Chemical machine learning with kernels: The impact of loss functions

Quang Van Nguyen,[1] Sandip De,[2,3] Junhong Lin,[1] and Volkan Cevher[1, a)]
[1)] *Laboratory for Information and Inference Systems (LIONS)*
École Polytechnique Fédérale de Lausanne, Switzerland
[2)] *Laboratory of Computational Science and Modelling (COSMO), Institute of Materials,*
École Polytechnique Fédérale de Lausanne, Switzerland
[3)] *National Center for Computational Design and Discovery of Novel Materials (MARVEL)*

Machine learning promises to accelerate materials discovery by allowing computational efficient property predictions from a small number of reference calculations. As a result, the literature has spent a considerable effort in designing representations that capture basic physical properties so far. In stark contrast, our work focuses on the less-studied learning formulations in this context in order to exploit inner structures in the prediction errors. In particular, we propose to directly optimize basic loss functions of the prediction error metrics typically used in the literature, such as the mean absolute error or the worst case error. In some instances, a proper choice of the loss function can directly reduce reasonably the prediction performance in the desired metric, albeit at the cost of additional computations during training. To support this claim, we describe the statistical learning theoretic foundations, and provide supporting numerical evidence with the prediction of atomization energies for a database of small organic molecules.

I. INTRODUCTION

Estimating the stability of molecules and materials is one of the most fundamental topics in computational quantum chemistry. Traditional approach is to use the density functional theory (DFT)[1,2] to solve Schrödinger's equation with some approximations to make the calculation computationally feasible. However, the DFT is still too expensive to employ in high-throughput screening of realistic materials in an optimal way.

Recently, there is a great deal of interest in the materials design using machine learning at quantum chemistry level with existing DFT data.[3–15] This research approach has been supported with strong preliminary evidence that we can simulate relatively large systems with thousands of atoms with accurate prediction performance.

As a result, a considerable effort has gone into building machine learning models for purpose of representing atomic data. To our knowledge, the existing literature mainly focus on the design of kernels along with the so-called "descriptors" or "fingerprints", e.g., bond lengths, bond angles, etc, to tailor machine learning procedures to capture subtle differences in atomic environments.

In addition, several chemical environment representations have been proposed in order to improve prediction accuracy. Some of the notable recent development include, Coulomb matrices[4,5], Bag of Bonds[11], representations based on Fourier series of atomic radial distribution functions[10], forces on atom[12], interatomic many body expansions[13] and alchemical and structural distribution[14], constant size descriptors[15] and references therein. The resulting machine learning frameworks often use a kernel ridge regression or neural networks with impressive prediction performance.

In this paper, we emphasize the learning formulations, i.e., the loss functions, which have received very little attention in the same context. To go beyond the root mean squared error (RMSE) metric, we provide learning theoretic arguments to motivate loss functions to improve predictions in the mean absolute error (MAE) and max absolute error (MaxAE) metrics.

The metric MAE had been cited in the very early forecasting literature as a primary measure of performance for forecasting models[16] and has recently come to our attention due to its robustness. MaxAE, on the other hand, is an upper bound for both RMSE and MAE and reflects the prediction with the highest inaccuracy.

The paper is organized as follows. Section II discusses the statistical learning perspective of the ground state energy regression problem, including regularized M-estimators, cross-validation method, and kernel trick. Section III discusses the mathematical details of basic convex optimization and numerical methods to approximate a solution of our proposed novel models used in predicting ground state energy. Finally, in section IV, we provide concrete numerical evidence with an previously published kernel to predict atomization energies involving a database of small organic molecules and improve the usual kernel ridge regression (KRR) at the expense of more computation.

Notation. The n-dimensional Euclidean space is denoted by $\mathbb{R}^n$. The transpose and the inverse of a positive definite matrix $\mathbf{K}$ are denoted by $\mathbf{K}^\top$ and $\mathbf{K}^{-1}$, respectively. Given a vector $\mathbf{x} \in \mathbb{R}^n$, we define the ℓ_1-norm as $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$; the ℓ_2-norm as $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$; and the ℓ_∞-norm as $\|\mathbf{x}\|_\infty = \max_{1 \leqslant i \leqslant n} |x_i|$. Finally, $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ denote respectively generic inner product and norm in a Hilbert space.

In the sequel, we represent the state of a molecule by a sequence $\{(r_k, z_k)\}_{k=1}^K$, where $r_k \in \mathbb{R}^3$ is the position of k-th atom and z_k is its atomic number. This physical state is translated into a vector-like representation $\mathbf{x} \in \mathbb{R}^n$, which is usually required to be invariant with respect to permutational, rotational and translational symmetries.[17]

[a)] Electronic mail: volkan.cevher@epfl.ch

II. LEARNING THEORY BASICS FOR REGRESSION

This section provides a learning theoretic background in support of the following basic claim.

Given an atomic representation, different learning formulations introduce different structures in the materials predictions. By choosing an appropriate learning formulation, we can optimize the relevant prediction metric.

A. Regression for atomization energies

We consider the following learning setting. Suppose that we observe a set of sample pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of different molecule representations $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ with the corresponding atomization energy $y_i \in \mathcal{Y} \subseteq \mathbb{R}$. Based on these data, we wish to estimate a function $y = f(\mathbf{x})$ to predict the atomization energy of new molecules. In what follows, we show that such an important feat is possible, given sufficient amount of train data.

The quality of a predictor is often evaluated in terms of test errors computed over test data $\{(\bar{\mathbf{x}}_j, \bar{y}_j)\}_{j=1}^N$. In practice, three following test errors will be typically involved: MAE, MaxAE, and RMSE. They are defined in the following fashion

$$\mathrm{MAE} = \frac{1}{l}\sum_{j=1}^{l} |f(\bar{\mathbf{x}}_j) - \hat{y}_j| = \frac{\|f(\bar{\mathbf{x}}) - \bar{\mathbf{y}}\|_1}{l},$$

$$\mathrm{MaxAE} = \max_{j\in\{1,\dots,l\}} |f(\bar{\mathbf{x}}_j) - \hat{y}_j| = \|f(\bar{\mathbf{x}}) - \bar{\mathbf{y}}\|_\infty,$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{l}\sum_{j=1}^{l} |f(\bar{\mathbf{x}}_j) - \bar{y}_j|^2} = \sqrt{\frac{\|f(\bar{\mathbf{x}}) - \bar{\mathbf{y}}\|_2^2}{l}}.$$

Here $f(\bar{\mathbf{x}}) = [f(\bar{\mathbf{x}}_1), \dots, f(\bar{\mathbf{x}}_l)]^\top$ and $\bar{\mathbf{y}} = [\bar{y}_1, \dots, \bar{y}_l]^\top$.

The above metrics and their corresponding utilities are intuitive to informed readers. For instance, RMSE metric looks at the average prediction error in the Euclidean distance, whereas MaxAE only takes care of the worst case error. The metric MAE takes, on the other hand, the spectrum and decreases the impact of outlier errors in the average as compared to RMSE. Hence, different applications may focus on any of these prediction metrics.

The regression framework discussed above can be categorized as a supervised learning problem. As a consequence, it has strong support from statistical learning theory[18–20], which will be introduced in what follows.

B. Supervised learning

In statistical learning theory, one typically assumes that all the elements from test data and train data are independently and identically drawn according to an unknown probability distribution (but one should keep in mind that the i.i.d. assumption can be further relaxed). The performance of the learning function is measured in terms of the expected loss/risk with respect to a loss function $\ell : \mathbb{R}\times\mathbb{R} \to \mathbb{R}_+$, see[19,20] for its precise definition. In the sequel, for ease of presentation, we simply identify the expected risk as the test error over the following test data $\{(\bar{\mathbf{x}}_i, \bar{y}_i)\}_{i=1}^l$, defined as

$$\mathcal{R}(f) = \frac{1}{l}\sum_{i=1}^{l} \ell(f(\bar{\mathbf{x}}_i), \bar{y}_i). \tag{1}$$

Such an argument will not cause any trouble when the amount of test data is sufficiently large. It is easy to see that MAE or RMSE is equivalent to $\mathcal{R}(f)$ with a suitable loss function. Note that MaxAE cannot be directly linked to (1), but in the later case, one can consider generalizing the definition of risk.

