A new formulation of gradient boosting

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 045022 https://doi.org/10.1088/2632-2153/ac1ee9

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
4 August 2021

ACCEPTED FOR PUBLICATION
18 August 2021

PUBLISHED
17 September 2021

PAPER

A new formulation of gradient boosting

Alex Wozniakowski[1,2,*], **Jayne Thompson**[3], **Mile Gu**[1,2,3,*] **and Felix C Binder**[4,5,*]

[1] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[2] Complexity Institute, Nanyang Technological University, Singapore
[3] Centre for Quantum Technologies, National University of Singapore, Singapore
[4] Institute for Quantum Optics and Quantum Information—IQOQI Vienna, Austrian Academy of Sciences, Boltzmanngasse 3, 1090 Vienna, Austria
[5] Atominstitut, Technische Universität Wien, 1020 Vienna, Austria
[*] Authors to whom any correspondence should be addressed.

E-mail: wozn0001@e.ntu.edu.sg, mgu@quantumcomplexity.org **and** quantum@felix-binder.net

Keywords: multi-target regression, boosting, ensemble learning, stacking, quantum computing, prior knowledge

Abstract

In the setting of regression, the standard formulation of gradient boosting generates a sequence of improvements to a constant model. In this paper, we reformulate gradient boosting such that it is able to generate a sequence of improvements to a nonconstant model, which may contain prior knowledge or physical insight about the data generating process. Moreover, we introduce a simple variant of multi-target stacking that extends our approach to the setting of multi-target regression. An experiment on a real-world superconducting quantum device calibration dataset demonstrates that our approach outperforms the state-of-the-art calibration model even though it only receives a paucity of training examples. Further, it significantly outperforms a well-known gradient boosting algorithm, known as LightGBM, as well as an entirely data-driven reimplementation of the calibration model, which suggests the viability of our approach.

1. Introduction

Like many supervised learning algorithms in the setting of regression, the standard formulation of gradient boosting is entirely data-driven. That is, the boosting machine receives training examples, as well as hyperparameters, such as the fitting criterion, then it initializes its fit of an additive expansion with a constant model. In turn, the boosting machine improves upon the constant model based entirely on the evidence present in the training examples. Then, the boosting machine returns the fitted additive expansion, which is a linear combination of basis functions [1–11].

But in the sciences, and many other disciplines, we may already have a nonconstant model, which is based on domain expertise. So, it seems natural to suggest that we reformulate gradient boosting such that it is able to leverage an existing, possibly nonconstant, model, and improve upon the existing model based entirely on the evidence present in the predictions made by the existing model, as well as the corresponding observations.

In this paper, we fulfill this proposal. To do so, we supplant the constant term in the standard additive expansion with a real-valued single-source prediction term, and we make a corresponding modification to the boosting machine. That is, our boosting machine receives finitely many predictions from an existing model, as well as the corresponding observations and hyperparameters, then it initializes its fit of an additive expansion with a coordinate functional in place of a constant model, which ensures the initial additive expansion always makes the same prediction as the existing model. In turn, the boosting machine improves upon the existing model based entirely on the evidence present in the predictions made by the existing model, as well as the corresponding observations. Then, the boosting machine returns the fitted additive expansion, which requires a prediction from the existing model in order to generate its own more accurate prediction.

Interestingly, a similar idea appears in the setting of multi-target regression, whereby the conventional multi-target stacking approach forms a composition of real vector-valued functions such that the first compositional layer maps an instance from the original domain to a multi-target prediction, then the second

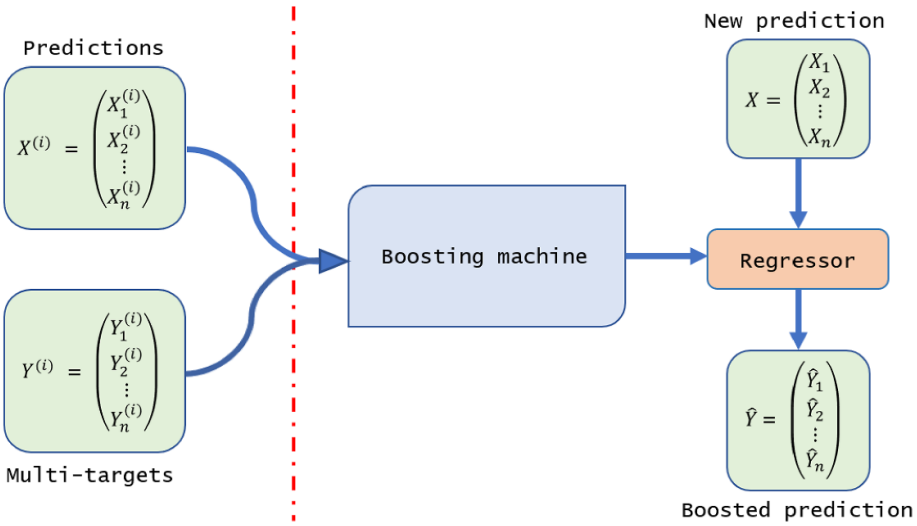

Figure 1. Conceptual representation of our learning framework. To the left of the red-dashed line, we start with finitely many multi-target predictions made by an existing model, as well as the corresponding multi-targets. To the right of the red-dashed line, a learning machine receives training examples, as well as hyperparameters. In turn, the learning machine prepares single-target training examples and invokes the boosting machine to fit an additive expansion for each single-target regression subtask. Then, the learning machine concatenates the fitted additive expansions and returns the resultant multi-target regressor, which predicts a multi-target, given a multi-target prediction from the existing model.

compositional layer maps the multi-target prediction to a more accurate multi-target prediction. Accordingly, we introduce a simple variant of multi-target stacking that supplies a learning machine with finitely many multi-target predictions made by an existing model, as well as the corresponding multi-targets and hyperparameters. In turn, the learning machine prepares single-target training examples and invokes the boosting machine to fit an additive expansion for each single-target regression subtask. Then, the learning machine concatenates the fitted additive expansions and returns the resultant real vector-valued function, which predicts a multi-target, given a multi-target prediction from the existing model; see figure 1. Notedly, multi-target stacking emanates from ensemble learning, and it is a form of regularization, or shrinkage, which is related to ideas on multivariate Gaussian mean estimation, early stopping, multiple linear regression, multi-task learning, and unsupervised pre-training of neural networks [12–25].

Our learning framework arose in the study of automated superconducting quantum device calibration. In this setting, a calibration procedure must infer a collection of control parameters that determine a superconducting quantum device's performance. To do so, the calibration procedure performs a sequence of experiments, where the result from one experiment becomes the input to the subsequent experiment, and so forth. A key experiment gives rise to a multi-target regression problem, where the low rate of calibration data acquisition makes it extremely difficult to outperform the state-of-the-art calibration model with an entirely data-driven approach [26–28]. Remarkably, the learning framework in this paper enabled us to leverage energy spectrum predictions made by the calibration model, as well as the measured energy spectrum, and improve upon the calibration model.

In the learning framework section, we review multi-target regression, then we describe the conventional multi-target stacking approach, as well as our variant of multi-target stacking. With regard to the latter, we review the standard formulation of gradient boosting, then we introduce our formulation of gradient boosting, as well as augmented gradient boosting algorithm with a built-in model selection step.

In the benchmark task section, we review the calibration of a nearest-neighbor coupled linear array of superconducting qubits, then we report the results from an experiment on a real-world calibration dataset. With regard to the experiment, figure 5 shows an optical micrograph of the actual superconducting quantum device. Figures 8 and 9 show the performance of the state-of-the-art calibration model and our learning framework on test examples, where we measure accuracy with absolute and squared error, respectively. Figures 10 and 11 show the performance of our learning framework on test examples, when the embedded boosting machine is either an off-the-shelf gradient boosting algorithm, known as LightGBM, or our formulation of gradient boosting, where we measure accuracy with absolute and squared error, respectively [29].

In the explainable machine learning section, we describe an entirely data-driven reimplementation of the calibration model, then we apply the Shapley additive explanations approach to uncover parameter dependencies in the calibration model [30]. Figures 12 and 13 show the performance of the data-driven

reimplementation and our learning framework on test examples, where we measure accuracy with absolute and squared error, respectively.

2. Learning framework

In this section, we begin with a review of the distribution-free setting of multi-target regression, wherein we assume that a multi-target loss function decomposes over the single-targets. Naturally, this leads us to discuss the independent model and conventional multi-target stacking approaches to multi-target regression, as well as our learning framework, which is a variant of the conventional multi-target stacking approach.

2.1. Multi-target regression

In the distribution-free setting of multi-target regression, let $\mathcal{X}$ denote the domain. We refer to elements of the domain as instances, and we regard instances as the input to a black box whose behavior we wish to model. If the domain is a Cartesian space with the usual definitions of vector addition and scalar multiplication, then we refer to entries of an instance as features. Let $\mathcal{Y} \subseteq \mathbb{R}^n$ denote the codomain, which is a Cartesian n-space with the usual definitions of vector addition, scalar multiplication, and inner product space structure. We refer to n-tuples in the codomain as multi-targets and to entries of multi-targets as single-targets, and we regard multi-targets as the output of the black box. As such, we begin with a sequence of ordered pairs

$$S = \{(X^{(i)}, Y^{(i)})\}_{i=1}^{m} \in (\mathcal{X} \times \mathcal{Y})^m, \tag{1}$$

which is supposed random, and we frequently refer to an ordered pair as an example. Accordingly, our goal is to find a multi-target regressor $f : \mathcal{X} \to \mathcal{Y}$, which is a rule that accurately assigns to each instance some multi-target.

In order to formalize the meaning of accuracy, we define a mapping $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, known as a multi-target loss function, and we denote the nonnegative loss of some multi-target regressor f on an example (X, Y) by $\ell(Y, f(X))$. Typical examples of a multi-target loss function, such as absolute error

$$\ell(Y, f(X)) = \sum_{j=1}^{n} |Y_j - f_j(X)| = \sum_{j=1}^{n} \ell_j(Y_j, f_j(X)) \tag{2}$$

and squared error

$$\ell(Y, f(X)) = \sum_{j=1}^{n} (Y_j - f_j(X))^2 = \sum_{j=1}^{n} \ell_j(Y_j, f_j(X)) \tag{3}$$

decompose over the single-targets

$$\ell(Y, f(X)) = \sum_{j=1}^{n} \ell_j(Y_j, f_j(X)), \tag{4}$$

where $Y_j \in \mathbb{R}$ denotes the jth single-target, $f_j(X)$ denotes the jth entry of the image of X under the multi-target regressor f, and $\ell_j : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ denotes the jth single-target loss function.

Importantly, a multi-target loss function is a random variable, so it has an associated distribution function. In turn, we define the expected multi-target loss $\mathbb{E}_{X,Y}[\ell(Y, f(X))]$ of the multi-target regressor f, which is equal to a sum of n expected single-target losses

$$\mathbb{E}_{X,Y}[\ell(Y, f(X))] = \mathbb{E}_{X,Y}[\sum_{j=1}^{n} \ell_j(Y_j, f_j(X))] = \sum_{j=1}^{n} \mathbb{E}_{X,Y_j}[\ell_j(Y_j, f_j(X))], \tag{5}$$

whenever we chose a multi-target loss function that satisfies equation (4).

In any empirical work, we must estimate each expected single-target loss to obtain an estimate of the expected multi-target loss, since we cannot directly calculate an expectation value in the distribution-free learning model. Consequently, we split equation (1) into m_{train} and m_{test} training and test examples, respectively, such that $m = m_{\text{train}} + m_{\text{test}}$. Then, we estimate the expected multi-target loss by

$$\frac{1}{n} \sum_{j=1}^{n} \left(\frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \ell_j(Y_j^{(i)}, f_j(X^{(i)})) \right), \tag{6}$$

where the quantity in parenthesis estimates the expected jth single-target loss. In particular, if the multi-target loss function is absolute error (equation (2)), then the quantity in parenthesis is known as the mean absolute error and equation (6) is known as the average mean absolute error. Similarly, if the multi-target loss function is squared error (equation (3)), then the quantity in parenthesis is known as the mean squared error and equation (6) is known as the average mean squared error.

2.2. Conventional multi-target stacking

Let us begin with the independent model approach, which is a straightforward way to extend supervised learning algorithms to the setting of multi-target regression, when they do not natively induce a real vector-valued function. That is, in this approach, a learning machine receives training examples, as well as hyperparameters. Then, for each single-target regression subtask, it slices single-target training examples and invokes an embedded learning algorithm to search a space of single-target regressors $f_j : \mathcal{X} \to \mathbb{R}$ to identify the best real-valued function in the space. Lastly, the learning machine concatenates the fitted single-target regressors into a multi-target regressor $\hat{f} = (\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n)$. Thusly, it outputs the resultant multi-target regressor, which predicts a multi-target, given an instance

$$\hat{Y} = \hat{f}(X) = \left(\hat{f}_1(X), \hat{f}_2(X), \ldots, \hat{f}_n(X)\right) \in \mathcal{Y},$$

where each entry is the image of an instance under a fitted single-target regressor.

