Machine Learning (2019) 108:1951–1974
https://doi.org/10.1007/s10994-019-05797-z

Boosting as a kernel-based method

Aleksandr Y. Aravkin[1] · **Giulio Bottegal**[2] · **Gianluigi Pillonetto**[3]

Received: 17 October 2017 / Accepted: 20 April 2019 / Published online: 17 May 2019

Abstract
Boosting combines weak (biased) learners to obtain effective learning algorithms for classification and prediction. In this paper, we show a connection between boosting and kernel-based methods, highlighting both theoretical and practical applications. In the ℓ_2 context, we show that boosting with a weak learner defined by a kernel K is equivalent to estimation with a special *boosting kernel*. The number of boosting iterations can then be modeled as a continuous hyperparameter, and fit (along with other parameters) using standard techniques. We then generalize the boosting kernel to a broad new class of boosting approaches for general weak learners, including those based on the ℓ_1, hinge and Vapnik losses. We develop fast hyperparameter tuning for this class, which has a wide range of applications including robust regression and classification. We illustrate several applications using synthetic and real data.

Keywords Boosting · Weak learners · Kernel-based methods · Reproducing kernel Hilbert spaces · Robust estimation

1 Introduction

Boosting is a popular technique to construct learning algorithms (Schapire 2003). The basic idea is that any *weak learner*, i.e. algorithm that is only slightly better than guessing, can be used to build an effective learning mechanism that achieves high accuracy. Since the introduction of boosting in Schapire's seminal work (Schapire 1990), numerous variants have been proposed for regression, classification, and specific applications including semantic

Editor: Thomas Gärtner.

✉ Aleksandr Y. Aravkin
saravkin@uw.edu

Giulio Bottegal
g.bottegal@tue.nl

Gianluigi Pillonetto
giapi@dei.unipd.it

[1] Department of Applied Mathematics, University of Washington, Seattle, WA 98195-4322, USA

[2] Department of Electrical Engineering, TU Eindhoven, 5600 MB Eindhoven, The Netherlands

[3] Department of Information Engineering, University of Padova, 35131 Padua, Italy

learning and computer vision (Schapire and Freund 2012; Viola and Jones 2001; Temlyakov 2000; Tokarczyk et al. 2015; Bissacco et al. 2007; Cao et al. 2014). In particular, in the context of classification, *LPBoost*, *LogitBoost* (Friedman et al. 2000), *Bagging and Boosting* (Lemmens and Croux 2006) and *AdaBoost* (Freund and Schapire 1997) have become standard tools, the latter having being recognized as the best off-the-shelf binary classification method (Breiman 1998; Zhang 2003; Zhu et al. 2009). In recent years, AdaBoost has been extended in several ways. In particular, in Cortes et al. (2014, 2017), AdaBoost has been used in the context of neural networks and deep decision trees, while the method introduced in Rätsch and Warmuth (2005) incorporates estimates of the achievable maximum margin between two classes. In Gao and Koller (2011), a multiclass method based on boosting and the hinge loss is discussed. Applications of the boosting principle are also found in decision tree learning (Tu 2005) and distributed learning (Fan et al. 1999), and classification tasks (Freund et al. 1999). For regression problems, *AdaBoost.RT* (Solomatine and Shrestha 2004; Avnimelech and Intrator 1999) and ℓ_2 *Boost* (Bühlmann and Yu 2003; Tutz and Binder 2007; Champion et al. 2014; Oglic and Gärtner 2016) are the most prominent boosting algorithms. In this context, boosting the weak learner usually corresponds to a kernel-based estimator with a heavily weighted regularization term. The fit on the training set improves at each iteration, and the procedure will over-fit as it continues (Bühlmann and Hothorn 2007). To avoid this, stopping criteria based on model complexity are used. Hurvich et al. (1998) propose a modified version of Akaike's information criterion (AIC); Hansen and Yu (2001) use the principle of minimum description length (MDL), and Bühlmann and Yu (2003) implement five-fold cross validation.

In this paper, we first focus on ℓ_2 boosting and consider linear weak learners induced by the combination of a quadratic loss and a regularizer induced by a kernel K. We show that the resulting boosting estimator is equivalent to estimation with a special *boosting kernel* that depends on K, as well as on the regression matrix, noise variance, and hyperparameters. This viewpoint leads to both greater generality and better computational efficiency. In particular, the number of boosting iterations ν is a continuous hyperparameter of the boosting kernel, and can be tuned by standard fast hyper-parameter selection techniques including SURE, generalized cross validation, and marginal likelihood (Hastie et al. 2001a). In Sect. 5, we show that tuning ν is far more efficient than applying boosting iterations, and non-integer values of ν can improve performance.

We then generalize to a wider class of problems, including robust regression, by combining the boosting kernel with piecewise linear quadratic (PLQ) loss functions (e.g. ℓ_1, Vapnik, Huber). The computational burden of standard boosting for general loss functions is high,[1] but the boosting kernel makes the approach tractable. We also use the boosting kernel to solve regularization problems in reproducing kernel Hilbert spaces (RKHSs), e.g. to solve classification formulations that use the hinge loss.

The organization of the paper is as follows. After a brief overview of boosting in regression and classification, we develop the main connection between boosting and kernel-based methods for finite-dimensional inverse problems in Sect. 2. Consequences of this connection are presented in Sect. 3 also discussing the case of function estimation from direct noisy data. In Sect. 4 we combine the boosting kernel with PLQ penalties to develop a new class of boosting algorithms for regression and classification in RKHSs. We also discuss new RKHSs induced by a generalized class of boosting kernels which allow to connect learning rates and consistency analysis of boosting with those related to kernel machines. In Sect. 5

[1] The estimator at each iteration is a linear function of the data only for the ℓ_2 loss.

we show numerical results for several experiments involving the boosting kernel. We end with discussion and conclusions in Sect. 6.

2 Boosting as a kernel-based method

In this section, we give a basic overview of boosting, and present the boosting kernel.

2.1 Boosting: notation and overview

Assume we are given a model $g(\theta)$ for some observed data $y \in \mathbb{R}^n$, where $\theta \in \mathbb{R}^m$ is an unknown parameter vector. Suppose our estimator $\hat{\theta}$ for θ minimizes some objective that balances variance with bias. In the boosting context, the objective is designed to provide a *weak estimator*, i.e. one with low variance in comparison to the bias.

Given a loss function $\mathcal{V}$ and a kernel matrix $K \in \mathbb{R}^{m\times m}$, the weak estimator can be defined by minimizing the regularized formulation

$$\hat{\theta} := \arg\min_{\theta} \left\{ J(\theta; y) := \mathcal{V}(y - g(\theta)) + \gamma \theta^T K^{-1} \theta \right\}, \tag{1}$$

where the regularization parameter γ is large and leads to over-smoothing. Boosting uses this weak estimator iteratively, as detailed below. The predicted data for an estimator $\hat{\theta}$ are denoted by $\hat{y} = g(\hat{\theta})$.

Boosting scheme

1. Set $\nu = 1$ and obtain $\hat{\theta}(1)$ and $\hat{y}(1) = g(\hat{\theta}(1))$ using (1);
2. Solve (1) using the current residuals as data vector, i.e. compute

$$\hat{\theta}(\nu) = \operatorname*{argmin}_{\theta} \; J(\theta; y - \hat{y}(\nu)),$$

   and set the new predicted output to

$$\hat{y}(\nu + 1) = \hat{y}(\nu) + g(\hat{\theta}(\nu)).$$

3. Increase ν by 1 and repeat step 2 for a prescribed number of iterations.

2.2 Using regularized least squares as weak learner

Suppose data y are generated according to

$$y = U\theta + v, \quad v \sim \mathcal{N}(0, \sigma^2 I), \tag{2}$$

where U is a known regression matrix of full column rank. The components of v are independent random variables, mean zero and variance σ^2.

We now use a quadratic loss to define the regularized weak learner. Let λ denote the kernel scale factor and set $\gamma = \sigma^2/\lambda$ so that (1) becomes

$$\hat{\theta} = \arg\min_{\theta} \|y - U\theta\|^2 + \frac{\sigma^2}{\lambda} \theta^T K^{-1} \theta \tag{3}$$

$$= \lambda K U^T (\lambda U K U^T + \sigma^2 I)^{-1} y. \tag{4}$$

Defining

$$P_\lambda := \lambda U K U^T, \quad f := U\theta, \tag{5}$$

we have the predicted data $\hat{y} = U\hat{\theta}$ given by

$$\hat{y} = \arg\min_f \|y - f\|^2 + \sigma^2 f^T P_\lambda^{-1} f \tag{6}$$

$$= P_\lambda (P_\lambda + \sigma^2 I)^{-1} y. \tag{7}$$

In (3) and (6), we have assumed K and P_λ invertible. If this does not hold, both of these problems can be easily reformulated using pseudoinverses, e.g. see Aravkin et al. (2017, Appendix), while (4) and (7) remain the same. All the derivations obtained below rely on (7) so that invertibility of K and P_λ is never required in what follows.

The following well-known connection (Wahba 1990) between (3) and Bayesian estimation is useful for theoretical development. Assume that θ and v are independent Gaussian random vectors with priors

$$\theta \sim \mathcal{N}(0, \lambda K), \quad v \sim \mathcal{N}(0, \sigma^2 I).$$

Then, given any regression matrix U and (possibly singular) covariance K, (4) and (7) provide the well-known expressions of the minimum variance estimates of θ and $U\theta$ conditional on the data y, e.g. see Anderson and Moore (1979) for derivations through projections onto subspaces spanned by random variables. In view of this, we refer to diagonal values of K as the *prior variances* of θ.