In this setting, one natural benchmark is the function $f^\star$ that minimizes the risk, over all possible (i.e., measurable) functions. Often times, however, we have to restrict our search to some hypothesis space $\mathcal{F}$ of functions from $\mathbb{R}^d$ to $\mathbb{R}$ to exploit additional structures, such as smoothness, in the problem or to save on computation associated with the training procedure. The canonical example is the kernel-based linear prediction in the space of functions represented as $f_\omega(\mathbf{x}) = \sum_{j=1}^p \omega_j \phi_j(\mathbf{x})$. Such an approach is also supported by many examples of consistent hypothesis spaces, i.e., $\inf_{f\in\mathcal{F}} \mathcal{R}(f) = \mathcal{R}(f^\star)$.[20] We will talk about how to choose a suitable hypothesis space in later subsections.

C. Regularized M-estimators

With the given hypothesis space, a natural idea for finding a good predictor is to solve the expected risk minimization, $\inf_{f\in\mathcal{F}} \mathcal{R}(f)$. However, as the expected risk cannot be known exactly, the expected risk minimization is replaced with the empirical risk minimization.

Directly minimizing the empirical loss can lead to an effect called overfitting, wherein we fit the training data extremely well (i.e., with low error), yet we obtain a model that produces very poor predictions on future test data whenever the test inputs differ from the training inputs. There exists an important solution to the overfitting phenomenon, the regularized M-estimators[21] (also referred to as the regularized empirical risk minimizations[18]), i.e.,

$$\hat{f}_\lambda \in \operatorname*{argmin}_{f\in\mathcal{F}} \{\mathcal{R}_n^\lambda(f) := \mathcal{R}_n(f) + \lambda\Omega(f)\}. \tag{2}$$

Here, $\lambda \in \mathbb{R}_+$ is a regularization parameter, Ω is a regularizer and the empirical risk $\mathcal{R}_n(f)$ is defined as

$$\mathcal{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell\big(f(\mathbf{x}_i), y_i\big).$$

The regularizer Ω imposes certain properties on the underlying function. A very common property is the smoothness, which is especially required for performing atomization energies regression. Let us consider the case

where f is represented via radial basis functions (RBFs) $\{\phi_k\}_{k\in\mathbb{N}}$, i.e.,

$$f(\mathbf{x}) = \sum_{k\in\mathbb{N}} \omega_k \phi_k(\mathbf{x}).$$

As radial basis functions are themselves smooth, imposing smoothness on f can be done by making the magnitude of weights $\|\omega\|^2 = \sum_{k\in\mathbb{N}} \omega_k^2$ decayed. Weight decay implicitly leads to smoothness with RBF basis functions because rapid changes in the slope of f (i.e., high curvature) can only be created in RBFs by adding and subtracting basis functions with large weight. If we consider the Hilbert space governed by RBFs, the magnitude of weights defines an Hilbertian norm in this space. This type of regularizer is referred to as the ridge regularizer.

In this paper, we mainly focus on ridge regularizer, but one should keep in mind that our approach is still applied to general regularizers. In order to control the complexity of the solution and to ensure generalizing well, the regularization parameter λ needs to be tuned in practice. We will discuss this after presenting statistical results for the regularized M-estimators.

D. Statistical results

A key tool for analyzing statistical results for the regularized M-estimators is the error decomposition. To introduce the error decomposition, we introduce an auxiliary function f_λ, defined as the solution of the regularized expected risk minimization,

$$f_\lambda \in \operatorname*{argmin}_{f\in\mathcal{F}} \{\mathcal{R}^\lambda(f) := \mathcal{R}(f) + \lambda\Omega(f)\}.$$

A simple calculation shows that the excess risk of the estimator $\hat{f}_\lambda$ can be decomposed as (e.g.[22])

$$\mathcal{R}(\hat{f}_\lambda) - \mathcal{R}(f^\star) \leqslant \mathcal{E}_{\mathbf{sam}} + \mathcal{E}_{\mathbf{app}}, \tag{3}$$

where

$$\mathcal{E}_{\mathbf{sam}} = \mathcal{R}(\hat{f}_\lambda) - \mathcal{R}_n(\hat{f}_\lambda) + \mathcal{R}_n(f_\lambda) - \mathcal{R}(f_\lambda),$$

$$\mathcal{E}_{\mathbf{app}} = \mathcal{R}^\lambda(f_\lambda) - \mathcal{R}(f^\star).$$

We provide the proof in the appendix.

The first term $\mathcal{E}_{\mathbf{sam}}$ in the above error decomposition is a random variable depending on the train set, the function class $\mathcal{F}$, and the regularized parameter λ. It is called the sample error. It measures the effect of minimizing the regularized empirical risk instead of the regularized expected risk. Typically, it can be controlled by a term which is decreasing with respect to both the train size and the regularization parameter λ.

The second term $\mathcal{E}_{\mathbf{app}}$ in the error decomposition above is deterministic. It only depends on the function class $\mathcal{F}$ and the regularization parameter λ. It measures how well the solution of the regularized expected risk minimization can be used to approximate $f^\star$. Typically, it is increasing with respect to the regularization parameter λ.

The regularization parameter λ hence controls an important trade-off in prediction performance (i.e., generalization), which has been extensively discussed in the literature[18,23]. An optimal trade-off based on the best choice of λ results in an excess risk that scales between the inverse and the inverse square root of the number of training data[21,24,25]. The following two examples provide statistical results for the estimators given by (2) with the square-norm penalty, considering two different settings.

Example II.1 Consider the setting of non-parametric regression with the square loss over a reproducing kernel Hilbert space (RKHS) $\mathcal{F}$ as those in[26,27]. It is known[28] that KRR, i.e., (2) with the square-norm penalty, has the following upper bound for the excess risk:

$$\mathbb{E}[\mathcal{R}(\hat{f}_\lambda) - \mathcal{R}(f^\star)] \lesssim \frac{c_1}{n\lambda^\gamma} + c_2\lambda^{2\zeta}.$$

Here, $\gamma \in [0,1]$ is related to the capacity condition of $\mathcal{F}$ and $\zeta \in [1/2, 1]$ is related to the regularity of the target function $f^\star$. The optimal error bound $\mathcal{O}(n^{-\frac{2\zeta}{2\zeta+\gamma}})$ is achieved when $\lambda_* \simeq n^{-\frac{1}{2\zeta+\gamma}}$. The best choice of λ depends on unknown distribution parameters ζ and γ. We thus, in practice, choose the regularized parameter λ_* by using the cross-validation methods.

Example II.2 Consider the setting of non-parametric classification over an RKHS $\mathcal{F}$ with the Hinge loss as in[24,25], or more generaly, a loss function with bounded gradients. Via Rademacher complexity[29], one can prove that the solution of (2) with the square-norm penalty has the following upper bound for the excess risk:

$$\mathbb{E}[\mathcal{R}(\hat{f}_\lambda) - \mathcal{R}(f^\star)] \lesssim \frac{c_1}{\sqrt{n\lambda}} + c_2\lambda^{\beta}.$$

Here, we assume that the approximation error satisfies $\mathcal{E}_{\mathbf{app}} \lesssim \lambda^\beta$, for some $\beta \in (0,1]$. The best attainable error bound from the above estimates is of order $\mathcal{O}(n^{-\frac{\beta}{2\beta+1}})$, and it is achieved when $\lambda_* \simeq n^{-\frac{1}{2\beta+1}}$. Using a more involved technique, it has been shown[25] that the error bound can be further improved to $\mathcal{O}(n^{-\alpha})$, where $\alpha \in (0,1]$ is a parameter depending on the data distribution and the hypothesis space $\mathcal{F}$.