To outperform the naïve independent model approach, the conventional multi-target stacking approach forms a composition of real vector-valued functions through consecutive invocations of the independent model approach. That is, in this approach, a learning machine receives training examples, as well as hyperparameters. Next, it induces a multi-target regressor $\hat{f}$ with the independent model approach. Subsequently, the learning machine replaces each instance $X^{(i)}$ in the training examples with either a prediction $\hat{Y}^{(i)} = \hat{f}(X^{(i)})$ or a concatenation of a prediction and instance $(\hat{Y}^{(i)}, X^{(i)})$, where $i = 1, 2, \ldots, m_{\text{train}}$. Then, it induces a multi-target regressor $\hat{g}$ with the independent model approach. Lastly, the learning machine composes the multi-target regressors. Thusly, it outputs the resultant composition of multi-target regressors, which predicts a multi-target, given an instance, by either

$$\hat{Y} = \hat{g}(\hat{f}(X)) = \left(\hat{g}_1(\hat{f}(X)), \hat{g}_2(\hat{f}(X)), \ldots, \hat{g}_n(\hat{f}(X))\right) \in \mathcal{Y},$$

or

$$\hat{Y} = \hat{g}(\hat{f}(X), X) = \left(\hat{g}_1(\hat{f}(X), X), \hat{g}_2(\hat{f}(X), X), \ldots, \hat{g}_n(\hat{f}(X), X)\right) \in \mathcal{Y},$$

according to the instance replacement strategy.

2.3. Variant of multi-target stacking

In our variant of multi-target stacking, we compose an existing real vector-valued model and a real vector-valued function from a modified gradient boosting approach, where we bypass the formation of the first compositional layer. That is, in our approach, we start with data in the form of equation (1), and we wrangle it into an $m \times 2n$ design matrix[5]

$$\left(\begin{array}{c|c} \text{———} X^{(1)} \text{———} & \text{———} Y^{(1)} \text{———} \\ \text{———} X^{(2)} \text{———} & \text{———} Y^{(2)} \text{———} \\ \vdots & \vdots \\ \text{———} X^{(m)} \text{———} & \text{———} Y^{(m)} \text{———} \end{array}\right), \tag{7}$$

where $X \in \mathcal{X} \subseteq \mathbb{R}^n$ denotes a multi-target prediction made by the existing model, $Y \in \mathcal{Y} \subseteq \mathbb{R}^n$ denotes a multi-target, and X_j corresponds to the single-target prediction of Y_j. Next, we split equation (7) into m_{train} and m_{test} rows such that $m = m_{\text{train}} + m_{\text{test}}$. Subsequently, a learning machine receives training examples with shape $m_{\text{train}} \times 2n$, as well as hyperparameters. Then, it slices single-target training examples with shape $m_{\text{train}} \times (n+1)$

[5] Notedly, we present our variant of multi-target stacking without the instances from the original domain, but it is straightforward to incorporate them, as in the conventional multi-target stacking approach.

$$\left(\begin{array}{c|c} \text{———}X^{(1)}\text{———} & Y_j^{(1)} \\ \text{———}X^{(2)}\text{———} & Y_j^{(2)} \\ \vdots & \vdots \\ \text{———}X^{(m_{\text{train}})}\text{———} & Y_j^{(m_{\text{train}})} \end{array}\right) \tag{8}$$

and invokes an embedded boosting machine to induce a single-target regressor for each single-target regression subtask, where $j = 1, 2, \ldots, n$. Lastly, the learning machine concatenates the fitted single-target regressors into a multi-target regressor $\hat{f} = (\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n)$. Thusly, it outputs the resultant multi-target regressor, which predicts a multi-target, given a multi-target prediction from the existing model

$$\hat{Y} = \hat{f}(X) = (\hat{f}_1(X), \hat{f}_2(X), \ldots, \hat{f}_n(X)) \in \mathcal{Y}.$$

In the next subsection, we review the boosting machine, which is the standard formulation of gradient boosting. Then, in the subsequent subsection, we introduce our modification to the boosting machine, as well as an augmented gradient boosting algorithm that includes a built-in model selection step, which allows us to specify the embedded boosting machine and complete the learning framework discussion.

2.3.1. Boosting machine: the standard formulation of gradient boosting

From the foregoing, let $V_j = \mathbb{R}^{\mathcal{X}}$ be the set of all single-target regressors under the natural operations of addition of two functions and multiplication of a function by a real number. We equip V_j with the following inner product

$$\langle f_j, g_j \rangle = \sum_{i=1}^{m_{train}} f_j(X^{(i)}) g_j(X^{(i)}),$$

so it is a Hilbert space of single-target regressors, where $||f_j|| = \sqrt{\langle f_j, f_j \rangle}$ denotes the norm induced by the inner product. Let $H_j \subseteq V_j$ denote a nonempty set of basis functions that is closed under scalar multiplication, where $\text{span}(\{H_j\})$ denotes the smallest subspace, which contains the set of basis functions. Also, let $\mathfrak{l}_j : V_j \to \mathbb{R}$ be an empirical loss functional defined by

$$\mathfrak{l}_j(f_j) = \sum_{i=1}^{m_{\text{train}}} \ell_j(Y_j^{(i)}, f_j(X^{(i)})),$$

where the single-target loss function on the right-hand side is assumed to be amenable to gradient-based methods. Then, the goal of functional gradient descent is to minimize the empirical loss function by taking steps in the direction of steepest descent subject to the constraint that the single-target regressor must be a linear combination of well-defined basis functions. Notedly, this constraint guarantees that the resultant single-target regressor is a well-defined function; and, in what follows, we refer to the linear combination as an additive expansion.

Before we derive the method of functional gradient descent, we must formulate the inner product between the gradient of the empirical loss functional at a single-target regressor and an arbitrary single-target regressor, since this mathematical object appears in a quadratic approximation that underlies the derivation. To do so, it is convenient to think of the derivative

$$D \in C^k(V_j) \to (\mathbb{R}^{m_{\text{train}}} \to (\mathbb{R}^{m_{\text{train}}} \to \mathbb{R}^n))$$

as a map from the space of functions on the Hilbert space with k continuous derivatives to the space of all maps from the vector space $\mathbb{R}^{m_{\text{train}}}$ to the space of all linear maps from the vector space $\mathbb{R}^{m_{\text{train}}}$ to $\mathbb{R}^n$, where k is as large as necessary. So the gradient

$$\nabla \in C^k(V_j) \to (\mathbb{R}^{m_{\text{train}}} \to (\mathbb{R}^{m_{\text{train}}})^*)$$

is a special case from which we can formulate the gradient of the empirical loss functional

$$\nabla(\mathfrak{l}_j) \in (R^{m_{\text{train}}} \to (R^{m_{\text{train}}})^*),$$

where $(\mathbb{R}^{m_{\text{train}}})^*$ denotes the dual space $(\mathbb{R}^{m_{\text{train}}} \to \mathbb{R})$, which consists of linear maps from the vector space $\mathbb{R}^{m_{\text{train}}}$ to the ground field of real numbers $\mathbb{R}$. Then, the gradient of the empirical loss functional at a single-target regressor

$$(\nabla(\mathfrak{l}_j))(f_j) \in (R^{m_{\text{train}}})^*$$

is a linear functional. Hence, the Riesz representation theorem enables us to formulate the inner product between the gradient of the empirical loss functional at a single-target regressor and an arbitrary single-target regressor

$$\langle((\nabla(\mathfrak{l}_j))(f_j), g_j\rangle = (\nabla(\mathfrak{l}_j))(f_j)(g_j),$$

where $f_j, g_j \in V_j$. Thus, we are ready to derive the method of functional gradient descent.

To begin the derivation, let $f_{j,k} \in \text{span}(\{H_j\})$ denote the current additive expansion, which follows from an integral number k of functional gradient descent updates to an initial additive expansion $f_{j,0} \in H_j$. Now, suppose we wish to improve upon the current additive expansion with a functional gradient descent update. To do so, we replace the objective function in the constrained optimization problem

$$\min_{b \in H_j} \mathfrak{l}_j(f_{j,k} + b)$$

with a quadratic approximation that is equivalent to a second-order Taylor approximation of the empirical loss functional at the current additive expansion

$$\min_{b \in H_j} \mathfrak{l}_j(f_{j,k}) + \langle((\nabla(\mathfrak{l}_j))(f_{j,k}), f_{j,k} + b - f_{j,k}\rangle + \frac{1}{2}||f_{j,k} + b - f_{j,k}||^2,$$

where we assume the Hessian of the empirical loss functional at the current additive expansion

$$(\nabla^2(\mathfrak{l}_j))(f_{j,k}) \in (R^{m_{\text{train}}} \to (R^{m_{\text{train}}})^*)$$

satisfies

$$(\nabla^2(\mathfrak{l}_j))(f_{j,k}) : f_{j,k} + b - f_{j,k} \mapsto f_{j,k} + b - f_{j,k}$$

so the the Riesz representation theorem implies the rightmost summand

$$\frac{1}{2}\langle((\nabla^2(\mathfrak{l}_j))(f_{j,k})(f_{j,k} + b - f_{j,k}),\, f_{j,k} + b - f_{j,k}\rangle = \frac{1}{2}\langle f_{j,k} + b - f_{j,k},\, f_{j,k} + b - f_{j,k}\rangle = \frac{1}{2}||f_{j,k} + b - f_{j,k}||^2.$$

Notedly, Zheng *et al* put forth this line of reasoning in a coordinate-dependent way, then Grubb and Bagnell put forth this line of reasoning in a coordinate-independent way, where they study the convergence analysis of functional gradient descent [31, 32].

To complete the derivation, we manipulate the approximant into an equivalent, but more manageable, form

$$\min_{b \in H_j} ||(f_{j,k} + r_{j,k}) - (f_{j,k} + b)||^2 = \min_{b \in H_j} ||r_{j,k} - b||^2, \tag{9}$$

where $r_{j,k} = -(\nabla(\mathfrak{l}_j))(f_{j,k})$ denotes the pseudo-residual. Then, we invoke a learning algorithm to fit a basis function to the pseudo-residual in the least squares sense, since the pseudo-residual does not generalize to out-of-sample instances

$$X \in \mathcal{X} \setminus \{X^{(1)}, X^{(2)}, \ldots, X^{(m_{\text{train}})}\} = \{X : X \in \mathcal{X} \text{ and } X \notin \{X^{(1)}, X^{(2)}, \ldots, X^{(m_{\text{train}})}\}\},$$

as pointed out by Friedman [7, 11]. Lastly, we append a weighted version of the fitted basis function to the current additive expansion to update the additive expansion.

To implement the method of functional gradient descent with the boosting machine, we fit an additive expansion

$$f_j(X) = \alpha_{j,0} + \sum_{k=1}^{K_j} \alpha_{j,k} b(X; \theta_{j,k})$$

through forward stagewise additive modeling, where $\alpha_{j,0}$ denotes a constant term, $\alpha_{j,k}$ denotes an expansion coefficient, $b(X; \theta_{j,k})$ denotes a basis function, which is characterized by a set of parameters $\theta_{j,k}$, and $k = 1, 2, \ldots, K_j$. That is, fitting begins with a constant function $f_{j,0}(X) = c$, where the constant offset value c is usually equal to 0 or the optimal constant model $\text{argmin}_c \sum_{i=1}^{m_{\text{train}}} \ell_j(Y_j^{(i)}, c)$. Next, a for loop executes the following statements $k = 1, 2, \ldots, K_j$ times:

(a) For $i = 1, 2, \ldots, m_{\text{train}}$, compute the pseudo-residual and evaluate at the current additive expansion

$$r_{j,k}^{(i)} = -\left.\frac{\partial \ell_j(Y_j, f_j(X^{(i)}))}{\partial f_j(X^{(i)})}\right|_{f_j(X^{(i)})=f_{j,k-1}(X^{(i)})}.$$

(b) Invoke a learning algorithm to fit a basis function to $\{X^{(i)}, r_{j,k}^{(i)}\}_{i=1}^{m_{\text{train}}}$ and learn the set of parameters $\theta_{j,k}$.
(c) Invoke an optimization algorithm to determine the expansion coefficient

$$\alpha_{j,k} = \underset{\alpha}{\text{argmin}} \sum_{i=1}^{m_{\text{train}}} \ell_j(Y_j^{(i)}, f_{j,k-1}(X^{(i)}) + \alpha b(X^{(i)}; \theta_{j,k})).$$

(d) Update the additive expansion $f_{j,k}(X) = f_{j,k-1}(X) + \alpha_{j,k} b(X; \theta_{j,k})$.

Lastly, the boosting machine returns the fitted additive expansion $\hat{f}_j = f_{j,K_j}$.

