Machine Learning in Materials Chemistry: An Invitation

Packwood, Daniel

Nguyen, Linh Thi Hoai

Cesana, Pierluigi

Zhang, Guoxi

他

出版情報：Machine Learning with Applications. 8, pp.100265-, 2022-06. Elsevier
バージョン：

Machine Learning with Applications 8 (2022) 100265

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Machine Learning in Materials Chemistry: An Invitation

Daniel Packwood [a], Linh Thi Hoai Nguyen [b], Pierluigi Cesana [b], Guoxi Zhang [a,c], Aleksandar Staykov [d], Yasuhide Fukumoto [b], Dinh Hoa Nguyen [b,d,*]

[a] Institute for Integrated Cell-Material Sciences (WPI-iCeMS), Kyoto University, Kyoto, Japan
[b] Institute of Mathematics for Industry (IMI), Kyushu University, Motooka 744, Nishi-ku, Fukuoka 819-0395, Japan
[c] Graduate School of Informatics, Kyoto University, Japan
[d] International Institute for Carbon-Neutral Energy Research (WPI-I^{2}CNER), Kyushu University, Motooka 744, Nishi-ku, Fukuoka 819-0395, Japan

ARTICLE INFO

Keywords:
Materials chemistry
Kernelized machine learning
Density functional theory
Bayesian optimization
Ensemble methods
Reinforcement learning
Federated learning

ABSTRACT

Materials chemistry is being profoundly influenced by the uptake of machine learning methodologies. Machine learning techniques, in combination with established techniques from computational physics, promise to accelerate the discovery of new materials by elucidating complex structure–property relationships from massive material databases. Despite exciting possibilities, further methodological developments call for a greater synergism between materials chemists, physicists, and engineers on one side, with computer science and math majors on the other. In this review, we provide a non-exhaustive account of machine learning in materials chemistry for computer scientists and applied mathematicians, with an emphasis on molecule datasets and materials chemistry problems. The first part of this review provides a tutorial on how to prepare such datasets for subsequent model building, with an emphasis on the construction of feature vectors. We also provide a self-contained introduction to density functional theory, a method from computational physics which is widely used to generate datasets and compute response variables. The second part reviews two machine learning methodologies which represent the status quo in materials chemistry at present – kernelized machine learning and Bayesian machine learning – and discusses their application to real datasets. In the third part of the review, we introduce some emerging machine learning techniques which have not been widely adopted by materials scientists and therefore present potential avenues for computer science and applied math majors. In the final concluding section, we discuss some recent machine learning-based approaches to real materials discovery problems and speculate on some promising future directions.

Contents

* Corresponding author at: International Institute for Carbon-Neutral Energy Research (WPI-I2CNER), and Institute of Mathematics for Industry (IMI), Kyushu University, Motooka 744, Nishi-ku, Fukuoka 819-0395, Japan.
E-mail addresses: dpackwood@icems.kyoto-u.ac.jp (D. Packwood), linh@imi.kyushu-u.ac.jp (L.T.H. Nguyen), cesana@imi.kyushu-u.ac.jp (P. Cesana), guoxi@ml.ist.i.kyoto-u.ac.jp (G. Zhang), alex@i2cner.kyushu-u.ac.jp (A. Staykov), yasuhide@imi.kyushu-u.ac.jp (Y. Fukumoto), hoa.nd@i2cner.kyushu-u.ac.jp (D.H. Nguyen).

https://doi.org/10.1016/j.mlwa.2022.100265
Received 23 September 2021; Received in revised form 9 December 2021; Accepted 23 January 2022
Available online 14 February 2022

1. Introduction

Materials science, like most fields of science, is undergoing a profound transformation due to the infusion of machine learning (ML) techniques. A quick search for "materials science machine learning" on the Web of Science reveals that nearly 2000 papers were published on this topic in 2020 alone, compared to only around 400 papers in 2017. While the exact cause of this excitement is difficult to trace, it likely reflects the maturation of computational physics methodology combined with the increasing availability of cheap computational resources. The influence of high-profile international projects such as the Materials Genome Initiative[1] and the NOMAD (Novel Materials Discovery) Consortium in Europe[2] has also been significant, articulating a clear and compelling vision for ML in materials science. Broadly speaking, this vision involves the extraction of complex structure–property relationships from material databases and predicting new materials based on them. At present, the majority of these materials-ML endeavors are being led by materials scientists themselves, and there are high demands for computer science and math majors to join these projects as collaborators. On the other hand, materials science is a specialized area demarcated by a highly developed vocabulary and methodology, and is not always accessible to researchers of other fields.

That said, materials science is not a well-defined field. Broadly speaking, it consists of a loose consortium of chemists and physicists (and, increasingly, biologists and engineers) who aim to create new types of materials (molecules, crystals, glasses, and so on) with useful physical properties. Much effort in materials science aims to understand the connection between material structure and material properties. Moreover, depending on how material structure is defined, two major flavors of materials science can be identified. One of these flavors, which we refer to as *materials engineering*, treats materials as continuous media. It defines structure in terms of the shape of the material, as measured on micrometer or larger scales, and is interested in how shapes and constitutive parameters (such as elastic moduli) affect physical properties. As such, it is well suited for studying solid materials such as crystals or glasses, with photonic crystals being a representative topic (Yang et al., 2021). The other flavor of materials science might be referred to as *materials chemistry*. It defines structure in terms of the spatial arrangement of atoms inside of a material, and explores how this atomic arrangement affects the properties of the material. Materials chemistry also studies crystals and other types of solid materials, however, in contrast with materials engineering, it also considers molecules themselves as potential materials.

While both flavors of materials science have enthusiastically adopted ML, the focus of this review is materials chemistry. Materials chemistry, being steeped in a mixture of quantum physics methodology and chemical intuition, is arguably more opaque towards computer science or math majors despite rich potential for spectacular applications. Indeed, recent trends towards nanoscale miniaturization in materials chemistry have called for greater orchestration with other scientific disciplines, including computer science and mathematics. Through impressive collaborations involving atom-scale engineering, computation, and mathematical modeling, sophisticated architectures such as nanomachines (Aprahamian, 2020; Balzani et al., 2000; Coskun et al., 2012) obtained by assembling and maneuvering molecules acting as (components of) small motors (Koumura et al., 1999; Kudernac et al., 2011) and shuttles have been reported (Bissell et al., 1994; Collin et al., 2001), as well as tiny shape-memory alloy structures capable of performing actuation in the wake of an elastic transformation of their lattice (Bhattacharya & James, 2005). This emerging research direction epitomizes the idea that the material or a molecule itself can act as a machine, a concept first envisioned by Feynman in the 1960s (Feynman, 1960a, 1960b), and was recognized with the 2016 Nobel Prize

[1] https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/materials_genome_initiative-final.pdf
[2] https://nomad-coe.eu/about/consortium

in Chemistry.[3] This research direction, which involves judicious application of structure–property relationships coupled with sophisticated experimental techniques for controlling the spatial positions of atoms and molecules, requires a breadth of skill far exceeding those found within isolated chemistry or physics departments alone; it demands the skill of computer science and math majors to develop powerful models for predicting material structure–property relationships.

In recent years, the development of materials chemistry has occurred in tandem with that of density functional theory (DFT), a technique from computational physics. Given the atomic structure of a material (that is, the spatial coordinates of its atoms), DFT computes the energy and various other physical properties of the material using the equations of quantum physics. While DFT's origins can be traced to the early days of quantum physics in the 1920s, the theory developed predictive power only from the 1990s onwards. Since then, DFT has gained an enormous following, a point vividly illustrated by the fact that, as of 2014, 12% of the highest-cited papers on physical sciences in the entire scientific literature were related to DFT, two of which were even in the top 10 (Van Noorden et al., 2014). Needless to say, DFT has had spectacular success in predicting and explaining certain experimental measurements, and has on occasion succeeded to predict entirely new classes of materials (such as topological insulators, now one of the major topics in materials chemistry (Kong & Cui, 2011)). While the accuracy of DFT is not perfect when compared to experiments (especially when predicting certain crystal properties such as band gaps, or when dealing with materials involving clusters of molecules), its predictions are usually reliable enough to assist materials chemists. While DFT itself is far from simple, the recent availability of user-friendly software has enabled DFT calculations to be performed routinely, giving it something of a black-box quality.

Until about 2010, DFT was mainly used to investigate the properties of individual materials in detail. However, from about 2010 onwards, a new trend emerged: the use of DFT to calculate properties for thousands or more different materials in sequence, thereby generating large databases of reliable, computation-derived data. Numerous databases of DFT-derived data now exist online, including the NOMAD database,[4] the Materials Project database (Jain et al., 2013b), the Quantum-Machine.org database (Blum & Reymond, 2009; Rupp et al., 2012), just to name a few, and private DFT databases are now routinely generated by companies and academic research groups (see (Himanen et al., 2019) for a list of major materials databases and infrastructures, as well as Table 2 of this review). This combination of sequential DFT calculations, as well as subsequent analysis, is commonly referred to as high-throughput computational materials science. Despite such high-throughput approaches hold great potential for discovering new materials with novel properties and functionality, they are time-consuming and can demand enormous computational resources. Additionally, the deluge of DFT-derived data has resulted in the paradoxical situation of "drowning in information and starving for knowledge" (Wilson, 1998). In other words, while materials chemistry data may be abundant, it is extremely difficult to extract useful conclusions from such databases. Small changes in atomic structure can lead to profound changes in material properties, making the relationship between structure and properties immensely complex even for seasoned chemists and physicists.

It is here where ML enters the scope of materials chemistry. As far as materials chemistry is concerned, ML is a collection of methodologies for learning the associations between patterns (atomic structure) and numbers (material properties such as energy, electrical conductivity, etc.) (Mohri et al., 2012). Over the last ten years, ML has been enthusiastically embraced by much of the materials chemistry community as a potential means to extract useful information from high-throughput calculations. In early applications during the period 2012 to 2016, ML demonstrated an impressive ability to learn DFT-calculated properties such as atomization energy from molecule and crystal datasets (Hansen et al., 2013; Rupp et al., 2012; Seko et al., 2015). Following these papers a tsunami of similar works was published, confirming the impressive ability of ML to extract structure–property relationships from material databases of various kinds (see (Chibani & Coudert, 2020) for a review). Some groups have deployed these ML-derived relationships into other simulation methods to reduce computational costs and efforts (Hörmann et al., 2019; Packwood, 2017). While these research lines continue to a considerable extent, more recent research trends have aimed to extract intelligible chemical insights from databases (Schwaller et al., 2021) and also to predict new materials for synthesis on the basis of machine-learned relationships (Antono et al., 2020; Gu et al., 2019). Other research groups have gone even further, incorporating such predictive ML into sophisticated robotics which can synthesize the predicted materials in an experimental laboratory (Burger et al., 2020; Shimizu et al., 2020).

While ML is reaching fruition in materials chemistry as a method for understanding DFT-derived datasets and predicting new materials, numerous problems remain. For example, the choice of feature vectors (descriptors) for encoding atomic structure is known to have a significant impact on the accuracy of ML predictions, however there is no systematic framework for constructing them. In addition, certain types of ML methodology are better suited for certain types of datasets than others. For some datasets, standard ML methods seem to fail completely. At present, it is often unclear *a priori* why some datasets are amenable to certain methods while others are not. Moreover, the range of ML methodology currently used by materials chemists is rather narrow. Thus, there remains plenty of scope for computer science and math majors to contribute their expertise to materials chemistry, particularly in the domain of method development, characterization, and application.

This paper provides a review and tutorial for ML of materials chemistry datasets. While the materials literature is replete with reviews on ML (see e.g., (Butler et al., 2018; Correa-Baena et al., 2018; Morgan & Jacobs, 2020; Ramprasad et al., 2017; Schleder et al., 2019; Schmidt et al., 2019) and (Mueller et al., 2016, Chapter 4), most of them are written for a materials chemistry audience. The present review is intended for computer science and math majors who are interested in collaborating with materials chemists or working with materials chemistry datasets. The present authors have backgrounds in applied mathematics and have worked in materials chemistry to varying degrees. We have experienced the difficulties of entering this field first-hand. In writing this review we have put ourselves in the shoes of a young computer science or math major and have asked ourselves what such a person would want out of a review such as this one. Presumably, they would have taken chemistry and physics classes at high school, and would be familiar with the basic concepts of material structure (atoms, chemical bonds, electrons, and so on). They might therefore wish to know specifically how these concepts are used when building ML models. They would probably wish to know how materials chemistry data are obtained, the extent to which it is reliable, and which distributional assumptions can be applied when dealing with it. If they are new to the field then they would probably prefer for the review to broadly describe the status quo, while keeping tangential discussions on novel or specialized developments to a minimum; on the other hand they would probably wish to know whether there are any emerging topics to which they could make a unique contribution. They probably prefer for the review to focus on one simple class of materials, avoiding jarring digressions on solid-state physics, molecular biology, etc., whenever new classes of materials are introduced. They would probably prefer for the review to be principled and self-contained, with the major ML methods presented as a consequence of a set of axioms within an abstract framework, rather than as pre-existing computer algorithms. This wish list is quite different from that of a young materials science major, and while the existing reviews in the materials

[3] https://www.nobelprize.org/prizes/chemistry/2016/press-release/
[4] https://nomad-coe.eu/about/consortium

literature describe similar methods and applications to the present one, none are specifically tailored to these needs.

The outline of this paper is as follows. After reviewing the typical flow of ML in materials chemistry (Section 2), we describe DFT, the primary method by which materials chemistry datasets are obtained. Our introduction to DFT puts emphasis on the approximations involved and gives general guidance towards good practice (Section 3). Section 4 is dedicated to the construction of feature vectors from materials datasets. Feature vectors are the mathematical objects which carry materials structure information (atoms, chemical bonding information, etc.) into the ML model. We review both the classical, well established methodologies as well as some of the most recent approaches. Following this, we review several kernelized ML methodologies (Section 5). While these techniques are described in detail in ML textbooks, they are reviewed here because they are the dominant ML methods used in materials chemistry at present and represent the status quo. Our review does not cover neural networks or deep learning. Although these methodologies are attracting growing interest from materials scientists of all types, the literature landscape remains fragmented making it challenging to provide comprehensive summary at this stage. The reader is referred to (Ko et al., 2021; Li et al., 2019; Tsubaki & Mizoguchi, 2020) for representative applications of neural networks to this area. We introduce in Section 6 several emerging ML methods including ensemble learning, reinforcement learning and federated learning. These techniques have received relatively little attention in materials chemistry despite the potential for impressive results, and are areas where computer science and math majors could make useful contributions. Section 7 gives an illustration of the application of various ML methods on a dataset. We conclude (Section 8) with future perspectives on potential avenues for collaboration between materials chemists and computer scientists and mathematicians.

2. The flow of ML for materials chemistry

In materials chemistry, a typical ML project begins with a list of candidate materials. These materials could be molecules, crystals, glasses, monomer units of a polymer chain, or something else. For pedagogical reasons we will assume without significant loss that these materials are molecules. Some examples of molecule structures are shown in Fig. 1-A. In later sections we will consider examples where the candidate materials are clusters of molecules. For the case where these materials are molecules, at a minimum our list would contain the chemical or structural formula for the molecules but little (if any) other information. An example of such a list is shown in Fig. 1-B. To the newcomer, it may not be obvious how to build ML models from such a list, let alone how to predict molecules which optimize some physical property. It may not even be clear how molecules are represented in this list.

In this section we briefly explain the process of transforming the list of candidate molecules into a format useful for ML. This process is described in Fig. 1-C. The two major steps – obtaining the response variables and constructing the feature vectors – are discussed in detail in Sections 3 and 4, respectively.

2.1. The list of candidate molecules and chemical representations

In an ideal situation, each molecule in the list of candidates would be represented by its three-dimensional structure. The structures shown in Fig. 1-A are three-dimensional structures and can be represented as an $n \times 3$ matrix of coordinates and an n-dimensional vector of atom types, where n is the number of atoms in the molecule. If bonding information is desired, then an additional matrix describing connectivity between atoms is required as well. However, it is unlikely that we would be provided with a list of matrices and vectors directly, as it may consume an enormous amount of storage space if the number of molecules is large.

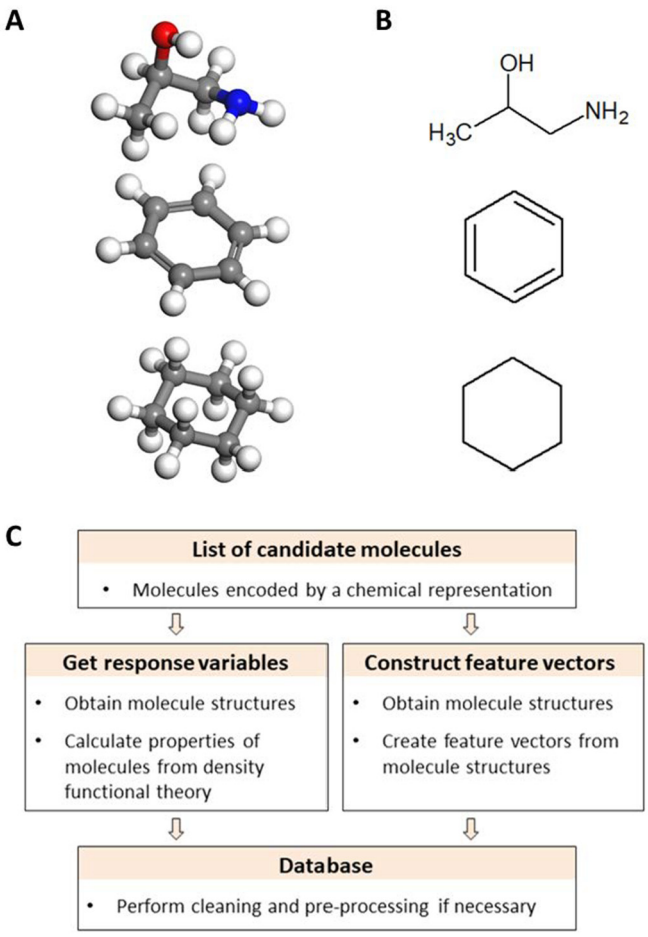


Fig. 1. (A) Three-dimensional molecule structures for aminopropan-2-ol (top), benzene (middle), and cyclohexane (bottom). White, gray, red, and blue balls represent hydrogen, carbon, oxygen, and nitrogen atoms, respectively. Sticks represent chemical bonds. (B) Structural formulas for the molecules shown in (A). Structural formulas are a type of chemical representation. (C) Flow diagram for preparing a database from a list of molecules.

It is therefore more common for lists of molecules to be compressed using some kind of *chemical representation*. A chemical representation reduces the dimensionality of the three-dimensional molecular structure into a chemically meaningful format while retaining as much chemical information as possible. Typical chemical representations include chemical formulas, structural formulas, SMILES, and SMARTS.

A *chemical formula* is arguably the simplest representation of a molecule. It consists of strings of characters describing the number of each type of atom in a molecule, in rough correspondence with their position in the three-dimensional atomic structure. For example, the chemical formula for the molecule propanol is represented as $CH_3CH_2CH_2OH$. While compact, considerable structural information is often lost in the chemical formula. For example, the cyclohexane and benzene molecules have chemical formulas C_6H_{12} and C_6H_6, respectively. Both of these molecules consist of a ring of six carbon atoms, however this fact may be unclear from their chemical formulas unless shown to a trained chemist. That said, the relationship between chemical formula and molecular structure is not unique, and there are cases where even a trained chemist would not be able to determine a molecule's three-dimensional structure from its chemical formula alone.

A *structural formula* is a two-dimensional projection of the three-dimensional structure of the molecule. Examples of structural formulas are shown in Fig. 1-B. In these formulas, atoms are represented by their symbols (such as O or Cl), and chemical bonds between atoms

are represented by lines. Double and triple bonds between atoms are represented by double and triple lines, respectively. The symbol for carbon (C) is typically not written explicitly, and is instead represented as a kink in the bond. Hydrogen atoms are typically ignored. Thus, the cyclohexane molecule is represented as a hexagon, whereas the benzene molecule is represented as a hexagon with alternating single and double lines. Plenty of examples can be found in elementary chemistry textbooks.

Compared to chemical formulas, structural formulas retain considerably more structural information, and a trained chemist can usually visualize the three-dimensional atomic structure simply by glancing at its structural formula. Indeed, for all practical purposes the relationship between structural formula and three-dimensional molecular structure is unique and can usually be recovered by using computational physics techniques (as described in more details in Section 3). However, structural formulas are not an ideal format for storing molecule structures; they must be stored in an image format, which may consume space if many molecules are present.

SMILES[5] is a relatively new kind of chemical representation which strikes a balance between chemical formulas and structural formulas. Like chemical formulas, SMILES describes the molecule using a short ASCII string. For example, melatonin ($C_{13}H_{16}N_2O_2$) is represented by the string CC(=O)NCCC1=CNc2c1cc(OC)cc2. However, compared to chemical formulas, SMILES retain considerably more information on the three-dimensional atomic structure. It is usually possible to generate a structural formula for a molecule from its SMILES string, providing that the molecule's structure is not particularly unusual. For details on the construction of SMILES strings, the reader is referred to (Weininger, 1988, 1990; Weininger et al., 1989). A similar chemical representation to SMILES is *SMARTS*[6] which uses ASCII strings for molecules based upon their substructural patterns. For example, the aliphatic amines can be written as the recursive SMARTS [([NH_2][CX_4]), ([NH]([CX_4])[CX_4]), ([NX_3]([CX_4])([CX_4])[CX_4])].

2.2. Obtaining the response variables and constructing feature vectors

Having clarified how molecule structures are represented in the list, we turn to the question of building a ML model. In a supervised ML setting, we need to construct a model of the form $y = f(\mathbf{x})$, where y is referred to as the *response variable* (possibly a vector) and $\mathbf{x}$ as the predictor variables. In a materials chemistry setting the response variable y corresponds to the physical property of interest (or properties of interest, in the vectorial case). The predictor variables are elements of a real-valued vector $\mathbf{x}$ which encodes the structure of the molecule in some way. $\mathbf{x}$ is the mathematical object that carries material structure information into the ML model. Materials chemists usually refer to $\mathbf{x}$ as a *feature vector*, and its entries as *features* or *descriptors*. In most applications of ML, the dimensionality of $\mathbf{x}$, as well as the definition of each feature, remains fixed for each molecule in the dataset.

Before building the ML model, two independent steps must be performed. In one of these steps, a subset of molecules must be selected, and for each of these molecules the response variable must be obtained. This subset becomes the dataset on which the ML model will be trained and tested. In most materials chemistry projects, the response variables are usually obtained using DFT. DFT is a well-established technique from computational physics which can compute the physical properties of molecules with good accuracy when compared to experimental measurements. Since such experimental measurements are often difficult to obtain, they are often substituted by DFT-calculated physical properties. DFT-derived data also allow us to make convenient distributional assumptions on the data. While DFT is now a fairly routine technique, it can be time-consuming. The expense of DFT implies limits to the size of the dataset. The calculation of response variables using DFT is described in detail in Section 3.

In the other step, we must decide upon a set of features to describe the molecules with. Starting from the three-dimensional structure of a molecule (obtained from its chemical representation), features are usually created considering which structural variables determine the physical property of interest, on the basis of chemical-domain knowledge (for example, inter-atomic distances are known to have a strong influence on molecule energies). While there is no systematic method for constructing features for molecules, a number of strategies has been developed. We discuss several of these in detail in Section 4.

2.3. Basic workflow for building ML models

Having determined a dataset of molecules from the list of candidates, having calculated the response variables for those molecules with DFT, and having determined the features to describe each molecule, we are almost in a position to apply ML and construct a predictive model.

The basic workflow for building ML models on the basis of supervised learning can be broken down as follows.

1. Data pre-processing
2. Feature selection
3. Model selection
4. Fitting the model to the training data
5. Using the model to predict values of the response variables for the test data.

In the remainder of this section, we introduce some techniques used for the first three steps mentioned above.

Data pre-processing

Typically a pre-processing step is needed before proceeding. This can be accomplished via elementary statistical analysis of the data, such as computing z-scores to detect outliers which should be re-examined or removed. While not always essential, it is sometimes desirable to transform the data in order to improve the predictive performance of the ML model. Such transformations typically involve conversion of categorical features into digital strings, normalization so that different features are measured on a comparable scale, and dimensionality reduction, in which the feature vectors are replaced with ones of lower dimension. Both normalization and dimensionality reduction can be performed using textbook methods.

Feature selection

Feature selection is the process of reducing the number of predictor variables (features) when developing a predictive model. It helps to reduce the computational costs of training models and can improve the model's predictive performance, as low generalization errors (prediction errors when the model applied to cases outside of the dataset) are often associated with low-dimensional feature vectors.

The choice of features depends on the structural information captured by the features and their relevance to the physical property of interest (Ponzoni et al., 2017). However, it is not always easy to determine which features one should keep. For example, there are some 200 scalar molecular descriptors contained in the Python package RDKit, including the molecular weight, molecular weight with hydrogen atoms ignored, the number of rings in the molecule, the number of carboxylic acid groups in the molecule, and so on. Of these 200 descriptors, only a handful might be relevant to the physical property of interest. How can we determine which features to use?

While feature selection can be performed on the basis of domain knowledge (i.e., chemical expertise), some systematic methods can be applied. One example is to solve a combinatorial optimization problem, where alternative subsets of features are selected and evaluated by looking at the predictive performance of an ML model trained using those features only. This method requires the family of ML models

[5] https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html
[6] https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html

to be fixed in advance (see the next section). Another example is to simply filter features according some statistical measures (e.g., Pearson correlation or $\tilde{\chi}^2$) which evaluate the relationship between each feature and the physical property. Here, features which score poorly according to these statistical measures are removed. A third example is feature selection methods which are performed automatically as part of the learning process. These methods include the penalized linear regressor (Lasso), decision trees, and random forest.