2.3 The boosting kernel

Define

$$S_\lambda = P_\lambda (P_\lambda + \sigma^2 I)^{-1}. \tag{8}$$

Fixing a small λ, the predicted data obtained by the weak kernel-based learner is

$$\hat{y}(1) = S_\lambda y,$$

where $\hat{y}(1)$ indicates that we performed one boosting iteration. According to the scheme specified in Sect. 2.1, as boosting iteration v increases, the estimate is refined as follows:

$$\begin{aligned}
\hat{y}(2) &= S_\lambda y + S_\lambda (I - S_\lambda) y \\
\hat{y}(3) &= S_\lambda y + S_\lambda (I - S_\lambda) y + S_\lambda (I - S_\lambda)^2 y \\
&\vdots \\
\hat{y}(v) &= S_\lambda \sum_{i=0}^{v-1} (I - S_\lambda)^i \, y.
\end{aligned} \tag{9}$$

We now show that the $\hat{y}(v)$ are kernel-based estimators from a special *boosting kernel*, which plays a key role for subsequent developments.

Proposition 1 *The quantity $\hat{y}(v)$ is a kernel-based estimator*

$$\hat{y}(v) = S_{\lambda,v} y = P_{\lambda,v} (P_{\lambda,v} + \sigma^2 I)^{-1} y,$$

where $P_{\lambda,\nu}$ is the boosting kernel defined by

$$\begin{aligned} P_{\lambda,\nu} &= \sigma^2 \left(I - P_\lambda \left(P_\lambda + \sigma^2 I \right)^{-1} \right)^{-\nu} - \sigma^2 I \\ &= \sigma^2 \left(I - S_\lambda \right)^{-\nu} - \sigma^2 I. \end{aligned} \tag{10}$$

Proof First note that S_λ satisfies

$$S_\lambda = P_\lambda \left(P_\lambda + \sigma^2 I \right)^{-1} = I - \sigma^2 \left(P_\lambda + \sigma^2 I \right)^{-1}. \tag{11}$$

This follows from adding the term $\sigma^2 \left(P_\lambda + \sigma^2 \right)^{-1}$ to (8) and observing that expression reduces to I. Plugging in the expression (10) for $P_{\lambda,\nu}$ into the right hand side of expression (11) for $S_{\lambda,\nu}$, we have

$$\begin{aligned} S_{\lambda,\nu} &= I - \sigma^2 \left(P_{\lambda,\nu} + \sigma^2 I \right)^{-1} \\ &= I - \sigma^2 \left(\sigma^2 \left(I - S_\lambda \right)^{-\nu} \right)^{-1} \\ &= I - \left(I - S_\lambda \right)^{\nu} \\ &= S_\lambda \sum_{i=0}^{\nu-1} \left(I - S_\lambda \right)^i, \end{aligned}$$

exactly as required by (9). □

In Bayesian terms, for a given ν, boosting returns the minimum variance estimate of the noiseless output f conditional on y, if f and v are independent Gaussian random vectors with priors

$$f \sim \mathcal{N}(0, P_{\lambda,\nu}), \quad v \sim \mathcal{N}(0, \sigma^2 I). \tag{12}$$

3 Consequences

In this section, we use Proposition 1 to gain new insights on boosting and to develop a fast method for tuning the boosting parameter ν.

3.1 Insights on the nature of boosting

We first derive a new representation of the boosting kernel $P_{\lambda,\nu}$ via a change of coordinates. Starting from (10), we have

$$\begin{aligned} P_{\lambda,\nu} &= \sigma^2 \left(I - P_\lambda \left(P_\lambda + \sigma^2 I \right)^{-1} \right)^{-\nu} - \sigma^2 I \\ &= \sigma^2 \left(I - \left(I - \sigma^2 \left(P_\lambda + \sigma^2 I \right)^{-1} \right) \right)^{-\nu} - \sigma^2 I \\ &= \frac{\sigma^2}{(\sigma^2)^\nu} \left(P_\lambda + \sigma^2 I \right)^{-\nu} - \sigma^2 I \end{aligned}$$

where we used (11) to move get from the first to the second line above. Now, let $V D V^T$ be the SVD of $U K U^T$. Then, we obtain

$$P_{\lambda,\nu} = \frac{\sigma^2}{(\sigma^2)^\nu}\left(\lambda U K U^T + \sigma^2 I\right)^\nu - \sigma^2 I$$
$$= \sigma^2 V\left[\left(\frac{\lambda D + \sigma^2 I}{\sigma^2}\right)^\nu - I\right] V^T \tag{13}$$

and the predicted output can be rewritten as

$$\hat{y}(\nu) = V\left(I - \sigma^{2\nu}\left(\lambda D + \sigma^2 I\right)^{-\nu}\right) V^T y.$$

In coordinates $z = V^T y$, the estimate of each component of z is

$$\hat{z}_i(\nu) = \left(1 - \frac{\sigma^{2\nu}}{\left(\lambda d_i^2 + \sigma^2\right)^\nu}\right) z_i, \tag{14}$$

and corresponds to the regularized least squares estimate induced by a diagonal kernel with (i, i) entry

$$\sigma^2\left(\frac{\lambda d_i^2}{\sigma^2} + 1\right)^\nu - \sigma^2. \tag{15}$$

In Bayesian terms, (15) is the prior variance assigned by boosting to the noiseless output $V^T U\theta$.

Equation (15) shows that boosting builds a kernel on the basis of the output signal-to-noise ratios $SNR_i = \frac{\lambda d_i^2}{\sigma^2}$, which then enter $\left(\frac{\lambda d_i^2}{\sigma^2} + 1\right)^\nu$. All diagonal kernel elements with $d_i > 0$ grow to ∞ as ν increases; therefore asymptotically, data will be perfectly interpolated but with growth rates controlled by the SNR_i. If SNR_i is large, the prior variance increases quickly and after a few iterations the estimator is essentially unbiased along the ith direction. If SNR_i is close to zero, the ith direction is treated as though affected by ill-conditioning, and a large ν is needed to remove the regularization on $\hat{z}_i(\nu)$.

This perspective makes it clear when boosting can be effective. When solving inverse problems, θ in (2) often represents the unknown input to a linear system whose impulse response defines the regression matrix U. For simplicity, assume that the kernel K is set to the identity matrix, so that the weak learner (3) becomes ridge regression and the d_i^2 in (15) reflect the power content of the impulse response at different frequencies. Then, *boosting can outperfom standard ridge regression if the system impulse response and input share a similar power spectrum*. Under this condition, boosting can inflate the prior variances (15) along the right directions. For instance, if the impulse response energy is located at low frequencies, as ν increases boosting will amplify the low pass nature of the regularizer. This can significantly improve the estimate if the input is also low pass. A numerical example is given in Sect. 3.3, while Sect. 3.4 shows the consequences of our findings for function reconstruction (estimation of θ from direct noisy measurements).

3.2 Hyperparameter estimation

In the classical scheme described in Sect. 2.1, ν is an iteration counter that only takes integer values, and the boosting scheme is sequential: to obtain the estimate $\hat{y}(m)$, one has to solve m optimization problems. Using (10) and (13), we can interpret ν as a kernel hyperparameter, and let it take real values. In the following we estimate both the scale factor λ and ν from the data, and restrict the range of ν to $\nu \geq 1$.

The resulting boosting approach estimates (λ, ν) by minimizing fit measures such as cross validation or SURE (Hastie et al. 2001a). In particular, this accelerates the tuning procedure, as it requires solving a single problem instead of multiple boosting iterations. Consider estimating (λ, ν) using the SURE method. Given σ^2 (e.g. using an unbiased estimator), such approach is based on optimization of the objective

$$\|y - \hat{y}(\nu)\|^2 + 2\sigma^2 \text{trace}(S_{\lambda,\nu}). \tag{16}$$

After performing just a single SVD, then interpreting ν as a kernel real hyperparameter, straightforward computations show that SURE estimates are given by

$$(\hat{\lambda}, \hat{\nu}) = \arg \min_{\lambda \geq 0, \nu \geq 1} \sum_{i=1}^{n} \frac{z_i^2 \sigma^{4\nu}}{\left(\lambda d_i^2 + \sigma^2\right)^{2\nu}} + 2\sigma^2 n - \sum_{i=1}^{n} \frac{2\sigma^{2\nu+2}}{(\lambda d_i^2 + \sigma^2)^{\nu}}, \tag{17}$$

which is a smooth 2-variable problem over a box, and can be easily optimized.

We can also extract some useful information on the nature of the optimization problem (17). In fact, denoting J the objective, we have

$$\begin{aligned} \frac{\partial J}{\partial \nu} &= 2 \sum_{i=1}^{n} \log(\alpha_i) z_i^2 \alpha_i^{2\nu} - 2\sigma^2 \sum_{i=1}^{n} \log(\alpha_i) \alpha_i^{\nu} \\ &= 2 \sum_{i=1}^{n} \log(\alpha_i) \alpha_i^{\nu} (z_i^2 \alpha_i^{\nu} - \sigma^2), \end{aligned} \tag{18}$$

where we have defined $\alpha_i := \frac{\sigma^2}{\lambda d_i^2 + \sigma^2}$. Simple considerations on the sign of the derivative then show that

– if

$$\lambda < \min_{i=1,\ldots,n} \frac{z_i^2 - \sigma^2}{d_i^2}, \tag{19}$$

then $\hat{\nu} = +\infty$. This means that we have chosen a learner so weak that SURE suggests an infinite number of boosting iterations as optimal solution;

– if

$$\lambda > \max_{i=1,\ldots,n} \frac{z_i^2 - \sigma^2}{d_i^2}, \tag{20}$$

then $\hat{\nu} = 1$. This means that the weak learner is instead so strong that SURE suggests not to perform any boosting iterations.

3.3 Numerical illustration: ridge regression

Consider (2), where $\theta \in \mathbb{R}^{50}$ represents the input to a discrete-time linear system. In particular, the signal is taken from Wahba (1990) and displayed in Fig. 1 (thick red line). The system is represented by the regression matrix $U \in \mathbb{R}^{200 \times 50}$ whose components are realizations of either white noise or low pass filtered white Gaussian noise with normalized band [0, 0.95]. The measurement noise is white and Gaussian, with variance assumed known and set to the variance of the noiseless output divided by 10.

