Tree based machine learning framework for predicting ground state energies of molecules

Burak Himmetoglu

Articles you may be interested in

Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity
The Journal of Chemical Physics **145**, 161102 (2016); 10.1063/1.4964627

Perspective: Machine learning potentials for atomistic simulations
The Journal of Chemical Physics **145**, 170901 (2016); 10.1063/1.4966192

The many-body expansion combined with neural networks
The Journal of Chemical Physics **146**, 014106 (2017); 10.1063/1.4973380

Acceleration of saddle-point searches with machine learning
The Journal of Chemical Physics **145**, 074106 (2016); 10.1063/1.4960708

Accelerating the search for global minima on potential energy surfaces using machine learning
The Journal of Chemical Physics **145**, 154106 (2016); 10.1063/1.4964671

Learning molecular energies using localized graph kernels
The Journal of Chemical Physics **146**, 114107 (2017); 10.1063/1.4978623

Tree based machine learning framework for predicting ground state energies of molecules

Burak Himmetoglu
Center for Scientific Computing, University of California, Santa Barbara, California 93106,
USA and Enterprise Technology Services, University of California, Santa Barbara, California 93106, USA

We present an application of the boosted regression tree algorithm for predicting ground state energies of molecules made up of C, H, N, O, P, and S (CHNOPS). The PubChem chemical compound database has been incorporated to construct a dataset of 16 242 molecules, whose electronic ground state energies have been computed using density functional theory. This dataset is used to train the boosted regression tree algorithm, which allows a computationally efficient and accurate prediction of molecular ground state energies. Predictions from boosted regression trees are compared with neural network regression, a widely used method in the literature, and shown to be more accurate with significantly reduced computational cost. The performance of the regression model trained using the CHNOPS set is also tested on a set of distinct molecules that contain additional Cl and Si atoms. It is shown that the learning algorithms lead to a rich and diverse possibility of applications in molecular discovery and materials informatics. *Published by AIP Publishing.* [http://dx.doi.org/10.1063/1.4964093]

I. INTRODUCTION

Advances in electronic structure theory in combination with ever increasing computing power have made quantum mechanical simulations of molecules and solids rather commonplace. As a consequence, a vast amount of compounds have been studied by various electronic structure methods. In the last decade, efforts to collect and catalog these simulation data, in addition to those obtained from experimental studies, have led to the ability to analyze and screen an extensive space of chemical compounds and materials.[1,2] The automated screening of the experimental and theoretical compound spaces has become a powerful tool not only for discovering new systems but also for rational design of chemicals and materials for targeted applications.[3–5] Moreover, with the availability of various databases, analyses based on powerful statistical/machine learning methods have become feasible. Despite the availability of a vast number of systems in these databases, screening for new systems which have not yet been reported before requires a large number of new simulations to be performed. Density functional theory[6] (DFT), based on the effective single-particle Kohn-Sham equations[7] has been a popular choice for performing accurate, yet computationally inexpensive simulations. Despite the relatively low cost of DFT simulations, screening the whole space of compounds and materials requires electronic structure predictions at a much lower computational cost, ideally without performing new simulations for each system. Machine learning algorithms are a perfect match for this task, since these algorithms could learn from a given database of electronic structure calculations and predict the electronic properties of a new set of systems (not included in the database) without the need for performing new simulations. Such a task requires a set of features (a.k.a descriptors) for each system which defines their electronic makeup. Then, the electronic structure can be described by a general nonlinear function of these features, which the learning algorithm determines by a sophisticated fit to a given database (i.e., training the learning algorithm). As a result, predictions on new systems can be readily obtained through the trained model parameters.

The idea of predicting electronic structure using data has already been investigated in the literature. Various learning algorithms and choice of features have been proposed to predict electronic structures of molecules and solids. Earliest investigations considered a combined DFT and machine learning based modeling for predicting potential energy surfaces, ground state energies, and formation enthalpies of molecules. These studies used neural networks[8–11] and support vector regression[12] as learning algorithms. In case of solids, features were constructed from calculated electronic band gaps, cohesive energies, and crystalline volumes for a set of inorganic compounds and used for predicting a new set of band gaps using support vector regression.[13] In principle, apart from the approximation to the exchange-correlation functional (E_{xc}), DFT calculations use atomic species and their positions as the only inputs for ground state electronic structure predictions.[14,37] In a similar manner, features for learning algorithms can be constructed based only on atomic coordinates and types, which provide an improvement over features based on calculated properties. For this objective, Coulomb matrices have been proposed as a robust set of features for a complete description of molecular systems.[15] With Coulomb matrices as features, various learning algorithms have been tested, ranging from nonlinear kernels to neural networks, which resulted in accurate predictions for atomization energies of molecules.[16]

Given the success of learning algorithms based on Coulomb matrices as features, it is therefore imperative

to analyze available datasets[17] with electronic structures determined by various DFT methods and implementations. For this purpose, we construct a database of electronic structure calculations based on the molecular structure data publicly available through the PubChem Substance and Compound database.[18] The electronic structure calculations are based on the highly scalable implementation of DFT which employs plane-waves and pseudopotentials. For the machine learning algorithm, we propose the adoption of boosted regression trees,[19] as a more computationally efficient alternative to previously used methods in the literature. As a demonstration, we show that the boosted regression trees outperform neural networks in predicting atomization energies, while significantly reducing the cost of model training. Our framework, based on Coulomb matrices as features and boosted regression trees as the machine learning algorithm, provides an accurate and efficient pathway to screen the chemical compound space for analyzing data and discovering new molecules.

The paper is organized as follows: We provide details about the computational methods used and the construction of the electronic structure database in Section II. In Section III, we provide a summary of the machine learning algorithms and the techniques used for model training. In Section IV, we present the main results and provide a discussion on the performance of the methods used. Finally, in Section V, we provide some concluding remarks.

II. METHODS

A. Obtaining data and computational methods

Molecular structures used in our study are generated from the PubChem Substance and Compound database.[18] The chemical structures which are probed have the substance identifier number (SID) ranging from 1 to 75 000. Using the structural information from this database, a subset of molecules are extracted based on the number and types of atoms, and the size of the molecules. This subset contains molecules that satisfy the following criteria: (i) Each molecule has to be composed of a subset of the elements from the set C, H, N, O, P and S (CHNOPS). (ii) Each molecule must have at least 2 and at most 50 atoms. (iii) The maximum distance between two atoms in a molecule must not exceed 25 a_0 (a_0 = 0.529 Å, i.e., Bohr radius), for the convergence of plane-wave calculations, where each molecule is placed in a cubic box of side length 30 a_0. (iv) There must be an even number of electrons in the molecule. Applying these criteria to the first 75 000 entries in the PubChem database leads to a subset of 16 242 molecules, whose structure data files (SDF) are converted into input files for electronic structure calculations.

