KTBoost: Combined Kernel and Tree Boosting

Fabio Sigrist*  
Lucerne University of Applied Sciences and Arts  
February 12, 2019

Abstract

In this article, we introduce a novel boosting algorithm called ‘KTBoost’, which combines kernel boosting and tree boosting. In each boosting iteration, the algorithm adds either a regression tree or reproducing kernel Hilbert space (RKHS) regression function to the ensemble of base learners. Intuitively, the idea is that discontinuous trees and continuous RKHS regression functions complement each other, and that this combination allows for better learning of both continuous and discontinuous functions as well as functions that exhibit parts with varying degrees of regularity. We empirically show that KTBoost outperforms both tree and kernel boosting in terms of predictive accuracy on a wide array of data sets.

1 Introduction

Boosting algorithms [Freund et al., 1996, Friedman et al., 2000, Mason et al., 2000, Friedman, 2001, Bühlmann and Hothorn, 2007] enjoy large popularity in both applied data analysis and machine learning research due to their high predictive accuracy observed on a wide range of data sets [Chen and Guestrin, 2016]. Boosting additively combines base learners by sequentially minimizing a risk functional. To the best of our knowledge, except for one reference [Hothorn et al., 2010], the large majority of boosting algorithms use only one type of functions as base learners. In this article, we relax this assumption by combining trees [Breiman et al., 1984] and reproducing kernel Hilbert space (RKHS) regression functions [Schölkopf and Smola, 2001, Berlinet and Thomas-Agnan, 2011] as base learners, and we empirically show that this combination of different base learners results in increased predictive accuracy compared to both only tree and kernel boosting.

To date, regression trees are the most common choice of base learners for boosting in both applied data analysis and machine learning research. In particular, a lot of effort has been made in recent years to develop tree-based boosting methods that scale to large data [Chen and Guestrin, 2016, Ke et al., 2017, Ponomareva et al., 2017, Prokhorenkova et al., 2018]. On the other hand, kernel machines show state-of-the-art predictive accuracy for many data sets as they can, for instance, achieve near-optimal test error [Belkin et al., 2018a], and kernel classifiers parallel the behaviors of deep networks as noted in Zhang

*Email: fabio.sigrist@hslu.ch. Address: Institute of Financial Services Zug IFZ, Lucerne University of Applied Sciences and Arts, Grafenauweg 10, 6302 Zug, Switzerland.
et al. [2017]. Roughly speaking, RKHS regression learns a non-parametric regression function in an infinite dimensional RKHS. See Sections 2.2 and 2.3 for more information on kernel machines and trees.

1.1 Summary of results

We introduce a novel boosting algorithm and a novel way of combining multiple types of base learners. Specifically, we propose a boosting algorithm denoted by ‘KTBoost’, which combines kernel and tree boosting. In each boosting iteration, the KTBoost algorithm chooses to add either a regression tree or a penalized RKHS regression function, also known as kernel ridge regression [Murphy, 2012], to the ensemble. This is done by first learning both a regression tree and a RKHS regression function using one step of a functional version of Newton’s method or functional gradient descent, and then selecting the base learner whose addition to the ensemble results in the lowest empirical risk.

The idea is that the KTBoost algorithm chooses in each iteration the best base learner from two fundamentally different function classes. Functions in an RKHS are continuous and, depending on the kernel function, they can also have higher regularity. Trees, on the other hand, are discontinuous functions. Intuitively, this combination is thus well suited for learning both continuous and discontinuous functions or functions that exhibit parts with varying degrees of regularity and discontinuities. In our experiments, we show on a large collection of data sets that this combination of trees and RKHS ridge regression functions as base learners leads to higher predictive accuracy compared to using only trees or only RKHS regression functions.

To briefly illustrate that the combination of tree and kernel boosting can lead to higher predictive accuracy, we report in Figure 1 the test mean square error (MSE) versus the number of boosting iterations for one data set (wine). The solid lines show average test MSEs over ten random splits into training, validation, and test data sets versus the number of boosting iterations. The confidence bands are obtained after point-wise excluding the largest and smallest MSEs. Tuning parameters of all methods are chosen on the validation data sets. See Section 4 for more details on the data set and the choice of tuning parameters.

For better comparison, the shrinkage parameter $\nu$, or learning rate, see Equation (4), in the boosting update is set to a fix value ($\nu = 0.1$) in this example. The figure clearly illustrates how the combination of tree and kernel boosting (KTBoost) results in a lower test MSE compared to both tree and kernel boosting. In Section 4 we perform a detailed empirical comparison where we also consider the shrinkage parameter as a tuning parameter.

To the best of our knowledge, our proposed way of combining different base learners, in particular regression trees and RKHS regression functions, is novel. In addition, the use of Newton boosting with RKHS ridge regression functions as base learners is also novel. Our approach is implemented in the Python package KTBoost, which is openly available on the Python Package Index (PyPI) repository.