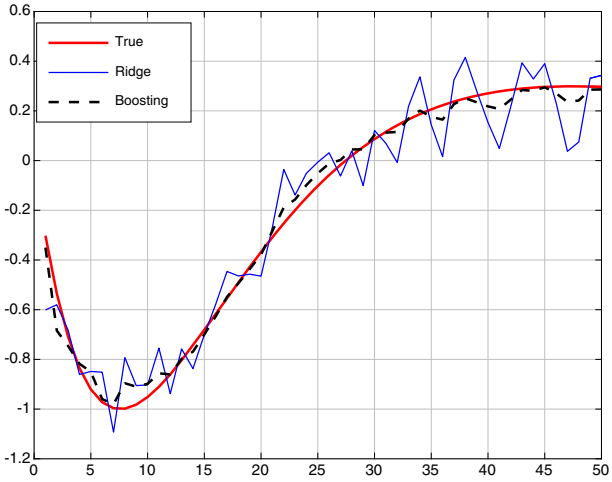

Fig. 1 True signal (thick red line), Ridge estimate (solid blue) and Boosting estimate (dashed black) obtained in the first Monte Carlo run. The system impulse response is a low pass signal (Color figure online)

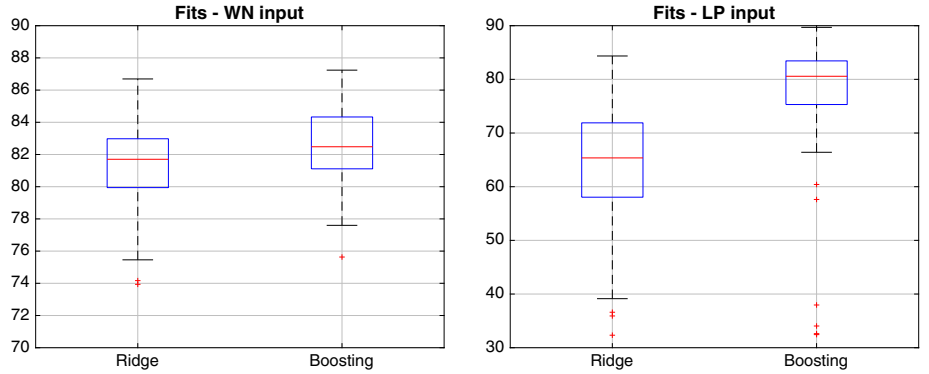

Fig. 2 Boxplot of the percentage fits obtained by Ridge regression and Boosting, using SURE to estimate hyperparameters; system impulse response is white noise (left) and low pass (right)

We use a Monte Carlo of 100 runs to compare the following two estimators

- **Boosting** boosting estimator with K set to the identity matrix and with (λ, ν) estimated using the SURE strategy (16).
- **Ridge** ridge regression (which corresponds to boosting with ν fixed to 1).

Figure 2 displays the box plots of the 100 percentage fits of θ, $100\left(1-\frac{\|\theta-\hat{\theta}\|}{\|\theta\|}\right)$, obtained by **Boosting** and **Ridge**. When the entries of U are white noise (left panel) one can see that the two estimators have similar performance. When the entries of U are filtered white noise (right panel) **Boosting** performs significantly better than **Ridge**. Furthermore, 36 out of the 100 fits achieved by **Boosting** under the white noise scenario are lower than those obtained adopting a low pass U, even though the conditioning of latter problem is much worse. The unknown θ represents a smooth signal. In Bayesian terms, setting K to the identity matrix corresponds to modeling it as white noise, which is a poor prior. If the nature of U is low pass, the energy of the d_i^2 are more concentrated at low frequencies. So, as ν increases, **Boosting** can inflate the prior variances associated to the low-frequency components of θ. The prior

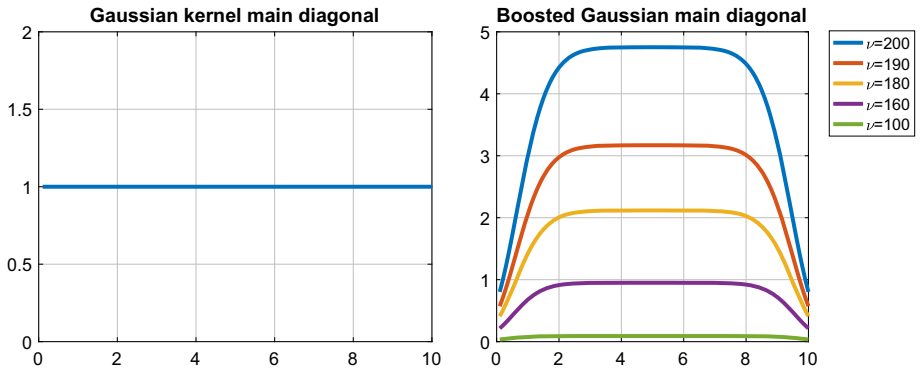

Fig. 3 Diagonal elements of the Gaussian kernel (left) and of the boosted Gaussian kernel as a function of boosting iterations ν (right). Under a Bayesian perspective, the right panel illustrates how boosting changes the prior variances of the components of θ. Exploiting boosting, the stationarity feature of the Gaussian kernel is lost and more variability is allowed to the signal (in particular in its central part) as ν increases

variances associated to high-frequencies induce low SNR_i, so that they increase slowly with ν. This does not happen in the white noise case, since the random variables d_i^2 have similar distributions. Hence, the original white noise prior for θ can be significantly refined only in the low pass context: it is reshaped so as to form a regularizer, inducing more smoothness. Figure 1 shows this effect by plotting estimates from **Ridge** and **Boosting** in a Monte Carlo run where U is low pass.

3.4 Numerical illustration: Gaussian kernel

Consider now a situation where direct measurements of θ are available, so that U is the identity matrix. This problem can be interpreted as function estimation where each θ_i corresponds to the function f evaluated at the input location x_i. The observation model is $y_i = f(x_i) + v_i$.

In function estimation, the kernel is often used to include smoothness information on f. The radial basis class of kernes is widely used, with the Gaussian kernel a popular choice:

$$\mathcal{K}(x, a) = \exp(-|x - a|^2), \quad |\cdot| = \text{Euclidean norm}. \tag{21}$$

This kernel describes functions known to be somewhat regular and, under a Bayesian perspective, it models f as a stationary Gaussian process. These features are illustrated in the left panel of Fig. 3, which shows the constant variance of f, and of Fig. 4, which shows normal and independent realizations drawn from the covariance (21). These realizations are examples of functions *a priori* preferred by the Gaussian kernel, before seeing data y.

We now consider the following key question: when can we expect better results using the Gaussian kernel as weak learner, compared to classical least squares estimates regularized with the same kernel? We can answer this question using the proposed boosting kernel.

Assume that $x_i = i/10$, $i = 1, \ldots, 100$, and that the noise variance is $\sigma^2 = 0.1^2$. Take K to be the 100×100 matrix with (i, j)-entry $K_{i,j} = \mathcal{K}(x_i, x_j)$ where $\mathcal{K}$ is the Gaussian kernel (21).[2] Furthermore, let the weak learner be λK with $\lambda = 1e - 4$. The effect of boosting on the Gaussian kernel can now be understood by computing first the SVD of K (see left and right panels of Fig. 5) and then the boosting kernel (13) for different values of ν. Using the Bayesian perspective, the right panel of Fig. 3 shows the (linearly interpolated) variances of

[2] Matrices K constructed this way are called Gram or kernel matrices in the machine learning literature.

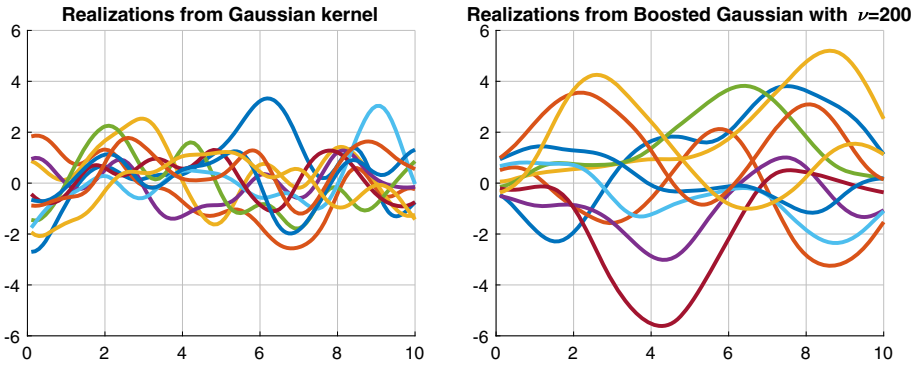

Fig. 4 Realizations from the Gaussian kernel (left) and the boosted Gaussian kernel with $\nu = 200$ (right)

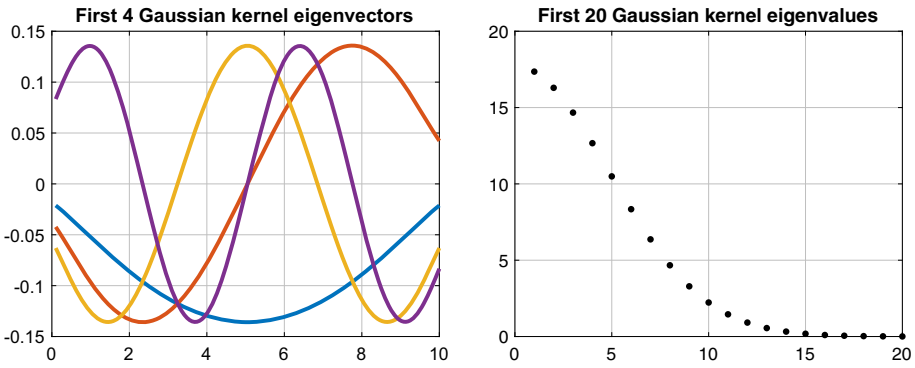

Fig. 5 Some eigenvectors and eigenvalues of the kernel matrix obtained from the Gaussian kernel (21) on the input locations $x_i = i/10, i = 1, \ldots, 100$. More regular eigenvectors are associated to larger eigenvalues

f as a function of the x_i. While the Gaussian kernel describes stationary processes (left panel of Fig. 3), its boosted version now models nonstationary phenomena. In particular, more variability is allowed to the signal in its central part as ν increases. The right panel of Fig. 4 shows normal realizations from the boosting kernel for $\nu = 200$. In comparison with the profiles in the left panel, more regular functions are now preferred, because we see from (13) that, as ν increases, boosting gives more weight variance to the first eigenvectors, whose power is located at low frequencies, see Fig. 5. The candidates also tend to assume larger values around the central part of the input space since, overall, the prior variance is made larger.