The electronic structure calculations are performed using the plane-waves pseudopotential implementation of DFT in the PWSCF code of the *Quantum ESPRESSO* package.[20] The exchange-correlation energy is approximated using the generalized gradient approximation (GGA) with the Perdew-Burke-Ernzerhof (PBE) parametrization.[21] The choice of the PBE functional is purely due to its computational efficiency in the plane-waves basis set. Since the aim of this work is to

predict simulation results, PBE is sufficient for our purposes. Functionals with exact-exchange that perform better can also be used for calculation of ground state energies which can be fed into the same machine learning algorithms used in this study. All of the atoms are represented by ultrasoft pseudopotentials.[22] The electronic wavefunctions and charge density are expanded up to kinetic energy cutoffs of 30 Ry and 300 Ry, respectively. The calculations are performed in a cubic box of side length 30 a_0 with a single k-point at Γ. These choices lead to the calculation of the ground state energy with a numerical error of about 1 kcal/mol for the largest molecule in the dataset.

For each molecule in the dataset, we compute the pseudo-atomization energy (E_{ps}), which is the quantity that serves as the prediction (outcome) for the learning algorithms. We compute E_{ps} using

$$E_{ps} = E_{gs} - \sum_{\alpha=1}^{N} n_\alpha E_\alpha^{PS},\qquad(1)$$

where E_{gs} is the calculated DFT ground state energy, N is the number of atoms in the molecule, the index α specifies the type of the atom (belonging to the CHNOPS set), n_α is the number of atoms of type α, and E_α^{PS} is the pseudo-energy of the isolated atom of type α calculated during pseudopotential generation. All pseudopotentials are generated from the PSLibrary repository, provided in the *Quantum ESPRESSO* distribution. The histogram for the calculated pseudo-atomization energies from the dataset of 16 242 molecules are shown in Fig. 1. The total variability in the dataset, quantified by the standard deviation of E_{ps}, is 3.66 Ry (1147.89 kcal/mol), which is larger than the variability reported in some of the earlier works in the literature,[16,23] indicating that a much wider range of molecular systems being included in the current study. The consequences of this difference will be discussed in Section IV.

B. Data description and visualization

In order to build models for predictions from machine learning algorithms, construction of feature vectors (a.k.a
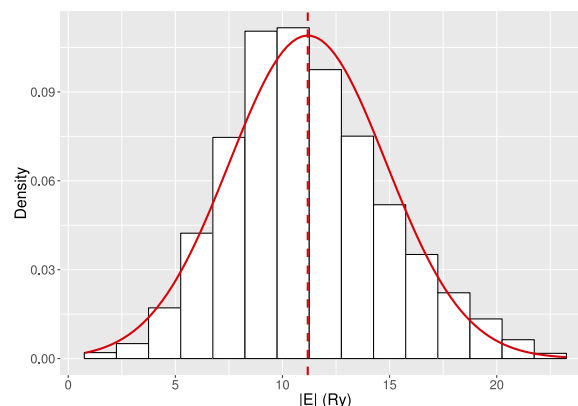

FIG. 1. Histogram of pseudo-atomization energies (E_{ps}). The mean value of $|E_{ps}|$ is 11.18 Ry (3506.37 kcal/mol) (indicated by the dashed vertical line) and the standard deviation is 3.66 Ry (1147.89 kcal/mol). A Gaussian probability density function with the same mean and standard deviation is also plotted for comparison.

descriptors) are needed to represent each molecule. We use the intermolecular Coulomb repulsion operators (will be referred to as Coulomb matrices from now on) introduced in Ref. 15, which are defined as

$$C_{IJ} = \begin{cases} 0.5\, Z_I^{2.4} & I = J \\ \dfrac{Z_I\, Z_J}{|\mathbf{R}_I - \mathbf{R}_J|} & I \neq J \end{cases}, \qquad (2)$$

where Z_I are atomic numbers, $\mathbf{R}_I$ are atomic coordinates, and indices I, J run over the atoms in a given molecule. The off-diagonal terms correspond to ionic repulsion between atoms I and J and the diagonal terms (both the pre-factor and power) are obtained from a fit of the atomic numbers to the energies of isolated atoms.[15] The Coulomb matrices represent the full set of parameters that DFT calculations take as inputs (Z_I and $\mathbf{R}_I$), aside from the approximation to E_{xc}, which are used to calculate the ground state energy. Therefore, the problem of predicting ground state energies (or atomization energies) using C_{IJ} is well defined. However, C_{IJ} does not provide a unique description, since a given molecule can be represented by more than one matrix that can be obtained by reshuffling the indices of the atoms. There are several ways to overcome this problem as described in Ref. 16. Here, aside from C_{IJ} itself, we will use its eigenspectrum (which is one of the unique representations proposed in Ref. 16) for a given molecule. Since we limited the number of atoms in a given molecule by 50, the Coulomb matrices we use are 50×50 matrices. Molecules with less than 50 atoms have their Coulomb matrices appended by columns and rows of 0 to complete them to have dimensions of 50×50.

A given molecule in the dataset numbered with index i is represented by a p-dimensional feature vector $\mathbf{x}_i$, where p is the total number of unique entries in the Coulomb matrix (i.e., the upper triangular part of the symmetric 50×50 matrix C_{IJ}, unrolled into a 1275 dimensional vector) or the number of eigenvalues (i.e., 50 dimensional vector of eigenvalues). The whole dataset is then cast in a data matrix $\mathbf{X}$ of dimensions $N \times p$, where N is the number of data points (16 242). In this representation, molecules are listed in rows, and each column is an entry in the p-dimensional feature vector $\mathbf{x}_i$ (for the ith molecule). Namely, for a given molecule with index i,

$$\mathbf{x}_i = \begin{bmatrix} C_{1,1}^{(i)} \\ C_{1,2}^{(i)} \\ \vdots \\ C_{1,50}^{(i)} \\ C_{2,2}^{(i)} \\ \vdots \\ C_{2,50}^{(i)} \\ \vdots \\ C_{49,50}^{(i)} \\ C_{50,50}^{(i)} \end{bmatrix}, \quad i = 1, \ldots, N \qquad (3)$$

when the full Coulomb matrix $C_{IJ}^{(i)}$ is used, while

$$\mathbf{x}_i = \begin{bmatrix} \lambda_1^{(i)} \\ \lambda_2^{(i)} \\ \vdots \\ \lambda_{50}^{(i)} \end{bmatrix}, \quad i = 1, \ldots, N \qquad (4)$$

when the eigenvalues $\lambda^{(i)}$ of $C_{IJ}^{(i)}$ are used. Finally, the data matrix $\mathbf{X}$ is explicitly given in terms of the feature vectors $\mathbf{x}_i$ by