1.2 Related work

Friedman et al. [2008] also propose to use different families of base learners in an ensemble. They combine rule based learners, where the rules are learned using a boosting-type algorithm, with linear functions and find that the combination of the two can lead to increased

\footnote{See https://github.com/fabsig/KTBoost for more information.}
predictive accuracy. However, their approach is different from ours since the base learners are combined by directly minimizing an L1 penalized loss, whereas we propose to combine the different base learners in the boosting algorithm. Further, the mboost R package of Hothorn et al. [2010] also allows for combining different base learners which include linear functions, one- and two-dimensional smoothing splines, spatial terms, regression trees, as well as user-defined ones. This approach is different from ours for the following reasons. First, in each boosting update, mboost chooses the best term by minimizing a least squares approximation to the negative gradient of the empirical risk. In contrast, our approach allows to learn the tree and RKHS regression function base learners using both Newton’s method or gradient descent. In addition, we then select the base learner whose addition to the ensemble directly results in the lowest empirical risk, instead of choosing the one that best approximates the negative gradient. Further, in mboost, one has to explicitly choose a type of base learner for each feature or group of features.

Concerning kernel boosting, Raskutti et al. [2014] and Wei et al. [2017] use boosting with early stopping to find minimizers in a RKHS. Apart from the fact that we combine kernel boosting with tree boosting, these approaches also differ from our proposed way of kernel boosting. For instance, Wei et al. [2017] follows the approach of Mason et al. [2000] by finding boosting updates as maximizers of the inner product of the risk functional.
gradient with the base learner under the condition that the base learners norm is smaller than one. In contrast, we choose boosting updates using either Newton’s method or a least squares approximation to the negative gradient. Further, Raskutti et al. [2014] and Wei et al. [2017] only consider interpolating kernel machines, whereas we also allow for a ridge penalty as a form of regularization in addition to early stopping.

2 Preliminaries

2.1 Boosting

There exist population as well as sample versions of boosting algorithms. For the sake of brevity, we only consider the latter here. Assume that we have data \( \{(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}, i = 1, \ldots, n\} \) from a probability distribution \( P_{X,Y} \). The goal of boosting is to find a function \( F: \mathbb{R}^p \rightarrow \mathbb{R} \) in a Hilbert space \( \Omega_S \) which allows for predicting \( y \) given \( x \). Note that \( y \) can be categorical, discrete, continuous, or of mixed type depending on whether the conditional distribution \( P_Y \mid X \) is absolutely continuous with respect to the Lebesgue, a counting measure, or a mixture of the two; see, e.g., Sigrist and Hirnschall [2017] for an example of the latter. Depending on the data or the goal of the application, the function \( F = (F^k)_{k=1}^d \) can also be multivariate. For instance, this is the case for multiclass classification where a function is learned for every class when using the entropy loss with a softmax function, or for generalized additive models for location, scale and shape (GAMLSS) where location, scale, and shape parameters are modeled as functions of predictor variables [Rigby and Stasinopoulos, 2005, Mayr et al., 2012]. For the sake of simplicity, we assume in the following that \( F \) is univariate. The extension to the multivariate case is straightforward; see, e.g., Sigrist [2018].

The goal of boosting is to find a minimizer \( F^*(\cdot) \) of the empirical risk functional \( R(F) \):

\[
F^*(\cdot) = \arg\min_{F(\cdot) \in \Omega_S} R(F) = \arg\min_{F(\cdot) \in \Omega_S} \sum_{i=1}^n L(y_i, F(x_i)),
\]

where \( L(Y, F) \) is an appropriately chosen loss function. For instance, for regression, one often uses the squared loss \( L(Y, F) = (Y - F)^2 / 2 \) with \( Y \in \mathbb{R} \), and for binary or multiclass classification, popular choices are the logistic regression loss \( L(Y, F) = -YF + \log(1 + e^F) \) or the exponential loss \( L(Y, F) = e^{(2Y-1)F} \) for \( Y \in \{0, 1\} \), and the entropy softmax loss \( L(Y, F) = \sum_{k=1}^K \left( -1 \{Y=k\} F_k + \log \left( \sum_{i=1}^K e^{F_i(X)} \right) \right) \) for \( Y \in \{1, \ldots, K\} \), respectively.