This example is simple but significant and can be repeated under any experimental condition using any kernel of interest. It shows how the boosting kernel and its Bayesian interpretation can be used to get a clear picture of the function class that can be reconstructed by boosting with high fidelity.

4 Boosting algorithms for general loss functions and RKHSs

In this section, we combine the boosting kernel with piecewise linear-quadratic (PLQ) losses to obtain tractable algorithms for more general regression and classification problems. We also consider estimation in Reproducing Kernel Hilbert (RKHS) spaces.

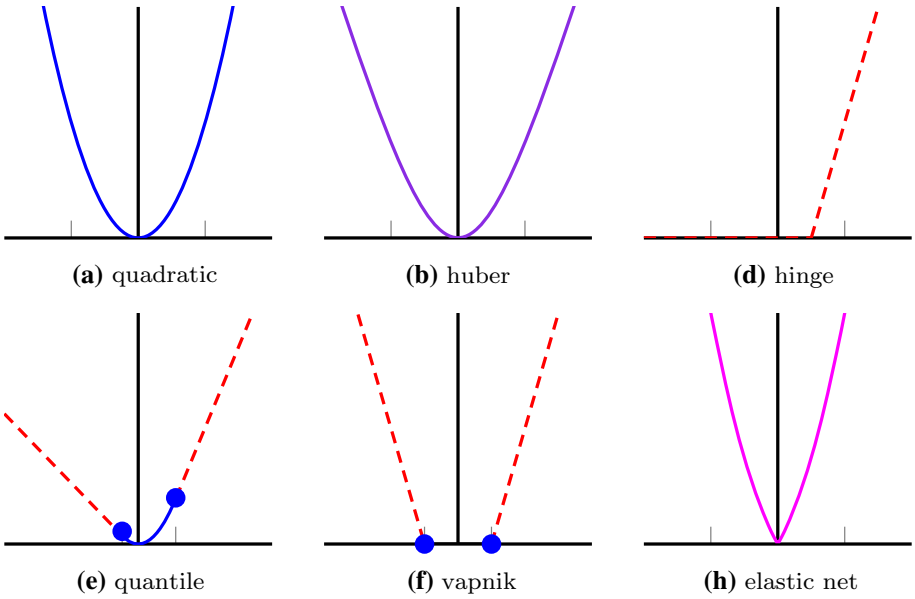

Fig. 6 Six common piecewise-linear quadratic losses

4.1 Boosting kernel-based estimation with general loss functions

We consider a kernel-based weak learner (1), based on a general (convex) penalty $\mathcal{V}$. Important examples include Vapnik's epsilon insensitive loss (Fig. 6f) used in support vector regression (Vapnik 1998; Hastie et al. 2001b; Schölkopf et al. 2000; Schölkopf and Smola 2001), hinge loss (Fig. 6d) used for classification (Evgeniou et al. 2000; Pontil and Verri 1998; Schölkopf et al. 2000), Huber and quantile huber (Fig 6b, e), used for robust regression (Huber 2004; Maronna et al. 2006; Bube and Nemeth 2007; Zou and Yuan 2008; Koenker and Geling 2001; Koenker 2005; Aravkin et al. 2014), and elastic net (Fig. 6f), a sparse regularizer that also finds correlated predictors (Zou and Hastie 2005; Li and Lin 2010; De Mol et al. 2009). The resulting boosting scheme is computationally expensive: $\hat{y}(\nu)$ requires solving a sequence of ν optimization problems, each of which must be solved iteratively. In addition, since the estimators $\hat{y}(\nu)$ are no longer linear, deriving a boosting kernel is no longer straightforward.

We combine general loss $\mathcal{V}$ with the regularizer induced by the boosting kernel from the linear case to define a new class of kernel-based boosting algorithms. More specifically, given a kernel K, let VDV^T be the SVD of UKU^T. First, assume $P_{\lambda,\nu}$ invertible. Then, the boosting output estimate $\hat{y}(\nu)$ is

$$\begin{aligned}\hat{y}(\nu) &= \arg\min_f \ \mathcal{V}(y-f) + \sigma^2 f^T P_{\lambda,\nu}^{-1} f \\ &= \arg\min_f \mathcal{V}(y-f) + f^T V \left[\left(\frac{\lambda D + \sigma^2 I}{\sigma^2} \right)^{\nu} - I \right]^{-1} V^T f, \end{aligned} \tag{22}$$

where the last line is obtained using (13). The solution depends on λ and σ^2 only through the ratio $\gamma = \sigma^2/\lambda$. Problem (22) is strongly convex since the loss $\mathcal{V}$ is convex and $P_{\lambda,\nu}$ has been assumed invertible.

Next, if $P_{\lambda,\nu}$ is not invertible, we can use (13) to obtain the factorization

$$P_{\lambda,\nu} = \sigma^2 A_{\lambda,\nu} A_{\lambda,\nu}^T,$$

where $A_{\lambda,\nu}$ is full column rank and contains the columns of the matrix

$$A_{\lambda,\nu} = V \left[\left(\frac{\lambda D + \sigma^2 I}{\sigma^2} \right)^{\nu} - I \right]^{1/2}$$

associated to the $d_i > 0$. The evaluation of $A_{\lambda,\nu}$ for different λ and ν is efficient. Now the output estimate is $\hat{y}(\nu) = A_{\lambda,\nu}\hat{a}(\nu)$, where $\hat{a}(\nu)$ solves the strongly convex optimization problem

$$\hat{a}(\nu) = \arg\min_a \left\{ \mathcal{V}(y - A_{\lambda,\nu} a) + a^T a \right\}. \tag{23}$$

The estimate of θ is given by $\hat{\theta} = U^{\dagger}\hat{y}(\nu)$, where $U^{\dagger}$ is the pseudo-inverse of U.

The new class of boosting kernel-based estimators defined by (23) keeps the advantages of boosting in the quadratic case. In particular, the kernel structure can decrease bias along directions less exposed to noise. The use of a general loss $\mathcal{V}$ allows a range of applications, with e.g. penalties such as Vapnik and Huber, guarding against outliers in the training set. Finally, the algorithm has clear computational advantages over the classic scheme described in Sect. 2.1. Whereas in the classic approach, $\hat{y}(m)$ require solving m optimization problems, in the new approach, given any positive λ and $m \geq 1$, the prediction $\hat{y}(m)$ is obtained by solving the convex optimization problem (22). This is illustrated in Sect. 5.

4.2 New boosting algorithms in RKHSs

We extend the new boosting algorithms to regularization in RKHSs. Consider reconstructing a function from n sparse and noisy data y_i collected on input locations x_i taking values on the input space $\mathcal{X}$. We want the function estimator to assume values in infinite-dimensional spaces, introducing suitable smoothness regularization to circumvent ill-posedness. We use $\mathcal{K}$ denote a kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ to capture smoothness properties of the unknown function. We can then use ℓ_2 Boost, with weak learner

$$\operatorname*{argmin}_{f \in \mathcal{H}} \sum_{i=1}^{n} \mathcal{V}_i(y_i - f(x_i)) + \gamma \|f\|_{\mathcal{H}}^2, \tag{24}$$

where $\mathcal{V}_i$ is a generic convex loss and $\mathcal{H}$ is the RKHS induced by $\mathcal{K}$ with norm denoted by $\|\cdot\|_{\mathcal{H}}$. From the representer theorem of Schölkopf et al. (2001), the solution of (24) is $\sum_{i=1}^{n} \hat{c}_i \mathcal{K}(x_i, \cdot)$ where the $\hat{c}_i$ are the components of the column vector

$$\operatorname*{argmin}_{c \in \mathbb{R}^n} \sum_{i=1}^{n} \mathcal{V}_i(y_i - K_{i,\cdot} c) + \gamma c^T K c, \tag{25}$$

and K is the kernel (Gram) matrix, with $K_{i,j} = \mathcal{K}(x_i, x_j)$ and $K_{i,\cdot}$ is the ith row of K. Using (25), we extend the boosting scheme from Sect. 2.1 with (24) as the weak learner. In particular, repeated applications of the representer theorem ensure that, for any value of the iteration counter ν, the corresponding function estimate belongs to the subspace spanned by the n kernel sections $\mathcal{K}(x_i, \cdot)$. Hence, ℓ_2 Boosting in RKHS can be summarized as follows.

Boosting scheme in RKHS

1. Set $\nu = 1$. Solve (25) to obtain $\hat{c}$ and $\hat{f}$ for $\nu = 1$, call them $\hat{c}(1)$ and $\hat{f}(\cdot, 1)$.

