Neurocomputing 74 (2011) 2526–2531

Contents lists available at ScienceDirect

Neurocomputing

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

Parameter-insensitive kernel in extreme learning for non-linear support vector regression

Benoît Frénay [a,b,*], Michel Verleysen [a]

[a] Machine Learning Group, ICTEAM institute, Université catholique de Louvain, Louvain-la-Neuve, BE 1348, Belgium
[b] Aalto University School of Science and Technology, Department of Information and Computer Science, P.O. Box 15400, FI-00076 Aalto, Finland

ARTICLE INFO

Available online 12 May 2011

Keywords:
Extreme learning machine
Support vector regression
ELM kernel
Infinite number of neurons

ABSTRACT

Support vector regression (SVR) is a state-of-the-art method for regression which uses the ε-sensitive loss and produces sparse models. However, non-linear SVRs are difficult to tune because of the additional kernel parameter. In this paper, a new parameter-insensitive kernel inspired from extreme learning is used for non-linear SVR. Hence, the practitioner has only two meta-parameters to optimise. The proposed approach reduces significantly the computational complexity yet experiments show that it yields performances that are very close from the state-of-the-art. Unlike previous works which rely on Monte-Carlo approximation to estimate the kernel, this work also shows that the proposed kernel has an analytic form which is computationally easier to evaluate.

1. Introduction

In machine learning, regression is a well-known problem which has been thoroughly studied. In the inductive setting, it boils down to making hypotheses on the underlying model, choosing an objective function and then estimating the model parameters. In that process, the squared error loss is often used for mathematical convenience, since it is differentiable. However, the squared error loss leads to non-sparse models. In order to get sparse models, Vapnik proposed to use the ε-sensitive loss [1]. This loss allows data lying within a thick tube at no cost and linearly penalises data outside the tube. Integrating the ε-sensitive loss into a regularised linear model leads to support vector regression (SVR, see e.g. [2–4]). Used in conjunction with kernels, SVRs are powerful non-linear models for regression which have been shown competitive in a wide number of applications.

However, even if SVRs are conceptually appealing, their training is difficultly affordable in practice. Indeed, three meta-parameters have to be tuned for non-linear problems (as detailed in Section 2): the regularisation constant, the tube width for the loss and the kernel parameter. Therefore, one rather uses least-square support vector machines (LS-SVMs, see e.g. [5,6]). However, LS-SVMs lack the sparsity of SVRs, since they relax the quadratic programming problem solved in SVRs by using a squared error loss instead of the ε-sensitive loss.

A similar problem occurs in non-linear classification when using Gaussian kernels. Recently, [7–9] have provided a solution by proposing a new parameter-insensitive kernel inspired from extreme learning. In other words, the choice of the meta-parameter associated to the kernel does not seem to affect the quality of classification. This paper extends this concept to regression. The proposed approach implies the optimisation of only two meta-parameters: the regularisation constant and the tube width. Therefore, the computational cost of non-linear SVR is significantly reduced. Experiments show that the approach yields state-of-the-art performances on various datasets.

The paper is organised as follows. Section 2 reviews SVR in the regression framework. Section 3 shortly introduces the basics of extreme learning. Section 4 derives the new kernel and shows that it has an analytical form under some assumptions. Eventually, the experiments carried in Section 5 show that our computationally cheaper approach achieves state-of-the-art results.

2. Support vector regression

Given a dataset $\mathcal{D}=\{(x_i,t_i)\}_{i\in 1..n}$ where $x\in\Re^d$ and $t\in\Re$ are respectively the inputs and targets, regression consists in building a model $f:\Re^d\to\Re$ which gives a good estimate $y=f(x)$. Usually, models are obtained by minimising an estimate of the expected value of the loss $\mathcal{L}(t,y)$. This estimate is called the empirical risk $\mathcal{R}_{emp}(f)$ and is computed from $\mathcal{D}$. In practice, the empirical risk

* Corresponding author at: Batiment Maxwell, place du Levant, 3, Louvain-la-Neuve, BE 1348, Belgium. Tel.: +32 10 47 81 33; fax: +32 10 47 2598.
E-mail address: benoit.frenay@uclouvain.be (B. Frénay).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.11.037

corresponding to the squared error loss, i.e. the mean squared error

$$\mathcal{R}_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(t_i,y_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i-t_i)^2 \tag{1}$$

is often used because it is mathematically convenient, but it leads to non-sparse models taking into account every single data. Vapnik has developed an alternative loss, called the ε-sensitive loss [1], which leads to sparse models depending only on a small subset of the whole dataset. This section reviews how to embed this loss into a linear model, called support vector regression (SVR, see e.g. [2–4]), which can be extended to deal with non-linear problems.