In the following subsection, we show that the standard formulation of gradient boosting generates a sequence of improvements to a constant model. In turn, we show how to formulate gradient boosting so that it generates a sequence of improvements to an existing, possibly nonconstant, model. Then, we introduce an augmented gradient boosting algorithm, which ensures robustness of our modification to the boosting machine.

2.3.2. Modification to the boosting machine

To show that the standard formulation of gradient boosting generates a sequence of improvements to a constant model, we split the proof in to two cases. In the first case, we assume the constant offset value is $c = 0$. As such, in iteration $k = 1$, the loop body updates the additive expansion to a weighted basis function. Then, in iteration $k = 2$, the loop body updates the additive expansion to a weighted basis function plus a weighted basis function, and so on. Thus, we deduce the following sequence of improvements to the constant zero model

$$0,\ \alpha_{j,1} b(X; \theta_{j,1}),\ \alpha_{j,1} b(X; \theta_{j,1}) + \alpha_{j,2} b(X; \theta_{j,2}),\ \ldots,\ \sum_{k=1}^{K_j} \alpha_{j,k} b(X; \theta_{j,k}).$$

In the second case, we assume the constant offset value is $c \neq 0$, and without loss of generality, we suppose it is equal to a nonzero sample mean $\bar{Y}_j = \text{argmin}_c \sum_{i=1}^{m_{\text{train}}} (Y_j^{(i)} - c)^2$. As such, in iteration $k = 1$, the loop body updates the additive expansion to the sample mean plus a weighted basis function. Then, in iteration $k = 2$, the loop body updates the additive expansion to the sample mean plus a weighted basis function plus a weighted basis function, and so on. Thus, we deduce the following sequence of improvements to the constant nonzero model

$$\bar{Y}_j,\ \bar{Y}_j + \alpha_{j,1} b(X; \theta_{j,1}),\ \bar{Y}_j + \alpha_{j,1} b(X; \theta_{j,1}) + \alpha_{j,2} b(X; \theta_{j,2}),\ \ldots,\ \bar{Y}_j + \sum_{k=1}^{K_j} \alpha_{j,k} b(X; \theta_{j,k}),$$

which follows from Friedman's original formulation of *LS_Boost*, as well as Bühlmann and Yu's formulation of *L_2Boosting* with the caveat that an expansion coefficient $\alpha_{j,k} = \nu$ corresponds to a shrinkage parameter $0 < \nu \leq 1$, where $k = 1, 2, \ldots, K_j$ [7, 9, 10] Therefore, we conclude that the standard formulation of gradient boosting always generates a sequence of improvements to a constant model.

To formulate gradient boosting so that it is able to generate a sequence of improvements to the existing model

$$X_j,\ X_j + \alpha_{j,1} b(X; \theta_{j,1}),\ X_j + \alpha_{j,1} b(X; \theta_{j,1}) + \alpha_{j,2} b(X; \theta_{j,2}),\ \ldots,\ X_j + \sum_{k=1}^{K_j} \alpha_{j,k} b(X; \theta_{j,k}), \tag{10}$$

we propose to fit the following additive expansion[6]

$$f_j(X) = \pi_j(X) + \sum_{k=1}^{K_j} \alpha_{j,k} b(X; \theta_{j,k}) = X_j + \sum_{k=1}^{K_j} \alpha_{j,k} b(X; \theta_{j,k}) \tag{11}$$

[6] As an aside, if H_j denotes a family of kernels, X_j denotes a baseline value of a molecular property, $K_j = 1$, and $a_{j,1} = 1$, then equation (11) recovers a model that appears in the Δ-machine learning approach to quantum chemistry [33].

Algorithm 1: *BaseBoost*

Input: single-target training examples
number of iterations K_j
single-target loss functions $\{\ell_j, \tilde{\ell}_j\}$
learning algorithm
optimization algorithm
Initialize: $f_{j,0}(X) = \pi_j(X) = X_j$ with the jth coordinate functional
for $k = 1$ **to** K_j **do**
(a) **for** $i = 1$ **to** m_{train} **do**
Compute the pseudo-residual and evaluate at the current additive expansion
$$r_{j,k}^{(i)} = -\frac{\partial \ell_j(Y_j, f_j(X^{(i)}))}{\partial f_j(X^{(i)})}\bigg|_{f_j(X^{(i)})=f_{j,k-1}(X^{(i)})}$$
end
(b) Invoke the learning algorithm to fit a basis function to $\{X^{(i)}, r_{j,k}^{(i)}\}_{i=1}^{m_{train}}$ and learn the set of parameters $\theta_{j,k}$
(c) Invoke the optimization algorithm to learn the expansion coefficient
$\alpha_{j,k} = \text{argmin}_\alpha \sum_{i=1}^{m_{train}} \tilde{\ell}_j(Y_j^{(i)}, f_{j,k-1}(X^{(i)}) + \alpha b(X^{(i)}; \theta_{j,k}))$
(d) Update the additive expansion
$f_{j,k}(X) = f_{j,k-1}(X) + \alpha_{j,k} b(X; \theta_{j,k})$
end
Output: Fitted additive expansion $\hat{f}_j = f_{j,K_j}$

Figure 2. A single-target regression boosting machine.

Algorithm 2: *BaseBoostCV*

Input: single-target training examples
number of iterations K_j
single-target loss functions $\{\ell_j, \tilde{\ell}_j\}$
learning algorithm
optimization algorithm
number of cross-validation folds k
k-fold cross-validation single-target loss function
Model selection: Evaluate the incumbent existing model and the candidate *BaseBoost* with an augmented k-fold cross-validation procedure. If the incumbent error is less than or equal to the cross-validation error of *BaseBoost*, then break $\hat{f}_j(X) = \pi_j(X) = X_j$. Otherwise, call *BaseBoost* to induce $\hat{f}_j$.
Output: Fitted single-target regressor $\hat{f}_j$

Figure 3. A single-target regression boosting machine with built-in model selection.

through forward stagewise additive modeling, where $\pi_j : X \mapsto X_j$ denotes the jth coordinate functional, which maps an instance to the jth single-target prediction. That is, fitting proceeds as before with the caveat that it begins with the jth coordinate functional $f_{j,0}(X) = \pi_j(X) = X_j$ in place of a constant function[7]. As such, in iteration $k = 1$, the loop body updates the additive expansion to the jth single target prediction plus a weighed basis function. Then, in iteration $k = 2$, the loop body updates the additive expansion to jth single target prediction plus a weighed basis function plus a weighed basis function, and so on; see the pseudocode in figure 2. Thus, we deduce equation (10).

To ensure robustness of our modification to the boosting machine, we introduce an augmentation to Algorithm 1, which begins with a built-in model selection step; see the pseudocode in figure 3. That is, the built-in model selection step randomly partitions equation (8) into k non-overlapping folds, where k is typically a natural number between 5 and 10, inclusive; see figure 4 for an illustration. Next, it repeats the following two steps k times with each of the withheld folds used exactly once as the validation examples:

- Of the k folds, withhold one for validation. Supply Algorithm 1 with the remaining $k-1$ folds, as well as hyperparameters.
- Independently evaluate the incumbent single-target regression predictions made by the existing model and the candidate fitted additive expansion on the withheld fold from the previous step by computing the average withheld fold loss of the candidate and incumbent.

[7] We conjecture the existing model accelerates functional gradient descent, whenever

$$\sum_{i=1}^{m_{\text{train}}} \ell_j(Y_j^{(i)}, c) > \sum_{i=1}^{m_{\text{train}}} \ell_j(Y_j^{(i)}, \pi_j(X^{(i)})) = \sum_{i=1}^{m_{\text{train}}} \ell_j(Y_j^{(i)}, X_j^{(i)}).$$

.

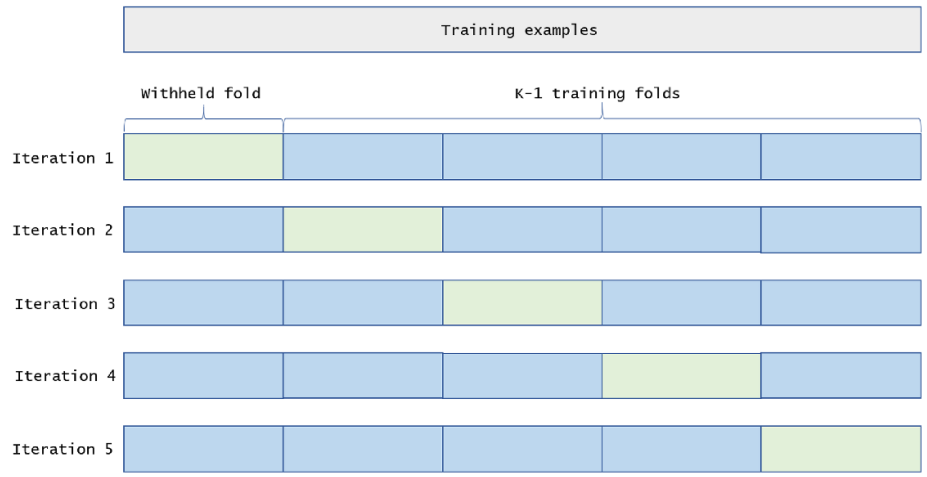

Figure 4. Illustration of an augmented 5-fold cross-validation procedure. We partition the single-target training examples (equation (8)) into five non-overlapping folds of equal size. In each iteration, we supply Algorithm 1 with four training folds, shown in light blue, then we independently evaluate the single-target regression predictions made by the existing model and the fitted additive expansion on the withheld fold, shown in light green. Lastly, we independently average their five withheld fold results, and we refer to these averages as the incumbent and cross-validation error, respectively.

Subsequently, the built-in model selection step independently averages the incumbent and candidate withheld fold results, and we refer to these averages as the incumbent and cross-validation error, respectively. Then, it selects the existing model, whenever the incumbent error is less than or equal to the cross-validation error. Namely, it breaks and returns the fitted single-target regressor $\hat{f}_j(X) = \pi_j(X) = X_j$. Otherwise, it selects Algorithm 1, which means it returns the fitted single-target regressor $\hat{f}_j = f_{j,K_j}$.

To complete the learning framework discussion, let us recall the step in which the learning machine invokes an embedded boosting machine. Notedly, we intend embedded boosting machine to mean either Algorithm 1 or Algorithm 2.

3. Benchmark task

In this section, we review the calibration of a nearest-neighbor coupled linear array of superconducting qubits, then we report the results from an experiment on a real-world calibration dataset. In the experiment, we compare our learning framework against the state-of-the-art calibration model, which contains an explicit description of the model Hamiltonian, as well as our learning framework, where the embedded boosting machine is an off-the-shelf gradient boosting algorithm, known as LightGBM, which initializes its fit of an additive expansion with a constant model [26–29].

3.1. Superconducting quantum device calibration procedure

In what follows, we study a nearest-neighbor coupled linear array of superconducting qubits with tunable qubit frequencies and inter-qubit interactions, where each qubit belongs to the span of the ground and first excited states of a nonlinear photonic resonator in the microwave regime. The total Hamiltonian of the nearest-neighbor coupled linear array is approximately described by the Bose–Hubbard model truncated at two local excitations

$$\begin{aligned}\mathcal{H} = &\sum_{j=1}^{n} \delta_j \hat{a}_j^\dagger \hat{a}_j + \frac{L}{2} \hat{a}_j^\dagger \hat{a}_j (\hat{a}_j^\dagger \hat{a}_j - 1) \\ &+ \sum_{j=1}^{n-1} g_{j,j+1} (\hat{a}_j^\dagger \hat{a}_{j+1} + \hat{a}_j \hat{a}_{j+1}^\dagger),\end{aligned} \tag{12}$$

where $n > 1$ is the number of qubits, $\hat{a}^\dagger$ ($\hat{a}$) is the bosonic creation (annihilation) operator, δ_j is the random on-site detuning, L is the on-site Hubbard interaction, and $g_{j,j+1}$ is the hopping rate between nearest neighbor sites[8].

[8] Notedly, it is possible to translate quantum evolution into the prototypical quantum circuit model [27].