2. Update c by solving (25) with the current residuals as the data vector:

$$\hat{c}(\nu+1) = \hat{c}(\nu) + \underset{c\in\mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} \mathcal{V}_i(y_i - K_i\hat{c}(\nu) - K_i c) + \gamma c^T K c,$$

and set the new estimated function to

$$\hat{f}(\cdot, \nu+1) = \sum_{i=1}^{n} \hat{c}_i(\nu+1)\mathcal{K}(x_i, \cdot).$$

3. Increase ν by 1 and repeat step 2 for a prescribed number of iterations.

There is a fundamental computational drawback related to this scheme which we have already encountered in the previous sections. To obtain $\hat{f}(\cdot, \nu)$ we need to solve ν optimization problems, each of them requiring an iterative procedure. Now, we define a new computationally efficient class of regularized estimators in RKHS. The idea is to obtain the expansion coefficients of the function estimate through the new boosting kernel. Letting $\gamma = \sigma^2/\lambda$ and $P_\lambda = \lambda K$, with K the kernel matrix, define the boosting kernel $P_{\lambda,\nu}$ as in (13). Then, we can first solve

$$\hat{b}(\nu) = \arg\min_b \left\{ \mathcal{V}(y - P_{\lambda,\nu} b) + b^T P_{\lambda,\nu} b \right\}, \tag{26}$$

with $\mathcal{V}$ defined as the sum of the $\mathcal{V}_i$. Then, we compute

$$\tilde{c} = K^{\dagger}\tilde{y}(\nu) \quad \text{with} \quad \tilde{y}(\nu) = P_{\lambda,\nu}\hat{b}(\nu),$$

and the estimated function becomes

$$\hat{f}(\cdot, \nu) = \sum_{i=1}^{n} \tilde{c}_i(\nu)\mathcal{K}(x_i, \cdot).$$

Note that the weights $\tilde{c}(\nu)$ coincide with $\hat{c}(\nu)$ only when the $\mathcal{V}_i$ are quadratic. Nevertheless, given any loss, (26) preserves all advantages of boosting outlined in the linear case. Furthermore, as in the finite-dimensional case, given any ν and kernel hyperparameter, the estimator (26) can compute $\tilde{c}(\nu)$ by solving a single problem, rather than iterating the boosting scheme.

Classification with the hinge loss Another advantage related to the use of the boosting kernel w.r.t. the classical boosting scheme arises in the classification context. Classification tries to predict one of two output values, e.g. 1 and -1, as a function of the input. ℓ_2 Boost could be used using the residual $y_i - f(x_i)$ as misfit, e.g. equipping the weak learner (24) with the quadratic or the ℓ_1 loss. However, in this context one often prefers to use the margin $m_i = y_i f(x_i)$ on an example (x_i, y_i) to measure how well the available data are classified. For this purpose, support vector classification is widely used (Schölkopf and Smola 2002). It relies on the hinge loss

$$\mathcal{V}_i(y_i, f(x_i)) = |1 - y_i f(x_i)|_+ = \begin{cases} 0, & m > 1 \\ 1 - m, & m \le 1 \end{cases}, \quad m = y_i f(x_i),$$

which gives a linear penalty when $m < 1$. Note that this loss assumes $y_i \in \{1, -1\}$. However, the classical boosting scheme applies the weak learner (24) repeatedly, and **residuals will not be binary** for $\nu > 1$. This means that ℓ_2 Boost cannot be used for the hinge loss.

This limitation does not affect the new class of boosting-kernel based estimators: support vector classification can be boosted by plugging in the hinge loss into (26):

$$\hat{b}(\nu) = \arg\min_{b} \sum_{i=1}^{n} |1 - y_i [P_{\lambda,\nu} b]_i|_+ + b^T P_{\lambda,\nu} b, \tag{27}$$

where we have used $[P_{\lambda,\nu} b]_i$ to denote the ith component of $P_{\lambda,\nu} b$.

4.3 More general boosting kernels class and convergence rate issues

So far, we have introduced boosting kernels $P_{\lambda,\nu}$ of the type (13) which correspond to symmetric positive-semidefinite matrices. In the previous subsections we have shown that such kernels allow the reconstruction of functions over general domains, and, using the representer theorem, estimation is reduced to solving the finite-dimensional Problem (26) which is regularized by the boosting kernel $P_{\lambda,\nu}$. Below, we discuss how to directly define boosting kernels over the entire domain $\mathcal{X} \times \mathcal{X}$. This generalized class, denoted below by $\mathcal{P}_{\lambda,\nu}$, allows new RKHSs and a new kernel-based perspective on boosting learning rates.

Assume we are given a kernel $\mathcal{K}$ and let μ_x be a nondegenerate σ-finite measure on $\mathcal{X}$. Define also the integral operator associated to the kernel as follows

$$L_{\mathcal{K}}[g](\cdot) := \int_{\mathcal{X}} \mathcal{K}(\cdot, u) g(u) d\mu_x(u). \tag{28}$$

Under general conditions related to Mercer theorem (Mercer 1909; Hochstadt 1973; Sun 2005), we can find eigenfunctions ρ_i and eigenvalues ζ_i of the integral operator induced by $\mathcal{K}$, i.e.

$$\int_{\mathcal{X}} \mathcal{K}(\cdot, x) \rho_i(x) d\mu_x(x) = \zeta_i \rho_i(\cdot), \quad 0 < \zeta_1 \leq \zeta_2 \leq \ \ldots \tag{29}$$

Then, the kernel can be diagonalized as follows

$$\mathcal{K}(a, x) = \sum_{i=1}^{\infty} \zeta_i \rho_i(a) \rho_i(x), \quad \zeta_i > 0 \ \ \forall i. \tag{30}$$

We can now define a new class of boosting kernels over generic domains by extending (13) as follows:

$$\mathcal{P}_{\lambda,\nu}(a, x) = \sigma^2 \sum_{i=1}^{\infty} \left(\frac{(\lambda \zeta_i + \sigma^2)^{\nu}}{\sigma^{2\nu}} - 1 \right) \rho_i(a) \rho_i(x). \tag{31}$$

The above definition leads to (13) when U is the identity matrix and μ_x concentrates uniformly the probability only on a finite set of input locations x_i. More generally, the entire class (31) inherits all the properties discussed in the finite-dimensional case. In particular, for function estimation one can now consider estimators

$$\hat{f} = \underset{f \in \mathcal{H}_{\lambda,\nu}}{\operatorname{argmin}} \sum_{i=1}^{n} \mathcal{V}_i(y_i - f(x_i)) + \gamma \|f\|^2_{\mathcal{H}_{\lambda,\nu}} \tag{32}$$

where $\mathcal{H}_{\lambda,\nu}$ is the RKHS induced by (31) and γ is a parameter independent of λ and σ^2. As ν increases, the estimator $\hat{f}$ relies more strongly on the first eigenfunctions of $\mathcal{K}$. In fact, on the basis of the ratio λ/σ^2, the kernel $\mathcal{P}_{\lambda,\nu}$ can assign most of the prior variance to the ρ_i associated to the largest eigenvalues ζ_i, for the same reasons as described in Sect. 3.4.

The introduction of the generalized boosting kernel means also that learning rates and consistency analysis on kernel machines (Wu et al. 2006; Smale and Zhou 2007; Steinwart 2002) apply immediately to our boosting context. For example, consider quadratic losses with

$$\mathcal{V}_i(y_i - f(x_i)) = \frac{(y_i - f(x_i))^2}{n}$$

and fix σ^2, ν and λ. Then, we can make γ suitably depend on the data set size n to guarantee the consistency of $\hat{f}$ and to obtain non asymptotic bounds around the estimate. In particular, if we replace the integral operator $L_{\mathcal{K}}$ used in Smale and Zhou (2007) with that induced by the boosting kernel, i.e.

$$L_{\mathcal{P}_{\lambda,\nu}}[g](\cdot) := \int_{\mathcal{X}} \mathcal{P}_{\lambda,\nu}(\cdot, u) g(u) d\mu_x(u), \tag{33}$$

all the bounds in Smale and Zhou (2007, Corollary 5) immediately apply.[3] For instance, suppose data $\{x_i, y_i\}$ are i.i.d. with input locations drawn from μ_x. If the regression function, i.e. the optimal predictor of future data, is in the range of $L^r_{\mathcal{P}_{\lambda,\nu}}$ for $0.5 < r \leq 1$, the error, as measured by the norm in the Lebesgue space equipped with μ_x, will decrease as $\frac{1}{n}^{\frac{r}{1+2r}}$. Hence, our kernel-based perspective clarifies that, under Smale and Zhou's integral operator framework, boosting significantly improves the convergence rate if ν can make r as close as possible to 1.

5 Numerical experiments

5.1 Boosting kernel regression: temperature prediction real data

To test boosting on real data, we use a case study in thermodynamic modeling of buildings. Eight temperature sensors produced by Moteiv Inc were placed in two rooms of a small two-floor residential building of about 80 m^2 and 200 m^3. The experiment lasted for 8 days starting from February 24th, 2011; samples were taken every 5 min. A thermostat controlled the heating systems and the reference temperature was manually set every day depending upon occupancy and other needs. The goal of the experiment is to assess the predictive capability of models built using kernel-based estimators.

We consider Multiple Input-Single Output (MISO) models. The temperature from the first node is the output (y_i) and the other 7 represent the inputs (u_i^j, $j = 1, .., 7$). The measurements are split into a training set of size $N_{id} = 1000$ and a test set of size $N_{test} = 1500$. The notation y^{test} indicates the test data, which is used to test the ability of our estimator to predict future data. Data are normalized so that they have zero mean and unit variance before identification is performed.

The model predictive power is measured in terms of k-step-ahead prediction fit on y^{test}, i.e.

$$100 \times \left(1 - \sqrt{\sum_{i=k}^{N_{test}} (y_i^{test} - \hat{y}_{i|i-k})^2} \Big/ \sqrt{\sum_{i=k}^{N_{test}} (y_i^{test})^2}\right).$$

[3] In Smale and Zhou (2007), λ denotes the regularization parameter. Hence, λ in Smale and Zhou (2007, Corollary 5) corresponds to the γ introduced in (32).