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_N^{\mathrm{T}} \end{bmatrix}, \quad i = 1, \ldots, N. \qquad (5)$$

The outcomes, namely, the pseudo-atomization energies E_{ps}, are also cast in an N-dimensional vector explicitly given by

$$\mathbf{y} = \begin{bmatrix} E_{ps}^{(1)} \\ E_{ps}^{(2)} \\ \vdots \\ E_{ps}^{(N)} \end{bmatrix}, \quad i = 1, \ldots, N. \qquad (6)$$

As will be explained in Sec. III, the objective of the machine learning problem is to find a nonlinear function $f(\mathbf{X})$ that learns a relationship between the data $\mathbf{X}$ and the outcomes $\mathbf{y}$.

Fig. 1 depicts the overall variability in the outcome vector $\mathbf{y}$, namely, the distribution of E_{ps} in the dataset. However, it is also useful to explore the features $\mathbf{X}$ and how they relate to E_{ps}. This is not a straightforward task, since the data matrix $\mathbf{X}$ have a very large number of columns (1275 when data are represented by C_{IJ}, 50 when data is represented by λ). Instead of plotting E_{ps} as a function of all possible features, a smaller set of features that describe the overall trends in the data can be constructed using principal components analysis (PCA).[24] PCA enables one to obtain linear combinations of features (i.e., columns of $\mathbf{X}$) that explain the largest variation in the data. Before utilizing PCA, the data matrix is centered via the transformation

$$X_{ik} \rightarrow X_{ik} - \mu_k, \quad \mu_k = \frac{1}{N} \sum_{i=1}^{N} X_{ik}. \qquad (7)$$

This transformation ensures that each feature has zero mean, i.e., the column sums of $\mathbf{X}$ are 0. With the centered data, the covariance matrix takes a simple form

$$\Sigma_{ij} = \frac{1}{N} \sum_{k=1}^{N} X_{ik}^T X_{kj}. \qquad (8)$$

The principal components are constructed from the eigenvectors of the covariance matrix and are ordered with increasing eigenvalues. For instance, the first principal component corresponds to the linear combination of columns of $\mathbf{X}$ with the largest eigenvalue, hence has the largest variance. Formally, the jth principal component vector is given by
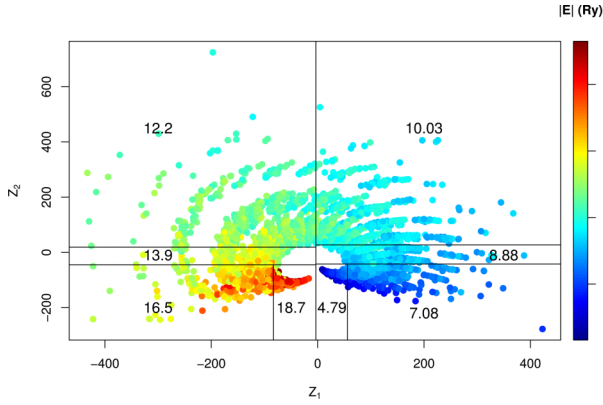
FIG. 2. E_{ps} as a function of the first two principal components Z_1 and Z_2. The reported values within each rectangular region are the mean values of E_{ps}, which are determined by the regression tree algorithm outlined in Section III.

$$(Z_j)_i = \sum_{k=1}^{p} \phi_k^{(j)} X_{ik}, \qquad (9)$$

where

$$\sum_{k=1}^{P} \Sigma_{ik} \phi_k^{(j)} = \sigma_{(j)}^2 \phi_i^{(j)}. \qquad (10)$$

In the above equation, $\sigma_{(j)}^2$ is the eigenvalue corresponding to principal component vector $\mathbf{z}_j$. Using this recipe, we have constructed the principal components of $\mathbf{X}$ using the eigenspectrum $\lambda^{(i)}$ (Eq. (4)). Among the 50 principal components, the first two account for 32% of the variability in the data (i.e., $\sigma_{(1)}^2 + \sigma_{(2)}^2 = 0.32 \times \text{Tr}[\Sigma]$). Fig. 2 illustrates E_{ps} as a function of the first two principal components (Z_1 and Z_2), which display a peculiar nonlinear dependence on the two features. In Sec. III, we will summarize some of the learning techniques that we have used to accurately model this nonlinear behavior of E_{ps} based on the data.

III. LEARNING METHODS

A. Overview

The main purpose of training a learning algorithm in the context of regression is to fit a nonlinear function $f(\mathbf{X})$ using the data $\mathbf{X}$ to predict the outcome $\mathbf{y}$. This is achieved via the minimization of a loss function which is a measure of the difference between the actual data and the fit. For instance,

in the case where f is parameterized by a p-dimensional vector θ, and the predictions of the outcomes $\mathbf{y}$ are given by $\hat{\mathbf{y}} = f_\theta(\mathbf{X})$, θ is obtained by optimizing

$$\min_\theta \left\{ \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + R(\theta) \right\}, \qquad (11)$$

where $y_i - \hat{y}_i$ are residuals and the loss function used for optimization is the residual sums squared (RSS). The term $R(\theta)$ is the regularization term and prevents overfitting.[24] For example, in the well-known case of LASSO,[24] a linear function $f_\theta(\mathbf{X}) = \mathbf{X} \cdot \theta$ is used for predictions $\hat{\mathbf{y}}$, and the regularization term is $R(\theta) = \gamma \|\theta\|$. While the parameter vector θ is obtained by minimizing Eq. (11), the regularization parameter γ is obtained by cross-validation. In this article, we consider two types of cross-validation approaches: (i) validation set and (ii) k-fold cross-validation. In both approaches, the full dataset is randomly divided into a training and test set. In this study, 70% of the data is randomly selected as the training set, and 30% as the test set. The training set is used to train the learning algorithm, i.e., to obtain values of the regularization parameter(s), while the test set, as an independent piece of the data, is used to report the accuracy of the trained model. The validation set and k-fold cross-validation approaches are schematically illustrated in Fig. 3. In the validation set approach, the training data are further split (randomly) into two, yielding a second level training set and a validation set. The model is trained on the training set for a range of parameters that determine the regularization term. Then, the model which results in the smallest mean squared error (MSE) is chosen. The MSE is the mean value of the error in the fit given by