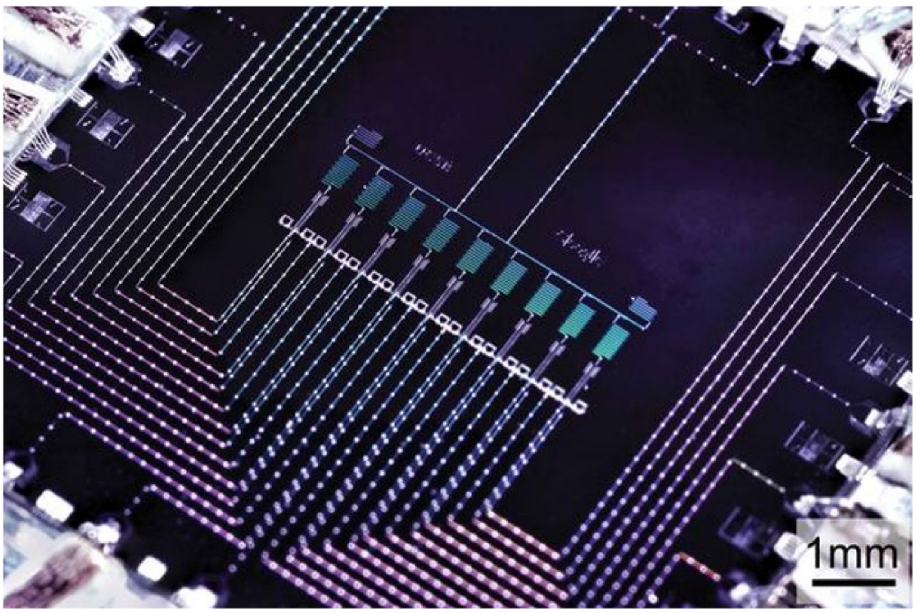

Figure 5. Optical micrograph of the superconducting quantum device, which shows a nearest-neighbor coupled linear array of nine superconducting qubits and eight interleaving couplers, as well as twenty-six control lines and nine readout resonators. Notedly, the four leftmost qubits and couplers were idle and the $n = 5$ rightmost qubits and four couplers were operational during the generation of the calibration dataset. From [26]; reprinted with permission from AAAS.

The calibration objective is to learn how to transform time-dependent control pulses, which emanate outside of the cryostat, i.e. the cooling device that encloses the superconducting quantum device, to entries in a matrix representation of equation (12). To do so, a calibration procedure proceeds in two steps [27]. In the first step, the procedure calibrates room temperature time-dependent control pulses to arrive orthogonally, synchronously, and without distortion at the superconducting quantum device. In the second step, the procedure infers a finite number of control model parameters through three substeps. In the first substep, the procedure fits the two lowest transition energies of each qubit as a function of qubit and coupler flux-biases. In the second substep, it benchmarks the collective dynamics of the superconducting quantum device with a many-body Ramsey spectroscopy technique such that all of the qubits are coupled and near resonance with each other [26]. Then, in the third substep, it invokes a numerical optimizer to minimize the absolute error between the measured and sorted eigenenergies obtained in the previous substep and the energy spectrum predictions made by the control model. Thus, the third substep gives rise to a multi-target regression problem, which is the focus of the next subsection.

3.2. Experiment on calibration dataset

The calibration dataset pertains to the second step in the aforedescribed calibration procedure. Figure 5 shows the nearest-neighbor coupled linear array of nine superconducting qubits and eight interleaving couplers, where the four leftmost qubits and couplers were idle and the $n = 5$ rightmost qubits and four couplers were operational during the data collection process. The relevant constituents include 136 of each: a collection of five qubit and four coupler biases, which was the input to the optimized control model, an energy spectrum prediction, which was the output from the optimized control model, and an instance of the many-body Ramsey spectroscopy technique, which measured and ascendingly ordered five eigenenergies belonging to equation (12), when it describes a photon hopping in a disordered potential; see appendix A.

With regard to the learning framework, we wrangle the data into equation (7) with shape 136×10, where $\mathcal{X} \subseteq \mathbb{R}^5$ denotes the domain of spectroscopic predictions from the optimized control model, $\mathcal{Y} \subseteq \mathbb{R}^5$ denotes the codomain of measured and ascendingly ordered eigenenergies, and figure 6 shows pairwise correlations of the columns. We randomly split equation (7) into $m_{\text{train}} = 95$ and $m_{\text{test}} = 41$ rows such that equation (8) has shape $95 \times (5 + 1)$ in each single-target regression subtask, where $j = 1, 2, \ldots, 5$. We explain how to access the remaining implementation details in appendix D.

Figure 7 depicts the built-in model selection step in Algorithm 2 with an augmented learning curve, where the single-target regression subtask is Y_3 and we measure accuracy with absolute error, so the appropriate unit is megahertz (MHz). That is, the augmented learning curve shows the training, cross-validation, and incumbent error in blue, orange, and red, respectively, where the single-target training example sizes vary between 23 and 95 ordered pairs, inclusive, and the incumbent error bounds the

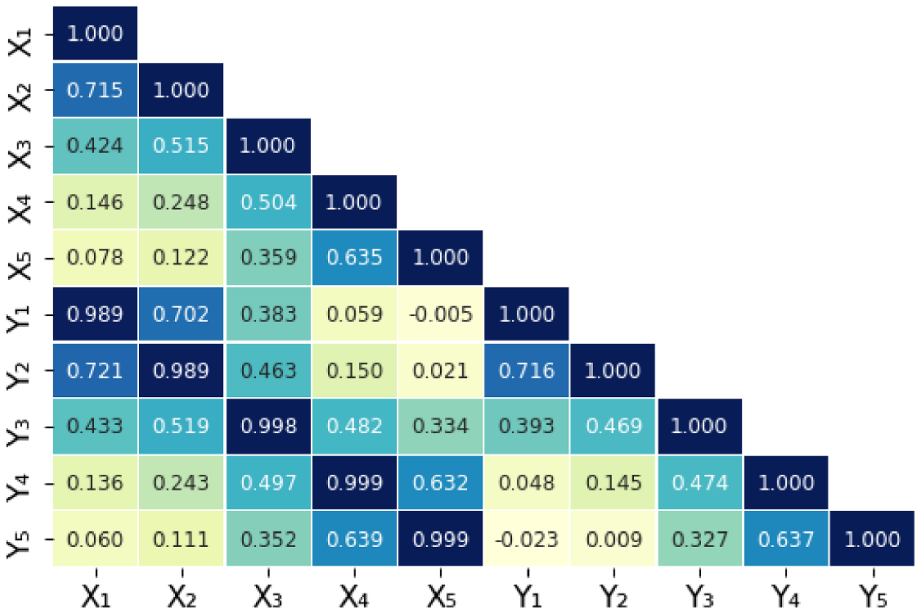

Figure 6. Pairwise correlations of the design matrix columns. The notation X_j refers to a single-target prediction from the state-of-the-art calibration model [26–28] and Y_j refers to a single-target from an instance of a many-body Ramsey spectroscopy technique [26], where each spectroscopic instance sorts the eigenenergies belonging to equation (12) into ascending order: $Y_1 \leq Y_2 \leq \cdots \leq Y_5$. Of note is the strong pairwise correlation between certain single-targets, such as Y_1 and Y_2, as well as single-target predictions and single-targets, such as X_1 and Y_2.

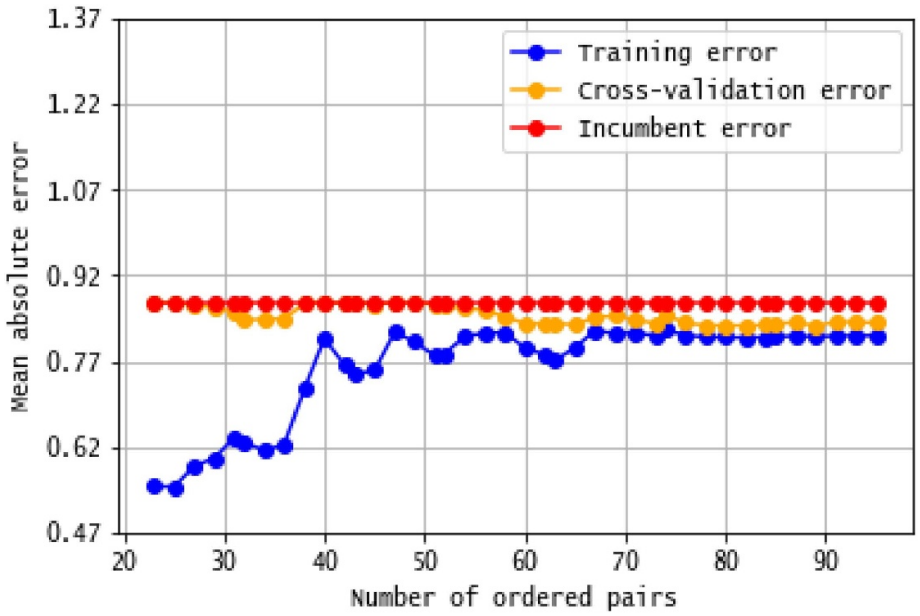

Figure 7. Augmented learning curve for the single-target regression subtask Y_3, where we measure accuracy with absolute error, so the appropriate unit is megahertz (MHz); see figures 18 and 19 in appendix B for the other single-target regression subtasks. We depict the built-in model selection step in Algorithm 2, where the single-target training example sizes vary between 23 and 95 ordered pairs, inclusive, and the incumbent error bounds the cross-validation error from above by 0.87 MHz due to the model selection criterion. Evidently, the built-in model selection step tended to chose the state-of-the-art calibration model [26–28], when there were less than 51 ordered pairs, and it always chose Algorithm 1, when there were 51, or more, ordered pairs. Notedly, the training and cross-validation errors tend to increase and decrease, respectively, and both errors exhibit random fluctuations that eventually taper off.

cross-validation error from above by 0.87 MHz due to the model selection criterion. Evidently, the built-in model selection step tended to chose the optimized control model, when there were less than 51 ordered pairs, and it always chose Algorithm 1, when there were 51, or more, ordered pairs. In general, the built-in model selection step always chose Algorithm 1, when there were 60, or more, ordered pairs in any single-target regression subtask; see figures 18 and 19 in appendix B.

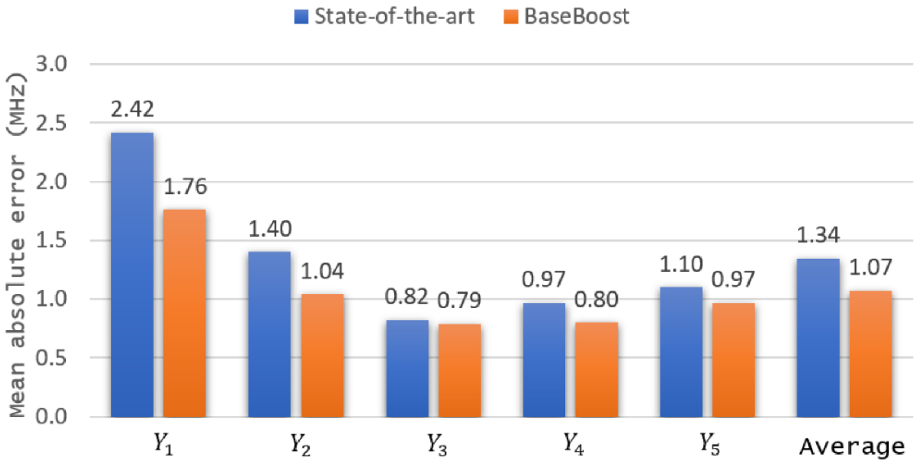

Figure 8. Evaluation of the state-of-the-art calibration model [26–28] and our learning framework on test examples, where the embedded boosting machine is Algorithm 2 such that Algorithm 2 called Algorithm 1 in every single-target regression subtask. The bar plot shows that our learning framework performs better than the calibration model on the multi-target regression task, as well as every single-target regression subtask, where we measure accuracy with absolute error. Notedly, Roushan *et al* estimate the minimal average mean absolute error to be 1 MHz [26].

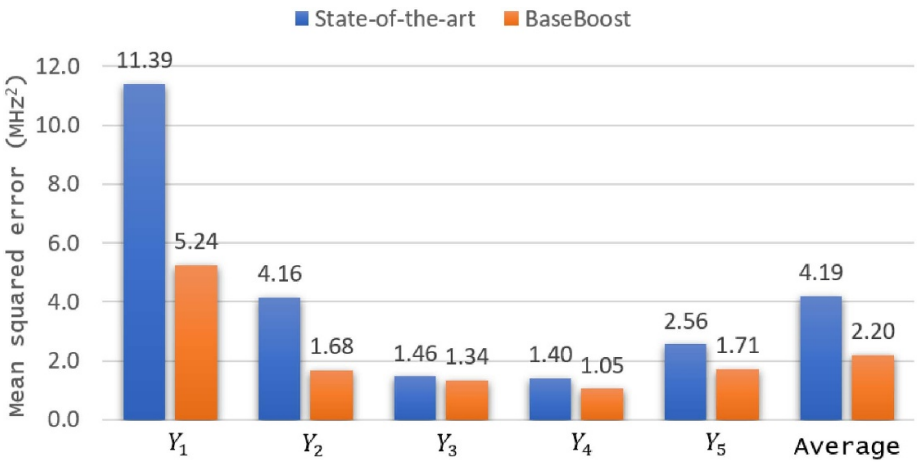

Figure 9. Evaluation of the state-of-the-art calibration model [26–28] and our learning framework on test examples, where the embedded boosting machine is Algorithm 2 such that Algorithm 2 called Algorithm 1 in every single-target regression subtask. The bar plot shows that our learning framework performs better than the calibration model on the multi-target regression task, as well as every single-target regression subtask, where we measure accuracy with squared error.