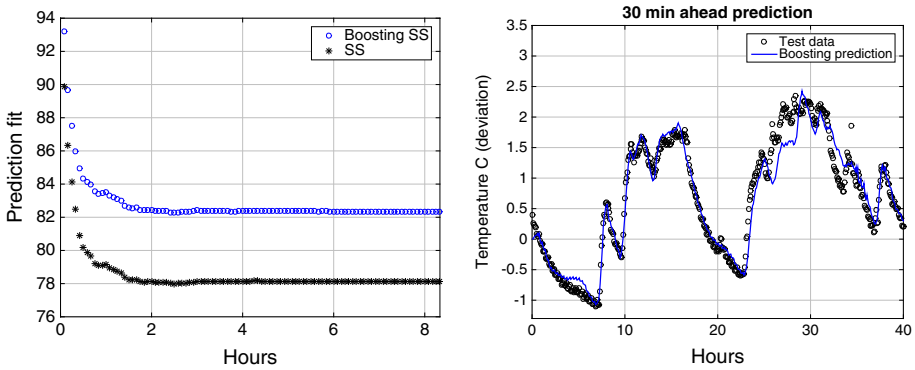

Fig. 7 Left: prediction fits obtained by the stable spine estimator (SS) and by Boosting equipped with the stable spline kernel (Boosting SS). Right: 30-min ahead temperature prediction from Boosting SS on a portion of the test set

We consider ARX models of the form

$$y_i = (g^1 \otimes y)_i + \sum_{j=1}^{7} (g^{j+1} \otimes u^j)_i + v_i,$$

where $\otimes$ denotes discrete-time convolution and the $\{g^j\}$ are 8 unknown one-step ahead predictor impulse responses, each of length 50. Note that when such impulse responses are known, one can use them in an iterative fashion to obtain any k-step ahead prediction. We can stack all the $\{g^j\}$ in the vector θ and form the regression matrix U with the past outputs and the inputs so that the model becomes $y = U\theta + v$. Then, we consider the following two estimators:

- **Boosting SS** this estimator regularizes each g^j introducing information on its smoothness and exponential decay by the stable spline kernel (Pillonetto and De Nicolao 2010). In particular, let $P \in \mathbb{R}^{50\times 50}$ with (i, j) entry $\alpha^{\max(i,j)}$, $0 \leq \alpha < 1$. Then, we recover θ by the boosting scheme (23) with $K = \text{blkdiag}(P, \ldots, P)$, and $\mathcal{V}$ set to the quadratic loss. Note that the estimator contains the three unknown hyperparameters ν, α and $\gamma = \sigma^2/\lambda$. To estimate them the training set is divided in half and hold-out cross validation is used.
- **Classical Boosting SS** the same as above except that ν can assume only integer values.
- **SS** this is the stable spline estimator described in Pillonetto and De Nicolao (2010) (and corresponds to **Boosting SS** with $\nu = 1$) with hyperparameters obtained via marginal likelihood optimization.

For **Boosting SS**, we obtained $\gamma = 0.02, \alpha = 0.82$ and $\nu = 1.42$; note that it is not an integer. For **Classical Boosting SS**, we obtained $\gamma = 0.03, \alpha = 0.79$ and $\nu = 1$. In practice, this estimator gives the same results achieved by **SS** so that our discussion below just compares the performance of **Boosting SS** and **SS**.

The left panel of Fig. 7 shows the prediction fits, as a function of the prediction horizon k, obtained by **Boosting SS** and **SS**. Note that the non-integer ν gives an improvement in performance. This means that in this experiment using a continuous ν improves also over the classical boosting. The right panel of Fig. 7 shows sample trajectories of half-hour-ahead boosting prediction on a part of the test set.

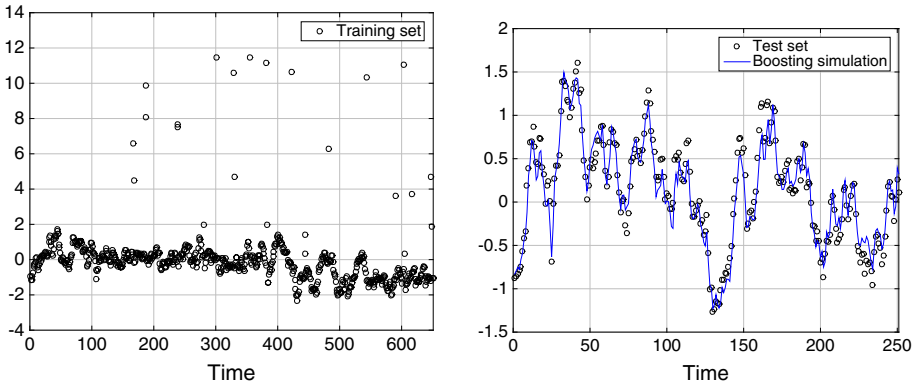

Fig. 8 Left: training set. Right: test set simulation from Boosting SS with ℓ_1 loss

5.2 Boosting kernel regression using the ℓ_1 loss: real data water tank system identification

We test our new class of boosting algorithms on another real data set obtained from a water tank system [see also Bottegal et al. (2016)]. In this example, a tank is fed with water by an electric pump. The water is drawn from a lower basin, and then flows back through a hole in the bottom of the tank. The system input is the voltage applied, while the output is the water level in the tank, measured by a pressure sensor at the bottom of the tank. The setup represents a typical control engineering scenario, where the experimenter is interested in building a mathematical model of the system in order to predict its behavior and design a control algorithm (Ljung 1999). To this end, input/output samples are collected every second, comprising almost 1000 pairs that are divided into a training and test set. The signals are de-trended, removing their means. The training and test outputs are shown in the left and right panel of Fig. 8. One can see that the second part of the training data are corrupted by outliers caused by pressure perturbations in the tank; these are due to air occasionally being blown into the tank. Our aim is to understand the predictive capability of the boosting kernel even in presence of outliers.

We consider a FIR model of the form

$$y_i = (g \otimes u)_i + v_i,$$

where the unknown vector $g \in \mathbb{R}^{50}$ contains the impulse response coefficients. It is estimated using a variation of the estimator **Boosting SS** described in the previous section: while the stable spline kernel is still employed to define the regularizer, the key difference is that $\mathcal{V}$ in (23) is now set to the robust ℓ_1 loss. The hyperparameter estimates obtained using hold-out cross validation are $\gamma = 17.18$, $\alpha = 0.92$ and $\nu = 1.7$. The right panel of Fig. 8 shows the boosting simulation of the test set. The estimate from **Boosting SS** predicts the test set with 76.2% fit. Using the approach $\mathcal{V}$ equal to the quadratic loss, the test set fit decreases to 57.8%.

5.3 Boosting in RKHSs: classification problem

Consider the problem described in Section 2 of Hastie et al. (2001a). Two classes are introduced, each defined by a mixture of Gaussian clusters; the first 10 means are generated from

a Gaussian $\mathcal{N}([1\ 0]^T, I)$ and remaining ten means from $\mathcal{N}([0\ 1]^T, I)$ with I the identity matrix. Class labels 1 and -1 corresponding to the clusters are generated randomly with probability 1/2. Observations for a given label are generated by picking one of the ten means m_k from the correct cluster with uniform probability 1/10, and drawing an input location from $\mathcal{N}(m_k, I/5)$. A Monte Carlo study of 100 runs is designed. At any run, a new data set of size 500 is generated, with the split given by 50% for training and 25% each for validation and testing. The validation set is used to estimate through hold-out cross-validation the unknown hyperparameters, in particular the boosting parameter ν. Performance for a given run is quantified by computing percentage of data correctly classified.

We compare the performance of the following two estimators:

- **Boosting+ℓ_1 loss** this is the boosting scheme in RKHS illustrated in the previous section (ν may assume only integer values) with the weak learner (24) defined by the Gaussian kernel

$$\mathcal{K}(x, a) = \exp(-10|x - a|^2), \quad |\cdot| = \text{Euclidean norm}$$

  setting each $\mathcal{V}_i$ to the ℓ_1 loss and using $\gamma = 1000$.
- **Boosting kernel+ℓ_1 loss** this is the estimator using the new boosting kernel. The latter is defined by the kernel matrix built using the same Gaussian kernel reported above, with $\sigma^2 = 1, \lambda = 0.001$ so that one still has $\gamma = 1000$. The function estimate is achieved solving (26) using the ℓ_1 loss.

Note that the two estimators contain only one unknown parameter, i.e. ν which is estimated by the cross validation strategy described above. The top left panel of Fig. 9 compares their performance. Interestingly, results are very similar, see also Table 1. This supports the fact that the boosting kernel can include classical boosting features in the estimation process. In this example, the difference between the two methods is mainly in their computational complexity. In particular, the top right panel of Fig. 9 reports some cross validation scores as a function of the boosting iterations counter ν for the classical boosting scheme. The score is linearly interpolated, since ν can assume only integer values. On average, during the 100 Monte Carlo runs the optimal value corresponds to $\nu = 340$, so on average, problems (24) must be solved 340 times. After obtaining the estimate of ν, to obtain the function estimate using the union of the training and validation data, another 340 problems must be solved.

In contrast, the boosting kernel used in (26) does not require repeated optimization of the weak learner. Using a golden section search, estimating ν by cross validation on average requires solving 20 problems of the form (26). Once ν is found, only one additional optimization problem must be solved to obtain the function estimate. Summarizing, in this example the boosting kernel obtains results similar to those achieved by classical boosting, but requires solving only 20 optimization problems rather than nearly 700. The computational times of the two approaches are reported in the bottom panel of Fig. 9.

Table 1 also shows the average fit obtained by other two estimators. The first estimator is denoted by **Boosting SVC**: it coincides with **Boosting kernel+ℓ_1 loss**, except that the hinge loss replaces the ℓ_1 loss in (26). The other one is **SVC** and corresponds to the classical support vector classifier. It uses the same Gaussian kernel defined above with the regularization parameter γ determined via cross validation on a grid containing 20 logarithmically spaced values on the interval [0.01, 100]. One can see that the best results are obtained by boosting support vector classification. Recall also that the hinge loss cannot be adopted using the classical boosting scheme as discussed at the end of the previous section.