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2. \qquad (12)$$

In case of LASSO, γ is the parameter determined from cross-validation. Finally, the first level training set (the original one with 70% of the data) is used to re-fit the model (with the regularization parameters fixed) and the final accuracy of the model is reported by the performance on the test set. While this method provides a clear path to determining the regularization parameters, the results generally depend on the way the data is split. The k-fold cross-validation approach attempts to resolve this issue by randomly dividing the training set into k groups of approximately equal size. For each division (i.e., fold), the validation set approach is repeated: the $(k-1)$ folds are used
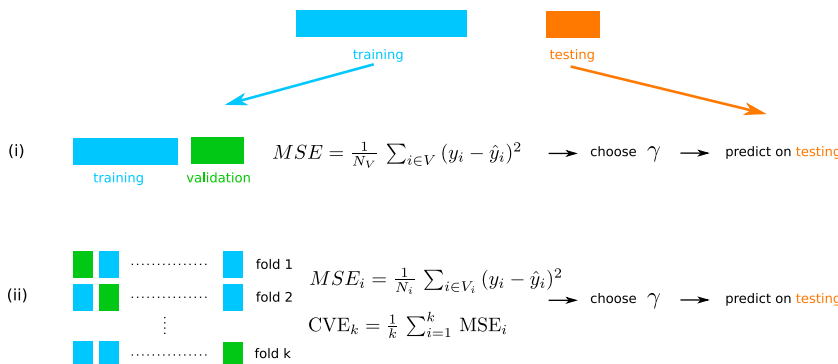

FIG. 3. (i) Validation set and (ii) k-fold cross-validation approaches used for training a model. In case of LASSO, a single regularization parameter γ is picked from the model with lowest cross-validation error. Final accuracy is reported on the independent testing set.

as the second level training set and the left out fold is used as the validation set. After obtaining k estimates for the MSE, the results are averaged and the model which leads to the smallest cross-validation error (CVE) is picked, as depicted in Fig. 3. k-fold cross-validation usually resolves the dependence of the model parameters on the splits used, at the expense of increased computational cost. We use 5-fold cross-validation for training learning algorithms in this study, except the case of neural networks when the full C_{IJ}'s are used as features (a validation set is used in that case).

B. Regression trees

The idea behind regression trees is to divide the space of features into regions where in each region, the value of the function $f(\mathbf{X})$ is the mean of the observations y_i inside. For example, Fig. 2 illustrates the result of a regression tree fitted using two features that are the principal components Z_1, Z_2. In each of the rectangular regions of the feature space spanned by $\{Z_1, Z_2\}$, the prediction for y_i is the average of the observations (values inside each rectangle in Fig. 2). Namely,

$$\hat{y}_{i \in R_J} = \frac{1}{N_J} \sum_{i \in R_J} y_i, \qquad (13)$$

where R_J is a region in the feature space and N_J is the number of data points in R_J. The regions R_J are determined by optimum splits in the feature space that leads to the smallest RSS. Formally, the following function is minimized to obtain the tree:

$$L = \sum_{t=1}^{T} \sum_{i \in R_t} (y_i - \hat{y}_i)^2 + \gamma \, T, \qquad (14)$$

where the first term is the RSS, and the second term acts as the regularization. T is the number of terminal nodes (or leaves) in a tree which coincide with the number of regions the feature space is split. The larger the number of terminal nodes T, the more complex the structure of $f(\mathbf{X})$ will be, which may lead to overfitting. The regularization term adds a penalty for complex trees to prevent overfitting. An example of a tree with 8 terminal nodes is shown in Fig. 4. As can

be seen, the regression tree is grown recursively. The largest reduction in RSS is obtained at the split $Z_1 = -3.31$, then in splits $Z_2 = -4.17$, $Z_2 = -45.3$, and so on.

While the regression tree algorithm is simple and has low computational cost, it has low predictive accuracy. The tree shown in Fig. 4 leads to a very rough description of the nonlinear behavior of E_{ps} as a function of the features, as can be seen in the regions of Fig. 2. An approach to more accurately describe the nonlinearities is to train an ensemble of trees and combine the predictions, also known as boosting.[19] The boosted tree algorithm starts with a null prediction ($\hat{y}_i^{(0)} = 0$) and sequentially train trees on the residuals from the previous tree. Namely,

$$\hat{y}_i^{(0)} = f_0(x_i) = 0, \ r_i^{(0)} = y_i,$$
$$f_1 : r_i^{(0)}, \ \hat{y}_i^{(1)} = \hat{y}_i^{(0)} + \eta \, f_1, \ r_i^{(1)} = r_i^{(0)} - \eta \, f_1(x_i),$$
$$\vdots$$
$$f_t : r_i^{(t-1)}, \ \hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta \, f_t, \ r_i^{(t)} = r_i^{(t-1)} - \eta \, f_t(x_i),$$
$$\vdots$$
$$f_R : r_i^{(R-1)}, \ \hat{y}_i^{(R)} = \hat{y}_i^{(R-1)} + \eta \, f_R, \ r_i^{(R)} = r_i^{(R-1)} - \eta \, f_R(x_i),$$

with the final prediction

$$f(\mathbf{X}) = \sum_{t=1}^{R} \eta \, f_t(\mathbf{X}), \qquad (15)$$

where $f_t : r_i^{(t-1)}$ denotes that the tree at iteration t is trained on the residuals from the prediction of the $(t-1)$th tree. The parameter η is known as shrinkage, which determines the weight of the predictions that are added on at each iteration. Both the shrinkage and the number of sequentially trained trees R are parameters to be determined by cross-validation. Fig. 5 illustrates the process of boosting trees with $\{Z_1, Z_2\}$ as features. While a single tree divides the feature space into rough regions (left panel in Fig. 5), with 100 trees (right panel of Fig. 5), the predicted E_{ps} values are almost indistinguishable to the eye from the original data (Fig. 2).

In this study, we use the computationally efficient and scalable implementation of the boosting algorithm XGBoost.[25] Apart from the shrinkage (η), number of trees (R), and the regularization term γ (Eq. (14)), XGBoost has several other parameters which are optimized to achieve highly accurate predictions. The parameters we have chosen to determine using cross-validation with their short description are listed in Table I, while a full description can be found in Ref. 25.