In general, feature selection methods can be classified into four types: filter methods, wrapper methods, embedded methods and feature importance methods. The second example in the above paragraph describes a filter method. Filter methods identify the features independently of any ML model. Wrapper methods make use of a model to select an important subset of features by recursive feature elimination. Embedded methods select features during the training process, where feature importance is calculated after training an ML model using all features when weights are assigned to each of those features. For a comprehensive review of feature selection algorithms, we refer the reader to (Khaire & Dhanalakshmi, 2019).

Feature selection is closely related to dimensionality reduction. Dimensionality reduction can be performed by creating a small number of new features by combining the original ones in some way. The most widely used example is principal component analysis (PCA), which produces a set of new, uncorrelated features (principal components) which are linear combinations of the original features. Importantly, much of the variation in the original features can often be captured using only a few principal components, allowing for these few principal components to be used in place of the original feature vectors. Moreover, examination of the weight coefficients for the original features within the principal components helps us to identify correlations between the original features. PCA has a long history of success particularly in the area of protein dynamics (David & Jacob, 2014; Lange & Grubmueller, 2008) and folding (Maisuradze et al., 2009). In the pioneering paper (Curtarolo et al., 2003), PCA and DFT calculations were employed as a tool to make quantitative predictions for the design of new crystal structures from a class of binary alloys. By unveiling correlations in the features of a large class of crystalline alloys, the authors were able to use PCA to develop a principle for structures–properties selection.

Model selection

Returning to the ML workflow, following pre-processing and feature selection, we specify the *training set* (usually selected randomly from the dataset, this is the set which will be utilized to train the ML model) and the *test set* (which corresponds to the complement of the training set in the original dataset and which will be utilized to test the ML model performance). From an algorithmic standpoint, both sets are well defined and re-initialization of these sets should be avoided, particularly during model validation. The selection of the training and test sets should be prior to the choice of a ML model.

Having pre-processed our database and determined the training and test data, we are finally ready to build the ML model. Formally, this involves the selection of a so-called family of ML models. A family refers to a set of models which have the same mathematical form, make the same assumptions on the dataset, and are indexed according to the values of their parameters. In order to determine an appropriate family of ML models, some insight into the data (for instance, via elementary statistical analysis) is helpful. For example, for many families of ML models, including support vector machine (SVM) and kernel ridge regression (KRR) models discussed later, good performance is guaranteed only when the data is selected independently and identically. If such distributional assumptions are violated, then a non-parametric family of models, such as random forests, might be considered. For other cases, heuristic rules of thumb can be used. For example, in our experience support vector regression (SVR) models tend to perform better on high-dimensional datasets compared to KRR models, but only when the amount of data is small. On the other hand, random forests tend to be robust against outliers, but also tend to give biased predictions when dealing with categorical variables. In Sections 5 and 6, we review a few families of supervised ML models and discuss their underlying mathematical principles.

Having selected the family of ML models we can then move on to training. This amounts to setting the values of the parameters of the family in order to maximize some accuracy criterion. However, for many families of models, these parameters are of one of two types. We refer to these two types as *intrinsic parameters* and *hyperparameters* respectively. This distinction is important, because during model training they are specified in different ways.

Intrinsic parameters appear as a consequence of the type of mathematical object that the family describes. For example, all models in the family of SVM models describe a linear hyperplane. For this family, the parameters describing the orientation of the linear hyperplane ($\mathbf{w}$ in Section 5.2.1) are the intrinsic parameters. Simple formulas for these parameters (such as Eq. (18a)) can usually be obtained by optimizing an appropriate objective function defined in terms of the training data. We discuss examples of such formulas and their derivations in Section 5. Given the training data, intrinsic parameters can be calculated directly (or occasionally, numerically) according to these formulas.

Hyperparameters account for the remaining model parameters. Broadly speaking, these parameters do not have a direct connection with the mathematical object described by the model family. For example, in the family of KRR models, a so-called ridge constant λ is usually included to prevent overfitting and to improve the numerical stability of a matrix inversion operation (see Eqs. (34) and (37) in Section 5.1.4). The ridge constant does not describe the mathematical object of this family (a linear hyperplane), but nonetheless has a major influence on model performance. Other common hyperparameters include those of the kernel function, as described extensively in Section 5. Hyperparameters are often set by cross-validation and grid search techniques. These are iterative techniques which evaluate model predictions in some way. At the beginning of each iteration, the values of the hyperparameters are specified, the values of the intrinsic parameters then calculated according to their formulas, and the accuracy of the resulting model is evaluated according to criteria such as mean absolute square error (MAE) or root mean square error (RMSE). It is quite common to see the coefficient of determination (R^2) used as evaluation criterion in the materials chemistry literature, despite the fact that it is excessively sensitive to outliers at the extremes of the data range and insensitive to those in the middle. In addition to these methods, there are several systematic selection methods based on information criteria, such as the Akaike information criterion (AIC) (Akaike, 1974), the Bayesian information criterion (BIC) (Schwarz, 1978), and Akaike's Bayesian information criterion (ABIC) (Akaike, 1980). The AIC and BIC take into account the risk of overfitting and underfitting by dealing with the trade-off between the goodness of fit and model simplicity.

3. Obtaining the response variables with DFT

In the majority of materials chemistry applications of ML at present, the response variables for the training and test data are calculated from computational simulations, rather than measured from laboratory experiments. Of the various simulation methods in the computational physics toolbox, DFT is the most popular. There are at least two reasons for DFT's popularity. Firstly, DFT achieves a favorable balance between physical accuracy and computation time. Secondly, the DFT method is implemented in numerous software packages, making it accessible to non-specialists. With these software packages, it is possible to compute the physical properties for large numbers of candidate materials with reasonable accuracy, all within days of computational time on typical high-performance computers.

ML models are data-agnostic, in the sense that they can be built independently of the data source (which might be experimental, computational or even a combination). However, in practical situations,

adjustments and customization of the ML model may be required when the source of the data changes. We refer to (Zakutayev et al., 2018) for a concise and critical discussion on the main advantages and disadvantages of using experimental versus computational data sources when building ML models. For a representative application of experimental datasets to training ML models in materials chemistry (aside from molecules), see (Legrain et al., 2018), where the performance of a ML model trained over experimental data from the ICSD dataset[7] for the predictions of new half-Heusler materials is compared with predictions based on high-throughput computations. It is noted that the latter shows inconsistencies in the predictions of various types, and a detailed discussion of causes and origins of such inconsistencies is presented (see also (Schmidt et al., 2019)). As an example of the use of a combination of experimental and computational data for building ML models, see (Hautier et al., 2010), where ML models were built on experimental databases and used to predict new ternary oxides structures for which phased stability is then predicted via DFT. Typically, the use of experimental datasets is limited by the availability of datasets which are sufficiently large and diverse. In experimental research, data size and data diversity are often conflicting aims (Zakutayev et al., 2018). More examples are reported in Section 8.1, where we give some perspectives on materials discovery.

From a model-building point of view, the use of computation (DFT)-derived data has two benefits. The first benefit is that DFT-derived data is noise-free. If the same physical property is computed repeatedly for the same molecule using the same computational settings, then the same result will always be obtained. In other words, there is no need to assume that measurements of a physical property contain (say) a normal random variable as an additive term. This point is non-trivial when compared to the case of data obtained from laboratory experiments, which will always contain some random noise due to inconsistencies in the condition of the laboratory apparatus, laboratory temperature, quality of the material sample, and so on. The second reason is that DFT-derived data are easy to collect in an independent and identically distributed (iid) fashion; in a sequence of molecules, the result of a DFT calculation for a particular molecule will not depend upon the result obtained for the previous molecules. Again, this is a non-trivial difference compared to the experimental situation, in which the condition of a laboratory apparatus will usually be affected by what materials have been inserted into it previously. Thus, providing that the molecules in the dataset have been selected independently and at random, we can assume that our training set contains noise-free, iid samples.

The following section provides an introduction to DFT. There are two reasons for introducing DFT to a computer science and math audience. The first concerns the accuracy of DFT-derived data. While DFT-derived data is noise-free, it is not error-free. In DFT, a small but often important contribution to the energy of the material (the so-called exchange–correlation energy) is unknown and needs to be approximated. This approximation introduces an error into the calculation. The fact that the approximations used are poorly understood makes this error particularly difficult to minimize. The first purpose of this section is to explain how this error arises and to introduce the approximations and corrections that are used to deal with this error. We hope that this section will enable the audience to understand and question the approximations used to obtain their dataset, and encourage them think about how this error could be accounted for in the ML model-building process.

The second reason for introducing DFT to the audience is that DFT itself is becoming a target of ML applications. The approximations used in DFT contain parameters which need to be carefully selected in order to achieve reliable predictions (such as the mixing parameters in the hybrid functionals, or the U parameter in the DFT+U method; see Section 3.3). A number of authors have recently attempted to optimize these parameters by application of ML optimization methods (for example (Yu et al., 2020)), and efforts in this area are likely to continue. However, in order to participate in this emerging area, some background knowledge of DFT is necessary.

[7] https://icsd.products.fiz-karlsruhe.de/

3.1. The Schrödinger equation

DFT is a method to approximate the solutions of the Schrödinger equation, an axiomatic equation from quantum physics. We will first briefly introduce the Schrödinger equation before discussing DFT in the following subsection.

In most areas of materials chemistry, the materials (individual molecules or otherwise) are modeled as a set of electrons in the presence of a static set of atomic nuclei. Given the coordinates of the nuclei, the (time-independent) Schrödinger equation for this model is written as

$$H\Psi(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n) = E\Psi(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n), \tag{1}$$

where $\mathbf{r}_k$ is the position vector of electron k, E is a scalar which corresponds to the energy of the electrons in the material, and $\Psi(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n)$ is a complex-valued, $3n$-dimensional function called the wave function. The square modulus of the wave function, $|\Psi(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n)|^2$, is interpreted as the probability of simultaneously finding electron 1 at position $\mathbf{r}_1$, electron 2 at position $\mathbf{r}_2$, …, and electron n at position $\mathbf{r}_n$. The coordinates of the nuclei enter through the Hamiltonian operator H, defined as

$$H = -\frac{\hbar^2}{2m}\sum_{i=1}^{n}\frac{\partial^2}{\partial \mathbf{r}_i^2} + \frac{e^2}{8\pi\varepsilon_0}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} - \frac{e^2}{4\pi\varepsilon_0}\sum_{i,I}\frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|}, \tag{2}$$

where $\hbar$, e, and ε_0 are physical constants (the reduced Planck constant, electron charge, and permittivity of vacuum, respectively), $\mathbf{R}_I$ is the position vector for nucleus I, and Z_I is the atomic number of nucleus I. In (2), the second sum is over all electron–electron pairs, and the third sum is over all electron–nuclei pairs. After operating on the wave function, the first term gives the kinetic energy of the electrons, the second gives the potential energy arising from repulsions between pairs of electrons, and the third gives the potential energy arising from attractive interactions between the electrons and nuclei.

It can be seen from Eq. (1) that the Schrödinger is an eigenvalue problem, with the wave function acting as the eigenfunction and the energy as the eigenvalue. In principle, Eq. (1) could be solved by specifying boundary conditions for the wave function and employing techniques from calculus and linear algebra. Doing so would give us a set of solutions in the form of eigenvalue–eigenfunction pairs $\{E_k, \Psi_k(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n)\}$. The solution corresponding to the lowest (most negative) eigenvalue is the one of primary interest to materials chemistry. This solution (which we label as $k = 0$ for convenience) is known as the *ground-state solution*. The ground-state energy E_0 is defined as the total energy of the material at zero temperature (0 Kelvin or −273 degrees Celsius). The ground-state wave function $\Psi_0(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n)$ is defined such that its square modulus is the probability density for the electrons at zero temperature. The other solutions to (1) are known as *excited states*, and are physically meaningful as well. However, for reasons that will become clear soon, they fall outside of the domain of DFT and are therefore rarely considered in materials chemistry, especially when high-throughput calculations and ML are involved.

The second term in the Hamiltonian operator in (2), which corresponds to electron–electron repulsions, forbids any attempt at an analytical solution to the Schrödinger equation except for trivial cases. During the 20th century, numerous numerical techniques were developed to solve (1). Most of these methods approximate the second term in some way, and then solve (1) for the energies and wave functions directly. These numerical techniques are derived from a rich and systematic theory (known as post-Hartree Fock theory), however they usually require enormous computational resources. In practice

they are limited to systems containing only a small number (10–100) electrons, and are too costly to be applied to large numbers of molecules (Kedziera & Kacmarek-Kedziera, 2017; Tsuneda, 2014). With some notable exceptions (Zhang & Gruneis, 2019), such methods are rarely used for computing physical properties for materials chemistry.

3.2. Kohn–Sham Density functional theory

DFT started life as a kind of theoretical curiosity in the late 1920s, before being developed into a systematic method for solving the Schrödinger equation during the second half of the 20th century. In contrast to the numerical methods mentioned above, DFT attempts to compute the ground-state energy from the density of electrons inside the material, without considering the wave function directly. In order to proceed, we must define the electron density. Recalling the discussion following Eq. (1), the probability per unit volume of finding electron 1 at position $\mathbf{r}$, irrespective of the position of the other electrons, is

$$\rho(\mathbf{r}) = \int \left|\Psi(\mathbf{r}, \mathbf{r}_2, \dots, \mathbf{r}_n)\right|^2 d\mathbf{r}_2 \dots \mathbf{r}_n, \tag{3}$$

where we have adopted the 'physics notation' $\int d\mathbf{r}_2 \dots \mathbf{r}_n$ to denote integration over $\left(\mathbf{R}^3\right)^{n-1}$, where $\mathbf{R}$ denotes the real numbers. Now, note that electrons have a fundamental physical property known as indistinguishability. This fundamental property, which arises from the quantum nature of ultra-small particles, means that it is impossible to uniquely label electrons, i.e., Eq. (3) applies equally well to all other electrons in addition to electron 1. Eq. (3) is therefore the probability of finding any electron at position $\mathbf{r}$ in space, irrespective of the position of the other electrons, and hence $\rho(\mathbf{r})$ can be understood as the electron density at point $\mathbf{r}$.

Modern DFT is motivated by the Hohenberg–Kohn theorem in 1964 (Hohenberg & Kohn, 1964) which roughly states that the ground-state energy is a functional of the electron density. In other words, $E_0 = F[\rho]$ for some functional F, where F maps functions of the type defined in Eq. (3) to the real numbers. While the Hohenberg–Kohn theorem was a theoretical breakthrough, it does not tell us the form of the functional F, nor does it give any clues about how the electron density could be constructed without knowing the wave function. These issues were addressed by Kohn and Sham in 1965 (Kohn & Sham, 1965), who created the Kohn–Sham method which forms the basis of much of DFT today.

The Kohn–Sham method follows a three-step strategy. In step 1, an artificial electron system in which electron–electron repulsions are absent is constructed. Crucially, this artificial system is constructed in such a way that its electron density $\rho_a(\mathbf{r})$ is exactly equal to $\rho(\mathbf{r})$, the electron density of the real material in question. In step 2, the Schrödinger Eq. (1) is solved for this artificial system, which is feasible due to the absence of the difficult second term in (2). In step 3, the ground-state energy of the real system is computed from $\rho_a(\mathbf{r})$ by means of the so-called Kohn–Sham energy functional.

In step 1 of the Kohn–Sham method, the artificial system contains the same number of electrons (n) as the real material in question. Due to the absence of electron–electron interactions, the wave function and Hamiltonian for the artificial system can be written as

$$\Psi_a(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n) = \phi_1(\mathbf{r}_1)\phi_2(\mathbf{r}_2)\cdots\phi_n(\mathbf{r}_n), \tag{4}$$

and

$$H_a = H_1 + H_2 + \cdots + H_n, \tag{5}$$

respectively. In the above, H_k is a so-called one-electron operator which operates on functions of $\mathbf{r}_k$ alone. The function $\phi_k(\mathbf{r})$ is called a one-electron wave function, and is an eigenfunction of H_k. Kohn and Sham showed that when the one-electron operators have the form

$$H_k = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial \mathbf{r}_k^2} - \frac{e^2}{4\pi\varepsilon_0}\sum_{I=1}^{N}\frac{1}{|\mathbf{r}_k - \mathbf{R}_I|} + \frac{e^2}{4\pi\varepsilon_0}\int \frac{\rho_a(\mathbf{r})}{\mathbf{r}_k - \mathbf{r}}d\mathbf{r} + v_{XC}(\mathbf{r}_i), \tag{6}$$

then $\rho_a(\mathbf{r}) = \rho(\mathbf{r})$ for all $\mathbf{r}$. Note that in (6), the nuclear coordinates $\mathbf{R}_I$ are those of the real system, whereas the electron coordinates $\mathbf{r}_i$ are those of the artificial system. The function $v_{XC}(\mathbf{r}_i)$, which is called the exchange–correlation potential, is assumed to be a known function. $\rho_a(\mathbf{r})$ is the electron density of the artificial system, and can be calculated as

$$\rho_a(\mathbf{r}) = |\phi_1(\mathbf{r})|^2 + |\phi_2(\mathbf{r})|^2 + \cdots + |\phi_n(\mathbf{r})|^2, \tag{7}$$

In step 2, the one-electron wave functions are computed by solving the eigenvalue problem for each one-electron operator. The electron density is then calculated according to (7). However, because the Hamiltonian in (6) itself depends upon the electron density, these calculations must be coupled with an iterative procedure called the self-consistent method. In the self-consistent method, a trial electron density is inserted in (6) and then an updated electron density is computed numerically using (7). This updated electron density is then used as the trial electron density in the next iteration. This procedure is iterated until the trial and updated electron densities converge in some sense. Technical details of the self-consistent method can be found in (Dederichs & Zeller, 1983).

For step 3, Kohn and Sham showed that the ground-state energy of the real system can be computed through the so-called Kohn–Sham functional. The Kohn–Sham functional is defined by

$$F[\rho_a] = -\frac{\hbar^2}{2\ m}\sum_{i=1}^{n}\int \phi_i^{\star}(\mathbf{r})\frac{\partial^2}{\partial \mathbf{r}^2}\phi_i(\mathbf{r})d\mathbf{r} - \frac{e^2}{4\pi\varepsilon_0}\int \frac{Z_I\rho_a(\mathbf{r})}{|\mathbf{r} - \mathbf{R}_I|}d\mathbf{r} + \\ + \frac{e^2}{8\pi\varepsilon_0}\int \frac{\rho_a(\mathbf{r}_1)\rho_a(\mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|}d\mathbf{r}_1 d\mathbf{r}_2 + E_{XC}[\rho_a], \tag{8}$$

where $F[\rho_a] = E_0$, the ground-state energy of the real material, and the asterisk indicates the complex conjugate. $E_{XC}[\rho_a]$ is called the exchange–correlation energy functional, and is defined through the identity $v_{XC}(\mathbf{r}) = \delta E_{XC}[\rho_a]/\delta\rho_a(\mathbf{r})$, where δ denotes the functional derivative. For the applications discussed in this paper, Eq. (8) is the final result of the DFT calculation. Functionals and expressions for ground-state properties other than the energy can be derived as well.

There are two points to note in passing. First, the one-electron orbitals and energies of the artificial system themselves have no direct physical meaning, and are only a means to obtain the electron density of the real system. Nonetheless, they are often used to approximate the excited states of the real system. This somewhat unprincipled approximation often yields surprising agreement with the excited states calculated by more rigorous methods (see (Stowasser & Hoffmann, 1999)). Second, Eq. (8) strictly pertains to the ground-state electron density and energy, respectively. While considerable efforts have been made to extend DFT to excited states (Escudero et al., 2017; Görling, 1996), these methods are not routine and cannot be employed reliably in a high-throughput fashion for generating databases. In ML projects, DFT-derived data tends to be limited to the ground-state properties of materials.

3.3. The exchange–correlation energy

The exchange–correlation potential $v_{XC}(\mathbf{r}_i)$, which is treated as a known function in the Kohn–Sham method, is unknown in practice. In turn, this means that the exchange–correlation energy functional $E_{XC}[\rho_a]$ is unknown. While the exchange–correlation energy is generally expected to be much smaller than the other terms in (8) (Pribram-Jones et al., 2015), in practice it must be approximated in some way to obtain reasonable results. This approximation is the source of the error mentioned above.

Over the last three decades, thousands of approximate exchange–correlation energy functionals have been published, each aiming to improve the accuracy of DFT for various classes of materials. This is where the shortcomings of DFT come to the fore: there is no universal, systematic theory from which these approximate functionals are derived. Rather, these approximate functionals are based on broad heuristics and are justified mainly by their performance against experimental

data. The lack of systematic theory means that we have no method for systematically reducing the error caused by these approximations. Fortunately, only a handful of approximate exchange–correlation energy functionals are used routinely for materials chemistry, and their domains of applicability have been well characterized. Here, we will simply list these functionals and give a brief description of the underlying assumptions and their performance. Each of these functionals decomposes the exchange–correlation energy into the so-called exchange and correlation contributions, i.e.,

$$E_{XC}[\rho_a] = E_X[\rho_a] + E_C[\rho_a], \tag{9}$$

and approximates $E_X[\rho_a]$ and $E_C[\rho_a]$ separately. Roughly speaking, the exchange contribution $E_X[\rho_a]$ measures the energy cost of switching the labels of the electrons (this unintuitive phenomenon is a consequence of the quantum mechanical laws governing ultra-small particles). The correlation contribution $E_C[\rho_a]$ measures the energy cost associated with correlated motions of electrons (such as when one electron moves as a result of repulsive forces arising from the motion of another electron).

The Local Density Approximation (LDA). The LDA, which was published in 1981, is the oldest and arguably simplest of the approximate exchange–correlation functionals (Ceperley & Alder, 1980; Perdew & Zunger, 1981). This approximation assumes that the electron density is locally smooth and does not undergo dramatic variations between nearby points. Consequently, the LDA works well for metallic systems. However, for molecules or solids containing covalent bonds, it tends to predict overly strong bond energies (Table 1 column 1).

The Generalized Gradient Approximation (GGA). The GGA uses LDA as a starting point, and relaxes the smoothness assumption by incorporating a dependence on the local electron density gradient. This heuristic approach is justified by observing that the local gradient of the electron density will appear as the second-order term in a Taylor's series expansion of the exchange–correlation energy functional. Numerous GGAs have been proposed, the most popular of which is the Perdew–Burke–Ernzerhof (PBE) functional, which was published in 1996 (Perdew et al., 1996). PBE provides reliable predictions for the energies of metals, covalently bonded solids, and molecules (Table 1 column 3), and is a good 'default' starting point for much materials research. However, it fails for materials involving long-distance electrostatic (van der Waals) interactions (which typically occur between molecules in an molecule cluster or molecular crystal). This shortcoming has become more serious over recent years as experimental materials chemists have increasingly moved towards systems dominated by van der Waals interactions (such as graphene-like two-dimensional materials).

Hybrid functionals. Hybrid functionals consist of mixing different fractions of LDA and GGA exchange energies with the so-called Hartree–Fock (HF) exchange energy. The HF exchange energy ($E_X^{\mathrm{HF}}[\rho_a]$) emerges rigorously from so-called Hartree–Fock theory, which is an older method for solving the Schrödinger equation. For theoretical reasons, the HF energy is expected to appear in the exchange energy functional $E_X[\rho_a]$ as well. While the LDA and GGA terms can be calculated using the methods described above, the most important part determining the quality of a hybrid functional and its applicability to a specific set of problems is the mixing weight of each term.

The most widely used hybrid functional for calculating molecule properties is the B3LYP (Becke, 3-parameter, Lee–Yang–Parr) functional. The functional mixing is given with the following expression:

$$E_{XC}^{\mathrm{B3LYP}} = (1-a)E_X^{\mathrm{LDA}} + aE_X^{\mathrm{HF}} + bE_X^{\mathrm{B}} + (1-c)E_C^{\mathrm{LDA}} + cE_C^{\mathrm{LYP}}. \tag{10}$$

In Eq. (10), E_X^{B} is the exchange functional from a GGA called Becke88, and E_C^{LYP} is the correlation functional from another GGA called Lee–Yang–Parr. E_X^{B} and E_C^{LYP} are mixed with the LDA exchange and correlation functional and the HF exchange through three parameters a, b, and c, which have been set to the values of $a = 0.2$, $b = 0.72$, and $c = 0.81$ by fitting to experimental data. The quality of the B3LYP functional strongly depends on the values of these three parameters, and they may need to be adjusted in order to reproduce better the experimental measurements for a given subset of molecules.

Table 1
Comparison of atomization energies for various molecules calculated from density functional theory with the LDA, PBE, and HSE06 exchange–correlation functionals. All energies are reported in electron volts. Calculations were performed with DFT as implemented in the Fritz Haber Institute ab initio molecular simulations (FHI-aims) package version 171221_1 (Blum et al., 2009), using "really tight" basis settings. Calculations with the LDA functional used the Perdew–Wang 1992 parameterization (Perdew & Wang, 1992). The HSE06 functional used 25% Hartree–Fock exchange and a range separation parameter of 0.11 Bohr^{-1}. All molecule structures were relaxed using the respective exchange–correlation energy functionals and "light" basis orbital settings prior to energy calculations. Experimental values (with zero-point vibrational energy removed) are from the Computational Chemistry Comparison and Benchmark Database (http://cccbdb.nist.gov/).