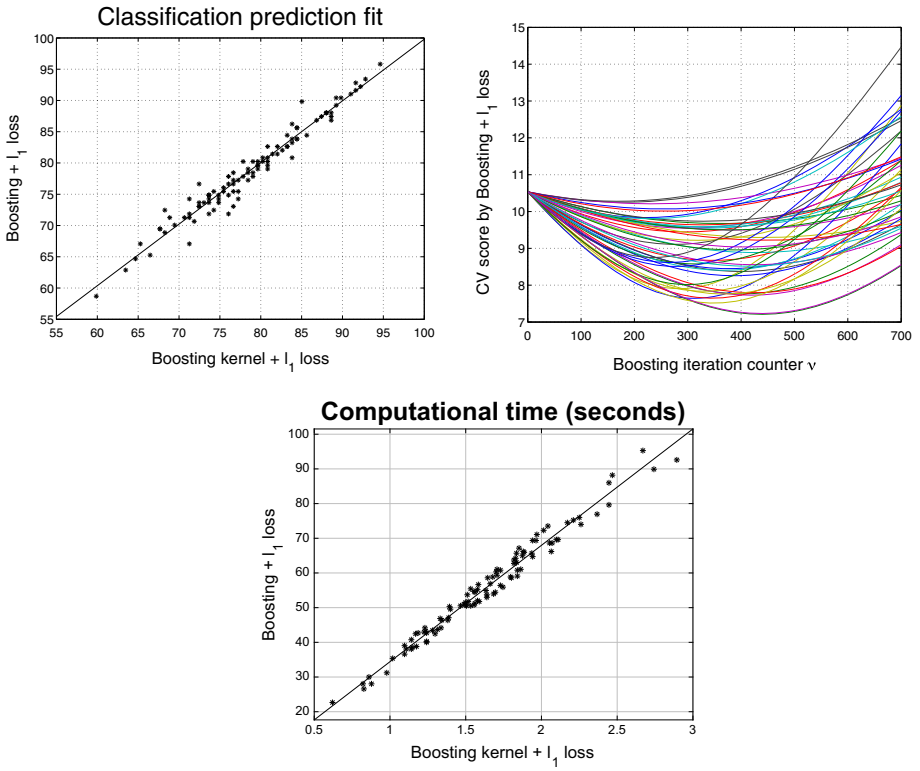

Fig. 9 Classification problem. Top left Fits obtained by the new boosting kernel (x-axis) vs fits obtained by the classical boosting scheme (y-axis). Both the estimators use the ℓ_1 loss. Top right Some cross validation scores computed using the classical boosting scheme equipped with the ℓ_1 loss as a function of the boosting iteration counter ν. Each curve corresponds to a different run. Bottom Computational times to solve a classification problem needed by the new boosting kernel (x-axis) and by the classical boosting scheme (y-axis)

Table 1 Average percentage classification fit

Boosting + ℓ_1	Boosting kernel + ℓ_1	Boosting SVC	SVC
78.91 %	79.15 %	79.73 %	78.12 %

5.4 Boosting in RKHSs: classification using the UCI machine learning repository

The new classifier **Boosting SVC**, which combines boosting and the hinge loss, is now tested using binary classification problems from the UCI machine learning repository.[4] We consider the following data sets from the Statlog project: breast cancer, ionosphere, monk 1, heart, sonar and Australian credit. Prediction performance of **Boosting SVC** (as defined in the previous section) is compared with that obtained by the classifiers introduced in Bühlmann and Yu (2003), which consider the same datasets. We compare with **L2 Boost**, a weighted version of L2 Boost called **L2 WCBoost** and **LogitBoost**. For details on these three classical boosting algorithms, see Bühlmann and Yu (2003). These approaches use an integer number

[4] http://www.ics.uci.edu/~mlearn/MLRepository.

Table 2 Average percentage classification fit with UCI data

	L2 Boost	L2 WCBoost (%)	LogitBoost (%)	Boosting SVC (%)
Breast cancer	96.4	96	96.2	96.1
Ionosphere	82.2	83.2	85.2	88.2
Monk 1	91.2	92.1	93	92.4
Hearth	83.3	82.5	84.1	83.9
Sonar	87.7	87.7	86.9	89.2
Credit	99.8	99.6	100	99.8

of boosting iterations ν, exploiting tree learners, stumps or componentwise smoothing splines, as described in Section 4.1 of Bühlmann and Yu (2003). These three estimators proved to have excellent prediction capability on the UCI data sets, outperforming two versions of CART based on trees (Hastie et al. 2001a).

As in Bühlmann and Yu (2003), the estimated percentage classification fit is calculated using an average of 50 random divisions into training set (90% of the data) and test set (10% of the data). The estimators **L2 Boost**, **L2 WCBoost** and **LogitBoost** use an oracle-based procedure to tune complexity, selecting the number of boosting iterations ν to maximize the test set fit. In addition, we report the best performance that can be obtained from these three methods, selecting either stumps or component smoothing splines. The proposed **Boosting SVC** is implemented using the Gaussian kernel

$$K(x, a) = \exp(-\beta|x - a|^2), \quad |\cdot| = \text{Euclidean norm}.$$

The estimator's structure contains three unknown parameters: the regularization parameter, the kernel width β and the (real-valued) ν. These parameters are estimated without resorting to the oracle, but using only the training set. We use hold-out cross validation (CV) with a random split that defines a validation set including $1/3$ of the available data. Gradient descent is used to optimize the CV score. We emphasize that **Boosting SVC**, can tune regularization by solving only a single optimization problem on a continuous domain, thanks to the parametrization using a real-valued rather than integer ν.

Results for the four estimators are show in Table 2. Even without using an oracle to tune complexity, the prediction performance of **Boosting SVC** is very similar to those of **L2 Boost**, **L2 WCBoost** and **LogitBoost** in breast cancer, monk 1, heart and Australian credit datasets. For ionosphere and sonar, **Boosting SVC** significantly outperforms the other three algorithms.

5.5 Boosting in RKHSs: regression problem

Consider now a regression problem where only smoothness information is available to reconstruct the unknown function from sparse and noisy data. As in the previous example, our aim is to illustrate how the new class of proposed boosting algorithms can solve this problem using a RKHS with a great computational advantage w.r.t. the traditional scheme. For this purpose, we just consider a classical benchmark problem where the unknown map is the Franke's bivariate test function f given by the weighted sum of four exponentials (Wahba 1990). Data set size is 1000 and is generated as follows. First, 1000 input locations x_i are drawn from a uniform distribution on $[0, 1] \times [0, 1]$. The data are divided in the same way described in the classification problem. The outputs in the training and validation data are

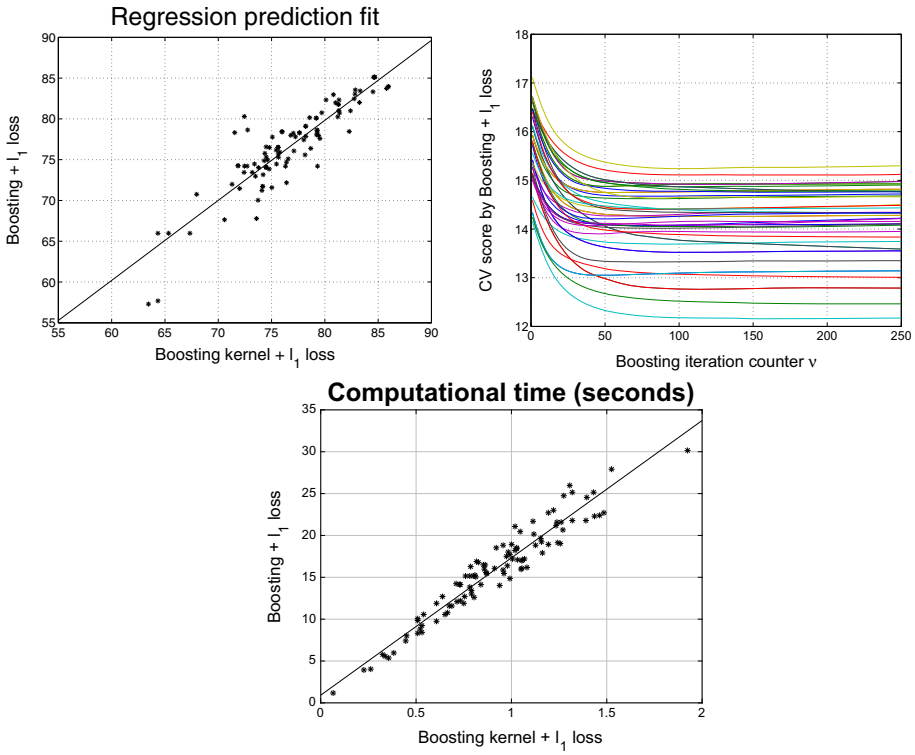

Fig. 10 Regression problem. Top left Fits obtained by the new boosting kernel (x-axis) vs fits obtained by the classical boosting scheme (y-axis). Both the estimators use the ℓ_1 loss. Top right Some cross validation scores computed using the classical boosting scheme equipped with the ℓ_1 loss as a function of the boosting iteration counter ν. Bottom Computational times to solve a regression problem needed by the new boosting kernel (x-axis) and by the classical boosting scheme (y-axis)

$$y_i = f(x_i) + v_i$$

where the errors v_i are independent, with distribution given by the mixture of Gaussians

$$0.9\mathcal{N}(0, 0.1^2) + 0.1\mathcal{N}(0, 1).$$

The test outputs y_i^{test} are instead given by noiseless outputs $f(x_i^{test})$. A Monte Carlo study of 100 runs is considered, where a new data set is generated at any run. The test fit is computed as

$$100\left(1 - \frac{|y^{test} - \hat{y}^{test}|}{|y^{test} - \text{mean}(y^{test})|}\right),$$

where $\hat{y}^{test}$ is the test set prediction.

