4

MACHINE LEARNING IN MATERIALS SCIENCE: RECENT PROGRESS AND EMERGING APPLICATIONS

TIM MUELLER[1], AARON GILAD KUSNE[2], AND RAMPI RAMPRASAD[3]

[1] Department of Materials Science and Engineering, Johns Hopkins University, Baltimore, MD, USA
[2] Material Measurement Laboratory, The National Institute of Standards and Technology, Gaithersburg, MD, USA
[3] Department of Materials Science and Engineering, University of Connecticut, Storrs, CT, USA

INTRODUCTION

Data-to-knowledge ideas are beginning to show enormous promise within materials science. Indeed, the concept of rationally designing materials through the effective use of data-driven methods forms the core of the U. S. Materials Genome Initiative. This paradigm for studying the materials property space has the potential to mitigate the cost, risks, and time involved in an Edisonian approach to the lengthy preparation–testing or the computation–experiment cycles that permeate current approaches to identify useful materials. Moreover, data-centric approaches can also yield valuable insights into the fundamental factors underlying materials behavior and can lead to the discovery of Hume-Rothery-like rules.

To significantly accelerate the pace of discovery using such data-driven paradigms, efficient and effective methods to (i) generate, (ii) manage, and (iii) utilize relevant information are necessary. The last of these tasks can be accomplished in a systematic way through an approach known as “machine learning,” a branch of

Reviews in Computational Chemistry, Volume 29, First Edition.
Edited by Abby L. Parrill and Kenny B. Lipkowitz.

artificial intelligence pertaining to the creation of models that can effectively learn from past data and situations. Machine learning schemes have already impacted areas such as cognitive game theory (e.g., computer chess), pattern (e.g., facial or fingerprint) recognition, event forecasting, and bioinformatics. They are beginning to make major inroads within materials science and hold considerable promise for materials research and discovery.[1,2] Some examples of successful applications of machine learning within materials research in the recent past include accelerated and accurate predictions (using past historical data) of phase diagrams,[3] crystal structures,[4,5] and materials properties,[6,7] as additional examples of prediction of materials properties,[8,9] development of interatomic potentials[10–12] as additional examples of development of interatomic potentials[13,14] and energy functionals[15] for increasing the speed and accuracy of materials simulations, on-the-fly data analysis of high-throughput experiments,[16] mapping complex materials behavior to set of process variables,[17] and so on.

Machine learning algorithms can be separated into two broad classes: *supervised* and *unsupervised* learning. In both of these classes, the algorithm has access to a set of observations known as *training data*. However the nature of the training data, and hence what can be accomplished with the data, differs between the two. In supervised learning, the training data consists of a set of *input values* (e.g., the structures of different materials) as well as a corresponding set of *output values* (e.g., materials property values). With these training data, the machine learning algorithm tries to identify a function that can make accurate predictions about the output values that will be associated with new input values. In unsupervised learning, there are no output values in the training data, and the goal is to identify patterns in the input values. A list of different methods and materials-related applications for each of these classes of algorithms is provided in Table 1. There is a third class, semi-supervised learning, in which some, but not all of the input values have corresponding output values. To date, semi-supervised learning algorithms have seen little use in materials science and engineering, and we do not cover them here.

This chapter is written for a materials researcher with an interest in machine learning methods. These methods come in many flavors under many names with a generous amount of jargon (as can be gleaned from Table 1). To effectively use

TABLE 1 Supervised and Unsupervised Learning Examples

	Example Methods	Selected Materials Applications
Supervised learning	Regularized least squares Support vector machines Kernel ridge regression Neural networks Decision trees Genetic programming	Predict processing structure–property relationships; develop model Hamiltonians; predict crystal structures; classify crystal structures; identify descriptors
Unsupervised learning	*k*-Means clustering Mean shift theory Markov random fields Hierarchical cluster analysis Principal component analysis Cross-correlation	Analyze composition spreads from combinatorial experiments; analyze micrographs; identify descriptors; noise reduction in data sets

learning schemes, a familiarity with the underlying mathematical tools and technical jargon is necessary. Thus, the next two sections of this chapter are almost entirely devoted to building this familiarity in one unified treatment (although relevant materials science illustrations are provided throughout those sections). Subsequent sections provide a rich assortment of pedagogical examples of successful applications of machine learning methods within materials science, and recommendations for useful machine learning-related resources.

SUPERVISED LEARNING

One of the fundamental goals of science is the development of theories that can be used to make accurate predictions. Predictive theories are generated through the scientific method. Here, existing *knowledge* is used to formulate *hypotheses*, which are then used to make *predictions* that can be empirically tested, with the goal of identifying the hypotheses that make the most accurate predictions. The scientific method can be expressed mathematically by considering predictive theory as a function f that maps a set of input data $\mathbf{x}$ to a predicted outcome y. The function may be relatively simple, as in Newton's laws of motion, or it may be complex, as in models that predict the weather based on meteorological observations. The collection of known input ($\mathbf{x}$) and output (y) values, called *training data*, may be generated through observations or controlled experiments. The goal of the scientist is to use such training data, as well as any other prior knowledge, to identify a function that is able to predict the output value for a new set of input data accurately. The process of identifying such a function from a set of known $\mathbf{x}$ and y values is called *supervised learning*. If the allowed output values y form a continuous range (e.g., melting points), then the process of searching for a function is known as *regression*. If the allowed output values form a discrete set (e.g., space groups), the process is then known as *classification*.

The *hypothesis space* contains all hypotheses (i.e., functions) that could be returned by the learning algorithm. An appealing choice for a hypothesis space might be the space in which every possible function is considered to be equally viable. However there will be many functions in this space that exactly reproduce the training data (Figure 1), and there will be no way to determine which functions would make the most accurate predictions. Thus, to identify a predictive function, it is necessary to use a hypothesis space that is constrained in a way that excludes some functions from consideration and/or is weighted to favor some functions more than others. For example, it is common to constrain the hypothesis space so that only functions expressing a linear relationship between the input and output values are considered (Figure 1d).

There may be no function in the hypothesis space that produces the observed output values for all possible sets of input values. This could happen for some combination of several reasons:

- The hypothesis space has been constrained in a way that excludes the function that perfectly maps the input values to the output values. For example, the hypothesis space might be constrained to include only linear functions, whereas no

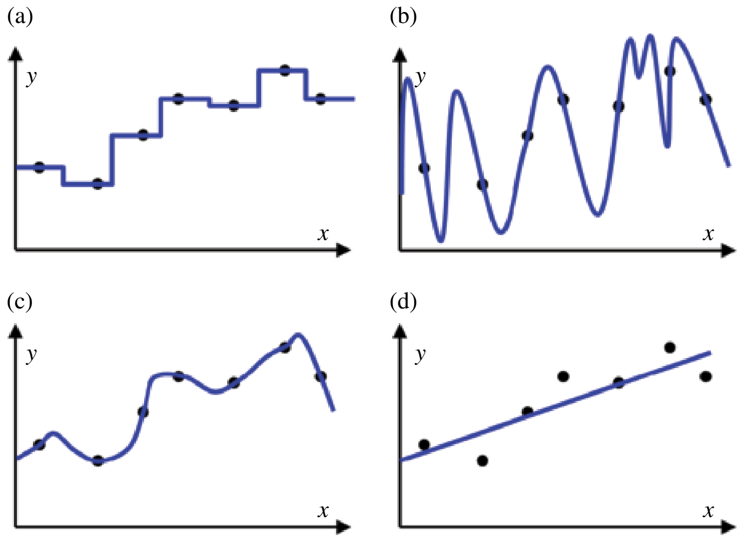


FIGURE 1 Four different functions (blue lines) fit to the same set of training data (black dots). (a), (b), and (c) reproduce the data, and (d) does not (color available in e-book version).

linear function of the input variables can reproduce the observed output values (Figure 1d).

- The observed output values are the result of a process that is inherently nondeterministic.
- Some input data that are relevant to calculating the correct output values are missing and/or unknown.

In these situations, any function in the hypothesis space will result in some error in the predicted values. To account for this error, the output values can be expressed as

$$y = f\left(\mathbf{x}\right) + E \qquad [1]$$

where f is a function contained in the hypothesis space and E is a random error. We will let g represent the probability distribution from which E is drawn. In other words, $g(a)$ is the probability density that $y - f(\mathbf{x}) = a$. The distribution g may depend on the input data, but for simplicity we will generally assume that it does not. In general, both the function f and the probability distribution g are unknown.

A Formal Probabilistic Basis for Supervised Learning

For a given probability distribution g, we can estimate the probability density that a function f satisfies Eq. [1]. This probability density is expressed as $P(f \mid \mathbf{D}, g)$, where $\mathbf{D}$ is the observed training data. This probability density can be evaluated using Bayes' rule.[18]

Bayes' rule is a fundamental statistical theorem that can be derived from the fact that the probability of two events, A and B, occurring is given by the probability of B occurring times the conditional probability that A occurs given that B has occurred. Mathematically, this is written as

$$P(A, B) = P(B)P(A \mid B) \tag{2}$$

Similarly,

$$P(A, B) = P(A)P(B \mid A) \tag{3}$$

Combining Eqs. [2] and [3] yields Bayes' rule

$$P(A \mid B) = \frac{P(B \mid A)}{P(B)} P(A) \tag{4}$$

In the context of supervised learning, Bayes' rule yields

$$P(f \mid \mathbf{D}, g) = \frac{P(\mathbf{D} \mid f, g)}{P(\mathbf{D} \mid g)} P(f \mid g) \tag{5}$$

The probability distribution $P(f \mid g)$ gives the probability density that a function f satisfies Eq. [1] given g prior to the observation of any training data. For this reason, it is known as the *prior probability distribution*, which is sometimes simply referred to as the *prior*. The probability distribution $P(f \mid \mathbf{D}, g)$ represents the probability of the same event after accounting for the training data. It is known as the *posterior probability distribution*. The distribution $P(\mathbf{D} \mid f, g)$, commonly known as the *likelihood function*, is the probability of observing the training data $\mathbf{D}$ given f and g. The remaining term in Eq. [5], $P(\mathbf{D} \mid g)$, does not depend on f and can effectively be treated as a normalization constant.

Bayes' rule provides a natural and intuitive framework (or basis) for understanding learning. Initially, the hypothesis space is constrained and/or weighted through the prior probability distribution. All functions that are excluded from the hypothesis space are assigned a probability of zero, and functions that are not excluded are assigned nonzero prior probabilities. These probabilities represent the prior belief that any particular function satisfies Eq. [1]. As training data are observed, these probabilities are updated to account for the new knowledge, resulting in the posterior probability distribution—this is the learning step. If additional training data ($\mathbf{D}_2$) were to be observed, a reapplication of Bayes' rule could be used to update the probability distributions further:

$$P(f \mid \mathbf{D}_2, \mathbf{D}, g) = \frac{P(\mathbf{D}_2 \mid f, \mathbf{D}, g)}{P(\mathbf{D}_2 \mid \mathbf{D}, g)} P(f \mid \mathbf{D}, g) \tag{6}$$

where the posterior distribution in Eq. [5] has become the prior distribution in Eq. [6]. Thus repeated application of Bayes' rule can be used to update the likelihood that a

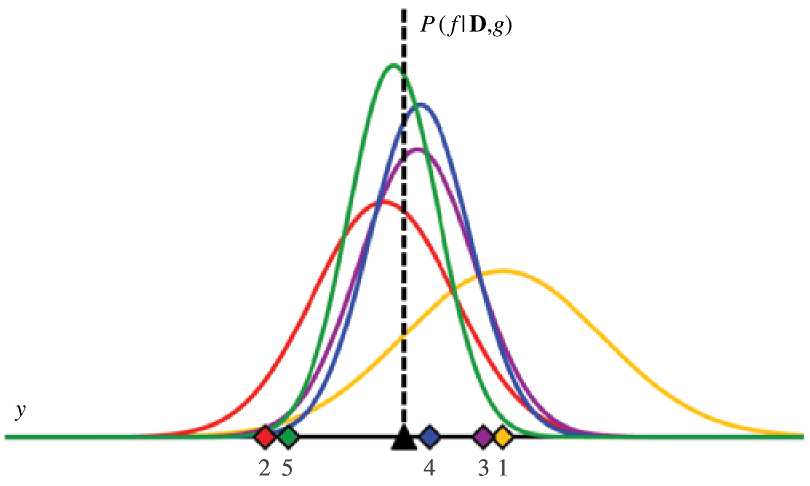


FIGURE 2 An example of how the posterior distribution changes with the addition of new training data points. In this example, f is a constant value indicated by the black triangle and g is a normal distribution with zero mean and known variance. The diamonds represent training data. Assuming a uniform prior in which all values for f are considered equally likely, the posterior distributions for f after the addition of the first (orange), second (red), third (purple), fourth (blue), and fifth (green) training data points are shown. (*See insert for color representation of the figure.*)

particular hypothesis is best as new data come in. An example of how the posterior distributions change with new data points is shown in Figure 2.

Because it can be awkward to deal with a probability distribution defined over many functions, learning algorithms will commonly return the single function $\hat{f}$ that maximizes the posterior probability density:

$$\hat{f} = \arg\max_{f} P(f \mid \mathbf{D}, g) \tag{7}$$

Combining Eq. [7] with Eq. [5] yields

$$\hat{f} = \arg\max_{f} \left[P(\mathbf{D} \mid f, g) P(f \mid g) \right] \tag{8}$$

Because the natural log function is monotonically increasing, Eq. [8] can be equivalently written

$$\hat{f} = \arg\min_{f} \left[-\ln\left(P(\mathbf{D} \mid f, g)\right) - \ln\left(P(f \mid g)\right) \right] \tag{9}$$

The term in square brackets on the right side of Eq. [9] is the *objective function* to be minimized. The function with the maximum posterior probability of satisfying Eq. [1] is the function that minimizes the objective function. The idea of finding a function that minimizes an objective function is common to most machine learning algorithms, and the Bayesian analysis enables interpretation of the components of the objective function. The first term, $-\ln(P(\mathbf{D} | f, g))$, represents the *empirical risk*, a measure of how well a given function reproduces the training data. The second term, $-\ln(P(f | g))$, is the *regularization term*, representing the constraints and weights that are put on the hypothesis space before the training data are known.

Equation [9] shows a commonly used framework for supervised learning algorithms. To determine the function $\hat{f}$ that is returned by the algorithm, there are five choices that typically need to be made. It is necessary to determine both what the **hypothesis space** should be and how the **empirical risk** will be calculated. There must also be an **optimization algorithm** that is capable of selecting a function from the hypothesis space (e.g., by minimizing the objective function). In addition, it is sometimes necessary to select the input values to be included in the **training data** and to **estimate the prediction error** for the selected function. In the following sections, each of these five choices is discussed in more detail.

The Hypothesis Space It is through the prior probability distribution $P(f | g)$ that the hypothesis space is constrained and/or weighted. Thus selecting the prior probability distribution is equivalent to deciding which hypotheses will be considered and to what extent some hypotheses should be preferred over others. Hypotheses often take the form of *models*, or parameterized functions, for which the parameters are unknown. Often the prior distribution is implicitly specified simply by choice of the model to be used; for example, the assumption that a function is linear (as in Figure 1d) implicitly assigns zero prior probability to all nonlinear functions.

The choice of the prior probability distribution can impact the effectiveness of the learning process significantly. Functions that are assigned a prior probability of zero are excluded from the hypothesis space and will not be considered no matter how strongly they are supported by the training data. On the other hand, a function that is assigned a small, but nonzero, prior probability could have a large posterior probability, provided it predicts the training data with relatively high accuracy. It is advantageous to choose prior distributions that assign the highest probability to the most accurate functions, as these distributions will generally result in an accurate choice of $\hat{f}$ with little (or no) additional training data.

The prior probability distribution is incorporated into the objective function through the regularization term, $-\ln(P(f | g))$. This term is often described as a way to penalize functions that are unlikely to make good predictions, and the benefit of a regularization term can be derived independently from Bayesian analysis.[19] The use of a regularization term can make an *ill-posed* problem, in which there is no unique solution and/or the solution does not change continuously with the training data, into a problem that is *well posed.*

The act of choosing a prior probability distribution can be controversial because it can introduce a subjective element into the learning process. However the choice of a prior must be made, either implicitly (e.g., by constraining the hypothesis space to only include certain functions) or explicitly. Here we describe five common strategies for selecting the prior probability distribution/regularization term.

Uninformative Priors If no information is known about the process being modeled, a natural choice for the prior distribution would be one that makes the fewest assumptions about the relative merits of different candidate functions. Such prior probability distributions are known as *uninformative priors*. An uninformative prior distribution that assigns equal probabilities to *all* possible functions would make it impossible to differentiate between functions that perfectly reproduce the training data. Thus if an uninformative prior distribution is to be used among functions in the hypothesis space, it is necessary to constrain the hypothesis space to only include certain functions.

The appropriate choice of an uninformative prior may depend on the problem being modeled and the way in which the functions in the hypothesis space are parameterized. A number of different strategies have been proposed, and a thorough review of these can be found in Ref. 20. One strategy for choosing uninformative priors is the principle of maximum entropy, championed by E. T. Jaynes.[21] This principle states that the prior probability distribution should be chosen in a way that maximizes the *information entropy* within the constraints of existing knowledge, where the information entropy of a probability density p is a measure of uncertainty defined as

$$S(p) = -\int_{-\infty}^{\infty} p(x)\ln\big(p(x)\big)dx \tag{10}$$

Jaynes has made the case that the distribution that maximizes the information entropy is the "maximally noncommittal" probability distribution.[21] For example, if a single scalar parameter is to be determined and nothing is known about it prior to the observation of the training data, then the principle of maximum entropy would state that the appropriate prior distribution is the *uniform prior* in which all possible values are considered equally likely.

A uniform prior that assigns equal prior probabilities to an infinite number of functions is an *improper prior*, in that it cannot be normalized to 1. In practice, such improper priors are widely used, as they may result in a proper (i.e., normalizable) posterior distribution.

Model Selection One strategy for selecting a prior distribution is to evaluate multiple possible priors and choose the one giving the posterior distribution with the lowest expected prediction error. To this end, the methods described in the section "Estimating the Prediction Error" may be employed. A common variation of this approach is *model selection*, in which the learning process is broken into two steps. In the first step, a parameterized function (a.k.a. *model*) is selected, effectively assigning zero prior probability to all other functions. (Different types of models are discussed in the section on "Supervised Learning Algorithms"; common examples include linear models, neural networks, etc.) In the second step the parameter values that minimize the objective function for the selected model are determined.

A general approach to model selection can be derived through a Bayesian analysis. For a given model and a given set of training data, different values of the model parameters will result in different values for the empirical risk. Let r represent the minimum value of the empirical risk achievable within a given model for a given set of training data. Schwartz demonstrated that under a wide range of conditions, the model with the greatest expected posterior probability is the one for which the following value is smallest:

$$2r + k\ln(n) \qquad [11]$$

where k is the number of parameters in the model and n is the number of elements ($\mathbf{x}, y$ pairs) in the training data.[22] The expression in Eq. [11] is commonly known as the *Bayesian information criterion* (BIC). Separately, Akaike derived a similar term to be minimized for model selection, commonly known as the *Akaike information criterion* (AIC):[23]

$$2r + 2k \qquad [12]$$

Both the Bayesian information criterion and Akaike information criterion are valuable for revealing a general rule for model selection: among models that reproduce the training data equally well, the one with the fewest parameters can be expected to have the lowest prediction error. This result is similar to Occam's razor, a commonly used heuristic that all else being equal, simple hypotheses should be favored over more complex ones. Intuitively, this insight can be understood by considering that if there are more parameters in a model, there is greater risk of selecting a function that happens to reproduce the training data well but has little predictive ability. This is known as *overfitting* the training data.

Prior Knowledge Determining the best way to constrain and/or weight the hypothesis space can be accomplished by incorporating *prior knowledge* about the process being modeled into the prior distribution. For example, physical arguments might suggest that the output value should be a linear function of a particular input value, or the expected magnitudes of some of the parameters that define the function might be known. The prior probability distribution can be constructed in a way that accounts for this knowledge, directing the learning process toward functions that are expected to be reasonable even before the observation of the training data. The use of a prior probability distribution that effectively takes into account external knowledge can significantly accelerate the learning process by making use of all available knowledge.

The principle of maximum entropy can be combined with prior knowledge to create a prior distribution that incorporates existing knowledge in a "maximally noncommittal" way. For example, if estimates for the mean and the variance of a parameter's value are available, then a Gaussian distribution over possible parameter values will maximize the information entropy and would therefore be the appropriate choice of a prior distribution for the parameter value under the principle of maximum entropy.