C. Neural networks

While neural networks are more popular for applications in classification problems, they can also be used for regression.[24] As input, neural networks take the feature vectors and using an activation function, nonlinear features are created and used for regression. The activation function connects the linear input through layers of neurons to the nonlinear output. The parameters used for connecting the layers in a neural network are obtained by optimizing the RSS (in regression setting) using the training data. A structure of a neural network with one (hidden) layer is shown in Fig. 6. Given the input
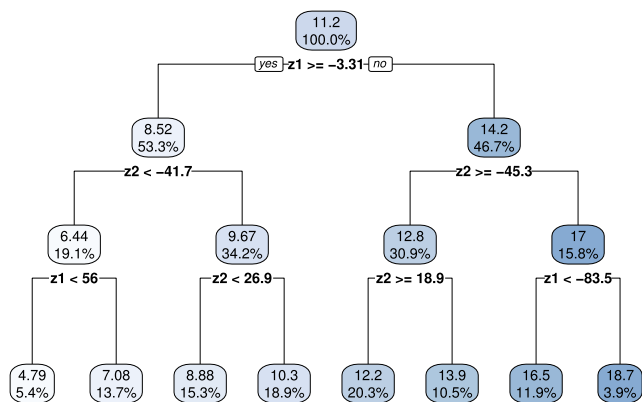

FIG. 4. Regression tree trained on the dataset with the first two principal components used as features. The blue boxes contain the value of the average value of E_{ps} as well as the percentage of data points in the region they belong to. These regions can also be seen in Fig. 2.
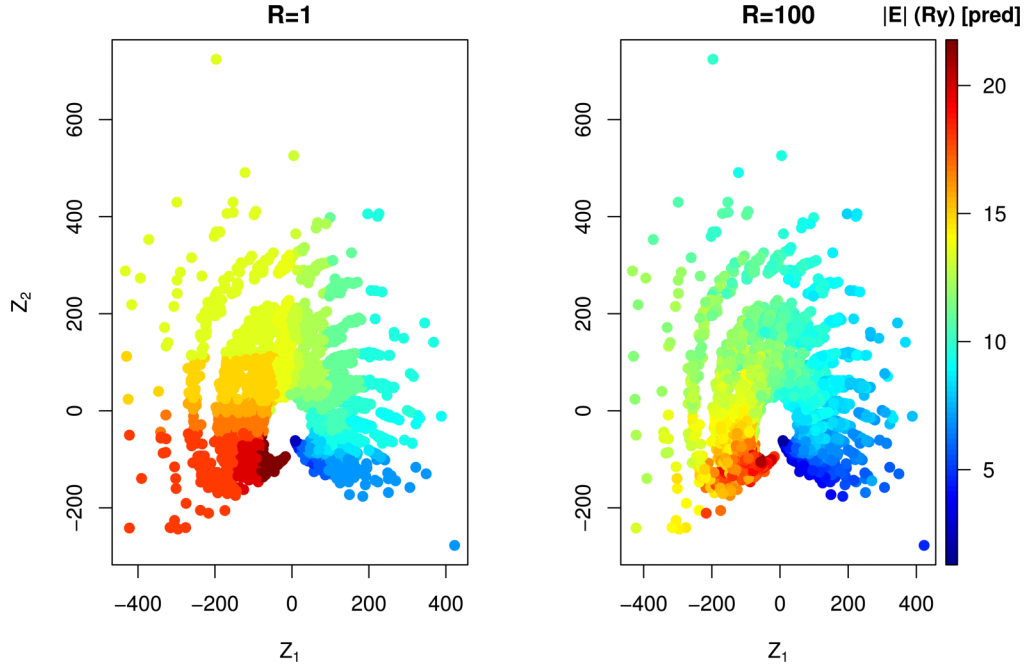
FIG. 5. Predicted E_{ps} using boosted regression trees for $R = 1$ (left panel) and $R = 100$ (right panel). While a single tree ($R = 1$) results in a rough division of the feature space, boosting many trees ($R = 100$) results in much higher accuracy.

vector of features x_i, a nonlinear output is obtained by the following equations:

$$\mathbf{a}^{(1)} \leftarrow \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{bmatrix}, \quad \mathbf{a}^{(2)} = g\left(\theta^{(1)} \cdot \mathbf{a}^{(1)}\right),$$

$$\mathbf{a}^{(2)} \leftarrow \begin{bmatrix} 1 \\ a_1^{(2)} \\ \vdots \\ a_h^{(2)} \end{bmatrix}, \quad \hat{\mathbf{y}} = (\theta^{(2)})^T \cdot \mathbf{a}^{(2)}. \tag{16}$$

In the first line, the vector $\mathbf{a}^{(1)}$ is constructed from the input features $\mathbf{x}$ by adding 1, which accounts for the constant term in regression. Then, the input vector $\mathbf{a}^{(1)}$ is transformed into a derived feature vector $\mathbf{a}^{(2)}$ in the hidden layer (with size h) using the nonlinear activation function $g(x)$. The transformation is parameterized by the coefficients $\theta_{ij}^{(1)}$ which are elements of a $(p + 1) \times h$ matrix. The output layer, which serves as the prediction vector $\hat{\mathbf{y}}$ (N dimensional), results from

a linear transformation via $\theta_j^{(2)}$ which is an N dimensional vector. The parameters $\theta^{(1)}$ and $\theta^{(2)}$ are obtained by minimizing Eq. (11) with the following regularization term:

$$R = \gamma \left[\sum_{i=1}^{p} \sum_{k=1}^{h} (\theta_{ik}^{(1)})^2 + \sum_{k=1}^{h} (\theta_k^{(2)})^2 \right]. \tag{17}$$

There are several choices for the activation function, but we adopt here the most widely used sigmoid function which is given by

$$g(\theta \cdot \mathbf{x}) = \frac{1}{e^{-\theta \cdot \mathbf{x}} + 1}. \tag{18}$$

TABLE I. XGBoost parameters.