Prior to evaluation on test examples, we induce two multi-target regressors with our learning framework. That is, the first multi-target regressor is the resultant of Algorithm 2, as the embedded boosting machine, where Algorithm 2 calls Algorithm 1 with $K_j = 1$ in every single-target regression subtask. The second multi-target regressor is the resultant of LightGBM, as the embedded boosting machine. Figures 8 and 9 compare the first multi-target regressor against the optimized control model on test examples, where we measure accuracy with absolute and squared error, respectively. Figures 10 and 11 compare the first multi-target regressor against the second multi-target regressor on test examples, where we measure accuracy with absolute and squared error, respectively. In summary, the first multi-target regressor outperforms the optimized control model, and they both significantly outperform the second multi-target regressor.

4. Explainable machine learning

In this section, we describe an entirely data-driven reimplementation of the optimized control model, and we compare it against our learning framework. Then, we apply the Shapley additive explanations approach to uncover parameter dependencies in the optimized control model [30].

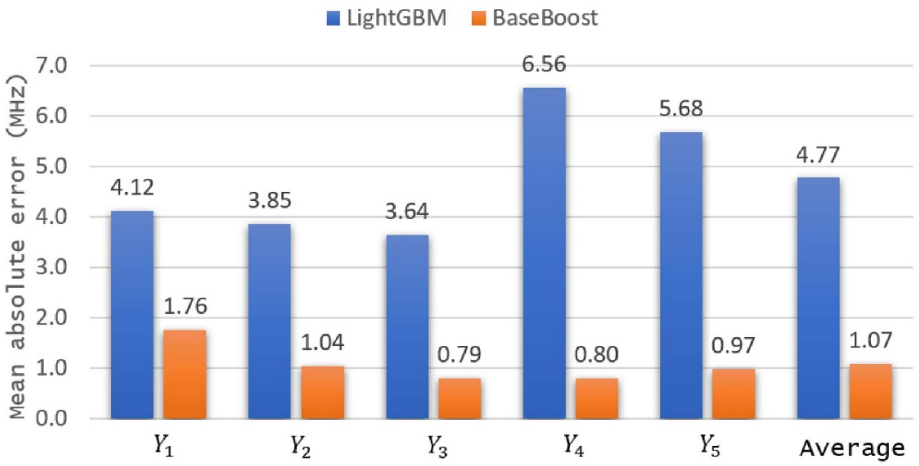

Figure 10. Evaluation of our learning framework on test examples, where the embedded boosting machine is either LightGBM [29] or Algorithm 2 such that Algorithm 2 called Algorithm 1 in every single-target regression subtask. The bar plot shows that our learning framework performs significantly better with our formulation of gradient boosting, where we measure accuracy with absolute error. Further, a comparison with figure 8 shows that LightGBM significantly degrades the energy spectrum predictions from the state-of-the-art calibration model [26–28]; and LightGBM is much more prone to overfitting in the single-target regression subtasks Y_4 and Y_5, which have the highest single-target sample standard deviations.

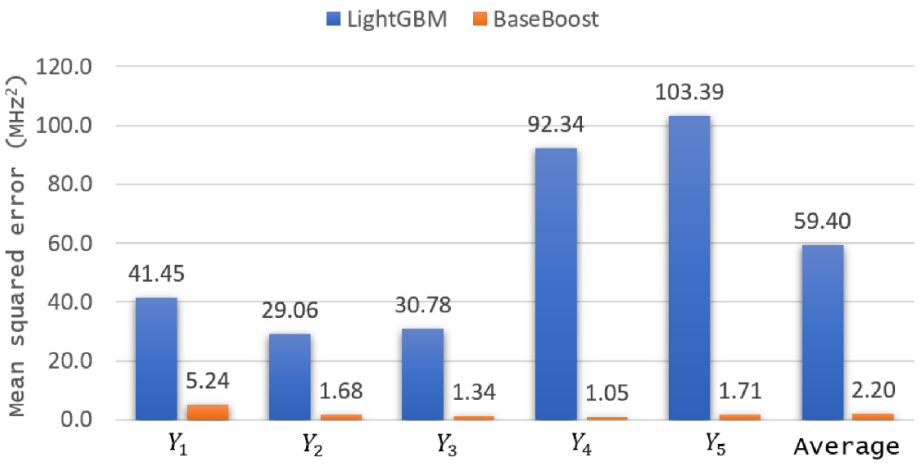

Figure 11. Evaluation of our learning framework on test examples, where the embedded boosting machine is either LightGBM [29] or Algorithm 2 such that Algorithm 2 called Algorithm 1 in every single-target regression subtask. The bar plot shows that our learning framework performs significantly better with our formulation of gradient boosting, where we measure accuracy with squared error. Further, a comparison with figure 9 shows that LightGBM significantly degrades the energy spectrum predictions from the state-of-the-art calibration model [26–28]; and LightGBM is much more prone to overfitting in the single-target regression subtasks Y_4 and Y_5, which have the highest single-target sample standard deviations.

4.1. Entirely data-driven reimplementation of the calibration model

In what follows, we refer to constituents from the calibration dataset. That is, we begin with data in the form of equation (1), where $X^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^9$ denotes an instance in the domain of qubit and coupler bias values, $Y^{(i)} \in \mathcal{Y} \subseteq \mathbb{R}^5$ denotes a multi-target in the codomain of measured and ascendingly ordered eigenenergies, and $i = 1, 2, \ldots, 136$, so $m = 136$. Next, we wrangle equation (1) into a design matrix with shape 136×14, and we split it into $m_{\text{train}} = 95$ and $m_{\text{test}} = 41$ rows such that the split indices preserve the association with our previous experiment on the calibration dataset. Subsequently, we induce a multi-target regressor with the independent model approach, where the training examples have shape 95×14, the single-target training examples have shape 95×10, and the embedded learning algorithm is a fully-connected neural network, which follows from model selection [34]; see appendix D.

Prior to evaluation on test examples, let us recall the first multi-target regressor from the experiment on calibration dataset subsection, which is the resultant of Algorithm 2, as the embedded boosting machine, where Algorithm 2 calls Algorithm 1 with $K_j = 1$ in every single-target regression subtask. Figures 12 and 13 compare the first multi-target regressor against the multi-target regressor from the independent model approach on test examples, where we measure accuracy with absolute and squared error, respectively. In

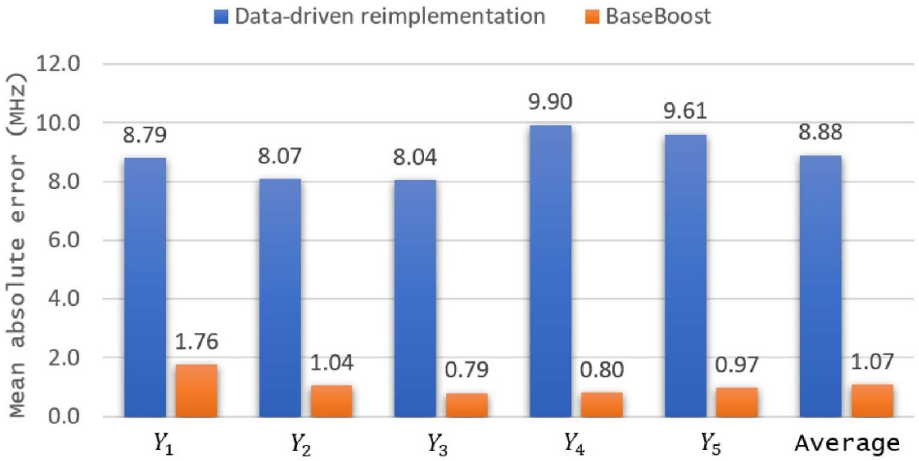

Figure 12. Evaluation of an entirely data-driven reimplementation of the calibration model and our learning framework on test examples, where the embedded boosting machine is Algorithm 2 such that Algorithm 2 called Algorithm 1 in every single-target regression subtask. The bar plot shows that our learning framework performs significantly better than the data-driven reimplementation, where we measure accuracy with absolute error. Further, a comparison with figure 8 shows that the data-driven reimplementation performs significantly worse than the actual calibration model [26–28].

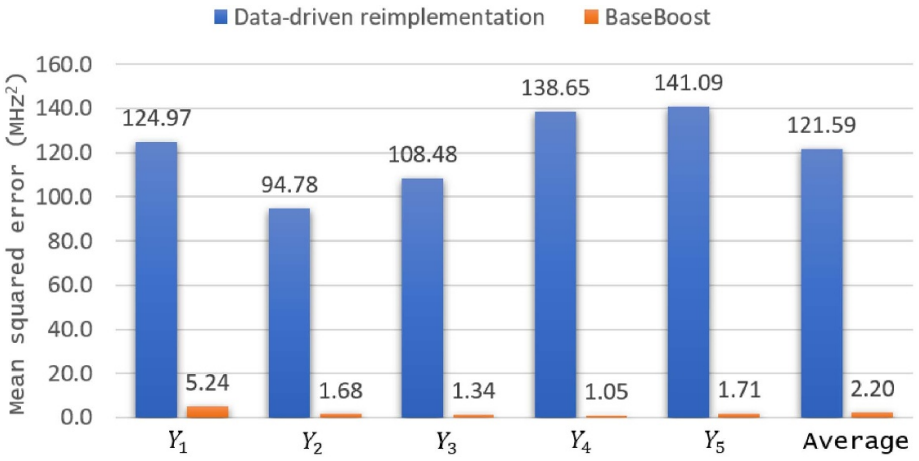

Figure 13. Evaluation of an entirely data-driven reimplementation of the calibration model and our learning framework on test examples, where the embedded boosting machine is Algorithm 2 such that Algorithm 2 called Algorithm 1 in every single-target regression subtask. The bar plot shows that our learning framework performs significantly better than the data-driven reimplementation, where we measure accuracy with squared error. Further, a comparison with figure 9 shows that the data-driven reimplementation performs significantly worse than the actual calibration model [26–28].

summary, the first multi-target regressor significantly outperforms the multi-target regressor from the independent model approach.

4.2. Shapley additive explanations approach

The Shapley additive explanations approach, known as SHAP, natively supports explainable machine learning in the setting of single-target regression. Accordingly, we supply SHAP with the single-target training examples from the previous subsection so that SHAP invokes a fully-connected neural network to induce a single-target regressor, as in the previous subsection. In turn, SHAP supplies an approximation algorithm with single-target training example predictions from the fitted single-target regressor. Then, the approximation algorithm returns nine SHAP values for each single-target training example prediction, where a SHAP value represents the importance of a qubit or coupler bias feature in a particular single-target training example prediction; we breifly review SHAP in appendix C.

Figure 14 shows a summary plot, where the single-target regression subtask is Y_1, the horizontal axis represents the SHAP value, the vertical axis represents the importance of a qubit or coupler bias feature, and the color represents the actual qubit or coupler bias value. Evidently, the most important feature is the qubit 8 bias and the second most important feature is the coupler 5/6 bias, where the former and latter correspond to the 8th qubit site near the physical boundary of the nearest-neighbor coupled linear array and the coupler

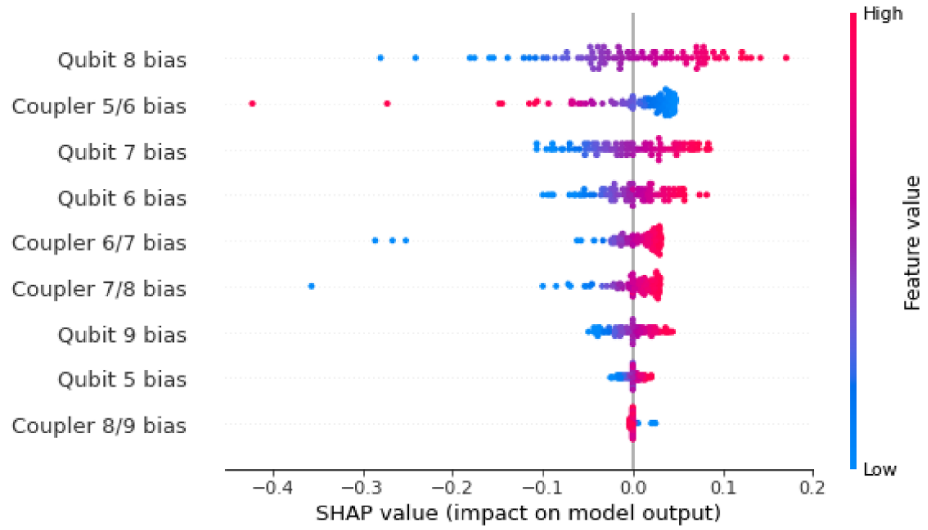

Figure 14. SHAP summary plot for the single-target regression subtask Y_1; see figures 20 and 21 in appendix C for the other single-target regression subtasks. The plot indicates the most important feature is the qubit 8 bias, and the least important feature is the coupler 8/9 bias. Moreover, the qubit 8 bias value tends to be high and the coupler 5/6 bias value tends to be low, when their SHAP value is positive. Similarly, the qubit 8 bias value tends to be low and the coupler 5/6 bias value tends to be high, when their SHAP value is negative.

between the 5th and 6th qubit sites near the artificial boundary of the nearest-neighbor coupled linear array, respectively; see figure 5. In comparison, the most important feature in the single-target regression subtasks Y_3 and Y_4 is the coupler 5/6 bias, and the most important feature in the single-target regression subtasks Y_2 and Y_5 is the coupler 8/9 bias; see figures 20 and 21 in appendix C.