Prior knowledge can be especially useful for setting the mean of the prior probability distribution. If the prior probability distribution has a nonzero mean $\bar{f}$, then the learning process can be recast in terms of a prior probability distribution with zero mean by replacing the function f with

$$\Delta f = f - \bar{f} \qquad [13]$$

Accordingly, each value y_i in the training data is replaced with

$$\Delta y_i = y_i - \bar{f}(\mathbf{x}_i) \qquad [14]$$

For example, if f calculates the energy of a compound, then $\bar{f}$ might be the composition-weighted average of the energies of the constitutive elements and $\boldsymbol{\Delta} f$ would be the formation energy.

The function $\boldsymbol{\Delta} f$ represents the difference between the actual function f and the expected function $\bar{f}$. Because f is expected to resemble $\bar{f}$, it can be expected that the norm of $\boldsymbol{\Delta} f$, represented by $\|\boldsymbol{\Delta} f\|$, is more likely to be small than it is to be large. For this reason, it is common to use a term that monotonically increases with $\|\boldsymbol{\Delta} f\|$ as the regularization term.

A variety of different norms can be used, and some of the most popular take the form of the L^p norm, defined by

$$\|f\| = \left(\int |f(\mathbf{x})|^p \, d\mathbf{x} \right)^{1/p} \qquad [15]$$

where $p \geq 1$ and the integral is over all values of $\mathbf{x}$. The L^1 and L^2 norms are commonly used, as are *smoothing norms* that favor functions that vary smoothly with the input values (i.e., their higher-order derivatives are small). An example of a widely used smoothing norm is the L^2 norm of the second derivative of $\boldsymbol{\Delta} f$.

It is generally a good idea to transform f to $\boldsymbol{\Delta} f$ if prior knowledge can be used to make a reasonable estimate for $\bar{f}$; otherwise $\bar{f} = 0$ is implicitly used. The approaches to supervised learning discussed in this chapter are equally applicable to f and $\boldsymbol{\Delta} f$.

Hyperpriors An alternative approach to selecting a particular prior distribution is to assign a probability distribution over the space of possible prior distributions. Such a probability distribution is known as a *hyperprior*. For example, if a Gaussian distribution with zero mean is used as the prior, a hyperprior could be constructed as a probability distribution over possible values of the variance of the Gaussian. The posterior distribution can then be calculated as a weighted average over prior distributions:

$$P(f \mid \mathbf{D}, g) = \int \frac{P(\mathbf{D} \mid f, g)}{P(\mathbf{D} \mid g)} P(f \mid g) P(P(f \mid g)) \qquad [16]$$

where $P(f|g)$ is the prior, $P(P(f|g))$ is the hyperprior, and the integral is over all possible prior distributions.

Many of the same challenges for determining a prior exist for determining a hyperprior, and there is an extra integration step that needs to be performed to arrive at the posterior. However the hyperprior allows for an extra layer of abstraction in situations in which the posterior may be particularly sensitive to the choice of a prior. It is possible to similarly define hyperhyperpriors, etc., but in practice this is rarely done.

The Empirical Risk The empirical risk represents the negative log probability of observing the training data for a given *f* and *g*. Assuming that all of the observations in the training data are independent, from Eq. [1] and the definition of *g*, we can write

$$P(\mathbf{D} \mid f, g) = \prod_i g\left(y_i - f(\mathbf{x}_i)\right) \tag{17}$$

where $\mathbf{x}_i$ is the *i*th set of input values in the training set, y_i is the *i*th output value, and the product is over all elements in the training set. Thus, under the assumption that the observations in the training set are independent of each other, the empirical risk can be written as

$$-\ln\left(P(\mathbf{D} \mid f, g)\right) = \sum_i -\ln\left(g\left(y_i - f(\mathbf{x}_i)\right)\right) \tag{18}$$

where the sum is over all elements in the training set. For example, if *g* is assumed to be Gaussian, then the empirical risk would depend on the sum of the squared differences between y_i and $f(\mathbf{x}_i)$. This approach leads to least-squares fitting, discussed in more detail in the section on "Regularized Least Squares."

The empirical risk is sometimes written more generally as

$$\sum_i L\left(y_i, f(\mathbf{x}_i)\right) \tag{19}$$

where *L* is a *loss function* that calculates the penalty (a.k.a. *loss*) for large differences between y_i and $f(\mathbf{x}_i)$. In practice many commonly used loss functions can be written in the form of Eq. [18], as a function of $y_i - f(\mathbf{x}_i)$.

Unlike the prior probability distribution, the empirical risk depends on the function *g*, which is in general unknown. As both *f* and *g* are unknown, it might make sense to treat the two similarly and to search for the pair of functions that together are most likely to satisfy Eq. [1]. The posterior and prior distributions would then be defined for the pair, (f, g), and application of Bayes' rule would yield

$$P(f, g \mid \mathbf{D}) = \frac{P(\mathbf{D} \mid f, g)}{P(\mathbf{D})} P(f, g) \tag{20}$$

as an alternative to Eq. [5]. However such an approach is not commonly used. Instead, it is more common to make the prior assumption that *g*, and hence the loss function, is known.

If a uniform prior probability distribution is used for all functions in the hypothesis spaces, then the optimization of the objective function (Eq. [9]) can be accomplished by minimizing the empirical risk. This approach is known as *empirical risk minimization*. Empirical risk minimization is equivalent to selecting the function that maximizes the likelihood function $P(\mathbf{D} \mid f, g)$ (i.e., the function that best reproduces the training data). Although empirical risk minimization is a widely used method, there is a risk of overfitting training data. This risk can often be mitigated by replacing the uniform prior used in empirical risk minimization with a prior distribution that favors simple functions or takes into account prior knowledge.

Optimization Algorithms Many machine learning algorithms involve the optimization of an objective function, as in Eq. [9]. For practical purposes, the functions in the hypothesis space are typically characterized by a set of unknown parameters; for example, the functions may be expressed as a linear combination of basis functions, in which the linear expansion coefficients are unknown parameters. Thus the problem of searching for the optimal function becomes a problem of finding the set of parameters that minimize the objective function. Many algorithms have been developed to address such optimization problems (e.g., gradient descent approaches, simulated annealing, etc.), and the field of general optimization algorithms is too large to discuss here. Instead we refer the reader to some of the many comprehensive books on the subject (e.g., Refs. 24–26).

For some objective functions, there may be no known algorithm that is able to find the globally optimal function and/or verify whether a particular function is globally optimal with a reasonable computational cost. However many machine learning algorithms use an objective function and hypothesis space that have been designed to facilitate the rapid identification of the globally optimal function. The ways in which this is done are described in the context of individual machine learning algorithms in the section on “Supervised Learning Algorithms.”

The Training Data The training data may be generated by observations of external events that cannot easily be controlled, such as climatological data used in weather forecasting. However in many cases it is possible to generate training data through controlled experiments. In each experiment, a set of input values are evaluated, and the corresponding output value becomes known once the experiment is complete. Because the generation of training data can be an expensive and/or time-consuming step in the learning process, it is desirable to minimize the total number of experiments that must be performed to achieve an acceptable level of prediction error. The field of *active learning*, also known as *design of experiments*, deals with determining the best set of experiments to perform (i.e., determining the best elements to include in training data) to minimize the total cost of generating the training data.

There are many different approaches to active learning, and we will not review them all here. Good overviews can be found in Refs. 27–29. A common approach to active learning is *uncertainty sampling*,[30] in which the next experiment is performed on input values for which there is a large amount of uncertainty in the predicted output value. A related approach is *query by committee*,[31] in which several different models are trained on the same data, and the next experiment is performed on a data point about which there is the least agreement among the models. However this approach can result in the sampling of outlier data points that are not representative of the space of possible input values. Alternatively, if the distribution of all possible input values is known, the input values can be selected in a way that takes this distribution into account. Such *density-weighted* methods can result in significant performance improvements over methods that do not account for the distribution of possible input values.[28]

If the hypothesis space consists of parameterized functions, the training data may be chosen in a way that minimizes some measure of the variance of the estimated parameter values. This is often accomplished by optimizing the *observed information matrix*, sometimes called simply the *information matrix*.[32] The observed information matrix is the Hessian of the empirical risk with respect to the function parameters, and it is often evaluated at the parameter values that minimize the empirical risk. It is an indicator of how informative the current training data are about the parameter values.

A number of different criteria have been developed to optimize the observed information matrix. Among the most common are *A-optimality*, in which the trace of the inverse information matrix is minimized, and *D-optimality*, in which the determinant of the information matrix is maximized.[33–35] An overview of these and many other optimality criteria can be found in Ref. 36. As the information matrix does not take into account the prior probability distribution over possible parameter values, an alternative approach is to use the Hessian of the objective function in place of the information matrix. In this approach, sometimes referred to as *Bayesian experimental design*,[37,38] the same matrix optimality criteria (A-optimality, D-optimality, etc.) may be used. In materials science, optimization of the information matrix has been used to select training data for cluster expansion models, as described in the section on "Lattice Models."

Estimating the Prediction Error The goal of a machine learning algorithm is to identify a function that makes accurate predictions for input values that are not included in the training data. Thus to evaluate the results of a learning algorithm, it is not sufficient to evaluate how well the function reproduces the training data (i.e., the empirical risk). Rather it is best to use a method that is capable of estimating the *prediction error* of a function over the distribution of all possible input values.

The estimation of the prediction error can be accomplished using *resampling* methods, in which functions are trained on one or more subsamples of the training data using the same learning algorithm that is used for the entire set of training data. These functions are then evaluated using subsamples of the training data on which they were not trained, providing an estimate of the predictive ability of the functions identified by the learning algorithm.

A common resampling technique is *cross-validation*,[39–42] in which the set of known observations are partitioned into two subsets. The first subset, the *training set*, is used to identify a likely function. The predictive power of this function is evaluated by calculating its prediction error when applied to the second subset, known as the *test set*. Averaging the cross-validation prediction error over multiple different partitions provides a measure of the estimated prediction error for a function trained on the entire known set of observations. A common variation of cross-validation approach is *k-fold cross-validation*, in which the set of observations are partitioned into k different subsets, and the prediction error on each subset is evaluated for a function trained on the $k-1$ remaining subsets. When k equals the number of samples in the training set, this approach is known as *leave-one-out cross-validation*.

Another popular resampling technique is *bootstrapping*,[43–45] in which the subsamples are drawn from the set of training data with replacement, meaning the same element can appear in the subsample more than once. Functions trained on these subsamples are then compared to a function trained on the entire set of training data. Bootstrapping is commonly used to estimate statistics such as the bias and variance in the output of the learning algorithm. Additional details about cross-validation, bootstrapping, and other resampling methods can be found in Refs. 46–50.

Resampling methods provide the best estimates of prediction error when the distribution of input values in the set of known observations is representative of the distribution of all possible input values. Training data that consist of uncontrollable empirical observations generally fit this description, provided the observations are effectively randomly drawn from the distribution of all possible input values. Alternatively, density-weighted active learning methods can generate a set of observations that are representative of the distribution of possible input values and are well suited for use in cross-validation algorithms.

Supervised Learning Algorithms

Many supervised learning algorithms have been developed, and it is not feasible to describe them all here. Instead, a brief overview of some of the most common approaches is provided. These approaches are all described in the context of the framework presented in the section entitled "A Formal Probabilistic Basis for Supervised Learning." In each of these approaches, there is a **hypothesis space**, **objective function**, and **optimization algorithm**. There are also sometimes algorithm-specific active learning methods used to generate an efficient set of **training data**. Approaches for **estimating prediction error** are fairly universal, so they will not be described in detail here.

Regularized Least Squares One of the most widely used methods for fitting a function to data is least-squares regression. The least-squares approach is characterized by the assumption that g is Gaussian:

$$g\left(y_i - f\left(\mathbf{x}_i\right)\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{\left(y_i - f\left(\mathbf{x}_i\right)\right)^2}{2\sigma^2}} \qquad [21]$$

The empirical risk is therefore

$$-\ln\left(P\left(\mathbf{D} \mid f, g\right)\right) = \frac{1}{2\sigma^2}\sum_i \left[\left(y_i - f\left(\mathbf{x}_i\right)\right)^2 - \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right)\right] \qquad [22]$$

where the sum is over all elements in the training set. The loss function depends on the squared difference between the observed and predicted output values, and empirical risk minimization yields

$$\hat{f} = \underset{f}{\arg\min} \sum_i \left(y_i - f\left(\mathbf{x}_i\right)\right)^2 \tag{23}$$

Equation [23] describes a *least-squares fit*, in which the selected function minimizes the sum of the squared errors over all of the elements in the training set. The loss function in Eq. [23], known as the *squared error loss*, is commonly used in machine learning algorithms. Under the assumptions that the errors are normally distributed with zero mean and constant variance (Eq. [21]), the least-squares fit returns the function that maximizes the probability of observing the training data. Similarly, if a uniform prior is assumed for the functions in the hypothesis space, the least-squares fit returns the function that maximizes the posterior probability distribution.

The least-squares fit is often a justifiable choice for function fitting. The assumption that g is Gaussian can be justified using the principle of maximum entropy, and the use of a uniform prior can be justified on the grounds that it is uninformative and hence will not bias the results. Perhaps most importantly from a practical perspective, the least-squares fit is conceptually simple and easy to implement. In a common implementation, known as *linear least squares* or *ordinary least squares*, the hypothesis space is restricted to include only linear functions of the input values. The optimal set of coefficients, $\hat{\boldsymbol{\beta}}$, is given by

$$\hat{\boldsymbol{\beta}} = \underset{\beta}{\arg\min} \sum_i \left(y_i - \mathbf{x}_i\boldsymbol{\beta}\right)^2 \tag{24}$$

where $\mathbf{x}_i$ is a row vector containing the input values and $\boldsymbol{\beta}$ is a column vector containing the unknown input values. From Eq. [24], the exact optimal solution can be calculated:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\mathbf{T}\mathbf{X}\right)^{-1}\mathbf{X}^\mathbf{T}\mathbf{y} \tag{25}$$

where $\mathbf{y}$ is a column vector in which the ith element is y_i and $\mathbf{X}$ is a matrix in which the ith row is $\mathbf{x}_i$. A unique solution to Eq. [25] only exists if the matrix $\mathbf{X}^\mathbf{T}\mathbf{X}$ is nonsingular.

It is often possible to improve upon least-squares fits by using a nonuniform prior distribution. For example, consider the situation in which a multivariate normal distribution is used as the prior:

$$P\left(\boldsymbol{\beta} \mid g\right) \propto e^{\frac{-\boldsymbol{\beta}^\mathrm{T}\Lambda\boldsymbol{\beta}}{2\sigma^2}} \tag{26}$$

where $\boldsymbol{\Lambda}$ is a positive definite matrix. The set of coefficients that minimize the objective function is given by

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\mathbf{T}\mathbf{X} + \Lambda\right)^{-1}\mathbf{X}^\mathbf{T}\mathbf{y} \tag{27}$$

Equation [27] represents a type of *regularized least-squares* fit known as *Tikhonov regularization*.[19] When a normal least-squares fit is ill posed (e.g., when $\mathbf{X}^\mathbf{T}\mathbf{X}$ is singular), Tikhonov regularization can make the problem well posed, such that a unique solution is guaranteed. Variants of Tikhonov regularization are popular in part because they are often easy to justify and robust and the optimal solution can be found with only slightly more computational cost than a least-squares fit.

When $\Lambda = \lambda\mathbf{I}$ in Eq. [27], where **I** is the identity matrix, it is known as *ridge regression*.[51] Ridge regression is equivalent to using the squared ℓ^2 norm of the coefficients as the regularization term, where the ℓ^p norm for a vector is defined as

$$\|\beta\| = \left(\sum_i \beta_i{}^p\right)^{1/p} \qquad [28]$$

for $p \geq 1$. (It is the discrete version of the L^p norm described by Eq. [15].) Another popular form of regularized least squares includes the use of an ℓ^1 norm instead of an ℓ^2 norm. The use of the ℓ^1 norm often results in a minimizing vector of coefficients, $\hat{\beta}$, in which many elements are identically 0. This approach is sometimes known as the *lasso estimator* or *compressive sensing*,[52,53] and it is useful when the solution is sparse. It is in general not as straightforward as using the ℓ^2 norm, but efficient algorithms for finding the optimal set of coefficients exist.[54] The ℓ^1 and ℓ^2 norms are revisited in the sections on "Similarity and Dissimilarity Measures" and "Lattice Models."

Linear least squares is a particularly convenient method for active learning approaches (described in the section on "The Training Data"), as the information matrix is independent of the values of the coefficients and proportional to $\mathbf{X^TX}$. For Bayesian experimental design, the matrix $\mathbf{X^TX} + \Lambda$ may be used in place of $\mathbf{X^TX}$.

Support Vector Machines Classification problems differ from regression problems in that the set of allowed output values is discrete, with each allowed output value corresponding to a different class. The same general learning framework could be used for classification problems as for regression problems, but in practice it can be difficult to work with discontinuous output values. Many classification problems can be simplified by recognizing that different classes often correspond to different regions in input value space (Figure 3). Thus instead of searching for a discontinuous function that predicts the output values directly, it is possible to search instead for a continuous dividing surface between the regions corresponding to different classes.

For example, consider a situation in which there are two classes, corresponding to $y = 1$ and $y = -1$. If the prior assumption is made that there is a linear hyperplane in the space of input values that divides the two different classes, then the classification function takes the form

$$f(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{w}\cdot\mathbf{x} - b > 0 \\ -1, & \text{if } \mathbf{w}\cdot\mathbf{x} - b < 0 \end{cases} \qquad [29]$$

where the vector of coefficients **w** (commonly known as *weights*) and the offset b define the hyperplane that divides the two classes (Figure 3). Thus the discrete classification function, $f(\mathbf{x})$, can be determined by learning the continuous function $\mathbf{w}\cdot\mathbf{x} - b$. We will call this continuous function $h(\mathbf{x})$.

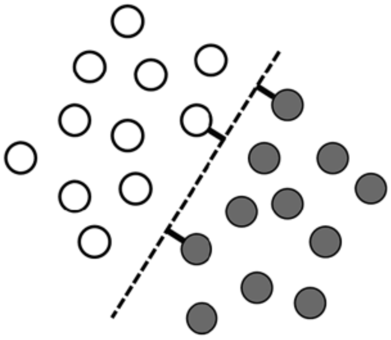


FIGURE 3 A linearly separable data set containing two classes (gray and white). Each point corresponds to coordinates (x_1, x_2) given by the input values, and the colors are determined by the corresponding y values. The dashed line shows the dividing plane, and the short thick lines are the support vectors.

The training data are *linearly separable* if there is at least one hyperplane that perfectly separates the two classes. There are usually many competing hyperplanes that can separate linearly separable data, and the goal of the learning algorithm is to find the one that is most likely to correctly separate data points that are not in the training set. One approach is to find the hyperplane that is farthest from the input values in the training data. In other words, the best hyperplane is the one with the longest *support vectors*, which are defined as the shortest vectors between the hyperplane and the nearest training data point of each class (Figure 3). *Support vector machines* are widely used supervised learning algorithms that identify such a hyperplane.[55] If there are more than two different classes present, support vector machines can be used to find the hyperplanes separating all pairs of classes. A brief introduction to support vector machines is presented in the following text, and a more comprehensive review of different support vector machine approaches can be found in Refs. 56 and 57.

Hyperplanes are defined by the unknown parameters **w** and b, and the optimal values for these parameters can be found by minimizing an objective function similar to the one in Eq. [9]. The loss function (Eq. [19]) is given by

$$L\left(y_i, h\left(\mathbf{x}_i\right)\right) = \begin{cases} 0, & \text{if } y_i h\left(\mathbf{x}_i\right) \geq 1 \\ \infty, & \text{if } y_i h\left(\mathbf{x}_i\right) < 1 \end{cases} \qquad [30]$$

and the regularization term is simply $(1/2)\|\mathbf{w}\|^2$. The loss function ensures that the plane separates the two classes, and the regularization term is minimized for the plane with the longest support vectors. This is a constrained quadratic optimization problem that can be solved using *quadratic programming*.[58,59] A similar approach, *least-squares support vector machines* (LS-SVM),[60] enables the calculation of the optimal weights by solving a linear system.