E. Cross-validation methods

Unfortunately, the best choice of the regularizer parameter λ depends on the data distribution, and in practice, people usually use cross-validation (CV)[30] to determine it. In CV, train data are divided into K roughly equal parts (or folds). For each $k \in \{1,\ldots,K\}$, k-fold will be used as validation set. Fit the model with a candidate parameter λ using the remaining $K-1$ folds by a specific algorithm to approximate a solution to (2), to obtain the corresponding predictor $\hat{f}_\lambda$. This partial

predictor is used to compute the validation error $E_k(\lambda)$ evaluating on validation set. After K rounds, the cross-validation error is obtained by averaging these K validation errors, i.e.,

$$\mathrm{CV}(\lambda) = \frac{1}{K}\sum_{k=1}^{K} E_k(\lambda),$$

and we choose λ_* to minimize this cross-validation error.

F. Kernel trick

In this subsection, we discuss kernel method, a common approach for atomization energies regression. It is based on choosing the hypothesis space $\mathcal{F}$ as an RKHS generated by a kernel. The essential of the kernel trick is to map the original representation $\mathbf{x} \in \mathbb{R}^d$ to a representation in a Hilbert space $\mathcal{F}$, called the feature space, by a feature map $\psi\colon \mathbf{x} \mapsto \psi(\mathbf{x})$ in such a way that $\langle f, \psi(\mathbf{x})\rangle = f(\mathbf{x})$ for all function $f \in \mathcal{F}$.

In other words, we use a feature map to map the data from the low-dimensional space into a higher (possibly infinite) dimension space such that in this space, the predictor f can be determined by a linear expression. We then define the kernel function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}')\rangle$. One can think of $\mathcal{K}$ as specifying similarity between instances and of the feature map ψ as mapping the domain set $\mathcal{X}$ into a space where these similarities are realized as inner products. The main advantage of such a trick is that it implements linear separators in high dimensional feature space without having to specify points in that space or expressing the feature map ψ explicitly.

We now return to the problem (2) and restrict it to $\mathcal{F}$ to obtain

$$\hat{f}_\lambda \in \operatorname*{argmin}_{f\in\mathcal{F}} \ \frac{1}{n}\sum_{i=1}^{n} \ell\big(\langle f, \psi(\mathbf{x}_i)\rangle, y_i\big) + \lambda\Omega(f). \tag{4}$$

Although $\mathcal{F}$ can be infinitely dimensional, solving (4) can be translated into solving an optimization problem in finite-dimensional setting due to the following famous representer theorem.

Theorem II.3 (Representer theorem[31]) *Suppose that $\mathcal{F}$ is the feature space corresponding to the feature map ψ defined on $\mathcal{X}$. Then the problem (4) with ridge regularizer possesses a solution of the following form*

$$\hat{f}_\lambda(\mathbf{x}) = \sum_{i=1}^{n} c_i^\natural \mathcal{K}(\mathbf{x}, \mathbf{x}_i),$$

where $\mathbf{c}^\natural = [c_1^\natural, \ldots, c_n^\natural]^\top \in \mathbb{R}^n$.

Let us denote the training matrix by $\mathbf{K} = \big[\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\big]_{1\leqslant i,j\leqslant n}$ and the i-th row of $\mathbf{K}$ by $\mathbf{K}_i$. From the representer theorem and a simple calculation as shown in the appendix, we show that solving the problem (4) with ridge regularizer is equivalent to solving the following optimization in finite-dimensional Euclidean space

$$\min_{\mathbf{c}\in\mathbb{R}^n} \ \frac{1}{n}\sum_{i=1}^{n} \ell\Big(\mathbf{c}^\top \mathbf{K}_i, y_i\Big) + \frac{\lambda}{2}\mathbf{c}^\top \mathbf{K}\mathbf{c}. \tag{5}$$

Example II.4 (Two common kernels) Gaussian kernel: $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|_2^2/2\sigma^2)$. Laplacian kernel: $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|_1/\sigma)$. Intuitively, the Gaussian or Laplacian kernel sets the inner product in the feature space between $\mathbf{x}$ and $\mathbf{x}'$ to be zero if the instances are far away from each other (in the original domain) and close to 1 if they are close. The parameter σ and the corresponding norms determine what we mean by "close".

G. The SOAP-Average kernel

Different kernels such as Gaussian kernel or Laplacian kernel are widely used in materials science community and have led to reasonable predictors. However, it is crucial to keep in mind that the way that molecules are represented as well as the way that the similarity between atomic configurations is measured do influence the quality of the predictor in kernel regressions.

Smooth Overlap of Atomic Positions (SOAP)[17] based kernels[32,33] have been reported among the best performing kernels for predicting electronic structure properties of materials and molecules. As a result, we use the SOAP-average kernel to measure the structural similarity between the molecules by combining the similarity measures of local environments here.

Within the SOAP formalism, the local environment of the i-th atom within a molecule A, i.e., the abstract descriptor of the arrangement of atoms in its vicinity, will be denoted by $\mathcal{X}_i^A$. The set of all atoms of species α of molecule A is denoted by A^α. The local density of i-th atom of species α is then constructed as the superposition of Gaussian functions of variance σ^2 centered on this atom. A cutoff distance of r_c is imposed via a smooth function to set the size of the local environment.

$$\rho^\alpha_{\mathcal{X}_i^A}(\mathbf{r}) = \sum_{j\in A^\alpha} \exp\left(-\frac{(\mathbf{r} - \mathbf{r}_{ij})^2}{2\sigma^2}\right) f_{r_c}(|\mathbf{r}_{ij}|), \tag{6}$$

where $\mathbf{r}_{ij}$ is the Euclidean distance between atom i-th and atom j-th. The SOAP kernel is then defined as the overlap of two local atomic neighbor densities, integrated over the set $\mathbf{SO}(3)$ of all three dimensional rotations, as follows:

$$\tilde{k}\big(\mathcal{X}_i^A, \mathcal{X}_j^B\big) = \int_{\mathbf{SO}(3)} \left| \sum_\alpha \int_{\mathbb{R}^3} \rho^\alpha_{\mathcal{X}_i^A}(\mathbf{r}) \rho^\alpha_{\mathcal{X}_j^B}(\hat{R}\mathbf{r})\, \mathrm{d}\mathbf{r} \right|^2 \mathrm{d}\hat{R}. \tag{7}$$

In practice this kernel can be computed efficiently by first expressing the density on spherical harmonics basis

[7]. The similarity measure $C_{ij}(A,B)$ between the local environments $\mathcal{X}_i^A$ and $\mathcal{X}_j^B$ of the molecules A and B is then determined by the SOAP average kernel function as

$$C_{ij}(A,B) = \frac{\tilde{k}(\mathcal{X}_i^A, \mathcal{X}_j^B)}{\sqrt{\tilde{k}(\mathcal{X}_i^A, \mathcal{X}_i^A)\tilde{k}(\mathcal{X}_j^B, \mathcal{X}_j^B)}}. \tag{8}$$

In order to extract a single similarity measure from the matrix of pairwise environment similarities $\mathbf{C}(A,B)$, the SOAP-average[32] kernel combines the similarity information from the local kernels into a global similarity measure, by taking average of all environment pair similarity values after raising them to the power ζ. As the normalized environment similarity values ranges between 0 and 1, 1 being identical: for value of $\zeta > 1$, higher similarity values naturally get higher weight in the averaged value.

$$\mathcal{K}_\zeta(A,B) = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij}(A,B)^\zeta \tag{9}$$

where, N and M is the number of atoms in molecule A and B respectively. This kernel can be applied to both molecules and crystals while combining a detailed and systematic description of atomic structures with a large degree of adaptability through its hyper-parameters.