2.1. Linear support vector regression

The ε-sensitive loss allows the estimate y lying in a thick tube around the observed target t at no cost and penalises it linearly outside the tube, i.e.

$$|y-t|_\varepsilon = \begin{cases} 0 & \text{if } |y-t| \le \varepsilon, \\ |y-t|-\varepsilon & \text{if } |y-t| > \varepsilon, \end{cases} \tag{2}$$

where ε denotes the half-width of the tube, as illustrated in Fig. 1(a). SVRs are linear models which try to find a compromise between the model complexity and the total ε-sensitive loss, i.e.

$$\left| \min_{w,b,\xi_i} \|w\|_2^2 + C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) \text{s.t.} \begin{cases} \langle w,x_i\rangle + b - t_i \ge \varepsilon + \xi_i^+, \\ t_i - \langle w,x_i\rangle + b \ge \varepsilon + \xi_i^-, \\ \xi_i^+, \xi_i^- \ge 0, \end{cases} \right. \tag{3}$$

where w is the weight vector, b is the bias, C is the regularisation constant and ξ_i^+, ξ_i^- are positive slack variables. The above optimisation problem minimises the norm $\|w\|_2^2$ in order to control the model complexity. Moreover, the objective function is regularised by the ε-sensitive loss: the regularisation constant C determines the compromise between model complexity and errors. Notice that the sum of ξ_i^+ and ξ_i^- is equal to the ε-sensitive loss term for the ith data.

Using the dual form of its Lagrangian, it can be shown [2] that Eq. (3) is equivalent to a quadratic programming problem expressed only in terms of the Lagrange multipliers α_i^+, α_i^- corresponding to the tube constraints in the primal form. The weight vector can then be expressed as

$$w = \sum_{i=1}^{n}(\alpha_i^- - \alpha_i^+)x_i, \tag{4}$$

where only data with different dual variables α_i^+, α_i^- are used to estimate w. These particular data are called *support vectors*. Fig. 1(b) shows a simple example of SVR, where f and the corresponding tube is shown, as well as the three support vectors and two slack variables.

Here, the compromise between model complexity, sparsity and over-fitting is determined by the regularisation constant C. Low values of C correspond to simpler and sparser models, whereas models inferred for large values of C better fit the training data. Moreover, the ε constant controls the width of the tube. Therefore, both the values of C and ε have to be selected simultaneously from training data. For example, a grid search can be done using 10-fold cross-validation (see e.g. [2]).

2.2. Non-linear support vector regression

SVR can be easily extended to handle non-linear regression problems. Indeed, only dot products appear in the dual of Eq. (3). Moreover, the model prediction for a new point is simply

$$f(x) = w\cdot x = \sum_{i=1}^{n}(\alpha_i^- - \alpha_i^+)\langle x_i,x\rangle. \tag{5}$$

Given a non-linear mapping ϕ to a high-dimensional space, the corresponding kernel is defined as

$$k(x,z) = \langle \phi(x),\phi(z)\rangle. \tag{6}$$

Since k computes a dot product, the mapping itself is not necessary. In practice, the Gaussian kernel

$$k(x,z) = \exp(-\gamma\|x-z\|_2^2), \tag{7}$$

which corresponds to an infinite dimensional space is often used and gives good results. However, using SVRs with a Gaussian kernel induces the need to tune three meta-parameters: the regularisation constant C, the tube half-width ε and the kernel scale γ. Using 10-fold cross-validation and only 10 possible values for each meta-parameter, not less than 10^4 SVRs have to be trained. The rest of this paper shows that it is possible to obtain results of the same quality by using a new parameter-insensitive kernel and therefore to reduce strongly the number of SVRs to be trained.

3. Extreme learning

This section introduces extreme learning, a recent trend in machine learning which aims at producing fast, but accurate models. The next section shows how to extend it to kernel-based machines.

3.1. Extreme learning machines

In [10,11], Huang et al. propose a new way to optimise networks with a single hidden layer of units that they call extreme learning. Firstly, a large hidden layer is created with random weights and biases, e.g. picked up uniformly in $[-3,3]$. Then, the output weights and bias are optimised by solving a linear system. As opposed to back-propagation [12], this approach does not get stuck in local minima and is very fast.

