Cogn Comput
DOI 10.1007/s12559-014-9255-2

An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels

Guang-Bin Huang

Received: 25 January 2014 / Accepted: 13 March 2014

Abstract Extreme learning machines (ELMs) basically give answers to two fundamental learning problems: (1) Can fundamentals of learning (i.e., feature learning, clustering, regression and classification) be made without tuning hidden neurons (including biological neurons) even when the output shapes and function modeling of these neurons are unknown? (2) Does there exist unified framework for feedforward neural networks and feature space methods? ELMs that have built some tangible links between machine learning techniques and biological learning mechanisms have recently attracted increasing attention of researchers in widespread research areas. This paper provides an insight into ELMs in three aspects, viz: random neurons, random features and kernels. This paper also shows that in theory ELMs (with the same kernels) tend to outperform support vector machine and its variants in both regression and classification applications with much easier implementation.

Keywords Extreme learning machine · Support vector machine · Least square support vector machine · ELM kernel · Random neuron · Random feature · Randomized matrix

Introduction

Support vector machine (SVM) [1] as a revolutionary machine learning technique has been playing an important role in both research and classification related applications in the past two decades. SVM achieves higher generalization performance than conventional artificial neural networks in most classification applications. In the original implementation of SVM, one has to handle a quadratic programming (QP) problem which is usually tedious and time consuming. As one of the main variant of SVM, Least square support vector machine (LS-SVM) [2] aims to avoid the QP problem using equality constraints instead of the inequality constraint adopted in conventional SVM. Compared with SVM, LS-SVM is ease of implementation. Extreme learning machines (ELMs) [3–7] become attractive to more and more researchers recently [8–20]. This paper aims to review the ELM from random neurons and kernels point of view and to build some relationship and links between ELM, SVM and other related machine learning techniques. Although it is out of question that SVM and its variants achieve surprising performance in most applications, different from some common concept in the research community SVM and LS-SVM may perhaps tend to achieve suboptimality in classification applications due to some optimization constraints. This may be true to some (if not most) of their variants as well.

G.-B. Huang (✉)
School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore
e-mail: egbhuang@ntu.edu.sg

Brief of SVM and LS-SVM

This section briefs the conventional SVM [1] and one of its main variants LS-SVM [2].

SVM

Given a set of training data $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbf{R}^d$ and $t_i \in \{-1, 1\}$, SVM aims to maximize the separating margin

Published online: 03 April 2014

of the two different classes as well as to minimize the training errors ξ_i, which is equivalent to:

$$\begin{aligned} \text{Minimize}: \quad & L_{PSVM} = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ \text{Subject to}: \quad & t_i(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)+b) \geq 1-\xi_i, \forall i \\ & \xi_i \geq 0, \forall i \end{aligned} \tag{1}$$

where C is a user specified parameter and provides a tradeoff between minimizing the training error and maximizing the distance $2/\|\mathbf{w}\|$ of the separating margin of the two different classes in the feature space; $\boldsymbol{\phi}: \mathbf{x}_i \to \boldsymbol{\phi}(\mathbf{x}_i)$ is a nonlinear mapping which maps the training data $\mathbf{x}_i$ from the input space to a feature space.

The corresponding Lagrange function of the primal SVM optimization (1) is:

$$\begin{aligned} L_{\mathrm{SVM}} = & \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\mu_i\xi_i \\ & - \sum_{i=1}^{N}\alpha_i\big(t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b)-(1-\xi_i)\big) \end{aligned} \tag{2}$$

Based on the Karush–Kuhn–Tucker (KKT) theorem [21], in order to find the optimal solutions of (2), we should have:

$$\frac{\partial L_{\mathrm{SVM}}}{\partial \mathbf{w}} = 0 \Longrightarrow \mathbf{w} = \sum_{i=1}^{N}\alpha_i t_i h(\mathbf{x}_i) \tag{3a}$$

$$\frac{\partial L_{\mathrm{SVM}}}{\partial \boldsymbol{\xi}} = 0 \Longrightarrow C = \alpha_i + \mu_i, \quad \forall i \tag{3b}$$

$$\frac{\partial L_{\mathrm{SVM}}}{\partial b} = 0 \Longrightarrow \sum_{i=1}^{N}\alpha_i t_i = 0 \tag{3c}$$

Substitute (3a)–(3c) into (2), therefore, to train such an SVM is equivalent to solving the following dual optimization problem:

$$\begin{aligned} \text{minimize}: & L_{D_{\mathrm{SVM}}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \boldsymbol{\phi}(\mathbf{x}_i)\cdot\boldsymbol{\phi}(\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ \text{subject to}: & \sum_{i=1}^{N}\alpha_i t_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad \forall i \end{aligned} \tag{4}$$

where each Lagrange multiplier α_i corresponds to a training sample $(\mathbf{x}_i, t_i)$. In SVM, the feature mapping $\boldsymbol{\phi}$ is usually unknown, kernel function $K(\mathbf{u},\mathbf{v}) = \boldsymbol{\phi}(\mathbf{u})\cdot\boldsymbol{\phi}(\mathbf{v})$ can be used instead. In this case, the corresponding dual optimization problem is:

$$\begin{aligned} \text{minimize}: & L_{D_{\mathrm{SVM}}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ \text{subject to}: & \sum_{i=1}^{N}\alpha_i t_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad \forall i \end{aligned} \tag{5}$$

In addition to the popular SVM cost function (1), in their classical paper on SVM [1], Cortes and Vapnik originally also proposed a more general SVM cost function [1]:

$$\begin{aligned} \text{Minimize}: \quad & L_{P_{\mathrm{SVM}}} = \frac{1}{2}\left(\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{N}\xi_i^{\sigma}\right) \\ \text{Subject to}: \quad & t_i(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)+b) \geq 1-\xi_i, \forall i \\ & \xi_i \geq 0, \forall i \end{aligned} \tag{6}$$

where $\sigma > 0$. SVM is more well known for the case where $\sigma = 1$.

LS-SVM

Least square support vector machine [2] targets at avoiding the QP problem (5) faced by classical SVM. Instead of the inequality constraints (1) adopted in the classical SVM, equality constraints are used in LS-SVM [2]. In LS-SVM, the classification problem is formulated as:

$$\begin{aligned} & \text{Minimize}: \mathrm{L}_{\mathrm{P_{LS-SVM}}} = \frac{1}{2}\left(\mathbf{w}\cdot\mathbf{w} + \mathrm{C}\sum_{\mathrm{i}=1}^{\mathrm{N}}\xi_{\mathrm{i}}^2\right) \\ & \text{Subject to}: \mathrm{t_i}(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_{\mathrm{i}})+\mathrm{b}) = 1-\xi_{\mathrm{i}}, \forall \mathrm{i} \end{aligned} \tag{7}$$

Based on the KKT theorem, to train such a LS-SVM is equivalent to solving the following dual optimization problem:

$$\begin{aligned} L_{\mathrm{LS-SVM}} = & \frac{1}{2}\left(\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{N}\xi_i^2\right) \\ & - \sum_{i=1}^{N}\alpha_i(t_i(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)+b)-1+\xi_i) \end{aligned} \tag{8}$$

Based on the KKT theorem, the following optimality conditions of (8) should be satisfied:

$$\frac{\partial L_{\mathrm{LS-SVM}}}{\partial \mathbf{w}} = 0 \to \mathbf{w} = \sum_{i=1}^{N}\alpha_i t_i \boldsymbol{\phi}(\mathbf{x}_i) \tag{9a}$$

$$\frac{\partial L_{\mathrm{LS-SVM}}}{\partial b} = 0 \to \sum_{i=1}^{N}\alpha_i t_i = 0 \tag{9b}$$

$$\frac{\partial L_{\mathrm{LS-SVM}}}{\partial \xi_i} = 0 \to \alpha_i = C\xi_i, \forall i \tag{9c}$$

$$\frac{\partial L_{\mathrm{LS-SVM}}}{\partial \alpha_i} = 0 \rightarrow t_i(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b) - 1 + \xi_i = 0, \quad \forall i \tag{9d}$$

Furthermore, the above equations can be equivalently written as:

$$\begin{bmatrix} \mathbf{0} & \mathbf{T}^T \\ \mathbf{T} & \frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\mathrm{LS-SVM}} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{T}^T \\ \mathbf{T} & \frac{\mathbf{I}}{C} + \mathbf{Z}\mathbf{Z}^T \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vec{\mathbf{1}} \end{bmatrix} \tag{10}$$

where

$$\mathbf{Z} = \begin{bmatrix} t_1 \boldsymbol{\phi}(\mathbf{x}_1) \\ \vdots \\ t_N \boldsymbol{\phi}(\mathbf{x}_N) \end{bmatrix} \qquad \boldsymbol{\Omega}_{\mathrm{LS-SVM}} = \mathbf{Z}\mathbf{Z}^T \tag{11}$$

The feature mapping $\boldsymbol{\phi}(\mathbf{x})$ is a row vector, $\mathbf{T} = [t_1, t_2, \cdots, t_N]^T$, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_N]^T$ and $\vec{\mathbf{1}} = [1, 1, \cdots, 1]^T$. For kernel function $K(\mathbf{u}, \mathbf{v}) = \boldsymbol{\phi}(\mathbf{u}) \cdot \boldsymbol{\phi}(\mathbf{v})$, we have matrix $\boldsymbol{\Omega}_{\mathrm{LS-SVM}}$:

$$\Omega_{\mathrm{LS-SVM}\,i,j} = t_i t_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) = t_i t_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{12}$$

Review of ELM

Since its inception, back-propogation (BP) learning algorithm [22–25] and its variants have been playing dominant roles in training feedforward neural networks. It is well known that BP learning algorithm and its variants face several challenging issues such as local minima, trivial human intervention and time consuming in learning. Although thousands of researchers (from almost all universities in the world) had spent almost twenty years (1986–2005) on studying feedforward neural networks, there has no much significant progress in finding efficient learning algorithms for feedforward neural network except for mainly being focusing on BP algorithms and its variants. In some sense, the main research stream itself in feedforward neural networks somehow has been stuck in "local minima" as BP algorithm does. SVM as an alternative solution to training feedforward neural networks was proposed in 1995 and became popular since the early of this century (around 2004) when some researchers may have started losing confidence on artificial neural networks.