H. Multiple kernel learning

The choice of the kernel is critical to the prediction's accuracy of any kernel regression model but in standard frameworks it is left to the user. While different kernels will lead to predictors with different qualities, all of them can be weighted to obtain a unique kernel to be used as the input for classical kernel-based learning algorithms to get a much better predictor. In this framework of multiple kernel learning problem, people find an optimal convex combination

$$\mathbf{K} = \sum_{j=1}^{m} \beta_j \mathbf{K}_j,\ \beta_j \geqslant 0,\ \sum_{j=1}^{m}\beta_j = 1,$$

of m given kernels $\mathbf{K}_1, \ldots, \mathbf{K}_m$. Efficient and scalable methods to find such optimal weights were studied intensively in pieces of literature such as[34–36] and references therein.

I. Error decomposition

We demonstrate in previous subsections that a solution for the minimization problem (5) with an appropriate regularization parameter λ and a suitable kernel has a good generalization performance. In general, Problem (5) is solved via an optimization procedure. Let $\hat{f}_{\lambda,\epsilon}$ be an ϵ-approximated solution of (5). A similar argument as that for (3), one can show that the statistical generalization error of $\hat{f}_{\lambda,\epsilon}$ can be estimated as

$$\mathcal{R}(\hat{f}_{\lambda,\epsilon}) - \mathcal{R}(f^\star) \leqslant \epsilon + \mathcal{E}_{\mathbf{sam}} + \mathcal{E}_{\mathbf{app}},$$

where

$$\mathcal{E}_{\mathbf{sam}} = \mathcal{R}(\hat{f}_{\lambda,\epsilon}) - \mathcal{R}_n(\hat{f}_\lambda) + \mathcal{R}_n(f_{\lambda,\epsilon}) - \mathcal{R}(f_\lambda),$$

$$\mathcal{E}_{\mathbf{app}} = \mathcal{R}^\lambda(f_\lambda) - \mathcal{R}(f^\star).$$

The term ϵ is called optimization error, while the other two terms are referred to as sample error and approximation error, respectively. Similar estimations on $\mathcal{E}_{\mathbf{sam}}$ and $\mathcal{E}_{\mathbf{app}}$ as those in Subsection II D can be developed using tools from probability theory and approximation theory, which should be studied in the future. In the coming section, we focus on the optimization error, i.e., we study optimization procedures for solving (5).

III. CONVEX OPTIMIZATION OF ENERGIES REGRESSION

A. The basics of convex optimization

Statistical learning problem of molecules' energies regression described in the previous section is modelled generically as the following composite convex optimization problem, considered as a sum of a data-fitting term and an explicit penalty term,

$$\Psi^\star := \min_{\mathbf{c}\in\mathbf{C}\subset\mathbb{R}^n} \{\Psi(\mathbf{c}) := g(\mathbf{c}) + h(\mathbf{M}\mathbf{c})\}, \tag{10}$$

where $\mathbf{C}$ is a convex subset, $\mathbf{M}$ is an $n \times m$ matrix, g and h are convex functions. In most cases, finding an exact solution of (10) is impossible. We hence try to find one its approximated solution, i.e., given a tolerance $\varepsilon > 0$, we will design methods in order to obtain $\mathbf{c} \in \mathbf{C}$ such that $\Psi(\mathbf{c}) - \Psi^\star \leqslant \varepsilon$.

Before reviewing efficient numerical methods to approximate an optimal solution $\mathbf{c}^\star$ of (10) as well as required assumptions on h and g in the next sections, it is worthy to note that (10) covers the classical kernel ridge regression. Indeed, given a kernel matrix $\mathbf{K}$, due to the representer theorem II.3, a predictor $\hat{f}_\lambda$ is determined by a regression vector $\mathbf{c}^\natural$ which is, in traditional kernel ridge regression[8], is estimated by $\mathbf{c}^\star$ defined via the following formula

$$\mathbf{c}^\star = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}\mathbf{y}, \tag{KRR}$$

where $\mathbf{y} = [y_1, \ldots, y_n]^\top$ is the vector of labels and $\mathbf{I}_n$ is the $n \times n$ identity matrix. Simple calculations show that $\mathbf{c}^\star$ is a solution to the following ridge-regularized least square minimization problem, considered as a particular instance of (10),

$$\min_{\mathbf{c}\in\mathbb{R}^n}\ \frac{1}{2}\|\mathbf{K}\mathbf{c} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\mathbf{c}^\top\mathbf{K}\mathbf{c}. \tag{ℓ_2}$$

This optimization problem however might have more than one solution than $\mathbf{c}^\star$.

Being inspired by a recent interest in MAE and MaxAE metrics within the materials science community, we seek

new kernel-based models that could improve these metrics while still keep RMSE in the same order of magnitude as (KRR). The first step is exploring in deep the inner structure of these metrics. For instance, in the light of the mathematical definition of MAE, we understand that in order to adapt it, instead of using ℓ_2-loss function as in (ℓ_2), it could be better if we use ℓ_1-loss function.

Example III.1 (Ridge ℓ_1-loss regression) Instead of estimating $\mathbf{c}^\natural$ by (ℓ_2), we estimate it by solving the following minimization problem:

$$\min_{\mathbf{c}\in\mathbb{R}^n} \|\mathbf{K}\mathbf{c}-\mathbf{y}\|_1 + \frac{\lambda}{2}\mathbf{c}^\top\mathbf{K}\mathbf{c}. \qquad (\ell_1)$$

Our second model deploys the ℓ_∞-norm loss function to adapt the structure of MaxAE.

Example III.2 (Ridge ℓ_∞-loss regression) Estimate regression coefficient $\mathbf{c}^\natural$ based on the following minimization problem:

$$\min_{\mathbf{c}\in\mathbb{R}^n} \|\mathbf{K}\mathbf{c}-\mathbf{y}\|_\infty + \frac{\lambda}{2}\mathbf{c}^\top\mathbf{K}\mathbf{c}. \qquad (\ell_\infty)$$

Both (ℓ_1) and (ℓ_∞) lie beyond the effective regime of methods of linear algebra and smooth optimization and we, therefore, need deeper numerical methods to approximate their solutions. Very common approaches in optimization and machine learning community are first-order methods. These methods obtain reasonable accuracy numerical solutions by using only first-order oracle information from the objective, such as gradient estimates. They can also handle the non-smooth variants by making use of the proximal mapping principle. Main advantages of these methods are their scalability and nearly dimension-independent convergence rates. Coupled with recent demand for low-to-medium accuracy solutions in applications, these methods indeed provide a critical trade-off between the complexity-per-iteration and the iteration-convergence rate along with the ability to distribute and parallelize computation.

Assumption 1. We assume that ∇g is L_g-Lipschitz continuous, i.e.,

$$(\forall \mathbf{c}_1\in\mathbb{R}^n)(\forall \mathbf{c}_2\in\mathbb{R}^n) \quad \|\nabla g(\mathbf{c}_1)-\nabla g(\mathbf{c}_2)\| \leqslant L_g\|\mathbf{c}_1-\mathbf{c}_2\|.$$

B. Another approach to (KRR) from optimization point of view

As being discussed above, (KRR) is one among possibly many approaches to obtaining a solution of (ℓ_2). However, (KRR) requires a matrix inversion operation whose computational cost will become more expensively when the size of the kernel increases. In order to avoid this demanding operation, we suggest using iterative methods in optimization to obtain a solution to (ℓ_2). Because (ℓ_2) is a smooth convex optimization problem, among efficient methods to solve it are various versions of gradient descent method and stochastic gradient method[37–39] and references therein. Such generic algorithmic procedures are of the following protocol:

Algorithm 1 (stochastic) Gradient method

1: Inputs: $\mathbf{c}^0\in\mathbb{R}^n$.
2: **for** $k=1,2,\ldots,N-1$ **do** : choose $G^k\in\mathbb{R}^n$, a stepsize $\gamma_k\in(0,+\infty)$ and update

$$\mathbf{c}^{k+1}=\mathbf{c}^k-\gamma_k G^k.$$

3: **end for**
4: **return** $\mathbf{c}^N$.