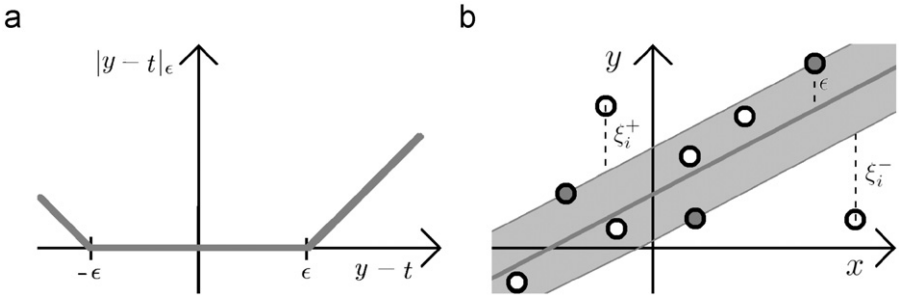

Fig. 1. (a) ε-sensitive loss and (b) example of SVR trained on a linear problem with two outliers.

More formally, let us define W and w being respectively the hidden and output weights. Moreover, we choose an activation function σ for hidden units and define X as the matrix whose rows are data. An additional column of ones is added to X to embed the biases directly in W, resulting in X_b. Therefore, the activation of hidden units corresponding to the data is simply

$$H = \sigma(X_b W). \tag{8}$$

Other types of hidden units can be used, e.g. radial basis function (RBF) hidden units [13–15]. If hidden weights are held fixed and the output activation is linear, the output weights are given by

$$\underset{w}{\operatorname{argmin}} \mathcal{L}(T, H_b w), \tag{9}$$

where $\mathcal{L}$ is the loss and H_b is the matrix H with an additional column of ones added in order to embed the bias in w. Using the squared error loss, the above equation becomes

$$\underset{w}{\operatorname{argmin}} \|H_b w - T\|^2. \tag{10}$$

The resulting model is called an extreme learning machine (ELM) and can be trained easily at a very low computational cost. Indeed, Eq. (10) is simply a linear regression whose solution is

$$w = H_b^{\dagger} T, \tag{11}$$

where $H_b^{\dagger} = (H_b' H_b)^{-1} H_b'$ is the Moore–Penrose pseudo-inverse of the complete activation matrix H_b. Huang et al. [10,11] have shown that, using the above setting, it is possible to solve regression problems very fast and yet to obtain satisfying results. Huang et al. [16] have also shown that the extreme learning machines (ELMs) are universal approximators and can be built incrementally, starting with one unit in the network.

3.2. Algorithms to train extreme learning machines

The number of units of an ELM cannot be set arbitrarily. For example, let us consider the approximation of a simple sine function, using 20 noisy training samples. Fig. 2(a) shows the mean squared error on an independent test set for different ELM sizes. Fig. 2(b) shows the resulting approximations with 2, 9 and 20 units, where 9 units is the optimal ELM size in Fig. 2(a). Here, choosing too many or not enough units leads respectively to overfitting or underfitting.

To our knowledge, three approaches have been proposed so far to choose the number of units in ELMs. First, Huang et al. [10,11,16] propose to build networks with different sizes, either incrementally or not, and to pick the best size using e.g. cross-validation methods. Second, Miche et al. [17,18] propose the OP-ELM framework which uses the LARS/LASSO to compute a L-1 regularisation path for the output weights, then pick up the set of units with the lowest LOO error and eventually retrain the model. Third, Liu et al. [7] use L-2 regularisation, i.e. Ridge regression.

These algorithms do not produce sparse models since they are essentially solving a linear regression with a squared error loss. Notice that OP-ELM is sparse in terms of units, not in terms of data. Indeed, only a few units are selected using L-1 regularisation, but output weights are computed from all the data. The next section shows how to introduce the ε-sensitive loss in ELMs in order to obtain sparse models.

4. Extreme learning with kernel-based models

Recently, several papers [7–9] have emphasised the similarities between ELMs and kernel-based methods. This section reviews ELM from the kernel point of view and shows how to create a new family of kernels inspired from extreme learning. An analytical expression is derived for one of these kernels and it is shown in Section 5 that the kernel parameter does not need to be tuned.

4.1. A change of perspective: the ELM kernel

In standard ELMs, the role of the hidden layer is to map the input to the hidden units. In other words, the first, hidden layer of the ELM is mapping data from the data space to some higher dimensional space, where each dimension corresponds to a hidden unit. Hence, this new high-dimensional space can be viewed as a feature space where the ELM just solves a linear regression. This new interpretation leads to the definition of the so-called ELM kernel.