ELM [3–7] was originally inspired by biological learning and proposed to overcome the challenging issues faced by BP learning algorithms. Brain learning is sophisticated; however, brain usually works universally for feature extraction, clustering, regression, classification and requires zero human intervention (in tuning "user specified parameters") and almost zero time in learning given particular samples in many cases. Inspired by these biological learning features, we have conjectured that some part of brain systems should have random neurons with *all* their parameters independent of the environments [3, 5–7, 26], and the resultant technique was referred to ELMs.[1] Its computer-based learning efficiency was verified as early as in 2004 [3], its universal approximation capability was rigorously proved in theory in 2006–2008 [6, 7, 26], and its concrete biological evidence seems to subsequently appear in 2011–2013 [27–30].

Unlike other so-called randomness (semi-randomness)-based learning methods/networks [31], all the hidden nodes in ELM are not only independent of the training data but also independent of each other. Although hidden nodes are important and critical, they need not be tuned and the hidden node parameters can be randomly generated beforehand. Unlike conventional learning methods which MUST see the training data before generating the hidden node parameters, ELM can generate the hidden node parameters before seeing the training data.

Learning Principles

ELM was first proposed for the single-hidden layer feedforward *neural* networks (SLFNs) and was then extended to the generalized single-hidden layer feedforward networks where the hidden layer need not be neuron alike [7, 26, 32].

From the network architecture point of view, the output function of ELM for generalized SLFNs is

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{13}$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_L]^T$ is the vector of the output weights between the hidden layer of L nodes to the $m \geq 1$ output nodes, and $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \cdots, h_L(\mathbf{x})]$ is the output (row) vector of the hidden layer with respect to the input $\mathbf{x}$. $h_i(\mathbf{x})$ is the output of the ith hidden node output, and the output functions of hidden nodes may not be unique. *Different*

[1] Instead of the ambiguous word "randomness" such as in "random features" and "random networks," "Extreme" here means to move beyond conventional artificial learning techniques and to move toward brain alike learning. ELM aims to break the barriers between the conventional artificial learning techniques and biological learning mechanism. "Extreme learning machine (ELM)" represents a suite of machine learning techniques in which hidden neurons need not be tuned. This includes but is not limited to random hidden nodes, it also includes kernels. On the other hand, instead of only considering network architecture such as randomness and kernels, in theory ELM also somehow unifies brain learning features, neural network theory, control theory, matrix theory, and linear system theory which were considered isolated with big gaps before. Details can be found in this paper.

output functions may be used in different hidden neurons. In particular, in real applications, $h_i(\mathbf{x})$ can be

$$h_i(\mathbf{x}) = G(\mathbf{a}_i, b_i, \mathbf{x}), \mathbf{a}_i \in \mathbf{R}^d, b_i \in R \tag{14}$$

where $G(\mathbf{a}, b, \mathbf{x})$ is a nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems [6, 7, 26]. For example, such nonlinear piecewise continuous functions can be but are not limited to:

1. Sigmoid function:

$$G(\mathbf{a}, b, \mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{a}\cdot\mathbf{x} + b))} \tag{15}$$

2. Fourier function[6, 33]:

$$G(\mathbf{a}, b, \mathbf{x}) = \sin(\mathbf{a}\cdot\mathbf{x} + b) \tag{16}$$

3. Hardlimit function [6, 34]:

$$G(\mathbf{a}, b, \mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{a}\cdot\mathbf{x} - \mathrm{b} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

4. Gaussian function [6, 32]:

$$G(\mathbf{a}, b, \mathbf{x}) = \exp(-b\|\mathbf{x} - \mathbf{a}\|^2) \tag{18}$$

5. Multiquadrics function [6, 32]:

$$G(\mathbf{a}, b, \mathbf{x}) = (\|\mathbf{x} - \mathbf{a}\|^2 + b^2)^{1/2} \tag{19}$$

Definition 3.1 A neuron (or node) is called a random neuron (node) if *all* its parameters (e.g., $(\mathbf{a}, b)$ in its output function $G(\mathbf{a}, b, \mathbf{x})$) are randomly generated based on a continuous sampling distribution probability.

$\mathbf{h}(\mathbf{x})$ actually maps the data from the d-dimensional input space to the L-dimensional hidden layer random feature space (*ELM feature space*) where the hidden node parameters are randomly generated according to any continuous sampling distribution probability, and thus, $\mathbf{h}(\mathbf{x})$ is indeed a random feature mapping.

Definition 3.2 [6, 7, 26] A hidden layer output mapping $\mathbf{h}(\mathbf{x})$ is said to be an ELM random feature mapping if all its hidden node parameters are randomly generated according to any continuous sampling distribution probability *and* such $\mathbf{h}(\mathbf{x})$ has universal approximation capability, that is, $\|\mathbf{h}(\mathbf{x})\boldsymbol{\beta} - f(\mathbf{x})\| = \lim_{L\to\infty} \|\sum_{i=1}^{L} \boldsymbol{\beta}_i h_i(\mathbf{x}) - f(\mathbf{x})\| = 0$ holds with probability one with appropriate output weights $\boldsymbol{\beta}$.

According to Bartlett's neural network generalization theory [35], for feedforward neural networks reaching smaller training error, the smaller the norms of weights are, the better generalization performance the networks tend to have. *We conjecture that this may be true to the generalized SLFNs where the hidden neurons may not be neuron alike* [7, 26].

From the learning point of view, unlike traditional learning algorithms [24, 36–42], ELM theory aims to reach the smallest training error but also the smallest norm of output weights [3, 5]:

$$\text{Minimize}: \|\boldsymbol{\beta}\|_{\mathrm{p}}^{\sigma_1} + \mathrm{C}\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_{\mathrm{q}}^{\sigma_2} \tag{20}$$

where $\sigma_1 > 0$, $\sigma_2 > 0$, $p, q = 0, \frac{1}{2}, 1, 2, \cdots, +\infty$, $\mathbf{H}$ is the hidden layer output matrix (*randomized matrix*):

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \vdots & h_L(\mathbf{x}_N) \end{bmatrix} \tag{21}$$

and $\mathbf{T}$ is the training data target matrix:

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \vdots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix} \tag{22}$$

ELM follows some learning rules (or learning principles):

Learning Principle I: Hidden neurons of SLFNs with almost any nonlinear piecewise continuous activation functions or their linear combinations can be randomly generated according to any continuous sampling distribution probability, and such hidden neurons can be independent of training samples and also its learning environment.

According to ELM learning theory, widespread type of feature mappings $\mathbf{h}(\mathbf{x})$ can be used in ELM so that ELM can approximate any continuous target functions (refer to Huang et al. [6, 7, 26, 32] for details). Activation functions such as sigmoid function used in artificial neural networks [40, 43] are oversimplified modeling of some live brain neurons and may be different from the truth. In real brain learning mechanism, the actual activation functions of living brain neurons are unknown.

It may have no way for human beings to know the exact activation function formula of live brain neurons, and individual neurons of same type may have similar but not the exact same activation function as well. Although the actual activation functions of living brain neurons are unknown, most likely the actual activation functions of many brain neurons are nonlinear piecewise continuous. Thus, *Learning Principle I of ELM may be widely adopted in some brain learning mechanism without the need of knowing the actual activation function of living brain neurons.* In fact, we have

Theorem 3.1 *Universal approximation capability* [6, 7, 26]: Given any bounded non-constant piecewise continuous function as the activation function in hidden neurons, if by tuning parameters of hidden neuron activation function SLFNs can approximate any target continuous function, then for any continuous target function $f(\mathbf{x})$ and any randomly generated function sequence $\{h_i(\mathbf{x})\}_{i=1}^{L}$, $\lim_{L\to\infty} \| \sum_{i=1}^{L} \boldsymbol{\beta}_i h_i(\mathbf{x}) - f(\mathbf{x})\| = 0$ holds with probability one with appropriate output weights $\boldsymbol{\beta}$.

In fact, SLFNs with almost any nonlinear piecewise continuous activation functions or their linear combinations not only can approximate any continuous target functions but also separate arbitrary disjoint regions of any shapes with hidden neurons randomly generated independent of training samples. Especially, the following theorem on ELM's classification capability has been proved in Huang et al. [32]:

Theorem 3.2 *Classification capability* [32]: Given a feature mapping $\mathbf{h}(\mathbf{x})$, if $\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ is dense in $C(\mathbf{R}^d)$ or in $C(M)$, where M is a compact set of $\mathbf{R}^d$, then SLFNs with random hidden layer mapping $\mathbf{h}(\mathbf{x})$ can separate arbitrary disjoint regions of any shapes in $\mathbf{R}^d$ or M.

Theorem 3.1 actually implies that if SLFNs with bounded non-constant piecewise continuous functions have universal approximation capability, no particular learning methods are required to adjust the hidden neurons. In other words, *as many (living brain) neurons' activation functions are bounded nonlinear piecewise continuous although their shapes/modeling may be unknown, hidden neurons in SLFNs (some parts of living brains) can be randomly generated without further tuning.*

Learning Principle II: For the sake of system stability and generalization performance, the norm of the output weights of generalized SLFNs need to be smaller with some optimization constraints.

Learning Principle III: *From optimization point of view*, the output nodes of SLFNs should have no biases (or set bias zero).

