Expert Systems With Applications 83 (2017) 177–186

Contents lists available at ScienceDirect

Expert Systems With Applications

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

On the comparison of random and Hebbian weights for the training of single-hidden layer feedforward neural networks

Kaveh Samiee*, Alexandros Iosifidis, Moncef Gabbouj

Department of Signal Processing, Tampere University of Technology, Tampere, Finland

ARTICLE INFO

Article history:
Received 3 January 2017
Revised 28 March 2017
Accepted 13 April 2017
Available online 25 April 2017

Keywords:
Hebbian learning
Oja's rule
Random feature space
Graph embedding
SLFN

ABSTRACT

In this paper, we provide an experimental study for two unsupervised processes, namely, the random initialization and the Hebbian learning, which can be used to determine the input weights in Single-hidden Layer Feedforward Neural Networks (SLFNs). In addition, a fusion technique that combines the two feature spaces is proposed. Experiments are conducted on six publicly available facial image datasets. Experimental results show that the proposed fusion technique can improve the performance of Hebbian and random feature spaces when they achieve similar performance. In the cases where the difference in performance of the two feature spaces is high, the proposed fusion scheme preserves the power of the most discriminating one and outperforms the average fused feature space. The experimental results show that there is a trade-off between the generalization of the Hebbian feature space and the low computational cost of the random one.

1. INTRODUCTION

Single-hidden layer feedforward neural networks (SLFNs) have attracted much attention and have been successfully applied in many data mining techniques such as regression (Wang, Er, & Han, 2015), classification (Huang, Zhou, Ding, & Zhang, 2012) and subspace learning (Iosifidis, 2015). This is mainly due to their ability to learn an arbitrary continuous function and classify disjoint regions, in addition to their good generalization ability in unseen data. Several effective ways have been proposed for training such neural networks, with Backpropagation (BP) (Rumelhart, Hinton, & Williams, 1986) being the most widely used so far. According to this approach, the network's parameters are randomly initialized and an iterative process is applied to update them by backpropagating the prediction of the network error on a set of training samples followed by the corresponding target vectors.

Another approach that has been proposed in early 1990's (Broomhead & Lowe, 1988; Chen, 1996; Chen, Cowan, & Grant, 1991; Park & Sandberg, 1991; Schmidt, Kraaijveld, & Duin, 1992) and has rejuvenated in the middle 2000's (Huang, Zhu, & Siew, 2004), under the term Extreme Learning Machine (ELM), sets the assumption that the learning processes adopted for the determination of the hidden layer and the output weights need not be connected. In addition, it is assumed that the network's hidden layer neurons can be randomly assigned, this way defining a random (nonlinear) mapping of the input space to a new (usually high-dimensional) feature space. By using a large number of (independent) hidden layer weights, it is expected that the problem to be solved is mapped to a linear problem in the new feature space and, thus, linear techniques such as the mean square estimation can be employed for the determination of the network's output weights (Lin, Liu, Fang, & Xu, 2015; Liu, Lin, Fang, & Xu, 2015; Schmidt et al., 1992; Schwenker, Kestler, & Palm, 2001). The fact that the network's hidden and output weights are determined independently has a number of advantages that can be exploited e.g. for facilitating parallel/distributed systems implementation (Aziz, 2014). In addition, it has been shown that it can provide a good performance in many medium-scale classification problems (Huang et al., 2012; Iosifidis, Tefas, & Pitas, 2015a).