Molecule	LDA	PBE	HSE06	Experiment
CH_4	20.05	18.22	18.08	18.23
C_2H_2	19.96	18.01	17.51	17.54
C_2H_4	27.43	24.81	24.41	24.39
C_6H_{12}	86.77	77.57	76.70	76.52
C_6H_6	68.33	61.20	59.98	59.19

While the B3LYP method shows remarkable results when applied to molecules, it has found only limited application in materials science where its achievements in modeling of metal crystal lattices, complex oxides and semiconductor materials are only moderate. Materials-oriented hybrid functionals have been developed, and among the most popular nowadays are the PBE0 and HSE (Heyd–Scuseria–Ernzerhof) families of functionals. The PBE0 functional mixes the Perdew–Burke–Ernzerhof (PBE) exchange energy and HF exchange energy in a set $3:1$ ratio, along with the full PBE correlation energy:

$$E_{XC}^{\mathrm{PBE0}} = 0.25E_X^{\mathrm{HF}} + 0.75E_X^{\mathrm{PBE}} + E_C^{\mathrm{PBE}}. \tag{11}$$

The HSE functional originates from the PBE0 functional and is given by the expression:

$$E_{XC}^{\mathrm{HSE}} = aE_X^{\mathrm{HF,\ SR}}(w) + (1-a)E_X^{\mathrm{PBE,SR}}(w) + E_X^{\mathrm{PBE,\ LR}}(w) + E_C^{\mathrm{PBE}}. \tag{12}$$

In Eq. (12), SR and LR denote the short range and long-range electron–electron interactions, respectively. The mixing parameter a determines the HF and GGA exchange mixing while the adjustable parameter w determines the relative weight of short range and long range interactions to the exchange energy. The functional known as HSE06 uses the values $a = 0.25$ and $w = 0.2$. For the value of $w = 0$ the method is identical with PBE0.

The performance of the HSE06 functional for calculating molecule properties is illustrated in column 4 of Table 1. These results are typical for hybrid functionals when applied to molecules. While it tends to give superior predictions than the LDA or GGA, considerable computational time is required for computing the Hartree–Fock exchange, which makes the HSE06 and other hybrid functionals less preferable for generating databases of material data.

It can be seen that the hybrid DFT functionals are parameter-dependent methods, and their application to specific problem or class of molecules or materials can potentially benefit from ML based optimization methods. ML methods therefore can not only assist the discovery of novel materials, but also optimize and improve DFT itself. This is an area where computer science and math majors could make potential contributions.

Alternatives to hybrid functionals. While hybrid functionals are usually more reliable than LDA or GGA-based functionals, their large computational cost can be prohibitive, particularly when high-throughput calculations are involved. Due to this, a number of alternative methods have been developed which retain some of the benefits of hybrid functions while keeping computational costs low. We finish this section by introducing three such methods, namely *van der Waals (vdW) functionals*, *vdW corrections, and DFU+U*.

vdW functionals mix the exchange and correlation terms of an LDA- or GGA-based functional with a non-local term. This non-local term accounts for long-distance correlations between electron motion, which is essential for modeling van der Waals interactions between materials. Several van der Waals functionals have been developed, the most prominent ones being from the so-called vdW-DF family of functionals (Klimeš et al., 2009, 2011). The present authors have the most experience working with the rev-vdW-DF2 functional, which yields excellent predictions for materials such as graphite or molecule monolayers on surfaces compared to the ordinary LDA or GGA-based functionals (Hamada, 2014). While the non-local term significantly increases computational costs, these costs remain modest compared to those of hybrid functionals.

In addition to vdW functionals, several vdW correction schemes have been developed. In these schemes, the electron density is first computed with the LDA or GGA, and then a correction term is added to account for the effects of van der Waals interactions (Tkatchenko & Scheffler, 2009). These correction schemes also yield reliable predictions for materials involving van der Waals interactions, without the additional cost of computing the non-local term.

The DFT+U method is an ad-hoc correction to the DFT-calculated energy (E_{DFT}) for systems such as transition metal oxides or metal complex molecules, in which electron–electron correlations are strong. This correction adds an energy penalty whenever electrons reside at the same point in space. The major motivation of DFT+U is to alleviate the so-called self-interaction error without incurring the large costs of computing a hybrid functional. The self-interaction error is an unphysical effect in which an electron interacts with itself; it manifests in the third term in the Kohn–Sham function in Eq. (8) whenever $\mathbf{r}_1 = \mathbf{r}_2$.

The correction used in the DFT+U method is inspired by the Hubbard model from solid-state physics, which describes hopping of electrons between points within a crystal. While several variants of this correction term exist, one of the most common is

$$E_{\text{tot}} = E_{\text{DFT}} + \frac{U-J}{2} \sum_{\sigma} \left(\sum_{m_1} n^{\sigma}_{m_1,m_1} - \sum_{m_1,m_2} n^{\sigma}_{m_1,m_2} n^{\sigma}_{m_2,m_1} \right) \tag{13}$$

For the precise definitions of the $n^{\sigma}_{m_1,m_1}$, $n^{\sigma}_{m_1,m_2}$ and $n^{\sigma}_{m_2,m_1}$, the reader is referred to (Dudarev et al., 1998). For practical purposes, it is sufficient to know that the $n^{\sigma}_{m_1,m_1}$ measures the number of electrons residing at the same site within the material, and n_{m_1,m_2} measures the degree of electron delocalization between sites. The parameters U and J measure the energies arising from (Coulomb) repulsions and exchange between electrons residing at the same site, respectively. In practice, it is more convenient to define the effective Hubbard parameter $U_{\text{eff}} = U - J$. The correction term is therefore controlled by adjusting the parameter U_{eff}.

As with hybrid functionals, the effectiveness of DFU+U depends upon the value of U_{eff} chosen. The estimation of the optimal value of U_{eff} is often based on empirical techniques where U_{eff} for adjusted until the DFT+U results reproduce correctly experimental values such as band gap, lattice parameters, or magnetic properties. A different approach is to perform DFT calculations with hybrid functionals, and then adjust the U_{eff} value until the DFT+U results satisfactory agree with the results obtained with hybrid functionals. Such an approach is solely based on computational methods and ideally would result in low-cost DFT+U calculations with accuracy similar to hybrid functional calculations. A recent study employed a ML approach to successfully set the value of U_{eff}, which in turn could be used for high-throughput calculations (Yu et al., 2020).

Which exchange–correlation functional or method should be used? This question continues to polarize the materials chemistry community. Some people insist on hybrid functionals, despite their high computational costs. Within chemistry departments in particular, DFT is often equated with B3LYP. Away from chemistry departments, it is not uncommon to find people with a near-religious zealotry for vdW corrections or the DFT+U method. Then there are people who disavow both approaches completely, preferring the simplicity of the GGA or the physically principled LDA. A computer science or math major should keep these points in mind before entering the DFT fray. Their choice of preferred functional will probably be influenced by the type of materials chemist that they first encounter.

As far as ML for materials chemistry is concerned, our general recommendation is to use the PBE functional by default. If materials with van der Waals interactions are of interest, then a van der Waals functional or van der Waals corrections could be considered. However, this recommendation should be accepted with a grain of salt; as the reader gains experience and confidence in using DFT, they will develop their own preferences.

3.4. Obtaining molecular structures from DFT

It can be seen from Eqs. (6) and (8), the Kohn–Sham method requires the three-dimensional structure of the molecule (the Cartesian coordinates of each atom) as input. However, as described in Section 2, it is unlikely that we would be supplied with this information in a materials chemistry project. More likely, we would receive a list of candidate molecules in some compact chemical representation. In order to correctly calculate the energy of the molecule, it is therefore necessary that we first generate the molecule structure from its chemical representation.

In fact, it is possible to use DFT to generate the molecule structure using an iterative scheme. Starting with a trial structure, the energy of the structure is calculated according to (8), as well as its derivatives with respect to the atomic coordinates. The atom positions are shifted a short distance in the direction of decreasing energy, yielding an updated structure. This process is repeated until the energy difference between subsequent structures is sufficiently small. This process is known as *structure relaxation*. Chemistry software such as *ChemSketch* or *ChemDraw* can be used to generate good trial structures from the structural formula for the molecule, based upon intuitive chemical expectations for the bond orientations in the molecule. Some software packages such as Turbomole can also generate trial structures from structural formulas in this manner.

While DFT-based structure relaxation is quite efficient for small molecules (less than around 50 atoms), for larger molecules it can become quite time-consuming. If we have a large database of medium or large molecules, it could be unreasonable to perform a structure relaxation for each one. For such cases, we can perform a so-called *classical structure relaxation* instead. Classical structure relaxations use a similar iterative procedure as the one described above, however the potential energy and its derivatives are calculated using approximate formulas derived from Newtonian mechanics (Lorenz & Doltsinis, 2017). Classical structure relaxations are extremely efficient, however the final results may differ slightly from ones obtained from DFT. These small differences may affect the reliability of subsequent DFT-calculated ground-state properties, and therefore caution is advised. Classical structure relaxations are rarely offered as part of DFT software packages, however they can be routinely performed in classical mechanics simulation software such as LAMMPS (Plimpton, 1995).

3.5. DFT software

One of the major reasons for the popularity of DFT for materials modeling is the availability of software implementing the Kohn–Sham method. When considering different software packages, it is important to be aware that each may be designed for different types of materials and may employ additional approximations.

Before purchasing a DFT software package, it is also important to ensure that you have a computer which can run it. Unfortunately, an ordinary PC or workstation will not do. Rather, a multi-core server running Linux CentOS (or other non-graphical Linux environment) is required. A single server will cost between 10,000–20,000 USD, and several servers may be needed to generate a sufficiently large dataset.

Table 2
List of some materials open-source databases.

	Database/Projects/ Material type	URL
1	Crystallography (Crystal structures for organic, inorganic, and metal–organic compounds and minerals)	http://www.crystallography.net/cod/
2	The Material Project (Inorganic compounds, molecules, nanoporous materials, and various DFT-calculated physical properties)	https://materialsproject.org/
3	AFLOW (Material compounds with various calculated properties)	http://aflowlib.org/
4	Open Quantum Materials Database (Various materials with DFT-calculated thermodynamic and structural properties)	http://oqmd.org/
5	Open Materials Database (Structural information for various materials)	http://openmaterialsdb.se/
6	NOMAD Repository and Archive (Computational-derived data for mainly inorganic materials)	http://nomad-repository.eu/
7	Quantum-machine (DFT-calculated datasets)	http://quantum-machine.org/datasets/
8	AtomWork (Inorganic and metallic materials)	https://crystdb.nims.go.jp/en/
9	MatWeb (Metal, plastic, ceramic, and composite materials)	http://www.matweb.com/
10	NIMS Materials (Polymer, inorganic, and metallic materials, with computational data)	https://mits.nims.go.jp/en/
11	Matmatch (Biological, ceramic, composite, glass, metal, and polymer materials)	https://matmatch.com/

DFT software will cost between 0 and 5,000 USD, with various licensing restrictions to consider. Most DFT software packages are ran from the Linux command line, and so a basic grasp of bash scripting is required to use them. A basic ability to write scripts in languages such as Python or R is useful for processing the output files of the calculation.

Popular DFT software packages for materials chemistry and ML projects include the Vienna Ab Initio Simulations Package,[8] (VASP) (Kresse & Furthmüller, 1996) Quantum Expresso[9] (Giannozzi et al., 2017), Gaussian[10] (Frisch et al., 2016), and the Friz Haber Institute Ab Initio Molecular Simulations Package (FHI-aims)[11] (Blum et al., 2009). Tutorials for all popular DFT software packages can be found online. While these packages all implement the Kohn–Sham method, they each use different numerical methods and approximations. Energies calculated from different software packages may therefore be difficult to compare, and users are advised to stick with a single software package for the duration of their project.

3.6. Open-source databases

An alternative way of obtaining data for ML is to use open-source databases. Many databases contain large amounts of DFT-calculated data, and are a good way for newcomers to learn about materials chemistry data without having to learn how to perform DFT calculations. These databases often contain structural information which is needed for creating feature vectors (see the next section). In Table 2, we give a non-exhaustive list of open-source materials databases and their URLs. Besides databases, some of these URLs offer analysis tools to aid in the design of novel materials (such as The Material Project), online applications for property predictions using ML, prototype encyclopedias (AFLOW), high-throughput toolkit-frameworks for preparing and running calculations, and analyzing results (Open Materials Database).

4. Obtaining the feature vectors (predictor variables)

At present, there is no systematic framework for constructing features or feature vectors for a specific purpose. Instead, feature vectors are created either from chemical intuition or by applying techniques from mathematics (such as computational topology) to vectorize the atom coordinates of the molecules in some way. The freedom involved in creating feature vectors has led to the proliferation of various types in the materials chemistry literature. The purpose of this section is to illustrate to computer science and math majors how basic material structural concepts, such as atoms and chemical bonds, can be codified in the form of feature vectors. We do this in the most straightforward way possible: by surveying the main types of feature vectors used in the materials chemistry literature. While our survey does not touch upon the latest developments, the feature vectors discussed here widely known for providing robust descriptions of molecules and producing ML models with good predictive performance.

While feature vectors can be created with great freedom, they should ideally satisfy four conditions: correlation with the property of interest (in this case, the DFT-calculated energy); invariance under permutations of atom numbering or rigid-body translations and rotations; uniqueness (i.e., no two molecules should give the same feature vector); and easy to calculate. In order to gain acceptance amongst the materials chemistry community, feature vectors should use familiar molecular structure concepts as well (such as inter-atomic distances). In practice, however, it is difficult to create feature vectors which satisfy all of such properties simultaneously. Moreover, in order to ensure that feature vectors are correlated with the physical property of interest, it is important to consider the three-dimensional structure of the molecule. This is because the physical properties of molecules are typically determined by the structure of the molecule (recall Eq. (8)). If our initial list of molecules does not contain three-dimensional structural information, it is first necessary to generate the three-dimensional structures from the chemical representations for every molecule in the list. In principle this can be done via DFT or the classical methods described in Section 3.4.

4.1. Structural key

Structural keys are arguably the simplest type of molecule feature vector. A structural key is a bit vector whose elements are either 0 or 1, where each bit indicates the presence or absence of a specific structural motif in the molecule (such as particular atoms, aromatic rings, or particular functional groups). To construct the bit vector, we need to specify in advance which structural motifs are important to the property of interest. In this sense, structural keys are constructed on the basis of chemical intuition. The calculation of structural keys can be time-consuming because a substructure search needs to be made for every structural motif in all the molecules in the database. Enormous computation times may be required if the database is large.

4.2. Molecular fingerprints

Molecular fingerprints improve the generality of structural keys by eliminating the need to specify pre-defined patterns in advance. A fingerprint algorithm examines the molecule and generates its fingerprint independently of the other molecules. Molecular fingerprints are extracted by a hashing procedure, where each atom in the molecule is assigned a unique integer string based on its environment in the molecule. An iterative procedure is then performed to extract all structural motifs from the molecule. A bit vector describing the presence or absence of structural motifs is then output. Fingerprints usually have sizes between 1000–4000 bits.

Many different types of molecular fingerprints can be defined, depending upon how the structural motifs are extracted. Some representative examples include the atom-pair and topological-torsion fingerprints (Carhart et al., 1985; Nilakantan et al., 1987), and Morgan/Circular fingerprints (Rogers & Hahn, 2010). Each fingerprinting typology is usually shown to be better suited for specific types of

[8] www.vasp.at
[9] www.quantum-expresso.org
[10] www.gaussian.com
[11] https://fhi-aims.org/

molecules. Capecchi et al. has recently invented a new fingerprint called MinHashed which is suitable for both small and large molecules by combining substructures and atom-pairs (Cereto-Massague et al., 2015).

4.3. Coulomb matrix

Coulomb matrices are one of the most popular approaches to forming feature vectors from three-dimensional molecular structures (Rupp et al., 2012). Consider a molecule with n atoms, and let $\mathbf{R}_i$ and Z_i denote the coordinate and atomic number of atom i, respectively. The Coulomb matrix C is an $n \times n$ matrix whose elements C_{ij} are defined as

$$C_{ij} = \begin{cases} \frac{1}{2} Z_i^{2.4} & \forall\, i = j, \\ \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} & \forall i \neq j, \end{cases}$$

where Z_i is the atomic number of atom i and $|\mathbf{R}_i - \mathbf{R}_j|$ is the Euclidean distance between atoms i and j. The diagonal of C consists of a polynomial fit of the nuclear charges to the total energies of the free atom. The off-diagonal elements represent the Coulomb repulsion for each pair of nuclei in the molecule. The Coulomb matrix is invariant under rotations, translation, and other symmetry operations. By construction, larger entries in the Coulomb matrix are associated with heavier atoms. With the exception of homometric structures the Coulomb matrix is a unique representation of a molecule. Homometric structures are compounds sharing the same set of interatomic distances. If these interatomic distances happen to be associated with the same types of atoms, then the two homometric structures will be represented by the same Coulomb matrix. Instances of homometric structures in a dataset are rare, but can occur when dealing with crystalline materials or small organic molecules.

As described in (Hansen et al., 2013), several types of feature vectors can be extracted from the Coulomb matrix. For brevity, we introduce only two: eigen-spectrum representations and vectorized sorted Coulomb matrices.

The eigen-spectrum representation is the vector $(\lambda_1, \lambda_2, \dots, \lambda_n)$ where λ_i is the ith ordered eigenvalue of the Coulomb matrix, that is, $\lambda_i \geq \lambda_{i+1}$ for $i = 1, \dots, n$ (see (Rupp et al., 2012)). In creating the eigen-spectrum representation, we have reduced a n^2-dimensional object to an n-dimensional one, and have therefore incurred some information loss. In addition to its simplicity, an advantage of the eigen-spectrum representation is that it is invariant under permutations of rows and columns of the Coulomb matrix. Care must be taken when dealing with databases of molecules containing different numbers of atoms. This is because the dimensionality of the eigen-spectrum representation is equal to the number of atoms n in the molecule. Different sized molecules will therefore result in eigen-spectra of different dimension. This problem can be dealt with by padding a zero tail onto the eigen-spectrum representation of each molecule, ensuring that all eigen-spectra have the same dimensionality.

Vectorized sorted Coulomb matrices are obtained by permuting the Coulomb matrix in such a way that the rows and columns of Coulomb matrix are ordered by their norm. This type of data representation ensures a well-defined ordering of the atoms in the Coulomb matrix. Feature vectors can then be obtained by vectorizing the Coulomb matrix.

Other types of feature vectors (such as vectorized random Coulomb matrices) can be constructed as well. Feature vectors based on Coulomb matrices have been remarkably successful in ML applications, particularly for predicting molecular atomization energies in which prediction errors as small as 3 kcal/mol have been reported (Hansen et al., 2013; Rupp et al., 2012).

An illustration of the Coulomb matrix is shown in Fig. 2 for the case of a methanol molecule. Its three-dimensional structure is shown on the left of the figure, and its Coulomb matrix is shown on the right. The eigen-spectrum representation for methanol works out to be $[86.71, 26.78, 0.19, 0.18, -0.47, -1.00]$. In Table 3, we compare this feature vector with the molecular fingerprints and structural keys discussed in the previous section. Here, we use MACCS (Molecular ACCess System) keys as our example of structural keys, which are one of the most popular types of structural keys available (Durant et al., 2002). There are two sets of MACCS keys, one consists of 960 keys and the other consists a subset of 166 keys. It can be seen that Coulomb matrices yield very different feature vectors than molecular fingerprints or structural keys; the choice of feature vector will therefore have a profound influence on the resulting ML model.

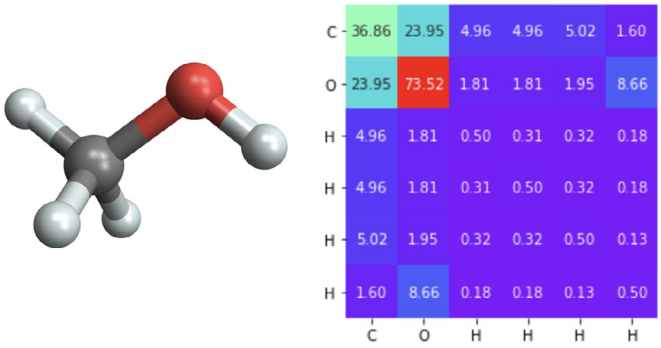


Fig. 2. Three-dimensional structure of methanol and the corresponding Coulomb matrix.

Table 3
Different molecular fingerprints for Methanol, as computed using RDkit (see https://www.rdkit.org/).

Feature vector type	Number of bits	Indexes of 1-bit
Structural key:		
MACCS key (166 keys)	167	[93, 139, 157, 160, 164]
Topological-path	64	[50, 59]
Molecular fingerprints:		
Atom-pairs	8388608	[1590305]
Morgan fingerprint	1024	[33, 131, 807]

4.4. Many-body tensor representation

The many-body tensor representation (MBTR) is a generalization of the Coulomb matrix (Huo & Rupp, 2017). It builds upon the bag-of-bonds concept introduced in (Hansen et al., 2015), and essentially represents a material as a probability distribution of many-body geometric features. By many-body geometric features, we mean features such as distances between pairs of atoms (second-order features), angles between groups of three atoms (third-order features), torsion angles between groups of four atoms (fourth-order features), and so on.

Fig. 3 displays the MBTR for a benzene molecule for the case of two-body features only. It can be seen that the MBTR consists of a family of curves $f(x, z_r, z_s)$ depending on the curve variable x and parameterized by z_r, z_s, where $x > 0$ and z_r and z_s are two choices of atom numbers from $z_1, z_2, \dots, z_n$,

$$f_2(x, z_r, z_s) = \sum_{i,j=1}^{n} w_2(i,j) D_\sigma(x, g_2(i,j)) C(z_r, z_i) C(z_s, z_j). \tag{14}$$

Here, $g_2(i,j)$ represents the distance between atoms i and j, and $D_\sigma(x, g_2(i,j))$ is a broadening function of width σ centered at $g_2(i,j)$, $C(z_r, z_i)$ is a function measuring the similarity between elements of type z_r and z_i, and $w_2(i,j)$ is a weight function which can be used to reduce contributions from widely separated atoms. In Fig. 3, we choose for $D_\sigma(x, g_2(i,j))$ a normal density function with mean $g_2(i,j)$ and standard deviation σ. For $w_2(i,j)$, we choose an exponentially decaying function of the distance between atoms i and j, and for $C(z_r, z_i)$ we choose the indicator function ($C(z_r, z_i) = 1$ if $z_r = z_i$, and 0 otherwise). The formula

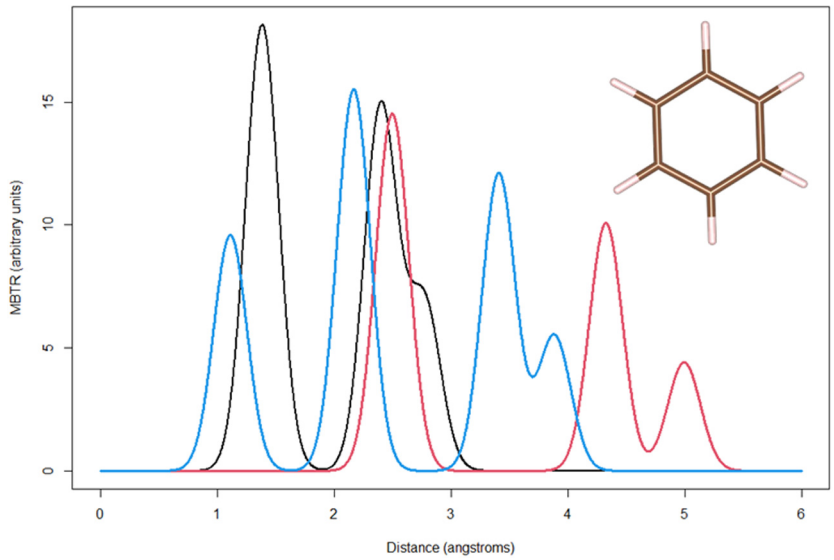


Fig. 3. Many-body tensor representation (MBTR) for a benzene molecule (Eq. (15)). Only two-body geometric features are included. Black: $z_r = z_s = C$. Red: $z_r = z_s = H$. Blue: $z_r = C, z_s = H$. MBTR was calculated using $w_2(i,j) = \exp(-0.2 d_{ij})$ where d_{ij} is the distance between atoms i and j, $D_\sigma(x, g_2(i,j))$ a normal density function with mean d_{ij} and standard deviation 0.2 Å, and $C(z_r, z_i) = 1$ if $z_r = z_i$ and 0 otherwise.

above can be directly generalized to higher-order features (see Eq.(3) in (Huo & Rupp, 2017)).

MBTR is an infinite-dimensional feature representation due to the dependence on the continuous variable x. While the MBTR itself is difficult to use as input to practical ML calculations, the distance between MBTRs of two different molecules may be used in certain ML methods such as the kernelized methods discussed in Section 5. For the case where only two-body geometric features are considered, the distance between MBTR's for molecules M and M' can be defined as

$$d_2^2(M, M') = \sum_{z_i, z_j} \int \left(f_2(x, z_i, z_j) - f_2'(x, z_i, z_j) \right)^2 dx. \tag{15}$$

Eq. (15) can be directly generalized to the cases of higher-order geometric features.

Ref. (Huo & Rupp, 2017) reported a KRR model built by using the MBTR which achieved near-perfect predictive accuracy when applied to the atomization energies of certain crystal structures (Huo & Rupp, 2017). Other papers have also confirmed the superior performance of the MBTR compared to the Coulomb matrix representation when used to build various ML models (Himanen et al., 2020). While the MBTR is attracting an increasing number of users, compared to the Coulomb matrix, it is yet to be widely adopted by the materials chemistry community. This is probably due to the foreboding generality of its formulation, as well as the need to fit parameters in the broadening functions and elsewhere. We will discuss an application of the MBTR to inter-molecular interaction energies in Section 5.1.4.