In contrast to the common understanding (since the inception of "artificial" neural networks) that the output nodes need to have the biases (and such biases may be used in different applications due to different reasons), this learning principle actually shows that from optimization constraints point of view such biases in the output nodes may cause some dilemma. This is also one of the key differences between ELM and other random methods such as Schmidt et al. [40]. Detail analysis will be given in section "Optimization Constraints with Kernels: ELM and SVM/LS-SVM".

Basic ELM

In order to satisfy Learning Principle II, the basic implementation of ELM (when $C = +\infty$) [3, 5] uses the minimal norm least square method instead of the standard optimization method in the solution:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{23}$$

where $\mathbf{H}^{\dagger}$ is the *Moore–Penrose generalized inverse* of matrix $\mathbf{H}$ [44, 45]. Different methods can be used to calculate Moore–Penrose generalized inverse of a matrix: orthogonal projection method, orthogonalization method, iterative method and singular value decomposition (SVD) [45]. The output weights $\boldsymbol{\beta}$ can also be obtained through other iterative methods [46, 47]. Unlike Schmidt et al. [40] which applies sigmoid function in the hidden layer, almost any nonlinear activation function can be used in this ELM implementation (23) and the hidden nodes need not be additive type or radial basis function (RBF) type. For example, it is challenging to provide direct solutions to threshold networks, and many researchers spent much effort in such investigation[48–51]. In fact, it seems that there was no direct solution on threshold networks in the past twenty years (around 1986-2005) until when ELM provides the direct solution [34].

Fernández-Delgado [46] also provides an alternative solution of ELM when $C = 0$.

Different type of constraints which are application dependent may exist for the optimization objective function (20). In order to compare with SVM and LS-SVM, two specific type of optimization constraints of ELM ($\sigma_1 = p = 2$) are especially considered as follows. It should be noted that:

(1) SVM and LS-SVM may only work with kernels as feature mapping $\boldsymbol{\phi}(\mathbf{x})$ in SVM and LS-SVM is unknown.
(2) ELM can work with kernels if $\mathbf{h}(\mathbf{x})$ is unknown.
(3) ELM can also work with ELM kernels or non-kernels if $\mathbf{h}(\mathbf{x})$ is known.

Inequality Optimization Constraints Based ELM

Consider $\sigma_1 = 2$, $\sigma_2 = 1$, $p = 2$, and $q = 1$ in ELM optimization formula (20) for binary classification case first ($m = 1$). From the standard optimization theory point of view, the objective of ELM in minimizing both the training errors ξ_i and the output weights $\boldsymbol{\beta}$ can be written as [52, 53]:

$$\begin{aligned}&\text{Minimize}: L_{P_{\text{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{Subject to}: t_i\boldsymbol{\beta}\cdot\mathbf{h}(\mathbf{x}_i) \geq 1-\xi_i, \quad i=1,\cdots,N \\ &\qquad \xi_i \geq 0, \quad i=1,\cdots,N\end{aligned} \tag{24}$$

Based on KKT conditions, to train ELM for binary classification is then equivalent to solving the following dual optimization problem:

$$\begin{aligned}&\text{minimize}: \quad L_{D_{\text{ELM}}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}t_it_j\alpha_i\alpha_jK(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ &\text{subject to}: \qquad 0 \leq \alpha_i \leq C, \quad i=1,\cdots,N\end{aligned} \tag{25}$$

where $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{h}(\mathbf{x}_i)\cdot\mathbf{h}(\mathbf{x}_j)$.

There may exist different ways to finding the Lagrange multipliers of the above-mentioned dual optimization problem (25) [52, 53].

Equality Optimization Constraints Based ELM

Consider $\sigma_1 = \sigma_2 = p = q = 2$ in ELM optimization formula (20) for both regression and (binary or multi-class) classification cases. For instance, we consider the ELM with multi-output nodes. For m-class of applications, ELM has m output nodes ($m > 2$). (Single output nodes can be used in binary classification applications.) If the original class label of the training sample is p, the expected output vector of the m output nodes is $\mathbf{t}_i = [0,...,0,\overset{p}{1},0,\cdots,0]^T$. In this case, only the pth element of $\mathbf{t}_i = [t_{i,1},\cdots,t_{i,m}]^T$ is one while the rest elements are set zero. From the standard optimization theory point of view, the objective of ELM in minimizing both the training errors and the output weights can be written as [32]:

$$\begin{aligned}&\text{Minimize}: \mathrm{L}_{P_{\text{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \mathrm{C}\frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{N}}\|\boldsymbol{\xi}_\mathrm{i}\|^2 \\ &\text{Subject to}: \mathbf{h}(\mathbf{x}_\mathrm{i})\boldsymbol{\beta} = \mathbf{t}_\mathrm{i}^\mathrm{T} - \boldsymbol{\xi}_\mathrm{i}^\mathrm{T}, \ \mathrm{i}=1,\cdots,\mathrm{N}\end{aligned} \tag{26}$$

where $\boldsymbol{\xi}_i = [\xi_{i,1},\cdots,\xi_{i,m}]^T$ is the training error vector of the m output nodes with respect to the training sample $\mathbf{x}_i$. Based on the KKT theorem, to train ELM is equivalent to solving the following dual optimization problem:

$$\begin{aligned}L_{D_{\text{ELM}}} = &\frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 \\ &- \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_{i,j}\left(\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}_j - t_{i,j} + \xi_{i,j}\right)\end{aligned} \tag{27}$$

where $\boldsymbol{\beta}_j$ is the vector of the weights linking the hidden layer to the jth output node, and $\boldsymbol{\beta} = [\boldsymbol{\beta}_1,\cdots,\boldsymbol{\beta}_m]$. We can have the KKT corresponding optimality conditions as follows:

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \boldsymbol{\beta}_j} = 0 \rightarrow \boldsymbol{\beta}_j = \sum_{i=1}^{N}\alpha_{i,j}\mathbf{h}(\mathbf{x}_i)^T \rightarrow \boldsymbol{\beta} = \mathbf{H}^T\boldsymbol{\alpha} \tag{28a}$$

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \boldsymbol{\xi}_i} = 0 \rightarrow \boldsymbol{\alpha}_i = C\boldsymbol{\xi}_i, \mathrm{i}=1,\ldots,\mathrm{N} \tag{28b}$$

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \boldsymbol{\alpha}_i} = 0 \rightarrow \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - \mathbf{t}_i^T + \boldsymbol{\xi}_i^T = 0, \mathrm{i}=1,\ldots,\mathrm{N} \tag{28c}$$

where $\boldsymbol{\alpha}_i = [\alpha_{i,1},\cdots,\alpha_{i,m}]^T$ and $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1,\cdots,\boldsymbol{\alpha}_N]^T$.

The above-mentioned KKT conditions can result in different solutions as follows. Depending on practical applications, users may use other methods such as iterative methods as well.

Kernel case

Substitute Eqs. (28a) and (28b) into Eq. (28c), we have:

$$\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)\boldsymbol{\alpha} = \mathbf{T} \tag{29}$$

From Eqs. (28a) and (29), we have:

$$\boldsymbol{\beta} = \mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \tag{30}$$

The ELM output function of ELM is:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \tag{31}$$

We can define a kernel matrix for ELM as follows:

$$\boldsymbol{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^T : \Omega ELM_{i,j} = \mathbf{h}(\mathbf{x}_i)\cdot\mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i,\mathbf{x}_j) \tag{32}$$

Then, the ELM output function (31) can be as:

$$\begin{aligned}\mathbf{f}(\mathbf{x}) &= \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \\ &= \begin{bmatrix} K(\mathbf{x},\mathbf{x}_1) \\ \vdots \\ K(\mathbf{x},\mathbf{x}_N)\end{bmatrix}^T\left(\frac{\mathbf{I}}{C} + \boldsymbol{\Omega} ELM\right)^{-1}\mathbf{T}\end{aligned} \tag{33}$$

In this specific case, similar to SVM and LS-SVM, the feature mapping $\mathbf{h}(\mathbf{x})$ need not be known to users, instead one may use its corresponding kernel $K(\mathbf{u},\mathbf{v})$ (e.g., $K(\mathbf{u},\mathbf{v}) = \exp(-\gamma\|\mathbf{u}-\mathbf{v}\|^2)$). When $\mathbf{h}(\mathbf{x})$ is randomly generated, we call $K(\mathbf{x}_i,\mathbf{x}_\mathrm{j}) = \mathbf{h}(\mathbf{x}_\mathrm{i})\cdot\mathbf{h}(\mathbf{x}_\mathrm{j})$ as ELM (random) kernel.