In deterministic gradient descent method, $G^k=\nabla\Psi(\mathbf{c}^k)$ while in stochastic gradient descent method, G^k is chosen to be an unbiased estimate of $\nabla\Psi(\mathbf{c}^k)$, i.e., $\mathbb{E}[G^k]=\nabla\Psi(\mathbf{c}^k)$.

C. A primal first-order method

Loss function h in general can be non-smooth as in the case of ℓ_1-loss and ℓ_∞-loss. In these situations, sharper efforts to deal with (10) are necessary due to the presence of a nontrivial matrix $\mathbf{M}$. However, in the case when this matrix is identity, i.e.,

$$\Psi^\star := \min_{\mathbf{c}\in\mathbb{R}^n}\{\Psi(\mathbf{c}) := h(\mathbf{c})+g(\mathbf{c})\}, \qquad (11)$$

where g satisfies Assumption 1, we can solve it efficiently by different versions of proximal-gradient method.

The optimization methods, including forward-backward, forward-backward-forward, Tseng's method, etc, are studied extensively in[40–43] and the references therein. These methods make use of a linear approximation of smooth g and simply including the nonsmooth term h in an explicit fashion as follows:

$$\mathbf{c}^{k+1}=\operatorname*{argmin}_{\mathbf{c}\in\mathbb{R}^n} g(\mathbf{c}^k)+\nabla g(\mathbf{c}^k)^\top(\mathbf{c}-\mathbf{c}^k)+\frac{1}{2\alpha_k}\|\mathbf{c}-\mathbf{c}^k\|_2^2+h(\mathbf{c}), \qquad (12)$$

with the step-size $\alpha_k\leqslant 1/L_g$. The update rule in the formal proximal-gradient method is rewritten as

$$\mathbf{c}^{k+1}=\operatorname{prox}_{\alpha_k h}(\mathbf{c}^k-\alpha_k\nabla g(\mathbf{c}^k)), \qquad (13)$$

where the proximal operator is defined as

$$\operatorname{prox}_h(\bar{\mathbf{c}})=\operatorname*{argmin}_{\mathbf{c}\in\mathbb{R}^n} h(\mathbf{c})+\frac{1}{2}\|\mathbf{c}-\bar{\mathbf{c}}\|_2^2. \qquad (14)$$

Simply set the step-size $\alpha_k=1/L_g$, the proximal gradient method achieves the following convergence rate

$$\Psi(\mathbf{c}^N)-\Psi^\star=\mathcal{O}\big(N^{-1}\big). \qquad (15)$$

This result is explained as follow: in order to obtain an ε-approximated solution, we need $\mathcal{O}(\varepsilon^{-1})$ iterations. This

rate can be upgraded to $\mathcal{O}(N^{-2})$ by making use of an extra-momentum step and hence only $\mathcal{O}(\varepsilon^{-1/2})$ iterations will be executed to get an ε-approximation solution. The full version of this method is the following:

Algorithm 2 Fast iterative shrinkage-thresholding algorithm (FISTA)[44]

1: Inputs: $\hat{\mathbf{c}}^1 = \mathbf{c}^0 \in \mathbb{R}^n$, $t_1 = 1$.
2: **for** $k = 1, 2, \ldots, N-1$ **do**

$$\begin{aligned}\mathbf{c}^k &= \mathrm{prox}_{\alpha_k h}(\hat{\mathbf{c}}^{\mathbf{k}} - \alpha_k \nabla g(\hat{\mathbf{c}}^{\mathbf{k}})),\\ t_{k+1} &= 0.5\left(1 + \sqrt{1 + 4t_k^2}\right)\\ \hat{\mathbf{c}}^{k+1} &= \mathbf{c}^k + \frac{t_k - 1}{t_{k+1}}\big(\mathbf{c}^{\mathbf{k}} - \mathbf{c}^{k-1}\big).\end{aligned}$$

3: **end for**
4: **return** $\mathbf{c}^N$.

Example III.3 (Proximal operators) . Given $\gamma > 0$ and $\bar{\mathbf{c}} \in \mathbb{R}^n$.

1. Proximal operator of ℓ_1-norm $\|\cdot\|_1$: finding a solution to

$$\underset{\mathbf{c}\in\mathbb{R}^n}{\operatorname{argmin}} \ \|\mathbf{c}\|_1 + \frac{1}{2\gamma}\|\mathbf{c} - \bar{\mathbf{c}}\|_2^2$$

is equivalent to solving following n convex problems in dimension 1:

$$\min_{c\in\mathbb{R}} \ |c| + \frac{1}{2\gamma}|c - \bar{c}|^2.$$

Elementary computations show that this problem has an analytical solution given by $\tau_\gamma(\bar{c})$, where τ is the shrinkage operator defined by $\tau_\gamma(c) = (|\bar{c}| - \gamma)_+ \operatorname{sign}(\bar{c})$.

2. Proximal operator of ℓ_∞-norm $\|\cdot\|_\infty$: finding a solution to

$$\hat{\mathbf{c}} = \underset{\mathbf{c}\in\mathbb{R}^n}{\operatorname{argmin}} \ \|\mathbf{c}\|_\infty + \frac{1}{2\gamma}\|\mathbf{c} - \bar{\mathbf{c}}\|_2^2$$

can be proceeded as follow:

   (a) Compute $\tilde{\mathbf{c}}$, the projection of $\gamma^{-1}\bar{\mathbf{c}}$ onto the unit ℓ_1-ball $\{\|\mathbf{c}\|_1 \leqslant 1\}$[45].

   (b) Apply[43] to obtain $\hat{\mathbf{c}} = \bar{\mathbf{c}} - \gamma\tilde{\mathbf{c}}$.

D. A primal-dual first-order method

When a non-identity matrix $\mathbf{M}$ is incorporated in the nonsmooth term as in the full version of (10), proximal-gradient methods are no longer available. In such a situation, efficient alternatives are primal-dual methods. Various references on these methods include[46–48] and references therein. These methods are basically built based on representing h via its Fenchel conjugate function h^*, defined as

$$h^*(\mathbf{d}) = \sup_{\mathbf{c}\in\mathbb{R}^n} \ \mathbf{c}^\top\mathbf{d} - h(\mathbf{c}).$$

Making use of Fenchel conjugation, we translate the original composite function incorporating loss function h together with composition with a non-identity matrix $\mathbf{M}$ into the dual function of h. This introduces new dual variables in primal-dual algorithms. More specifically,

$$h(\mathbf{Mc}) = \sup_{\mathbf{d}\in\mathbb{R}^n} \ (\mathbf{Mc})^\top\mathbf{d} - h^*(\mathbf{d}).$$

A very common primal-dual method used to construct an approximated solution to (10) is the following:

Algorithm 3 Accelerated Primal-Dual method[49]

1: Inputs: $\mathbf{c}^1 \in \mathbb{R}^n$, $\mathbf{d}^1 \in \mathbb{R}^n$, $\mathbf{c}_{ag}^1 = \mathbf{c}^1$, $\mathbf{d}_{ag}^1 = \mathbf{d}^1$, $\bar{\mathbf{c}}^1 = \mathbf{c}^1$.
2: **for** $k = 1, 2, \ldots, N-1$ **do**

$$\begin{aligned}\mathbf{c}_{md}^k &= (1 - \beta_k^{-1})\mathbf{c}_{ag}^k + \beta_k^{-1}\mathbf{c}^k,\\ \mathbf{d}^{k+1} &= \mathrm{prox}_{\tau_k h^*}(\mathbf{d}^k - \mathbf{M}\bar{\mathbf{c}}_k),\\ \mathbf{c}^{k+1} &= \mathbf{c}^k - \eta_k(\nabla g(\mathbf{c}_{md}^k) + \mathbf{M}^\top\mathbf{d}^{k+1}),\\ \mathbf{c}_{ag}^{k+1} &= (1 - \beta_k^{-1})\mathbf{c}_{ag}^k + \beta_k^{-1}\mathbf{c}^{k+1},\\ \mathbf{d}_{ag}^{k+1} &= (1 - \beta_k^{-1})\mathbf{d}_{ag}^k + \beta_k^{-1}\mathbf{d}^{k+1},\\ \bar{\mathbf{c}}^{k+1} &= \theta_k(\mathbf{c}^{k+1} - \mathbf{c}^k) + \mathbf{c}^{k+1}.\end{aligned}$$

3: **end for**
4: **return** $\mathbf{c}_{ag}^N$ and $\mathbf{d}_{ag}^N$.

