Applied Soft Computing xxx (2018) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

# Letter: On non-iterative learning algorithms with closed-form solution

Ponnuthurai Nagaratnam Suganthan

*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore*

ARTICLE INFO

*Article history:*
Received 1 April 2018
Received in revised form 24 June 2018
Accepted 5 July 2018
Available online xxx

*Keywords:*
Kernel ridge regression
Random vector functional link
Extreme learning machines
Randomized neural networks
Non-iterative learning

ABSTRACT

This letter discusses non-iterative learning methods with closed-form solution such as the kernel ridge regression and randomization based single hidden layer feedforward neural networks like random vector functional link (RVFL). Similarities and differences between these methods are also discussed. Irrelevance of kernel-trick for randomized neural networks is explained. The need for dual formulation or constrained optimization formulation for kernel methods and RVFL is distinguished. Finally, the articles in this special issue focusing on non-iterative learning methods with closed-form solution are summarized. A common conclusion in these articles is that the RVFL developed in the early 1990s outperforms the extreme learning machines (ELM). This conclusion is consistent with the earlier findings [1–3] that the direct links enhance the performance of the RVFL.



## 1. Introduction

There are several motivations for writing this letter. During the review process of the special issue[1] some authors exhibited confusions between kernel methods, neural networks, kernel functions, activation functions, kernel tricks and so on. In addition, authors also attempted to propose new names for existing methods, for example, constrained optimization formulation or dual formulation for RVFL in order to develop kernel RVFL. In fact, the final solution of kernel RVFL is identical to the kernel ridge regression (KRR) solution developed in the late 1990s. Hence, it was felt that these issues can be documented for the benefits of the research community. Another motivation is to summarize the special issue articles with closed form solutions as they are closely related and several of these articles also compare methods such as RVFL, ELM, KRR, kernel ELM, and so on.

The major variation between RVFL and ELM is the presence and absence of direct connections between inputs and outputs, respectively. For the first time, the importance of direct connections between inputs and outputs was demonstrated experimentally in [1–3]. Some of the special issue articles [4–7] also compared RVFL and ELM and arrived at the same conclusion.

This letter is organized as follows. In the next two sections original KRR and RVFL are introduced. Subsequently, issues such as kernel functions, activation functions, kernel tricks, dual formulation (i.e. constrained optimization formulation) are discussed. Finally the special issue articles dealing with these issues are summarized.

## 2. Kernel ridge regression (KRR)

Ridge regression (RR) is a popular machine learning algorithm based on least squares. A ridge or regularization term is added to the least squares learning objective to mitigate the over-fitting (i.e. the model-complexity). The ridge regression is formulated as:

$$\min_{w} \left[ \sum_{i} (x_i w - y_i)^2 + \lambda \|w\|^2 \right] \tag{1}$$

where $\mathbf{X} = \left[x_1^T, x_2^T, \ldots, x_n^T\right]^T$ is an $n \times d$ data matrix, $Y = [y_1, y_2, \ldots, y_n]^T$ is an $n \times 1$ output vector and $w$ is a $d \times 1$ weight vector in the single output configuration. The regularization parameter $\lambda$ is generally determined through cross-validation. The above learning objective has a closed form solution given by:

$$w = (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^T Y \tag{2}$$

where $\mathbf{I}$ is an identity matrix. Saunders et al. [8] proposed to enhance the ridge regression to perform non-linear regression by applying the kernel trick. According to the representer Theorem,

*E-mail address:* epnsugan@ntu.edu.sg

[1] Applied Soft Computing journal's special issue on 'Non-iterative Approaches in Learning' guest edited by Dr P. N. Suganthan, Prof. Sushmita Mitra and Dr Ivan Tyukin with September 2016 as the paper submission deadline.

https://doi.org/10.1016/j.asoc.2018.07.013
1568-4946/

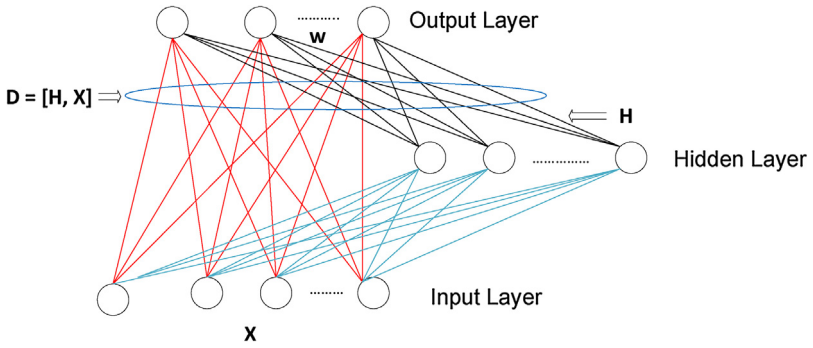


**Fig. 1.** The structure of RVFL.

one can express the solution of $w$ as a linear combination of the samples in the feature space $\phi(x)$ as $w = \sum_i \alpha_i \phi(x_i)$. The learning objective is then rewritten as:

$$\min_{\alpha} \|Y - \mathbf{K}\alpha\|^2 + \lambda \alpha^T \mathbf{K} \alpha \tag{3}$$

Similarly, its closed form solution is:

$$\alpha = (\mathbf{K} + \lambda \mathbf{I})^{-1} Y \tag{4}$$

where $\mathbf{K}$ is a kernel matrix and $\mathbf{K}_{ij} = k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. Thus, instead of explicitly computing the coordinates of the samples in feature space through $\phi(x)$, one can simply use the kernel trick with kernel functions such as Gaussian kernel, linear kernel, polynomial kernel and so on.