Note that the mixture noise can model the effect of outliers which affect, on average, 1 out of 10 outputs. This motivates the use of the robust ℓ_1 loss. Hence, the function is still reconstructed by **Boosting+ℓ_1 loss** and **Boosting kernel+ℓ_1 loss** which are implemented exactly in the same way as previously described. Figure 10 displays the results with the same rationale adopted in Fig. 9. The fits are close each other but, at any run, the classical boosting scheme requires solving hundreds of optimization problems, while the boosting kernel-based

Table 3 Average percentage regression fit

Boosting + ℓ_1	Boosting kernel + ℓ_1	Gaussian kernel + ℓ_1
76.62 %	76.75 %	75.19 %

approach needs to solve around 15 problems on average. The computational times of the two approaches are reported in the bottom panel of Fig. 10.

Finally, Table 3 reports the average fits including those achieved by **Gaussian kernel+ℓ_1 loss**, which is implemented as the estimator **SVC** described in the previous section except that the hinge loss is replaced by the ℓ_1 loss. The best results are achieved by boosting kernel with ℓ_1.

6 Conclusion

In this paper, we presented a connection between boosting and kernel-based methods. We showed that in the context of regularized least-squares, boosting with a weak learner is equivalent to using a boosting kernel. This connection also implies that learning rates and consistency analysis on kernel based methods (Wu et al. 2006; Smale and Zhou 2007; Steinwart 2002) can be immediately used in the boosting context.

In the paper, we also developed three specific applications of the theoretical relationship between classical boosting and the boosting kernel. (1) Better understanding of boosting estimators and when they work effectively; (2) efficient hyperparameter estimation for boosting (3) development of a general class of boosting schemes for misfit measures, including ℓ_1, Huber and Vapnik, for both Euclidean and Reproducing Kernel Hilbert Spaces.

Hyperparameter tuning avoids sequential application of weak learners, which is crucial for boosting with general losses $\mathcal{V}$, as each boosting run would itself require an iterative algorithm. When working over RKHS, the boosting kernel has equal or better performance than classical methods at a dramatically reduced computational cost.

In addition to computational efficiency, treating boosting iterations ν as a continuous hyperparameter (rather than a discrete iteration) can improve prediction. In some of the experiments we obtained $\nu = 1.42$ as estimate of ν, with better prediction performance than an integer ν of 1 or 2.

Acknowledgements Funding was provided by Washington Research Foundation.

References

Anderson, B. D. O., & Moore, J. B. (1979). *Optimal filtering*. Englewood Cliffs, NJ: Prentice-Hall.

Aravkin, A., Burke, J., Ljung, L., Lozano, A., & Pillonetto, G. (2017). Generalized Kalman smoothing. *Automatica*, *86*, 63–86.

Aravkin, A., Kambadur, P., Lozano, A., & Luss, R. (2014). Orthogonal matching pursuit for sparse quantile regression. In *International conference on data mining (ICDM)* (pp. 11–19). IEEE.

Avnimelech, R., & Intrator, N. (1999). Boosting regression estimators. *Neural Computation*, *11*(2), 499–520.

Bissacco, A., Yang, M. H., & Soatto, S. (2007). Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. In *2007 IEEE conference on computer vision and pattern recognition* (pp. 1–8). IEEE.

Bottegal, G., Aravkin, A., Hjalmarsson, H., & Pillonetto, G. (2016). Robust EM kernel-based methods for linear system identification. *Automatica*, *67*, 114–126.
Breiman, L. (1998). Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, *26*(3), 801–849.
Bube, K., & Nemeth, T. (2007). Fast line searches for the robust solution of linear systems in the hybrid ℓ_1/ℓ_2 and huber norms. *Geophysics*, *72*(2), A13–A17.
Bühlmann, P., & Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, *22*, 477–505.
Bühlmann, P., & Yu, B. (2003). Boosting with the L2 loss: Regression and classification. *Journal of the American Statistical Association*, *98*(462), 324–339.
Cao, X., Wei, Y., Wen, F., & Sun, J. (2014). Face alignment by explicit shape regression. *International Journal of Computer Vision*, *107*(2), 177–190.
Champion, M., Cierco-Ayrolles, C., Gadat, S., & Vignes, M. (2014). Sparse regression and support recovery with L2-boosting algorithms. *Journal of Statistical Planning and Inference*, *155*, 19–41.
Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., & Yang, S. (2017) AdaNet: Adaptive structural learning of artificial neural networks. In *International conference on machine learning* (pp. 874–883).
Cortes, C., Mohri, M., & Syed, U. (2014). Deep boosting. In *International conference on machine learning* (pp. 1179–1187).
De Mol, C., De Vito, E., & Rosasco, L. (2009). Elastic-net regularization in learning theory. *Journal of Complexity*, *25*(2), 201–230.
Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, *13*, 1–150.
Fan, W., Stolfo, S., & Zhang, J. (1999). The application of AdaBoost for distributed, scalable and on-line learning. In *Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 362–366). ACM.
Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.
Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal—Japanese Society for Artificial Intelligence*, *14*(771–780), 1612.
Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, *28*(2), 337–407.
Gao, T., & Koller, D. (2011). Multiclass boosting with hinge loss based on output coding. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (pp. 569–576).
Hansen, M., & Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, *96*(454), 746–774.
Hastie, T., Tibshirani, R., & Friedman, J. (2001a). *The elements of statistical learning. Springer series in statistics* (Vol. 1). Berlin: Springer.
Hastie, T., Tibshirani, R., & Friedman, J. (2001b). *The elements of statistical learning. Data mining, inference and prediction*. Canada: Springer.
Hochstadt, H. (1973). *Integral equations*. New York: Wiley.
Huber, P. J. (2004). *Robust statistics*. New York: Wiley.
Hurvich, C., Simonoff, J., & Tsai, C. L. (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *60*(2), 271–293.
Koenker, R. (2005). *Quantile regression*. Cambridge: Cambridge University Press.
Koenker, R., & Geling, O. (2001). Reappraising medfly longevity: A quantile regression survival analysis. *Journal of the American Statistical Association*, *96*, 458–468.
Lemmens, A., & Croux, C. (2006). Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, *43*(2), 276–286.
Li, Q., & Lin, N. (2010). The Bayesian elastic net. *Bayesian Analysis*, *5*(1), 151–170.
Ljung, L. (1999). *System identification, theory for the user*. Upper Saddle River: Prentice Hall.
Maronna, R., Martin, D., & Yohai, V. (2006). *Robust statistics. Wiley series in probability and statistics*. New York: Wiley.
Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society London*, *209*(3), 415–446.
Oglic, D., & Gärtner, T. (2016). Greedy feature construction. In *Advances in neural information processing systems* (pp. 3945–3953).
Pillonetto, G., & De Nicolao, G. (2010). A new kernel-based approach for linear system identification. *Automatica*, *46*(1), 81–93.
Pontil, M., & Verri, A. (1998). Properties of support vector machines. *Neural Computation*, *10*, 955–974.

Rätsch, G., & Warmuth, M. K. (2005). Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, *6*(Dec), 2131–2152.
Schapire, R. (2003). The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification* (pp. 149–171). Springer.
Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227.
Schapire, R., & Freund, Y. (2012). *Boosting: Foundations and algorithms*. Cambridge: MIT Press.
Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. *Neural Networks and Computational Learning Theory*, *81*, 416–426.
Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond (adaptive computation and machine learning)*. Cambridge: MIT Press.
Schölkopf, B., & Smola, A. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge: MIT Press.
Schölkopf, B., Smola, A., Williamson, R., & Bartlett, P. (2000). New support vector algorithms. *Neural Computation*, *12*, 1207–1245.
Smale, S., & Zhou, D. (2007). Learning theory estimates via integral operators and their approximations. *Constructive Approximation*, *26*, 153–172.
Solomatine, D., & Shrestha, D. (2004) AdaBoost.RT: A boosting algorithm for regression problems. In *Proceedings of the 2004 IEEE international joint conference on neural networks* (Vol. 2, pp. 1163–1168). IEEE.
Steinwart, I. (2002). On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, *2*, 67–93.
Sun, H. (2005). Mercer theorem for RKHS on noncompact sets. *Journal of Complexity*, *21*(3), 337–349.
Temlyakov, V. (2000). Weak greedy algorithms. *Advances in Computational Mathematics*, *12*(2–3), 213–227.
Tokarczyk, P., Wegner, J., Walk, S., & Schindler, K. (2015). Features, color spaces, and boosting: New insights on semantic classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(1), 280–295.
Tu, Z. (2005). Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *Tenth IEEE international conference on computer vision, 2005. ICCV 2005* (Vol. 2, pp. 1589–1596). IEEE.
Tutz, G., & Binder, H. (2007). Boosting ridge regression. *Computational Statistics and Data Analysis*, *51*(12), 6044–6059.
Vapnik, V. (1998). *Statistical learning theory*. New York, NY: Wiley.
Viola, P., & Jones, M. (2001). Fast and robust classification using asymmetric AdaBoost and a detector cascade. *Advances in Neural Information Processing System*, *14*, 1311–1318.
Wahba, G. (1990). *Spline models for observational data*. Philadelphia: SIAM.
Wu, Q., Ying, Y., & Zhou, D. (2006). Learning rates of least-square regularized regression. *Foundations of Computational Mathematics*, *6*, 171–192.
Zhang, T. (2003). Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, *49*(3), 682–691.
Zhu, J., Zou, H., Rosset, S., & Hastie, T. (2009). Multi-class AdaBoost. *Statistics and Its Interface*, *2*(3), 349–360.
Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320.
Zou, H., & Yuan, M. (2008). Regularized simultaneous model selection in multiple quantiles regression. *Computational Statistics and Data Analysis*, *52*(12), 5296–5304.