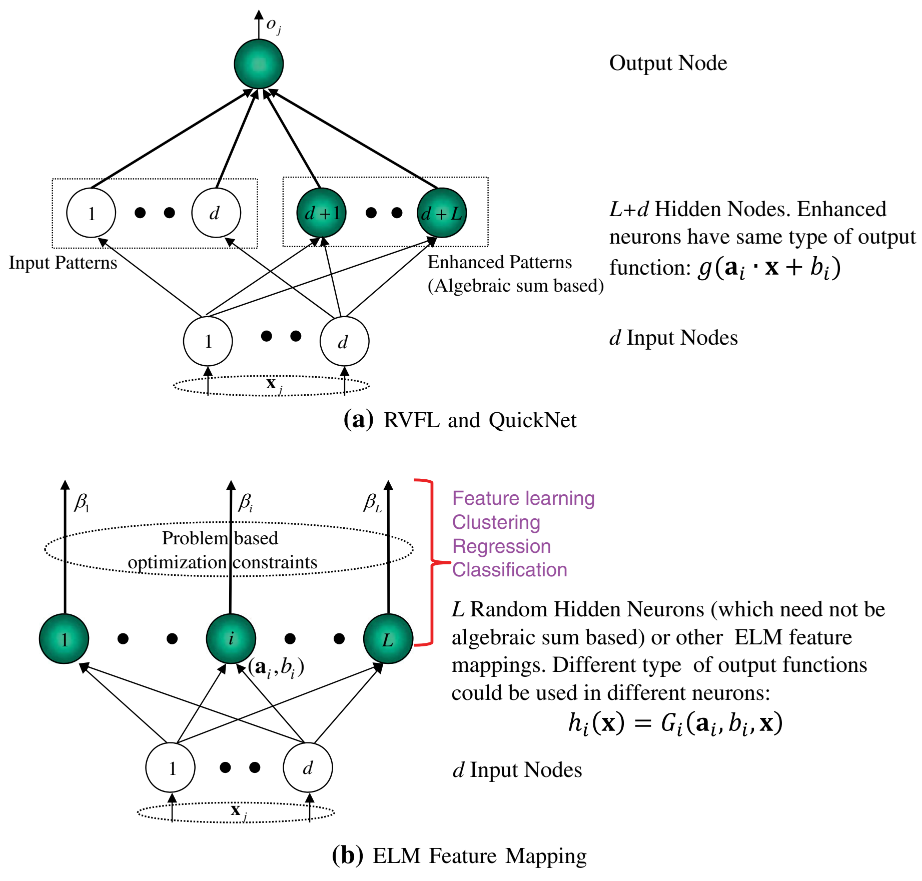

o_j
Output Node
1
d
$d+1$
$d+L$
L+d Hidden Nodes. Enhanced neurons have same type of output function: $g(\mathbf{a}_i \cdot \mathbf{x} + b_i)$
Input Patterns
Enhanced Patterns (Algebraic sum based)
d Input Nodes
$\mathbf{x}_j$
(a) RVFL and QuickNet
β_1
β_i
β_L
Problem based optimization constraints
Feature learning
Clustering
Regression
Classification
i
$(\mathbf{a}_i, b_i)$
L Random Hidden Neurons (which need not be algebraic sum based) or other ELM feature mappings. Different type of output functions could be used in different neurons:
$h_i(\mathbf{x}) = G_i(\mathbf{a}_i, b_i, \mathbf{x})$
d Input Nodes
(b) ELM Feature Mapping

Fig. 1 Difference and relationship among RVFL [54], QuickNet [41, 42] and ELM: ELM could be considered to simply RVFL and QuickNet with the introduction of optimization constraints. However, in essence, ELM was proposed with the inspiring biological learning where one may not have a way to know the exact shares/modeling of the neurons. ELM using standard feedforward neural networks is efficient in regression, classification, clustering[55] and feature learning

Non-kernel Case

From Eqs. (28a) and (28b), we have:

$$\boldsymbol{\beta} = C\mathbf{H}^T\xi \tag{34}$$

$$\xi = \frac{1}{C}(\mathbf{H}^T)^{\dagger}\boldsymbol{\beta} \tag{35}$$

From Eq. (28c), we have:

$$\begin{aligned} &\mathbf{H}\boldsymbol{\beta} - \mathbf{T} + \frac{1}{C}(\mathbf{H}^T)^{\dagger}\boldsymbol{\beta} = 0 \\ &\mathbf{H}^T\left(\mathbf{H} + \frac{1}{C}(\mathbf{H}^T)^{\dagger}\right)\boldsymbol{\beta} = \mathbf{H}^T\mathbf{T} \\ &\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \end{aligned} \tag{36}$$

In this case, the ELM output function is:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \tag{37}$$

Readers can refer to Huang et al. [32] for details of the above-mentioned.

Randomness: Random Neurons and Random Weights

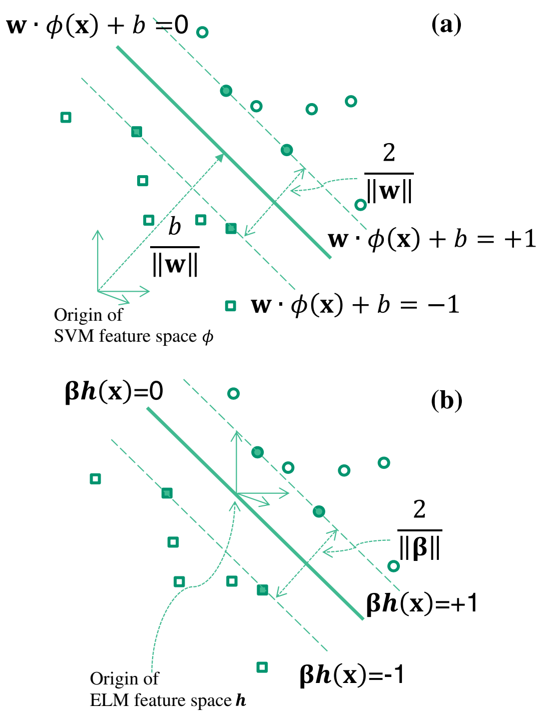
Randomness is ambiguous and confusing in the related research community. Huang et al. [26, 56] brief the relationship and difference between RBF networks, random vector version of the functional-link (RVFL) net [31], QuickNet [41, 42, 57, 58] and other related works.

Baum [59] found from experiments in 1988 that the weights of the connections on one level can be simply fixed. Baum [59] did not discuss whether all the hidden node biases b_i should be randomly generated. Baum's experiments may be related to Rosenblatt's perceptron [36–39]. We will discuss Rosenblatt's perceptron in section "Conclusions".

Almost in the same time (1989), White [41, 42, 57, 58] propose a network called QuickNet which uses "random hidden nodes" in the SLFNs augmented by connections from the input layer to the output layer.[2]

In 1992, Schmidt et al. [40] suggest to use random hidden nodes in the SLFNs with sigmoid activation functions. The universal approximation capability of the

[2] We would like thank Halbert White for the fruitful discussions on ELM during our personal communications and meetings in 2011.



$\mathbf{w}\cdot\phi(\mathbf{x})+b=0$
(a)
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}\cdot\phi(\mathbf{x})+b=+1$
$\mathbf{w}\cdot\phi(\mathbf{x})+b=-1$
Origin of SVM feature space ϕ
$\boldsymbol{\beta}\boldsymbol{h}(\mathbf{x})=0$
(b)
$\frac{2}{\|\boldsymbol{\beta}\|}$
$\boldsymbol{\beta}\boldsymbol{h}(\mathbf{x})=+1$
$\boldsymbol{\beta}\boldsymbol{h}(\mathbf{x})=-1$
Origin of ELM feature space $\boldsymbol{h}$

Fig. 2 As SVM was originally proposed for classification, universal approximation capability was not considered at the first place. Actually the feature mappings $\boldsymbol{\phi}(\mathbf{x})$ are unknown and may not satisfy universal approximation condition, b need to be present to absorb the system error. ELM was originally proposed for regression, the feature mappings $\mathbf{h}(\mathbf{x})$ are known, and universal approximation capability was considered at the first place. In ELM, the system error tends to be zero and b should not be present

proposed solution was not proved. ELM theory shows instead of sigmoid function, ELM with wide type of activation functions has universal both approximation capability and classification capability. Unlike ELM, there is no optimization constraints required in Schmidt et al. [40], and thus, it may have a gap among the random weight network proposed by Schmidt et al. [40], SVM, neural network generalization performance and system stability, and such gap is finally filled by ELM theory and techniques.

In 1994, Pao et al. [54] proposed a random vector version of the functional-link (RVFL) net.[3] Both RVFL [54] and White's QuickNet [41, 42] have the same network architecture and network output function $f_L(\mathbf{x})=\sum_{i=1}^{L}\boldsymbol{\beta}_i h_i(\mathbf{x})+\boldsymbol{\theta}\cdot\mathbf{x}$ but different learning algorithms. QuickNet readjusts the output weights of the existing hidden nodes after a new hidden node is added. RVFL uses the conventional gradient descent method which ELM tries to avoid. Both RVFL and QuickNet share the same network architecture and do not use standard feedforward neural networks (cf. Fig. 1). Unlike ELM, this type of RVFL uses a direct link between the input layer and the output layer. In other words, the hidden layer feature mapping in RVFL and QuickNet is $[h_1(\mathbf{x}),\cdots,h_L(\mathbf{x}),\mathbf{x_1},\cdots,\mathbf{x_N}]$, which is not a random feature (cf. Definition 3.2), while the random feature in ELM is $[h_1(\mathbf{x}),\cdots,h_L(\mathbf{x})]$.

In 1995, Igelnik and Pao [31] try to prove the universal approximation of RVFL based on partial randomness in hidden neurons, which is different from ELM in which hidden neurons are linearly independent: in RVFL, the input weights $\mathbf{a}_i$ are "uniformly" drawn from a probabilistic space $V_\alpha^d=[0,\alpha\Omega]\times[-\alpha\Omega,\alpha\Omega]^{d-1}$ (d: the input dimension). The hidden node biases b_i depend on the weights $\mathbf{a}_i$ and some other parameters $\mathbf{y}_i$ and u_i: $b_i=-(\alpha\mathbf{a}_i\cdot\mathbf{y}_i+u_i)$ which depends on the training data distribution. Igelnik and Pao [31] only prove the universal approximation capability of RVFL with random input weights and tuned hidden neuron biases; it does not address the universal approximation capability of a standard SLFN with both random input weights and random hidden neuron biases. Thus, although some kind of randomness (i.e., random input weights) is used in this type of RVFLs [31], unlike ELM, RVFL uses neither random hidden neurons (cf. Definition 3.1) nor random features (cf. Definition 3.2). This subtle point of "data independent hidden neurons" in ELM may actually disclose the essence of some biological learning and show the significant difference between biological learning methods (e.g., ELM) and artificial learning methods (e.g., RVFL, etc.). On the other hand, ELM theories on the universal approximation capability can be linearly extended to RVFL and QuickNet.