For boosting, one assumes that the function space \( \Omega_S = \text{span}(S) \) is the span of a set of base learners \( S = \{f_j : \mathbb{R}^p \rightarrow \mathbb{R}\} \). Boosting then finds \( F^*(\cdot) \) in sequential way by iteratively adding an update \( f_m \) to the current estimate \( F_{m-1} \):

\[
F_m(x) = F_{m-1}(x) + f_m(x), \quad f_m \in S, \quad m = 1, \ldots, M,
\]

such that the empirical risk is minimized

\[
f_m = \arg\min_{f \in S} R(F_{m-1} + f).
\]

Since the risk in (3) usually cannot be minimized explicitly, one uses an approximate minimizer. Depending on whether gradient or Newton boosting is used, the update \( f_m \) is
either obtained as the least squares approximation to the negative functional gradient or by applying one step of the functional Newton method which corresponds to minimizing a second order Taylor expansion of the risk functional; see Section 3 or Sigrist [2018] for more information. For increased predictive accuracy [Friedman, 2001], an additional shrinkage parameter $\nu > 0$ is added to the update equation to obtain

$$F_m(x) = F_{m-1}(x) + \nu f_m(x).$$

(4)

2.2 Kernel ridge regression

Assume that $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive definite kernel function. Then there exists a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ with an inner product $\langle \cdot, \cdot \rangle$ such that (i) the function $K(\cdot, x)$ belongs to $\mathcal{H}$ for all $x \in \mathbb{R}^d$ and (ii) $f(x) = \langle f, K(\cdot, x) \rangle$ for all $f \in \mathcal{H}$. Suppose we are interested in finding the minimizer

$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \Vert f \Vert^2_{\mathcal{H}},$$

(5)

where $\lambda \geq 0$ is a regularization parameter. The representer theorem [Schölkopf et al., 2001] then states that there is a unique minimizer of the form

$$f(\cdot) = \sum_{j=1}^{n} \alpha_j K(x_j, \cdot)$$

(6)

and (5) can be written as

$$\|y - K\alpha\|^2 + \lambda \alpha^T K \alpha,$$

where $y = (y_1, \ldots, y_n)^T$, $K \in \mathbb{R}^{n \times n}$ with $K_{ij} = K(x_i, x_j)$, and $\alpha = (\alpha_1, \ldots, \alpha_n)^T$. Taking derivatives and equaling them to zero, we find the explicit solution as

$$\alpha = (K + \lambda I_n)^{-1} y,$$

(7)

where $I_n$ denotes the $n$-dimensional identity matrix. Popular kernel choices include the exponential or Laplace kernel

$$K(x_1, x_2) = \exp \left( -\|x_1 - x_2\|/\rho \right)$$

as well as the Gaussian or radial kernel

$$K(x_1, x_2) = \exp \left( -\|x_1 - x_2\|^2/\rho^2 \right),$$

where $\| \cdot \|$ denotes the L2 norm. The choice of the kernel determines the properties of the functions such as the degree of smoothness. The parameter $\lambda$ determines the amount of regularization and, for the above mentioned kernels, $\rho$ controls how fast the kernel decays with distance $\|x_1 - x_2\|$.

There is a close connection between Gaussian process regression and kernel regression. The solution to (5) is the posterior mean conditional on the data of a zero-mean Gaussian process with covariance function $K$. In spatial statistics [Diggle et al., 1998], this is also called a Kriging estimator. The regularization parameter $\lambda$ is the variance of the so-called idiosyncratic nugget effect, and for, e.g., the Laplace and the radial kernel, $\rho$ is a range
parameter that determines how fast the correlation of the process at points \( x_1 \) and \( x_2 \) decays with distance \( \|x_1 - x_2\| \).

Further, since \( f(x) = y^T(K + \lambda I_n)^{-1}k(x) \) where

\[
k(x) = (K(x_1, x), \ldots, K(x_n, x))^T,
\]

kernel regression can also be interpreted as a two-layer neural network.

2.3 Regression trees

We denote by \( \mathcal{T} \) the space which consists of regression trees \[ Breiman et al., 1984]. Regression and classification trees can be represented in various form. For notational brevity, we follow the notation used in \[ Chen and Guestrin, 2016\]; i.e., a regression tree is given by

\[
f_T(x) = w_{s(x)},
\]

where \( s : \mathbb{R}^p \to \{1, \ldots, J\} \), \( w \in \mathbb{R}^J \), and \( J \in \mathbb{N} \) denotes the number of terminal nodes of the tree \( f_T(x) \). \( s \) determines the structure of the tree, i.e., the partition of the space, and \( w \) denotes the leaf values. As in \[ Breiman et al., 1984\], we assume that the partition of the space made by \( s \) is a binary tree where each cell in the partition is a rectangle of the form

\[
R_j = (l_1, u_1] \times \cdots \times (l_p, u_p] \subset \mathbb{R}^p \text{ with } -\infty \leq l_m < u_m \leq \infty \text{ and } s(x) = j \text{ if } x \in R_j.
\]

3 Combined kernel and tree boosting

Let \( R^2(F_{m-1} + f) \) denote the following functional, which is proportional to the second order Taylor approximation around \( F_{m-1} \) of the empirical risk in \[ 1\]:

\[
R^2(F_{m-1} + f) = \sum_{i=1}^{n} g_{m,i}f(x_i) + \frac{1}{2} h_{m,i}f(x_i)^2,
\]

where \( g_{m,i} \) and \( h_{m,i} \) are the gradient and the Hessian:

\[
g_{m,i} = \frac{\partial}{\partial F} L(y_i, F) \bigg|_{F=F_{m-1}(x_i)},
\]

\[
h_{m,i} = \frac{\partial^2}{\partial F^2} L(y_i, F) \bigg|_{F=F_{m-1}(x_i)}.
\]