The main drawback of primal-dual methods in comparison to primal methods is that these methods introduce new dual variables which will increase the size of the problem. However, due to the flexibility of these methods, they can handle very general models.

Similar to the proximal-gradient method, the primal-dual method presented above requires the computation of the proximal operator of Fenchel conjugate function which can be deduced from the proximal operator of the original function itself due to[43].

Convergence's rate: simply set $\beta_k = \frac{k+1}{2}$, $\theta_k = \frac{k-1}{k}$, $\eta_k = \frac{3k}{4\eta}$ and $\tau_k = \frac{1}{\eta}$ for $\eta = 2L_g + 2\|\mathbf{M}\|(N-1) + \frac{N\sqrt{13(N-1)}}{2\tilde{D}}$ with $\tilde{D} > 0$, the convergence rate that a primal-dual method can achieve is

$$\Psi(\mathbf{c}_{ag}^N) - \Psi^\star = \mathcal{O}\big(N^{-1}\big).$$

IV. NUMERICAL EXPERIMENTS

A. GDB9 data set

GDB9 dataset[50] consisting of chemical representations and the internal energies U_0 (Hartree) at absolute zero temperature of 133884 small organic molecules. We divide this dataset into two parts: the train set contains 100000 molecules and the test set contains 33884

molecules. Using (9) with $\zeta \in \{2, 3, 4\}$, we obtain there learning kernels matrices $\mathbf{K}_2$, $\mathbf{K}_3$ and $\mathbf{K}_4$. They are then weighted as follow

$$\mathbf{K} = \frac{256}{273}\mathbf{K}_2 + \frac{16}{273}\mathbf{K}_3 + \frac{1}{273}\mathbf{K}_4.$$

Portions of this 100000×100000-matrix will be incorporated in regression models (KRR), (ℓ_1) and (ℓ_∞) to predict U_0 for 33884 molecules in the whole test set.

B. Computational complexity

With a given kernel matrix of size $n \times n$, (KRR) itself requires the computation of the inverse of an $n \times n$-matrix and the multiplication of an $n \times n$-matrix and an n-coordinates vector. The best known overall complexity is $\mathcal{O}(n^{2.373})$.

On the one hand, recall that both models (ℓ_1) and (ℓ_∞) with SOAP-REMatch kernels could be solved numerically by standard optimization approaches such as Algorithm 3 and the complexity of this method is essentially the computational complexity of the proximal operator. For instance, since the computational complexity of the soft-threshold is only $\mathcal{O}(n)$, the overall complexity of (ℓ_1) is overall $\mathcal{O}(nN)$ with N is the total number of iterations.

On the other hand, it was discussed previously that primal-dual methods iteratively build an approximate solution with the convergence rate of $\mathcal{O}(1/N)$.

In order to accelerate the convergence's speed, we propose two techniques concerning models (ℓ_1) and (ℓ_∞). These techniques allow us to apply Algorithm 3 either on the primal problem or on the dual problem.

Preconditioning. Our main idea to replace the original kernel matrix by a small perturbation determined by a small parameter ρ, i.e., setting $\mathbf{M} = \mathbf{K} + \rho\mathbf{I}_n$ and then making the change of the variable $\mathbf{d} = \mathbf{M}\mathbf{c}$ to reformulate (ℓ_1) as

$$\min_{\mathbf{d}\in\mathbb{R}^n} \ \|\mathbf{d} - \mathbf{y}\|_1 + \frac{\lambda}{2}\mathbf{d}^\top\mathbf{M}^{-1}\mathbf{d}, \tag{16}$$

and (ℓ_∞) to be

$$\min_{\mathbf{d}\in\mathbb{R}^n} \ \|\mathbf{d} - \mathbf{y}\|_\infty + \frac{\lambda}{2}\mathbf{d}^\top\mathbf{M}^{-1}\mathbf{d}. \tag{17}$$

This small value of ρ can be selected by the cross-validation procedure. Many researchers choose to set it to be a small number such as 10^{-8}. The described perturbation technique will remove the non-identity matrix from the non-smooth function. As a consequence, these new problems are solved efficiently and quickly by FISTA (Algorithm 2) with the overall complexity $\mathcal{O}(n^{2.273}+nN)$. Thanks to the convergence rate $\mathcal{O}(1/N^2)$ of FISTA, our execution will converge to a solution in predefined tolerance within a reasonably small number of iterations N, and hence it only requires a computation cost not much more than (KRR). This observation measured by necessary CPU times to run these models, will be recorded in Figure 1.

Dual formulation. The preconditioning technique presented above requires a matrix inversion computation. This operation is required even for (KRR). In order to avoid this expensive calculation, we introduce the use of dual formulations of (16) and (17), respectively as,

$$\min_{\mathbf{d}\in\mathbb{R}^n} \ g^*(\mathbf{d}) + \frac{1}{2\lambda}\mathbf{d}^\top\mathbf{K}\mathbf{d}, \tag{18}$$

where g^* is the Fenchel conjugate function of either $\|\cdot - \mathbf{y}\|_1$ for (ℓ_1) or $\|\cdot - \mathbf{y}\|_\infty$ for (ℓ_∞). These dual formulations can be solved efficiently by FISTA. By duality[43], the original coefficient is then recovered by $\mathbf{c}^N = -\lambda^{-1}\mathbf{d}^N$. Because the computational complexity of the proximal operator of g^* is the same as that of of g, the total computational complexity in these cases is only $\mathcal{O}(nN)$.

C. Cross-validation

We tune the parameter λ using 10-folds cross-validation by screening 15 values on a base-10 logarithmic grid from 10^{-9} to 10^0. This procedure can be parallelized in practice.

D. Simulation results

Different training sizes of 500, 1000, 5000, 10000, 25000, extracted from the train set of 100000 molecules, are used to predict U_0 using (KRR), (ℓ_1) and (ℓ_∞) for molecules in the test set and then we compute the metrics: MAE, MaxAE and RMSE on test molecules. We also record training times on a single CPU and standard deviations which are obtained by running 10 tests with randomly chosen varying training set data sub-selection. Results, recorded in Figures 1-3, show that (ℓ_1) improves significantly MAE in comparison with (KRR) while (ℓ_∞) achieves a better MaxAE than (KRR).

Remark IV.1 We observe from FIG. 4 that CPU times consumed by (ℓ_1) and (ℓ_∞) excess those of (KRR) as these methods need to execute an iterative algorithm for each value of hyper-parameters. We particularly note that (ℓ_∞) is more computationally expensive among three models and the reason is that it used an extra inner loop to compute proximal operator of ℓ_∞-loss function.

V. CONCLUSIONS

We study the similarity in the structure of loss functions and the corresponding statistical errors to understand the impact of loss functions on these metrics. We the present novel settings together with efficient algorithms to approximate a numerical solution for molecules'

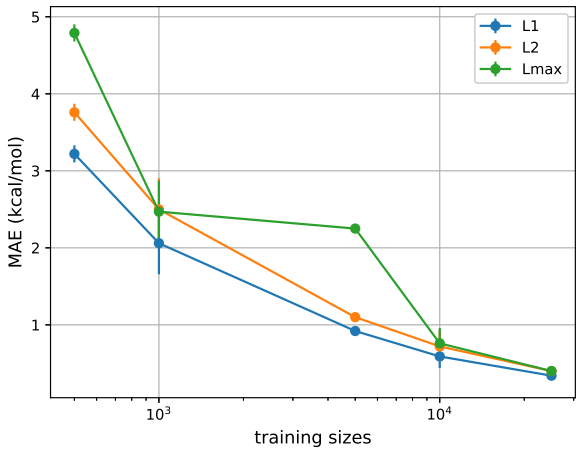

FIG. 1. Mean absolute error on test data: learning curve (unit: kcal/mol, 10 tests implemented on GDB9 data set).