If the training data are not linearly separable, then the loss function in Eq. [30] will always be infinite for at least one element of the training set, and no optimal set of parameters will be found. This problem can be addressed by introducing nonnegative *slack variables*, ξ_i, that allow for some data points to be misclassified.[55] The slack variables effectively measure the distance between the hyperplane and the misclassified data points. The loss function becomes

$$L\left(y_i, h\left(\mathbf{x}_i\right)\right) = \begin{cases} 0, & \text{if } y_i h\left(\mathbf{x}_i\right) \geq 1 - \xi_i \\ \infty, & \text{if } y_i h\left(\mathbf{x}_i\right) < 1 - \xi_i \end{cases} \qquad [31]$$

and the regularization term is $(1/2)\|\mathbf{w}\|^2 + C\sum_i \xi_i$, where C is an adjustable parameter that determines how severely misclassification should be penalized. An alternative equivalent formulation is to use the *hinge loss*:

$$L\left(y_i, h\left(\mathbf{x}_i\right)\right) = \begin{cases} 0, & \text{if } y_i h\left(\mathbf{x}_i\right) \geq 1 \\ C\left(1 - y_i h\left(\mathbf{x}_i\right)\right), & \text{if } y_i h\left(\mathbf{x}_i\right) < 1 \end{cases} \qquad [32]$$

where the corresponding regularization term is $(1/2)\|\mathbf{w}\|^2$. By comparing the loss functions in Eqs. [30] and [32], it can be seen that the hinge loss simply replaces the infinite penalty for misclassification with a penalty that scales linearly with the degree of misclassification.

The optimal set of weights for support vector machines can be found by solving the *dual problem*, in which the weights are written as

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \qquad [33]$$

where the sum is over all elements in the training set. The vector of coefficients $\boldsymbol{\alpha}$ is given by

$$\underset{\alpha}{\arg\max}\left(\sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)\right) \qquad [34]$$

subject to the constraints

$$\begin{gathered} 0 \leq \alpha_i \leq C, \\ \sum_i \alpha_i y_i = 0 \end{gathered} \qquad [35]$$

where each sum is over all elements in the training set. The dual problem formulation makes it possible to use support vector machines to classify data that are not linearly separable using the *kernel trick*, in which the input variables are transformed in a way that makes the data linearly separable.

The Kernel Trick Linear least squares and support vector machines are popular due to their speed and simplicity, which come from the underlying assumption that the solution must be a linear function of the input variables. However in many cases this assumption is unrealistic. For example, consider the classification problem shown in Figure 4a. There is no linear plane that will separate the data, which means that a

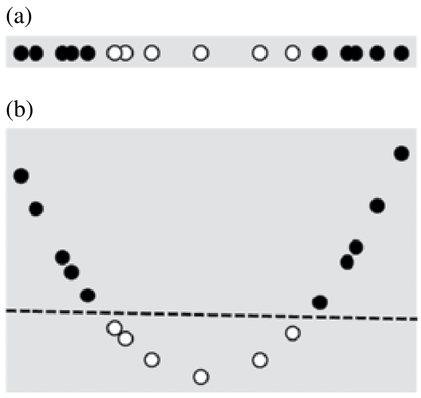


FIGURE 4 (a) A one-dimensional data set that is not linearly separable. (b) The same data set mapped to two dimensions using a feature map that makes it linearly separable. The dashed line is the separating plane.

linear support vector machine will fail to find a good solution. However it is possible to transform the original one-dimensional input value (x_1) to a two-dimensional *feature space*, (x_1,x_1^2), in which linear separation is possible (Figure 4b). Such transformations of the input variables are known as *feature maps*.

Feature maps can be used in machine learning algorithms by simply substituting the transformed input variables, $\varphi(\mathbf{x})$, for the original input variables $\mathbf{x}$. However in some cases, such as when the feature space has an infinite number of dimensions, it may not be practical to explicitly make such a transformation. Alternatively, many algorithms, including support vector machines (Eq. [34]) and linear least squares, can be expressed in a way that depends only on the input variables through inner products such as $\mathbf{x}_1 \cdot \mathbf{x}_2$. In such cases, it is sufficient to find a *kernel* $k(\mathbf{x}_i, \mathbf{x}_j)$ that returns the dot product between $\varphi(\mathbf{x}_i)$ and $\varphi(\mathbf{x}_j)$:

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) \equiv \varphi\left(\mathbf{x}_i\right) \cdot \varphi\left(\mathbf{x}_j\right) \qquad [36]$$

To use a learning algorithm in a given feature space, all that needs to be done is to use the kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ in place of the inner product $\mathbf{x}_i \cdot \mathbf{x}_j$ throughout the learning algorithm. This is known as the *kernel trick*.[61]

The direct use of the kernel function saves the trouble of having to apply a feature map and calculate the inner products explicitly. When using the kernel trick, it is not even necessary to know which feature maps produce the kernel—any symmetric, continuous, positive definite kernel can be used, as every such kernel corresponds to an inner product in some feature space.[62] (A positive definite kernel is defined as a kernel for which $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ for any real-valued $\{c_1, \ldots, c_n\}$ and $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.) Such kernels are known as *Mercer kernels* or *reproducing kernels*. Some examples of reproducing kernels are shown in Table 2.

TABLE 2 Examples of Symmetric, Continuous, Positive Definite Kernels

Name	$k(\mathbf{x}_i, \mathbf{x}_j)$
Linear	$\mathbf{x}_i \cdot \mathbf{x}_j + c$
Polynomial kernel with degree d	$(c_1(\mathbf{x}_i \cdot \mathbf{x}_j) + c_2)^d$
Gaussian	$e^{\frac{-\lVert \mathbf{x}_i - \mathbf{x}_j \rVert^2}{2\sigma^2}}$
Laplacian	$e^{\frac{-\lVert \mathbf{x}_i - \mathbf{x}_j \rVert}{\sigma}}$

The scalar parameters c, c_1, c_2, and σ are all adjustable, with the constraints that σ and c_1 are positive and d is a positive integer.

Reproducing Kernel Hilbert Spaces For a reproducing kernel $k(\mathbf{x}_1, \mathbf{x}_2)$, consider the space of functions f that are given by

$$f(\mathbf{x}) \equiv \sum_j \alpha_j k(\mathbf{x}_j, \mathbf{x}) \tag{37}$$

where α_j are scalar coefficients and the sum may contain any number of terms. The inner product $\langle\rangle_{\mathrm{H}}$ on this space is defined such that

$$\left\langle k(\mathbf{x}_i, \mathbf{x}), k(\mathbf{x}_j, \mathbf{x}) \right\rangle_{\mathrm{H}} = k(\mathbf{x}_i, \mathbf{x}_j) \tag{38}$$

This function space is known as a *reproducing kernel Hilbert space* (*RKHS*).[63] Each RKHS has a norm, $\lVert f \rVert_{\mathrm{H}}$, which is defined as

$$\lVert f \rVert_{\mathrm{H}} \equiv \sqrt{\langle f, f \rangle_{\mathrm{H}}} \tag{39}$$

The norm of the RKHS depends on the underlying kernel. For example, the norm of the linear RKHS is given by

$$\lVert f \rVert_{\mathrm{H}}^2 = w^2 \tag{40}$$

where $f(x) = wx$. Thus in a linear RKHS, functions with a steeper slope will have larger norms. The norm of the Gaussian RKHS is

$$\lVert f \rVert_{\mathrm{H}}^2 = \int_{-\infty}^{\infty} F(\omega)^2 e^{\frac{\sigma^2 \omega^2}{2}} d\omega \tag{41}$$

where $F(\omega)$ is the Fourier transform of $f(x)$. Thus in the Gaussian RKHS, functions that fluctuate more rapidly will have larger norms.

Reproducing kernel Hilbert spaces have special properties that make them particularly useful for supervised machine learning. Consider a situation in which the hypothesis space is an RKHS. If a Gaussian kernel is used, for example, the hypothesis space would consist of linear combinations of Gaussian functions. Let the regularization term be given by $r(\lVert f \rVert_{\mathrm{H}})$, where r is a monotonically increasing function. Let the empirical risk take the general form of ● $_i L(y_i, f(\mathbf{x}_i))$, where the sum is over all data points in the training set. The objective function is therefore

$$\sum_i L\left(y_i, f\left(\mathbf{x}_i\right)\right) + r\left(\|\Delta f\|_{\mathrm{H}}\right) \quad [42]$$

The *representer theorem* then states that the function that minimizes this objective function must take the form

$$\hat{f}\left(\mathbf{x}\right) = \sum_i c_i k\left(\mathbf{x}_i, \mathbf{x}\right) \quad [43]$$

where the sum is over all elements of the training set.[64,65] Thus the problem of finding $\hat{f}$ is reduced to the problem of finding the coefficients c_i that minimize the objective function.

If a squared loss function is used and $r\left(\|\Delta f\|_{\mathrm{H}}\right) = \lambda \|\Delta f\|_{\mathrm{H}}^2$ for some positive scalar λ, then it is straightforward to show that the objective function can be written as

$$\frac{1}{2}\|\mathbf{y} - \mathbf{Kc}\|^2 + \lambda \mathbf{c}^{\mathrm{T}}\mathbf{Kc} \quad [44]$$

where $\mathbf{K}$ is a matrix in which $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for some $\mathbf{x}_i$ and $\mathbf{x}_j$ in the training data and $\mathbf{c}$ is a column vector in which the *i*th element is c_i. The unique solution to this problem is

$$\hat{\mathbf{c}} = \left(\mathbf{K} + 2\lambda\mathbf{I}\right)^{-1} \quad [45]$$

This approach is known as *kernel ridge regression*.[66] It is similar to regularized linear least squares (Eq. [27]) with two major differences. The first is that it is no longer necessary to work in a hypothesis space of linear functions—this solution holds for any RKHS. The second is that the number of rows and columns in the matrix to be inverted is now equal to the number of elements in the training set, rather than the number of input variables. Thus although the calculations may take longer in situations in which there are a lot of training data, this approach can be used for a much wider variety of hypothesis spaces.

The flexibility and simplicity of kernel ridge regression have made it a popular tool in materials science. It has been used for a variety of ends, including the prediction of materials properties from descriptors, development of model Hamiltonians, and generation of density functionals. These are described in more detail in the sections on "Materials Property Predictions Based on Data from Quantum Mechanical Computations," "Development of Interatomic Potentials," and "Developing and Discovering Density Functionals."

Neural Networks An intuitive and effective approach to machine learning is to mimic the biological brain. This is the idea behind a class of machine learning algorithms known as *neural networks*.[67] In a neural network, artificial neurons (a.k.a. nodes) are linked together in a way that resembles the connections between neurons in the brain (Figure 5). The input values ($\mathbf{x}$) are passed directly into a set of neurons

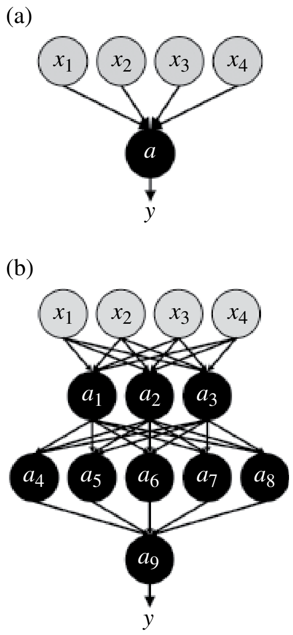


FIGURE 5 (a) A perceptron. (b) A multilayer neural network containing many perceptron-like nodes. Nodes representing input variables (x_1, x_2, …) are gray, and nodes with activation functions (a_1, a_2, …) are black.

that comprise the first layer of the neural network, and these neurons use *activation functions* to calculate output values that are then used as input values by the next set of neurons. This process proceeds throughout the network until reaching a neuron that produce the final output value (y). Some networks may be constructed to output multiple values.

The hypothesis space in a neural network is defined by the topology of the connections between nodes and the parameterized activation functions used by the nodes. One of the simplest neural networks, consisting only of a single node, is known as a *perceptron* (Figure 5a).[68] The activation function of a perceptron compares a weighted sum of the input values to a threshold value. If the weighted sum is larger than the threshold value, the perceptron produces "1" as the output value. If it is lower, the perceptron produces "0" as the output value. Mathematically, we write this activation function as

$$a(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} > b \\ 0 & \text{if } \mathbf{w} \cdot \mathbf{x} \le b \end{cases} \qquad [46]$$

where b is the threshold value and $\mathbf{w}$ is the vector of weights (Figure 6a). The perceptron is a linear classifier that is similar to a support vector machine, where the plane that separates two classes is determined by $\mathbf{w}$. The weights in a perceptron are typically optimized using a gradient descent algorithm to minimize the squared loss.

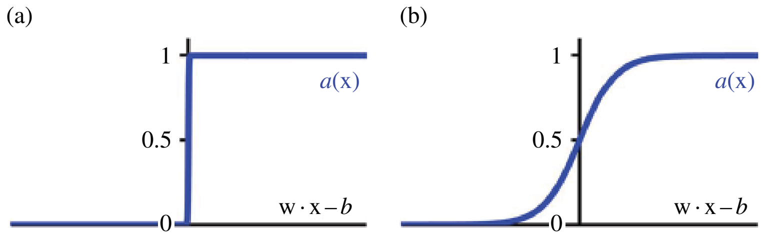


FIGURE 6 (a) The discontinuous activation function in Eq. [46]. (b) A sigmoid activation function.

A perceptron is an example of a *feed-forward* network, in which there are no loops. More complicated feed-forward neural networks can be created by combining multiple perceptron-like nodes in a multilayer network (Figure 5b). Such networks are sometimes referred to as *multilayer perceptrons*. The optimization of the weights of a multilayer perceptron is more complicated than that of single node, but it can be accomplished efficiently using a *backpropagation* algorithm that effectively minimizes the squared loss using gradient descent.[69] However the backpropagation algorithm requires that the activation function is differentiable, which is not the case when using the function in Eq. [46]. To avoid this problem, in multilayer neural networks the discontinuous step function used in Eq. [46] is replaced by a continuous sigmoid function such as a logistic function (Figure 6b). The use of continuous activation functions results in neural networks that can output a continuous range of output values, making neural networks valuable tools for regression as well as classification.[70,71]

Multilayer perceptrons with sigmoid activation functions parameterized using backpropagation have been widely and successfully used for classification and regression. However there are a variety of alternatives to this approach. One alternative is *recurrent neural networks*, in which loops are allowed in the network, enabling the network to model dynamic processes. Different activation functions and optimization algorithms have also been developed to improve the performance of neural networks. A more extensive discussion of the different types of neural networks can be found in Refs. 72 and 73. Across all types of neural networks, regularization is typically done by penalizing the complexity of the network as measured by factors such as the number of nodes in the network and the norm of the network weights.

Neural networks have had a long history of success in materials science and engineering, especially in the development of accurate interatomic potentials and in the mapping of complex materials behavior (flow stress, fatigue behavior, microstructure, etc.) to materials processing parameters (heat treatment, deformation, cold working, etc.). Examples touching on these developments can be found in the sections on "Development of Interatomic Potentials" and "Materials Processing and Complex Materials Behavior," respectively.

Decision Trees Decision trees are among the oldest approaches to machine learning, particularly for classification problems. Historically they have been among the most widely studied machine learning methods, and more comprehensive reviews

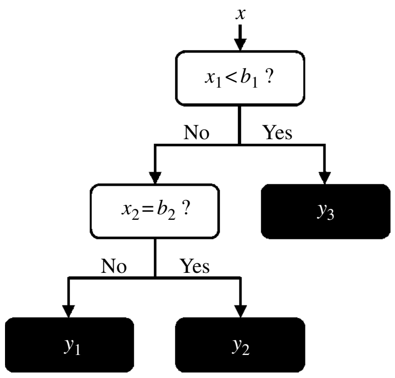


FIGURE 7 An example of a simple decision tree. The leaf nodes are in black.

of decision trees can be found in Refs. 74–77. Within materials science, decision trees have been recently used to predict tribological properties (specifically, the coefficient of friction) of various materials based on easily accessible properties (or descriptors) of the materials and their constituents (e.g., melting point, Madelung constant, density, etc.),[78] as described in the section on "Materials Processing and Complex Materials Behavior." They have also been used to classify zeolite structures based on topological descriptors.[79] Here we provide a brief overview of the idea behind decision trees and some common implementations.

Decision trees are similar to neural networks in that the function is represented as a network of connected nodes. However in a decision tree, the network takes a hierarchical treelike structure, in which each node may only have a single parent node (Figure 7). The evaluation of the function starts at the topmost parent node, known as the *root node*, and proceeds down through the tree until reaching a node with no children, known as the *leaf node*. At each node along the way, there are multiple possible branches, each of which leads to a different child node. The choice of which branch to follow at each node is determined by the value of one of the input variables. Thus the set of all input values determines the path through the tree, and the output value is determined by the leaf node that is reached at the end of the path. Decision trees are commonly used for classification, in which each leaf node corresponds to a different class. However they may also be used for regression, in which each leaf node corresponds to a different numerical value.

There will generally be many different decision trees that are capable of reproducing the training data. The most efficient trees will be those that have, on average, the shortest path between the root and leaf nodes. Unfortunately the problem of finding such trees is *NP-complete*,[80] meaning that it is unlikely that the globally optimal solution can be found using an algorithm that scales as a polynomial of the number of possible output values. Given the computational cost of finding a globally optimal solution, decision trees are commonly constructed using a greedy algorithm that finds a solution by making a locally optimal choice at each node. Some of the most successful algorithms are based on an approach known as *top-down induction of* *decision trees* (*TDIDT*).[81] In the TDIDT approach the tree is recursively built from the training data, starting at the root node. At each node an input variable is selected, and child nodes are created for each possible value of that variable. The training data are then divided into subsets based on the value of the input variable and passed to the appropriate child nodes. The process is then repeated at each child node, with each of the subsets divided and passed on to the next level of child nodes. If any of the subsets are empty, then a child node is not created, and if only one child node is created, then the node becomes a leaf node. A common constraint is that there can be no path through the tree in which the same variable is evaluated more than once.

The TDIDT approach provides a general framework for creating decision trees, but it does not specify the order in which the input variables should be selected. A common approach for classification problems is to choose the input variable that produces subsets with the lowest average information entropy, where the information entropy for a subset is given by

$$-N\sum_i p_i \ln p_i \qquad [47]$$

In Eq. [47] the sum is over all classes in the subset, N is the total number of elements in the subset, and p_i is the fraction of elements in the subset that belong to the ith class. Information entropy is minimized for subsets that contain only one class and maximized for subsets that contain equal numbers of every class, and as a result this approach facilitates the rapid subdivision of the training data into subsets that are pure. A similar approach is to use the Gini index in place of the information entropy,[82] where the Gini index is defined as

$$N\left(1-\sum_i p_i^2\right) \qquad [48]$$

When using a decision tree for regression, there is a continuous range of allowed output values, and it is common for every element in the training set to have a different output value. The objective of the decision tree is to provide an estimate of the output value that is close to the true output value. At each node in the tree, an estimate of the output value can be calculated based on the average of all output values in the leaf nodes below that node (Figure 8). The tree can then be constructed in a way that attempts to minimize the average estimation error along the path. One way to accomplish this within the TDIDT framework is to choose at each node the attribute that minimizes the average variance in output values within each of the subsets.[82,83]

There are two general strategies that can be used to improve the predictive accuracy of decision trees. The first is a regularization approach, in which the complexity of the tree is reduced by *pruning*, or removing, branches of the tree.[82,84,85] The pruned branches are typically those that were poorly represented in the training data and/or are many layers from the root node. For regression, the pruned branches can be replaced by a function (e.g., using linear regression) that estimates the output values for the remaining set of input values. Resampling methods (described in the section on "Estimating the Prediction Error") may be used to determine the optimal degree

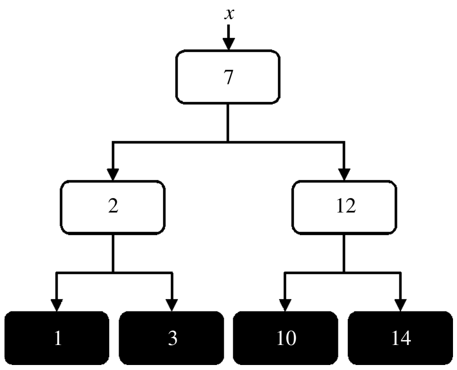


FIGURE 8 An illustration of how the output value could be estimated at each node in a regression decision tree trained on four output values.

of pruning. An alternative approach to improving the prediction power is to generate ensembles of decision trees. For example, in the *random forest* approach, an ensemble of decision trees is created by introducing a stochastic element into the algorithm for constructing trees.[86] The prediction is then based on the mode (for classification) or average (for regression) of predictions made by the members of the ensemble. Variations of this approach, known as *ensemble learning*, are described in the next section.