The KTBoost algorithm then works as follows. In each boosting iteration \( m \), a candidate regression tree \( f_T^m(x) \) and a candidate RKHS ridge regression function \( f_K^m(x) \) are found as minimizers to the second order Taylor approximation \( R^2(F_{m-1} + f) \) of the empirical risk around the current estimate \( F_{m-1} \). This corresponds to applying one step of Newton’s method. The algorithm then chooses to add either the regression tree or the RKHS regression function to the ensemble in such a way that the addition of the base learner to the ensemble according to Equation \[ 4\] results in the lowest empirical risk. Algorithm \[ 1\] summarizes this.

It can be shown that candidate trees \( f_T^m(x) \) can be found as weighted least squares minimizers; see, e.g., \[ Chen and Guestrin, 2016\] or \[ Sigrist, 2018\]. The candidate penalized RKHS regression functions \( f_K^m(x) \) can be found as shown in the following Proposition \[ 3.1\].
Algorithm 1: KTBoost

Initialize $F_0(x) = \tilde{y}$.

for $m = 1$ to $M$ do

Compute the gradient and the Hessian

$$g_{m,i} = \frac{\partial}{\partial F} L(y_i, F) \bigg|_{F=F_m(x)}$$

$$h_{m,i} = \frac{\partial^2}{\partial^2 F} L(y_i, F) \bigg|_{F=F_m(x)}$$

Find the candidate regression tree $f^T_m(x)$ and RKHS function $f^K_m(x)$

$$f^T_m(x) = \arg\min_{f \in T} R^2(F_{m-1} + f) + \lambda \|f\|_H^2$$

$$f^K_m(x) = \arg\min_{f \in \mathcal{H}} R^2(F_{m-1} + f) + \lambda \|f\|_H^2$$

where the approximate risk $R^2(F_{m-1} + f)$ is defined in [9]

if $R(F_{m-1} + \lambda f^T_m(x)) \leq R(F_{m-1} + \lambda f^K_m(x))$ then

$f_m(x) = f^T_m(x)$

else

$f_m(x) = f^K_m(x)$

end if

Update $F_m(x) = F_{m-1}(x) + \nu f_m(x)$

end for

Proposition 3.1. The optimal kernel ridge regression solution $f^K_m(x)$ in the Newton update step of boosting is given by

$$f^K_m(x) = k(x)^T \alpha_m,$$

where $k(x)$ is defined in [8] and

$$\alpha_m = D_m (D_m K D_m + \lambda I_n)^{-1} D_m y_m,$$

where

$$D_m = \text{diag} \left( \sqrt{h_{m,i}} \right),$$

$$y_m = (-g_{m,1}/h_{m,1}, \ldots, -g_{m,n}/h_{m,n})^T,$$

and $I_n$ is the identity matrix of dimension $n$.

Proof. We have

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} g_{m,i} f(x_i) + \frac{1}{2} h_{m,i} f(x_i)^2 + \lambda \|f\|_H^2$$

$$= \min_{f \in \mathcal{H}} \sum_{i=1}^{n} h_{m,i} \left( \frac{g_{m,i}}{h_{m,i}} - f(x_i) \right)^2 + \lambda \|f\|_H^2$$

$$= \min_{\alpha} \|D_m y_m - D_m K \alpha\|^2 + \lambda \alpha^T K \alpha.$$
If we take derivatives with respect to $\alpha$, equal them to zero, and solve for $\alpha$, we find that

$$
\alpha_m = (KD_mD_mK + \lambda K)^{-1}KD_my_m
$$

$$
= D_m(D_mKD_m + \lambda I_n)^{-1}D_my_m.
$$

In the above formulation, KTBoost uses Newton boosting; i.e., the updates $f_m$ are found by minimizing a second order approximation to the empirical risk. If either the loss function is not twice differentiable in its second argument, the second derivative is zero or constant on a non-null set of the support of $X$, one can alternatively use gradient descent instead of the Newton method. Further, gradient boosting is computationally less expensive than Newton boosting since the kernel matrix does not depend on the iteration number $m$; see Section 3.1 for more details.

In general, the gradient boosting version of KTBoost is simply obtained as a special case of the Algorithm 1 by setting $h_{m,i} = 1$. For the RKHS regression function, it follows that $D_m = I_n$. The updates $f_m$ then correspond to $L^2$ approximations of either regression trees or RKHS regression functions to the negative functional gradient.