To complete our study of the parameter dependencies in the optimized control model, let us recall that each instance of the many-body Ramsey spectroscopy technique measures and ascendingly orders the eigenenergies. Accordingly, we might expect that on average, over all instances, the feature dependence would be qualitatively similar in each single-target regression subtask. Indeed, under independent and identically distributed sampling of the input parameters, we would expect the data to exhibit a symmetry under permutation among the local bias and coupling parameters in equation (12). In line with this intuition, we observe a noticeably marked dependence on the coupler bias features closest to the physical or artificial boundaries in every single-target regression subtask. However, more generally, the permutation symmetry is broken in the calibration dataset, not least because the model consists of few sites and is patently not well approximated by closed boundary conditions. Some of the individual single-targets, for instance, have a stronger dependence on specific on-site biases than others. This suggest that different sites correlate more strongly with larger or smaller eigenenergies. An example is the aforementioned strong dependence of the single-target Y_1 on the on-site bias at site 8. We attribute this to the geometry of the physical configuration and note that this asymmetric feature dependence is already present in the predictions made by the optimized control model.

5. Conclusion

We have developed a formulation of gradient boosting that is able to leverage domain expertise and improve upon an existing, possibly nonconstant, model. Our formulation is based on a modification to the additive expansion and a corresponding modification to the boosting machine. As a corollary of our work, we have established a template, which enables the reformulation of other boosting algorithms [3, 35–39]. Additionally, we have introduced a variant of multi-target stacking that extends our approach to the setting of multi-target regression, where the results from our experiment indicate the viability of our approach, especially in the low data regime. In future, we plan to evaluate our approach on other datasets. Also, it may be interesting to supplant the constant term in the additive expansion with a real-valued multiple-source prediction term and make the corresponding modification to a boosting algorithm.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/a-wozniakowski/scikit-physlearn.

Acknowledgments

Alex Wozniakowski designed the learning framework, reformulated gradient boosting, developed the open-source, scikit-physlearn repository, performed the data analysis, and prepared the manuscript. All authors contributed to the explanation of the calibration procedure and model and reviewed the manuscript. Moreover, we are grateful to Benjamin Chiaro, who ran the calibration experiment on the superconducting quantum device, collected the calibration dataset, and shared it with us during his time as a graduate student at UC Santa Barbara, and to Pedram Roushan for helpful discussions. This work is supported by the Singapore Ministry of Education Tier 1 Grant RG162/19, Singapore National Research Foundation Fellowship NRF-NRFF2016-02 and NRF-ANR Grant NRF2017-NRF-ANR004 VanQuTe, the Quantum Engineering Program QEP-SF3, and the FQXi large Grant FQXi-RFP-IPW-1903; Alex Wozniakowski was partially supported by the Grant TRT 0159 on mathematical picture language from the Templeton Religion Trust and thanks the Academy of Mathematics and Systems Science (AMSS) of the Chinese Academy of Sciences for their hospitality, where part of this work was done. Felix C. Binder acknowledges funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 801110 and the Austrian Federal Ministry of Education, Science and Research (BMBWF).

Appendix A. Additional calibration dataset details

To generate each multi-target in the calibration dataset, an instance of the many-body Ramsey spectroscopy technique begins by setting the parameters in equation (12) such that the on-site detuning is sampled uniformly in $[-100, 100]$ MHz, the hopping rate is sampled uniformly in $[0, 50]$ MHz, and the on-site Hubbard interaction is fixed at 0. Then, the many-body Ramsey spectroscopy technique iteratively applies the time-domain spectroscopy circuit shown in figure 15 to fully resolve the energy spectrum of equation (12) with randomly programmed parameters, where $k = 1, 2, \ldots, 5$ denotes the choice of superposition qubit and readout resonator.

Namely, in the kth run of the time-domain spectroscopy circuit, every qubit starts in the fiducial state $|0\rangle$ and there is no photon in the system. Next, a microwave pulse is applied to the kth qubit, e.g. $k = 1$ in figure 15, which places the kth qubit in a superposition of the standard basis and initializes a single-photon in the system. Subsequently, the system evolves according to the time-independent Hamiltonian (equation (12)) with randomly programmed parameters. Then, a microwave pulse is applied to the kth qubit to measure either $\langle\sigma^X\rangle$ or $\langle\sigma^Y\rangle$, and from the measurement of these observables the value $\langle\sigma^X\rangle + i\langle\sigma^Y\rangle$ is instantiated.

Once $k = 1, 2, \ldots, 5$ iterations of the time-domain spectroscopy circuit have been completed, there exists an ordered basis of initial states. As such, one can deduce the energy eigenstates through inner product computations. Then, the peaks in the fast Fourier transform of $\langle\sigma^X\rangle + i\langle\sigma^Y\rangle$ are identified as the eigenenergies of the Hamiltonian, and these eigenenergies are sorted into ascending order such that $Y_1 \leq Y_2 \leq \cdots \leq Y_5$.

To generate each multi-target prediction in the calibration dataset, the control model maps a collection of five qubit and four coupler bias features from an instance of the many-body Ramsey spectroscopy technique to the 5×5 single-photon block matrix in the representation of equation (12). Next, a numerical eigensolver produces five eigenenergy approximations, and they are sorted in ascending order such that $X_1 \leq X_2 \leq \cdots \leq X_5$.

Figure 16 depicts qubit and compler bias feature boxplots from top-to-bottom respectively, where Qubit j denotes the qubit bias corresponding to qubit site j and Coupler $j/j+1$ denotes the coupler bias corresponding to the nearest neighbor coupler for qubit sites j and $j+1$. Figure 17 jointly depicts single-target predictions and single-targets.

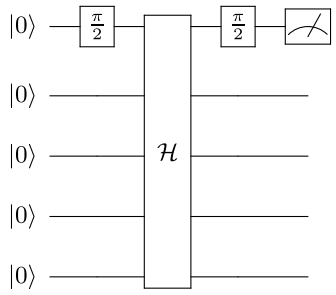

Figure 15. Time-domain spectroscopy circuit from the many-body Ramsey spectroscopy technique. Initially, every qubit is in the fiducial state $|0\rangle$, then a microwave pulse is applied to the kth qubit, which places the kth qubit in a linear combination of the standard basis. Subsequently, the quantum system evolves according to the time-independent Hamiltonian (equation (12)) with randomly programmed parameters. Then, a microwave pulse is applied to measure either $\langle\sigma^X\rangle$ or $\langle\sigma^Y\rangle$, where σ^X and σ^Y denote Pauli operators.

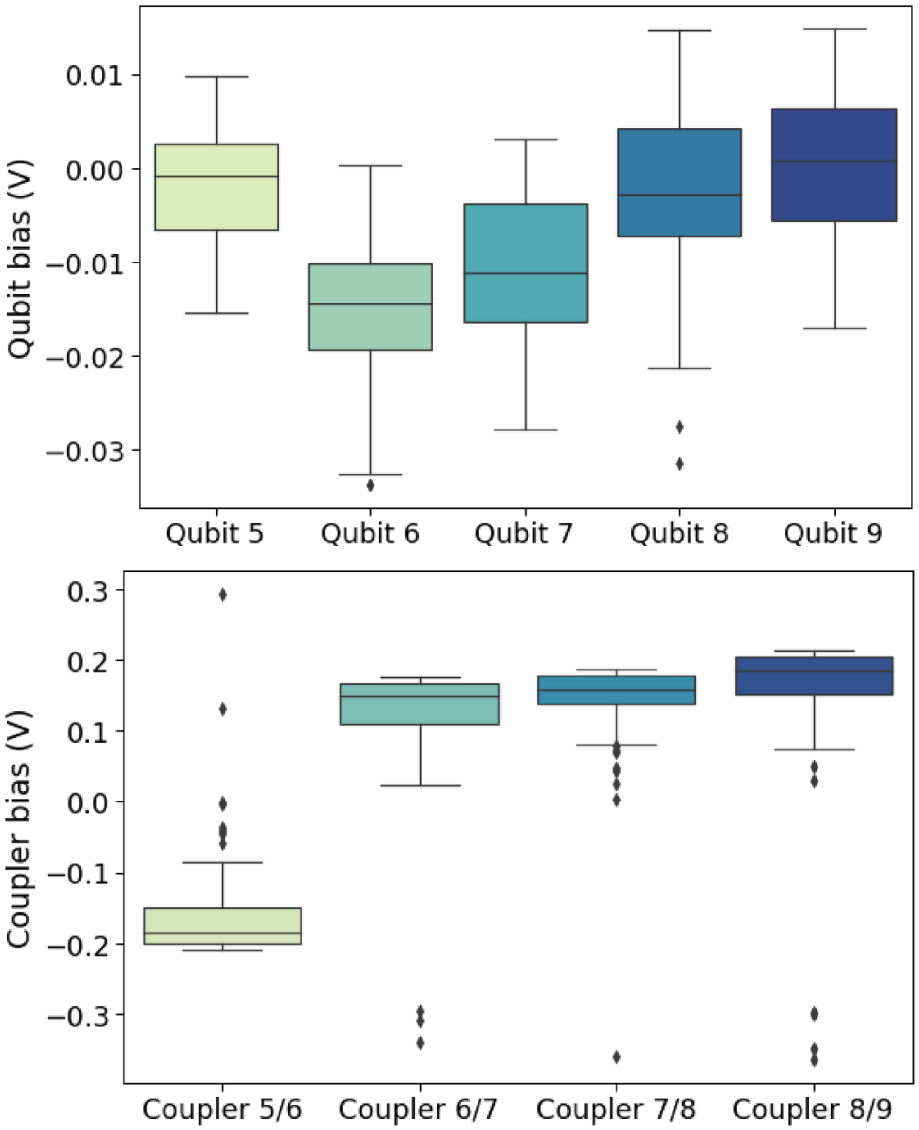

Figure 16. Boxplots of qubit and coupler bias feature with training example indices. Here Qubit j denotes the qubit bias corresponding to qubit site j and Coupler $j/j+1$ denotes the coupler bias corresponding to the nearest neighbor coupler for qubit sites j and $j+1$.

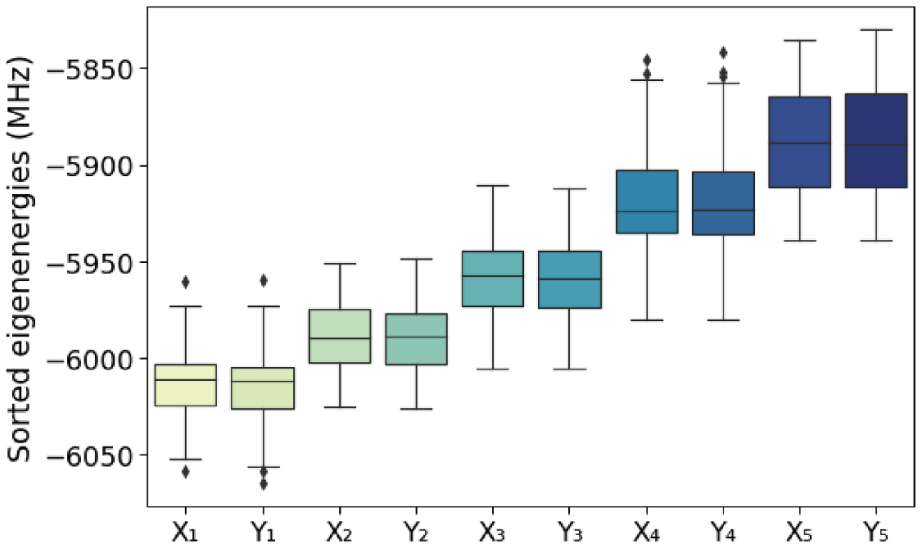

Figure 17. Boxplot of single-target predictions X_j and single-targets Y_j with training example indices.

Appendix B. Augmented learning curves

In the main body, figure 7 depicts the built-in model selection step in Algorithm 2, where the single-target regression subtask is Y_3. In the appendix, figure 18 depicts the built-in model selection step in Algorithm 2, where the single-target regression subtask is Y_1 (top) and Y_2 (bottom). Similarly, figure 19 depicts the built-in model selection step in Algorithm 2, where the single-target regression subtask is Y_4 (top) and Y_5 (bottom). We list the single-target training example sizes in appendix D.