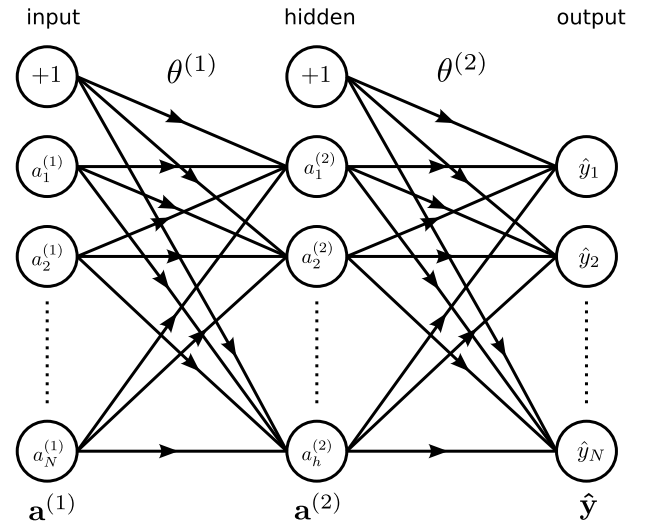| Parameter	Description
R	Number of trees grown
η	Shrinkage
γ	Regularization term
MD	Maximum number of terminal nodes (T) in a tree
CST	Subsample ratio of randomly chosen features while training a tree
MCW	Minimum number of data points in a region of feature space in each tree

FIG. 6. A schematic representation of single layer neural network. Each line represents a term that connects a feature ($\mathbf{a}^{(1)}$) to a derived feature ($\mathbf{a}^{(2)}$) or a derived feature ($\mathbf{a}^{(2)}$) to the output ($\hat{\mathbf{y}}$). The activation functions use the parameters $\theta^{(1)}$ between the input and hidden layer, while a linear transformation using $\theta^{(2)}$ is used between the hidden layer and output.

It is possible to obtain more complex nonlinear relationships between the features **x** and the output layer, by including more hidden layers. While more hidden layers may result in higher accuracy, because of the added computational cost, we have limited our study to single-hidden-layer neural networks. In fact, the computational cost of training even a single-layer neural network is much higher than a boosted regression tree, and its discussion is included for the sake of assessing the prediction accuracy of the latter method.

IV. RESULTS AND DISCUSSION

A. Model training and accuracy

Using the cross-validation techniques outlined in the Sec. III, we have trained both boosted regression trees and neural networks. Either the Coulomb matrices ($C_{IJ}^{(i)}$) or their eigenspectrum ($\lambda^{(i)}$) was used as features for training. The training set, 70% of the full dataset chosen randomly, is used to determine the parameters of the models. The parameters obtained for boosted regression trees using 5-fold cross-validation is shown in Table II. As an illustration, we show in Fig. 7 the results of the 5-fold cross-validation with $\lambda^{(i)}$ as features. Since the space of parameters to be optimized is multi-dimensional (Table I), we present root mean squared error (RMSE) (i.e., the square root of MSE in Eq. (12)), when MCW, CS, and γ are fixed to their optimized values from Table II.

For neural networks, a 5-fold cross-validation approach is used for training the model when $\lambda^{(i)}$ are used as features, while the validation set approach is used when $C_{IJ}^{(i)}$ are used

TABLE II. Parameters determined for boosted regression trees using 5-fold cross-validation for $C_{IJ}^{(i)}$ and $\lambda^{(i)}$ based features. Definitions of the parameters are given in Table I.

	R	η	γ	MD	CST	MCW
$\lambda^{(i)}$	600	0.0156	0.0	16	0.4	10
$C_{IJ}^{(i)}$	400	0.0625	0.0	6	0.2	10

as features due to heavy computational cost in this case. In the validation set approach, the initial training set (70% of the original dataset) is split further into a validation set (40% of initial training set) and a second level training set (60% of initial training set) randomly. The cross-validation results in $h = 25$ (size of the hidden layer) and $\gamma = 1.0$ (the regularization parameter) when $\lambda^{(i)}$ are used, while $h = 25$ and $\gamma = 0.1$ is obtained when $C_{IJ}^{(i)}$ are used. The resulting cross-validation (or validation set) and test errors, measured by RMSE, for the trained models are summarized in Table III. The best performance is obtained with the boosted regression tree algorithm when the Coulomb matrices $C_{IJ}^{(i)}$ are used as features. The use of the eigenspectrum $\lambda^{(i)}$ as features only slightly increases the RMSE, while the computational cost is much smaller. Therefore, it is reasonable to use the $\lambda^{(i)}$ instead of the full $C_{IJ}^{(i)}$. In Fig. 8 we show the difference between predicted and actual E_{ps} evaluated on the test set, using $\lambda^{(i)}$.

As we pointed out earlier, the boosted regression tree algorithm is computationally more efficient than neural networks. For example, with $\lambda^{(i)}$ as features, the elapsed time to train the boosted regression tree model (the model
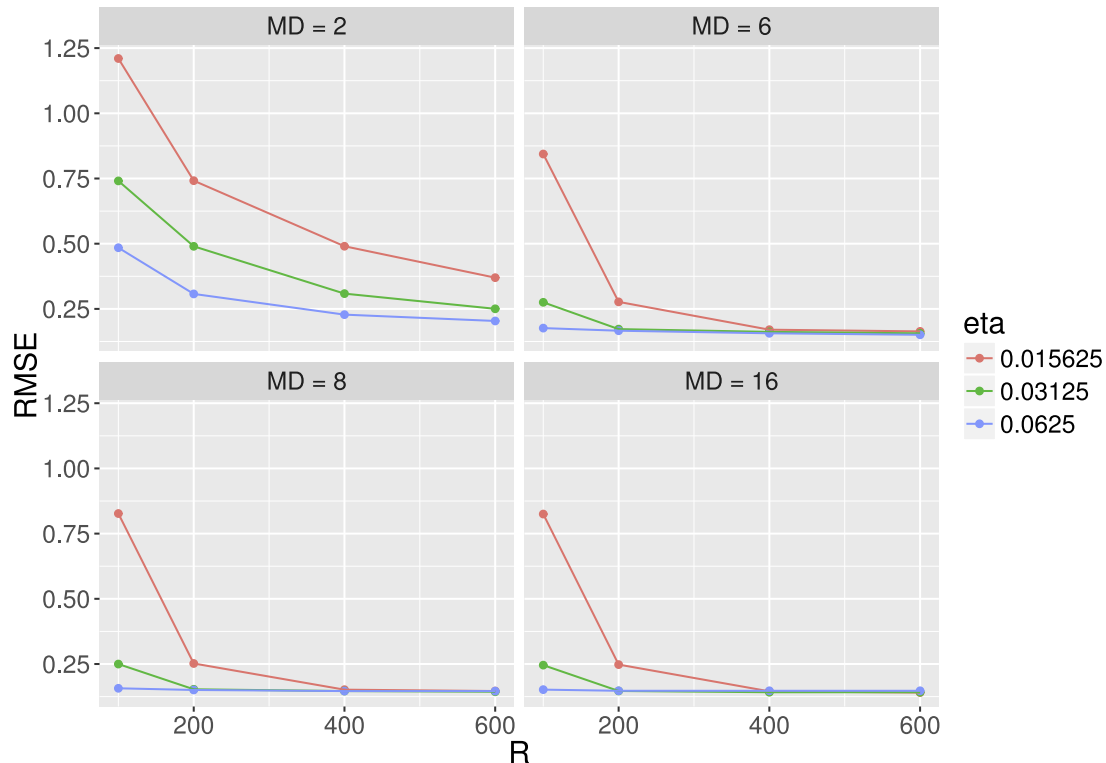