Concerning the RKHS boosting part, the update equation $F_m(x) = F_{m-1}(x) + \nu f_m(x)$ can be replaced by simply updating the coefficients $\alpha'_m$ where $\alpha'_m$ denotes the cumulative sum of all coefficients up to iteration $m$. I.e., for the cases where the update step consists of a RKHS function, one updates the coefficients as $\alpha'_m = \alpha'_{m-1} + \nu \alpha_m$ with the convention $\alpha'_0 = \alpha'_{-1}$ if a regression tree is selected in iteration $m$. If stage-wise predictions should be made, for e.g., trace plots of error measures versus iteration numbers, one still needs to keep track of the full set of coefficients.

3.1 Reducing computational costs for large data

The computationally expensive part for learning the regression trees consists of finding the splits when growing the trees. There are several approaches in the literature on how this can be done efficiently for large data; see, e.g., [Chen and Guestrin 2016] or [Ke et al. 2017].

Concerning RKHS ridge regression, there are also several approaches that allow for computational efficiency in the large data case. Examples of this include low rank approximations based on, e.g., the Nyström method [Williams and Seeger 2001] and extensions of it such as divide-and-conquer kernel ridge regression [Zhang et al. 2013, 2015], and early stopping of iterative optimization methods [Yao et al. 2007, Blanchard and Krämer 2010, Raskutti et al. 2014, Ma and Belkin 2017]. Another approach adopted mainly in spatial statistics is to use a kernel function that has a compact support [Gneiting 2002, Bevilacqua et al. 2018]. The resulting kernel matrix $K$ is then sparse and can thus be efficiently inverted or factorized.

If gradient descent is used instead of the Newton-Raphson method, the RKHS function $f^K_m(x)$ can be found efficiently by noting that $\alpha_m = (K + \lambda I_n)^{-1}y_m$, where $y_m$ is the negative gradient vector $y_m = (-g_{m,1}, \ldots, -g_{m,n})^T$. This means that the matrix $(K + \lambda I_n)^{-1}$, or a Cholesky factor of it, needs to be calculated only once and not in every iteration $m$.

In our empirical analysis, we use the Nyström method for dealing with larger data sets. The Nyström method approximates the kernel $K(\cdot, \cdot)$ as described in the following. One first chooses a set of $l$ samples $x_1^*, \ldots, x_l^*$. Often this is done simply by sampling uniformly from the data. We call these samples Nyström samples in the following. Further, we
denote the kernel matrix that corresponds to these points as $K^*$. The Nyström method then approximates the kernel $K(\cdot, \cdot)$ as

$$K(x, y) \approx k^*(x)^T K^{*-1} k^*(y),$$

where $k^*(x) = (K(x, x_1^*), \ldots, K(x, x_l^*))^T$. In particular, the reduced-rank Nyström approximation to the full kernel matrix $K$ is given by

$$K \approx KK^{*-1}K^T.$$

4 Experimental results

In the following, we compare the KTBoost algorithm with tree and kernel boosting using several Delve, Keel, Kaggle, and UCI data sets shown in Table 1. We consider both regression as well as binary and multiclass classification data sets. Further, our choice of data sets includes data sets of different sizes in order to investigate the performance on both smaller and larger data sets. We use the squared loss for regression, the logistic regression loss for binary classification, and the cross-entropy loss with the softmax function for multiclass classification. Further, for the RKHS ridge regression, we use a Gaussian kernel

$$K(x_1, x_2) = \exp\left(-\frac{||x_1 - x_2||^2}{\rho^2}\right).$$

Concerning the boosting updates, we use gradient boosting for the regression data sets, and for the classification data sets, we use boosting with Newton updates since this can result in more accurate predictions [Sigrist, 2018]; see Section 5 for a brief comparison. However, for some classification data sets (adult, ijcnn, epileptic, and satimage), Newton boosting is computationally infeasible on a standard single CPU computer with the current implementation of KTBoost, despite the use of the Nyström method with a reasonable number of Nyström samples, say 1000, since the weighted kernel matrix in Equation (10) changes with every boosting iteration. We thus use gradient boosting in this case. All calculations are done with the Python package KTBoost on a standard laptop with a 2.9 GHz quad-core processor and 16GB of RAM.

For the larger data sets (liberty, sarcos, adult, ijcnn), we run into memory limitations, in particular when choosing tunings parameters which is done in parallel, and also into unfeasible computational times. We thus use the Nyström method described in Section 3.1. Specifically, we use $l = 1000$ Nyström samples, which are uniformly from the training data. In general, the larger the number of Nyström samples, the lower the approximation error but the higher the computational costs. We restrict ourselves to $l = 1000$ due to computational constraints. Williams and Seeger [2001] report good results with $l \approx 1000$ for several data sets.