It should be highlighted that in ELM random hidden nodes mean *all* the parameters are randomly generated and independent of training data. For example, if additive hidden nodes are used, $h_i(\mathbf{x})=g(\mathbf{a}_i\cdot\mathbf{x}+b_i)$ where g is any nonlinear piecewise continuous activation function. In ELM, both $\mathbf{a}_i$ and b_i will be randomly generated and independent of the training data. Unlike ELM, none of the earlier methods require the optimization constraints in its output weights. Without the necessary optimization constraints, it would be impossible to build up the links between those traditional neural network learning methods and support vector machines.

Optimization Constraints with Kernels: ELM and SVM/LS-SVM

It has been analyzed in theory and verified by simulations over wide types of applications [32, 52, 53] that SVM and LS-SVM may provide solutions suboptimal to ELMs if the same kernels are used. Recently, more and more research works show that ELM-based implementation outperforms

[3] We would like to thank Boris Igelnik for discussing the relationship and difference between RVFL and ELM in our personal communication, and for sharing the RVFL patent information.

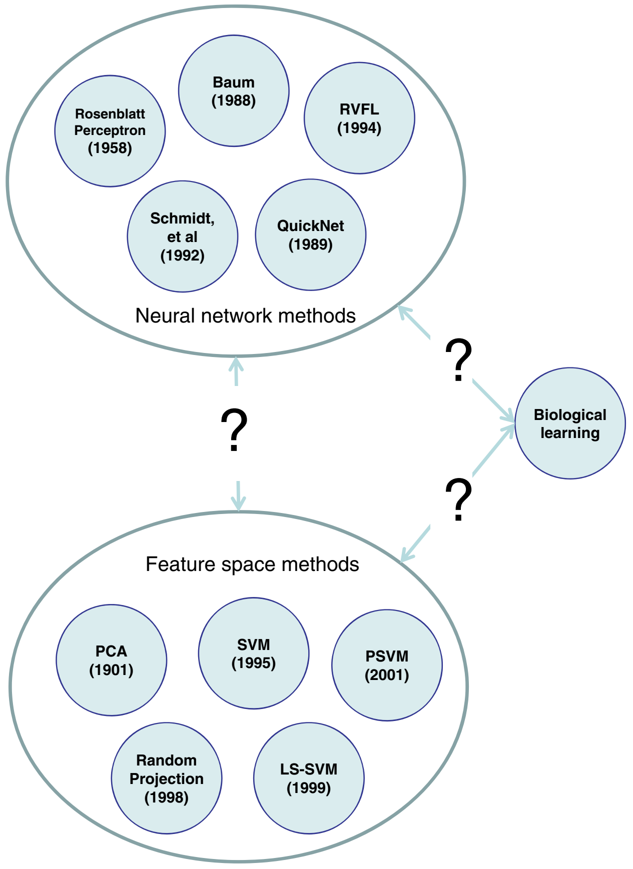


Fig. 3 Missing relationship among artificial neural networks, feature space methods and biological learning mechanism

SVM with classification accuracy rate significantly improved [32, 60–64]. Based on our earlier works [32, 52, 53], this section further gives a more complete and focusing discussion on the reasons why in contrast to the common understanding and sense of the research community SVM and LS-SVM perhaps tend to lead to suboptimal solutions.

Bias b

Just due to different ways of handling the output node bias b, different variants (e.g., LS-SVM, proximal support vector machine (PSVM) [65]) have even been proposed, which implies the output node bias b plays a critical role in SVM and its variants. The separation capability of SVM was considered more important than its regression capability when SVM was first proposed to handle binary classification applications. In this case, its universal approximation capability may somehow have been neglected [1]. This may also partially be due to the inborn reason that the feature mapping $\boldsymbol{\phi}(\mathbf{x})$ in SVM is unknown,[4] and thus, it is difficult to study the universal approximation capability of SVM without knowing feature mapping $\boldsymbol{\phi}(\mathbf{x})$.

From function approximation point of view, since $\boldsymbol{\phi}(\mathbf{x})$ is unknown and may not have universal approximation capability [1], given a target continuous function $f(\mathbf{x})$ and any small error bound $\epsilon > 0$, there may not exist $\mathbf{w}$ such that $\|\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) - f(\mathbf{x})\| < \epsilon$. In other words, there may exist some system errors even if SVM and its variants with appropriate kernels can classify different classes well, and these system errors need to be absorbed by the bias b. This may be the reason why in principle the bias b has to remain in the optimization constraints (1) and (7) for SVM and LS-SVM solutions.

From classification point of view, it is reasonable to think that the separating hyperplane in SVM may not necessarily pass through the origin in the SVM feature space and thus a bias b is preferred in the optimization constraints of SVM and LS-SVM so that the separating hyperplane can be adjusted accordingly: $\mathbf{w} \cdot \phi(\mathbf{x}_i) + b = 0$ (cf. Fig. 2a).

However, all the parameters of the ELM mapping $\mathbf{h}(\mathbf{x})$ are randomly generated, and the ELM mapping $\mathbf{h}(\mathbf{x})$ becomes known to users finally. Such ELM feature mapping $\mathbf{h}(\mathbf{x})$ includes but is not limited to sigmoid functions, RBF hidden nodes, fully complex nodes [4], wavelets [66, 67] and Fourier series. According to ELM theories [3, 5–7, 26], ELM with almost any nonlinear piecewise continuous functions $\mathbf{h}(\mathbf{x})$ has the universal approximation capability, and the separating hyperplane tends to pass through the origin in the ELM feature space. Thus, the bias b is required neither in the output nodes nor in the ELM's optimization constrains (24) and (26) (cf. Fig. 2b).

When a kernel is used in ELM, the feature mapping $\mathbf{h}(\mathbf{x})$ need not be known to users, instead one may use its corresponding kernel $K(\mathbf{u}, \mathbf{v})$. ELM originally focuses on the regression of "generalized" single-hidden layer feedforward neural networks (SLFNs) and ELM naturally has the universal approximation capability at the first place by default. In fact if some target functions cannot be approximated by a SLFN, such a SLFN need not be considered at all. In other words, the universal approximation capability is a necessary condition required for $\mathbf{h}(\mathbf{x})$ in ELM. In this case, the bias b is not required either. Suykens et al. [68] also show that the bias b may not be required in the solutions.[5] However, Suykens et al. [68] do not point out that the existence of bias b may result in additional constraints and make the final solution tend to be suboptimal although the difference in classification rates

[4] This is also the reason why SVM and its variants focus on kernels while ELM is valid for both kernel and non-kernel cases.

[5] We would like to thank Johan A. K. Suykens for showing us the analysis of the role of the bias b of LS-SVM in their monograph [68] in our personal communication.

obtained by the solutions with and without the bias b may not be apparent in some binary cases.

Hyperplane Constraint: $\sum_{i=1}^{N} \alpha_i t_i = 0$

KKT conditions (3c) and (9b) are necessary conditions for one to find the optimal solutions of the classical SVM and LS-SVM, that is:

$$\begin{aligned} \frac{\partial L_{\mathrm{SVM}}}{\partial b} = 0 &\Longrightarrow \sum_{i=1}^{N} \alpha_i t_i = 0 \\ \frac{\partial L_{\mathrm{LS-SVM}}}{\partial b} = 0 &\Longrightarrow \sum_{i=1}^{N} \alpha_i t_i = 0 \end{aligned} \tag{38}$$

That means, due to the existence of b in SVM and LS-SVM, one of the necessary conditions for both SVM and LS-SVM optimization solutions is $\sum_{i=1}^{N} \alpha_i t_i = 0$.

In contrast, according to ELM theories [3, 5–7, 26], ELM has the universal approximation capability and thus the separating hyperplane of ELM basically tends to pass through the origin in the ELM feature space (cf. Fig. 2b), there is no bias b in the optimization constraints of ELM (24) and (26). Thus, ELM does not have the necessary condition $\sum_{i=1}^{N} t_i \alpha_i = 0$ in its dual optimization problem. In other words, compared with SVM and LS-SVM, ELM has similar dual optimization objective functions[6] but with the loose conditions. The solution space (the hyperplane $\sum_{i=1}^{N} \alpha_i t_i = 0$) of SVM and LS-SVM is a subset of ELM optimal solution space. Thus, SVM and LS-SVM perhaps tend to find a solution sub-optimal to ELM's solution when the same kernels are used. In general, we may have:

> If same kernels are used in ELM and SVM/LS-SVM, SVM and LS-SVM naturally lead to sub-optimal solutions.

Feature mapping to form the kernels can be unknown mappings or random feature mappings (ELM random hidden layer output mappings).

Another open problem would be: is this so-called necessary condition $\sum_{i=1}^{N} \alpha_i t_i = 0$ for SVM and LS-SVM really reasonable?

The hyperplane $\sum_{i=1}^{N} \alpha_i t_i = 0$ in which SVM and LS-SVM search for the optimal solutions in the feature space only depends on the target output vector $[t_1, \cdots, t_N]^T$ and is independent of the input features $\mathbf{x}_i$. Given two applications, for examples, weather forecasting and face-based gender recognition. Suppose that there are 50 samples for sunny cities (50 positive samples) and 50 samples for raining cities (50 negative samples) in the weather forecasting case, and there are 50 boy face photos (50 positive samples) and 50 girl face photos (50 negative samples) in the gender recognition case. Since target labels t_i are same in both weather forecasting and gender recognition cases, the same necessary condition $\sum_{i=1}^{N} \alpha_i t_i = 0$ need to be satisfied for both applications (although in different feature spaces). In other words, both weather forecasting and gender recognition applications which have no relevance at all now become relevant in SVM and LS-SVM's solutions, this is obviously contradictory to the irrelevance fact. This actually implies that the so-called necessary condition $\sum_{i=1}^{N} \alpha_i t_i = 0$ required in SVM and its variants may not be reasonable and the bias b should not be requested. This causes some interesting and unexpected dilemma in SVM and LS-SVM optimization problems.[7] As analyzed before, on the other hand, in SVM and LS-SVM, feature mapping $\boldsymbol{\phi}(\mathbf{x})$ is unknown and b is required to absorb the system errors (or to shift the separating hyperplane in the feature space properly).