4.5. Chemically driven persistence image

Recently, a new representation of molecular structures using persistent homology (PH), a technique from computational topology, has been introduced in (Carlsson, 2009; Edelsbrunner & Harer, 2008). In this approach, the three-dimensional structure of a molecule is regarded as a point cloud in $\mathbb{R}^3$. The points in a point cloud are slowly expanded, and as they expand, connected components (rings and cavities) appear and vanish. The connected components, as well as the times at which they appear and vanish during the expansion, have a sensitive dependence on the positions of the atoms in the molecule. The times at which a connected component appears and vanishes are plotted in a two-dimensional scatter plot known as a *persistence diagram*. Feature vectors for the molecules can then be obtained by vectorizing the persistent diagram.

The PH approach can be improved by introducing atomistic information to ensure uniqueness (e.g., by giving the points an initial radius based upon atomic radii). In (Townsend et al., 2020), the difference in electro-negativity for connected components was incorporated. The advantages of persistence images for molecular representation are stability, computational tractability, and the ability to yield feature vectors with a fixed dimensionality regardless of molecular size. In addition, PH has become more accessible in recent years as several libraries implementing PH algorithms have become available. These include the GUDHI library for topological data analysis[12] and HomCloud[13].

4.6. Molecular similarity

For many families of ML models, it is the distance between feature vectors that directly enters the model rather than the feature vectors themselves. This is useful for situations in which it is difficult to construct appropriate feature vectors for the materials of interest; we can quantify the similarity between pairs of materials instead.

This strategy was recently employed in (Packwood & Hitosugi, 2018) to analyze how molecules assemble on metal surfaces. Such molecular assemblies could serve as components for the nano-machines described in the introduction. In (Packwood & Hitosugi, 2018), the authors considered a simplified model for the molecular assembly process, and rigorously characterized the so-called configuration space for the model. Each element of the configuration space corresponds to one way in which the molecules can be arranged on the surface, and the molecular assembly process can be regarded as a trajectory through this space. By devising a measure of similarity based upon a metric between pairs of configurations, the authors could implement an (unsupervised) ML method to predict the chemical properties needed for the molecule to assemble in a specific way.

While similarities were used in the above example for unsupervised ML, they could easily be used for building supervised ML models of the type described in Section 5. For example, the kernel in Eq. (30) depends directly on the squared distance between feature vectors. This distance could be replaced with another similarity metric between pairs of molecule or material candidates. For the case of molecules, two well-known similarity metrics are the Tanimoto similarity (Tanimoto, 1957) and Tversky similarity (Tversky, 1977).

[12] https://gudhi.inria.fr/doc/3.4.1/
[13] https://homcloud.dev/index.html

Besides ML model-building, molecular similarity can also be applied to other tasks such as similarity searching (Willett et al., 1998), property prediction (Brown & Martin, 1996), synthesis design (Wipke & Rogers, 1984), virtual screening (Cereto-Massague et al., 2015), cluster analysis (Cruz et al., 1996), and molecular diversity analysis (Golbraikh, 2000).

4.7. Feature vectors for crystalline materials

While this review focuses on molecules, the field of materials chemistry is not restricted to molecules, and a great deal of effort has gone into developing feature vectors for crystalline materials as well. We make a few remarks about these cases before returning to molecules in the next section.

The structure of a crystal can be visualized as an infinite tessellation of a group of atoms or molecules in space. Crystal structures therefore possess periodic symmetry, which is difficult to account for in the construction of feature vectors. The more prominent efforts at developing feature vectors for crystal systems have focused on incorporating periodicity into the methodologies developed for molecules. The success of the Coulomb matrix has been a particular source of inspiration, and generalizations such as the Ewald sum matrix and the sine matrix have been proposed which incorporate periodic symmetry (Himanen et al., 2020). In contrast to the Coulomb matrix, there are some popular methods for describing molecules, such as the MBTR described above and SOAP (smooth overlap of atomic positions) (De et al., 2016), which can be easily applied to periodic systems after making some minor adjustments.

As well as generalizing molecule-based methods, there have been efforts to develop entirely new methods for the case of crystals. A prominent example is the use of simulated diffraction patterns (Ziletti et al., 2018). In an experimental laboratory, a diffraction pattern is obtained by propagating an x-ray or electron beam into a crystal. The particles of the beam (x-ray photons or electrons) scatter as they interact with the atoms inside of the crystal, and wave interference effects between scattered particles results in the appearance of a pattern of spots when the outgoing beam is detected. The symmetry of this pattern reflects the periodic symmetry of the crystal, and the intensity of the spots reflects the types of atoms contained inside; in short, the diffraction pattern provides a unique fingerprint for the crystal. In the work of (Ziletti et al., 2018), the diffraction patterns were used as features for developing a ML classification model. In Ref. (Packwood, 2020), a similar strategy was used to determine the atomic structure of a metal surface and the surface of a molecular thin film on the basis of an unsupervised learning approach. These approaches are novel because they acknowledge crystal symmetry from the outset, and are not mere generalizations of molecule-based descriptors. On the other hand, their generation can be time-consuming and may require substantial computational resources (the diffraction patterns reported in (Packwood, 2020) were obtained by integrating the time-dependent Schrödinger equation using a finite-elements technique!).

5. Kernelized classification and regression models for materials chemistry

In what follows we review families of ML models based upon the so-called kernel functions. These models represent the status quo for materials chemistry applications at present. By discussing these methods in some detail, computer science and math majors will gain a sense for the majority of the ML applications to materials chemistry so far. In contrast to reviews oriented towards materials science majors, we will not introduce these models as mere black-box algorithms. Rather, we will take a principled approach and illustrate how they are built from a set of starting axioms. However, for ease of reading, proofs and certain other concepts such as complexity bounds will be omitted.

We will use the notation $\mathcal{X}$ to denote the set of all molecules (or materials) of interest, and x to denote an element (a molecule) from $\mathcal{X}$. Instead of $\mathbf{x}$, we will use either $\phi(x)$ or $f(x)$ to denote the feature vector for x. As this notation makes clear, the feature vector is treated as a map from $\mathcal{X}$ to $\mathbf{R}^d$, where d is the dimensionality of the feature vector. We will assume that the feature vectors, or at least the distances between them, are given (perhaps constructed with the methods described in the previous section).

5.1. Classification and regression models

ML models based upon kernel functions rely on a mathematical trick which, in essence, transforms a training dataset in such a way that linear classification or regression techniques can be applied. Despite their apparent simplicity, these models are often very accurate. Moreover, the relatively small number of parameters involved in these models often means that they can be trained with smaller datasets compared to other types of models such as, for instance, neural networks. While types of such models exist, here we focus on only two: kernelized support vector machine (KSVM) and KRR models. Both types of models adequately illustrate the concept of the kernelized ML methods.

5.1.1. Classical SVM

Before introducing KSVM models, it will be instructive to first examine their non-kernelized counterpart, classical support vector machine (classical SVM) models. Consider some training data containing n molecules $x_1, x_2, \dots, x_n, \in \mathcal{X}$ and suppose that molecule k has a label y_k attached, where $y_k = +1$ if molecule k has a physical property of interest and $y_k = -1$ otherwise. Our task is to use the training dataset to determine a function $g : \mathcal{X} \to \{-1, +1\}$ which can predict whether a molecule has the property of interest or not.

To proceed, let $\phi(x_k) = (\phi_1(x_k), \phi_2(x_k), \dots, \phi_m(x_k))$ be a feature vector of m descriptors for molecule x_k. Molecule x_k is therefore represented as a point in $\mathbb{R}^m$, whose coordinates are given by the components of $\phi(x_k)$. The classical SVM method makes the following harsh assumption: the set $\mathcal{X}$ is *linearly separable*, i.e., a linear hyperplane can be drawn through $\mathbb{R}^m$ such that all molecules lying on one side of the hyperplane have label -1, and all molecules lying on the other side of the hyperplane have label $+1$. Under this assumption, the classification function g can be constructed by finding an equation for this hyperplane.

In general, an infinite number of hyperplanes is possible, each of which has the form $\mathbf{w} \cdot \phi(x) + b = 0$ for any choice of $x \in \mathcal{X}$, where $\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}$. In the SVM method, we choose the hyperplane which maximizes the so-called *margin*. The margin, denoted by ρ, is the minimum distance between the hyperplane and any of the points $\phi(x_1), \phi(x_2), \dots, \phi(x_n)$ in the training data.

The (intrinsic) parameters $\mathbf{w}$ and b can be determined by solving a constrained optimization problem. We first consider the maximal (hard) margin classifier, which is formulated as the following minimization problem

$$\min_{\mathbf{w}\in\mathbb{R}^m} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 \tag{16a}$$

$$\text{s.t.} \quad y_i\left(\mathbf{w}\cdot\phi(x_i)+b\right) \geq 1, i = 1, \dots, n \tag{16b}$$

Let $\alpha_i \in \mathbb{R}, \alpha_i \geq 0$ be the Lagrange multipliers associated with the constraint (16b). Define the Lagrangian

$$L(\mathbf{w}, b, \alpha) \triangleq \frac{1}{2}\|\mathbf{w}\|_2^2 - \alpha_i\left[y_i\left(\mathbf{w}\cdot\phi(x_i)+b\right)-1\right]. \tag{17}$$

As is known from optimization theory, the infimum of this Lagrangian is achieved when the first-order conditions are satisfied, i.e., when the partial derivatives of $L(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b vanish. This yields the system of equations

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \tag{18a}$$

$$0 = \sum_{i=1}^{n} \alpha_i y_i \tag{18b}$$

Next, substituting (18a) back to (17) gives us the infimum of the Lagrangian as in the following.

$$\inf_{\mathbf{w}\in\mathbb{R}^m,\, b\in\mathbb{R}} L(\mathbf{w}, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j). \tag{19}$$

We then obtain the following dual optimization problem of the primal optimization problem (16),

$$\max_{\alpha_i\in\mathbb{R}, \alpha_i\geq 0} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j) \tag{20a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{20b}$$

$$\alpha_i \geq 0, i = 1, \ldots, n \tag{20c}$$

In passing, note that the equations above are strictly for the case of the 'hard' maximal margin classifier. As shown in (Cristianini & Shawe-Taylor, 2000), the development of alternative SVM-based classifiers (such as the soft margin classifier) involves dual optimization problem similar to (20), namely

$$\max_{\alpha_i\in\mathbb{R}, \alpha_i\geq 0} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j) \tag{21a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{21b}$$

$$0 \leq \alpha_i \leq C, i = 1, \ldots, n \tag{21c}$$

where the constraint (21) is often called the box constraint. The optimization problems (20) and (21) are convex and can be easily solved by off-the-self software.

Returning to the maximal margin classifier, it can be shown that the plane which maximizes the margin (known as the *maximal margin plane*) satisfies (see (Mohri et al., 2012))

$$\sum_{k=1}^{n} \alpha_k y_k \phi(x_k) \cdot \phi(x) + b = 0, \tag{22}$$

where x denotes a molecule from $\mathcal{X}$ that we wish to classify, and

$$b = y_i - \sum_{k=1}^{n} \alpha_k y_k \phi(x_k) \cdot \phi(x_i), \tag{23}$$

for any choice of $i \in \{1, 2, \ldots, n\}$. The function for predicting whether a molecule $x^\star \in \mathcal{X}$ has the desired property or not can be found by projecting its feature vector onto the maximal margin hyperplane. This is done by computing $\text{sgn}(\mathbf{w} \cdot \phi(x^\star) + b)$, or equivalently

$$g(x^\star) = \text{sgn}\left(y_i + \sum_{k=1}^{n} \alpha_k y_k \left(\phi(x_k) \cdot \phi(x^\star) - \phi(x_k) \cdot \phi(x_i) \right) \right). \tag{24}$$

Note that a feature vector $\phi(x_k)$ only contributes to the hyperplane when the corresponding coefficient α_k is non-zero, and hence is called a *support vector*. In fact, it can be shown that all such support vectors lie exactly at a distance ρ from the hyperplane (Mohri et al., 2012).

The hyperplane in (22) and (23) is calculated entirely from the training data, and may not perform well when tested against other molecules. In order to verify the accuracy of the model derived above, it needs to be shown that the generalization error,

$$r(g(X)) = P(g(X) \neq Y), \tag{25}$$

where X is a randomly chosen molecule from $\mathcal{X}$ with label Y, is small when g is calculated according to (24). In fact, it is known that the maximal margin hyperplane minimizes a so-called complexity bound on (25), providing that the training data are selected independently and identically from $\mathcal{X}$ (see (Mohri et al., 2012) for details). This provides the theoretical justification for the choice of the maximal margin hyperplane.

Despite its sound theoretical footing, the classical SVM method as presented above is rarely justified for real applications, including those related to materials chemistry. The relationship between feature vectors and molecule properties is complicated, and it is unlikely that we could ever construct a set of feature vectors which leads to linear separability, regardless of the depth of our intuition. However, this problem can be elegantly circumvented by the use of the so-called *kernel trick*, which leads us to the versatile KSVM method.

5.1.2. Kernels

A kernel K is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. When the matrix $\mathbf{K} = [K(x_i, x_j)]_{m\times m}$ is symmetric positive semidefinite for any choice of $x_1, x_2, \ldots, x_m \in \mathcal{X}$, $\mathbf{K}$ is said to be a positive definite symmetric (PDS) kernel. Furthermore, we have the following important theorem: given a PDS kernel $\mathbf{K}$, there exists a Hilbert space H and a mapping $\phi : \mathcal{X} \to H$ such that

$$K(x, x') = \langle \phi(x), \phi(x') \rangle \tag{26}$$

for all $x, x' \in \mathcal{X}$, and moreover

$$h(x) = \langle h, \mathbf{K}(x, \cdot) \rangle \tag{27}$$

for all $x \in \mathcal{X}$ and $h \in H$. The property in (27) is called the *reproducing property*, and the Hilbert space H is therefore called a *reproducing kernel Hilbert space* (RKHS). Proof of this theorem can be found in (Mohri et al., 2012, Theorem 5.2). In view of the theorem, it is possible to think of feature vectors as elements of a RKHS (i.e., a Hilbert space with the reproducing property). Thus, the inner product between any pair of feature vectors could be computed using a kernel function according to (26), providing that the kernel function associated with this RKHS is known. This is a powerful observation: as can be seen from Eqs. (22)–(24), the maximal marginal hyperplane can be computed from these inner products (and hence from a kernel) alone, without handling the feature vectors explicitly. This is the essence of the kernel trick.

Moreover, by thinking in terms of a RKHS, it is possible to work implicitly with high-dimensional feature vectors by starting with low-dimensional ones. This is because the PDS kernels associated with high-dimensional RKHSs can often be written in terms of low-dimensional vectors. For example, suppose that as a first step in our ML research we proposed to use two-dimensional feature vectors $f(x_i) = (f_1(x_i), f_2(x_i))$ to describe the molecules. Upon consulting with a materials chemist, we would probably be told that such a low-dimensional representation would be inadequate, and hence that we should consider a high dimensional RKHS H_1 instead. Suppose that we guessed that H_1 was associated with the PDS kernel

$$K(x_i, x_j) = \left(f(x_i) \cdot f(x_j) + c \right)^2, \tag{28}$$

where c is a constant parameter. Noting that this kernel can also be written as $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, where

$$\phi(x_i) = \left(f_1(x_i)^2, f_2(x_i)^2, \sqrt{2} f_1(x_i) f_2(x_i), \sqrt{2c} f_1(x_i), \sqrt{2c} f_2(x_i), c \right), \tag{29}$$

we see that H_1 is a six-dimensional space whose feature vectors have elements of the form shown in (29). By appropriate choice of kernel, it therefore is possible to work in a high-dimensional RKHS even if one starts with a low-dimensional feature representation for the molecules. In particular, the Gaussian kernel

$$K(x_i, x_j) = \exp\left(-\frac{|f(x_i) - f(x_j)|^2}{2\sigma^2} \right), \tag{30}$$

where $\sigma > 0$ is a constant parameter (a hyperparameter), corresponds to an infinite-dimensional RKHS. This kernel allows us to implicitly work with infinite-dimensional feature vectors by means of the low-dimensional feature vectors $f(x_i)$. For this reason, the Gaussian kernel is extremely popular for building ML models. In the context of kernelized ML, we refer to the low-dimensional feature vectors $f(x_k)$ as *input feature vectors* to distinguish them from the high-dimensional feature vectors $\phi(x_k)$ belonging to the RKHS of interest.

5.1.3. Kernelized SVM

Let us now return to the linear separability assumption of the classical SVM methods, which states that the molecules in the training data $x_1, x_2, \dots, x_n$ can be separated according to their classification by a linear hyperplane. Let $\phi(x_1), \phi(x_2), \dots, \phi(x_n)$ be a set of (potentially high-dimensional) feature vectors such that the linear separability assumption holds. Supposing these feature vectors lie in a RKHS equipped with kernel K, we can use (26) to rewrite (22)–(24) as follows,

$$\sum_{k=1}^{n} \alpha_k y_k K(x_k, x) + b = 0, \tag{31}$$

$$b = y_i - \sum_{k=1}^{n} \alpha_k K(x_k, x), \tag{32}$$

$$g(x') = \operatorname{sgn}\left(y_i + \sum_{k=1}^{n} \alpha_k y_k (K(x_k, x') - K(x_k, x_i)) \right). \tag{33}$$

Here, we have employed the kernel trick, and can hence compute the maximal marginal hyperplane in the high-dimensional space even though the high-dimensional feature vectors $\phi(x_1), \phi(x_2), \dots, \phi(x_n)$ are unknown. Eqs. (31)–(33) constitute the KSVM method.

In the KSVM method, we have essentially shifted one unknown (the high-dimensional feature vectors $\phi(x_i)$) into another unknown (the kernel function associated with the RKHS to which $\phi(x_i)$ belongs). In practical ML problems, we usually consider a variety of candidate PDS kernels, each having various functional forms, various parameter values, and built from various choices of low-dimensional input features. The SVM method is tested for each candidate kernel, and the kernel leading to a hyperplane with low generalization error (estimated from a set of test data) is selected as the predictive model.

Numerous examples of the KSVM method can be found in the materials chemistry literature. A representative study (Yao et al., 2015) trained a SVM model to recognize pesticide molecules from a database of 86 organic molecules. Using input features with five dimensions and the Gaussian kernel in (30), the authors built an SVM model which achieved an 85% success rate when compared with test data. This is an impressive result considering the small size of their training set. More importantly, they applied their SVM model to 11 new molecules, and predicted that they all had pesticidal activity. Upon synthesizing and testing their results in the laboratory, they confirmed this prediction for all but two of the molecules. In another study (Packwood et al., 2017) SVM methods were used as part of a larger ML scheme to learn an interaction energy function for molecules adsorbed on a metal surface. Using a dataset consisting of pairs of molecules in various orientations, as well as Coulomb matrices to build input feature vectors, they constructed three SVMs: one to classify whether the pairs of molecules were not interacting (had interaction energy with magnitude below a cut-off value), another to classify whether the pair of molecules were undergoing an repulsive interaction, and a third to classify whether the pair of molecules were undergoing an attractive interaction. The three SVMs achieved success rates near 92%, 98%, and 98%, respectively, when tested against test data. Using these SVMs, the authors were then able to predict how the molecules should be placed on the surface in order to minimize the total energy.

5.1.4. Kernelized ridge regression

In addition to classification, it is also possible to build powerful regression models using the kernel trick. The scenarios for regression and classification are similar: we have training data containing n molecules, $x_1, x_2, \dots, x_n$, where molecule k has a label y_k attached. In contrast to the classification scenario, in regression the label y_k can take any value in $\mathbb{R}$. We therefore wish to build a function $q : \mathcal{X} \to \mathbb{R}$ which can predict the value of y for any molecule in the set $\mathcal{X}$.

To proceed, consider an RKHS with a feature map ϕ and PDS kernel K. Suppose that the feature vector $\phi(x)$ has dimension m for all $x \in \mathcal{X}$. m may be very large. Consider a hyperplane of the form $g(x) = \mathbf{w} \cdot \phi(x)$ for all $x \in \mathcal{X}$, where $\mathbf{w}$ is an m-dimensional vector. Following the ridge regression technique from statistics, we estimate $\mathbf{w}$ by minimizing the objective function

$$F(\mathbf{w}) = \lambda |\mathbf{w}|^2 + \sum_{k=1}^{n} (\mathbf{w} \cdot \phi(x_k) - y_k)^2, \tag{34}$$

where λ is a positive hyperparameter, leading to the hyperplane of equation

$$q(x) = \sum_{k=1}^{n} \alpha_k \phi(x_k) \cdot \phi(x), \tag{35}$$

where the coefficients are given by $\boldsymbol{\alpha} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda \mathbf{I})^{-1}\mathbf{Y}$, where $\boldsymbol{\alpha} \triangleq (\alpha_1, \alpha_2, \dots, \alpha_n)^T$, $\mathbf{Y} \triangleq (y_1, y_2, \dots, y_n)^T$, $\boldsymbol{\Phi}$ the $n \times m$ matrix with kth row $\phi(x_k)$, and $\mathbf{I}$ is an $n \times n$ identity matrix (Mohri et al., 2012). Substituting the feature vector inner products with the kernel K into (34) yields

$$q(x) = \sum_{k=1}^{n} \alpha_k K(x_k, x). \tag{36}$$

Accordingly, we can rewrite $\boldsymbol{\alpha}$ by

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{Y}, \tag{37}$$

with $\mathbf{K} \triangleq \boldsymbol{\Phi}\boldsymbol{\Phi}^T$. Eqs. (36) and (37) together are the predictive equations for the kernel ridge regression (KRR) method.

The KRR method therefore allows us to work implicitly in a high-dimensional space in which the property of interest is a linear function, even if the feature vectors are unknown. In practice, the KRR method runs in the same way as the KSVM method: we consider a variety of candidate PDS kernels, each having various functional forms, hyperparameter values, and each built from various choices of low-dimensional input features. The hyperplanes are tested for each candidate kernel, and the one leading to a low generalization error is selected as our predictive model. Providing that the sample data is independent and identically chosen from $\mathcal{X}$, the hyperplane obtained from minimizing (34) also minimizes a complexity bound on the generalization error (see (Mohri et al., 2012) for details).

The materials chemistry literature is replete with examples of the kernel ridge regression technique. The pioneering example is from (Rupp et al., 2012) which used KRR method to learn the atomization energy for small organic molecules from a training set of over 7,000 examples. Here, atomization energy is the energy required to break a molecule into its constituent atoms. By using feature vectors built from Coulomb matrices, as well as the Gaussian kernel in (30), the authors succeeded to build a KRR model which performed favorably against test data.

In Fig. 4, we apply the KRR method to the case of three interacting benzene molecules. Here, data was have been collected by using a technique called *ab initio molecular dynamics* (AIMD). In AIMD, the motions of the atoms are simulated by integrating Newton's equations of motion, with the potential energy calculated from DFT (Eq. (8)) at each time-step. In these AIMD simulations, the three benzene molecules were placed into a box with dimensions $15\text{Å} \times 15\text{Å} \times 15\text{Å}$ with periodic boundary conditions in all directions, and the dynamics was simulated for 10,000 time steps of length 1 fs. This resulted in a dataset consisting of 10,000 snapshots of the three benzene molecules in various locations, and the total energy of the system at each step. We refer to this as the D3 dataset. A single frame from the D3 dataset is shown in Fig. 4(A). From these snapshots we also extracted 10,000 pairs of molecules at random and calculated the total energy for each pair (resulting in a second dataset D2 consisting of pairs of molecules and their total energy). Finally, we extracted 10,000 single molecules at random and calculated their total energy (resulting in a third dataset D1 consisting of single molecules and their total energies). For each dataset, we trained KRR models using Gaussian kernels and many-body tensor representations. Many-body tensor representations used two-body geometric features (interatomic distances) Gaussian densities, delta-correlation functions, and unit weights.

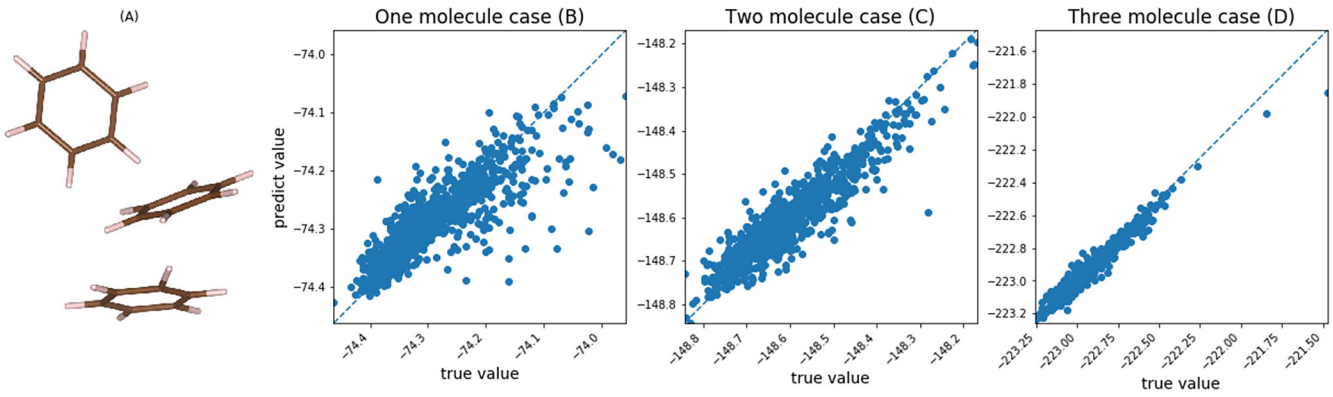


Fig. 4. Benzene interaction energy learned from KRR. The parameter $\alpha = 0$ for all the cases and $\gamma = 0.005$ for one molecule case (B) and $\gamma = 0.0035$ for the two and three molecules cases (C), (D).