All data sets are randomly split into three parts of equal size to obtain training, validation and test sets. Learning is done on the training data, tuning parameters are chosen on the validation data, and model comparison is done on the test data. All input features are standardized using the training data to have approximately mean zero and variance one. In order to quantify variability in the results, we use ten different random splits of the data into training, validation and test sets.
Concerning tuning parameters, we select the number of boosting iterations $M$ from \{1, 2, ..., 1000\}, the learning rate $\nu$ from \{1, $10^{-1}$, $10^{-2}$, $10^{-3}$\}, the maximal depth of the trees from \{1, 5, 10\}, and the kernel ridge regularization parameter $\lambda$ from \{1, 10\}. The kernel range parameter $\rho$ is chosen using $k$-nearest neighbors distances as described in the following. We first calculate the average distance of all $k$-nearest neighbors in the training data, where $k$ is a tuning parameter selected from \{5, 50, 500, 5000, $n - 1$\} and $n$ is the size of the training data. We then choose $\rho$ such that the kernel function has decayed to a value of 0.01 at this average $k$-nearest neighbors distance. This is motivated by the fact that for a corresponding Gaussian process with such a covariance function, the correlation has decayed to a level of 1\% at this $k$-nearest neighbor distance. If the training data contains less than 5000 (or 500) samples, we use $n - 1$ as the maximal number for the $k$-nearest neighbors. In addition, we include $\rho$ which equals the average ($n - 1$)-nearest neighbor distance. The latter choice is done in order to also include a range which results in a kernel that decays slowly over the entire space. For the large data sets where the Nyström method is used, we calculate the average $k$-nearest neighbors distance based on the Nyström samples. I.e., in this case, the maximal $k$ equals $l - 1$.

The results are shown in Table 2. For the regression data sets, we show the average test mean square error (MSE) over the different sample splits, and for the classification data sets, we calculate the average test error rate. The numbers in parentheses show standard deviations over the different sample splits. In addition, we report in Table 3 p-values from t-tests comparing KTBoost with tree and kernel boosting. All tests are done using a
Table 2: Comparison of KTBoost with tree and kernel boosting using test mean square error for regression datasets and test error rate for classification datasets. The numbers in parentheses are standard deviations.

| Data        | KTBoost | Tree   | Kernel |
|-------------|---------|--------|--------|
| abalone     | 4.648   | 5.069  | 4.639  |
|             | (0.248) | (0.261)| (0.255)|
| ailerons    | 2.642e-08 | 8.11e-08 | 2.642e-08 |
|             | (6.19e-10)| (2.39e-09)| (6.19e-10)|
| bank8FM     | 0.0009154 | 0.0009451 | 0.0009451 |
|             | (4.02e-05)| (2.47e-05)| (5.83e-05)|
| elevators   | 4.83e-06 | 5.658e-06 | 5.182e-06 |
|             | (2.9e-07)| (1.44e-07)| (3.89e-07)|
| energy      | 0.2819  | 0.3354  | 1.303   |
|             | (0.0372)| (0.093)| (0.377)|
| housing     | 12.67   | 15.13   | 13.62   |
|             | (3.19)  | (3.23)  | (2.51)  |
| liberty     | 14.53   | 14.54   | 15.19   |
|             | (0.323) | (0.314)| (0.345)|
| NavalT      | 6.513e-09 | 1.146e-06 | 6.513e-09 |
|             | (1.15e-09)| (1.58e-07)| (1.15e-09)|
| parkinsons  | 73.27   | 81.12   | 73.34   |
|             | (1.98)  | (2.44)  | (1.91)  |
| puma32h     | 6.498e-05 | 6.506e-05 | 0.0006947 |
|             | (2.27e-06)| (2.13e-06)| (2.2e-05)|
| sarcos      | 7.992   | 9.602   | 17.77   |
|             | (0.206) | (0.207)| (0.586)|
| wine        | 0.4443  | 0.4709  | 0.506   |
|             | (0.012) | (0.0169)| (0.0106)|
| adult       | 0.1282  | 0.1279  | 0.1631  |
|             | (0.00295)| (0.00313)| (0.00512)|
| cancer      | 0.03624 | 0.04148 | 0.03581 |
|             | (0.00744)| (0.0153)| (0.0107)|
| ijcnn       | 0.01223 | 0.01228 | 0.0387  |
|             | (0.000685)| (0.000702)| (0.00516)|
| ionosphere  | 0.08718 | 0.1034  | 0.1068  |
|             | (0.017) | (0.0226)| (0.0239)|
| sonar       | 0.1943  | 0.2229  | 0.1929  |
|             | (0.0394)| (0.05) | (0.0491)|
| car         | 0.03993 | 0.04115 | 0.04097 |
|             | (0.00505)| (0.00685)| (0.00624)|
| epileptic   | 0.3538  | 0.3733  | 0.442   |
|             | (0.00612)| (0.00614)| (0.0265)|
| glass       | 0.3083  | 0.3153  | 0.3444  |
|             | (0.0711)| (0.0589)| (0.0581)|
| satimage    | 0.08895 | 0.1124  | 0.0903  |
|             | (0.00452)| (0.00504)| (0.00417)|

regression model with a random effect at the sample split level to account for correlation among the results for different sample splits.