Ensemble Learning In ensemble learning, the function $\hat{f}$ is created by combining the outputs from an ensemble of different learning algorithms (e.g., by taking the average). This approach can be a simple and effective way to increase the size of the hypothesis space, and it often results in a function with greater predictive accuracy than any of the individual algorithms in the ensemble. It is particularly popular for classification problems.

Various approaches to ensemble learning are well reviewed in Refs. 87–89. Two of the most common classes of algorithms are *bagging* algorithms,[90] in which the members of the ensemble are created by randomly resampling the training data, and *boosting* algorithms,[91,92] in which the training data is resampled in a way that assigns extra weight to input values for which the *losses* (described in the section on "The Empirical Risk") are particularly high. The performance of some boosting algorithms can be adversely sensitive to random classification noise (i.e., erroneous classifications) in the training data.[93]

Genetic Programming It is often desirable to express a function *symbolically*, as a simple combination of variables and basic mathematical operators. For example, the simple formula $F = ma$ is generally preferable to expressing force as a linear combination of many basis functions. The field of identifying such simple formulas that best predict the output values is known as *symbolic regression*. In symbolic regression the hypothesis space contains all symbolic formulas that combine the input variables with a set of mathematical operators, and the space is typically

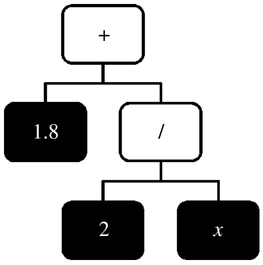


FIGURE 9 A treelike representation of the expression $1.8 + (2/x)$.

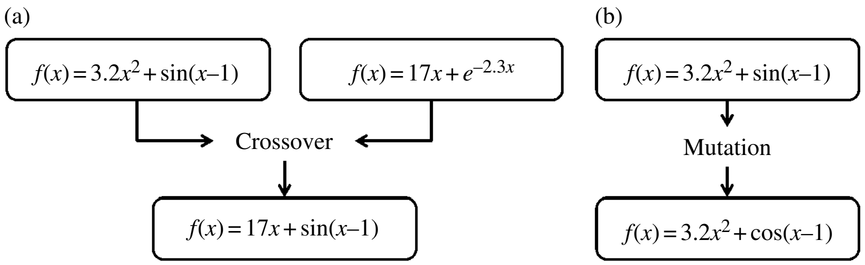


FIGURE 10 (a) An example of a crossover operation. (b) An example of a mutation operation.

regularized in a way that favors simple formulas (e.g., formulas that contain fewer variables or operators). Functions are typically represented in a treelike structure,[94] where input variables and constant values are represented by the leaf nodes and the remaining nodes are mathematical operators or simple functions (e.g., trigonometric functions). An example of such a diagram is given in Figure 9.

As with other supervised learning methods, symbolic regression is typically accomplished by searching through hypothesis space to find the formula that minimizes an objective function. A common way to do this is through *genetic programming*,[95,96] in which the search for the optimal function is performed using a *genetic algorithm*. In genetic programming a population of candidate formulas evolves in a way that mimics the process of natural selection, favoring the formulas that are most *fit*, that is, formulas that produce the lowest values for the objective function. In a typical implementation, an initial population of candidate functions is created, and those that are least fit are discarded. The remaining functions are used to generate a new generation of functions by using *crossover* operations, in which features of the functions are combined to create "children," and *mutation* operations, in which a feature of the function is randomly changed. Examples of crossover and mutation are shown in Figure 10. Repetition of this process results in a series of increasingly fit generations of functions.

The genetic programming algorithm will generate a set of candidate functions with varying degrees of fitness and *complexity*. The complexity of the function may be measured by factors such as the number of nodes in the treelike representation of the function, and it is common for complexity to be measured in a way that penalizes

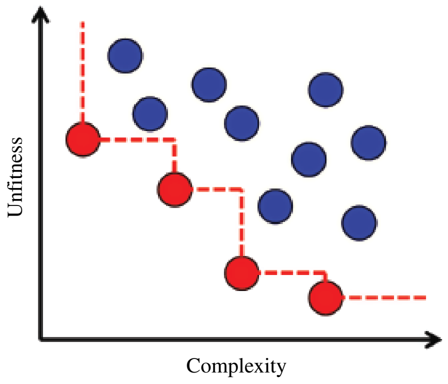


FIGURE 11 The Pareto frontier for a set of functions with different levels of fitness and complexity. Each dot represents a function, and the black dots (lower left dots connected by dashed line) are the ones on the Pareto frontier.

advanced operators such as *acos* more than simple operators like addition. Although it is possible to use a complexity-dependent regularization term in the objective function,[95,97] an alternative approach is to generate a *Pareto frontier* of the functions with respect to fitness and complexity.[98] The Pareto frontier is defined as the set of all candidate functions for which there is no other known function that is both more fit and less complex. The Pareto frontier approach enables a transparent evaluation of the fitness-complexity trade-off in the population of functions generated by the genetic programming algorithm (Figure 11).

In materials science and engineering, genetic programming has been used by several researchers to develop predictive models of the properties of cement, concrete,[99–101] asphalt,[102] and the effects of processing parameters on metal alloys.[103–105] It has also recently been applied at the atomic scale to determine the structural features of hydrogenated amorphous silicon that most strongly influence hole trap depths.[106] This last application is described in more detail in the section on "Structure–Property Relationships in Amorphous Materials," and a thorough review of genetic programming can be found in Ref. 107.

UNSUPERVISED LEARNING

While supervised learning is focused on finding the function (f) that maps a set of input data ($\mathbf{x}$) to a corresponding output value (y), unsupervised learning focuses on finding the relationship among the input data $\mathbf{x}$ themselves. In other words, while supervised learning seeks to determine relationship between $\mathbf{x}$ and y through the conditional density $P\left(f \mid \mathbf{x}, y, g\right)$, unsupervised learning seeks to determine the properties of the joint marginal density $P(\mathbf{x})$.

We illustrate these notions with an elementary example. Twenty data points are drawn from two Gaussian probability density functions (PDFs) of unit standard deviation, one centered at (−3, 0) and the other at (3, 0) (see Figure 12a). The points

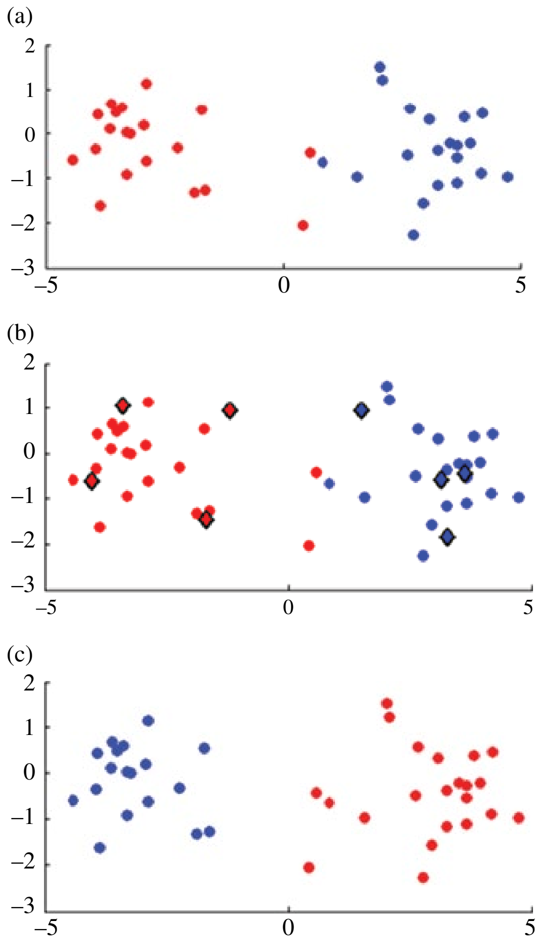


FIGURE 12 (a) Data points generated from two Gaussian PDFs with unit standard deviation and means of −3 and 3. (b) Classification results using SVM with four labeled points indicated with diamonds. (c) Clustering results with *k*-means.

are color coded to identify their PDF of origin. Four additional points (indicated with diamonds) are drawn from each PDF and are appropriately labeled (i.e., *y* values are given). In other words, we are provided *both* **x** and *y* values for a subset of four data points and are asked to classify each of the other data points as to which PDF they came from. One of many supervised learning classification algorithms can be used to accomplish this task, with the resulting color coding shown in Figure 12b (color in e-book version; light gray and dark gray in printed book version). The method used here was *support vector machines* (SVM) (described previously). It is clear that the SVM results are a perfect match to the true labels. However, it might also be apparent that the performance of the SVM method here is strongly dependent upon the training data used.

Alternatively, imagine that we are only told the number of classes (i.e., the number of PDFs used to generate the data), with no labels provided. We are then tasked with sorting the samples into potential clusters associated with the underlying PDFs. This is an example of unsupervised learning, as only the **x** values are provided. Applying the *k-means* unsupervised learning algorithm, discussed in the section "Combinatorial (or *k*-Means) Methods," gives the results seen in Figure 12c. The clustering results can now be used to learn something about the underlying PDF.

In this chapter we group unsupervised learning algorithms into two broad classes: **cluster analysis**, in which the input data are grouped into clusters based on a *similarity measure*, and **dimensionality reduction**, in which the data is represented in a simplified form. When applied to very large collections of data, both cluster analysis and dimensionality reduction are commonly considered *data mining* methods. These approaches are described in more detail in the following sections.

Cluster Analysis

Cluster analysis is the unsupervised parallel to classification. In classification, as discussed in the section on "Support Vector Machines," a set of data with class labels is used to learn rules of data class membership. These rules can then be applied to determine the class membership of unlabeled data. If no labels are given, classification becomes more of an open problem, relying only on *similarity measures* to group data into clusters where data that share a cluster show greater similarity to each other than to data in other clusters. Due to the open nature of cluster analysis, it is generally performed for the goals of either *data complexity reduction* or *exploratory data analysis*.

Complexity reduction is achieved by assigning one representative data value for each cluster that will be used to replace the values of all those in the cluster. This may be an original data point or a function of data in the cluster, such as the mean. For example, if structure micrographs are taken of multiple material samples, where each sample is one of *N* material types, those micrographs can be sorted into *N* clusters corresponding to the material types. One micrograph can then be chosen from each cluster to represent the members of that cluster.

Clustering for exploratory data analysis is used to "present the data to the analyst such that he or she can see patterns and formulate interesting hypotheses about the data."[108] If, in the case of the micrographs of *N* material types, the value of *N* were unknown, clustering analysis will provide an estimate of this number.

When performing cluster analysis, it is necessary to choose a **similarity measure** that will be used to group the data as well as a way to evaluate the **performance** of the algorithm. It is also often necessary to change the way in which the data are represented (i.e., through **dimensionality reduction**). Finally, it is necessary to choose the **clustering algorithm** that is most appropriate for the problem. Because the choice of data representation and measure can often have a greater impact on the final data analysis than do the clustering algorithms used, they should be selected with care. These topics are discussed in more detail in the following sections. Applications of these ideas are described in the later sections on "Phase Diagram Determination" and "Automated Micrograph Analysis."

Similarity and Dissimilarity Measures One of the most important decisions to be made in a clustering algorithm is the *measure* used to evaluate the similarity or dissimilarity between data; this will be discussed further in the example in the section on "Phase Diagram Determination" (similarity measures are also important in supervised learning situations, e.g., within kernel ridge regression alluded to section "Reproducing Kernel Hilbert Spaces"; this will be discussed in the section on "Materials Property Predictions Based on Data from Quantum Mechanical Computations"). A dissimilarity measure takes the place of a distance metric when comparing two pieces of data—the more different the two pieces of data, the greater the value of the measure. Alternatively, a similarity measure increases with greater similarity.

A dissimilarity measure $d(\mathbf{x}_i, \mathbf{x}_j)$ is a *metric* if it meets the following conditions:

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) \geq 0, \quad \text{nonnegativity}$$

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) = 0 \quad \text{if} \quad \mathbf{x}_i = \mathbf{x}_j, \quad \text{identity of indiscernibles}$$

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) = d\left(\mathbf{x}_j, \mathbf{x}_i\right), \quad \text{symmetry}$$

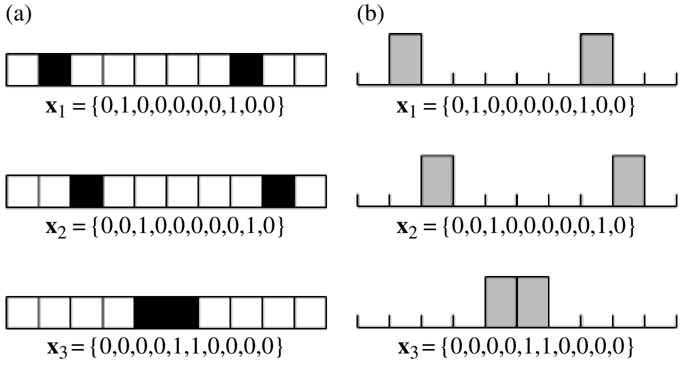
$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) \leq d\left(\mathbf{x}_i, \mathbf{x}_k\right) + d\left(\mathbf{x}_k, \mathbf{x}_j\right), \quad \text{triangle inequality}$$

Measures that meet these conditions can be thought of as measures of distances between data points.

For low-dimensional data, it is common to use the norm of the difference between two data points as a dissimilarity measure. In other words, the dissimilarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ is given by

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left\|\mathbf{x}_i - \mathbf{x}_j\right\| \qquad [49]$$

Higher-dimensional data may require a different choice of measure that preserves aspects of the data structure. For example, consider the situation shown in Figure 13a where each data point is a simple black-and-white image with 10 pixels. The vector



(a)
(b)
$\mathbf{x}_1 = \{0,1,0,0,0,0,0,1,0,0\}$
$\mathbf{x}_1 = \{0,1,0,0,0,0,0,1,0,0\}$
$\mathbf{x}_2 = \{0,0,1,0,0,0,0,0,1,0\}$
$\mathbf{x}_2 = \{0,0,1,0,0,0,0,0,1,0\}$
$\mathbf{x}_3 = \{0,0,0,0,1,1,0,0,0,0\}$
$\mathbf{x}_3 = \{0,0,0,0,1,1,0,0,0,0\}$

FIGURE 13 (a) Three images that are equally dissimilar according to ℓ^1 and ℓ^2 measures. Neither measure properly represents the perceived similarities. (b) Histogram representation of the same data.

x represents the colors of the pixels in the image, with x_i representing the color of the ith pixel. While the image labeled $\mathbf{x_1}$ appears to be more similar to $\mathbf{x}_2$ than $\mathbf{x}_3$, the ℓ^2 (and ℓ^1) measure between any two images gives the same value. For this case, we would prefer a measure that identifies $\mathbf{x_1}$ and $\mathbf{x}_2$ as more similar despite the slight shift in data structures, a common situation in high-dimensional data. These important issues are addressed in the section on "Bin-by-Bin and Cross-Bin Measures."

The choice of measure can greatly impact the efficacy and speed of a machine learning algorithm. This is especially important for complex, high-dimensional data and those machine learning algorithms whose input is a dissimilarity matrix **D**, where

$$D_{ij} = d\left(\mathbf{x}_i, \mathbf{x}_j\right) \qquad [50]$$

There are many measures to choose from, and the decision of which to use generally depends on a cost–benefit analysis between the resultant algorithm performance and the computational cost of evaluating the dissimilarity matrix. For example, the use of *dynamic time warping* (DTW) greatly improves the analysis of diffraction patterns over the use of the ℓ^1 norm but comes at a significantly greater computational cost.[109] In the following sections some common measures are described.

Bin-by-Bin and Cross-Bin Measures The image example shown in Figure 13 falls into the broader class of situations in which **x** can be represented as a set of bins in a histogram (Figure 13b). Such situations include high-dimensional vector data or image data as both can be represented as a list of numbers indexed by dimension or pixel. There are a great number of measures for differentiating between data points using a *bin-by-bin* comparison, in which the elements in the vector $\mathbf{x}_1$ are individually compared to the corresponding elements in another vector $\mathbf{x}_2$.

The most common bin-by-bin measure is the norm of the difference between the two data points, as shown in Eq. [49]. Particularly common choices are ℓ^p norms (Eq. [28]). The ℓ^1 norm, also known as the taxicab distance, is often used for high-dimensional spectral or histogram data as it is less susceptible to outlier values. The ℓ^2 norm, or Euclidean distance, is typically used for 2D or 3D data.

Another measure is the information theory-based Kullback–Leibler divergence, which describes the efficiency of coding one histogram **h** using another histogram **k** as the codebook:[110,111]

$$d_{\mathrm{KL}}\left(\mathbf{h}, \mathbf{k}\right) = \sum_i h_i \ln \frac{h_i}{k_i} \qquad [51]$$

However, this is nonsymmetric. A symmetric alternative is the Jeffreys divergence:[112,113]

$$d_{\mathrm{JS}}\left(\mathbf{h}, \mathbf{k}\right) = \frac{1}{2}\left(\sum_i h_i \ln \frac{h_i}{m_i} + \sum_i k_i \ln \frac{k_i}{m_i}\right) \qquad [52]$$

where $m_i = (h_i + k_i)/2$ and the square root of the Jeffreys divergence is a metric.

A measure for normalized histograms is the Hellinger distance, which was found to provide good performance in distinguishing between histograms while also having a low computational requirement:[114,115]

$$d_{\mathrm{H}}(\mathbf{h}, \mathbf{k}) = \sqrt{1 - \sum_i \sqrt{h_i k_i}} \qquad [53]$$

An information theory- or statistics-based metric is appropriate when dealing with histograms that represent probability densities. For instance, the 1-point statistics of an image (or micrograph), discussed in the section on "*N*-Point Cross-Correlations for *N*-Point Statistics," is a normalized histogram that describes the probability of observing pixels of different states in the image (e.g., see Figure 25b). The 1-point statistics can be used to represent each image when differentiating between samples of different statistical structures.

Although bin-by-bin measures are simple and easy to implement, they fare poorly when features in the data can undergo shifts between neighbor bins or pixels (Figure 13). Such a shift is a typical problem found in microscopy, where two images of the same sample might not be perfectly aligned. A related issue arises when the two histograms are of different lengths.

A solution to the problem of feature shifting and different histogram lengths is to introduce a term that provides an expected correspondence between bin numbers. This results in a *cross-bin* measure. An example is the quadratic form distance:[116]

$$d_A(\mathbf{h}, \mathbf{k}) = \sqrt{(\mathbf{h} - \mathbf{k})^{\mathrm{T}} \mathbf{A} (\mathbf{h} - \mathbf{k})} \qquad [54]$$

where the matrix **A** determines the similarity measure between bin indices. If the expected correspondence between bins is not known, a measure can be used that allows for some drift in features across neighboring bins or different histogram lengths. For example, if feature shifting is possible with a maximum shift length of *L* bins, **A** can be given by

$$A_{ij} = \begin{cases} 0 & \text{for } |i-j| > L \\ 1 - |i-j|/L & \text{for } |i-j| \le L \end{cases} \qquad [55]$$

which allows for features to be compared over a window of *L*.

Structure-Preserving Measures Two structure-preserving measures are the *dynamic time warping* measure (DTW)[117] and the *earth mover's distance* (EMD).[118] DTW began as a measure for analyzing temporal sequence data where two similar sequences may vary in time or speed. For example, DTW is used in automated speech recognition when speaking speed is an issue. DTW has recently found use in materials science for comparing diffraction patterns from samples that differ slightly due to lattice expansion, resulting in diffraction peak shifting and broadening.[109] EMD is a popular technique for comparing images using histogram representations such as 1-point statistics, when issues like histogram length can be a problem.

In *dynamic time warping* (DTW), the dissimilarity measure is determined by the amount of warping needed to map one histogram into another. This measure is, however, not a metric as it is nonsymmetric and does not obey the triangle inequality. For the two histograms **h** and **k** of length N and M, respectively, DTW begins with the construction of the cross-bin dissimilarity matrix **D** (Eq. [50]). The minimum *warping distance* between $(i=1, j=1)$ and $(i=N, j=M)$ is then computed recursively using the function $\gamma(h_i, k_j)$ defined by

$$\gamma\left(h_i, k_j\right) = D_{ij} + \min\left\{\gamma\left(h_{i-1}, k_j\right), \gamma\left(h_i, k_{j-1}\right), \gamma\left(h_{i-1}, k_{j-1}\right)\right\} \qquad [56]$$

A localization constraint can be included, requiring that features should only be considered similar within a window w, in which case the index j is restricted by

$$j \in \left\{\mathbf{max}\left(1, i-w\right), \ldots, \mathbf{min}\left(1, i+w\right)\right\} \qquad [57]$$

An example is shown in Figure 14. The values of **h** and **k** are given in Figure 14a. For $D_{ij} = \left|h_i - k_j\right|$, the DTW path costs are shown in Figure 14b, and the path is shown in gray. The corresponding point-by-point mapping is shown in Figure 14c.