FIG. 7. RMSE (in Ry) from 5-fold cross-validation for boosted regression trees for $\lambda^{(i)}$ used as predictors as a function of number of trees (R) and maximum number of trees (MD). Some of the parameters are fixed in the plots: MCW = 10, CS = 0.4, $\gamma = 0.0$.

TABLE III. 5-fold CV (validation set in case of neural network with $C_{IJ}^{(i)}$) and test RMSE in kcal/mol. The difference between 5-fold CV and test RMSE is smaller than that of validation set and test RMSE as expected, since 5-fold CV reduces the dependence of RMSE to the way the data is split.

Feature	Method	5-fold CV (or validation set) error	Test error
$\lambda^{(i)}$	Boosted tree	44.09	41.81
	N. network	62.15	60.06
$C_{IJ}^{(i)}$	Boosted tree	38.75	36.63
	N. network	45.25	58.39

determined by cross-validation whose parameters are listed in Table II) is 12.1 s (on a laptop with 8 CPU cores). Instead, the elapsed time to train the neural network model (the model with $h = 25$ and $\gamma = 1.0$ determined by cross-validation) is 258.2 s. In case of the elapsed time to obtain the fit to the testing data (after the models are trained), the neural network model takes 0.02 s, while the boosted regression tree model takes 0.01 s. In case of $C_{IJ}^{(i)}$ as features, the training times are 17.3 s for the boosted regression tree and 16 917.2 s for the neural network, while the fitting times to the test data are 0.01 and 0.2 s, respectively. While the fitting time for neural networks are almost the same as boosted regression trees, the vast difference between training times, in addition to increased accuracy, makes boosted regression tree the preferred algorithm.

Previous studies which analyzed the GDB database[26,27] of molecules have found much smaller test RMSE for their best achieving learning methods.[16,23] For example, Ref. 16 reported a RMSE of 20.29 kcal/mol for multi-layer neural networks and 13.18 kcal/mol for a nonlinear kernel method when $\lambda^{(i)}$ were used as features. These are smaller values of RMSE compared to our values of 41.81 and 60.06 kcal/mol for boosted trees and single layer neural networks, respectively. The main difference is due to the fact that the variability in the GDB database is much smaller than our dataset which is based on the PubChem data. Ref. 16 used a subset of the

GDB database that contains 7165 molecular structures which has maximum 23 atoms per molecule. The reported standard deviation of the atomization energies (RMSE when mean value of **y** is used as the prediction) was 223.92 kcal/mol. In our study, we have 16 242 structures with a maximum of 50 atoms per molecule, while the standard deviation of the atomization energies is 1147.89 kcal/mol, which indicates that the dataset used in this study encompasses a larger range of molecules. As a result of the larger variance in the training data (almost five times of Ref. 16), the learning algorithms result in higher RMSE values. Notice also that the number of molecules in this study is much larger, leading to a computationally more expensive model training when a method like neural network regression is used. It is possible to obtain much better accuracies by including more data with the use of the full PubChem database, instead of the first 75 000 entries as we did in this work. In addition, using several copies of the same molecule by representing them via randomly re-ordered Coulomb matrices (a method introduced in Ref. 16 to address the uniqueness problem of $C_{IJ}^{(i)}$) would reduce the variance in the dataset, leading to better accuracies. While the reported RMSE values are higher than what would be desired (e.g., the accuracy of a few kcal/mol), the inclusion of more data presents a clear path to reach more accurate models. As an example, we have tested the accuracy of the randomly re-ordered Coulomb matrices and found that with only 4 random re-orderings included (yielding a training set four times larger than original), the RMSE of 36.63 kcal/mol (Table III) reduces to 27.74 kcal/mol. It is also possible to use an ensemble method,[28] where the predictions of boosted trees and neural networks are combined. In each method, the worst predictions are on different molecules, and by combining the two on the regions where they perform the best, improved accuracies can be obtained. Another approach has been proposed in a more recent work, where feature learning increased prediction accuracies.[29] While all of these approaches would lead to increased accuracy, they also come with added computational cost and are beyond the scope of this work. Therefore, we leave these tasks for a future study.

B. Predictions on independent datasets

In Subsection IV A, we have trained models to predict E_{ps} based on a dataset of molecules made up of elements in the CHNOPS set. Namely, each molecule contains at least one of the atoms in the set $\{C, H, N, O, P, S\}$. While a random division of the dataset into training and test sets allowed us to evaluate the accuracy of the models, another possible measure for performance would be based on a set of molecules outside the CHNOPS set. For this purpose, we have constructed two distinct datasets in order to test the models trained in the previous subsection: (i) Cl-set, which contains at least one Cl atom in addition to at least one of CHNOPS, and (ii) Si-set, which contains at least one Si atom in addition to at least one of CHNOPS. From the first 75 000 entries in the PubChem database, we have found that there are 159 molecules in the Si-set and 4114 in the Cl-set. Similar to the original CHNOPS set, each molecule in both Si and Cl sets has a maximum
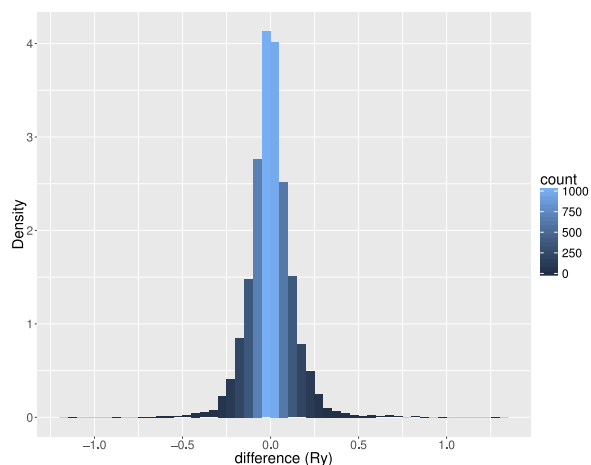

FIG. 8. The difference between actual and predicted E_{ps} on the test set. Predictions are made using the boosted regression tree with $\lambda^{(i)}$ as features. Color coding represents the number of molecules that fall into each bin in the histogram.