We find that KTBoost outperforms both tree and kernel boosting for the large majority of data sets that we consider. In particular, KTBoost performs either significantly better than both tree and kernel boosting or the difference in predictive accuracy is not significant. In more detail, KTBoost has a higher predictive accuracy than both tree and kernel boosting.
Table 3: p-values of t-tests comparing KTBoost with tree and kernel boosting. The p-values are calculated by taking into account that the results of the different methods are dependent for the same sample split.

| Data     | KTBoostVsTree | KTBoostVsKernel |
|----------|---------------|-----------------|
| abalone  | 3.73e-07      | 0.557           |
| ailerons | 1.23e-13      | 1               |
| bank8FM  | 0.0221        | 0.0351          |
| elevators| 5.45e-06      | 0.000153        |
| energy   | 0.0657        | 1.32e-05        |
| housing  | 0.0821        | 0.253           |
| liberty  | 0.715         | 2.36e-07        |
| NavalT   | 2.72e-09      | 1               |
| parkinsons| 1.51e-06     | 0.145           |
| puma32h  | 0.375         | 7.33e-15        |
| sarcos   | 4.69e-09      | 8.99e-13        |
| wine     | 6.78e-05      | 3.78e-08        |
| adult    | 0.589         | 2.95e-10        |
| cancer   | 0.2           | 0.891           |
| ijcnn    | 0.258         | 6.14e-08        |
| ionosphere| 0.0434       | 0.0375          |
| sonar    | 0.00847       | 0.853           |
| car      | 0.501         | 0.647           |
| epileptic| 2.86e-06      | 2.92e-06        |
| glass    | 0.66          | 0.19            |
| satimage | 4.03e-07      | 0.185           |

for seventeen out of twenty-one data sets, and it is significantly better at the 5% significance level for six data sets. In only four cases either kernel or tree boosting performs marginally better than KTBoost. In all these four cases, the difference is clearly not significant with p-values above 0.5. In summary, our empirical findings show that the combination of regression trees and RKHS regression functions as base learners can result in increased predictive accuracy compared to using only regression trees or kernel regression functions as base learners.

We note that we consider data sets with a sample size between a few hundred and approximately one hundred thousand samples. Very large data sets with several hundred thousands or millions of samples are not considered since the computational costs of the current Python implementation of KTBoost are prohibitive, in particular when choosing tuning parameters on a single CPU computer. Nonetheless, we believe that our results are important for both future research and also for applied data analysis, where small- to moderately-sized data sets are common.

Further, we note that for both tree and kernel boosting, one can consider additional tuning parameters. For instance, for trees, this includes the minimal number of samples per leaf, row and column sub-sampling, and penalization of leave values. Concerning the kernel boosting part, further tuning parameters include varying range parameters for different features, the smoothness of the kernel function, or, in general, the class of kernel functions.
Due to limits on computational costs, we have not considered all possible combinations of tuning parameters. However, it is likely that a potential increase in predictive performance in either tree or kernel boosting will also result in an increase in accuracy of the combined KTBoost algorithm.

5 Conclusions

We have introduced a novel boosting algorithm, which combines trees and RKHS ridge regression functions as base learners. Intuitively, the idea is that discontinuous trees and continuous RKHS regression functions complement each other since trees are better suited for learning rougher parts of functions and RKHS regression functions can better learn smoother parts of functions. We have compared the predictive accuracy of KTBoost with tree and kernel boosting in an extensive empirical study, and we find that KTBoost results in higher predictive accuracy compared to tree and kernel boosting for the large majority of data sets. In particular, KTBoost has either significantly higher accuracy or no significant differences are observed in our empirical evaluation.

In the future, the KTBoost algorithm should be implemented in a computationally efficient way such that it scales better to larger data sets. This is important not only for applying the gradient descent version of the KTBoost algorithm to very large data sets but also for applying the Newton boosting version to moderately-sized data sets. In fact, on the smaller classification data sets (cancer, ionosphere, sonar, car, glass), we generally observe higher predictive accuracy when using Newton boosting instead of gradient boosting; a finding which is in line with the results of Sigrist [2018]. In Table 4, we additionally report the results when using gradient boosting instead of Newton boosting. We find increased predictive accuracy of Newton boosting compared to gradient boosting for 11 out of the 15 combinations of the five data sets and the three boosting algorithms (KTBoost, tree boosting, kernel boosting). This highlights the importance of a computationally efficient implementation that allows for applying Newton boosting to moderately-sized and large data sets. Such an implementation can be done using both a fast tree-building algorithm as in Chen and Guestrin [2016], Ke et al. [2017], or Prokhorenkova et al. [2018] as well as a RKHS ridge regression algorithm and implementation that scales to very large data. Concerning the latter, several potential strategies how this can be done are briefly outlined in Section 3.1.