Poggio et al. [69] analyze the roles of the bias b *from kernel property point of view* and show that

(1) For SVM with order 1 conditional positive definite kernels, the bias b and the associated constraint $\sum_{i=1}^{N} \alpha_i t_i = 0$ are "a natural choice and is 'de facto' required for infinite dimensional kernels".
(2) For SVM with positive definite kernels, the bias b is not required but allowed.
(3) It is also quite reasonable to have a constant b play the role a threshold.

However, *from the optimization point of view*, our above-mentioned analysis shows that the bias b should not exist no matter whether the kernel is positive, conditional positive, neither positive nor conditional positive, or even ELM random kernels. Steinwart et al. [70] points out that "The geometrical illustration of SVMs' linear decision surface in the feature space, which justified the use of an offset b that moves the decision surface from the origin, has serious flaws". Unlike our analysis, Poggio et al. [69] and Steinwart et al. [70] do not show that the existence of b in SVM and its variants may actually result in some contradiction and making some irrelevant applications become "somehow relevant." This potential contradiction was first pointed out

[6] Here, we only consider ELM specially for binary classification applications which SVM and LS-SVM can handle. However, ELM solutions need not be tightened in binary cases, the same solution can be applied to multi-class cases and regression cases.

[7] This dilemma may have existed to other random methods with biases in the output nodes [40] if the structure risks were considered in order to improve the generalization performance. In this case, Schmidt et al. [40] would provide suboptimal solutions too. Furthermore, to our best knowledge, all of those random methods [31, 40] have not considered structure risks at all and thus may become overfitting easily.

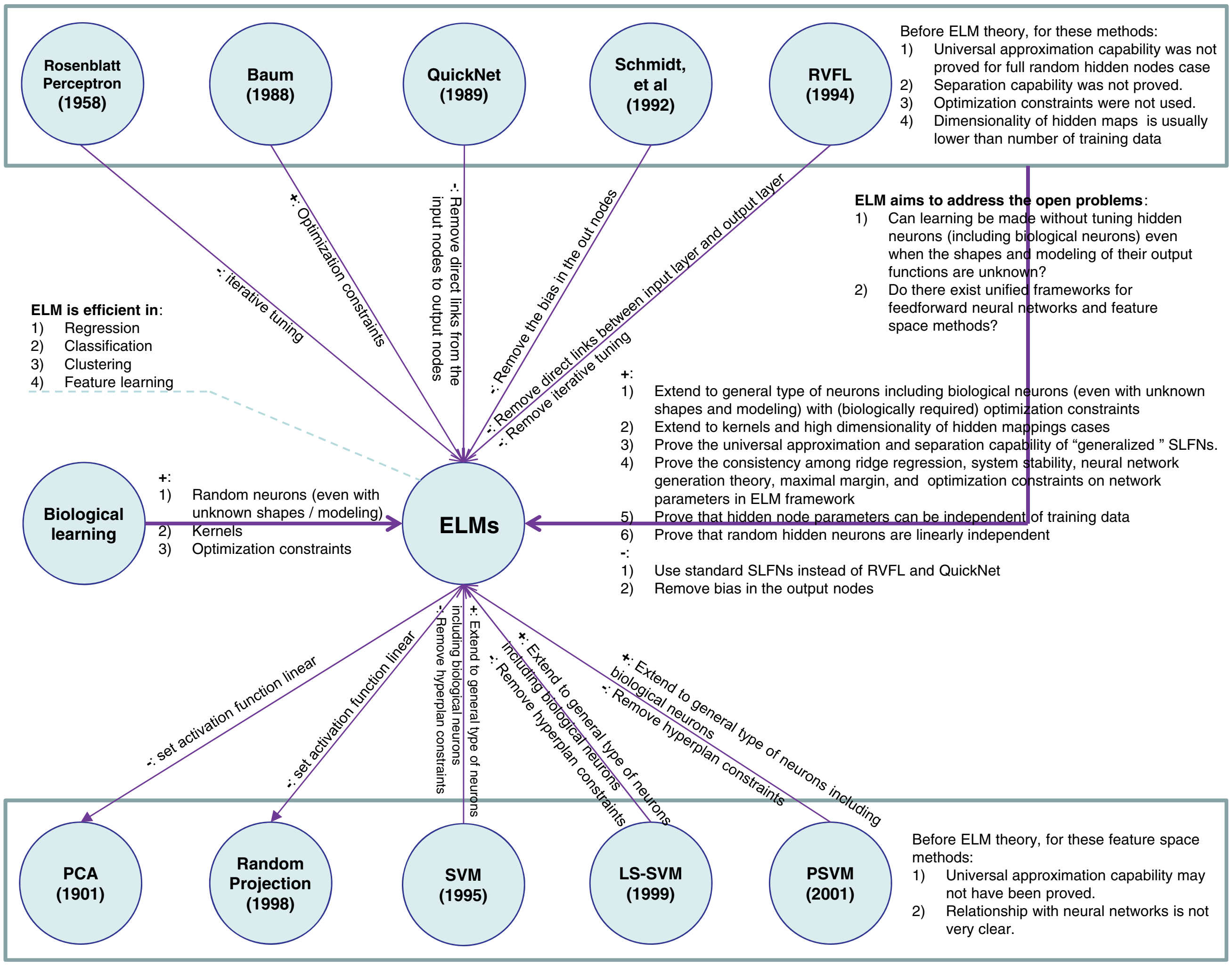

Rosenblatt Perceptron (1958)
Baum (1988)
QuickNet (1989)
Schmidt, et al (1992)
RVFL (1994)
Before ELM theory, for these methods:
1) Universal approximation capability was not proved for full random hidden nodes case
2) Separation capability was not proved.
3) Optimization constraints were not used.
4) Dimensionality of hidden maps is usually lower than number of training data
-: iterative tuning
+: Optimization constraints
-: Remove direct links from the input nodes to output nodes
-: Remove the bias in the out nodes
-: Remove direct links between input layer and output layer
-: Remove iterative tuning
ELM aims to address the open problems:
1) Can learning be made without tuning hidden neurons (including biological neurons) even when the shapes and modeling of their output functions are unknown?
2) Do there exist unified frameworks for feedforward neural networks and feature space methods?
ELM is efficient in:
1) Regression
2) Classification
3) Clustering
4) Feature learning
+:
1) Extend to general type of neurons including biological neurons (even with unknown shapes and modeling) with (biologically required) optimization constraints
2) Extend to kernels and high dimensionality of hidden mappings cases
3) Prove the universal approximation and separation capability of "generalized" SLFNs.
4) Prove the consistency among ridge regression, system stability, neural network generation theory, maximal margin, and optimization constraints on network parameters in ELM framework
5) Prove that hidden node parameters can be independent of training data
6) Prove that random hidden neurons are linearly independent
-:
1) Use standard SLFNs instead of RVFL and QuickNet
2) Remove bias in the output nodes
Biological learning
+:
1) Random neurons (even with unknown shapes / modeling)
2) Kernels
3) Optimization constraints
ELMs
-: set activation function linear
-: set activation function linear
+: Extend to general type of neurons including biological neurons
-: Remove hyperplan constraints
+: Extend to general type of neurons including biological neurons
-: Remove hyperplan constraints
+: Extend to general type of neurons including biological neurons
-: Remove hyperplan constraints
PCA (1901)
Random Projection (1998)
SVM (1995)
LS-SVM (1999)
PSVM (2001)
Before ELM theory, for these feature space methods:
1) Universal approximation capability may not have been proved.
2) Relationship with neural networks is not very clear.

Fig. 4 Difference and relationship between ELM and other related techniques: "+" and "−" mean "Add" some new elements and "Remove" some existing elements, respectively. Basic ELM could be considered as a nonlinear extensions of random projection ([72] and [18]) and the principal component analysis (PCA) [73]. ELMs fill the gap among artificial learning methods and biological learning mechanism

in our earlier work [52] which shows that b should not be used. Although almost at the same time Steinwart et al. [70] found that b is not required in SVM with positive definite kernels, Steinwart et al. [70] adopt the same viewpoint as Poggio et al. [69] that b is required for SVM with non-positive definite kernels. Both Huang et al. [52] and Steinwart et al. [70] focus on binary classification cases.

Maximal Separating Margins in Feature Space

Neural network generalization performance theory [35] shows that for feedforward neural networks reaching smaller training error, the smaller the norms of weights are, the better generalization performance the networks tend to have. On the other hand, after hidden neuron parameters are randomly generated, $\mathbf{h}(\mathbf{x})$ is known to users, ELM output function (13) becomes linear, and only its coefficient vector $\boldsymbol{\beta}$ is unknown, according to the ridge regression theory [71], the smaller the norm of the output weights $\boldsymbol{\beta}$, the more stable the linear system is and the better generalization performance the system can achieve.

