants as well as preliminaries and methods relevant for this paper. In particular, it is discussed how to perform efficient model selection and optimization of the L2 regularization parameter in ELM, which is important for training robust models. Furthermore, BIP is discussed, since it is useful to adapt the scaling of the hidden layer weights such that the neurons operate in an optimal regime. BIP is also important because it allows us to conclude that any observed differences in performance between ELMs are due to the different weight generation scheme. Section 3 discusses the proposed binary and ternary weight schemes. Finally, Section 4 contains the experiments and analysis which form the validation for the proposed approach.

2. Preliminaries

2.1. Regression/classification

In this paper, the focus is on the problem of regression, which is about establishing a relationship between a set of output variables (continuous) $y_i \in \mathbb{R},\ 1 \le i \le M$ (single-output here) and another set of input variables $\mathbf{x}_i = (x_i^1, \ldots, x_i^d) \in \mathbb{R}^d$. Note that although in this paper the focus is on regression, the proposed pretraining approach can just as well be used when applying the ELM in a classification context.

2.2. Extreme Learning Machine (ELM)

The ELM algorithm is proposed by Huang et al. [2] and uses Single-Layer Feedforward Neural Networks (SLFN). The key idea of ELM is the random initialization of a SLFN weights. Below, the main concepts of ELM as presented in [2] are reviewed.

Consider a set of N distinct samples $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Then, a SLFN with M hidden neurons is modeled as the following sum

$$\sum_{i=1}^{M} \boldsymbol{\beta}_i f(\mathbf{w}_i \cdot \mathbf{x}_j + b_i), \quad j \in [1, N], \tag{1}$$

with f being the activation function, $\mathbf{w}_i$ the input weights to the ith neuron in the hidden layer, b_i the hidden layer biases and $\boldsymbol{\beta}_i$ the output weights.

In the case where the SLFN would perfectly approximate the data (meaning the error between the output $\hat{y}_i$ and the actual value y_i is zero), the relation is

$$\sum_{i=1}^{M} \boldsymbol{\beta}_i f(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = y_j, \quad j \in [1, N], \tag{2}$$

which can be written compactly as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}, \tag{3}$$

where $\mathbf{H}$ is the hidden layer output matrix defined as

$$\mathbf{H} = \begin{pmatrix} f(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & f(\mathbf{w}_M \cdot \mathbf{x}_1 + b_M) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & f(\mathbf{w}_M \cdot \mathbf{x}_N + b_M) \end{pmatrix} \tag{4}$$

and $\boldsymbol{\beta} = (\beta_1 \ldots \beta_M)^T$. With these notations, the theorem presented in [2] states that with randomly initialized input weights and biases for the SLFN, and under the condition that the activation function f is infinitely differentiable, then the hidden layer output matrix can be determined and will provide an approximation of the target values as good as desired (non-zero).

Algorithm 1. Standard ELM.

```
Given a training set $(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, an activation function
  $f : \mathbb{R} \mapsto \mathbb{R}$ and $M$ hidden nodes:
1. Randomly assign input weights $\mathbf{w}_i$ and biases $b_i$, $i \in [1, M]$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate output weights matrix $\beta = \mathbf{H}^\dagger \mathbf{Y}$.
```

The proposed solution to the equation $\mathbf{H}\beta = \mathbf{Y}$ in the ELM algorithm, as $\beta = \mathbf{H}^\dagger \mathbf{Y}$ has three main properties making it a rather appealing solution:

1. It is one of the least-squares solutions to the mentioned equation, hence the minimum training error can be reached with this solution.
2. Among the least-squares solutions, it is the solution with the smallest norm.
3. This smallest norm solution among the least-squares solutions is unique and is $\beta = \mathbf{H}^\dagger \mathbf{Y}$.

The reason why the smallest norm solution is preferred, is because smaller norm solutions tend to have better generalization performance, as discussed in [9]. Theoretical proofs and a more thorough presentation of the ELM algorithm are detailed in the original paper in which Huang et al. present the algorithm and its justifications [2]. Furthermore, the hidden nodes need not be 'neuron-alike' [10–12].

Finally, it is recommended to have a bias in the output layer (e.g. achieved by concatenating the $\mathbf{H}$ matrix with a column of ones). Although this output bias is often not included in the description of the ELM (since theoretically it is not needed), having the output bias allows the ELM to adapt to any non-zero mean in the output at the expense of only a single extra parameter, namely the extra output weight. This way, the rest of the nonlinear weights can focus on fitting the nonlinear part of the problem. In a different context of deep learning [13], decomposing the problem into a linear part and a nonlinear part has proven to be very effective.

Given a set of candidate neurons, what remains is optimizing the ELM's other parameters like the subset of M neurons to use or the regularization parameter. Approaches for picking a subset of M neurons include model structure selection using an information criterion like BIC and cross-validation using a criterion like the leave-one-out error (described in the next section). Other approaches include methods which first generate a larger than needed set of neurons, and consequently prune this set of neurons (for example OP-ELM [3], TROP-ELM [14]), or incremental ways for determining a set of hidden layer neurons (for example I-ELM [12], CI-ELM [10], EM-ELM [11]).

An optimization mechanism that is orthogonal to optimizing the subset of neurons is that of batch intrinsic plasticity (BIP) pretraining (see Section 2.5), which is a method for optimizing the output distribution of a given neuron, such that the amount of information encoded about the inputs is maximized. Also, the proposed binary and ternary weight schemes (see Section 3) can be considered as orthogonal to optimizing the subset of neurons, since – like batch intrinsic plasticity pretraining – it takes place before the training and optimization of ELM. Therefore, both BIP and the proposed binary and ternary weight schemes can be applied as a step in many different ELM variants, and are not restricted to a particular ELM.