Let us define the mapping $\phi : \Re^d \to \Re^p$ such that the ith component of $\phi(x)$ is equal to the activation of the ith hidden unit. Given two data points x and z and an ELM with p units, the corresponding ELM kernel function is defined as

$$k(x,z|p) = \frac{1}{p} \langle \phi(x), \phi(z) \rangle, \tag{12}$$

which is simply the dot product in the feature space, normalised by the number of hidden units p. By construction, k is a valid kernel and can be used by any kind of kernel-based method. Indeed, it corresponds to the dot product of x and z in the feature space defined by the mapping ϕ, i.e. the hidden layer of an ELM.

The feature space associated to the ELM kernel is built randomly, since each dimension corresponds to a hidden unit

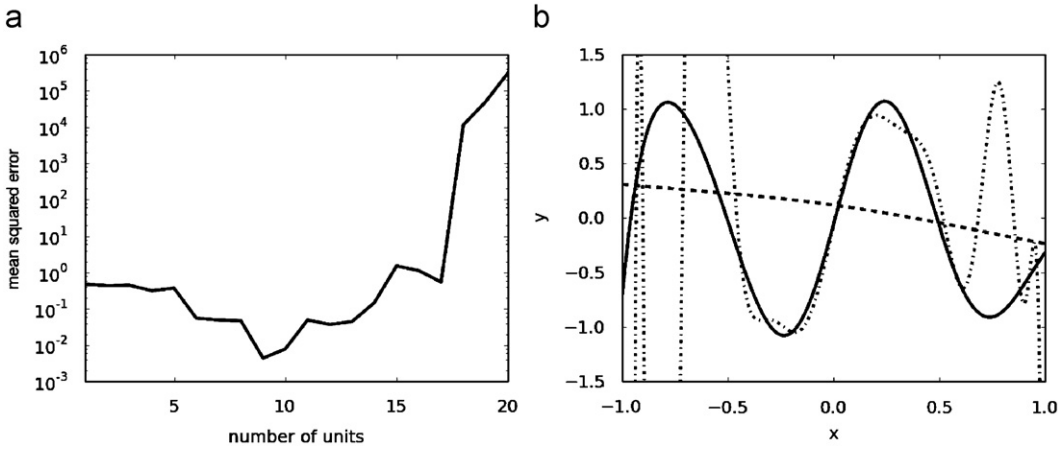

Fig. 2. Sine function approximation from 20 noisy data using ELMs: (a) evolution of the mean squared error computed on an independent test set of 10^3 samples as the number of units increases and (b) models obtained with 2 units (dashed line), 9 units (plain line) and 20 units (dotted line).

with random weights and bias. However, each data is mapped to the same feature space using the hidden layer of the same ELM.

4.2. Limit case and analytical form of the ELM kernel

It has been shown in [8,9] that the number of units used to evaluate the ELM kernel does not matter for classification tasks, as long as it is large enough. Let us assume a network with a single hidden layer of units, whose hidden layer weights are picked randomly from a given prior. If we let the number of units p grow to infinity, the ELM kernel becomes

$$\lim_{p\to+\infty} k(x,z|p) = \lim_{p\to+\infty} \frac{1}{p} \langle \phi(x), \phi(z) \rangle = \lim_{p\to+\infty} \frac{1}{p} \sum_{i=1}^{p} \sigma(w_i x)\sigma(w_i z) = \mathbb{E}_w[\sigma(wx)\sigma(wz)]. \quad (13)$$

In other words, the limit $k(x,z|p\to+\infty)$ can be interpreted as the expected value of $\sigma(wx)\sigma(wz)$, i.e. the covariance between the activations of a random hidden unit alternatively fed with x and z. In the rest of this paper, $k(\cdot,\cdot|p\to+\infty)$ is referred to as the *asymptotic ELM kernel*.