In Figs. 4 (B)-(D) we show predictions of the KRR models compared to test data for the D1, D2, and D3 datasets, respectively. It can be seen that the accuracy is lowest for the case of the D1 dataset (Fig. 3B) and highest for the D3 dataset (Fig. 3D). The fact that the accuracy is highest for the D3 dataset (which consists of three interacting bodies), despite the absence of third-order structural features, suggests that such features are not necessary in order to build accurate ML models for the energy. Instead, the model performance appears to result from the sensitivity of the MBTR to variations in the dataset. The D1 dataset consists of a single benzene molecule, and its structural variations consist of minute changes in bond lengths within the molecule due to thermal fluctuations of the atom positions during the AIMD simulation. The peak positions in the MBTR therefore show only minor variations across the D1 dataset, which may be inadequate to capture the correlation between structure and energy in the dataset. On the other hand, in the D2 and especially the D3 datasets, the peak positions in the MBTR show larger variations due to the large fluctuations in distances between benzene molecules during the AIMD simulations. Sufficient variation in the peak positions in the MBTR across the dataset therefore appears to be important to adequately capture the structure–energy correlation in the dataset.

5.1.5. Support vector regression

KRR is derived from the square-loss objective function in Eq. (34). However, there is no requirement that we use a square-loss objective function, and indeed other regression methods can be derived by considering alternative objective functions. Here we briefly introduce SVR, another regression method which is occasionally applied in materials chemistry. Similar to the KRR method, the SVR method attempts to construct a hyperplane of the form $g(x) = \mathbf{w} \cdot \phi(x) + b$, where b is a constant. However, in contrast to KRR, the SVR method is derived from the objective function

$$F(\mathbf{w}) = \frac{1}{2}|\mathbf{w}|^2 + C\sum_{k=1}^{n} |y_k - (\mathbf{w} \cdot \phi(x_k) + b)|_\varepsilon \tag{38}$$

where C is a positive constant and $|\cdot|_\varepsilon$ is the so-called ε-insensitive loss, which is defined as

$$|y_k - (\mathbf{w} \cdot \phi(x_k) + b)|_\varepsilon = \max(0, |y_k - (\mathbf{w} \cdot \phi(x_k) + b)| - \varepsilon) \tag{39}$$

for any $\varepsilon > 0$. This objective function can be understood by considering the case where the feature vectors $\phi(x_k)$ are one-dimensional. In this case, the SVR method attempts to fit a tube of radius ε to the training data. All feature vectors from the training data that lie within this tube contribute no loss to the objective function. Feature vectors which lie outside of the tube contribute a loss proportional to their distance from the center of the tube. By substituting the argmin of (38) into the right-hand side of $g(x)$ yields, with some abuse of notation,

$$g(x) = \sum_{k=1}^{n} (\alpha'_k - \alpha_k) K(x_k, x) + b \tag{40}$$

as our prediction model (Mohri et al., 2012). Here, α'_k and α_k are constants and we have used the kernel trick $K(x_k, x) = \phi(x_k) \cdot \phi(x)$. b can be obtained from any point $\phi(x_j)$ such that $0 < \alpha_j < C$ as

$$b = -\sum_{k=1}^{n} (\alpha'_k - \alpha_k) K(x_k, x_j) + y_j + \epsilon, \tag{41}$$

or from any point $\phi(x_j)$ such that $0 < \alpha'_j < C$ as

$$b = -\sum_{k=1}^{n} (\alpha'_k - \alpha_k) K(x_k, x_j) + y_j - \epsilon. \tag{42}$$

In the SVR method, ε and C are free parameters which can be adjusted along with those of the kernel. The coefficients α'_k and α_k are then determined by numerical methods.

The SVR method is analogous to the SVM method in two respects. First, the tube width ε which appears in the SVR method is analogous to the margin ρ which appears in the SVM method. Second, it can be shown that the coefficients α_k and α'_k are non-zero only when the training feature vector $\phi(x_k)$ lies on the tube's surface. In analogy with the SVM method, such feature vectors can be thought of as supporting the tube. In the SVR method, these feature vectors are also referred to as support vectors.

The presence of support vectors is one of the main advantages of the SVR method, as it means all other training data can be neglected when storing the hyperplane $h(x)$. In other words, the SVR method leads to *sparse solutions*. The parameter ε therefore controls the balance between the sparsity of the solution and the accuracy of $h(x)$. Larger values of ε lead to wider tubes and hence a smaller number of support vectors, however the resulting model $h(x)$ may ignore some key points and fail to capture some important variation in the training data.

While not as popular as KRR, the SVR method has been applied to several problems in materials chemistry, including the prediction of molecule energies (Balabin & Lomakina-Rumyantseva, 2011) and chemical potency (Rodríguez-Pérez et al., 2017).

5.2. Bayesian ML

While not always evident in practice, probability theory and ML have a close connection. Indeed, many common ML methods are generalizations of methods found in statistics – a field which is firmly rooted in probability theory.

Mathematically, the notion of probability is clear: probability is defined as a special case of a measure, which is a type of set function. However, the physical interpretation of probability is ambiguous. The most common physical interpretation is the so-called frequentist interpretation, which is based upon our intuitive understanding of chance. For the case of the SVM method, $P(g(X) \neq Y)$ is understood as the fraction of times that a molecule is placed on the wrong side of the hyperplane, when tested for a large random sample of molecules.

The Bayesian interpretation of probability is an alternative to the frequentist interpretation. In the Bayesian interpretation of probability, $P(g(X) \neq Y)$ is understood as the *degree to which we personally believe* that a randomly chosen molecule would be placed on the wrong side of the hyperplane. The interpretation of probability in terms of "personal beliefs" has an important consequence. In the real world, personal beliefs can, in principle, change, in the face of new data. Thus, if we are to interpret probabilities in terms of beliefs, then we should have some method for adjusting probabilities as new data is acquired. In Bayesian probability theory, this adjustment is done by means of the Bayes' rule,

$$P(A|B) \propto P(B|A)P(A). \tag{43}$$

While this formula is valid regardless of the interpretation of probability used, it takes on a special meaning when the Bayesian interpretation is employed. For example, let A denote the event that a randomly chosen molecule appears on the wrong side of the hyperplane (i.e., $P(A) = P(g(X) \neq Y)$). Moreover, let B denote the event that a specific molecule x_i appears on the correct side of the hyperplane. Then, $P(A)$ measures our *initial degree of belief* that a randomly chosen molecule would appear on the wrong side of the hyperplane. $P(A|B)$ would then measure our *adjusted degree of belief* that a randomly chosen molecule would appear on the wrong side of the hyperplane given that molecule x_i appears on the correct side. In general, $P(A)$ and $P(A|B)$ will be different, because the fact that x_i appears on the correct side of the hyperplane constrains the ways in which the hyperplane can be oriented in space. These constraints are accounted for through the term $P(B|A)$, which scales our initial degree of belief $P(A)$ appropriately. By allowing for our personal beliefs to be adjusted as new data is obtained, the Bayesian interpretation of probability naturally lends itself to the creation of probabilistic models of the learning process via Bayes' rule (43).

Bayesian ML methods make use of the Bayesian interpretation of probability. While several Bayesian ML methods exist, arguably the most popular and successful method for materials chemistry applications is Bayesian optimization. Bayesian optimization allows us to screen databases for materials with optimal properties while reducing inefficient trial-and-error. Bayesian optimization can also be used to predict how to design materials in order to maximize a physical property of interest. Over the last few years, Bayesian optimization has become widespread in materials chemistry and has acquired a status-quo quality. Alongside the kernel methods introduced above, Bayesian optimization has arguably become one of the two main pillars for ML applications in materials chemistry at present. The Bayesian viewpoint also provides a unique take on materials chemistry and connects it with probability theory, thereby creating potential connections with fields such as stochastic analysis. Such connections could be explored by computer science and math majors. For the reasons listed above, we introduce Bayesian optimization in some detail here.

Bayesian optimization utilizes a regression method called Gaussian process regression (GPR). In the next subsection, we will review the GPR method before introducing Bayesian optimization in full. Before continuing, we must introduce some terminology. In the Bayesian interpretation of probability, $P(A)$ is referred to as the *prior probability* of A, and $P(A|B)$ is referred to as the *posterior probability* of A (or, more explicitly, the posterior probability of A given B). Often, Bayes' rule is written as the equality $P(A|B) = L(B|A)P(A)$, where $L(B|A)$ is proportional to $P(B|A)$. $L(B|A)$ is called the *likelihood function*.

5.2.1. Gaussian process regression

As with the KRR method, the purpose of the GPR method is to predict some real-valued property of each element (molecule or material) from a set $\mathcal{X}$. However, instead of building a single function for making such predictions, GPR builds a probability distribution over a space of functions. Functions to which the probability distribution gives more weight are predicted to have lower generalization errors.

Let $\mathcal{X}$ be a collection of molecules and consider a function $r : \mathcal{X} \to \mathbb{R}$ that we wish to estimate. Here, r corresponds to the physical property of interest. We suppose that r belongs to a space of functions which are smooth and finite for all molecules in $\mathcal{X}$. Suppose that we have a sample of n molecules $x_i, x_j, \dots, x_k \in \mathcal{X}$ for which it is known that $r(x_i) = y_i$, $r(x_j) = y_j, \dots, r(x_k) = y_k$. Now consider another set of m molecules $x_\alpha, x_\beta, \dots, x_\gamma \in \mathcal{X}$ for which the values of $r(x_\alpha), r(x_\beta), \dots, r(x_\gamma)$ are unknown. Let the notation $\mathbf{r} \in \delta\mathbf{v}$ denote the event where the point $\mathbf{r} = (r(x_\alpha), r(x_\beta), \dots, r(x_\gamma))$ is contained in a cube centered at point $\mathbf{v} = (v_\alpha, v_\beta, \dots, v_\gamma)$ and with side lengths δ. Using this notation, we can write

$$\begin{aligned} &P(\mathbf{r} \in \delta\mathbf{v} | r(x_i) = y_i, \dots, r(x_k) = y_k) \\ &= L(r(x_i) = y_i, \dots, r(x_k) = y_k | \mathbf{r} \in \delta\mathbf{v}) P(\mathbf{r} \in \delta\mathbf{v}). \end{aligned} \tag{44}$$

as our posterior probability for the event $\mathbf{r} \in \delta\mathbf{v}$. In (44), the left-hand and right-hand probabilities are, respectively, the posterior and prior probabilities for the event $r \in \delta\mathbf{v}$, and L is the likelihood function. We can obtain a more useful expression in a heuristic way, by letting $\delta \to 0$, in a formal sense, and rewriting (44) in terms of probability density functions. This gives us

$$p(v_\alpha, \dots, v_\gamma | y_i, \dots, y_k) = \ell(y_i, \dots, y_k | v_\alpha, \dots, v_\gamma) \pi(v_\alpha, \dots, v_\gamma) \tag{45}$$

where p denotes the posterior probability density function, π is the prior probability density function and ℓ is the likelihood. In the GPR method, the prior probability density π is assumed to be the multivariate Gaussian probability density

$$\pi(v_\alpha, \dots, v_\gamma) = \frac{1}{(2\pi)^{m/2} |\mathbf{K}|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{v} - \boldsymbol{\mu})^T \mathbf{K}^{-1} (\mathbf{v} - \boldsymbol{\mu}) \right). \tag{46}$$

In (46), the $m \times 1$ column matrix $\boldsymbol{\mu}$ is called the *mean vector*, and the $m \times m$ matrix $\mathbf{K}$ is called the *covariance matrix*. $|\mathbf{K}|$ denotes the determinant of $\mathbf{K}$. In Eq. (46), $\mathbf{v}$ is treated as an $m \times 1$ column vector. Thus, if $\pi(v_\alpha, \dots, v_\gamma)$ is particularly large, it means that we have a high degree of belief that the $r(x_\alpha) = v_\alpha$, $r(x_\beta) = v_\beta$, ..., and $r(x_\gamma) = v_\gamma$ for the unknown function r, in the absence of any data.

While the multivariate Gaussian density is assumed mainly for mathematical convenience, it comes with an additional advantage: it allows us to express our personal beliefs about the function r via the mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{K}$. If we let μ_α be our intuitive guess for the value of $r(x_\alpha)$, then we can set $\boldsymbol{\mu} = (\mu_\alpha, \mu_\beta, \dots, \mu_\gamma)$. Likewise, if we let $K_{\alpha\beta}$ measure the degree to which we believe $|r(x_\alpha) - r(x_\beta)|$ to be small, then we can write $\mathbf{K} = [K_{\alpha\beta}]_{m \times m}$. The elements of the covariance matrix therefore correspond to our intuitive beliefs about the smoothness of r across the space $\mathcal{X}$.

In the GPR method, the likelihood density ℓ is also written in terms of a Gaussian probability density (see (Packwood, 2017) for details). The posterior density in (45) then becomes a product of two Gaussian densities and can be computed analytically via matrix manipulations. The result is

$$p(v_\alpha, \dots, v_\gamma | y_i, \dots, y_k) \propto \exp\left(-\frac{1}{2} (\mathbf{v} - \boldsymbol{\mu}^\star)^T (\mathbf{K}^\star)^{-1} (\mathbf{v} - \boldsymbol{\mu}^\star) \right). \tag{47}$$

In (47), the *posterior mean vector* and *posterior covariance matrix* are defined as

$$\boldsymbol{\mu}^\star = \boldsymbol{\mu} + \mathbf{K}_{\alpha:\gamma,i:k} \mathbf{K}_{i:k,i:k}^{-1} (\mathbf{y}_{i:k} - \boldsymbol{\mu}_{i:k}) \tag{48}$$

and

$$\mathbf{K}^\star = \mathbf{K} - \mathbf{K}_{\alpha:\gamma,i:k} \mathbf{K}_{i:k,i:k}^{-1} \mathbf{K}_{i:k,\alpha:\gamma}, \tag{49}$$

respectively, where

$$\mathbf{K}_{\alpha:\gamma,i:k} = \begin{bmatrix} K_{\alpha i} & K_{\alpha j} & \cdots & K_{\alpha k} \\ K_{\beta i} & K_{\beta j} & \cdots & K_{\beta k} \\ \vdots & \vdots & \ddots & \vdots \\ K_{\gamma i} & K_{\gamma j} & \cdots & K_{\gamma k} \end{bmatrix}, \tag{50}$$

$\mathbf{K}_{i:k,i:k}$ is defined similarly, $\mathbf{K}_{i:k,\alpha:\gamma}$ is the transpose of $\mathbf{K}_{\alpha:\gamma,i:k}$, $\mathbf{y}_{i:k} = [y_i, y_j, \dots, y_k]^T$, and $\boldsymbol{\mu}_{i:k} = [\mu_i, \mu_j, \dots, \mu_k]^T$. Eqs. (47)–(49) are the

predictive equations of the GPR method. If we are interested in the posterior probability for a single molecule or material, then (47) simplifies to

$$p(\upsilon_\alpha|y_1,\dots,y_n) = \frac{1}{\sqrt{2\pi K^\star_{\alpha\alpha}}}\exp\left(-\frac{(\upsilon_\alpha-\mu^*_\alpha)^2}{2K^\star_{\alpha\alpha}}\right), \tag{51}$$

where

$$\mu^\star_\alpha = \mu_\alpha + \mathbf{K}_{\alpha,i:k}\mathbf{K}^{-1}_{i:k,i:k}(\mathbf{y}_{i:k}-\boldsymbol{\mu}_{i:k}), \tag{52}$$

and

$$K^\star_{\alpha\alpha} = K_{\alpha\alpha} - \mathbf{K}_{\alpha,i:k}\mathbf{K}^{-1}_{i:k,i:k}\mathbf{K}_{i:k,\alpha}. \tag{53}$$

In the above, $\mathbf{K}_{\alpha,i:k} = [K_{\alpha i}, K_{\alpha j},\dots,K_{\alpha k}]$ and $\mathbf{K}_{i:k,\alpha}$ is the transpose of $\mathbf{K}_{\alpha,i:k}$. In practice, Eqs. (51)–(53) are more useful than (47)–(49).

Providing that it adequately encodes our beliefs about the smoothness of the function r, the covariance matrix $\mathbf{K}$ can be chosen somewhat freely. Mathematically speaking, it is only necessary for $\mathbf{K}$ to be symmetric positive definite. A popular choice for the covariance matrix is the squared exponential function

$$K_{ij} = a\exp(-\mathbf{b}\cdot\mathbf{d}(x_i,x_j)), \tag{54}$$

where

$$\mathbf{b}\cdot\mathbf{d}(x_j,x_j) = \sum_{k=1}^{p} b_k\left(f_k(x_i)-f_k(x_j)\right)^2, \tag{55}$$

$f_k(x_i)$ and $f_k(x_j)$ are components of the (low-dimensional) input feature vectors for molecules x_i and x_j, respectively, and a, b_1, $b_2,\dots,b_p$ are positive hyperparameters. $b_1,b_2,\dots,b_p$ measure the inverse relaxation scale of the function f. Thus, if we intuitively believe that $r(x)$ should undergo rapid changes with respect to $f_k(x)$, then b_k should be given a large value (corresponding to a short relaxation scale in the kth direction). Conversely, if we believe that $r(x)$ only changes slowly with respect to $f_k(x)$, then b_k should be given a smaller value. The hyperparameters should be set carefully, as poor choices of a and (especially) b are detrimental to the performance of the GPR method. While it is possible to determine good choices for the hyperparameters by applying techniques such as empirical Bayes or cross-validation to the training data (Packwood, 2017), there is no guarantee that the resulting posterior distribution will perform well when tested against additional data. In practice, such techniques should be used as a rough guide, with the final hyperparameter values selected after considering the physics of the materials in question.

In passing, note that the posterior mean of the GPR method is actually equivalent to the predictions of the KRR method discussed in Section 5.1.4. Indeed, rewrite Eq. (36) as $q(x) = \mathbf{K}(\cdot,x)\boldsymbol{\alpha}$, where $\mathbf{K}(\cdot,x) = [K(x_1,x),K(x_2,x),\dots,K(x_n,x)]$, where $x_1,x_2,\dots,x_n$ denote the molecules in the training data. Then, substituting Eq. (37) into the expression for $q(x)$ above yields

$$q(x) = \mathbf{K}(\cdot,x)(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{Y}. \tag{56}$$

It can be seen that the KRR predictions in (56) and predictions of the GPR posterior mean in (52) are equivalent in the limit $\lambda\to 0$, providing that the prior mean vector is set to zero. This demonstrates an agreement between the frequentist and Bayesian interpretations of the regression problem, at least as far as predictions are concerned. Moreover, it can be seen that the kernel matrix, which appears in the frequentist interpretation of the regression problem, is analogous to the covariance matrix which appears above.

5.2.2. Example calculation of the posterior distribution

Here, we apply the GPR method to the situation illustrated in Fig. 5-A which considers the interaction between two nitrogen-doped pentacene molecules (N-pentacene). The two molecules are flat and lie in separate, parallel planes. The two planes are separated by a distance of 3.5Å, which means that the two molecules are close enough for a van der Waals interaction to exist between them. We wish to determine the energy-minimizing angle (defined in Fig. 5-A) for the molecule in the upper plane. Despite being a toy problem, this is a prototype for other important problems in materials chemistry, such as predicting the configuration of molecule clusters.

Fig. 5-B shows the interaction energy as a function of orientation angle (thick blue line). It was painstakingly calculated over a grid of 500 orientations, using DFT with the PBE exchange–correlation functional (Perdew et al., 1996) and van der Waals corrections (Tkatchenko & Scheffler, 2009), as implemented in the FHI-aims code (Blum et al., 2009). The energies have been normalized so that the global minimum is at 0 and the maximum energy is at 1. The blue line can be considered exact within the approximations of DFT. Two distinct energy minima are evident, and the curve is asymmetric due to the presence of the nitrogen atom. The curve is somewhat bumpy as well, which can probably be traced to the complicated shape of the electron wave functions in such molecules (Troisi et al., 2005).

While the calculations only require around 20s per orientation on our computer (a Real Computing C-Server with 32 3.2 GHz Intel Xeon processors), much longer times may be required for larger molecules or more complicated systems. In practice, it would be desirable to estimate the optimal orientation using only a small amount of training data. We therefore applied the GPR method to this dataset, using 5 randomly chosen orientations as training data. For illustration purposes, we consider one-dimensional input feature vectors $f(x) = f_1(x)$, where $f_1(x)$ is the orientation of the upper molecule. The hyperparameters a and b_1 were set to 1 and 10^{-3}, respectively. In Fig. 5-B, the posterior distribution is visualized by its mean ($\mu^\star$, thin red curve) and its standard deviation ($\mu^\star \pm \sqrt{K^\star}$, black curves). While the exact curve (blue) does not match the posterior mean so well, it mostly lies within one standard deviation of the posterior mean. The posterior distribution therefore places weight mostly on the correct parts of the function space, and could therefore be used for further analysis such as Bayesian optimization.

An important technical point needs to be mentioned. It can be seen from Eqs. (52) and (53) that $\mathbf{K}^{-1}_{i:k,i:k}$, the inverse of the covariance matrix, needs to be calculated in order to obtain the posterior distribution. In practice, the covariance matrix is often ill-conditioned, especially when the training data contains two or more similar cases (such as two close angles in the above example), which may result in numerical instabilities. One way to deal with this problem is to replace $\mathbf{K}^{-1}_{i:k,i:k}$ with $(\mathbf{K}_{i:k,i:k}+\sigma\mathbf{I})^{-1}$, where σ is a small positive constant and $\mathbf{I}$ is the $n\times n$ identity matrix (with n being the cardinality of the training set). Here, $\sigma\mathbf{I}$ acts as a regularizing term for the kernel matrix, and prevents columns from being identical to each other. In practice, σ is treated as a hyperparameter. In the calculation in Fig. 5(B), we set $\sigma = 10^{-7}$. Note that adding σI is equivalent to performing GPR on a training set of noisy data, in which the observations of the function values are in the form of $y_k = f(x_k)+w_k$, where w_k is sampled from a normal distribution with mean zero and variance σ^2. In fact, when this regularizing term is added, the GPR method becomes exactly equivalent to KRR method (see Eq. (56)).

5.2.3. Bayesian optimization

The goal of Bayesian optimization is to find the molecule $x^\star\in\mathcal{X}$ which has an optimal physical property and minimizes a cost function r. Bayesian optimization uses the posterior distribution of the GPR method to find the global minimizer $x^\star$. Given a posterior distribution (51) computed using an initial sample of training data, a single iteration of the method runs as follows. We first use an *acquisition function* to identify a molecule which is likely to improve our estimate of $x^\star$. The function is then measured for this molecule in some way (either by DFT calculation or experiment), and the posterior distribution is then computed again, this time by including the new measurement in the training data. This process continues iteratively, with the training data increasing by one molecule at each step, until the global optimizer $x^\star$ is obtained.

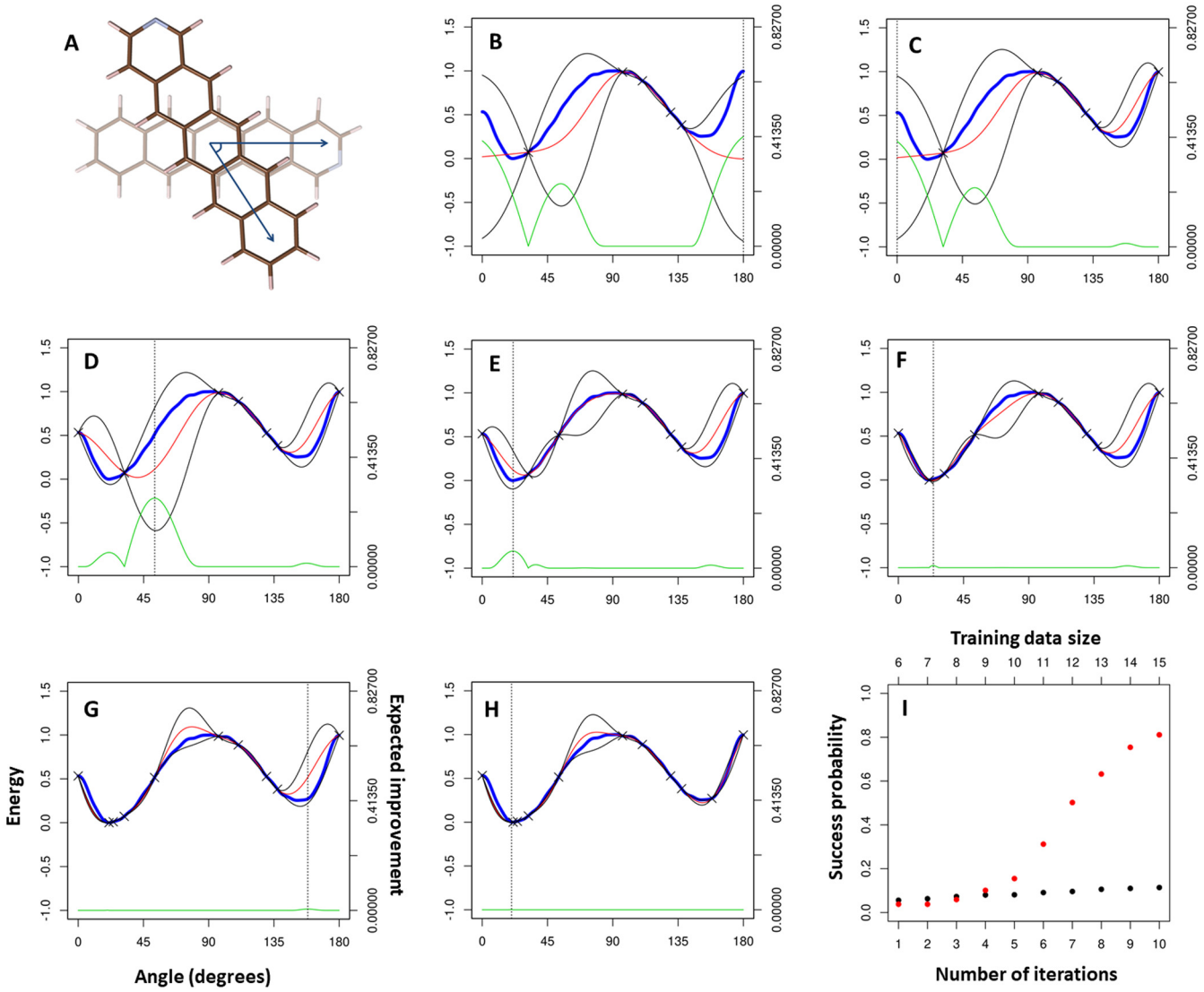


Fig. 5. Application of Bayesian optimization to a molecule alignment problem. (A) Two N-pentacene molecules lying in a stacked configuration (see text). The two molecules lie in parallel planes, with the planes parallel to the page. The darker molecule lies in the plane closer to the reader. The orientation angle of this molecule is defined. White, brown, and white parts correspond to hydrogen, carbon, and nitrogen atoms, respectively. (B) Exact interaction energy between the molecules (thick blue curve) calculated as a function of orientation angle. The energy has been normalized (see text). Thin red and black curves correspond to the posterior mean and posterior mean ± posterior standard deviation, as calculated from Gaussian process regression. Black crosses indicate training data. A green curve indicates the expected improvement. A dotted vertical line indicates the maximum of the expected improvement. (C – H) Posterior distribution plotted after 1, 2, 3, 4, and 5 iterations of Bayesian optimization, respectively. (I) Success probability of Bayesian optimization (red points) compared to uniform random sampling (see text).