The main limitation in ELM networks is due to the fact that they require a large number of neurons in the hidden layer (which is usually comparable to the number of training data) to achieve a good performance (Iosifidis, Tefas, & Pitas, 2017). In fact, by increasing the number of hidden neurons, the performance of ELMs can significantly improve whilst its generalization ability is weakened due to over-fitting. Furthermore, kernel versions of ELM networks can be obtained by letting the number of hidden layer neurons go to infinity (Iosifidis, Tefas, & Pitas, 2015b; Williams, 1998). While this approach can be adopted in small and medium-scale problems, it is infeasible in large-scale problems, due to dramatic growth in the computational cost. On the other hand, the adop-

* Corresponding author.
E-mail addresses: kaveh.samiee@student.tut.fi (K. Samiee), alexandros.iosifidis@tut.fi (A. Iosifidis), moncef.gabbouj@tut.fi (M. Gabbouj).

http://dx.doi.org/10.1016/j.eswa.2017.04.025
0957-4174/

tion of a relatively small number of hidden layer weights that can summarize/highlight the characteristics of the training data can overcome this limitation. This can be done e.g. by applying clustering on the training samples in order to determine the center vectors of Radial Basis Function (RBF) networks (Schwenker et al., 2001), or by applying eigen decomposition to the kernel matrix formed by the training data and keeping the eigenvectors corresponding to the largest eigenvectors (Iosifidis et al., 2015b). Additionally, as it was shown in (Kannappan, Tamilarasi, & Papageorgiou, 2011; Papakostas, Koulouriotis, Polydoros, & Tourassis, 2012), linear and non-linear Hebbian learning methods can be employed in the training of Fuzzy Cognitive Maps (FCM) classifiers. Similarly, Hebbian learning can be used for unsupervised training of neural networks (Hebb, 2002) or as an alternative to Principal Component Analysis (PCA) for feature selection (Seret, Maldonado, & Baesens, 2015).

In this paper, we provide an extensive experimental study that uses both random and unsupervised learning approaches for the initialization of the hidden layer weights of a single-hidden layer feedforward neural network. For unsupervised learning of the hidden layer weights, we employ Hebbian learning algorithms that are able to reveal the principal subspace of the training data. We observe that both approaches have their advantages and disadvantages. Random hidden layer weights initialization is cheap, but the number of hidden layer neurons required for achieving good performance is high. On the other hand, Hebbian learning has a higher training computational cost, but the number of neurons needed to achieve a satisfactory performance is usually smaller than the random selection approach. As described in (Iosifidis et al., 2015a), the determination of the network output weights in ELM networks can be considered as a subspace learning problem. We therefore formulate the network's output weights learning problem as a bi-fold subspace learning problem, where the goal is to determine the optimal linear combination of the data representations in the two feature spaces, namely, those obtained by applying random and Hebbian hidden layer weights initialization.

In summary, the contributions of this paper are:

- We provide an extensive experimental study comparing random selection and biologically inspired unsupervised learning approaches for the determination of the input weights in SLFN networks. To the best of our knowledge, such a direct comparison of these two approaches is missing from the related literature.
- We propose a new fusion scheme for combining these two approaches that leads to more robust solutions, when compared to each of them independently.

The remainder of the paper is structured as follows. In Section 2 we briefly describe SLFN networks, random-SLFN networks, Hebbian learning and Hebbian-SLFN networks. In Section 3, we describe a novel fusion scheme that is able to determine an optimal linear combination of the two learning approaches. Section 4 describes the experiments conducted and Section 5 concludes the paper.

2. SLFN networks

Let us denote by $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, \ldots, N$, a set of training samples followed by the corresponding target vectors $\mathbf{t}_i \in \mathbb{R}^C$. For classification problems formed by C classes, the elements of the target vectors are set equal to $t_{ij} = 1$ if $\mathbf{x}_i$ belongs to class j and $t_{ij} = -1$ if $\mathbf{x}_i$ belongs to a class $k \neq j$ (Iosifidis, 2015).