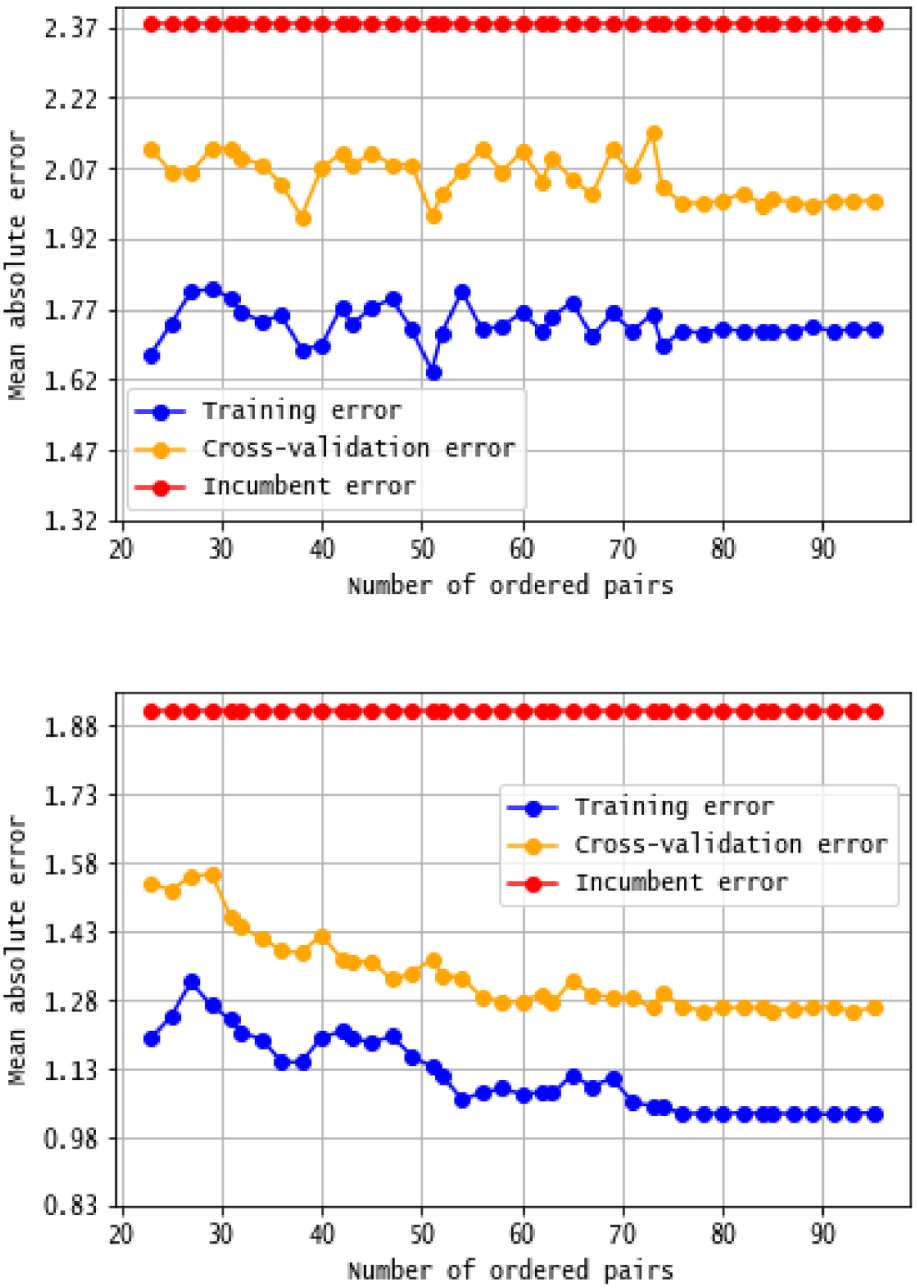

Figure 18. Augmented learning curve for the single-target regression subtasks Y_1 (top) and Y_2 (bottom), where we measure accuracy with absolute error, so the appropriate unit is megahertz (MHz). We depict the built-in model selection step in Algorithm 2, where the single-target training example sizes vary between 23 and 95 ordered pairs, inclusive. Evidently, the built-in model selection step always chose Algorithm 1.

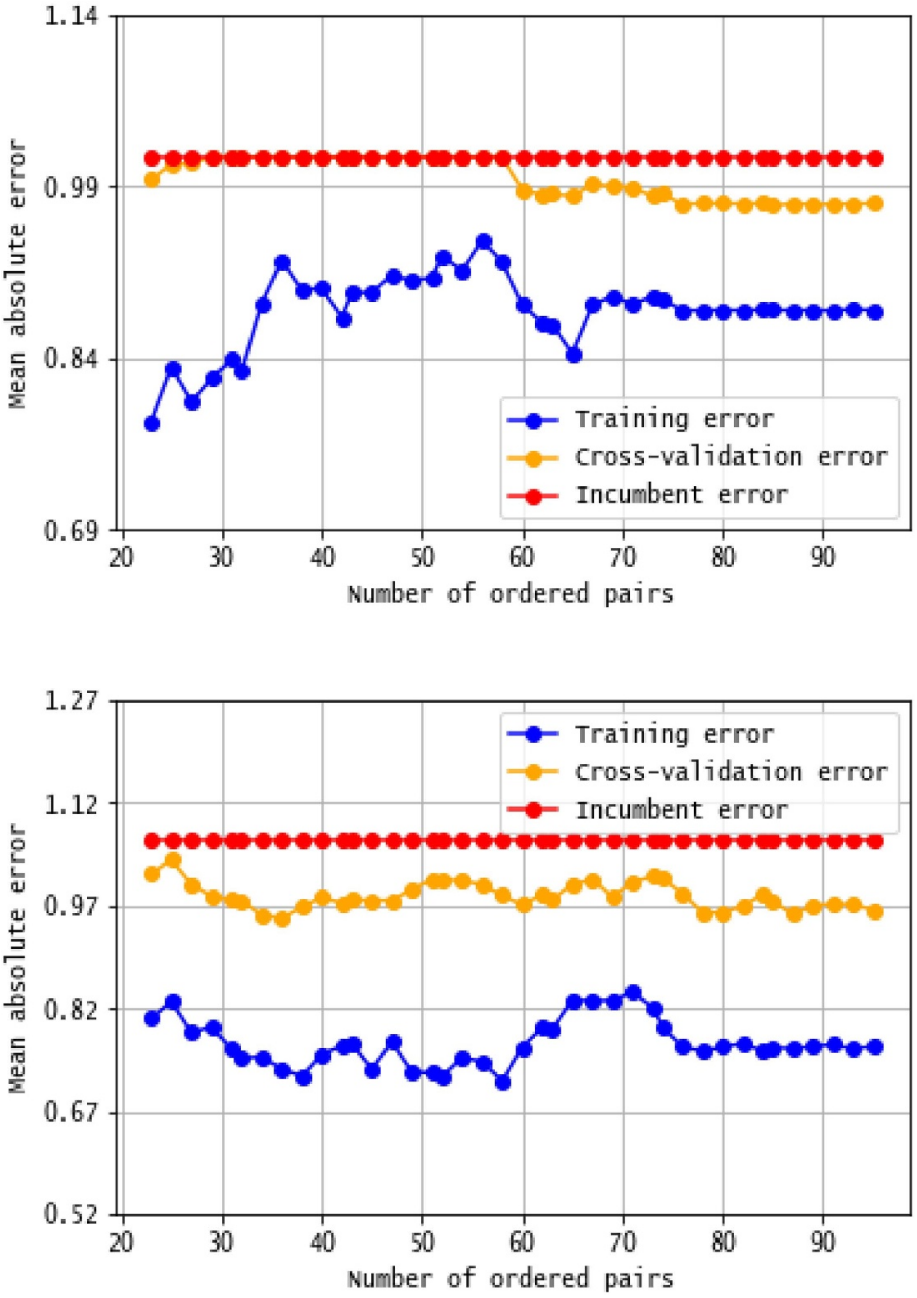

Figure 19. Augmented learning curve for the single-target regression subtasks Y_4 (top) and Y_5 (bottom), where we measure accuracy with absolute error, so the appropriate unit is megahertz (MHz). We depict the built-in model selection step in Algorithm 2, where the single-target training example sizes vary between 23 and 95 ordered pairs, inclusive. Evidently, the built-in model selection step tended to chose the state-of-the-art calibration model [26–28], when there were less than 60 ordered pairs, and it always chose Algorithm 1, when there were 51, or more, ordered pairs in the single-target regression subtask Y_4. In contrast, the built-in model selection step always chose Algorithm 1 in the single-target regression subtask Y_5.

Appendix C. SHAP approach

In the SHAP approach, the objective is to replicate an individual single-target training example prediction from a single-target regressor with an explanation model, where the coefficients measure the feature importance. Štrumbelj and Kononenko showed that these coefficients, known as SHAP values, are equivalent to Shapley values in cooperative game theory [30]. The explanation model is defined as an affine function of binary variables

$$g(z') = \phi_0 + \sum_{k=1}^{M} \phi_k z'_k \tag{C1}$$

where $z' \in \{0,1\}^M$ is a set of binary variables, M is the number of features under consideration, and ϕ_k is a real-valued feature attribution, known as a SHAP value, for the kth feature. As the computation of Shapley values has an exponential time complexity, the SHAP software approximates the coefficients with insights from additive feature attribution methods; see [30].

In the explainable machine learning section, we utilize the model-agnostic approximation method, known as Kernel SHAP, to compute the SHAP values. This method approximates each individual training example prediction from a fitted single-target regressor, where $M = 9$ in equation (C1). The importance I of each feature is defined as the sum of absolute SHAP values

$$I_k = \sum_{i=1}^{m_{\text{train}}} |\phi_k^{(i)}|, \tag{C2}$$

which defines an ordering. Accordingly, in each summary plot, the features are sorted in ascending order from bottom-to-top.

C.1. SHAP summary plots

In the main body, figure 14 depicts a summary plot, where the single-target regression subtask is Y_1. In the appendix, figure 20 depicts a summary plot, where the single-target regression subtask is Y_2 (top) and Y_3 (bottom), and figure 21 depicts a summary plot, where the single-target regression subtask is Y_4 (top) and Y_5 (bottom).

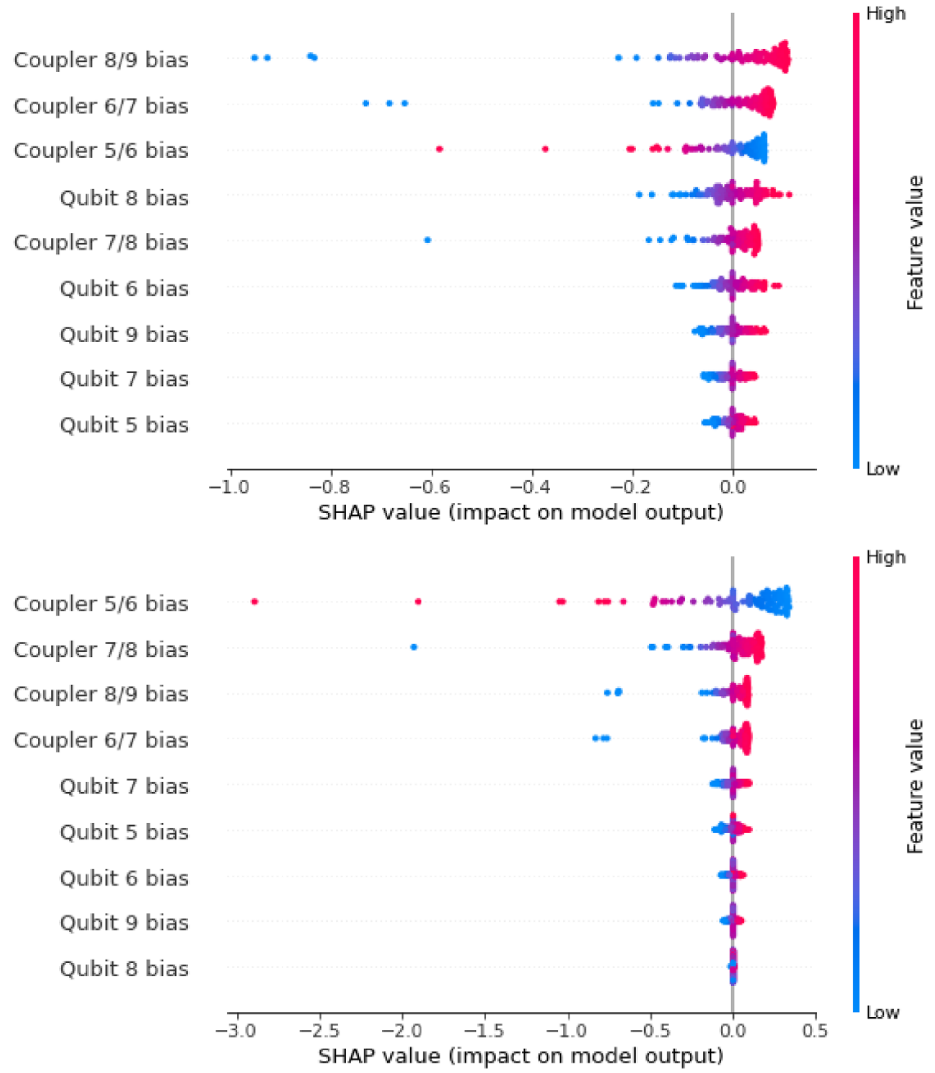

Figure 20. SHAP summary plot for the single-target regression subtasks Y_2 (top) and Y_3 (bottom). The top plot indicates the most important feature is the coupler 8/9 bias, and the least important feature is the qubit 5 bias. The bottom plot indicates the most important feature is the coupler 5/6 bias, and the least important feature is the qubit 8 bias.