Amongst the various choices of acquisition function for Bayesian optimization, the most popular is arguably the expected improvement. The expected improvement for molecule x_α is defined as

$$EI(x_\alpha) = \langle \max(r_{\min} - R(x_\alpha), 0) \rangle, \tag{57}$$

where the angular brackets indicate the expected (average) value with respect to the posterior distribution, $r_{\min}$ is the minimum function value in the training data, and R is a random function sampled from the posterior distribution. We then choose the molecule for which EI is maximized. The expected improvement therefore considers our current best guess for the minimum function value, and predicts the molecule which will improve upon the guess the most. The expected improvement can be calculated analytically with the formula

$$EI(x_\alpha) = (r_{\min} - \mu^\star_\alpha)\Phi\left(\frac{r_{\min} - \mu^\star_\alpha}{\sqrt{K^\star_{\alpha\alpha}}}\right) + \sqrt{K^\star_{\alpha\alpha}}\,\varphi\left(\frac{r_{\min} - \mu^\star_\alpha}{\sqrt{K^\star_{\alpha\alpha}}}\right), \tag{58}$$

where $\Phi(u)$ and $\varphi(u)$ are the normal distribution function and normal density function, respectively, evaluated at point u. Both functions can be accessed within interpreted programming environments such as Python and R. Proof of Eq. (58) can be found in (Packwood, 2017).

Continuing with our example above, we can use Bayesian optimization to predict the energy-minimizing orientation for the upper N-pentacene molecule. The green curve in Fig. 5-B is the expected improvement. It maximizes at 180 degrees, and therefore we perform the DFT calculation for this orientation and update the training data by adding this result to it. Fig. 5-C plots the posterior distribution again, this time calculated from the updated training data. It can be seen that the mean of the posterior distribution matches the exact blue curve a little more closely. The expected improvement now maximizes at 0 degrees. Continuing in this manner (Figs. 5-D, 5-E), we see that the global minimum is essentially reached after 4 iterations of Bayesian optimization, corresponding to 9 training data points. This requires 9 DFT calculations and hence a computational time of around $9 \times 20s = 3$ minutes (excluding the negligible time required to calculate the posterior distribution and expected improvement). In subsequent steps, the Bayesian optimization algorithm searches in the vicinity of the global minimum and attempts to improve on the result (Figs. 5-G,

5-H). In Fig. 5-I, we plot the success probability (the probability that the minimum energy in the training data is less than 0.01, and hence sufficiently close to the global minimum). This probability was calculated by repeating Bayesian optimization 1000 times with the same hyperparameters as before, but with the 5 initial training data points chosen randomly each time. For the first few iterations of Bayesian optimization, the success probability (red points) is comparable to the result obtained when orientations are chosen at random (black points). However, after about 6 iterations, the success probability for Bayesian optimization increases, reaching over 80% after 10 iterations compared to only around 10% for random sampling. Thus, Bayesian optimization is clearly the preferred method for this case.

It should be noted that Bayesian optimization acquires information from the domain $\mathcal{X}$ globally before converging to the global minimum. This behavior is called exploration–exploitation. In the example above, the exploratory behavior dominates for the first few iterations (where data is acquired from across the domain and the success probability is relatively low). Once sufficient data is acquired to estimate the location of the global minimum, Bayesian optimization switches to exploitation behavior as it hones-in on the global minimum. This behavior is in contrast to that of gradient optimizers, which rely on local derivatives to direct the search towards the nearest local minimum. For the case of the data above, such a gradient optimizer may descend into the shallow right-hand local minimum if the initial conditions are not chosen appropriately. The exploration–exploitation behavior of Bayesian optimization is therefore an enormous advantage when multiple minima are present across the domain.

6. Emerging learning approaches

Until now, our review has focused on the status quo of ML applications to materials chemistry. In doing so, we hope to have prepared the readers for engaging with the vast literature on this field. In this section, we introduce a few ML methods which are emerging in this field. Of the numerous new approaches which are entering this field, we discuss the ones which appear particularly receptive to the unique skills of computer science and math majors.

6.1. Ensemble methods

Ensemble methods employ the idea of combining the predictions of several base estimators to improve the generalizability and robustness over single estimators. Ensemble methods are classified into *bagging* (bootstrap aggregation), *boosting*, and *stacking* methods.

The main principle of the bagging method is to build several estimators independently from each other on bootstrapped subsets of the training set and then taking (weighted) averages of the predictions. This averaging process yields a more accurate estimate than each individual predictor. Random forest (RF) is an example of bagging methods.

The boosting approach is based on building estimators sequentially to reduce the bias of the combined estimators. The underlying rationale is combining several weak models to produce a powerful ensemble. Examples of boosting methods are adaptive boosting (Adaboost) and gradient boosted trees (GBT).

Both bagging and boosting methods, in most of the cases, use a single base learning algorithm, so that we have *homogeneous* weak learners that are trained over different sets. However, there exist some methods that utilize different types of base learners, resulting in *heterogeneous* ensemble models.

Stacking methods belong to the above-mentioned heterogeneous ensemble models that employ heterogeneous weak learners. These learners are trained in parallel and their predictions are then combined to form new feature vectors to train a new ML model called the meta-model which, in practice, can be any ML model. For example, for a regression problem, we could choose SVR, KKR, RF, and GBT models as weak learners. For each of these learners, the prediction over the dataset will be collected into a column of a new feature matrix. This feature matrix along with the original target vector are used to train a meta-model. This two-level stacking method can be considered as a generalization of the bagging method based on the above-mentioned four weak learners, where the deterministic weights for the average are replaced by the ones calculated from training the meta-model. Generalization to multi-level stacking methods is straightforward.

Ensemble learning (EL) methods mentioned above are often in the top ranking of many ML competitions, including Kaggle.[14] For Python user, many EL methods are implemented in Scikit-Learn library (Pedregosa et al., 2011). For readers who are not familiar with programming, we recommend using PyCaret,[15] an easy-to-use ML library where a few line of code is required for constructing ML models. Both libraries are open source.

It should be noted that EL methods are essentially different from federated learning (FL) approaches described in Section 6.3. While FL is implemented by the same ML model at each local system with different subsets of data and different model parameters, EL utilizes distinct ML models with the same dataset.

Before reviewing RF and GBT, we give a brief introduction to the decision tree method which is used as the base learner in both methods.

6.1.1. Decision tree

Decision tree is a non-parametric supervised learning method used for classification and regression. This method is very popular in ML and data mining due to its intelligibility and simplicity. The method aims to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A decision tree model is constructed by recursively partitioning the feature space into smaller and smaller subspaces, so that the samples within a subspace having similar target values can be more easily identified. The final tree will contain decision nodes and leaf nodes. A decision node has two or more branches, each represents values for the feature under testing, while the leaf node represents a decision on the target. The topmost decision node is called the root node.

Suppose that node m contains a set, denoted by Q_m, of N_m samples. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m, the data is partitioned into two subsets:

$$Q_m^{left}(\theta) = \{\mathbf{x} = \{x_1, x_2, \dots, x_j, \dots, x_n\} \in Q_m | x_j \le t_m\},$$
$$Q_m^{right}(\theta) = Q_m \backslash Q_m^{left}(\theta).$$

The quality of a candidate split of node m is then computed using an impurity function for classification problem or loss function for regression problem:

$$G(Q_m, \theta) = \frac{N_m^{left}}{N_m} H(Q_m^{left}(\theta)) + \frac{N_m^{right}}{N_m} H(Q_m^{right(\theta)}).$$

Here, N_m^{left} and N_m^{left} are respectively the cardinalities of Q_m^{left} and Q_m^{right}; H is the metric for measuring the best split in each node. For classification problem, the common metrics are Gini impurity or information gain (based on entropy). In regression problems, MSE or MAE is often used as such metric. The parameter $\theta = (j, t_m)$ is then selected at each node such that the corresponding split candidate minimizes the impurity, i.e.,

$$\theta^\star = \underset{\theta}{\operatorname{argmin}}\, G(Q_m, \theta).$$

Subsequently, the above process is repeated for subsets $Q_m^{left(\theta^\star)}$ and $Q_m^{right}(\theta^\star)$ until the maximum allowable depth is reached.

The prediction value at each leaf node will be assigned to be the average value of all the samples on that node for the regression problem. The label for each leaf in a classification tree is a specific, deterministic class or a probability distribution over the classes. Mathematically, a

[14] https://www.kaggle.com/competitions
[15] https://pycaret.readthedocs.io/en/latest/

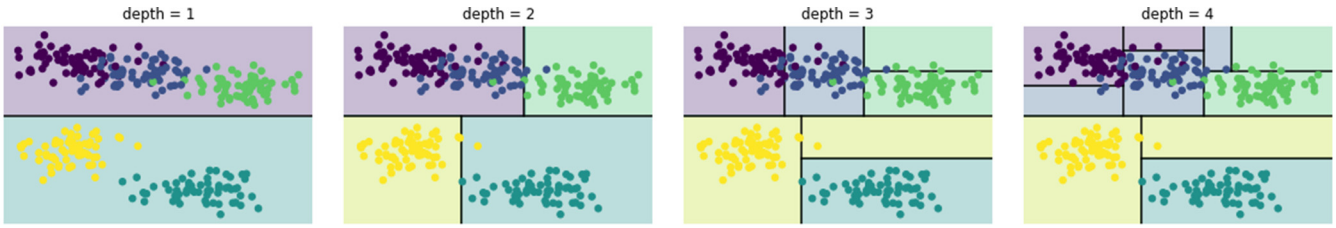


Fig. 6. Step by step splitting of the feature space for decision tree classification.

tree can be seen as a piecewise constant approximation where the value on each subspace created from the splits is the average of target values of the test sample on that subspace. Fig. 6 gives an illustration of a tree learned from artificial data shown in dots through four splitting steps. This is an illustration of a classification problem. The dataset is a union of 5 clusters each of which contains 60 normally distributed points centered randomly in the two-dimensional space. Clusters are demonstrated by different colors. A split θ is depicted by a line (or hyperplane in multidimensional space). The tree becomes more and more complicated when the depth increases. The tree that goes deeply tends to learn highly irregular patterns, thus causes overfitting, i.e., has low bias but very high variance.

6.1.2. Random forest

RF is one type of EL methods built on decision trees. The RF model takes the average prediction of multiple decision tree models trained on the whole or different subsets of the training set and/or on the whole or different subset of features. This method, like other bagging methods, helps to reduce the variance, therefore overcomes the overfitting of each decision tree component, if any. Generally, the performance boost in the final model is achieved at the expense of small bias increase and some loss of interpretability. The training and prediction phases of RF are very fast due to the simplicity of the underlying decision trees. They can be parallelized easily because of the entire independence of the individual trees. Furthermore, this is an extremely flexible nonparametric model. In many cases, it outperforms other methods which are under-fitted on some databases or tasks.

6.1.3. Gradient boosted tree

Decision trees of a fixed size are also commonly used as base learners for gradient boosting. The method is built as an additive model in a forward stage-wise fashion.

Suppose that we have a training set consisting of n samples. Let x_i, y_i, respectively be the feature vector and target value of the ith sample in the training set. Let $L(x, y)$ be an arbitrary differentiable loss function, such as least squares for regression, binomial deviance, or exponential loss for classification. The model is initialized with a constant value

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum L(y_i, \gamma).$$

At the kth step, let $h_k(x)$ is the decision tree fitted to pseudo-residues

$$r_{ik} = -\left[\frac{\partial L(y_i, F_{k-1}(x_i))}{\partial F_{k-1}(x_i)}\right] \qquad \text{for } i = 1, 2, \dots, n.$$

In other words, we train $h_k(x)$ on the training set $\{(x_i, r_{ik})\}_{i=1}^n$. This means that at each iteration, a tree is fitted on the negative gradient of the given loss function of the training samples. The model is updated along the way as

$$F_k(x) = F_{k-1}(x) + \gamma_k h_k(x),$$

where γ_k is chosen such that it minimizes the loss function

$$\gamma_k = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_{k-1}(x_i) + \gamma h_k(x_i)).$$

6.2. Reinforcement learning

Reinforcement learning (RL) is based on the principle of dynamic programming, where one or multiple agents dynamically interact with an environment (see Fig. 7(a)) and maximize the returned rewards so as to derive an optimal policy (or control actions, see Fig. 7(b)). The environment in RL is usually formulated as a Markov decision process (MDP), where a finite number of states is assumed and there is a probability to switch from one state to another.

One important and interesting feature of RL is that it is a model-free approach. In other words, one does not need to assume a system model, but can just apply an action and observe the returned reward from the environment to decide the next action.

We use the RL notions from the work of Sutton and Barto for multiple agents, as described in Fig. 7(a) (where for simplicity we focus our attention on the mathematical description for the one-agent model). The agent and the environment interact with each other, where at each time step $t = 0, 1, 2, \dots$, the agent observes the environment state S_t based on which it chooses a suitable action A_t. Upon such action, the agent receives a reward R_{t+1} (which is a real number) and a new environment state S_{t+1}. The essence of RL is therefore to find the set of optimal actions A_t so as to maximize the total reward, starting from an initial action A_0 and an initial environment state S_0. Note that the agent in general does not know exactly the model of the environment, i.e. how S_{t+1} depends on S_t and A_t, hence it needs to learn its actions based on the observed states S_t and received rewards R_t. This is also a merit of RL as a *model-free* approach.

Denote the sets of actions, states and rewards by $\mathcal{A}_t, \mathcal{S}_t$ and $\mathcal{R}_t$, respectively. Suppose that those sets contain finite numbers of elements which can be randomly switched from one to another. Then the probability for the environment state and reward to occur at the time step t, given the previous environment state and agent action, is

$$p(s', r|s, a) \triangleq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}. \tag{59}$$

The sequence of the probability function $p(\cdot)$ in (59) is often assumed to be described by a MDP.

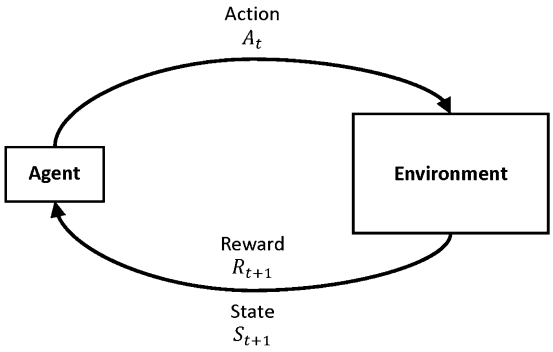
In RL framework, a *discount rate* $\gamma \in [0, 1]$ is usually employed, and the goal of the agent is to maximize the expected discounted return,

$$G_t \triangleq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

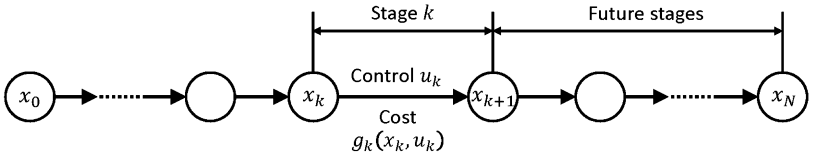
which reflects how the agent appreciates the future rewards. Since the agent must determine the actions to achieve the above goal, it needs to find the best *policy* to move at the next time step. Denote such policy by π and the probability of taking action a in state s under policy π by $\pi(a|s)$. Consequently, the value function of a state s under a policy π, denoted $v_\pi(s)$, is defined as

$$v_\pi(s) \triangleq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right\}, \tag{60}$$

where $\mathbb{E}_\pi$ denotes the expected value with respect to π. This function is also called the *state-value function* for policy π. In a similar manner,



(a) General description of reinforcement learning (Sutton and Barto, 2020).



(b) Reinforcement learning seen from decision and control viewpoint (Bertsekas, 2021).

Fig. 7. Illustration for reinforcement learning (Bertsekas, 2021; Sutton & Barto, 2020).

we can define the *action-value function* $q_\pi(s,a)$ for policy π as follows.

$$q_\pi(s,a) \triangleq \mathbb{E}_\pi[G_t|S_t=s,A_t=a] = \mathbb{E}_\pi\left\{\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle| S_t=s,A_t=a\right\}. \tag{61}$$

Denote

$$v_*(s) \triangleq \max_\pi v_\pi(s)\ \forall\, s\in\mathcal{S},\ \ q_*(s) \triangleq \max_\pi q_\pi(s,a)\ \forall\, s\in\mathcal{S}, a\in\mathcal{A}. \tag{62}$$

Noting that $G_t = R_{t+1}+\gamma G_{t+1}$, one can obtain

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)(r+\gamma v_*(s')) \tag{63}$$

and

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)(r+\gamma \max_{a'} q_*(s',a')) \tag{64}$$

which are the Bellman optimality equations revealing how the values of a state and an action depend on that of future states and actions. Those equations show a strong relation of RL with dynamic programming, where one wants to find the optimal policy to maximize the state-value or action-value function by solving a Bellman equation. In fact, having $q_*(s,a)$ makes it easier to choose actions.

In general, it is not easy to solve the Bellman optimality equations above, and thus usually approximation methods are needed to find the optimal policies. Policy iteration and value iteration are such methods (see (Sutton & Barto, 2020) for details), but they require a complete model of the environment and expensive computational expense. To overcome the above drawbacks, Monte Carlo methods can be utilized, where sample transitions of the MDP can be simulated and then averaged in order to solve the Bellman optimality equations. Such Monte Carlo simulations are based on experience, hence they are an example of *learning from experience* methods. Letting V be an estimate of v_π for states S_t, the Monte Carlo methods update becomes

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{65}$$

with a constant step-size $\alpha > 0$.

Another remarkable class of methods for RL is called *temporal-difference (TD) learning*. The simplest form of TD learning, called TD(0), updates

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]. \tag{66}$$

The advantage of TD methods over Monte Carlo methods is that they learn to act from the current actions, and do not need to wait until the end of a sample process as in Monte Carlo methods. The popular *Q-learning* algorithm is in fact an off-policy TD method, i.e. it is independent of the policy being followed. Q-learning is represented by the following update

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)], \tag{67}$$

where $Q(S_t,A_t)$ is the action-value function to directly approximate the optimal q_*.

Finally, we introduce a type of RL method called *policy gradient*, in which parameterized policies are learned. Denote $\theta\in\mathbb{R}^{d'}$ the policy parameter vector, $\pi(a|s,\theta) = \Pr\{A_t=a|S_t=s,\theta_t=\theta\}$ the probability that action a is taken at time t given that the environment is in state s at time t with parameter θ. Then policy gradient methods aim to maximize a scalar performance index, denoted by $J(\theta)$, by updating the policy parameter vector θ in a gradient ascent, as follows,

$$\theta_{t+1} = \theta_t + \alpha\nabla J(\theta_t), \tag{68}$$

where $\alpha > 0$ is a step-size. Usually the gradient $\nabla J(\theta_t)$ is not known exactly, hence a stochastic approximation of it, denoted by $\widehat{\nabla} J(\theta_t)$, is employed. If the value functions are also utilized in the learning process, then the methods are called *actor-critic*, where the former term refers to the policy learning and the latter term indicates the value function learning.

There have been only a few papers on utilizing RL for materials (Liu et al., 2020a; Mills et al., 2020; Neil et al., 2018; Popova et al., 2018). In (Liu et al., 2020a), RL was combined with DL to obtain a deep reinforcement learning (DRL) approach for materials synthesis at the atomic-scale (in particular thin-films using atomic layer deposition) and the Stein variational policy gradient (SVPG) method was employed to train agents for optimizing specific materials descriptors. Lattice-based Monte Carlo simulations have been performed as an environment for training agents, in which an initial atomic configuration and rate parameters for distinct events which can occur were provided as inputs. Consequently, the simulations proceed with the events based on their probabilities. The state, action space, and rewards for agents are 2D images of surfaces of the thin-film growth process, the rate of deposition of atoms, temperature, and a reward function with the surface roughness as its input.

Note that we do not introduce the background of DL and its uses in materials chemistry, in order to keep the current review more concise

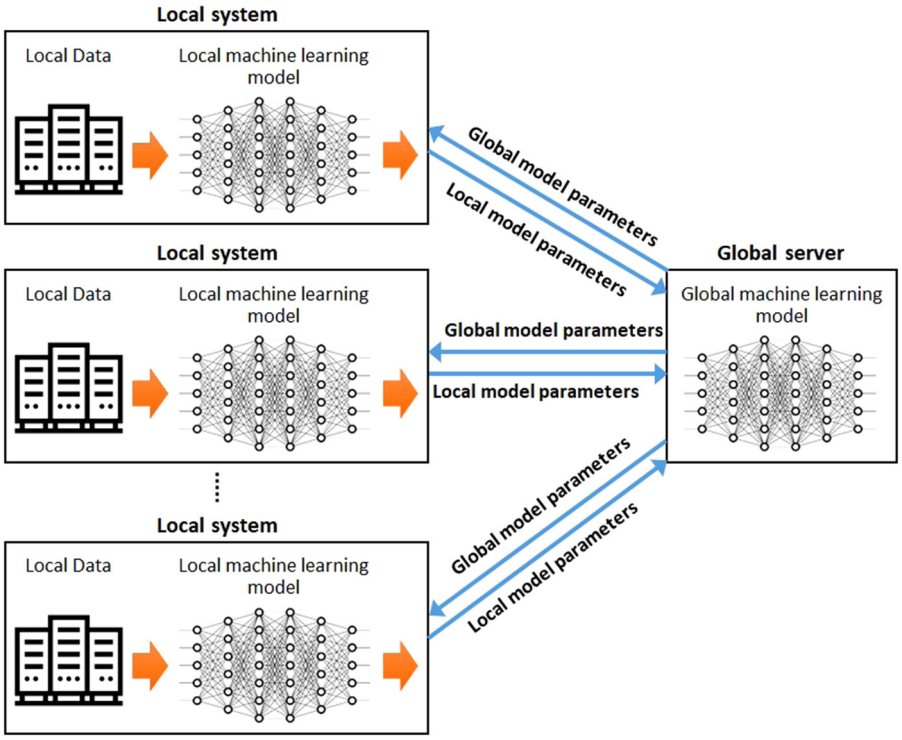


Fig. 8. Illustration for the operation of federated learning.

and focused. Interested readers can refer to other works for such purposes, see e.g., (Chibani & Coudert, 2020; Elton et al., 2019).

In (Neil et al., 2018), 19 new open-source benchmarks have been proposed to explore the abilities of RL in *de novo* molecular design. The SMILES notation (Section 2.1) has been employed to represent molecular structures as labeled graphs with atoms as vertexes and bonds as edges, and to encode them as sequences of symbols. Different reward functions have been used in (Neil et al., 2018) including the SMARTS and Tanimoto similarity defined for fingerprint vectors of molecules based on the Functional Connectivity Fingerprint Counts (FCFC4) method for single-objective maximization, and a more complex function for multi-objective maximization.

DRL and SMILES have also been employed in (Popova et al., 2018) for *de novo* materials discovery. Here an approach named ReLeaSE (Reinforcement Learning for Structural Evolution) was proposed in which one generative and one predictive deep neural network (DNN) were employed to train the system at two phases of the proposed approach. In the first phase, both DNNs were separately trained using supervised learning. In the second phase, the generative DNN acted as a centralized agent, while the predictive DNN serves as the environment in the RL framework. The set of actions was defined as an alphabet containing letters and symbols utilized to define canonical SMILES strings that are most commonly used to encode chemical structures, whereas the set of states consisting of all possible strings in the alphabet with lengths from zero to some fixed value T. The reward function was set as a function of the predicted property of the material.