Table 4: Results for small classification data sets using gradient boosting.

| Data    | KTBoost     | Tree       | Kernel     |
|---------|-------------|------------|------------|
| cancer  | 0.04105     | 0.04541    | **0.03537**|
|         | (0.0157)    | (0.0153)   | (0.0106)   |
| ionosphere | **0.08632** | 0.09487    | 0.1137     |
|          | (0.0142)    | (0.0207)   | (0.0322)   |
| sonar   | **0.1957**  | 0.2514     | 0.2057     |
|          | (0.056)     | (0.0671)   | (0.0647)   |
| car     | **0.0533**  | 0.05538    | 0.05347    |
|          | (0.00937)   | (0.00949)  | (0.00911)  |
| glass   | **0.3208**  | 0.3417     | 0.3292     |
|          | (0.071)     | (0.0772)   | (0.0544)   |

Another interesting direction for future research is to investigate to which extent other
base learners such as neural networks [Huang et al., 2018, Nitanda and Suzuki, 2018] can be used in addition to trees and RKHS regression.

References

M. Belkin, D. J. Hsu, and P. Mitra. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31, pages 2306–2317, 2018a.

M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 541–549, 2018b.

A. Berlinet and C. Thomas-Agnan. Reproducing kernel Hilbert spaces in probability and statistics. Springer Science & Business Media, 2011.

M. Bevilacqua, T. Faouzi, R. Furrer, and E. Porcu. Estimation and prediction using generalized Wendland covariance functions under fixed domain asymptotics. The Annals of Applied Statistics (accepted), 2018.

G. Blanchard and N. Krämer. Optimal learning rates for kernel conjugate gradient regression. In Advances in Neural Information Processing Systems, pages 226–234, 2010.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. Classification and regression trees. CRC press, 1984.

P. Bühlmann and T. Hothorn. Boosting algorithms: Regularization, prediction and model fitting. Statistical Science, pages 477–505, 2007.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794. ACM, 2016.

P. J. Diggle, J. Tawn, and R. Moyeed. Model-based geostatistics. Journal of the Royal Statistical Society: Series C (Applied Statistics), 47(3):299–350, 1998.

Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In Icml, volume 96, pages 148–156. Bari, Italy, 1996.

J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics, 28(2):337–407, 2000.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics, pages 1189–1232, 2001.

J. H. Friedman, B. E. Popescu, et al. Predictive learning via rule ensembles. The Annals of Applied Statistics, 2(3):916–954, 2008.
T. Gneiting. Compactly supported correlation functions. Journal of Multivariate Analysis, 83(2):493–508, 2002.

T. Hothorn, P. B"uhlmann, T. Kneib, M. Schmid, and B. Hofner. Model-based boosting 2.0. Journal of Machine Learning Research, 11(Aug):2109–2113, 2010.

F. Huang, J. Ash, J. Langford, and R. Schapire. Learning deep resnet blocks sequentially using boosting theory. In ICML, volume 80, pages 2058–2067, 2018.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems, pages 3149–3157, 2017.

S. Ma and M. Belkin. Diving into the shallows: a computational perspective on large-scale shallow learning. In Advances in Neural Information Processing Systems, pages 3778–3787, 2017.

L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean. Boosting algorithms as gradient descent. In Advances in neural information processing systems, pages 512–518, 2000.

A. Mayr, N. Fenske, B. Hofner, T. Kneib, and M. Schmid. Generalized additive models for location, scale and shape for high dimensional data: a flexible approach based on boosting. Journal of the Royal Statistical Society: Series C (Applied Statistics), 61(3):403–427, 2012.

K. P. Murphy. Machine Learning: A Probabilistic Perspective. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

A. Nitanda and T. Suzuki. Functional gradient boosting based on residual network perception. In ICML, volume 80, pages 3819–3828, 2018.

N. Ponomareva, S. Radpour, G. Hendry, S. Haykal, T. Colthurst, P. Mitrichev, and A. Grushetsky. Tf boosted trees: A scalable tensorflow based framework for gradient boosting. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 423–427. Springer, 2017.

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. In Advances in Neural Information Processing Systems 31, pages 6638–6648. Curran Associates, Inc., 2018.

G. Raskutti, M. J. Wainwright, and B. Yu. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. The Journal of Machine Learning Research, 15(1):335–366, 2014.

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape. Journal of the Royal Statistical Society: Series C (Applied Statistics), 54(3):507–554, 2005.

B. Sch"olkopf and A. J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2001.
B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

F. Sigrist. Gradient and newton boosting for classification and regression. *arXiv preprint arXiv:1808.03064*, 2018.

F. Sigrist and C. Hirnschall. Grabit: Gradient tree boosted tobit models for default prediction. *arXiv preprint arXiv:1711.08695*, 2017.

Y. Wei, F. Yang, and M. J. Wainwright. Early stopping for kernel boosting algorithms: A general analysis with localized complexities. In *Advances in Neural Information Processing Systems*, pages 6065–6075, 2017.

C. K. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel ridge regression. In *Conference on Learning Theory*, pages 592–617, 2013.

Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *The Journal of Machine Learning Research*, 16(1):3299–3340, 2015.