Another shift-resilient measure that has proven popular in the computer vision community is the *earth mover's distance (EMD)* or the *Wasserstein metric*.[118] EMD measures the minimum amount of "work" needed to construct one histogram using another (Figure 15). This can be visualized by thinking of one histogram as a series of mounds of earth and the other as a series of holes. EMD calculates the minimum amount of work needed to transport the earth from the mounds to the holes, where the weight is given by the bin intensity and the distance is given by a measure of bin-to-bin or pixel-to-pixel distance. EMD obeys the conditions of a metric if the distance between bins (or pixels) is defined by a metric and if both histograms have the same total mass. For a more detailed description of the EMD algorithm, see Refs. 118 and 119.

The improved analysis performance achieved by utilizing the DTW or EMD measures comes at the cost of a significantly increased computation. However, fast implementations of each can be found that greatly reduce their computational cost.[120–122]

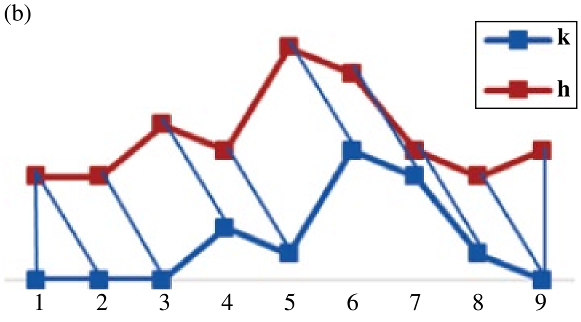
For data features that undergo more extreme changes but still require a high measure of similarity, the data is typically first converted to a new representation, using feature extraction methods in which the feature changes become negligible and have minor impact on the measure. An appropriate moment invariant feature extraction method will allow for similar features to be identified despite changes in location, scale, orientation, shear, and/or dilation.

Clustering Algorithms Many types of clustering algorithms exist. While a theoretical taxonomy of clustering methods has yet to be developed, there are several general groupings of clustering methods. In the following sections, four of these general groups are described: **combinatorial methods**, **mode seeking**, **mixture models**, and **hierarchical clustering**.

(a)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
h	0	0	2	1	5	4	1	0	1
k	0	0	0	2	1	5	4	1	0

(b)



(c)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	0	0	2	3	8	12	13	13
2	0	0	0	2	3	8	12	13	13
3	2	2	2	0	1	4	6	7	9
4	3	3	3	1	0	4	7	6	7
5	8	8	8	4	4	0	1	5	10
6	12	12	12	6	7	1	0	3	7
7	13	13	13	7	6	5	3	0	1
8	13	13	13	9	7	10	7	1	0
9	14	14	14	10	7	11	10	1	1

FIGURE 14 (a) Histograms **h** and **k**. (b) DTW mapping between **h** and **k**. (c) DTW path cost matrix with minimal cost path of 1 shown in gray.

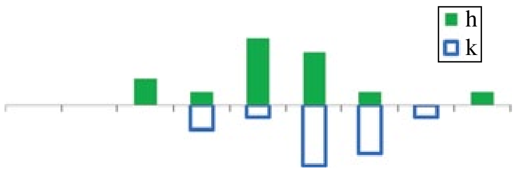


FIGURE 15 Histograms **h** and **k** with **h** shown as EMD earth mounds and **k** as holes.

Combinatorial (or k*-Means) Methods* Combinatorial methods utilize the similarity between data to sort the samples into clusters. This is accomplished by using a dissimilarity matrix to achieve a minimization of intracluster scatter (or equivalently, maximizing the intercluster scatter), defined as

$$W(C) = \frac{1}{2}\sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)=k} d(\mathbf{x}_i, \mathbf{x}_j) \qquad [58]$$

where C is a function that assigns a cluster label to a data point, the outer sum is over all clusters, and the inner sums are over all pairs of points within each cluster.

Filtering through all possible clustering configurations for large data sets is infeasible, so most combinatorial methods rely on iterative greedy descent to optimize clustering results. The most common combinatorial method, *k-means*, uses this approach. In the k-means algorithm, the intracluster scatter $W(C)$ is defined as

$$W(C) = \frac{1}{2}\sum_{k=1}^{K} \sum_{C(i)=k} \|\mathbf{x}_i - \bar{\mathbf{x}}_k\|^2 \qquad [59]$$

where $\bar{\mathbf{x}}_k$ is the mean of the points in cluster k. We assume here that the number of clusters K is known and that each point can be assigned only to one cluster. The k-means algorithm minimizes $W(C)$ with an iterative decent method.

The algorithm is initialized by first randomly distributing k cluster center points $\mathbf{v}_k$ in the data vector space. Each data point is then assigned a cluster based on the nearest center point. For each cluster, the mean $\bar{\mathbf{x}}_k$ is calculated and the k cluster centers are moved to the cluster mean: $\mathbf{v}_k = \bar{\mathbf{x}}_k$. Each data point is then reassigned to a cluster based on proximity to the new set of center points. The process repeats until the assignments of all points remain constant. Because the results depend on the initial choice of the center points, this method is typically repeated with multiple initial random assignments, and the cluster configuration that minimizes $W(C)$ is retained. An example of this algorithm is illustrated in Figure 16.

The k-means algorithm assumes that clusters are spherical in shape and of similar size and density. For cases where these assertions are not true, k-means will not perform well. Outliers and empty clusters will also provide difficulty.

Statistical Techniques: Mixture Models and Mode Seeking Both *mixture models* and *mode-seeking methods* assume the data are statistical in nature, and the probability of observing each data point is described by a probability density function (PDF) $P(\mathbf{x})$ over the data vector space. The PDF is assumed to be the sum of *class-associated PDFs* $P_i(\mathbf{x})$ each associated with a different underlying classification. $P(\mathbf{x})$ is then given by

$$P(\mathbf{x}) = \sum_{i=1}^{K} \pi_i P_i(\mathbf{x}) \qquad [60]$$

where π_i are normalizing constants. Mixture models assume that the general type of each class-associated PDF is known (e.g., Gaussian, Bernoulli, etc.); however, each

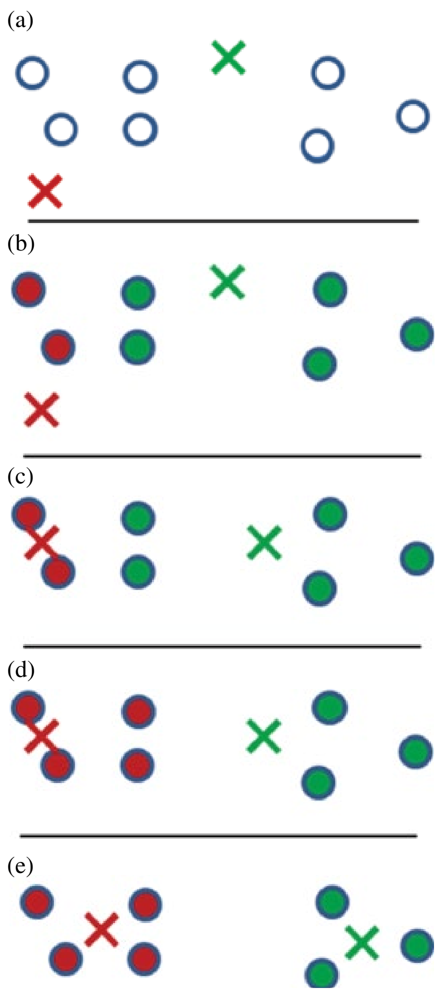


FIGURE 16 k-Means for simulated data with $k=2$: (a) cluster centers indicated by "X" are initialized, (b) cluster assignment based on proximity to centers, (c) cluster centers computed, (d) cluster assignment recomputed, and (e) convergence.

class-associated PDF has a set of unknown parameters including its location in the vector space. The most likely values of these parameters are determined given the data **x** using a gradient descent method. Similarly the most likely generating class-associated PDF is determined for each datum, and the class association is used to determine cluster membership. In the particular case where all $P_i(\mathbf{x})$ are Gaussians—the *Gaussian mixture model*—$P(\mathbf{x})$ can be rewritten:

$$P(\mathbf{x}) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{x} \mid \mu_i, \Sigma_i) \qquad [61]$$

where the unknown parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the D-dimensional mean (for an $\mathbf{x}$ of dimension D) and $D \times D$-dimensional covariance, respectively. These parameters are initialized and the most likely class association $z \in \{1,\ldots,K\}$ for each datum is determined by evaluating $p(z \mid \mathbf{x})$ given by

$$p\left(z=j \middle| \boldsymbol{x}\right)=\frac{\pi_j \mathcal{N}(\mathbf{x} \mid \mu_j,\Sigma_j)}{\sum_{i=1}^{K}\pi_i \mathcal{N}(\mathbf{x} \mid \mu_i,\Sigma_i)} \qquad [62]$$

The class assignments are then used to recalculate the PDF parameters. The cycle is repeated using a gradient descent method until convergence in parameter values and class associations.

Mode-seeking methods also assume that $P(\mathbf{x})$ is composed of class-associated PDFs, but the general type of each $P_i(\mathbf{x})$ is unknown other than that each $P_i(\mathbf{x})$ has one major mode (i.e., the maximum of the PDF). Mode-seeking methods utilize the local density of points to estimate the topology of $P(\mathbf{x})$ empirically and assign each data point to the most likely mode and thus to a cluster.

For data of high dimension, evaluating the empirical PDF over the entire vector space can be infeasible. One solution is to evaluate the PDF only in the vicinity of data points. *Mean shift theory*,[123] a popular mode-seeking method, utilizes this technique to "walk" each data point toward its mode by following increasing local densities of data points. This is accomplished through the use of a *window function* to identify local regions of points. The window function has a nonzero value within the region and a value of zero otherwise. A typical window function is the truncated Gaussian. The algorithm begins by either partitioning the vector space in the vicinity of data points into regions enclosed by a window function or by centering a window function at each data point. Each point within each region is then assigned a mass by evaluating the window function at its location. The center of mass within the window is computed, and the window center is then shifted to this location. An illustration of this process can be found in Figure 17. The process is repeated until the center of mass reaches a *convergence point* at a local density maximum for each starting region. Points that fall within the original regions are identified with their convergence point, and the convergence points are given class labels.

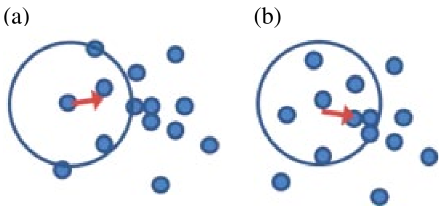


FIGURE 17 An illustration of mean shift theory: (a) center of mass computed for region of interest indicated by circle and (b) region of interest is shifted to center of mass. Center of mass recomputed.

Hierarchical Cluster Analysis Clustering algorithms can provide either flat results or hierarchical results. Flat methods provide only one clustering result as a function of algorithm parameters. It is up to the user to vary the algorithm parameters to achieve other clustering results. Hierarchical clustering analysis (HCA) provides a range of clustering resulting in a binary tree or dendogram (Figure 27). There are two methods of HCA—agglomerative, the bottom-up approach, and divisive, the top-down approach.

Agglomerative HCA begins with each data point in its own cluster. At each higher level in the tree, the two clusters with the smallest intercluster dissimilarity are merged. There are three common types of agglomerative HCA derived from the choice of dissimilarity metric (Figure 18). *Single linkage* agglomerative clustering defines cluster dissimilarity as the dissimilarity between the two points (one in each cluster) that have the minimum dissimilarity. *Complete linkage* HCA defines the cluster dissimilarity as that of the two points with the greatest dissimilarity. *Group average* HCA uses the average dissimilarity between the two groups. A discussion of the pros and cons of the different dissimilarity metrics can be found in Ref. 32.

Divisive HCA begins with all data points in one cluster. At each lower level in the tree, the cluster with the largest intracluster dissimilarity is split into two. Various methods have been proposed to accomplish this. One method is to use k-means clustering with $k=2$. Another method proposed by Macnaughton et al.[124] involves taking the data point with the largest average intracluster dissimilarity, removing it from the initial cluster A and placing it into a new cluster B. Points are moved one by one from A to B by selecting those points with the largest average dissimilarity to points in A minus the dissimilarity with points in B. The process ends once the remaining points in A are each less dissimilar to the others in A than to the points in B. Further discussion of the divisive algorithms can be found in Ref. 32.

Model Evaluation and Selection The variety of different clustering algorithms available presents a challenge of how to choose the best algorithm for a given problem. For some algorithms, such as k-means, it is also necessary to choose the number of

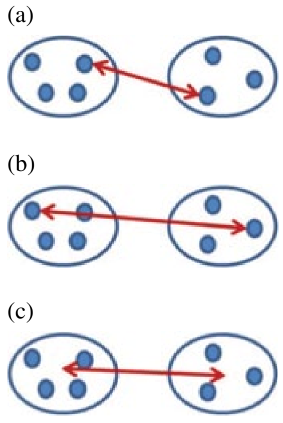


FIGURE 18 Agglomerative HCA cluster dissimilarities: (a) single linkage, (b) complete linkage, and (c) group average.

clusters. Supervised learning methods have "teacher guidance" in the form of known output values (y) in the training set, whereas clustering analysis does not generally have labeled data. Thus in clustering analysis there is generally no way to evaluate the empirical risk associated with a particular choice of clusters or to use cross-validation (described in the section on "Estimating the Prediction Error") to evaluate model quality. For this reason, clustering evaluation is often accomplished by having an expert inspect the clustering results. In situations where data labels are provided for a small data subset or information is known of the function that generated the data, a testing scheme similar to cross-validation can be used to determine performance. First, the algorithm is tested on simulated data with a similar generating function. Then the algorithm is tested on labeled data to evaluate performance. Finally, the algorithm is run on unlabeled experimental data and inspected by an expert.

An alternative approach that has become popular for evaluating clustering algorithms is *cluster stability*. A clustering result is considered *stable* if similar results are obtained when the algorithm is tested on different sets of data drawn from the same underlying models. In that sense, this approach is similar to cross-validation. For algorithms such as k-means, in which the number of clusters is an unknown parameter that must be set, clustering stability is commonly implemented to select the number of clusters to use. There are a variety of different measures of cluster stability, and a good discussion of these can be found in Ref. 125.

Determining the number of clusters to include in a clustering algorithm is similar in some ways to the problem of choosing the number of parameters to fit in a supervised learning problem. If too many clusters are chosen, there is a risk of overfitting the data—for example, consider the extreme case in which every data point is assigned to its own cluster. Contrarily, if too few clusters are chosen, then the clusters might not represent the data well. As in the case of supervised learning, these concerns can be balanced by selecting the number of clusters that optimizes measures such as the Akaike information criterion or Bayes information criterion (described in the section on "Model Selection" for supervised learning).[126] However this approach requires some way of estimating the likelihood of the result returned by the clustering algorithm. Several additional approaches have been developed to choose the number of clusters for use in a clustering algorithm, and a good review of these can be found in Ref. 127.

Prior Knowledge Prior knowledge of the data can aid in algorithm selection. For instance, if the data generation is known to be statistical, a statistical clustering method may provide the best results. If clusters are expected to have varying densities and sizes, a mixture model might be a good choice. If the clusters are nonglobular in shape, a graphical model may provide adequate results. Also, certain algorithms are known to provide good results at low computational cost for high-dimensional data. In practice, there are well-established clustering methods that have become popular, and they are often the first go-to algorithms tested. These include two of the algorithms described here: k-means and hierarchical cluster analysis.

Sometimes prior knowledge can be incorporated in the clustering algorithm through the use of constraints on the allowed results. This approach is analogous to restricting the hypothesis space in supervised learning. The clustering algorithm is expressed as a constraint satisfaction problem, in which a set of constraints are placed over the possible cluster labels for each data point. The constraints allow for the integration of prior knowledge into the solution requirement and permit use of information from different domains. For example, in the case of phase diagram generation, a constraint can be imposed to require that points clustered together in a phase region should be similar in structure and inhabit a connected region of the ternary composition space. Constraint satisfaction problems can be expressed in a *constraint programming* framework that provides an environment in which constraint satisfaction problems can be stated and solved, without specifying any one solution method. Constraint satisfaction problem algorithms can provide one solution, all possible solutions, or the optimal solution given an objective function. More information on constraint satisfaction problem and constraint programming can be found in Ref. 128.

Dimensionality Reduction

The type of clustering algorithm used is typically determined by the dimensionality of $\mathbf{x}$. There are many effective algorithms for clustering low-dimensional data, especially for the more easily visualized data of three dimensions or less, but higher-dimensional data falls prey to the "curse of dimensionality"—data points become highly sparse due to the large volume of the data vector space, resulting in poor clustering analysis and increased computational costs.

Different solutions exist to deal with the curse of dimensionality. These solutions provide a lower-dimensional data representation that results in appropriate or improved clustering results while reducing computational costs. One solution is dimension reduction through **latent variable analysis** (LVA). This method relies on the fact that the data points exist on or near a manifold of lower dimension than the vector space. The manifold is described by a set of latent variables that are more representative of the samples being clustered.

Another solution is to select a list of data structures, or features, that better represent the samples. The data is preprocessed using a set of **feature extraction** algorithms and mapped to a lower-dimensional feature space. Feature extraction requires prior knowledge of the representative features. For example, rather than using a million pixel image of a fingerprint to identify a unique user, the existence and location of fingerprint features such as ridge terminations and ridge bifurcation can be identified in the images and used to map and identify users in a lower-dimensional fingerprint feature space.

Latent Variable Analysis Latent variable analysis techniques comprise a subset of unsupervised methods used for dimension reduction, compression, and data visualization. LVA is of particular interest for treating high-dimensional data that exists on or near a lower-dimensional manifold described by "latent variables." For example, if a 1000 pixel image is padded with zeroes and then translated in its new borders to produce a set of output images, the output images will exist on a two-dimensional manifold associated with the latent variables of x and y translations. Identifying the images with their location on the 2D manifold provides a low-dimensional means of representation. Furthermore, while visualizing the set of images in the 1000-dimensional space may be difficult, visualizing the images as points in 2D is not.

Principal Component Analysis One of the most popular methods for latent variable analysis is principal component analysis (PCA). PCA identifies a linear subspace onto which (i) orthogonal projections of data have a maximum variance and (ii) which minimizes the mean squared distance between the data and its projections (Figure 19). The subspace therefore is the best linear representation of the collection of data.

The linear subspace is found by first normalizing the data. That is accomplished by subtracting the mean of each M-dimensional variable to get the normalized data matrix $\mathbf{Y}$, where the ith row of $\mathbf{Y}$ is $\mathbf{x}_i - \bar{\mathbf{x}}$. The covariance matrix $\mathbf{S} = \mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ is then computed and evaluated for its eigenvectors $\mathbf{u}_i$ and corresponding eigenvalues λ_i:

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{63}$$

The PCA subspace of dimension D is given by selecting the D eigenvectors with the largest eigenvalues. The eigenvectors are called *principal components*, and the corresponding eigenvalues describe the data variance along that principal component. The set of D principal components describes a subspace, within the original data vector space, where the principal components are the orthogonal basis. Projecting the data vectors onto this basis gives a reduced data representation of dimension D:

$$y_{\mathrm{PCA},i} = \mathbf{y}^{\mathrm{T}}\mathbf{u}_i \tag{64}$$

where $\mathbf{y} = \mathbf{x} - \bar{\mathbf{x}}$. The original data can then be reconstructed by summing over the contributions along each principal component:

$$\mathbf{x} \approx \sum_{i=1}^{D} \left(\mathbf{y}^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i + \sum_{i=1}^{D} \left(\bar{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i \tag{65}$$

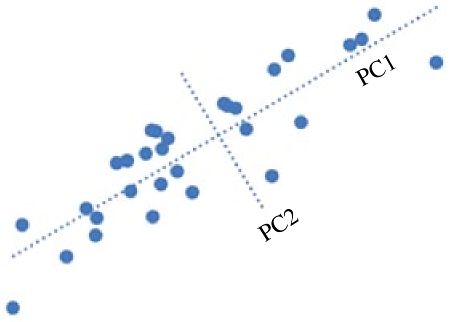


FIGURE 19 The first two principal components, PC1 and PC2, for data generated using a Gaussian PDF.