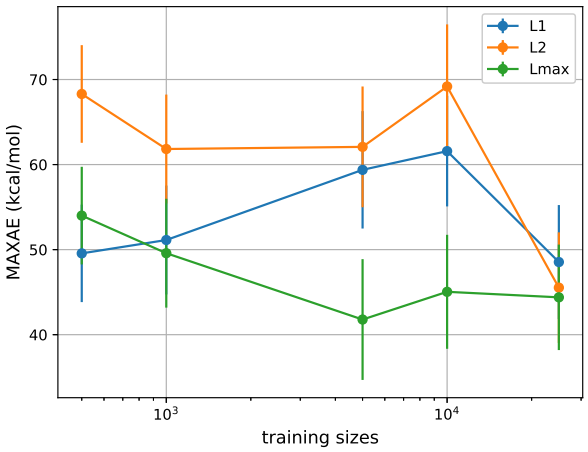

FIG. 3. Max absolute error on test data: learning curve (unit: kcal/mol, 10 tests implemented on GDB9 data set).

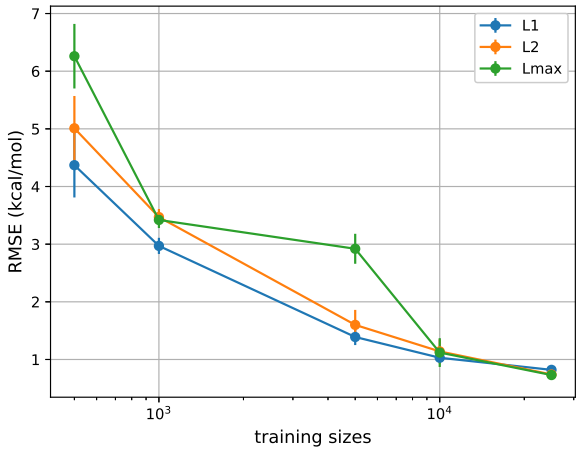

FIG. 2. Root mean square error on test data: learning curve (unit: kcal/mol, 10 tests implemented on GDB9 data set).

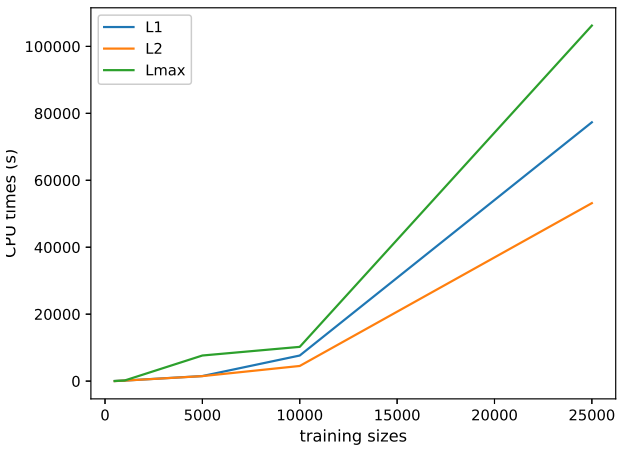

FIG. 4. CPU times in seconds (for each train size, 10 tests are implemented and the average time is then computed).

atomization energy prediction via statistical learning theory. We also describe numerical advantages of our approach with two new models: ridge ℓ_1-loss minimization and ridge ℓ_∞-loss minimization. To the best of our knowledge, these formulations are the first considering a very generic point of view of the chemical machine learning from statistical learning theory. This new insight highlights the potentials of our current research.

ACKNOWLEDGEMENTS

This research was supported by the NCCR MARVEL, a National Centres of Competence in Research, funded by the Swiss National Science Foundation. This project has also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 725594 - time-data).

The authors would like to thank Prof. Michele Ceriotti for useful discussions.

REFERENCES

[1] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," Physical review **136**, B864 (1964).
[2] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," Physical review **140**, A1133 (1965).
[3] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, "Fast and accurate modeling of molecular atomization ener-

gies with machine learning," Physical review letters **108**, 058301 (2012).
[4]G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, A. V. Lilienfeld, and K.-R. Müller, "Learning invariant representations of molecules for atomization energy prediction," in *Advances in Neural Information Processing Systems* (2012) pp. 440–448.
[5]G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, "Machine learning of molecular electronic properties in chemical compound space," New Journal of Physics **15**, 095003 (2013).
[6]G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, "Machine learning of molecular electronic properties in chemical compound space," New Journal of Physics **15**, 095003 (2013).
[7]A. P. Bartók, M. J. Gillan, F. R. Manby, and G. Csányi, "Machine-learning approach for one- and two-body corrections to density functional theory: Applications to molecular and condensed water," Phys. Rev. B **88**, 054104 (2013).
[8]M. Rupp, "Machine learning for quantum mechanics in a nutshell," International Journal of Quantum Chemistry **115**, 1058–1073 (2015).
[9]F. Faber, A. Lindmaa, O. A. von Lilienfeld, and R. Armiento, "Crystal structure representations for machine learning models of formation energies," International Journal of Quantum Chemistry **115**, 1094–1101 (2015).
[10]O. A. von Lilienfeld, R. Ramakrishnan, M. Rupp, and A. Knoll, "Fourier series of atomic radial distribution functions: A molecular fingerprint for machine learning models of quantum chemical properties," International Journal of Quantum Chemistry **115**, 1084–1093 (2015).
[11]K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. Von Lilienfeld, K.-R. Mller, and A. Tkatchenko, "Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space," The journal of physical chemistry letters **6**, 2326–2331 (2015).
[12]V. Botu and R. Ramprasad, "Learning scheme to predict atomic forces and accelerate materials simulations," Physical Review B **92**, 094306 (2015).
[13]B. Huang and O. A. von Lilienfeld, "Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity," (2016).
[14]F. A. Faber, A. S. Christensen, B. Huang, and O. A. von Lilienfeld, "Alchemical and structural distribution based representation for universal quantum machine learning," The Journal of Chemical Physics **148**, 241717 (2018).
[15]C. R. Collins, G. J. Gordon, O. A. von Lilienfeld, and D. J. Yaron, "Constant size descriptors for accurate machine learning models of molecular properties," The Journal of Chemical Physics **148**, 241718 (2018).
[16]D. J. Wright, G. Capon, R. Page, J. Quiroga, A. A. Taseen, and F. Tomasini, "Evaluation of forecasting methods for decision support," International journal of forecasting **2**, 139–152 (1986).
[17]A. P. Bartók, R. Kondor, and G. Csányi, "On representing chemical environments," Physical Review B **87**, 184115 (2013).
[18]V. N. Vapnik and V. Vapnik, *Statistical learning theory*, Vol. 1 (Wiley New York, 1998).
[19]F. Cucker and D. X. Zhou, *Learning theory: an approximation theory viewpoint*, Vol. 24 (Cambridge University Press, 2007).
[20]I. Steinwart and A. Christmann, *Support vector machines* (Springer Science & Business Media, 2008).
[21]S. A. Geer, *Empirical Processes in M-estimation*, Vol. 6 (Cambridge university press, 2000).
[22]O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in neural information processing systems* (2008) pp. 161–168.
[23]S. Boucheron, O. Bousquet, and G. Lugosi, "Theory of classification: A survey of some recent advances," ESAIM: probability and statistics **9**, 323–375 (2005).
[24]T. Zhang, "Statistical behavior and consistency of classification methods based on convex risk minimization," Annals of Statistics , 56–85 (2004).
[25]I. Steinwart and C. Scovel, "Fast rates for support vector machines using gaussian kernels," The Annals of Statistics , 575–607 (2007).
[26]S. Smale and D.-X. Zhou, "Learning theory estimates via integral operators and their approximations," Constructive approximation **26**, 153–172 (2007).
[27]A. Caponnetto and E. De Vito, "Optimal rates for the regularized least-squares algorithm," Foundations of Computational Mathematics **7**, 331–368 (2007).
[28]J. Lin and V. Cevher, "Optimal rates for spectral-regularized algorithms with least-squares regression over hilbert spaces," arXiv preprint arXiv:1801.06720 (2018).
[29]P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," Journal of Machine Learning Research **3**, 463–482 (2002).
[30]R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, Vol. 14 (Stanford, CA, 1995) pp. 1137–1145.
[31]T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," The annals of statistics , 1171–1220 (2008).
[32]S. De, A. P. Bartók, G. Csányi, and M. Ceriotti, "Comparing molecules and solids across structural and alchemical space," Physical Chemistry Chemical Physics **18**, 13754–13769 (2016).
[33]A. P. Bartók, S. De, C. Poelking, N. Bernstein, J. R. Kermode, G. Csányi, and M. Ceriotti, "Machine learning unifies the modeling of materials and molecules," Science advances **3**, e1701816 (2017).
[34]S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," Journal of Machine Learning Research **7**, 1531–1565 (2006).
[35]H. Kadri, A. Rakotomamonjy, F. Bach, and P. Preux, "Multiple operator-valued kernel learning," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12 (Curran Associates Inc., USA, 2012) pp. 2429–2437.
[36]H. Kadri, E. Duflos, P. Preux, S. Canu, A. Rakotomamonjy, and J. Audiffren, "Operator-valued kernels for learning from functional response data," The Journal of Machine Learning Research **17**, 613–666 (2016).
[37]Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*, Vol. 13 (Siam, 1994).
[38]D. Scieur, F. Bach, and A. d'Aspremont, "Nonlinear acceleration of stochastic algorithms," in *Advances in Neural Information Processing Systems* (2017) pp. 3985–3994.
[39]Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," SIAM Journal on Optimization **22**, 341–362 (2012).
[40]Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2)," in *Doklady AN USSR*, Vol. 269 (1983) pp. 543–547.
[41]S. Boyd and L. Vandenberghe, *Convex optimization* (Cambridge university press, 2004).
[42]L. M. Briceno-Arias, P. L. Combettes, J.-C. Pesquet, and N. Pustelnik, "Proximal algorithms for multicomponent image recovery problems," Journal of Mathematical Imaging and Vision **41**, 3–22 (2011).
[43]H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*, Vol. 408 (Springer, 2011).
[44]A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," SIAM journal on imaging sciences **2**, 183–202 (2009).
[45]J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l_1-ball for learning in high dimensions," in *Proceedings of the 25th international conference on Machine learning* (ACM, 2008) pp. 272–279.
[46]A. Chambolle and T. Pock, "A first-order primal-dual algorithm