Let us consider a particular case where (i) the weights and biases of the hidden layer are picked randomly from an isotropic Gaussian distribution with variance σ_w^2 and (ii) the activation function is the sigmoid *erf* function defined as

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, dt. \quad (14)$$

It can be shown [19] that the asymptotic ELM kernel admits the following analytical expression:

$$k(x,z|p\to+\infty) = \frac{2}{\pi} \arcsin \frac{1+\langle x,z \rangle}{\sqrt{\left(\frac{1}{2\sigma_w^2}+1+\langle x,x \rangle\right)\left(\frac{1}{2\sigma_w^2}+1+\langle z,z \rangle\right)}}. \quad (15)$$

Since an analytical expression is available, it is no longer necessary to actually build ELMs in order to implement this kernel. Therefore, the implementation is straightforward. In the rest of this paper, we use the normalised version of the asymptotic ELM kernel

$$\tilde{k}(x,z|p\to+\infty) = \frac{k(x,z|p\to+\infty)}{\sqrt{k(x,x|p\to+\infty)k(z,z|p\to+\infty)}} = \frac{\mathbb{E}_w[\sigma(wx)\sigma(wz)]}{\sqrt{\mathbb{E}_w[\sigma(wx)\sigma(wx)]\mathbb{E}_w[\sigma(wz)\sigma(wz)]}}, \quad (16)$$

which can be interpreted as the correlation between the activations of a random hidden unit alternatively fed with x and z. Hence, we have $\tilde{k}(x,x|p\to+\infty)=1$. Fig. 3 shows the shape of the resulting kernel evaluated around zero for different values of σ_w. Even if the shape itself changes, the scale of the kernel is constant and does not depend on σ_w. Section 5 shows that the σ_w parameter does not affect the results obtained when using SVR with this kernel, if it is chosen large enough.

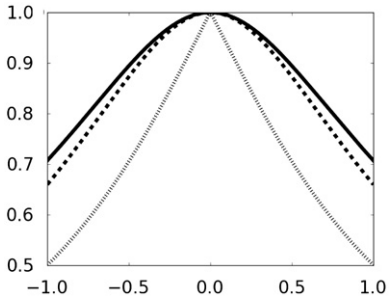

Fig. 3. Normalised asymptotic ELM kernel evaluated around zero for different values of σ_w: $\sigma_w = 10^{-3}$ (plain line), $\sigma_w = 1$ (dashed line) and $\sigma_w = 10^3$ (dotted line).

5. Experiments

This section aims to answer two questions: (i) is it necessary to tune the σ_w parameter for the normalised asymptotic ELM kernel and (ii) does SVR get state-of-the-art results with that kernel?

5.1. Experimental setting

SVR results are obtained using the LIBSVM library [20] and several datasets of various sizes coming from the UCI machine learning repository [21] (Abalone, Cancer and Machine CPU) and from [22] (Ailerons, CompActs, Elevators, Stock and Triazines); see Table 1 for details. The datasets are separated into two groups: small datasets (several hundred of instances) and large datasets (several thousands of instances).

For each dataset, we used 10-fold cross-test (see Fig. 4). Each dataset is split into 10 folds which are alternatively used as independent test sets for the models built on the nine remaining folds. These training data are then in turn split into 10 folds which are alternatively used as validation sets for the models built on the nine remaining folds. Validation sets are used to select the best values for the meta-parameters. The criterion used here to compare models is the mean squared error (MSE).

The standard Gaussian kernel and the normalised asymptotic ELM kernel are used respectively with values in a logarithmic scale from 10^{-3} to 10^{0} for γ and from 10^{-3} to 10^{3} for σ_w. The other meta-parameter values are respectively taken logarithmically from 10^{-2} to 10^{6} for C and from 10^{-5} to 10^{1} for ε. For the meta-parameters γ, C and ε, each logarithmic step consists in multiplying their value by $\sqrt{10}$: C takes e.g. 17 different values. For each trained SVR, the inputs and targets are normalised using the mean and standard deviations computed from the training data. However, the differences between true targets and predictions are multiplied afterwards by the standard deviation in order to be expressed in terms of original units.

Table 1
Detailed list of datasets used for experiments, ordered by size and split into two groups.

Short name	Group	Size	Dimensionality	Full name
Triazines	Small	186	60	Inhibition of dihydrofolate reductase by triazines
Cancer		192	32	Wisconsin prognostic breast cancer
CPU		209	6	Relative CPU performance data
Stock		950	9	Daily stock prices dataset
Abalone	Large	4177	8	Abalone data
Ailerons		7129	5	Delta ailerons control
CompActs		8192	21	Computer activity database
Elevators		9517	6	Delta elevators control