To this end, possibly what hinders the widespread adoption of RL in materials discovery is formulating the problem into a rigorous mathematical framework and choosing effective reward functions. We anticipate that the combination of RL and DL, i.e., DRL may become dominant methodology in this direction.

6.3. Federated learning

Unlike conventional ML approaches where local data from local systems are shared with a data server (distributed ML) or with other local systems (decentralized ML), federated learning (FL) proposes to share local model parameters without the need to share local data, as illustrated in Fig. 8. As a result, the data privacy of local systems is protected, which is critical in a number of applications, e.g., in medical and health science where data of patients are confidential (Sheller et al., 2020). Moreover, the number of model parameters are often much smaller than the amount of data to be shared, therefore the amount of information exchange in FL is lighter than that of conventional ML approaches. Decentralized FL architectures consisting of local systems directly sharing local model parameters with one another and operating without the need for a central server hosting a global model are also available.

Federated averaging, a simple FL implementation based on the stochastic gradient descent (SGD) algorithm, is described in (McMahan et al., 2017). Suppose that there are K local systems (or clients) collaborating to learn an ML model whose parameters are denoted by w and suppose that an objective function in form of a finite-sum, $f(w) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(w)$, needs to be minimized, i.e.,

$$\min_{w\in\mathbb{R}^d} f(w). \tag{69}$$

Let $k = 1, 2, \ldots, K$ be the index for the local clients, and $\mathcal{P}_k$ be the set of indexes of data points on client k. Letting n_k denote $|\mathcal{P}_k|$, the (69) can be rewritten as

$$\min_{w\in\mathbb{R}^d} \sum_{i=1}^{n} \frac{n_k}{n} F_k(w), \tag{70}$$

where $F_k(w) \triangleq \frac{1}{n_k}\sum_{i\in\mathcal{P}_k} f_i(w)$. For details on implementing federated averaging over local clients we refer to (McMahan et al., 2017).

To the best of our knowledge, applications of FL in materials chemistry-related problems are yet to be reported. As a nascent and emerging methodology, FL seems to mainly be of interest in areas such as medical care and finance, where data protection and privacy are crucial (see, e.g., (Yang et al., 2019) for a discussion). Possible directions for the adoption of FL in materials chemistry may arise in situations where one of the parties in a collaboration belongs to a private company and therefore needs to retain the confidentiality

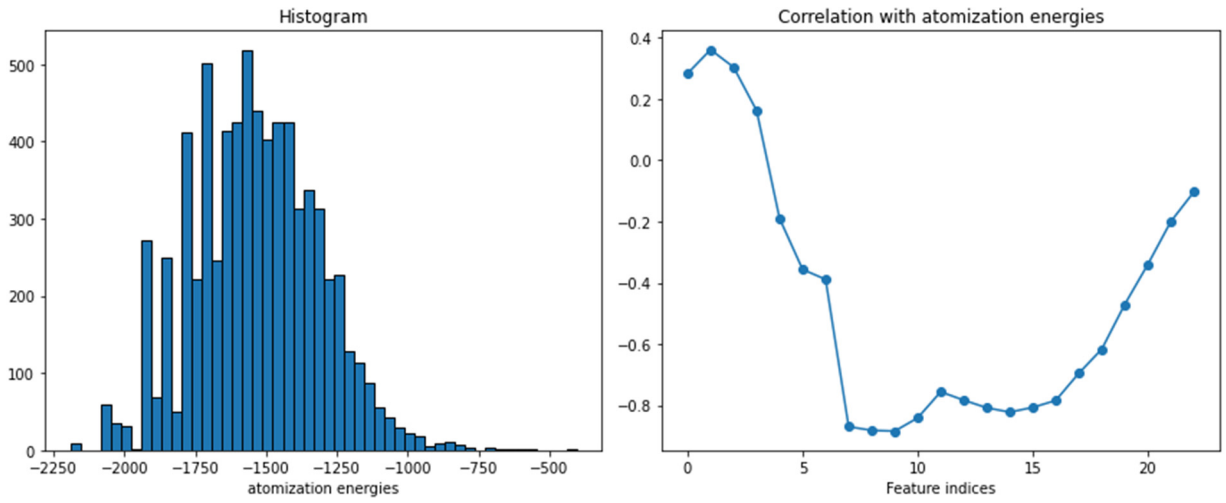


Fig. 9. Data exploration.

and ownership of data. Another direction is to combine FL with other ML techniques in order to enhancing their performance and accelerate materials discovery.

7. Illustration of ML for materials

The purpose of this section is to walk the reader step-by-step through the basic ML workflow (after feature vectors have been created and response variable data has been obtained) and compare the use of some ML methods introduced in previous sections for a specific dataset.

In order to test some of the ML models introduced in Sections 5 and 6, we train a series of ML methods using publicly available datasets. We make use of the QM7 dataset[16] which contains 7165 molecules, each composed of up to 23 atoms. QM7 dataset is a subset of the larger GDB-13 (Blum & Reymond, 2009) dataset, which consists of nearly 1 billion stable and synthetically accessible organic molecules.

The QM7 dataset provides 23×23 Coulomb matrices, their eigen-spectra (resulting in a feature matrix of dimension 7165×23), and atomization energies are computed for each molecule using DFT as implemented in FHI-aims using the PBE0 hybrid functional. The target vector (response variable values) y contains atomization energies in units of kcal/mol.

Before proceeding, we examine the data at hand more closely. Fig. 9 illustrates the distribution of atomization energies–target values (left) and the Pearson correlation of the atomization energies with each of the columns in the feature matrix X (right). We test whether the energies are normally distributed. To do so, we set up a classical hypothesis test with p-value equal to 0.05. We use D'Agostino and Pearson's test that combines skew and kurtosis to produce an omnibus test of normality (D'Agostino, 1971; D'Agostino & Pearson, 1973). Based on our calculations we discover that the null-hypothesis (that the data is not normally distributed) can be rejected.

The profile of the energy histogram is in fact right-skewed, as seen in Fig. 9. The small density in the right-side of the histogram implies a harder task for training regression models to make accurate predictions for high-energy molecules (Fernández et al., 2018). This is a manifestation of data imbalance, a problem usually defined for classification and possibly not common for regression. Due to data imbalance, the regression model will likely return sharper predictions for molecules with energy values in a neighborhood of the mean and possibly give unacceptable estimates for molecules with large energy values. Data imbalance appears to some extent in almost all real-world applications, which may be caused by biased sampling or measurement errors. Even though several methods have been proposed for dealing with imbalanced data, including simply harvesting more data, so-called up/down samplings, and transformation of the vector of target (response) variables, data imbalance remains a challenging problem (see (Krawczyk, 2016)). Decision trees often perform well on imbalanced datasets. In this methods, information from the tails can also be extracted and processed by the model.

The Pearson correlation coefficient is a classical tool for quantifying the degree of linear correlation between two sets of data. The graph on the right of Fig. 9 shows that only the first four largest eigenvalues of the Coulomb matrix have positive correlations with the atomization energy, while the others have a negative correlation. Most of them have correlations of large magnitude (0.2 or more), suggesting that they should be retained when used to train the models.

We train models using four different methods and the QM7 dataset. The methods are SVR (see Section 5.1.5), RF (see Section 6.1.2), GBT (Section 6.1.3), and KRR (Section 5.1.4). For each model, 20% of the dataset is held out as the test set, while the rest is used as the training set. Grid search and cross-validation are used to tune hyperparameters, and normalization is not applied. Scikit-learn libraries (Pedregosa et al., 2011) are used for this numerical calculation. To compare the performance of different ML methods, the mean square error (MSE) is employed for comparison.

7.1. Parameter setting

For the SVR model, a linear kernel, i.e., $K(x, y) = x^T y$, is chosen as a simple baseline model, with regularization parameter set to 1.

The RF model was built using 20-fold cross-validation. Grid search is performed for the number of estimators taking 25 points spaced evenly on the interval $(50, 150)$. We discover the optimal random forest model from the above grid as the one with 108 estimators.

GBT for regression models was performed with 100 boosting stages. The learning rate γ_k at each stage is equal to 0.1. The learning rate controls the contribution of the weak learners in the final combination. The number of boosting stages and learning rate strongly interact. Smaller values of learning rate require a larger number of boosting stages to maintain a constant training error. The loss function is chosen to be the squared loss, so that the initial prediction constant function $F_0(x)$ takes the mean of the target values in the training set.

The KRR model was built with 10−fold cross validation. Grid search is performed for regularization strength α taking 10 points spaced evenly on the interval $(10^{-10}, 10^{-5})$. The selected kernel is either a radial basis function (rbf) based kernel given by,

$$K_{\text{rbf}}(x, y) = \exp(-\gamma \|x - y\|^2),$$

[16] http://quantum-machine.org/datasets/

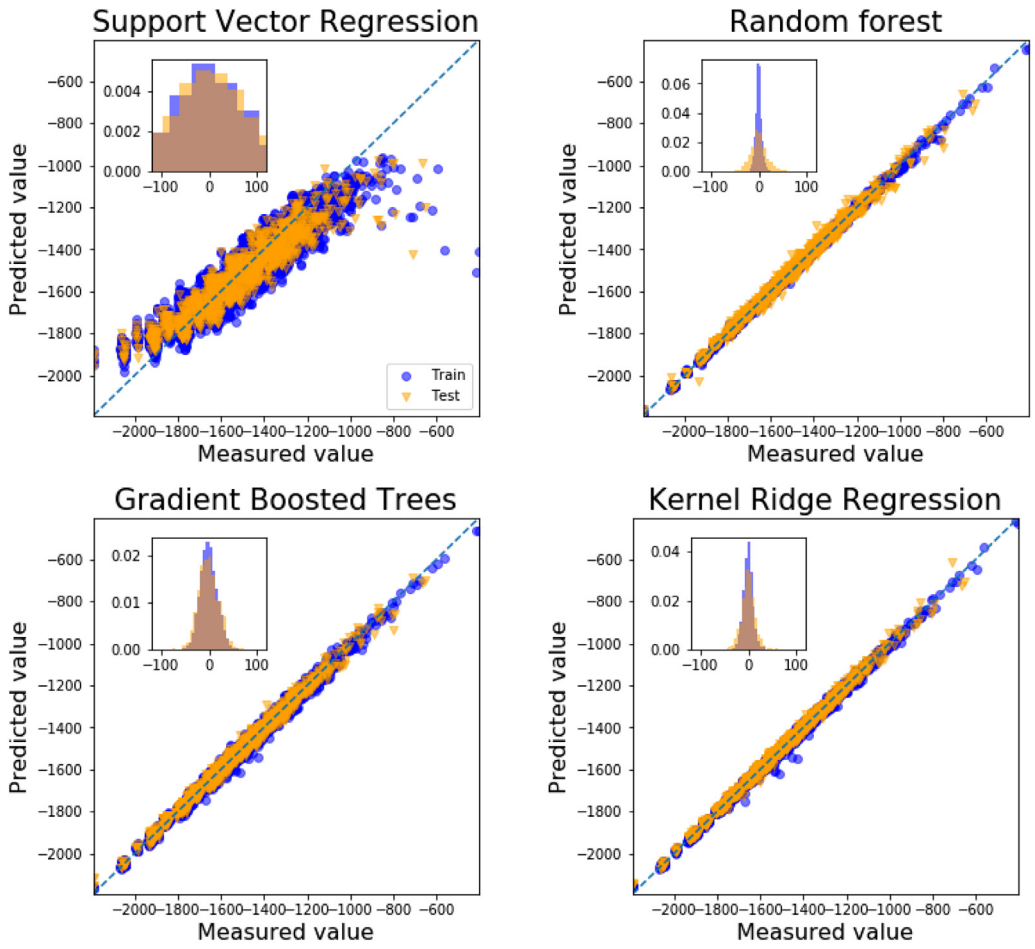


Fig. 10. Prediction results of ML methods on QM7 dataset. The dashed line indicates the predictions coincide with observations. The axes inserted within each of the graphs demonstrate the distributions of the error for the training set and the test set. See on-line version of this article for the colors.

or a Laplacian kernel given below,

$$K_{\text{Laplacian}}(x, y) = \exp(-\gamma \|x - y\|_1),$$

where $\|\cdot\|_1$ denotes the Manhattan distance. The parameter γ is set to be one of 10 points evenly sampled on a log scale in the interval $(-12, -9)$. The best KRR model from the above grid is the one with $\alpha = 3.5938$, $\gamma = 2.1544$, and with Laplacian kernel.

7.2. Numerical results

Simulation results for the employed methods are displayed in Fig. 10. After choosing the parameters and after training the model over the training set, we perform comparisons between observations and the predictions of the values of atomization energies for each molecule in the training set and test set. The observed values of the response variables are represented by the x-coordinate and the predicted values are represented as the y-coordinate.

Looking at the result of SVR, we see that there are many training points that are far from the line $y = x$, indicating that the model has a poor generalization error. We can also see that the linear kernel is too simple to capture the non-linear nature of the data. Furthermore, the effect of imbalanced data can be observed, as the data points with high energies are poorly predicted as the model is biased towards the mean.

To this end, the RF model performs the best on the QM7 dataset, followed by the KRR and GBT models with SVR being the worst performer. Their MSE measurements are respectively 113.60, 137.45, 364.58, and 7002.42. The superior performance of the RF model might be explained by the fact that it is a non-parametric model, and hence that it makes no assumptions on the distribution of the data.

8. Conclusions and outlooks

8.1. Accelerated materials discovery

Over the last decade, the integration of data science and computational chemistry and physics have resulted in new paradigms for the design of new materials. Spearheaded by the Materials Genome Initiative in the US,[17] (Liu et al., 2020b) as well as by analogous programs around the world, use of high-throughput computational methods has allowed systematic exploration of the chemical space (that is, the set containing all possible molecules that can be, in principle, created according to chemical rules (Dobson, 2004)) by overcoming the intrinsic limitations of trial-and-error approaches, in an effort at the intersection of chemistry, statistics and computer science. In this arena, statistical and ML methods can play a leading role for deducing predictive models for structure–property relationships. We therefore end this paper by providing a general discussion about how ML is used for real materials discovery.

The design of materials with targeted properties via chemical intuition and trial-and-error has traditionally been the dominant approach in materials chemistry. In the data-driven era, computational power and statistics offer an avenue for by-passing the inefficiency bottlenecks of this old approach, as well as to overcome limitations of human experience. In a nutshell, the idea consists in exploiting atomistic calculations to compute properties and structures for a limited set of

[17] https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/materials_genome_initiative-final.pdf

molecules which represents a sample of a high dimensional chemical space. Then, by using the properties of the initial set of molecules to train ML models and determine correlations among the structures and their properties, and rules for predicting new materials can be deduced.

Conceptually, the most general strategy for the identification of a material with desired target properties is the direct inspection of the chemical space and in the determination of the properties for each single molecule populating the chemical space. Such a program is exclusively based on computational brute-force and turns to be unpractical in most realistic situations. First of all, direct investigation of the chemical space entails listing a countable, possibly infinite, collection of molecules, which makes automatization an unfeasible task. Additionally, the determination of physical and chemical properties for each individual molecule, either via experimental measurement or numerical computation, can be a very demanding task in terms of processing time and resources.

A number of contributions have appeared over the last decade using accelerated strategies for materials discovery based on ML methods, rather than brute-force methods. Besides the above-mentioned work (Curtarolo et al., 2003) that paves the way to using atomistic calculations and statistics in a coordinated way to extract patterns and correlation in materials datasets (see also (Curtarolo et al., 2012)), prediction of properties of dielectric polymers based on a KRR model is illustrated in (Pilania et al., 2013). The concept in (Pilania et al., 2013) is based on a superposition principle which is proven to produce effective regression for n-block molecules ($n > 4$) obtained by assembling n molecular units taken from a set of allowed molecular blocks and by training a model over 4-block polymers. The same superposition principle for n-block dielectric polymers is exploited in combination with a strategy for modifying existing structures with the goal of designing novel materials with targeted properties in (Wang et al., 2014). Additional examples of a scheme for the discovery of polymers with high-dielectric constant crystalline structures presented in (Mannodi-Kanakkithodi et al., 2016) (see also (Sharma et al., 2014)).

As a paradigm of a strategy particularly suitable for scaling and automatization as well as testing with experimental data, we illustrate the approach of (Pilania et al., 2013) and (Wang et al., 2014) in the following. The strategy involves an evolution operation on a set of molecules, leading to sequences of generations of offsprings. An iterative scheme is set up so that the new generations of molecules undergo subsequent down selection and optimization until a class of molecules with properties sufficiently close to a prescribed target is discovered. A flow chart of the algorithm developed in (Mannodi-Kanakkithodi et al., 2016) is displayed in Fig. 11. The core of the algorithm is constituted by a ML model, trained off-line, for predictions of a list of molecular properties. The purpose of exploiting statistical regression is to allow on-the-fly predictions of properties for high volumes of molecules bypassing time-consuming DFT computations, thus guaranteeing up-scaling of the numerical platform.

A randomized set of molecules (or other types of materials) provides the initial dataset from which both a training set as well as a testing set for the ML algorithm are extracted. Chemical properties of the training set molecules are obtained also off-line with DFT. Besides the standard requirements to avoid poor generalization errors and overfitting, choice of the training set should be representative of the chemical space, so that meaningful predictions for more general molecules can be effectively performed. Training sets in (Mannodi-Kanakkithodi et al., 2016) and (Pilania et al., 2013) populated by 4-block molecules assembled from a pool of seven elementary units are shown to successfully produce predictions for properties of n-block molecules (with blocks taken from the same pool of elementary units) for n up to 8.

Translation of chemical formulas into digital strings (feature vectors) allows the necessary task of processing the information stored in the training set, so that quantitative predictions can be performed for new generations of molecules. This procedure originates from the

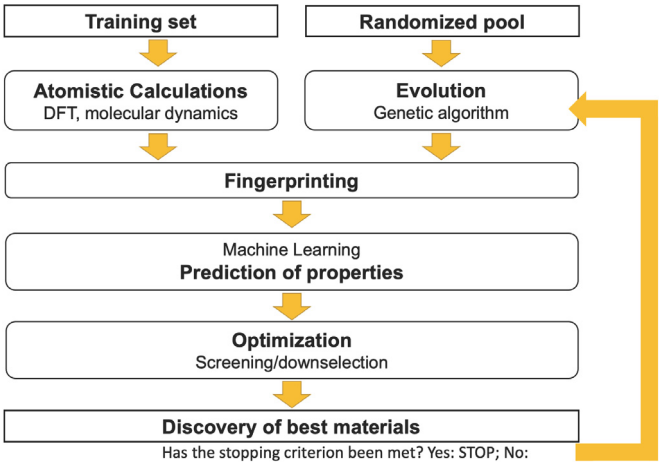


Fig. 11. Conceptualization of a simple iterative loop for the accelerated discovery of materials and structures.

assumption that a (somehow direct, although unknown) relation exists between the desired physical property and the nature of the molecule's building blocks. Correlations between the chemical properties and fingerprint entries can be unveiled by computing classical statistical descriptors (such as, for vectorial fingerprints, the Pearson correlation matrix). In this way, trends in the data can be observed and empirical rules in terms of positive (constructive) and negative (destructive) contributions of the individual chemical unit on the overall properties of interest can be deducted. Both in (Mannodi-Kanakkithodi et al., 2016) and (Pilania et al., 2013), features consist of simple enumeration of the elementary building blocks resulting in the introduction of an array (fingerprint) for each molecule. This choice responds to the requirement that molecular features should be simple to automatize, intuitive and easy to generalize, so that any molecules that can be possibly created at each iteration, given the chemical space, can be fingerprinted.

In the above references, search and discovery of new materials with desired properties is conducted via direct inspection over a pool of candidates updated dynamically at each iteration of the algorithm. Pools are composed of molecules for which properties are predicted on-the-fly via ML, thus bypassing time-consuming DFT calculations. A score, computed as a function of properties predicted via ML, is associated to each molecule. Molecules with highest scores are stored as the solution to the optimization problem. On the contrary, molecules with low score may be disregarded or undergo evolution process ultimately giving rise to a new generation and a new pool of candidates for screening in the next iteration. Design of the evolution method leverages the availability of high volumes of potential candidates for which properties can be predicted with ML, instead of being computed with DFT. An important design requirement, the evolution algorithm allows transfer of information to the future generations so that molecules with properties close to the desired target can effectively be discovered (see (Lookman et al., 2019) where an approach aimed at extracting valuable information from data uncertainty as well is shown).

A thorough, although highly inefficient, strategy for evolution is enumeration, that is, the listing of all possible molecules that can be assembled by simply considering all admissible combinations and permutations of admissible elementary blocks. For instance, in the case of n-block polymers obtained by combining a list of 7 admissible blocks as in (Mannodi-Kanakkithodi et al., 2016; Pilania et al., 2013), the number of potential candidates is exponential in n leading soon to an unmanageable flow of operations even for relatively low n even after ruling out duplicates or unfeasible candidates.

More efficient approaches for the generation of molecules or materials are based on rule-based chemical filters, random search and, most

relevant, the use of genetic algorithms (see (Jain et al., 2013a) for a comparison of the properties of the above-mentioned methods). These are a family of computational models for the generation of individuals starting from an initial set of ancestors via a cascade of cross-mutations and mating (crossover). Finally, iterations are arrested when a stopping criterion measured in terms of molecules performance (such as, for instance, molecules' score or number of molecules sufficiently close to a target score) is met.

Even after completing the above protocol, investigations of thermodynamical stability are required to determine which predicted materials are expected to be synthesizable. Alongside classical methods such as computation of formation energy or simple DFT-based structural relaxation, nowadays one often encounters other methods. One such method uses a criterion based on the energy convex hull, which is particularly relevant for solid-state materials. This prescribes a compound as being unstable if its DFT-computed ground state energy lies (sufficiently) above the convex hull of the predicted formation energies of all competing chemical phases in a chemical space (see (Akbarzadeh et al., 2007; Armiento, 2020; Kozinsky & Singh, 2021) for details).

In (Balachandran et al., 2018), a ML model for classification of perovskites was trained from a dataset of experimentally reported perovskite compounds and utilized to predict new perovskites. Thermodynamical stability of all discovered perovskites was investigated with DFT computations and convex hull analysis using data from the Open Quantum Materials Database (see Table 2). We also refer to (Li et al., 2018), where various ML models are trained for the classification of perovskite compounds as being stable or unstable based on the energy convex hull criterion, and a number of available ML models were tested for regression on the values of the energy convex hull as well as for DFT-calculated formation energies.

In (Scheleder et al., 2020), the thermodynamic stability of two-dimensional materials was assessed using a ML method for classifying into classes corresponding to low, medium, and high stability. The boundary for separating those classes was also based on the formation energy and the energy above the convex hull.

8.2. Outlooks

Control of material properties via design of nano- and micro-scale features is one of the most promising arena for applications of automatized data-driven methods and ML. However, despite the availability of vast and cheap computational resources, there remains significant rooms for major advances.

One of the main difficulties lies in the understanding of highly complicated correlations between atom-scale features and meso/macro-scale material properties. Intuition and simple empirical rules may suffice in a number of cases such as dielectrics (Pilania et al., 2013; Wang et al., 2014), and perovskites (Balachandran et al., 2011; Jain et al., 2013a). Nevertheless, the mapping between molecular features and material properties remains in general one of the central open problems in materials science. In addition to the ML methods presented here, one could also apply DNNs, fuzzy models, and expert systems to learn this mapping, just to name a few. While DNNs and fuzzy models can be used to approximate unknown and uncertain material feature–property relationships, expert systems provide a formal means to incorporate the valuable knowledge of materials chemists into the materials discovery process. While expert systems and similar methods have not been widely reported in the materials chemistry literature yet, they could serve as an intellectual framework for a genuine collaboration between materials chemists, computer science, and math majors. Having made it to the end of this review, we invite intrepid readers to pursue this direction.

CRediT authorship contribution statement

Daniel Packwood: Took part in simulation, Writing the paper. **Linh Thi Hoai Nguyen:** Took part in simulation, Writing the paper. **Pierluigi Cesana:** Took part in simulation, Writing the paper. **Guoxi Zhang:** Took part in simulation. **Aleksandar Staykov:** Took part in writing the paper. **Yasuhide Fukumoto:** Supervised the work. **Dinh Hoa Nguyen:** Conceptualize the idea, Took part in writing the paper, Supervise the work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Daniel Packwood is supported in part by JSPS KAKENHI, Japan Grant Numbers JP21K05003, JP20KK0160, JP19H04574, and JP19H00922. Pierluigi Cesana's work is supported by JSPS KAKENHI, Japan Grant Number JP21H00102. Pierluigi Cesana holds an honorary appointment at La Trobe University and is a member of GNAMPA. Dinh Hoa Nguyen is supported in part by JSPS KAKENHI, Japan Grant Number JP19K15013.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*, 716–723.

Akaike, H. (1980). Seasonal adjustment by a bayesian modeling. *Journal of Time Series Analysis*, *1*(1), 1–13.

Akbarzadeh, A. R., Ozoliņš, V., & Wolverton, C. (2007). First-principles determination of multicomponent hydride phase diagrams: Application to the Li-Mg-N-H system. *Advanced Materials*, *19*(20), 3233–3239.

Antono, E., Matsuzawa, N. N., Ling, J., Saal, J. E., Arai, H., Sasago, M., & Fujii, E. (2020). Machine-learning guided quantum chemical and molecular dynamics calculations to design novel hole-conducting organic materials. *Journal of the Physics and Chemistry A*, *124*, 8330–8340.

Aprahamian, I. (2020). The future of molecular machines. *ACS Central Science*, *6*(3), 347–358.

Armiento, R. (2020). Database-driven high-throughput calculations and machine learning models for materials design. In *vol. 968 of Machine learning meets quantum physics, Lecture notes in Physics*. Cham: Springer.