For the above-mentioned two specific solutions of ELM, the distance of the separating boundaries of the two classes in the ELM feature is $2/\|\boldsymbol{\beta}\|$, thus to minimize the norm of the output weights $\boldsymbol{\beta}$ in ELM (20) is actually to maximize the separating margin. (cf. Fig. 2b)

Thus, ELM learning theory and algorithms naturally unify the neural network generalization performance theory, linear system stability theory, ridge regression theory and maximal separating margin concept. However, in other words, the maximal separating margin concept is a specific case of neural network generalization performance theory, linear system stability theory and ridge regression theory when they are used in binary classification applications. Since neural network generalization performance theory, linear system stability theory and ridge regression theory are also valid in regression and multi-class classifications, the same ELM solutions can be used in regression, binary and multi-class classification applications. SVM and LS-SVM face difficulty in handling regression and multi-class classifications as the maximal margin concept may only be valid and "visible" directly in binary classifications. The maximal margin concept used in SVM may also be valid only if $\|\mathbf{w}\|_2^2$ is considered; however, σ_1, σ_2, p and q in ELM optimization formula (20) could have different combinations due to the requirements raised in different applications. In addition to the maximum margin concept used in SVM, another reason why $\|\mathbf{w}\|_2^2$ is adopted in SVM is that $\boldsymbol{\phi}(\mathbf{x})$ is unknown and only kernels $K(\mathbf{u},\mathbf{v}) = \boldsymbol{\phi}(\mathbf{u}) \cdot \boldsymbol{\phi}(\mathbf{v})$ can be used in its dual optimization problems which require $\|\mathbf{w}\|_2^2$ in order to generate kernels properly (cf. (3a) and (9a)).

Multi-class Cases

Maximal separating margin concept can be "visible" in binary classification only. It is difficult to directly apply the maximal separating margin concept in multi-class classification and regression applications. This may be the reason why in most applications of SVM and LS-SVM, one has to convert multi-classification applications and regression applications to some type of binary classification problems, and indirectly resolve them using the classical binary solutions. Such conversion from multi-class classification applications to binary classification applications could actually result in distorting the original applications, the larger the number of classes is, the more serious the distortion becomes. For instance, given a 10-class classification applications with 10 data in each class. The One-Against-All (OAA) method often used in SVM and LS-SVM converts this multi-class application into a binary classification in this way: The ith SVM or LS-SVM is trained with 10 examples in the ith class with positive labels and all the other 90 examples from the remaining 9 classes with negative labels. In other words, such conversion has made balanced multi-class classifications become imbalanced binary classifications. The application property and distribution have been changed. However, ELM does not face such conversion distortion issue. Same ELM solutions can directly be used in regression, binary and multi-class classification applications without reformulating applications.

Conclusions

Both BP algorithm and SVMs have been playing irreplaceable roles in machine learning and computational intelligence, especially in neural networks. Without the work done by the pioneers like Werbos [22, 25], Rumelhart et al. [23, 24] and others, research in the area of neural networks may not have revived and spread quickly to the whole world since 1980s, let alone the further extensive interest in computational intelligence. Without the revolutionary SVM work by Cortes and Vapnik [1], many popular applications may not have been so successful and possibly many applications may not even have been appearing. Research on machine learning and computational intelligence area may have been halted for 10 years or more. From the scientific point of view, this paper mainly summarizes the reasons why the original SVM may not naturally lead to the optimal classification solutions in contrast to the common understanding held by the research community. Instead, SVM and its variants may perhaps tend to achieve sub-optimal solutions, especially compared with ELM *when the same kernels are used.*

Before ELM learning algorithms and theories were proposed, the relationship among different learning methods was not clear (cf. Fig. 3). ELM aims to provide a biologically inspired simple and efficient unified learning framework which fills the gap between artificial learning methods and biological learning mechanism (cf. Fig. 4). Several empirical attempts on randomness-based learning have been proposed in the literature since 1958 when Rosenblatt's perceptron [36–39] was proposed. Out of all the randomness attempts [24, 31, 36–42, 54], Rosenblatt's perceptron may be the first artificial learning machine in the related area.[8] Rosenblatt's perceptron used a linear threshold function in the hidden layers, and the weights between input layers and the first hidden layer are randomly generated with possible numbers $\{+1, -1, 0\}$. The weights between the first hidden layer and the rest of the hidden/output layers are obtained using reinforcement learning. Unlike all those randomness attempts which use algebraic sum of input weights and use sigmoid/threshold activation functions, ELM can use wide variety of activation functions in hidden neurons as long as they are nonlinear piecewise continuous. Such activation function need not be algebraic sum based. ELM theories do not care about the exact activation function formula of biological neurons which in fact human beings may not have a way to know. The randomness of Rosenblatt's perceptron focuses on the random "wiring" between the input layer (sensory layer) and the first hidden layer (associator layer) which shows the connectivity of nodes in the input layer (elements in sensory layer) and nodes in the first hidden layer.

[8] We thank Bernard Widrow for mentioning the potential links between Rosenblatt's perceptron and ELM in our personal communications.

The randomness of ELM focuses on the strength of weights of the connections between different layers.

Unlike all those randomness attempts, as discussed in this paper, ELM learning theory and algorithms naturally unify the neural network generalization performance theory, linear system stability theory, ridge regression theory and maximal separating margin concept. On the other hand, ELM theories and algorithms argue that random hidden neurons may capture the essence of some brain learning mechanism as well as the intuitive sense that the efficiency of the brain learning need not rely on computing power of neurons. This may somehow hint at possible reasons why brain is more intelligent and effective than the conventional artificial learning techniques-based computers. ELM could possibly help build some tangible link between machine learning and biological learning in some way.

ELM theories and corresponding learning algorithms may have addressed John von Neumann's concern [74, 75] why "an imperfect neural network, containing many random connections, can be made to perform reliably those functions which might be represented by idealized wiring diagrams" [36]. As shown in Theorems 3.1 and 3.2, and verified by numerous applications of ELM [5, 32, 55, 76], as long as the output functions of hidden neurons are nonlinear piecewise continuous and even if their shapes and modeling are unknown, (biological) neural networks with random hidden neurons attain both universal approximation and classification capabilities, and the changes in *finite* number of hidden neurons and their related connections do not affect the overall learning capabilities of the networks with gigantic number of hidden neurons. ELM may also be efficient in clustering [55, 77, 78], feature learning [14, 79, 80] and dimensionality reduction.

To summarize in a nutshell, this paper provides an insight into ELM as follows:

1. To clarify the relationship between ELM and other related learning methods in neural networks (QuickNet, RVFL, etc.) and feature space methods (PCA, random projection, SVM and its variants such as LS-SVM).
2. To show that ELM provides a simple unified learning mechanism (for feature learning, clustering, regression and classification) in feedforward neural networks and feature space methods.
3. To show the possible reasons on why ELM outperforms SVM and its variants with higher learning accuracy in many applications.
4. To further build up the relationship between ELM and the biological learning mechanism.

References

1. Cortes C, Vapnik V. Support vector networks. Mach Learn. 1995;20(3):273–97.
2. Suykens JAK, Vandewalle J. Least squares support vector machine classifiers. Neural Process Lett. 1999;9(3):293–300.
3. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), vol. 2, (Budapest, Hungary); 2004. p. 985–990, 25–29 July.
4. Li M-B, Huang G-B, Saratchandran P, Sundararajan N. Fully complex extreme learning machine. Neurocomputing 2005;68:306–14.
5. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: theory and applications. Neurocomputing. 2006;70:489–501.
6. Huang G-B, Chen L, Siew C-K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw. 2006;17(4):879–92.
7. Huang G-B, Chen L. Convex incremental extreme learning machine. Neurocomputing. 2007;70:3056–62.
8. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A. OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw. 2010;21(1):158–62.
9. Frénay B, Verleysen M. Using SVMs with randomised feature spaces: an extreme learning approach. In: Proceedings of the 18th European symposium on artificial neural networks (ESANN), (Bruges, Belgium); 2010. pp. 315–320, 28–30 April.
10. Frénay B, Verleysen M. Parameter-insensitive kernel in extreme learning for non-linear support vector regression. Neurocomputing. 2011;74:2526–31.
11. Cho JS, White H. Testing correct model specification using extreme learning machines. Neurocomputing. 2011;74(16):2552–65.
12. Soria-Olivas E, Gomez-Sanchis J, Martin JD, Vila-Frances J, Martinez M, Magdalena JR, Serrano AJ. BELM: Bayesian extreme learning machine. IEEE Trans Neural Netw. 2011;22(3):505–9.
13. Xu Y, Dong ZY, Meng K, Zhang R, Wong KP. Real-time transient stability assessment model using extreme learning machine. IET Gener Transm Distrib. 2011;5(3):314–22.
14. Saxe AM, Koh PW, Chen Z, Bhand M, Suresh B, Ng AY. On random weights and unsupervised feature learning. In: Proceedings of the 28th international conference on machine learning, (Bellevue, USA); 2011. 28 June–2 July.
15. Saraswathi S, Sundaram S, Sundararajan N, Zimmermann M, Nilsen-Hamilton M. ICGA-PSO-ELM approach for accurate multiclass cancer classification resulting in reduced gene sets in which genes encoding secreted proteins are highly represented. IEEE-ACM Trans Comput Biol Bioinform. 2011;6(2):452–63.
16. Minhas R, Mohammed AA, Wu QMJ. Incremental learning in human action recognition based on snippets. IEEE Trans Circuits Syst Video Technol. 2012;22(11):1529–41.
17. Decherchi S, Gastaldo P, Leoncini A, Zunino R. Efficient digital implementation of extreme learning machines for classification. IEEE Trans Circuits Syst II. 2012;59(8):496–500.
18. Gastaldo P, Zunino R, Cambria E, Decherchi S. Combining ELMs with random projections. IEEE Intell Syst. 2013;28(6):46–8.
19. Lin J, Yin J, Cai Z, Liu Q, Li K, Leung VC. A secure and practical mechanism for outsourcing ELMs in cloud computing. IEEE Intell Syst. 2013;28(6):35–8.
20. Akusok A, Lendasse A, Corona F, Nian R, Miche Y. ELMVIS: a nonlinear visualization technique using random permutations and ELMs. IEEE Intell Syst. 2013;28(6):41–6.
21. Fletcher R. Practical methods of optimization: volume 2 constrained optimization. New York:Wiley; 1981.