A SLFN network that can be used in order to learn the mapping function $\mathbf{x}_i \rightarrow \mathbf{t}_i$ is formed by D input neurons (D is equal to the dimensionality of the input data), L hidden neurons and C output neurons (C is equal to the dimensionality of the target vectors) neurons. The number L of hidden layer neurons is a parameter of the network. Given an activation function $g(\mathbf{w}_j, b_j, \mathbf{x}_i)$ for the network's hidden layer neurons, where $\mathbf{w}_j \in \mathbb{R}^D$ and $b_j \in \mathbb{R}$ denote the weight vector and the bias value of the jth hidden neuron respectively, and using a linear activation function for the network's output layer neurons, the output of the network for an input vector $\mathbf{x}_i$ is given by $\mathbf{o}_i = [o_{i1}, \ldots, o_{iC}]^T$, where:

$$o_{ij} = \mathbf{v}_j^T \mathbf{h}_i + \beta_j, \quad j = 1, \ldots, C, \tag{1}$$

where $\mathbf{v}_j \in \mathbb{R}^L$ and β_j are the jth output weight and the corresponding bias value and $\mathbf{h}_i \in \mathbb{R}^L$ is the representation of $\mathbf{x}_i$ in the feature space determined by the network's hidden layer outputs:

$$\mathbf{h}_i = [g(\mathbf{w}_1, b_1, \mathbf{x}_i), \ldots, g(\mathbf{w}_L, b_L, \mathbf{x}_i)]^T. \tag{2}$$

Theoretically, a conventional SLFN network with L hidden neurons is capable of learning L distinct observations in the training set. In such a SLFN network, input weights and the bias of the hidden neurons may be chosen randomly (Huang, Chen, & Siew, 2006; Lin et al., 2015; Liu et al., 2015). The network's output weights can be also estimated using a gradient-descent method or Least Mean Squares (LMS) algorithm.

2.1. Random-SLFN networks

Random-SLFN is a generalized form of SLFNs in which randomly generated weights of the hidden layer are not tuned (Huang et al., 2004). In fact, the output of the hidden layer in random-SLFNs can be considered as the random projection of the input followed by a nonlinearity. By exploiting a large number of hidden layer neurons, the random selection of the hidden layer weights corresponds to a nonlinear mapping of the input space to a high-dimensional feature space (the so-called ELM space).

After obtaining the data representations in this high-dimensional space, the weights between the hidden and the output layers can be found by a linear solution such as least square estimation. That is, by assuming that the network outputs for the training data are equal to the corresponding target vectors, i.e. $\mathbf{o}_i = \mathbf{t}_i$ and that the output layer bias values β_j are equal to zero, the network output weights $\mathbf{V} \in \mathbb{R}^{L \times C}$ are calculated by:

$$\mathbf{V} = \mathbf{H}^{\dagger} \mathbf{T}^T, \tag{3}$$

where $\mathbf{H} \in \mathbb{R}^{L \times N}$ is a matrix containing the training data representations in the random space, $\mathbf{T} \in \mathbb{R}^{C \times N}$ is a matrix containing the target vectors and $\mathbf{H}^{\dagger} = (\mathbf{H}\mathbf{H}^T)^{-1}\mathbf{H}$ is the Moore-Penrose pseudo-inverse of $\mathbf{H}$. Due to the fact that matrix $\mathbf{H}\dagger$ cannot be sometimes accurately calculated (when $L > N$) and in order to avoid overfitting, a regularization term can be added in (3) (Deng, Zheng, & Chen, 2009), i.e.:

$$\mathbf{V} = \left(\frac{1}{c}\mathbf{I} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{H}\mathbf{T}^T \tag{4}$$

2.2. Principal subspace learning and Hebbian rule

The synapses between input and hidden layers in random-SLFNs are defined as (randomly determined) static weights. Though this assumption can lead to a faster and usually less costly initialization process and it has been claimed that it is biologically plausible (Huang, 2014), it requires a large number of (independent) hidden layer weights. Other biologically plausible learning approaches concerning dynamical cognitive systems state that synapses plasticity plays the main role in learning by adjusting the weights over time. The first hypothesis about the synapses plasticity was proposed by Hebb (2002) and states that the weight between neurons A and B must get strengthened if A participates in