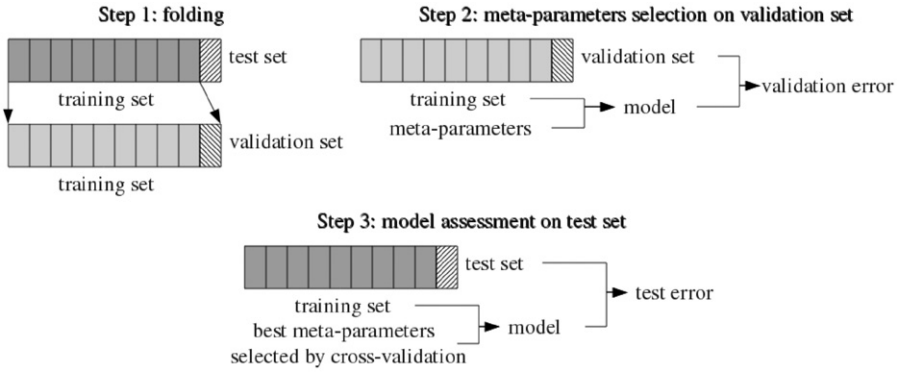

Fig. 4. Illustration of the cross-test method.

Table 2
Means and 95% confidence intervals for the test MSE obtained on small datasets using SVR with (i) the Gaussian kernel and (ii) the normalised asymptotic ELM kernel for different σ_w. Results which are not significantly different from the best result (underlined) are in bold font.

	Gaussian kernel	Normalised asymptotic ELM kernel						
		$\sigma_w = 1e{-3}$	$\sigma_w = 1e{-2}$	$\sigma_w = 1e{-1}$	$\sigma_w = 1e{+0}$	$\sigma_w = 1e{+1}$	$\sigma_w = 1e{+2}$	$\sigma_w = 1e{+3}$
Cancer	**1.1e+3**	**1.0e+3**	**1.0e+3**	**1.1e+3**	**1.0e+3**	**1.0e+3**	**1.1e+3**	**1.0e+3**
	[8.6e+2, 1.3e+3]	[8.9e+2, 1.2e+3]	[8.9e+2, 1.2e+3]	[9.4e+2, 1.2e+3]	[8.7e+2, 1.2e+3]	[8.7e+2, 1.2e+3]	[9.3e+2, 1.2e+3]	[8.6e+2, 1.2e+3]
CPU	**4.5e+3**	1.3e+4	1.0e+4	**3.0e+3**	**3.9e+3**	**3.9e+3**	**4.1e+3**	**4.1e+3**
	[−1.0e+3, 1.0e+4]	[3.9e+3, 2.2e+4]	[3.5e+3, 1.8e+4]	[1.4e+3, 4.6e+3]	[1.1e+3, 6.7e+3]	[1.1e+3, 6.8e+3]	[1.1e+3, 7.1e+3]	[1.2e+3, 7.0e+3]
Stock	4.2e−1	5.0e+0	4.8e+0	6.8e−1	4.3e−1	**3.5e−1**	**3.6e−1**	**3.8e−1**
	[3.7e−1, 4.8e−1]	[4.3e+0, 5.7e+0]	[4.5e+0, 5.2e+0]	[6.1e−1, 7.6e−1]	[3.8e−1, 4.9e−1]	[3.2e−1, 3.8e−1]	[3.2e−1, 4.0e−1]	[3.5e−1, 4.2e−1]
Triazines	3.0e−2	**2.2e−2**	**2.2e−2**	**2.2e−2**	**2.1e−2**	**2.1e−2**	**2.0e−2**	**2.0e−2**
	[1.4e−2, 4.7e−2]	[1.8e−2, 2.7e−2]	[1.5e−2, 2.9e−2]	[1.6e−2, 2.8e−2]	[1.3e−2, 2.9e−2]	[1.5e−2, 2.7e−2]	[1.4e−2, 2.6e−2]	[1.4e−2, 2.6e−2]

Table 3
Means and 95% confidence intervals for the test MSE obtained on large datasets using SVR with (i) the Gaussian kernel and (ii) the normalised asymptotic ELM kernel for different σ_w. Results which are not significantly different from the best result (underlined) are in bold font.