Balabin, R., & Lomakina-Rumyantseva, E. (2011). Support vector machine regression (LS-SVM)—an alternative to artificial neural networks (ANNs) for the analysis of quantum chemistry data. *Physical Chemistry Chemical Physics*, *13*, 11710–11718.

Balachandran, P. V., Broderick, S. R., & Rajan, K. (2011). Identifying the 'inorganic gene' for high-temperature piezoelectric perovskites through statistical learning. *Proceedings of the Royal Society of London, Series A (Mathematical and Physical Sciences)*, *467*, 2271–2290.

Balachandran, P. V., Emery, A. A., Gubernatis, J. E., Lookman, T., Wolverton, C., & Zunger, A. (2018). Predictions of new ABO_3 perovskite compounds by combining machine learning and density functional theory. *Physical Review Materials*, *2*, Article 043802.

Balzani, V., Credi, A., Raymo, F. M., & Stoddart, J. F. (2000). Artificial molecular machines. *Angewandte Chemie (International Edition in English)*, *39*, 3348–3391.

Bertsekas, D. P. (2021). Reinforcement learning and optimal control. In *Lecture Notes* http://web.mit.edu/dimitrib/www/RLTopics_2021_Lect1.pdf.

Bhattacharya, K., & James, R. D. (2005). The material is the machine. *Science*, *307*(5706), 53–54.

Bissell, R. A., Córdova, E., Kaifer, A. E., & Stoddart, J. F. (1994). A chemically and electrochemically switchable molecular shuttle. *Nature*, *369*(6476), 133–137.

Blum, V., Gehrke, R., Hanke, F., Havu, P., Havu, V., Ren, X., Reuter, K., & Scheffler, M. (2009). Ab initio molecular simulations with numeric atom-centered orbitals. *Computer Physics Communications*, *180*, 2175–2194.

Blum, L. C., & Reymond, J.-L. (2009). 970 Million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society*, *131*, 8732–8733.

Brown, R. D., & Martin, Y. C. (1996). Use of structure activity data to compare structure-based clustering methods and descriptors for use in compound selection. *Journal of Chemical Information and Computer Sciences*, *36*, 572–584.

Burger, B., Maffettone, P. M., Gusev, V. V., Aitchison, C. M., Bai, Y., Wang, X., Li, X., Alston, B. M., Li, B., Clowes, R., Rankin, N., Harris, B., Sprick, R. S., & Cooper, A. I. (2020). A mobile robotic chemist. *Nature*, *583*, 237–241.

Butler, K. T., Davis, D. W., Catwright, H., Isayev, O., & Walsh, A. (2018). Machine learning for molecular and materials science. *Nature*, *559*, 547–555.

Carhart, R. E., Smith, D. H., & Venkataraghavan, R. (1985). Atom pairs as molecular features in structure–activity studies: Definition and applications. *Journal of Chemical Information and Computer Sciences*, *25*, 64–73.

Carlsson, G. (2009). Topology and data. *Bulletin (New Series) of The American Mathematical Society*, *46*, 255–308.

Ceperley, D. M., & Alder, B. J. (1980). Ground state of the electron gas by a stochastic method. *Physical Review Letters*, *45*(7), 566–569.

Cereto-Massague, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallvé, S., & Pujadas, G. (2015). Molecular fingerprint similarity search in virtual screening. *Methods*, *71*, 58–63.

Chibani, S., & Coudert, F.-X. (2020). Machine learning approaches for the prediction of material properties. *APL Materials*, *8*, 080701–080711.

Collin, J. P., Dietrich-Buchecker, C., Gaviña, P., Jimenez-Molero, M. C., & Sauvage, J. P. (2001). Shuttles and muscles: Linear molecular machines based on transition metals. *Accounts of Chemical Research*, *34*(6), 477–487.

Correa-Baena, J.-P., Hippalgaonkar, K., van Duren, J., Jaffer, S., Chandrasekhar, V. R., Stevanovic, V., Wadia, C., Guha, S., & Buonassisi, T. (2018). Accelerating materials development via automation, machine learning, and high-performance computing. *Joule*, *2*(8), 1410–1420.

Coskun, A., Banaszak, M., Astumian, R. D., Stoddart, F. J., & Grzybowski, B. A. (2012). Great expectations: can artificial molecular machines deliver on their promise? *Chemical Society Reviews*, *41*, 19–30.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

Cruz, R., Rojas, G., Quintero, M., & Lopez, N. (1996). Cluster analysis from molecular similarity matrices using a non-linear neural network. *Journal of Mathematical Chemistry*, *20*, 385–394.

Curtarolo, S., Morgan, D., Persson, K., Rodgers, J., & Ceder, G. (2003). Predicting crystal structures with data mining of quantum calculations. *Physical Review Letters*, *91*(13), Article 135503.

Curtarolo, S., Setyawan, W., Hart, G. L., Jahnatek, M., Chepulskii, R. V., Taylor, R. H., Wang, S., Xue, J., Yang, K., Levy, O., Mehl, M. J., Stokes, H. T., Demchenko, D. O., & Morgan, D. (2012). AFLOW: An automatic framework for high-throughput materials discovery. *Computational Materials Science*, *58*, 218–226.

D'Agostino, R. B. (1971). An omnibus test of normality for moderate and large sample size. *Biometrika*, *58*, 341–348.

D'Agostino, R. B., & Pearson, E. S. (1973). Tests for departure from normality. Empirical results for the distribution of b_2 and $\sqrt{b_1}$. *Biometrika*, *60*, 613–622.

David, C. C., & Jacob, D. J. (2014). Principal component analysis: a method for determining the essential dynamics of proteins. In D. Livesay (Ed.), *Protein dynamics. Methods in molecular biology (methods and protocols), vol. 1084*. Totowa, NJ: Humana Press.

De, S., Partok, A. P., Csanyi, G., & Ceriotti, M. (2016). Comparing molecules and solids across structural and alchemical space. *Physical Chemistry Chemical Physics*, *18*(13754).

Dederichs, P. H., & Zeller, R. (1983). Self-consistency iterations in electronic-structure calculations. *Physical Review B*, *10*, 5462–5472.

Dobson, C. M. (2004). Chemical space and biology. *Nature*, *432*, 824–828.

Dudarev, S. L., Botton, G. A., Savrasov, S. Y., Humphreys, C. J., & Sutton, A. P. (1998). Electron-energy-loss spectra and the structural stability of nickel oxide: An LSDA+U study. *Physical Review B*, *57*(1505).

Durant, J. L., Leland, B. A., Henry, D. R., & Nourse, J. G. (2002). Reoptimization of MDL keys for use in drug discovery. *Journal of Chemical Information and Computer Sciences*, *42*, 1273–1280.

Edelsbrunner, H., & Harer, J. (2008). Persistent homology–a survey. In J. E. Goodman, J. Pach, & R. Pollack (Eds.), *Surveys on discrete and computational geometry: Twenty years later. Contemporary Mathematics, vol. 453*. Providence: AMS.

Elton, D. C., Boukouvalas, Z., Fuge, M. D., & Chung, P. W. (2019). Deep learning for molecular design–a review of the state of the art. *Molecular System Design and Engneering*, *4*, 828–849.

Escudero, D., Laurent, A. D., & Jacquemin, D. (2017). Time-dependent density functional theory: A tool to explore excited states. In J. Leszczynski (Ed.), *Handbook of Computational Chemistry, vol. 2*. Heidelberg, Germany: Springer.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from Imbalanced Data Sets*. Cham: Springer.

Feynman, R. (1960a). There's plenty of room at the bottom. *Engineering and Science February*, *23*, 22–36.

Feynman, R. (1960b). The wonders that await a micro-microscope. *COMSAT Technical Review*, *43*, 45–47.

Frisch, M. J., Trucks, G. W., Schlegel, H. B., Scuseria, G. E., Robb, M. A., Cheeseman, J. R., Scalmani, G., Barone, V., Petersson, G. A., Nakatsuji, H., Li, X., Caricato, M., Marenich, A. V., Bloino, J., Janesko, B. G., Gomperts, R., Mennucci, B., Hratchian, H. P., Ortiz, J. V. .... (2016). *Gaussian 16 revision C.01*. Wallingford CT: Gaussian Inc.

Giannozzi, P., Andreussi, O., Brumme, T., Bunau, O., Nardelli, M. B., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Cococcioni, M., Colonna, N., Carnimeo, I., Corso, A. D., de Gironcoli, S., Delugas, P., Jr, R. A. D., Ferretti, A., Floris, A., & Fratesi, G. (2017). Advanced capabilities for materials modelling with quantum expresso. *Journal of Physics: Condensed Matter*, *29*, Article 465901.

Golbraikh, A. (2000). Molecular dataset diversity indices and their applications to comparison of chemical databases and QSAR analysis. *Journal of Chemical Information and Computer Sciences*, *40*, 414–425.

Görling, A. (1996). Density-functional theory for excited states. *Physical Review A*, *54*(3912).

Gu, G. H., Noh, J., Kim, I., & Jung, Y. (2019). Machine learning for renewable energy material. *Journal of Materials Chemistry A*, *7*, 17096–17117.

Hamada, I. (2014). Van der waals density functional made accurate. *Physical Review B*, *89*, Article 121103.

Hansen, K., Biegler, F., Ramakrishnan, R., Pronobis, W., von Lilienfeld, O. A., Muller, K.-R., & Tkatchenko, A. (2015). Machine learning predictions of molecular properties: accurate many-body potentials and nonlocality in chemical space. *The Journal of Physical Chemistry Letters*, *6*, 2326–2331.

Hansen, K., Montavon, G., Biegler, F., Fazli, S., Rupp, M., Scheffler, M., von Lilienfeld, O. A., Tkatchenko, A., & Müller, K.-R. (2013). Assessment and validation of machine learning methods for predicting molecular atomization energies. *Journal of Chemical Theory Computation*, *9*, 3404–3419.

Hautier, G., Fischer, C. C., Jain, A., Mueller, T., & Ceder, G. (2010). Finding nature's missing ternary oxide compounds using machine learning and density functional theory. *Chemistry of Materials*, *22*(12), 3762–3767.

Himanen, L., Geurts, A., Foster, A. S., & Rinke, P. (2019). Data-driven materials science: Status, challenges, and perspectives. *Advanced Science*, *6*(21), Article 1900808.

Himanen, L., Jager, M. O. J., Morooka, E. V., Canova, F. F., Ranawat, Y. S., Gao, D. Z., Rinke, P., & Foster, A. S. (2020). DScribe: Library of descriptors for machine learning in materials science. *Computer Physics Communications*, *247*, Article 106949.

Hohenberg, P., & Kohn, W. (1964). Inhomogeneous electron gas. *Physical Review*, *136*, B864–B871.

Hörmann, L., Jeindl, A., Egger, A. T., Scherbela, M., & Hofmann, O. T. (2019). Sample: Surface structure search enabled by coarse graining and statistical learning. *Computer Physics Communications*, *244*, 143–155.

Huo, H., & Rupp, M. (2017). Unified representation for machine learning of molecules and crystals. arXiv:1704.06439.

Jain, A., Castelli, I. E., Hautier, G., Bailey, D. H., & Jacobsen, K. W. (2013a). Performance of genetic algorithms in search for water splitting perovskites. *Journal of Materials Science*, *48*, 6519–6534.

Jain, A., Ong, S. P, Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., & K.A., Persson (2013b). Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, *1*(1), Article 011002.

Kedziera, D., & Kacmarek-Kedziera, A. (2017). Remarks on wave function theory and methods. In J. Leszczynski, A. Kaczmarek-Kedziera, T. Puzyn, M. G. Papadopoulos, H. Reis, & M. K. Shukla (Eds.), *Handbook of Computational Chemistry*. Springer International Publishing.

Khaire, U. M., & Dhanalakshmi, R. (2019). Stability of feature selection algorithm: A review. *Journal of King Saud University- Computer and Information Sciences*, in press.

Klimeš, J., Bowler, D. R., & Michaelides, A. (2009). Chemical accuracy for the van der Waals density functional. *Journal of Physics: Condensed Matter*, *22*, Article 022201.

Klimeš, J., Bowler, D. R., & Michaelides, A. (2011). Van der Waals density functionals applied to solids. *Physical Review B*, *83*, Article 195131.

Ko, T. W., Finkler, J. A., Goedecker, S., & Behler, J. (2021). A fourth-generation high-dimensional neural network protential with accurate electrostatics including non-local charge transfer. *Nature Communications*, *12*(398).

Kohn, W., & Sham, L. J. (1965). Self-consistent equations including exchange and correlation effects. *Physical Review*, *140*(A1133).

Kong, D., & Cui, Y. (2011). Opportunities in chemistry and materials science for topological insulators and their nanostructures. *Nature Chemistry*, *3*(845).

Koumura, N., Zijlstra, R. W., van Delden, R. A., Harada, N., & Feringa, B. L. (1999). Light-driven monodirectional molecular rotor. *Nature*, *401*(6749), 152–155.

Kozinsky, B., & Singh, D. J. (2021). Thermoelectrics by computational design: Progress and opportunities. *Annual Review of Materials Research*, *51*(1), 565–590.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress of Artificial Intelligence*, *5*, 221–232.

Kresse, G., & Furthmüller, J. (1996). Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, *54*(11169).

Kudernac, T., Ruangsupapichat, N., Parschau, M., Maciá, B., Katsonis, N., Harutyunyan, S. R., Ernst, K. H., & Feringa, B. L. (2011). Electrically driven directional motion of a four-wheeled molecule on a metal surface. *Nature*, *479*(7372), 208–211.

Lange, O. F., & Grubmueller, H. (2008). Full correlation analysis of conformational protein dynamics. *Proteins*, *70*(4), 1294–1312.

Legrain, F., Carrete, J., Roekeghem, A. van., Madsen, G. K. H., & Mingo, N. (2018). Materials screening for the discovery of new half-Heuslers: Machine learning versus ab initio methods. *Journal of Physical Chemistry B*, *122*(2), 625–632.

Li, W., Jacobs, R., & Morgan, D. (2018). Predicting the thermodynamic stability of perovskite oxides using machine learning models. *Computational Materials Science*, *150*, 454–463.

Li, J., Zhang, H., & Chen, J. Z. Y. (2019). Structural prediction and inverse design by a strongly correlated neural network. *Physical Review Letters*, *123*, Article 108002.

Liu, S., Borodinov, N., Vlcek, L., Lu, D., Laanait, N., & Vasudevan, R. K. (2020a). Application of variational policy gradient to atomic-scale materials synthesis. https://arxiv.org/abs/2006.15644.

Liu, Y., Niu, C., Wang, Z., Gan, Y., Zhu, Y., Sun, S., & Shen, T. (2020b). Machine learning in materials genome initiative: A review. *Journal of Materials Research and Technology*, *57*, 113–122.

Lookman, T., Balachandran, P. V., Xue, D., & Yuan, R. (2019). Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design. *Npj Computational Materials*, *5*(21), 1–17.

Lorenz, C., & Doltsinis, N. L. (2017). Molecular dynamics simulation: From ab initio to coarse grained. In J. Leszczynski (Ed.), *Handbook of Computational Chemistry, vol. 1*. Heidelberg, Germany: Springer.

Maisuradze, G. G., Liwo, A., & Scheraga, H. A. (2009). Principal component analysis for protein folding dynamics. *Journal of molecular biology*, *385*(1), 312–329.

Mannodi-Kanakkithodi, A., Pilania, G., Huan, T. D., Lookman, T., & Ramprasad, R. (2016). Machine learning strategy for accelerated design of polymer dielectrics. *Scientific Reports*, *6*(20952).

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proc. of the 20th International Conference on Artificial Intelligence and Statistics 2017*, (pp. 1–10).

Mills, K., Ronagh, P., & Tamblyn, I. (2020). Finding the ground state of spin Hamiltonians with reinforcement learning. *Nature Machine Intelligence*, *2*, 509–517.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of machine learning*. Cambridge, MI, USA: MIT Press.

Morgan, D., & Jacobs, R. (2020). Opportunities and challenges for machine learning in materials science. *Annual Review of Materials Research*, *50*(1), 71–103.

Mueller, T., Kusne, A. G., & Ramprasad, R. (2016). Machine learning in materials science: Recent progress and emerging applications. *Reviews in computational chemistry. Vol. 29 of Reviews in Computational Chemistry. Parrill, A.L., Lipkowitz, K. B. (Editors)*. John Wiley & Sons Inc.

Neil, D., Segler, M., Guasch, L., Ahmed, M., Plumbley, D., Sellwood, M., & Brown, N. (2018). Exploring deep recurrent models with reinforcement learning for molecule design. In *Proc. of 6th International Conference on Learning Representations (ICLR 2018)*.

Nilakantan, R., Bauman, N., Dixon, J. S., & Venkataraghavan, R. (1987). Topological torsion: A new molecular descriptor for SAR applications. Comparison with other descriptors. *Journal of Chemical Information and Computer Sciences*, *27*, 82–85.

Packwood, D. M. (2017). Bayesian optimization for materials science. In M. Kotani (Ed.), *Springer briefs in the mathematics of materials, vol. 3*. Tokyo, Japan: Springer.

Packwood, D. M. (2020). Exploring the configuration spaces of surface materials using time-dependent diffraction patterns and unsupervised learning. *Scientific Reports*, *10*(5868).

Packwood, D. M., Han, P., & Hitosugi, T. (2017). Chemical and entropic control on the molecular self-assembly process. *Nature Communications*, *8*(14463).

Packwood, D. M., & Hitosugi, T. (2018). Materials informatics for self-assembly of functionalized organic precursors on metal surfaces. *Nature Communications*, *9*(2469).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O, Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perdew, J. P., Burke, K., & Ernzerhof, M. (1996). Generalized gradient approximation made simple. *Physical Review Letters*, *77*(18), 3865–3868.

Perdew, J. P., & Wang, Y. (1992). Accurate and simple analytic representation of the electron-gas correlation energy. *Physical Review B*, *45*(13244).

Perdew, J. P., & Zunger, A. (1981). Self-interaction correction to density-functional approximations for many-electron systems. *Physical Review B*, *23*(5048).

Pilania, G., Wang, C., Jiang, X., Rajasekaran, S., & Ramprasad, R. (2013). Accelerating materials property predictions using machine learning. *Scientific Reports*, *3*(2810).

Plimpton, S. (1995). Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, *117*, 1–19.

Ponzoni, I., Sebastián-Pérez, V., Requena-Triguero, C., Roca, C., Martínez, M. J., Fiorella Cravero, J. A. P., Arrayás, R. G., Adrio, J., & Capillo, N. E. (2017). Hybridizing feature selection and feature learning approaches in QSAR modeling for drug discovery. *Scientific Reports*, *7*(2403).

Popova, M., Isayev, O., & Tropsha, A. (2018). Deep reinforcement learning for de novo drug design. *Science Advances*, *4*(7), 1–14.

Pribram-Jones, A., Gross, D. A., & Burke, K. (2015). DFT: A theory full of holes? *Annual Review of Physical Chemistry*, *66*, 283–304.

Ramprasad, R., Batra, R., Pilania, G., Mannodi-Kanakkithodi, A., & Kim, C. (2017). Machine learning in materials informatics: recent applications and prospects. *Npj Computational Materials*, *3*(1), 54.

Rodríguez-Pérez, R., Vogt, M., & Bajorath, J. (2017). Support vector machine classification and regression prioritize different structural features for binary compound activity and potency value prediction. *ACS Omega*, *2*, 6371–6379.

Rogers, D., & Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, *50*, 742–754.

Rupp, M., Tkatchenko, A., Müller, K.-R., & von Lilienfeld, O. A. (2012). Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, *108*, Article 058301.

Scheleder, G. R., Acosta, C. M., & Fazzio, A. (2020). Exploring two-dimensional materials thermodynamic stability via machine learning. *ACS Applied Materials Interfaces*, *12*(18), 20149–20157.

Schleder, G. R., Padilha, A. C. M., Acosta, C. M., Costa, M., & Fazzio, A. (2019). From DFT to machine learning: recent approaches to materials science - a review. *Journal of Physics: Materials*, *2(3)*(032001).

Schmidt, J., Marques, M. R. G., Botti, S., & Marques, M. A. L. (2019). Recent advances and applications of machine learning in solid-state materials science. *Npj Computational Materials*, *5*(1), 83.

Schwaller, P., Probst, D., Vaucher, A. C., Nair, V. H., Kreutter, D., Laino, T., & Reymond, J.-L. (2021). Mapping the space of chemical reactions using attention-based neural networks. *Nature Machine Intelligence*, *3*, 144–152.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*, 461–464.

Seko, A., Togo, A., Hayashi, H., Tsuda, K., Chaput, L., & Tanaka, I. (2015). Prediction of low-thermal-conductivity compounds with first-principles anharmonic lattice-dynamics calculations and Bayesian optimization. *Physical Review Letters*, *115*, Article 205901.

Sharma, V., Wang, C., Lorenzini, R., Ma, R., Zhu, Q., Sinkovits, D., Pilania, G., Oganov, A., Kumar, S., Sotzing, G., Boggs, S., & Ramprasad, R. (2014). Rational design of all organic polymer dielectrics. *Nature Communications*, *5*(4845).

Sheller, M. J., Edwards, B., Reina, G. A., Martin, J., Pati, S., Kotrotsou, A., Milchenko, M., Xu, W., Marcus, D., Colen, R. R., & Bakas, S. (2020). Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, *10*, 12598, 1–12.

Shimizu, R., Kobayashi, S., Watanabe, Y., Ando, Y., & Hitosugi, T. (2020). Autonomous materials synthesis by machine learning and robotics. *APL Materials*, *8*, Article 111110.

Stowasser, R., & Hoffmann, R. (1999). What do the Kohn-Sham orbitals and eigenvalues mean? *Journal of the American Chemical Society*, *121*(3414).

Sutton, R. S., & Barto, A. G. (2020). *Reinforcement learning: An introduction* (2nd ed.). Cambridge, MA, USA: The MIT Press.

Tanimoto, T. (1957). Tanimoto similarity coefficient. *Bulletin Del la Société Vaudoisedes Sciences Naturelles*, *37*, 241–272.

Tkatchenko, A., & Scheffler, M. (2009). Accurate molecular van der waals interactions from ground-state electron density and free-atom reference data. *Physical Review Letters*, *102*, Article 073005.

Townsend, J., Micucci, C. P., Hymel, J. H., Maroulas, V., & Vogiatzis, K. D. (2020). Representation of molecular structures with persistent homology for machine learning applications in chemistry. *Nature Communications*, *11*(3230).

Troisi, A., Orlandi, G., & Anthony, J. E. (2005). Electronic interactions and thermal disorder in molecular crystals containing cofacial pentacene units. *Chemistry of Materials*, *17*(5024).

Tsubaki, M., & Mizoguchi, T. (2020). Quantum deep field: Data-driven wave function, electron density generation, and atomization energy prediction and extrapolation with machine learning. *Physical Review Letters*, *125*, Article 206401.

Tsuneda, T. (2014). *Density functional theory in quantum chemistry*. Tokyo, Japan: Springer.

Tversky, A. (1977). Features of similarity. *Psychological Review*, *84*(4), 327–352.

Van Noorden, R., Maher, B., & Nuzzo, R. (2014). The top 100 papers. *Nature*, *514*(7524), 550–553.

Wang, C. C., Pilania, G., Boggs, S. A., Kumar, S., Breneman, C., & Ramprasad, R. (2014). Computational strategies for polymer dielectrics design. *Polymer*, *55*(4), 979–988.

Weininger, D. (1988). Smiles, a chemical language and information system – part 1: Introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, *28*, 31–36.

Weininger, D. (1990). Smiles – part 3: Depict. graphical depiction of chemical structures. *Journal of Chemical Information and Modeling*, *30*(3), 237–243.

Weininger, D., Weininger, A., & Weininger, J. L. (1989). Smiles – part 2: Algorithm for generation of unique smiles notation. *Journal of Chemical Information and Modeling*, *29*(2), 97–101.

Willett, P., Barnard, J. M., & Downs, G. M. (1998). Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, *38*, 983–996.

Wilson, E. O. (1998). *Consilience - the unity of knowledge*. Random House, New York: First Vintage Books Edition.

Wipke, W. T., & Rogers, D. (1984). Artificial intelligence in organic synthesis. SST: starting material selection strategies. An application of superstructure search. *Journal of Chemical Information and Computer Sciences*, *24*, 71–81.

Yang, Y., Ji, C., & Deng, K. (2021). Rapid design of metamaterials via multi-target Bayesian optimization. *The Annals of Applied Statistics*, *76*, 8–796.

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligence System Technology*, *10*(2).

Yao, T. T., Cheng, J. L., Xu, B. R., Zhang, M. Z., Hu, Y. Z., Zhao, J. H., & Dong, X. W. (2015). Support vector machine (SVM) classification model based on rational design of novel tetronic acid derivatives as potential insecticidal and acaricidal agents. *RSC Advances*, *6*(49195).

Yu, M., Yang, S., Wu, C., & Marom, N. (2020). Machine learning the Hubbard U parameter in DFT+U using Bayesian optimization. *Npj Computational Materials*, *6*(180).

Zakutayev, A., Wunder, N., Schwarting, M., Perkins, J. D., White, R., Munch, K., Tumas, W., & Phillips, C. (2018). An open experimental database for exploring inorganic materials. *Scientific Data*, *5*(1), Article 180053.

Zhang, I. Y., & Gruneis, A. (2019). Coupled cluster theory in materials science. *Frontiers in Materials*, *6*(123).

Ziletti, A., Kumar, D., Scheffler, M., & Ghiringhelli, L. M. (2018). Insightful classification of crystal structures using deep learning. *Nature Communications*, *9*(2775).