If the sums in Eq. [65] contain all principal components, then the right side reproduces **x** exactly. However if some of the principal components with the smallest eigenvalues are left out of the sums, then the right side of Eq. [65] provides an approximation for **x**. This approach is sometimes used to reduce the amount of noise in a data set.

By choosing the first two or three principal components (setting D to 2 or 3), the PCA representation can be used to visualize high-dimensional data in the reduced PCA space. As an example, samples of varying composition from the Fe–Co–Ni material system were characterized for their powder patterns (integrated X-ray diffraction patterns) and are shown in Figure 20a.[129] Five samples were drawn from one region of the composition space (red) and five from another (blue) (color available in e-book version). The powder patterns shown are each a list of 89 intensities for corresponding 2θ values or each are 89-dimensional vectors that are impossible to visualize for analysis by inspection. The two-dimensional ($D=2$) PCA representation of these powder patterns is shown in Figure 20b. Here one can clearly see that the 2D principal components describe a space in which the 10 powder patterns are easily separated by their locations in the composition space.

Typically, the principal components can also be investigated as vectors themselves—vectors that describe the orthogonal basis of greatest variance in the original vector space. The first two principal components are shown in Figure 21a. Correspondence can be seen between the positive peaks of these PCA-based powder patterns and the original data. However, the first principal component does not represent a realistic powder pattern due to its negative values. Nonnegative matrix factorization, described in the next section, is used when the components are required to be positive definite.

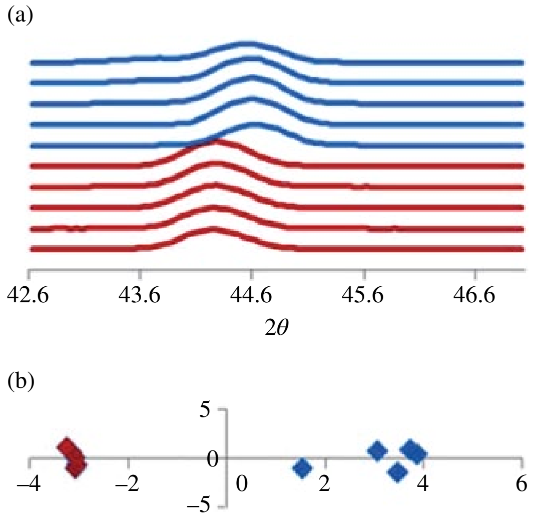


FIGURE 20 (a) Powder patterns for 10 samples from Fe–Co–Ni combinatorial library ternary spread. (b) The powder patterns projected into first two principal components.

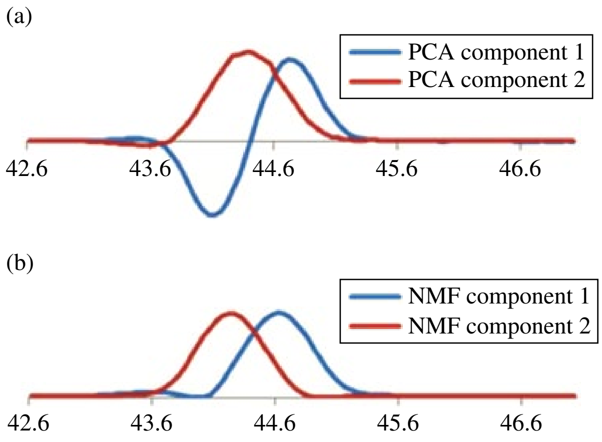


FIGURE 21 (a) The first two principal components for the powder diffraction patterns shown in Figure 20. (b) The first two components for NMF, which look like original powder patterns.

When the number of data points N is smaller than the number of data dimensions D, those data points inhabit an N-dimensional subspace of the D-dimensional vector space. As a result, only $N-1$ principal components will have nonzero variance and nonzero eigenvalues. In this situation it is possible to reduce the computational cost of eigenanalysis of the D^2 matrix $\mathbf{S}$, by replacing $\mathbf{S}$ with the matrix $\mathbf{S}' = (1/N)\mathbf{X}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{u}_i$ with $\mathbf{v}_i = \mathbf{X}\mathbf{u}_i$. The new matrix $\mathbf{S}'$ has reduced dimensions of N^2, simplifying eigenanalysis. Recovering the principal components only requires the simple computation $(1/(N\lambda_i)^{1/2})\mathbf{X}^{\mathrm{T}}\mathbf{v}_i$.

Nonnegative Matrix Factorization Non-negative Matrix Factorization (NMF) is a means of identifying components under the constraint that those components must be positive definite. This method is typically used when the original data is positive definite and one wants the components to resemble the original data. The first two NMF components for the data in Figure 20a are shown in Figure 21b. While the first two PCA components do not look like powder patterns, the first two NMF components look very much like the two types of powder patterns in the original data. Although useful in data analysis, NMF components are not unique as they are based on the initial values used in determining the components.

Metric Multidimensional Data Scaling Metric multidimensional data scaling (MMDS) methods project high-dimensional data into a lower dimension while attempting to preserve pairwise distances. This is done by minimizing a loss function. An illustrative application of this method for phase diagram determination using X-ray diffraction data is presented in the section titled "Phase Diagram Determination."

For example, consider the specific case of *classical multidimensional scaling*, also known as *principal coordinate analysis*. Let $d(\mathbf{x}_i, \mathbf{x}_j)$ represent the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, and let $\mathbf{z}_i$ represent the projection of $\mathbf{x}_i$ onto a lower-dimensional plane. The projection plane is chosen in a way that minimizes

$$\sum_{i,j}\left[d\left(\mathbf{x}_i,\mathbf{x}_j\right)-d\left(\mathbf{z}_i,\mathbf{z}_j\right)\right]^2 \qquad [66]$$

This problem can be solved in a way similar to principal component analysis, except that the eigendecomposition of the dissimilarity matrix $\mathbf{D}$ is used in lieu of the covariance matrix $\mathbf{S}$. In general, *metric multidimensional scaling*, the dissimilarity function $d(\mathbf{x}_i, \mathbf{x}_j)$, can be any metric—it need not be the Euclidean distance.

Feature Extraction Feature extraction is the process of identifying pertinent structural information in data and is typically performed as a preprocessing step along with operations such as *noise smoothing* and *bias subtraction*. Feature extraction can be highly useful when comparing similar data from different instruments. It is also useful when dealing with transformed images or crystals (see discussion in the section on "Materials Property Predictions Based on Data from Quantum Mechanical Computations") where suitable distance metrics must be defined to account for such transformations as well as for identifying grain boundaries and shapes in automated micrograph analysis (see discussion in the section on "Automated Micrograph Analysis"). Sets of data originating from different instruments can be reduced to sample-specific features, thereby removing the effects of the instruments, and then mapped into the same feature space for shared analysis. Features are selected to be both domain and task specific. For example, a surface topography image can be reduced in both dimension and complexity to the single scalar value of average texture for the task of comparing sample roughness. Alternatively, the same image could be reduced to a histogram of peak heights for the task of comparing nanostructured surfaces. Feature extraction is commonly used to analyze images, where generic features include 1D peaks and their 2D counterpart "blobs," edges, corners, ridges, lines, ellipses, and other shapes, as well as shapes of different scales and orientations.

Data Preprocessing: Noise Smoothing and Bias Subtraction Preceding feature extraction, experimentalists frequently deal with issues of signal bias and noise in high-dimensional data. A bias is an unwanted systematic signal added to each measurement, while noise is an unwanted statistical variance in the measurements; both can come from a collection of different sources associated with the measurement instrument and the sample itself. For instance, when measuring X-ray diffraction, unwanted background signal may appear due to the sample substrate (e.g., diffraction peaks associated with a silicon substrate) as well as from the "background radiation" associated with the instrument setup. Sample-dependent bias and background bias can be mitigated by subtracting the baseline signal measured from a "blank" sample. Other instrument-dependent biases and noise are typically characterized and discussed in the associated instrument literature.

When information about the bias is not known, a common method for quantifying bias involves curve fitting, where the curve is typically a polynomial function

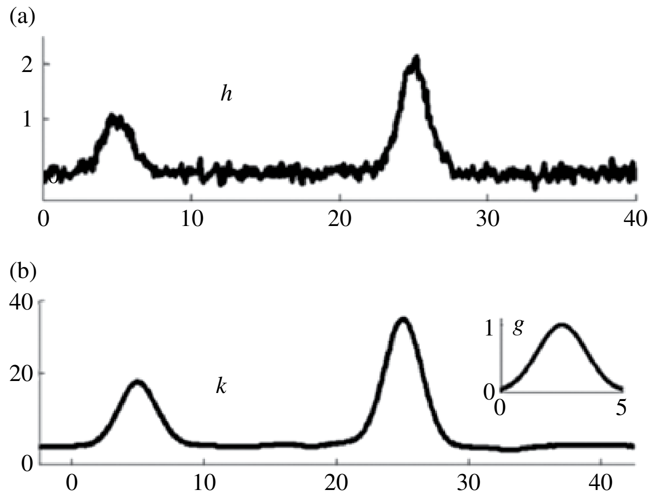


FIGURE 22 (a) An original noisy signal h with two Gaussian peaks. (b) The Gaussian filter g and the convolved signal $k = h * g$ with local maxima at the location of the original peaks and reduced noise.

of low degree or a spline function. If the desired measurement signal is positive with a minimum of zero, the bias curve is fit to the bottom of the signal and then subtracted. Alternatively, the bias is automatically removed when subtracting the mean of each variable, a common data preprocessing technique for machine learning algorithms.

While the bias is characterized by a signal, the noise is characterized with intensity, often measured in decibels, as a function of frequency. When the noise profile is known, data can be preprocessed with an appropriate band-pass filter. Because the analyst does not typically have access to the instrument characterization information, various noise smoothing techniques are used to reduce the influence of noise in a signal. One approach is to use a moving average filter, which replaces each signal value with the average of its neighbors within a given range. Another is to use a Gaussian smoothing filter, which convolves the signal with a windowed Gaussian of given standard deviation (Figure 22). This method replaces each original signal value with a weighted average of its neighbors, with closer neighbors given a greater weight than those further away. Another technique involves fitting a smoothing spline, which uses a least-squares local fit to replace the original signal with a series of continuous piecewise polynomials. Many other methods for bias subtraction and noise smoothing exist; additional information can be found in Refs. 130 and 131.

Cross-Correlation and Wavelets Consider a situation in which the data can be expressed as a continuous (or nearly continuous) function in space, such as a high-resolution image. We will call this function $h(x)$. Now consider a feature in the form of a localized pattern to be matched, expressed as a function $g(x)$. Cross-correlation can be used to identify regions of the data that match the desired feature well. In one dimension, the cross-correlation is given by

$$(h * g)(x) = \int_{\tau} f^{*}(x) g(x+\tau) d\tau \qquad [67]$$

and the integral is easily extended to multiple dimensions. Peaks in the output of the cross-correlation identify the location of the feature signal in the data. Similarly, for discrete signals, the cross-correlation is

$$(h * g)(n) = \sum_{m} \mathbf{h}^{*}[m] \mathbf{g}[n+m] \qquad [68]$$

where $\mathbf{h}^*[m]$ represents the complex conjugate of the mth element of $\mathbf{h}$ and $\mathbf{g}[n+m]$ represents the $(n+m)$th element of $\mathbf{g}$. An example is given in Figure 22, where an original noisy signal $\mathbf{h}$ with two Gaussian peak features of standard deviation 1 is shown. To identify the peaks, a Gaussian feature signal $\mathbf{g}$ is convolved with $\mathbf{h}$ giving the signal $\mathbf{k}$ with local maxima at the locations of the original peaks. Convolving a signal with a Gaussian has an added benefit of noise smoothing.

This process can be generalized to identify a family of feature signals that vary in properties such as scale and orientation through the use of wavelets. In Eq. [67] the function $g(x)$ is replaced with a parameterized "mother wavelet" $w_\theta(x)$, where θ is a set of parameters that control the size and/or shape of the wavelet.

A common wavelet for detecting 1D peaks and 2D round blobs of different scales is the Laplacian of Gaussian (LoG) wavelet. In two dimensions, this wavelet is given by

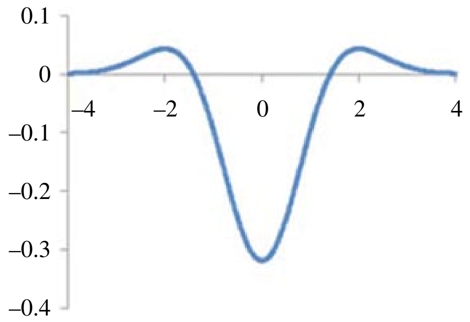
$$w_a(x, y) = -\frac{1}{\pi a^2}\left(1 - \frac{x^2+y^2}{2a}\right) e^{\frac{-(x^2+y^2)}{2a}} \qquad [69]$$

This wavelet (shown in Figure 23) is the convolution of a Gaussian filter for noise smoothing and a Laplacian for detection of sharp spatial derivatives. Peaks or blobs of scale a are identified in the LoG output with sharp peaks at the center of the feature in the original data.

Edges and Closed Boundaries It is sometimes important to identify different regions in an image as, for instance, identifying boundaries between different grains. These boundaries can then be extended and connected to detect regions of heterogeneity and to identify their shape.

A common method for edge detection in 2D data is the *Canny algorithm*. The original image is first smoothed by convolving with a Gaussian filter to reduce the effect of single pixel noise on edge detection. The "edge strength" and "edge angle" of each pixel is then determined through the use of image gradients, computed by convolution with *gradient masks*. For example, the Sobel gradient masks are convolved

(a)



(b)

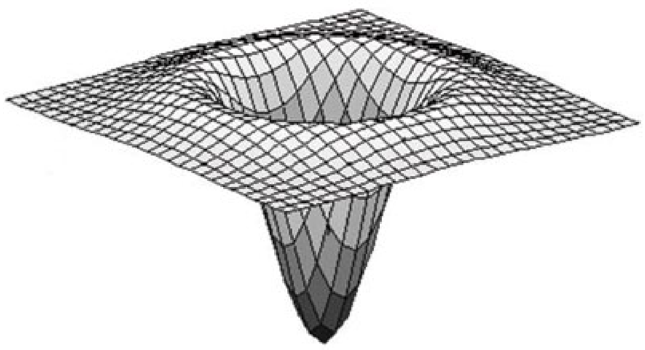


FIGURE 23 (a) 1D Laplacian of Gaussian (LoG) with $a = 1$. (b) 2D LoG given by Eq. [69].

with image **h** to give the *x*-directional image gradient G_x and the *y*-directional image gradient G_y:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{h},\ G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{h} \qquad [70]$$

The edge strength G and the edge angle θ are then given as

$$G = \sqrt{G_x^2 + G_y^2},\ \Theta = \arctan\left(\frac{G_y}{G_x}\right) \qquad [71]$$

An upper edge strength threshold is used to identify the start of an edge. The edge is then followed using its direction, until edge strength drops beneath a lower threshold. Once edges have been identified, they can be used to identify closed boundaries for cohesive analytically defined shapes such as ellipses through the Hough transform or more general shapes through the generalized Hough transform.

Shape Identification with Moment Invariants Having identified closed boundaries, the enclosed regions can be quantified for shape using *moment invariants*, so called for their invariance to effects such as changes in location, scale, or orientation and in some cases shear and dilation. For multiple images in which the object's shape is changing, moment invariants can also be used to quantify and track the change.

First, each object must be defined by an indicator function:

$$D_i(\bar{r}) = \begin{cases} 1 & \text{for } \bar{r} \text{ inside the object} \\ 0 & \text{for } \bar{r} \text{ outside the object} \end{cases} \qquad [72]$$

where $\bar{r}$ is a vector in the image space. The 2D moments of an object is then given by

$$\mu_{pq} \equiv \iint d^2 r x^p y^q D(r), \quad \{p,q\} \in \mathbb{Z}^+ \qquad [73]$$

where the summation is performed over the object area A and both p and q are positive integers. With respect to the center of mass of the object of interest, the central moments are given by

$$\bar{\mu}_{pq} \equiv \iint d^2 r \left(x - \frac{\mu_{10}}{A} \right)^p \left(y - \frac{\mu_{01}}{A} \right)^q D(r) \qquad [74]$$

These moments can be made invariant to the set of similarity transforms including translations, rotations, and isotropic scaling or the larger set of affine transforms, which also include homogenous shears and anisotropic dilations. A pair of invariant moments, ω_1 and ω_2, have been used to identify and track particle shape in micrographs:[132]

$$\omega_1 = \frac{A^2}{2\pi\left(\bar{\mu}_{20} + \bar{\mu}_{02}\right)} \quad \omega_2 = \frac{A^4}{16\pi^2\left(\bar{\mu}_{20}\bar{\mu}_{02} - \bar{\mu}_{11}^2\right)} \qquad [75]$$

ω_1 is invariant to similarity transforms, while ω_2 is considered an absolute moment invariant—invariant to the full set of affine transformations.

Computing these moments for an object of interest gives a point in the ω_1, ω_2 vector space as shown in Figure 24. The region's shape is given by its location in the ω_1, ω_2 space. Points along the curve $\omega_2 = \omega_1{}^2$ indicate N-sided symmetric polygons, with lower symmetry shapes falling to the left of the curve, for example, rectangles of different length–width ratios fall to the left of the (ω_1, ω_2) point indicated for the square. A larger set of moment invariants, including higher-order moments that can distinguish between right- and left-handed versions of noncentrosymmetric objects, has also been used in automated micrograph analysis (discussed further in the section "Automated Micrograph Analysis").

N-*Point Cross-Correlations for* N-*Point Statistics* N-Point statistics are useful when investigating spatial data containing multiple states, for example, a micrograph of a three-phase material. For such an image, the number of pixels of each state can be counted and placed in a normalized histogram to give the relative probability of finding any one state. This is called the 1-point statistics of the data (Figure 25).

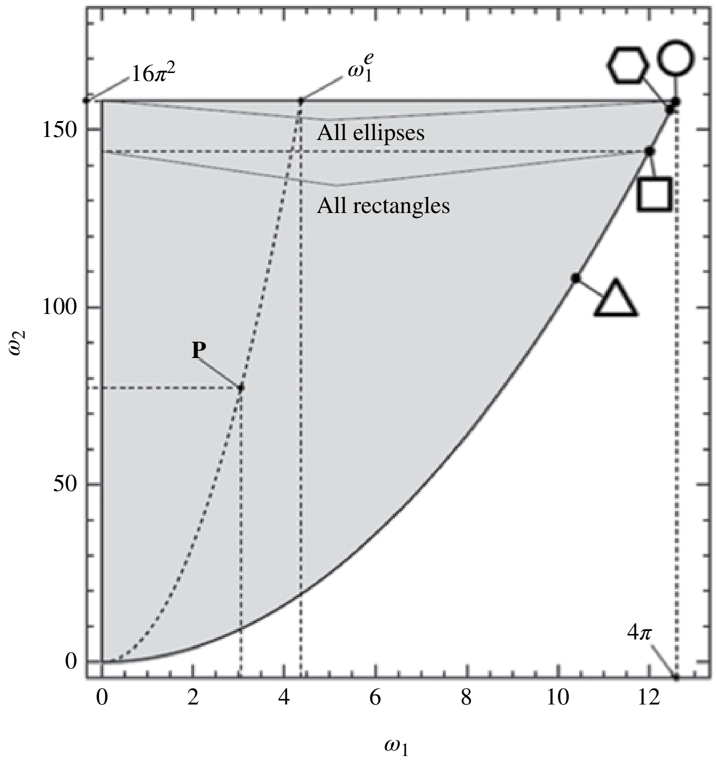


FIGURE 24 The moment invariant vector space for shape identification. Reprinted from Ref. 128, with permission from Elsevier.

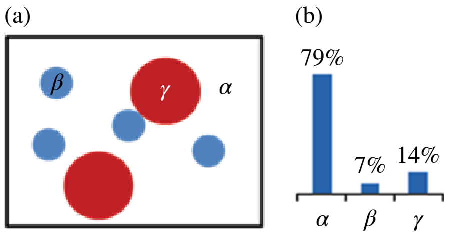


FIGURE 25 (a) Image containing three state regions α, β, γ. (b) 1-point statistics for state probability in (a).