The kernel ridge regression (KRR) can be used to solve classification problems by defining the output vector $Y$ with 0-1 coding [9]:

$$Y_{ij} = \begin{cases} 1 & \text{if } i\text{th sample belongs to } j\text{th class} \\ 0 & \text{otherwise} \end{cases}$$

Multiple outputs would be required for multi-class classification with 0-1 coding. KRR has demonstrated highly competitive performances in a recent extensive benchmarking studies [10].

## 3. Random vector functional link network (RVFL)

Neural networks are commonly used for function approximation. These networks are usually trained by minimizing a loss function while the errors are back-propagated to determine the suitable network parameters by using gradient-based methods. The problems with back-propagation based optimization methods are slow training process, the solution may not converge to a single global minimum if there exists many local minima and it is also sensitive to learning rate setting. Such issues can be circumvented by randomization based methods, for example, randomly fixing the network configurations such as the connections or some parts of the network parameters, or randomly corrupting the input data or the parameters during the training [11–17]. Among these, the random vector functional link network (RVFL) [11] shown in Fig. 1 performs better [1,2].

RVFL is a single layer feed-forward neural network in which weights and biases of the hidden neurons are randomly generated within a suitable range. The inputs $\mathbf{D}$ of output neurons in RVFL consist of non-linearly transformed features $\mathbf{H}$ from hidden layer neurons and original input features $\mathbf{X}$ directly from input layer. Suppose that the input data has $d$ features and there are $J$ hidden neurons, there are total $d+J$ inputs for each output node. The direct links greatly improve the performance of RVFL networks [1,2]. Since the hidden layer parameters are randomly generated and kept fixed, the learning objective is reduced to computing output weights, $w$, only. The learning objective is as follows:

$$\min_{w} \|\mathbf{D}w - Y\|^2 + \lambda \|w\|^2 \tag{5}$$

A closed form solution can be obtained by using either regularized least squares (i.e. $\lambda \neq 0$) or Moore-Penrose pseudoinverse (i.e. $\lambda = 0$). Using Moore-Penrose pseudoinverse, the solution is given by: $w = \mathbf{D}^+ Y$ while using the regularized least squares (or ridge regression), the closed form solution is given by:

$$\text{Primal Space:} \quad w = (\mathbf{D}^T\mathbf{D} + \lambda \mathbf{I})^{-1} \mathbf{D}^T Y \tag{6}$$

We can transform the above primal space solution to the dual space solution as follows:

$$(\mathbf{D}^T\mathbf{D} + \lambda \mathbf{I})\mathbf{D}^T = \mathbf{D}^T(\mathbf{D}\mathbf{D}^T + \lambda \mathbf{I}) \tag{7}$$

$$(\mathbf{D}^T\mathbf{D} + \lambda \mathbf{I})^{-1}(\mathbf{D}^T\mathbf{D} + \lambda \mathbf{I})\mathbf{D}^T(\mathbf{D}\mathbf{D}^T + \lambda \mathbf{I})^{-1} Y = (\mathbf{D}^T\mathbf{D} + \lambda \mathbf{I})^{-1}\mathbf{D}^T(\mathbf{D}\mathbf{D}^T + \lambda \mathbf{I})(\mathbf{D}\mathbf{D}^T + \lambda \mathbf{I})^{-1} Y \tag{8}$$

$$\mathbf{D}^T(\mathbf{D}\mathbf{D}^T + \lambda \mathbf{I})^{-1} Y = (\mathbf{D}^T\mathbf{D} + \lambda \mathbf{I})^{-1}\mathbf{D}^T Y \tag{9}$$

$$\mathbf{D}^T(\mathbf{D}\mathbf{D}^T + \lambda \mathbf{I})^{-1} Y = \quad w \tag{10}$$

Hence, the output can be expressed as follows for the case of dual space:

$$\text{Dual Space:} \quad w = \mathbf{D}^T(\mathbf{D}\mathbf{D}^T + \lambda \mathbf{I})^{-1} Y \tag{11}$$

Depending on the number of training samples or total feature dimensions (i.e. input features plus total number of hidden neurons), dual or primal solution can be used to reduce the complexity of the matrix inversion.

## 4. Kernel function versus activation function

The function $k(x_i, x_j)$ is known as an inner-product kernel or simply a kernel. It computes the inner product of the images under an embedding $\phi$ of two data points. One can thus, create a Kernel or Gram matrix $\mathbf{K}$ given a kernel and a training set. A valid $\mathbf{K}$ is positive semi-definite ($v^T\mathbf{K}v \geq 0$) implying all its eigenvalues are positive and it is symmetric $\mathbf{K}_{ij} = \mathbf{K}_{ji}$. Hence, a valid kernel is also called Mercer kernel. The Kernel matrix contains all the information required for the learning procedure except the target or class labels. One can simply transform the input data into a higher dimensional feature space using kernel trick to potentially better discriminate the data samples.

An activation function maps a linearly weighted summation of inputs to an output in a neural network. These bounded functions, $f(\cdot)$, are generally used in the hidden and output layers of neural networks to mimic the firing of biological neurons. Some of the

Please cite this article in press as: P.N. Suganthan, Letter: On non-iterative learning algorithms with closed-form solution, Appl. Soft Comput. J. (2018), https://doi.org/10.1016/j.asoc.2018.07.013