TABLE IV. Test RMSE (kcal/mol) on the Cl and Si sets. Predictions are based on the boosted regression tree algorithm trained on the CHNOPS set.

Set	RMSE ($\lambda^{(i)}$)	RMSE ($C_{IJ}^{(i)}$)
Cl	75.18	114.20
Si	89.13	86.53

of 50 atoms per molecule, and even number of electrons. Using the boosted regression trees we have trained in the previous subsection, we predict E_{ps} in Cl and Si sets, for which the results are presented in Table IV. Since the models are not trained with molecules comprising of Si or Cl, this test assesses the applicability of our method when predictions on molecules with new elements need to be obtained. As expected, the RMSE values are higher for the Cl and Si sets, than that of the test errors reported in Table III. Therefore the models can only be used as exploratory tools when predictions on completely separate datasets are needed.

V. CONCLUSIONS

In this work, we have proposed the use of boosted regression trees as a higher accuracy and computationally much more efficient alternative to some other machine learning methods proposed for electronic structure prediction. We have tested the performance of boosted regression trees using the PubChem database, and shown that it outperforms single-layer neural networks in predicting ground state energies (equivalently E_{ps}). Due to the ability to grow many trees in parallel, boosted regression trees are much faster than neural networks, which require large matrix operations. We have also shown that the trained algorithms can be used for the predicting electronic structure of molecules containing elements other than the ones included in the training set. While the prediction accuracy is reduced in this case, the method is still applicable for exploratory studies.

Machine learning techniques provide a rich possibility of applications for quantum mechanical simulations in computational chemistry and materials science.[30] In fact, there has been several other compelling applications of learning algorithms for predicting electronic properties other than atomization energies for molecules,[23] prediction of electronic structure of solids,[31,32] finding DFT functionals[33] and determining potential energy surfaces.[34–36] The adoption of boosted regression trees for the learning method, as proposed here, would reduce the cost of model training compared to computationally heavier algorithms like neural networks and support vector regression, without sacrificing, and possibly increasing prediction accuracies.

With the ability to predict electronic properties without performing new simulations for each molecule, machine learning techniques open up exciting pathways for rational design of new compounds. Combined with numerous efforts to catalog and standardize datasets, these methods will be invaluable for many scientific and technological applications.

[1]A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson, APL Mater. **1**, 011002 (2013).

[2]S. Curtarolo, W. Setyawan, G. L. W. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M. J. Mehl, H. T. Stokes, D. O. Demchenko, and D. Morgan, Comput. Mater. Sci. **58**, 218 (2012).

[3]G. L. W. Hart, S. Curtarolo, T. B. Massalski, and O. Levy, Phys. Rev. X **3**, 041035 (2013).

[4]G. Hautier, C. C. Fischer, A. Jain, T. Mueller, and G. Ceder, Chem. Mater. **22**, 3762 (2010).

[5]G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky, Comput. Mater. Sci. **111**, 218 (2016).

[6]P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

[7]W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[8]J. Behler, Phys. Chem. Chem. Phys. **13**, 17930 (2011).

[9]L. Hu, X. Wang, L. Wong, and G. Chen, J. Chem. Phys. **119**, 11501 (2003).

[10]R. M. Balabin and E. I. Lomakina, J. Chem. Phys. **131**, 074104 (2009).

[11]J. Sun, J. Wu, T. Song, L. Hu, K. Shan, and G. Chen, J. Phys. Chem. A **118**, 9120 (2014).

[12]R. M. Balabin and E. I. Lomakina, Phys. Chem. Chem. Phys. **13**, 11710 (2011).

[13]J. Lee, A. Seko, K. Shitara, K. Nakayama, and I. Tanaka, Phys. Rev. B **93**, 115104 (2016).

[14]In practice, the situation is complicated further by the choice of pseudopotentials, basis sets, kinetic energy cut-offs, boundary conditions (especially for molecular systems) and the simulation code.[37]

[15]M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, Phys. Rev. Lett. **108**, 058301 (2012).

[16]K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, J. Chem. Theory Comput. **9**, 3404 (2013).

[17]J. Hill, G. Mulholland, K. Persson, R. Seshadri, C. Wolverton, and B. Meredig, MRS Bull. **41**, 399 (2016).

[18]S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang, and S. H. Bryant, Nucleic Acids Res. **44**, D1202 (2015).

[19]J. H. Friedman, Ann. Stat. **29**, 1189 (2001).

[20]P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. Chiarotti, M. Cococcioni, I. Dabo *et al.*, J. Phys.: Condens. Matter **21**, 395502 (2009).

[21]J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).

[22]D. Vanderbilt, Phys. Rev. B **41**, 7892 (1990).

[23]G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, New J. Phys. **15**, 095003 (2013).

[24]J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning* (Springer series in statistics Springer, Berlin, 2001).

[25]T. Chen and C. Guestrin, preprint arXiv:1603.02754 (2016).

[26]T. Fink, H. Bruggesser, and J.-L. Reymond, Angew. Chem., Int. Ed. **44**, 1504 (2005).

[27]T. Fink and J.-L. Reymond, J. Chem. Inf. Model. **47**, 342 (2007).

[28]J. H. Friedman and B. E. Popescu, Ann. Appl. Stat. **2**, 916 (2008).

[29]H. Dai, B. Dai, and L. Song, preprint arXiv:1603.05629 (2016).

[30]M. Rupp, Int. J. Quantum Chem. **115**, 1058 (2015).

[31]K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K. R. Müller, and E. K. U. Gross, Phys. Rev. B **89**, 205118 (2014).

[32]F. Faber, A. Lindmaa, O. A. von Lilienfeld, and R. Armiento, Int. J. Quantum Chem. **115**, 1094 (2015).

[33]J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, Phys. Rev. Lett. **108**, 253002 (2012).

[34]S. Lorenz, M. Scheffler, and A. Gross, Phys. Rev. B **73**, 115431 (2006).

[35]S. Lorenz, A. Gross, and M. Scheffler, Chem. Phys. Lett. **395**, 210 (2004).

[36]A. Khorshidi and A. A. Peterson, Comput. Phys. Commun. **207**, 310 (2016).

[37]K. Lejaeghere *et al.*, Science **351**, 1415 (2016).