	Gaussian kernel	Normalised asymptotic ELM kernel						
		$\sigma_w = 1e{-3}$	$\sigma_w = 1e{-2}$	$\sigma_w = 1e{-1}$	$\sigma_w = 1e{+0}$	$\sigma_w = 1e{+1}$	$\sigma_w = 1e{+2}$	$\sigma_w = 1e{+3}$
Abalone	**4.4e+0**	5.2e+0	5.0e+0	**4.5e+0**	**4.4e+0**	**4.4e+0**	**4.4e+0**	**4.4e+0**
	[4.1e+0, 4.7e+0]	[4.8e+0, 5.6e+0]	[4.6e+0, 5.4e+0]	[4.2e+0, 4.7e+0]	[4.2e+0, 4.6e+0]	[4.0e+0, 4.8e+0]	[4.1e+0, 4.7e+0]	[4.2e+0, 4.7e+0]
Ailerons	**2.7e−8**	3.6e−8	3.6e−8	**2.7e−8**	**2.8e−8**	**2.8e−8**	**2.8e−8**	**2.8e−8**
	[2.6e−8, 2.9e−8]	[3.4e−8, 3.9e−8]	[3.4e−8, 3.8e−8]	[2.5e−8, 2.9e−8]	[2.6e−8, 3.1e−8]	[2.7e−8, 2.9e−8]	[2.6e−8, 3.0e−8]	[2.5e−8, 3.1e−8]
CompActs	**8.8e+0**	5.9e+1	5.2e+1	**8.9e+0**	1.2e+1	1.2e+1	1.2e+1	1.2e+1
	[8.3e+0, 9.3e+0]	[5.2e+1, 6.7e+1]	[4.6e+1, 5.8e+1]	[8.0e+0, 9.8e+0]	[1e+1, 1.4e+1]	[1.0e+1, 1.3e+1]	[1.0e+1, 1.3e+1]	[1.0e+1, 1.3e+1]
Elevators	**2.0e−6**	2.2e−6	2.2e-6	**2.0e−6**	**2.0e−6**	**2.0e−6**	**2.0e−6**	**2.0e−6**
	[2.0e−6, 2.1e−6]	[2.1e−6, 2.2e−6]	[2.1e−6, 2.3e−6]	[1.9e−6, 2.1e−6]	[2.0e−6, 2.1e−6]	[1.9e−6, 2.1e−6]	[1.9e−6, 2.1e−6]	[1.9e−6, 2.1e−6]

5.2. Results on real datasets

Tables 2 and 3 show the results obtained on the small and large datasets, respectively. For each dataset and kernel, the mean and 95% confidence interval of the MSE computed using 10-fold cross-test are given. For each dataset, the MSE is expressed in terms of original units. Best results are underlined and results which are not significantly different are in bold font. Here, a result is significantly different from the best result if its mean does not belong to the confidence interval of the best result. The best results are comparable to the results obtained in [18], except for Triazines which is not used in that work.

Tables 2 and 3 show that the results obtained with different values of σ_w are very similar. For six out of the eight datasets (Cancer, CPU, Triazines, Abalone, Ailerons and Elevators), the results are not significantly different when σ_w is equal to or larger than 10^{-1}. Moreover, these results are not significantly different from the results obtained using the standard Gaussian kernel, except for Triazines where the Gaussian kernel result is slightly worst. For CompActs, the result obtained with $\sigma_w = 10^{-1}$ is still not significantly different from the result obtained using the standard Gaussian kernel. Moreover, the results obtained using larger values of σ_w are only slightly worst. For Stock, the results obtained with $\sigma_w \geq 10$ are not significantly different from the best result. Moreover, the result obtained using the standard Gaussian kernel is slightly worst.

According to Tables 2 and 3, it seems that using the normalised asymptotic ELM kernel with e.g. $\sigma_w = 1$ or 10 allows obtaining results which are close if not identical to the results obtained using the Gaussian kernel. It means that the proposed kernel is in fact parameter-insensitive, in the sense that it is not necessary to tune the parameter σ_w to obtain good results. In practice, this observation reduces the number of meta-parameters from three to two: the regularisation constant C and the tube half-width ε.

Therefore, the proposed approach speeds up the meta-parameter optimisation step, so that non-linear problems can be tackled at the same computational cost than linear problems.

6. Conclusion

This paper shows that the ELM kernel can be successfully used for support vector regression. Using this kernel with SVR is in fact equivalent to optimising extreme learning machines using the ε-sensitive loss. Moreover, this paper proposes a new asymptotic view for the ELM kernel when the number of units grows to infinity. In that limit case, this paper also shows that the proposed kernel has an analytical form under certain assumptions on the hidden units of the extreme learning machine.

Experimental results suggest that the performances do not depend strongly on the only parameter of the ELM kernel. Indeed, performances which are close if not identical to the state-of-the-art performances are obtained as soon as the kernel parameter is chosen large enough, e.g. $\sigma_w = 1$ or 10. Therefore, the proposed approach reduces the number of meta-parameters to be tuned from three to two. As a matter of fact, the computational cost of non-linear SVR is strongly reduced with almost no consequence on the obtained performances.