22. Werbos PJ. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvord University; 1974.
23. Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. In: Rumelhart DE, McClelland JL, editors. Parallel distributed processing: explorations in the microstructures of cognition, vol: foundations. Cambridge, MA: MIT Press; 1986. p. 318–62.
24. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagation errors. Nature. 1986;323:533–6.
25. Werbos PJ. The roots of backpropagation : from ordered derivatives to neural networks and political forecasting. New York:-Wiley; 1994.
26. Huang G-B, Chen L. Enhanced random search based incremental extreme learning machine. Neurocomputing. 2008;71:3460–8.
27. Sosulski DL, Bloom ML, Cutforth T, Axel R, Datta SR. Distinct representations of olfactory information in different cortical centres. Nature. 2011;472:213–6.
28. Eliasmith C, Stewart TC, Choo X, Bekolay T, DeWolf T, Tang Y, Rasmussen D. A large-scale model of the functioning brain. Science. 2012;338:1202–5.
29. Barak O, Rigotti M, Fusi S. The sparseness of mixed selectivity neurons controls the generalization–discrimination trade-off. J Neurosci. 2013;33(9):3844–56.
30. Rigotti M, Barak O, Warden MR, Wang X-J, Daw ND, Miller EK, Fusi S. The importance of mixed selectivity in complex cognitive tasks. Nature. 2013;497:585–90.
31. Igelnik B, Pao Y-H. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans Neural Netw. 1995;6(6):1320–9.
32. Huang G-B, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B. 2012;42(2):513–29.
33. Rahimi A, Recht B. Uniform approximation of functions with random bases. In: Proceedings of the 2008 46th annual allerton conference on communication, control, and computing, p. 555–561, 23–26 Sept 2008.
34. Huang G-B, Zhu Q-Y, Mao KZ, Siew C-K, Saratchandran P, Sundararajan N. Can threshold networks be trained directly? IEEE Trans Circuits Syst II. 2006;53(3):187–91.
35. Bartlett PL. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inform Theory. 1998;44(2):525–36.
36. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev. 1958;65(6):386–408.
37. Rosenblatt F. Principles of Neurodynamics: perceptrons and the theory of brain mechanisms. New York:Spartan Books; 1962.
38. Block HD. The perceptron: a model for brain function. I. Rev Modern Phys. 1962;34(1):123–35.
39. Block HD, Knight JBW, Rosenblatt F. Analysis of a four-layer series-coupled perceptron. II. Rev Modern Phys. 1962;34(1): 135–42.
40. Schmidt WF, Kraaijveld MA, Duin RP. Feed forward neural networks with random weights. In: Proceedings of 11th IAPR international conference on pattern recognition methodology and systems, (Hague, Netherlands); 1992. p. 1–4.
41. White H. An additional hidden unit test for neglected nonlinearity in multilayer feedforward networks. In: Proceedings of the international conference on neural networks. 1989. p. 451–455.
42. White H. Approxiate nonlinear forecasting methods. In: Elliott G, Granger CWJ, Timmermann A, editors. Handbook of economics forecasting. New York: Elsevier; 2006. p. 460–512.
43. Loone SM, Irwin GW. Improving neural network training solutions using regularisation. Neurocomputing. 2001;37:71–90.
44. Serre D. Matrices: theory and applications. New York:Springer; 2002.
45. Rao CR, Mitra SK. Generalized Inverse of matrices and its applications. New York:Wiley; 1971.
46. Fernández-Delgado M, Cernadas E, Barro S, Ribeiro J, Nevesb J. Direct kernel perceptron (DKP): Ultra-fast kernel elm-based classification with non-iterative closed-form weight calculation. Neural Netw. 2014;50(1):60–71.
47. Widrow B, Greenblatt A, Kim Y, Park D. The no-prop algorithm: A new learning algorithm for multilayer neural networks. Neural Netw. 2013;37:182–8.
48. Toms DJ. Training binary node feedforward neural networks by backpropagation of error. Electron Lett. 1990;26(21):1745–6.
49. Corwin EM, Logar AM, Oldham WJB. An iterative method for training multilayer networks with threshold function. IEEE Trans Neural Netw. 1994;5(3):507–8.
50. Goodman RM, Zeng Z. A learning algorithm for multi-layer perceptrons with hard-limiting threshold units. In: Proceedings of the 1994 IEEE workshop of neural networks for signal processing. 1994. p. 219–228.
51. Plagianakos VP, Magoulas GD, Nousis NK, Vrahatis MN. Training multilayer networks with discrete activation functions. In: Proceedings of the IEEE international joint conference on neural networks (IJCNN'2001), Washington D.C., U.S.A.; 2001.
52. Huang G-B, Ding X, Zhou H. Optimization method based extreme learning machine for classification. Neurocomputing. 2010;74:155–63.
53. Bai Z, Huang G-B, Wang D, Wang H, Westover MB. Sparse extreme learning machine for classification. IEEE Trans Cybern. 2014. doi:10.1109/TCYB.2014.2298235.
54. Pao Y-H, Park G-H, Sobajic DJ. Learning and generalization characteristics of the random vector functional-link net. Neurocomputing. 1994;6:163–80.
55. Huang G, Song S, Gupta JND, Wu C. Semi-supervised and unsupervised extreme learning machines. IEEE Trans Cybern. 2014. doi:10.1109/TCYB.2014.2307349.
56. Huang G-B, Li M-B, Chen L, Siew C-K. Incremental extreme learning machine with fully complex hidden nodes. Neurocomputing. 2008;71:576–83.
57. Lee T-H, White H, Granger CWJ. Testing for neglected nonlinearity in time series modes: a comparison of neural network methods and standard tests. J Econ. 1993;56:269–90.
58. Stinchcombe MB, White H. Consistent specification testing with nuisance parameters present only under the alternative. Econ Theory. 1998;14:295–324.
59. Baum E. On the capabilities of multilayer perceptrons. J Complexity. 1988;4:193–215.
60. Le Q, Sarlós T, Smola A. Fastfood approximating kernel expansions in loglinear time. In: Proceedings of the 30th international conference on machine learning, (Atlanta, USA), 16–21 June 2013.
61. Huang P-S, Deng L, Hasegawa-Johnson M, He X. Random features for kernel deep convex network. In: Proceedings of the 38th international conference on acoustics, speech, and signal processing (ICASSP 2013), Vancouver, Canada, 26–31 May 2013.
62. Lin J, Yin J, Cai Z, Liu Q, Li K, Leung VC. A secure and practical mechanism for outsourcing elms in cloud computing. IEEE Intell Syst. 2013;28(6):7–10.
63. Rahimi A, Recht B. Random features for large-scale kernel machines. In: Proceedings of the 2007 neural information processing systems (NIPS2007), 3–6 Dec 2007. p. 1177–1184.
64. Kasun LLC, Zhou H, Huang G-B, Vong CM. Representational learning with extreme learning machine for big data. IEEE Intell Syst 2013;28(6):31–4.

65. Fung G, Mangasarian OL. Proximal support vector machine classifiers. In: International conference on knowledge discovery and data mining, San Francisco, California, USA, 2001. p. 77–86.
66. Daubechies I. Orthonormal bases of compactly supported wavelets. Commun Pure Appl Math. 1988;41:909–96.
67. Daubechies I. The wavelet transform, time-frequency localization and signal analysis. IEEE Trans Inform Theory. 1990;36(5):961–1005.
68. Suykens JAK, Gestel TV, Brabanter JD, Moor BD, Vandewalle J. Least squares support vector machines. Singapore: World Scientific; 2002.
69. Poggio T, Mukherjee S, Rifkin R, Rakhlin A, Verri A. "*b*," (A.I. Memo No. 2001–011, CBCL Memo 198, Artificial Intelligence Laboratory, Massachusetts Institute of Technology), 2001.
70. Steinwart I, Hush D, Scovel C. Training SVMs without offset. J Mach Learn Res .2011;12(1):141–202.
71. Hoerl AE, Kennard RW. Ridge regression: biased estimation for nonorthogonal problems. Technometrics. 1970;12(1):55–67.
72. Kaski S. Dimensionality reduction by random mapping: fast similarity computation for clustering. In: Proceedings of the 1998 IEEE international joint conference on neural networks, Anchorage, USA, 4–9 May 1998.
73. Pearson K. On lines and planes of closest fit to systems of points in space. Philos Mag. 1901;2:559–72.
74. von Neumann J. The general and logical theory of automata. In: Jeffress LA, editor. Cerebral mechanisms in behavior. New York: Wiley; 1951. p. 1–41. 1951.
75. von Neumann J. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In: Shannon CE, McCarthy J, editors. Automata studies. Princeton: Princeton University Press; 1956. p. 43–98.
76. Minhas R, Baradarani A, Seifzadeh S, Wu QMJ. Human action recognition using extreme learning machine based on visual vocabularies. Neurocomputing. 2010;73:1906–17.
77. Wang J, Kumar S, Chang S-F. Semi-supervised hashing for large-scale search. IEEE Trans Pattern Anal Mach Intell. 2012;34(12): 2393–406.
78. He Q, Jin X, Du C, Zhuang F, Shi Z. Clustering in extreme learning machine feature space. Neurocomputing. 2014;128: 88–95.
79. Jarrett K, Kavukcuoglu K, Ranzato M, LeCun Y. What is the best multi-stage architecture for object recognition. In: Proceedings of the 2009 IEEE 12th international conference on computer vision, Kyoto, Japan, 29 Sept–2 Oct 2009.
80. Pinto N, Doukhan D, DiCarlo JJ, Cox DD. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. PLoS Comput Biol. 2009;5(11):1–12.