A 2-point statistics image gives the probability of finding a pixel of state A separated from a pixel of state B (where B can equal A) by a vector of length R at an angle θ. Such statistics give information about the uniformity of an image. This is useful when characterizing material structural heterogeneity in a structure micrograph (Figure 26). This concept is further discussed in the section on "Automated Micrograph Analysis."

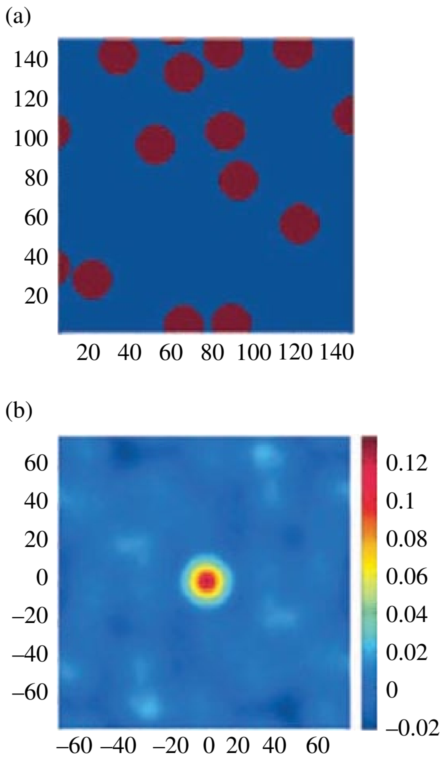


FIGURE 26 (a) An image with two states: dark gray on light gray background (red on blue background in color insert). (b) 2-point statistics for a vector beginning and ending in a dark gray (red in color insert) state. Reproduced from Ref. 169, with kind permission from Springer Science and Business Media. (*See insert for color representation of the figure.*)

The 2-point statistics are given by evaluating the 2-point cross-correlation. For an image containing N_p particles, each particle is identified with an indicator function as in Eq. [72]. The two-point cross-correlation for all pairs of particles is then given by

$$C(r) = \frac{1}{V}\sum_{i=1}^{N_p} D_i(r) * D_i(r) + \frac{1}{V}\sum_{i=1}^{N_p}\sum_{\substack{j=1 \\ i \neq j}}^{N_p} D_i(r - r_i) * D_j(r - r_j) \qquad [76]$$

where "*" represents a convolution and V is the volume of the region of interest, in this case the area of the image. The first sum contains the autocorrelation function for each particle, giving a nonzero value in every direction from $r = 0$ to $r = D_r$, where D_r is the diameter of the particle in the r direction. The second sum contains the cross-correlation between each particle and its neighbors, providing information about the overlap volume of two particles as a function of their relative position. Figure 26 shows an image with two states: dark gray and light gray (red and blue in the e-book version). Figure 26b shows the 2-point statistics of this image for a vector beginning and ending in a red state. The lack of symmetric peaks, concentric to the image center (similar to lattice diffraction), indicates a lack of periodicity in the red state. Also, the center of Figure 26b contains a 2D symmetric peak of radius D_r, which corresponds to the average radius of the red state domains.

SELECTED MATERIALS SCIENCE APPLICATIONS

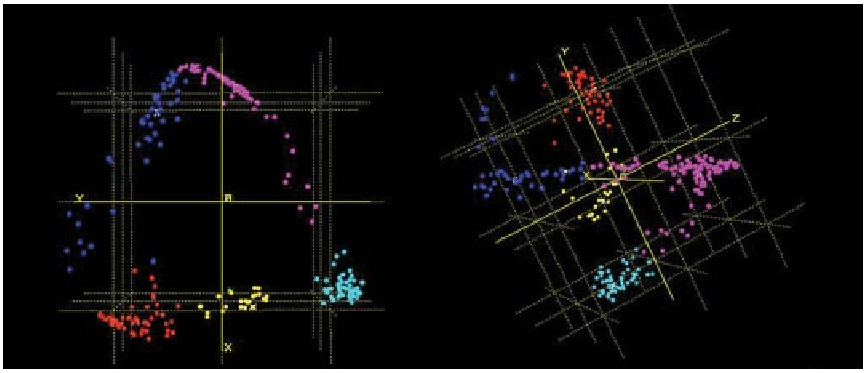
The tutorial in the prior sections is intended to provide the mathematical background needed to perform analysis of data and learning from data. Many of the techniques and algorithms described have already been utilized within the materials science community. In the following, we present an assortment of examples to illustrate how these methods are used, including some of those described earlier in greater detail.

Phase Diagram Determination

An exciting arena in which data mining methods are expected to play an important role is in the automated determination of phase diagrams using high-throughput combinatorial experiments.[133] Using thin-film compositional spreads, large fractions of phase diagrams can be mapped out with a high density of data points on a single wafer. We illustrate this concept using a recent assessment of the Fe–Ga–Pd ternary system.[3] Natural thin-film composition spreads of Fe–Ga–Pd were deposited at room temperature on oriented Si wafers, followed by postannealing and rapid characterization. The composition spread wafer contained over 500 individual $1.75 \times 1.75\,\text{mm}^2$ samples with continuously varying composition across the wafer. X-ray microdiffraction (XRD) was performed on each of these squares leading to a normalized intensity versus 2θ angle for each square.

Because each XRD spectrum can be viewed as a vector (in 2θ space), clustering analysis becomes a viable option to distinguish between spectra and to group spectra. The first step in this process is to select a suitable "distance" measure definition between any two given spectra (e.g., the Euclidean norm or 1−Pearson correlation coefficient), leading to a distance matrix **D** (Eq. [50]). If there exist N spectra, **D** is an $N \times N$ matrix, with each matrix element D_{ij} representing the "distance" between spectra i and j. The distance matrix may be used to cluster the N spectra, with each spectrum being represented by a point in the visualization space; however, any attempt to visualize the N points in a three-dimensional plot leads to a problem, because the visualization space is required to be $(N-1)$ dimensional. To better understand this, let us consider two spectra, or rather the "points" corresponding to those two spectra, S_1 and S_2. The first point may be placed in an arbitrary location, and the second point at a distance D_{12} from the first point. Consider now a third spectrum, represented by point S_3, and place this at a distance D_{13} with respect to the first point and D_{23} with respect to the second point. Repeating this process for successive spectra leads to the problem that all interpoint distances can no longer be maintained in three-dimensional space. We could try to place successive points in the best possible location, however, such that the chosen distances are the best approximations to the real

(a)



(b)

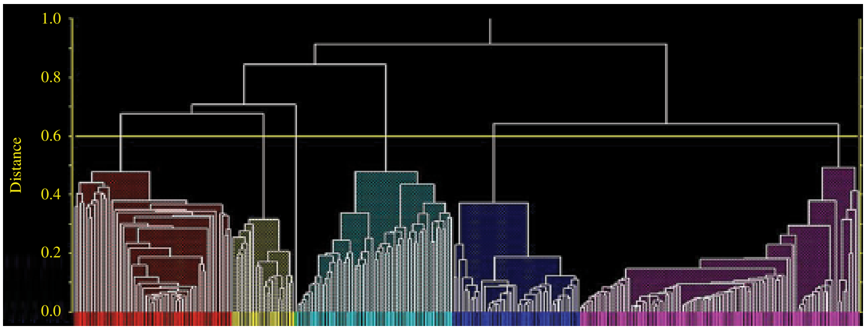


FIGURE 27 (a) Two different 3D views of an approximation of the distribution of points defined by the distance matrix, using metric multidimensional data scaling (MMDS). For this system, MMDS approximates well the distance matrix, accounting for about 93% of the information in the matrix. Each point in the MMDS plot corresponds to a spectrum. Points that are close together correspond to spectra that are similar to each other, as measured using the Pearson correlation coefficient. Color (e-book version) is used to identify groups, and groups are defined using the dendrogram. (b) A dendrogram based on the distance matrix. Points on the x-axis correspond to spectra. The height at which the lines leading from the spectra are connected indicates their similarity. The cut level is set at 0.6, creating six major groups. Reprinted with permission from Ref. 3. Copyright 2007, AIP Publishing LLC.

distances. Although this procedure discards some of the real distance or similarity information, it provides us with a way to visualize the distance matrix in regular three-dimensional space. A formal strategy to perform this is the metric multidimensional data scaling (MMDS) scheme described previously, the results of which are shown in Figure 27a for the example of the Fe–Ga–Pd ternary. Clusters of points are color coded (in e-book version) to make the grouping of spectra self-evident.

A more rigorous procedure to determine the groupings is to use a dendrogram as described in the section on "Hierarchical Cluster Analysis." An example of this is shown for the Fe–Ga–Pd system in Figure 27b. Each discrete point along the abscissa represents one of the spectra, and neighboring points are close in distance. Vertical lines from each spectrum connect via horizontal lines at some height. The height at which they connect is determined by the similarity between the spectra—the greater the height at which the connection occurs, the larger is the distance between the spectra. The groupings of the spectra then become self-evident. If we stop making groups at some threshold similarity level, we will be left with a small number of groups. In Figure 27b, the threshold chosen is indicated by the long horizontal line, leading to six distinct groups. The MMDS and the dendrogram schemes generally lead to similar groupings.

The six groupings derived in Figure 27b indicate the possibility of six different regions of similar structure. Identification of these regions can then be done by comparing the XRD spectra of the most representative member (i.e., the member having the smallest distance to all other members) of a group of spectra to a set of reference spectra from the NIST crystallographic databases. The result for the Fe–Ga–Pd system is shown in Figure 28. The regions and boundaries are clearly revealed in the phase diagram. This procedure is being adopted for the creation of phase diagrams in an automated high-throughput manner for several systems.[133]

Similarly, work is underway to automate the analysis of pure phases from these phase response diagrams and to identify the phase mixture ratios for each sample. These investigations include the use of high-speed methods[134] that as of yet still require verification by a crystallographer to ensure agreement with physics principles (e.g., Gibbs phase rule) and offline methods that seek to integrate such physics

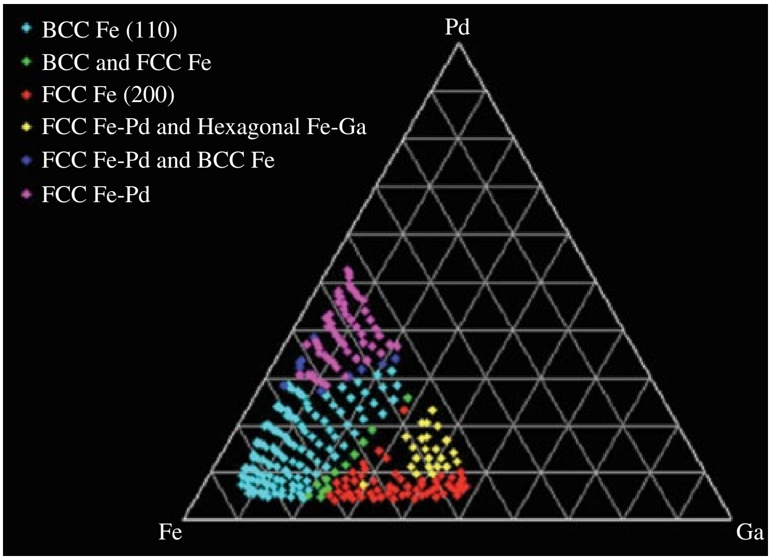


FIGURE 28 The distribution of diffraction patterns produced by using groups of spectra derived from clustering analysis and comparing the most representative patterns from those clusters with a database of reference patterns. Reprinted with permission from Ref. 3. Copyright 2007, AIP Publishing LLC (color available in e-book version).

principles into the automated analysis through the use of constraints and similarity preserving measures.[109,135]

The previous methods have been extended with new clustering algorithms to provide on-the-fly analysis of combinatorial library diffraction data.[12] Diffraction data is imported, processed, and analyzed in real time during data collection to provide phase response diagrams to the user. These methods can also utilize critically evaluated structure data from the Inorganic Crystal Structure Database to provide improved phase response diagram results. With such high-speed techniques, the user can analyze a combinatorial library rapidly, identify potential phase boundaries, and then perform a focused study on those boundaries of interest, reducing both analysis time and beam time.

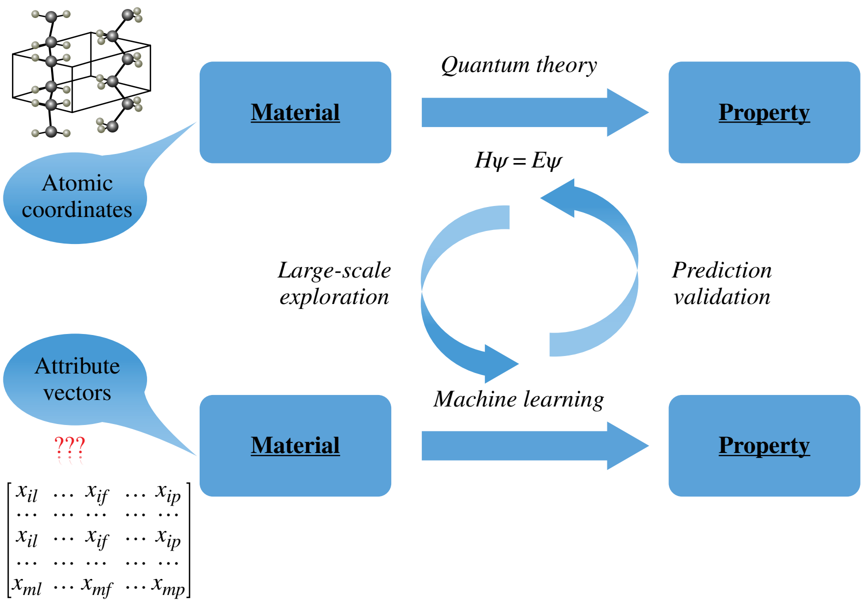
Materials Property Predictions Based on Data from Quantum Mechanical Computations

When confronted with a new material, it would be advantageous if its properties could be predicted using past knowledge pertaining to other similar known materials, rather than by resorting to new experiments or laborious calculations. As highlighted earlier, this paradigm may be used to predict crystal structures of new compounds knowing only the crystal structure information of other known compounds.

What if there is insufficient past information? If the goal is to derive an accurate determination of intrinsic materials properties (in addition to the crystal structure), one typically resorts to computations based on quantum mechanics. These are accurate, versatile, and nonsystem-specific methods but also time consuming. If quantum mechanical data is already available for a number of compounds within a subclass, can this information be used to predict properties of new systems within the same subclass? This is the question we answer in this section using several recent publications as examples.[6,7,136]

The basic idea of how to predict properties using quantum-derived data and machine learning is depicted in Figure 29. We reduce materials within a subclass to numerical fingerprints that represent the material uniquely. These are, of course, the feature vectors in the parlance of the machine learning community. Various machine learning methods may then be applied to map the fingerprints to properties. Once such a mapping is accomplished using a set of training data, new materials properties can then be computed efficiently without resorting to quantum mechanical calculations.

The biggest hurdle in creating such a prediction machine is the reliable definition of a material fingerprint. That fingerprint must be invariant with respect to translational and rotational operations, as well as to the ordering of atoms of the system, because the properties of the system are invariant with respect to such transformations (these issues will crop up again in a subsequent section on the development of interatomic potentials, where the demands of fingerprint choices are even more stringent). These issues were already recognized in the section on "Feature Extraction" and are particularly severe in systems containing multiple numbers and types of atoms. A good example of the need to satisfy fingerprint invariance is the work by



Atomic coordinates
Material
Quantum theory
$H\psi = E\psi$
Property
Large-scale exploration
Prediction validation
Attribute vectors
???
Material
Machine learning
Property
$\begin{bmatrix} x_{il} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{il} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{ml} & \cdots & x_{mf} & \cdots & x_{mp} \end{bmatrix}$

FIGURE 29 A perspective on the role machine learning can play in accelerating quantum mechanical computations.

Rupp et al.[136] The goal of that study was to predict accurate molecular atomization energies quickly. The machine learning model used was based on kernel ridge regression (KRR) (see the section on "The Kernel Trick") using Gaussian kernels. The model was trained and tested on >7000 organic molecules containing up to seven atoms (including H, C, N, O, and S). The molecular atomization energies for training and testing were computed using density functional theory (DFT).[137,138]

Recall that KRR relies on the notion that two materials with similar fingerprints have similar properties. Consider two systems I and J with fingerprints $\mathbf{x}_i$ and $\mathbf{x}_j$. The similarity of the two vectors may be measured in many ways, including the Euclidean norm of the difference between the two vectors, $\| \mathbf{x}_i - \mathbf{x}_j \|$, or the dot product of the two vectors $\mathbf{x}_i \cdot \mathbf{x}_j$. In this example we use the former and refer to it as $|\mathbf{x}_{ij}|$. Clearly, if $|\mathbf{x}_{ij}| = 0$, materials I and J are equivalent (insofar as we can conclude based on the fingerprints), and their property values y_i and y_j should thus be the same. When $|\mathbf{x}_{ij}| \uparrow 0$, materials I and J are inequivalent; $y_i - y_j$ is (usually) nonzero and depends on $|\mathbf{x}_{ij}|$, as prescribed by the KRR scheme.

The molecular fingerprint employed by Rupp et al. is based on the "Coulomb" matrix $\mathbf{M}$ defined as

$$M_{ij} = \begin{cases} 0.5Z_i^{2.4} & \text{for } i = j \\ \dfrac{Z_i Z_j}{|R_i - R_j|} & \text{for } i \neq j \end{cases} \qquad [77]$$

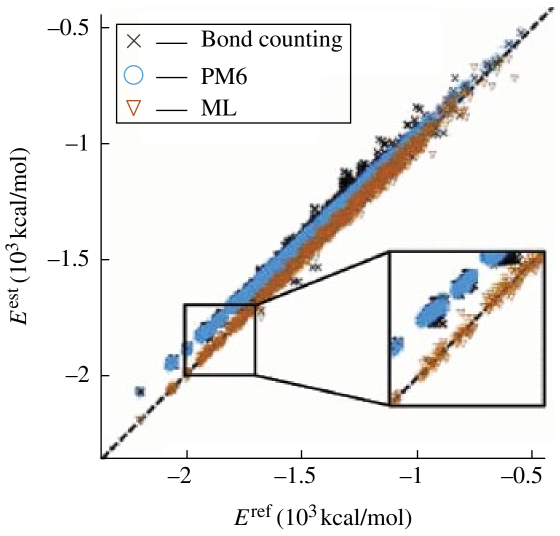


FIGURE 30 Correlation of the DFT results (E^{ref}) with the machine learning (ML) model estimates (E^{est}) of atomization energies for a data set exceeding 7000 organic molecules, along with a comparison of results from bond counting[139] and semiempirical quantum chemistry (PM6) calculations.[140] Reprinted with permission from Ref. 136. Copyright 2012 by the American Physical Society.

where Z_i and R_i are the atomic number and position of atom i, respectively. The off-diagonal elements correspond to the Coulomb repulsion between atoms i and j, while the diagonal elements are a polynomial fit of the atomic energies to nuclear charge. $\mathbf{M}$ is diagonalized, and the eigenvalues, λ_i, are ordered with decreasing absolute values. For matrices differing in dimensionality, the eigenvalue vector $\boldsymbol{\lambda}$ of the smaller system is expanded in size by adding zeros (the dimensionality of $\mathbf{M}$ for a particular molecule is equal to the number of atoms in the molecule). The $\boldsymbol{\lambda}$ vector constitutes the fingerprint of a molecule. It should be noted that this fingerprint is invariant to uniform translation, rotation, and atomic permutation of the molecule. The "distance" between two molecules is then measured by the Euclidean (ℓ^2) norm of the difference between two fingerprint vectors (see the section on "Regularized Least Squares"). The fingerprints, or equivalently the distances, were then used by Rupp et al. within a KRR method to establish a protocol for predicting molecular atomization energies.

Figure 30 displays the predictive capability of the machine learning (ML) scheme and contrasts it to the predictions of other methods. In subsequent work, Hansen et al. provide a deeper analysis of the KRR-based prediction scheme and explore the applicability of neural networks to accomplish such predictions.[7] Using such machine learning paradigms to predict properties other than the molecular atomization energies has also been explored recently.[7,15]

The ML protocol for molecular systems is significant, but it does not address critical issues associated with periodic solid state systems. More specifically, the Coulomb matrix-based fingerprint is not invariant with respect to translated unit cell definitions; moreover, several degrees of freedom are "lost" when using the eigenvalue vector of the Coulomb matrix, $\boldsymbol{\lambda}$, as the fingerprint (note that the total number of degrees of freedom of an N atom system is $3N-6$, but the dimension of $\boldsymbol{\lambda}$ is just N).