for convex problems with applications to imaging," Journal of mathematical imaging and vision **40**, 120–145 (2011).
[47]P. L. Combettes and J.-C. Pesquet, "Primal-dual splitting algorithm for solving inclusions with mixtures of composite, lipschitzian, and parallel-sum type monotone operators," Set-Valued and variational analysis **20**, 307–330 (2012).
[48]Q. Tran-Dinh, O. Fercoq, and V. Cevher, "A smooth primal-dual optimization framework for nonsmooth composite convex minimization," SIAM Journal on Optimization **28**, 96–134 (2018).
[49]Y. Chen, G. Lan, and Y. Ouyang, "Optimal primal-dual methods for a class of saddle point problems," SIAM J. Optim. **24**, 1779–1814 (2014).
[50]R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," Scientific data **1**, 140022 (2014).

APPENDIX

In this appendix, we provide the proofs for some of the elementary inequalities in this paper.

A. Proof of (3)

We have

$$\begin{aligned}
&\mathcal{R}(\hat{f}_\lambda) - \mathcal{R}(f^\star) \\
\leqslant&\mathcal{R}^\lambda(\hat{f}_\lambda) - \mathcal{R}(f^\star) \\
=&[\mathcal{R}(\hat{f}_\lambda) - \mathcal{R}_n(\hat{f}_\lambda) + \mathcal{R}_n(f_\lambda) - \mathcal{R}(f_\lambda)] + [\mathcal{R}_n^\lambda(\hat{f}_\lambda) - \mathcal{R}_n^\lambda(f_\lambda)] \\
&+ [\mathcal{R}^\lambda(f_\lambda) - \mathcal{R}(f^\star)] \\
\leqslant&[\mathcal{R}(\hat{f}_\lambda) - \mathcal{R}_n(\hat{f}_\lambda) + \mathcal{R}_n(f_\lambda) - \mathcal{R}(f_\lambda)] + [\mathcal{R}^\lambda(f_\lambda) - \mathcal{R}(f^\star)] \\
=&\mathcal{E}_{\mathbf{sam}} + \mathcal{E}_{\mathbf{app}},
\end{aligned}$$

where for the inequality, we used the fact that $\mathcal{R}_n^\lambda(\hat{f}_\lambda) \leqslant \mathcal{R}_n^\lambda(f_\lambda)$ since $\hat{f}_\lambda$ is a solution for (2).

B. Proof for (5)

Due to the representer theorem, instead of solving optimization problem (4) with respect to functions f, we find the weights of the predictor, a vector $\mathbf{c}^\natural$ in a finite dimensional space. For $\hat{f}_\lambda(\mathbf{x}) = \sum_{i=1}^n c_i^\natural \mathcal{K}(\mathbf{x}, \mathbf{x}_i)$, we have that for all i,

$$\begin{aligned}
\ell(\hat{f}_\lambda(\mathbf{x}_i), y_i) &= \ell\bigg(\sum_{j=1}^n c_j^\natural \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j), y_i\bigg) \\
&= \ell\bigg(\sum_{j=1}^n c_j^\natural \mathbf{K}_{ij}, y_i\bigg) = \ell\Big((\mathbf{c}^\natural)^\top \mathbf{K}_i, y_i\Big),
\end{aligned}$$

and since

$$\begin{aligned}
\Big\langle \hat{f}_\lambda, \psi(\mathbf{x}) \Big\rangle = \hat{f}_\lambda(\mathbf{x}) &= \sum_{i=1}^n c_i^\natural \mathcal{K}(\mathbf{x}, \mathbf{x}_i) \\
= \sum_{i=1}^n c_i^\natural \left\langle \psi(\mathbf{x}), \psi(\mathbf{x}_i) \right\rangle &= \left\langle \psi(\mathbf{x}), \sum_{i=1}^n c_i^\natural \psi(\mathbf{x}_i) \right\rangle,
\end{aligned}$$

we have $\hat{f}_\lambda = \sum_{i=1}^n c_i^\natural \psi(\mathbf{x}_i)$ and hence,

$$\begin{aligned}
\|\hat{f}_\lambda\|^2 &= \left\langle \sum_{i=1}^n c_i^\natural \psi(\mathbf{x}_i), \sum_{i=1}^n c_i^\natural \psi(\mathbf{x}_i) \right\rangle \\
&= \sum_{i,j=1}^n c_i^\natural c_j^\natural \left\langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \right\rangle \\
&= \sum_{i,j=1}^n c_i^\natural c_j^\natural \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) \\
&= (\mathbf{c}^\natural)^\top \mathbf{K} \mathbf{c}^\natural.
\end{aligned}$$

The problem (4) with ridge regularizer now becomes

$$\min_{\mathbf{c} \in \mathbb{R}^n} \ \frac{1}{n} \sum_{i=1}^n \ell\Big(\mathbf{c}^\top \mathbf{K}_i, y_i\Big) + \frac{\lambda}{2} \mathbf{c}^\top \mathbf{K} \mathbf{c}.$$