Acknowledgements

The authors would like to thank Prof. Schrauwen (Ghent University, Belgium) for his useful remarks concerning the ELM kernel convergence, Mr. de Lannoy for his useful comments on this paper and Mr. Durvaux from BELNET-BEgrid for his precious technical help to get the experimental results on time. The authors also used the UCL CISM in order to get additional experimental results and would like to thank Mr. Van Renterghem for his technical help. Benoît Frénay would like to thank the FNRS which supported him during his stay at the Altoo University.

References

[1] V. Vapnik, The Nature of Statistical Learning, Springer, New York, 1995.
[2] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (3) (2004) 199–222.
[3] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.
[4] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, 2001.
[5] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters 9 (3) (1999) 293–300.
[6] J.A.K. Suykens, T.V. Gestel, J.D. Brabanter, B.D. Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific Publishing Co., Singapore, 2002.
[7] Q. Liu, Q. He, Z. Shi, Extreme support vector machine classifier, in: PAKDD, Lecture Notes in Computer Science2008, pp. 222–233.
[8] B. Frénay, M. Verleysen, Using SVMs with randomised feature spaces: an extreme learning approach, in: Proceedings of the 18th European Symposium on Artificial Neural Networks (ESANN)2010, pp. 315–320.
[9] G.-B. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classification, Neurocomputing, in Press, Corrected Proof.
[10] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[11] G.-B. Huang, C.-K. Siew, Extreme learning machine with randomly assigned RBF kernels, International Journal of Information Technology 11 (1) (2005) 16–24.
[12] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, 1998.
[13] G.-B. Huang, C.-K. Siew, Extreme learning machine with randomly assigned RBF kernels, International Journal of Information Technology 11 (1) (2005) 16–24.
[14] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062.
[15] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (16–18) (2008) 3460–3468.
[16] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 879–892.
[17] Y. Miche, A. Sorjamaa, A. Lendasse, OP-ELM: theory, experiments and a toolbox, ICANN, Lecture Notes in Computer Science, vol. 5163, Springer, 2008, pp. 145–154.
[18] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally-pruned extreme learning machine, IEEE Transactions on Neural Networks 21 (1) (2010) 158–162.
[19] C. Williams, Computing with infinite networks, in: Advances in Neural Information Processing Systems, MIT Press, 1996, pp. 295–301.
[20] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, 2001. Software available at ⟨http://www.csie.ntu.edu.tw/∼cjlin/libsvm⟩.
[21] UCI machine learning repository. ⟨http://archive.ics.uci.edu/ml/datasets.html⟩.
[22] ⟨http://www.liaad.up.pt/∼ltorgo/Regression/DataSets.html⟩.

Benoît Frénay received the Engineer's degree from the Université catholique de Louvain (UCL), Belgium, in 2007. He is now Ph.D. student at the UCL Machine Learning Group. His main research interests in machine learning include support vector machines, extreme learning, graphical models, classification, data clustering, probability density estimation and label noise.

Michel Verleysen was born in 1965 in Belgium. He received the M.S. and the Ph.D. degrees in Electrical Engineering from the Université catholique de Louvain (Belgium) in 1987 and 1992, respectively. He was an Invited Professor at the Swiss Ecole Polytechnique Fédérale de Lausanne (E.P.F.L.), Switzerland, in 1992, at the Université d'Evry Val d'Essonne (France) in 2001, and at the Université Paris 1—Panthéon-Sorbonne in 2002–2004. He is a former Research Director with the Belgian FNRS (Fonds National de la Recherche Scientifique) and a Professor at the Université catholique de Louvain. He is Editor-in-Chief of the Neural Processing Letters journal, Chairman of the Annual European Symposium on Artificial Neural Networks (ESANN) Conference, Associate Editor of the IEEE Transactions on Neural Networks journal, and member of the editorial board and program committee of several journals and conferences on neural networks and learning. He is the author or the co-author of about 200 scientific papers in international journals and books or communications to conferences with reviewing committee. He is the co-author of the scientific popularisation book on artificial neural networks in the series "Que Sais-Je?," in French. His research interests include machine learning, artificial neural networks, self-organisation, timeseries forecasting, non-linear statistics, adaptive signal processing, and high-dimensional data analysis.