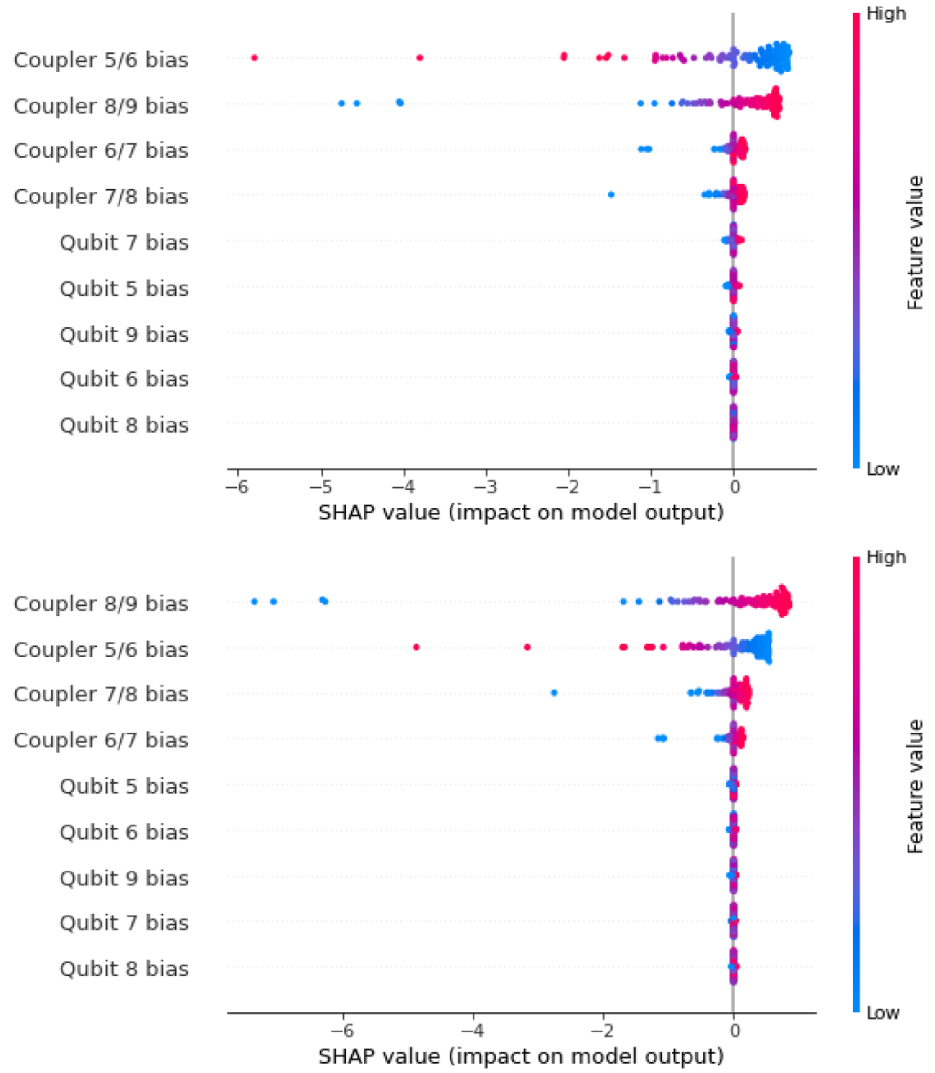

Figure 21. SHAP summary plot for the single-target regression subtasks Y_4 (top) and Y_5 (bottom). The top plot indicates the most important feature is the coupler 5/6 bias, and the least important feature is the qubit 8 bias. The bottom plot indicates the most important feature is the coupler 8/9 bias, and the least important feature is the qubit 8 bias.

Appendix D. Implementation

In our experiment, we use the open-source, scikit-physlearn repository, developed by Alex Wozniakowski [40, 41], as well as the Microsoft LightGBM repository [29], the scikit-learn repository [34], and the SHAP repository [30]. Notably, the Python Package Index, known as PyPI, hosts the scikit-physlearn package:

```
pip install scikit-physlearn
```

and the scikit-physlearn documentation contains an installation guide to build from source [41]. To run the experiment visit the paper_results child directory of the examples directory in the scikit-physlearn repository [40].

Notably, in Algorithm 2, which always called Algorithm 1, the basis function is sklearn.ensemble.StackingRegressor, where the first layer corresponds to sklearn.neural_network.MLPRegressor and lightgbm.LGBMRegressor and the second layer corresponds to sklearn.neural_network.MLPRegressor. Also, in the implementation of Algorithm 1, we appended a lasso term to the Lagrangian form of the optimization problem

$$\operatorname*{argmin}_{\alpha} \sum_{i=1}^{m_{\text{train}}} \ell_j(Y_j^{(i)}, f_{j,k-1}(X^{(i)}) + \alpha b(X^{(i)}; \theta_{j,k})) + \lambda|\alpha|.$$

To access the other hyperparameters, visit the main_body.py file in the paper_results directory.

The off-the-shelf gradient boosting algorithm corresponds to lightgbm.LGBMRegressor, where the other hyperparameters choices are shown in the boost_wout_prior_knowledge.py file in the paper_results directory. The embedded learned algorithm in the entirely data-driven reimplementation of the calibration model corresponds to sklearn.neural_network.MLPRegressor, where the other hyperparameters choices are shown in the supplementary.py file in the paper_results directory. The training example sizes in figures 7, 18, and 19 correspond to

$$23, 25, 27, 29, 31, 32, 34, 36, 38, 40, 42, 43, 45, 47, 49, 51, 52, 54, 56, 58,$$
$$60, 62, 63, 65, 67, 69, 71, 73, 74, 76, 78, 80, 82, 84, 85, 87, 89, 91, 93, 95.$$

ORCID iDs

Alex Wozniakowski https://orcid.org/0000-0003-2867-1175
Jayne Thompson https://orcid.org/0000-0002-3746-244X
Mile Gu https://orcid.org/0000-0002-5459-4313
Felix C Binder https://orcid.org/0000-0003-4483-5643

References

[1] Tukey J 1977 *Exploratory Data Analysis* (Reading, MA: Addison-Wesley)
[2] Kearns M and Valiant L 1988 Learning boolean formulae or finite automata is as hard as factoring *Technical Report* TR-14-88 Harvard University Aiken Computation Laboratory
[3] Mallat S and Zhang Z 1993 Matching pursuits with time-frequency dictionaries *IEEE Trans. Signal Process.* **41** 3397–3415
[4] Freund Y and Schapire R 1997 A decision-theoretic generalization of on-line learning and an application to boosting *J. Comput. Syst. Sci.* **55** 119–39
[5] Breiman L 1997 Arcing the edge *Technical Report* Stanford University, Department of Statistics
[6] Friedman J, Hastie T and Tibshirani R 2000 Additive logistic regression: a statistical view of boosting *Ann. Stat.* **28** 337–407
[7] Friedman J 2001 Greedy function approximation: a gradient boosting machine *Ann. Stat.* **29** 1189–232
[8] Schapire R *et al* 2002 Incorporating prior knowledge into boosting *ICML'02: Proc. of the Nineteenth Int. Conf. on Machine Learning* pp 538–45
[9] Bühlmann P and Hothorn T 2007 Boosting algorithms: regularization, prediction and model fitting *Stat. Sci.* **22** 477–505
[10] Bühlmann P and Bin Y 2011 Boosting with the l_2 loss *J. Am. Stat. Assoc.* **98** 324–39
[11] Hastie T, Tibshirani R and Friedman J 2009 *The Elements of Statistical Learning* (Berlin: Springer)
[12] James W and Stein C 1961 Estimation with quadratic loss *Symp. on Mathematical Statistics and Probability* vol 4.1 pp 361–79
[13] Wolpert D 1992 Stacked generalization *Neural Netw.* **5** 241–59
[14] Sjöberg J *et al* 1995 Nonlinear black-box modeling in system identification: a unified overview *Automatica* **31** 1691–724
[15] Breiman L 1996 Stacked regressions *Mach. Learn.* **24** 49–64
[16] Breiman L and Friedman J 1997 Predicting multivariate responses in multiple linear regression *R. Stat. Soc. Ser.* B **59** 3–54
[17] Caruana R 1997 Multitask learning *Mach. Learn.* **28** 41–75
[18] Erhan D *et al* 2010 Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.* **11** 625–60
[19] Djorgovski S G *et al* 2011 Towards an automated classification of transient events in synoptic sky surveys *Conf. on Intelligent Data Understanding*
[20] Mahabal A *et al* 2011 Real time classification of transient events in synoptic sky surveys *Proc. Int. Astron. Union* **7** 355–7
[21] Mahabal A *et al* 2011 Discovery, classification and scientific exploration of transient events from the Catalina Real-time Transient Survey *Bull. Astron. Soc. India* **39** 387–408
[22] Borchani H *et al* 2015 A survey on multi-output regression *Data Mining and Knowledge Discovery* **5** 216–33
[23] Spyromitros-Xioufis E *et al* 2016 Multi-target regression via input space expansion: treating targets as inputs *Mach. Learn.* **104** 55–98
[24] Waegeman W, Dembczyński K and Eyke H 2019 Multi-target prediction: a unifying view on problems and methods *Data Min. Knowl. Discovery* **33** 293–324
[25] Chen Y *et al* 2020 Non-unique games over compact groups and orientation estimation in cryo-EM *Inverse Problems* **36** 1–39
[26] Roushan P *et al* 2017 Spectroscopic signatures of localization with interacting photons in superconducting qubits *Science* **358** 1175–9
[27] Neill C *et al* 2018 A blueprint for demonstrating quantum supremacy with superconducting qubits *Science* **360** 195–9
[28] Chiaro B *et al* 2019 Growth and preservation of entanglement in a many-body localized system (arXiv:1910.06024)
[29] Guolin K *et al* 2017 LightGBM: a highly efficient gradient boosting decision tree *Advances in Neural Information Processing Systems 30* (Red Hood, NY: Curran Associates, Inc.) pp 3149–57 (available at: http://toc.proceedings.com/39083webtoc.pdf)
[30] Lundberg S and Lee S-I 2017 A unified approach to interpreting model predictions *Advances in Neural Information Processing Systems 30* (Red Hood, NY: Curran Associates, Inc.) pp 4765–74 (available at: http://toc.proceedings.com/39083webtoc.pdf)
[31] Zheng Z *et al* 2007 A general boosting method and its application to learning ranking functions for web search *NIPS: Proc. 20th Int. Conf. on Neural Information Processing* pp 1697–704
[32] Grubb A and Bagnell D 2011 Generalized boosting algorithms for convex optimization *ICML'11: Proc. Twenty-Eighth Int. Conf. on Machine Learning* pp 1209–16
[33] Ramakrishnan R *et al* 2015 Big data meets quantum chemistry approximations: the Δ-machine learning approach *J. Chem. Theory Computat.* **11** 2087–96
[34] Pedregosa F *et al* 2011 Scikit-learn: machine learning in python *J. Mach. Learn. Res.* **12** 2825–30
[35] Mason L *et al* 1999 Boosting algorithms as gradient descent *NIPS: Proc. 12th Int. Conf. on Neural Information Processing* pp 512–18
[36] Vincent P and Bengio Y 2002 Kernel matching pursuit *Mach. Learn.* **48** 165–87

[37] Milanfar P 2013 A tour of modern image filtering: new insights and methods, both practical and theoretical *IEEE Signal Process. Mag.* **30** 106–28
[38] Romano Y and Elad M 2015 Boosting of image denoising algorithms *SIAM J. Imaging Sci.* **8** 1187–1219
[39] Sigrist F 2018 Gradient and newton boosting for classification and regression (arXiv:1808.03064)
[40] Wozniakowski A 2020 Scikit-physlearn (available at: github.com/a-wozniakowski/scikit-physlearn)
[41] Wozniakowski A 2020 Scikit-physlearn manual (available at: scikit-physlearn.readthedocs.io)