A recent development obviates these issues within the context of one-dimensional infinite polymeric chains.[6] For definiteness, let us assume that the building blocks of these chains are drawn from a pool of the following seven possible molecular fragments or building blocks: CH_2, SiF_2, $SiCl_2$, GeF_2, $GeCl_2$, SnF_2, and $SnCl_2$. Setting all the building blocks of a chain to be CH_2 leads to polyethylene (PE), a common, inexpensive polymeric insulator. The rationale in Ref. 6 for introducing the other group 14 halides was to interrogate the beneficial effects (if any) those blocks might have on various properties when included in a base polymer such as PE. The properties we focus here on include the atomization energy, the formation energy, the lattice constant, the spring constant, the band gap, the electron affinity, and the optical and static components of the dielectric constant.

The initial data set for 175 such polymeric chains containing four building blocks per repeat unit was generated using DFT. Each infinite 1D polymer system was then represented using a numerical fingerprint. One possibility for creating the fingerprint is to use the chemical and structural information associated with the building blocks. Because the polymer systems consist of seven possible building units, the fingerprint vector may be defined in terms of seven fractions, $|f_1, f_2, f_3, f_4, f_5, f_6, f_7\rangle$, where f_i is the fraction of building unit i, that is, the fragments CH_2, SiF_2, $SiCl_2$, GeF_2, $GeCl_2$, SnF_2, and $SnCl_2$. One can extend the components of the fingerprint vector to include clusters of two or three building units of the same type occurring together; such a fingerprint vector could be represented as $|f_1, \ldots, f_7; g_1, \ldots, g_7; h_1, \ldots, h_7\rangle$, where g_i and h_i are, respectively, the fraction of building unit pairs of type i and the fraction of building unit triplets of type i. We make the important observation that this definition of fingerprint takes care of rotational and permutation invariance requirements, as well as the translation and inversion invariance requirements encountered in infinite periodic systems.

Next, the DFT data is used to train a KRR-based machine learning model. Of the DFT-derived data set consisting of 175 polymers, 130 were used in the training set and the remainder in the test set to allow for validation of the machine learning model. Once the machine has learned how to map between the fingerprints and the properties using the training set, predictions were made and the model was validated. Furthermore, several systems with eight-block repeat units were considered (in addition to the 175 four-block systems). Results for all eight properties are shown in Figure 31. As can be seen, the level of agreement between the DFT and the machine learning schemes is uniformly good for all properties across the four-block training and test set, as well as the "out-of-sample" eight-block test set, indicative of the high-fidelity nature of this approach. The performance of the KRR machine learning scheme, even when dealing with situations that may be inherently highly nonlinear (e.g., the dependence of the band gap on chemistry), is not surprising because Gaussian kernels based on an underlying similarity measure were used.

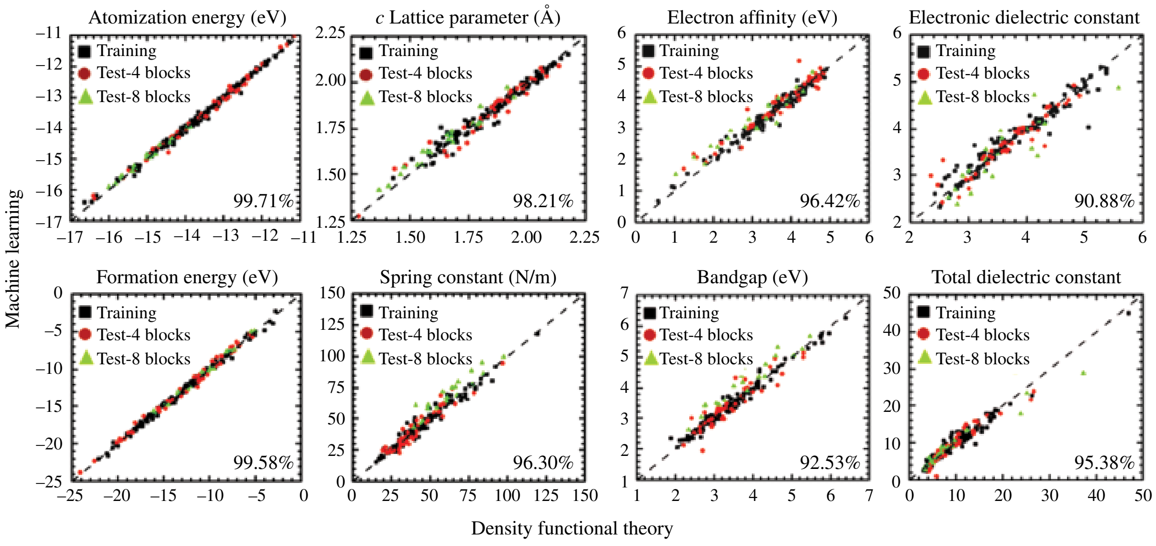


FIGURE 31 Performance of machine learning for a set of eight properties of 1D polymeric chains made from CH_2, SiF_2, $SiCl_2$, GeF_2, $GeCl_2$, SnF_2, and $SnCl_2$ building blocks. The training set was composed of systems containing 4-block repeat units, and the test set involved systems with both 4-block and 8-block repeat units. Pearson's correlation index is indicated in each of the panels to quantify the agreement between the two schemes.

More recent developments provide additional evidence that machine learning methods can effectively be used to perform accelerated property predictions.

Development of Interatomic Potentials

Many physical properties of materials are determined by the total potential energy: the equilibrium structure of a molecule or a crystal is the one having the lowest total energy; surfaces, interfaces, defects, and other (meta)stable structures correspond to local minima in the total energy hypersurface in phase space (also referred to as the potential energy surface or PES); the curvatures of the PES at energy minima are related to the vibrational and phonon frequencies, force constants, and elastic moduli; activation energies of chemical reactions, diffusion, and phase transitions are “passes” in the PES between local minima (“valleys”) corresponding to the reactants and products; and so on. Methods to determine the PES (or at least critical portions of the PES) are thus extremely useful.

Strategies for determining PESs efficiently and deriving properties from those PESs have been the focus of research over many years and across many disciplines. First principle methods based on quantum mechanics provide such a strategy, but they can be time-intensive, especially if the system contains more than a 1000 atoms (the length-scale challenge) or if the dynamics of even a small system has to be studied over a long period of time (the time-scale challenge). One practical solution to circumvent such spatiotemporal challenges associated with quantum methods is to use empirical and semiempirical interatomic potentials (sometimes called force fields) to characterize the PES. These methods predict the PES of a collection of atoms using predecided and preparameterized functional forms of the potential energy in terms of the atomic-level geometry. Because the Schrödinger equation is not solved, such methods come at a much lower computational cost compared to quantum mechanics-based methods. The downside is that they also come with either reduced accuracy or a narrower domain of applicability (i.e., parameterizations determined for a specific system in a specific environment will not, in general, be transferrable to new systems or new environments of the same system). Examples of such schemes include the Lennard-Jones potentials for noble gases, Stillinger–Weber, Tersoff, or Brenner and Garrison potentials for covalent systems, and the embedded atom method potentials for metals.[141]

The fidelity with which a given interatomic potential can predict properties depends on the choice of the functional form of the potential and the associated parameterization. While simplicity is desired, this also leads to inaccuracies and a narrow domain of applicability. Moreover, real interatomic interactions are intrinsically complex. A pathway to creating complex functional dependences of the energy with respect to the geometry of the system (i.e., atomic positions) may be provided by modern machine learning methods. This has, in fact, already been exploited effectively via artificial neural networks[8,9,142,143] with the energy function trained using DFT data. In other words, the interatomic potentials constructed from machine learning methods are trained to reproduce DFT results (of energies, forces, and dynamics), but those functions reproduce the PES at a small fraction of the cost of DFT calculations.

Neural networks were discussed earlier in the section on "Supervised Learning Algorithms" and will not be discussed in detail here. It suffices to say that Figure 32 is a schematic of a typical feed-forward neural network that has been used in the past to map the coordinates of a set of N atoms, $\mathbf{R}_i$, $i = 1 - N$, to the total energy, E_s, of the system. Again, a critical ingredient that enters this scheme is the fingerprint vector. Because the demands on the interatomic potentials for predicting energies (and especially forces) are high, particular care must be taken in the choice of the fingerprints. In keeping with the philosophy that interatomic potentials take the atom type and position information as the starting point in the process of providing energies and forces, it is tempting to use the Cartesian coordinates of the atoms and the atom types as the fingerprints. However, because atomic Cartesian coordinates are not invariant with respect to a rotation or translation of the system, such coordinates cannot be directly used in this mapping process. Thus, a transformation of the Cartesian coordinates to a more appropriate set of coordinates is required so that the chemical environment of a given atom can be properly captured. "Symmetry functions" offer one such set of coordinates. An example of a radial symmetry function is

$$G_i = \sum_{j \neq i} e^{-\eta\left(R_{ij} - R_\mathrm{s}\right)^2} f_\mathrm{c}\left(R_{ij}\right) \tag{78}$$

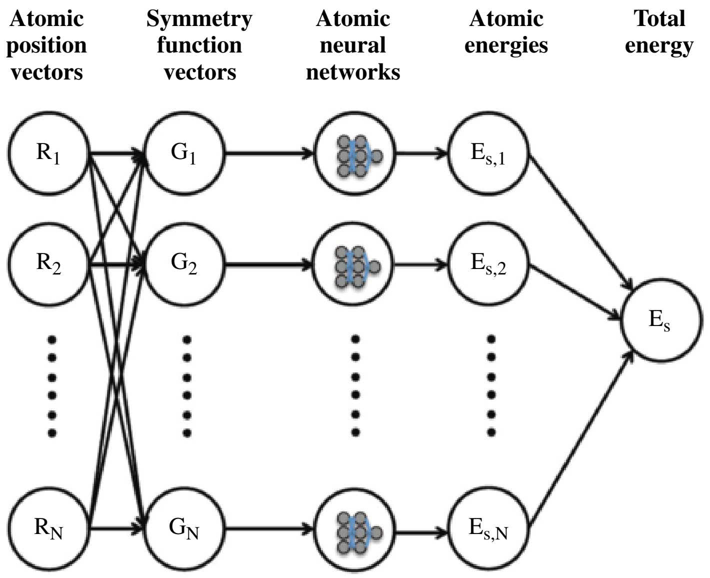


FIGURE 32 Structure of a high-dimensional neural network (NN) potential based on an expansion of the total energy as a sum of atomic energy contributions.

where η and R_s are parameters that define the Gaussians, R_{ij} is the distance between atoms i and j, and f_c is a cutoff function defined as

$$f_c\left(R_{ij}\right) = \begin{cases} 0.5 \times \left[\cos\left(\frac{\pi R_{ij}}{R_s}\right) + 1\right] & \text{for } R_{ij} \leq R_c \\ 0 & \text{for } R_{ij} > R_c \end{cases} \qquad [79]$$

Likewise, angular symmetry functions can be defined. Symmetry functions are essentially a set of functions of the atomic positions, allowing for a transformation of the $\mathbf{R}_i$ vectors to the $\mathbf{G}_i$ vectors. Because the $\mathbf{G}_i$ vectors depend only on the interatomic distances and angles, they are invariant to translations and rotations in molecular as well as in periodic solid state systems. For a given spatial distribution of atoms, the $\mathbf{G}_i$ vector components, corresponding to various choices of the parameters η and R_s, represent (the environment of) atom i. Two atoms residing in exactly the same chemical environment will have exactly the same symmetry function representations; moreover, the set of $\mathbf{G}_i$ vectors will be identical for all energetically equivalent representations of the system.

An additional appeal of the neural network configuration of Figure 32 is that this scheme allows for the partitioning of the total energy into atomic energies, as a mapping between each of the $\mathbf{G}_i$ vectors (corresponding to an atom i) is established with the corresponding atomic energy $E_{s,i}$ (this neural network is hence called an "atomic" neural network). The sum of all the $E_{s,i}$s is of course the total energy, E_s, of the system. Once trained on a set of atomic configurations, the atomic neural network is capable of providing $E_{s,i}$ when the $\mathbf{G}_i$ of an atom in a new configuration is supplied (regardless of the number of atoms in the new configuration). The neural network-based interatomic potential definition is thus complete.

Figure 33 compares the radial distribution function (RDF) of a silicon melt at 3000 K for a cubic 64-atom cell obtained using MD simulations based on DFT, several interatomic potentials (including the Bazant, Lenosky, and Tersoff potentials), and the neural network potential.[143] It can be seen that the RDF obtained from a neural network potential-based MD simulation best matches the DFT result, while there are significant deviations for the other empirical potentials, attesting to the versatility, fidelity, and efficiency of the neural network-based potentials. Such NN-based schemes have been used in several applications to date ranging from molecules, solids, and surfaces to elemental and compound systems (the latter involves enhancements to deal with electrostatic interactions between atoms of unlike types).[9]

Finally, we also note that paralleling the developments based on neural networks is an alternate strategy based on KRR using Gaussian kernels as the learning scheme.[12] These potentials, referred to as Gaussian approximation potentials, or GAP, also lead to a decomposition of the total energy into atomic energies and a mapping of atomic environments (written in terms of a bispectrum, a 3-point correlation function) to atomic energies. Both the neural network and KRR-based schemes are capable of providing analytical forces and hence allow for efficient MD

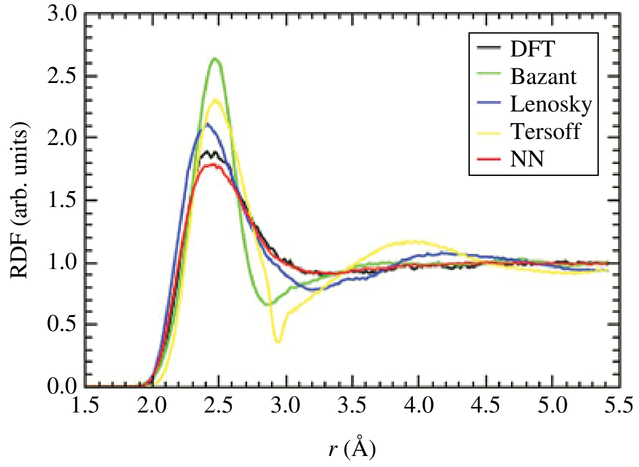


FIGURE 33 Radial distribution function of a silicon melt at 3000 K obtained from MD simulations based on DFT, empirical interatomic potentials (including those by Bazant, Lenosky and Tersoff), and NN-based potentials. Reprinted with permission from Ref. 143. Copyright 2007 by the American Physical Society. (*See insert for color representation of the figure.*)

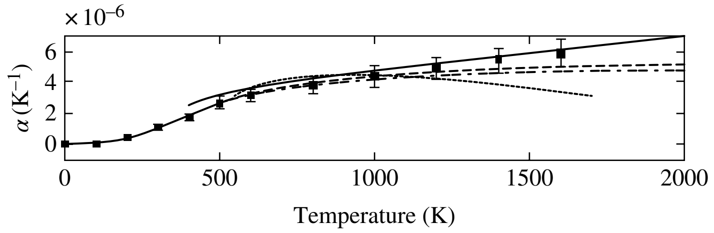


FIGURE 34 Linear thermal expansion coefficient of diamond in the GAP model (dashed line) and DFT (dash-dotted line) using the quasiharmonic approximation and derived from MD (216 atoms, 40 ps) with GAP (solid) and the Brenner potential (dotted). Experimental results are shown with squares. Reprinted with permission from Ref. 12. Copyright 2010 by the American Physical Society.

simulations. The GAP scheme competes with the neural network-based approach in terms of speed, versatility, and fidelity but has the attractive feature that its learning model (unlike neural networks) is more physically motivated being based on the concept of similarity. Figure 34 shows an example of the prediction of the coefficient of thermal expansion of diamond using DFT and GAP potentials within the quasiharmonic approximation and by MD simulations utilizing the Brenner and GAP potentials.[12] Comparing the predicted results with experiment, it is evident that the quasiharmonic approximation is inadequate at high temperatures (although GAP potentials reproduce DFT results) and that MD simulations utilizing the GAP potentials outperform all other schemes. Details of the GAP potential generation method may be found elsewhere.[12]

A more recent development utilizes the capability to predict atomic forces directly given just the atomic configuration [Refs. 1 and 2]. Such a capability may then be used within large-scale geometry optimizations and molecular dynamics simulations. The basic premise of such a scheme is that the behavior of an atom in a molecule, liquid or solid is governed by the force it experiences. Within this development, a recipe has been proposed to numerically represent atomic environments, which can be mapped to vectorial quantities such as the atomic force using machine learning methods. What results then is a force field. Several examples are provided as to how such a force field for Al can be used to go far beyond the length-scale and time-scale regimes presently accessible using quantum-mechanical methods. It is argued that pathways are available to systematically and continuously improve the predictive capability of such a learned force field in an adaptive manner, and that this concept can be generalized to include multiple elements. Strategies are also available to develop, and use, such a force field on-the-fly during the course of a DFT based molecular dynamics simulation, thus leading to hybrid DFT/ML molecular dynamics simulations.

Crystal Structure Predictions (CSPs)

One of the greatest challenges in computational materials design is to predict the crystal structure of a material that has not yet been synthesized.[144,145] A straightforward approach to crystal structure prediction is to search the potential energy surface of various atomic arrangements in an attempt to identify the atomic arrangement (crystal structure) with the minimum energy. Such an approach can make use of machine learning algorithms to predict the potential energy surface, as discussed in the section on "Development of Interatomic Potentials." However it is also possible to directly use machine learning to predict crystal structure, by mining a database of known crystal structures to determine the probability that a material with a given composition will have a given *structure type* (e.g., spinel, fcc, etc.). Using this latter approach, Fischer et al. developed a method for structure prediction called the "Data Mining Structure Predictor" (DMSP).[146]

In the DMSP approach, the probability distribution over possible ground states for a given chemical space (e.g., Al–Ti structures) is calculated using a truncated "cumulant expansion":

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{i} p(x_i) \prod_{j,k} \frac{p(x_j, x_k)}{p(x_j)\, p(x_k)} \qquad [80]$$

where $\mathbf{x}$ is the set of ground state structure types at different compositions in the chemical space, $p(x_i)$ is the probability distribution over possible values for the ith element of $\mathbf{x}$, and $p(x_j, x_k)$ is the joint probability distribution over possible values for the jth and kth elements of $\mathbf{x}$. The probability distributions $p(x_i)$ and $p(x_j, x_k)$ were calculated using a Bayesian analysis of structures in the Inorganic Crystal Structure Database (ICSD),[147] which contains over 100,000 known structures.

The probabilities calculated using DMSP were used by Fischer et al. to generate a ranked list of the structure types that are most likely to occur in different chemical spaces. They demonstrated that 90% of the time, the experimentally observed structure type was in the top five structures identified by the DMSP ranking. This compares favorably to just choosing the structure types that occur most frequently, which have only a 62% success rate for the top five structures.

Hautier et al. extended the DMSP approach to ternary oxides, using it to predict likely new ternary oxide compounds.[4] After further filtering the list of likely compounds using DFT calculations, they identified 355 ternary oxides that are likely to exist but were not in the ICSD. To test these predictions they searched for the predicted structures in the PDF4+ database, a database of known diffraction data that had not been used to parameterize the DMSP model. Of these 355 oxides, the PDF4+ database contained structural information for 146 (the remaining structures might not exist or may not have been discovered yet). The known diffraction data matched the predicted structure types in 140 of the 146 cases, suggesting that their method has a high success rate in predicting the structures of new ternary oxides.

The drawback to the DMSP approach is that it is limited to known structure types for which there is sufficient training data available, and it will never be able to predict the existence of a structure type that has never been seen before. Thus it would be difficult to effectively extend this method to the more complex space of materials containing four or more components.

Developing and Discovering Density Functionals

We now consider the question as to whether the intrinsic accuracy of DFT computations can themselves be improved via machine learning. Among all quantum mechanics-based computational methods aimed at materials property predictions, DFT presently offers the best trade-off between computational cost and accuracy. Several approximations are made to render the solution of the DFT Kohn–Sham equations practical, the most critical involving the electron–electron exchange–correlation interaction. While the Hohenberg–Kohn theorems of DFT demonstrate the existence of a *universal* exchange–correlation functional of the electron density, the explicit functional form is unknown. The fidelity of DFT results depends on this approximation. In important cases (such as strongly correlated systems and systems bound by weak van der Waals interactions), the currently used approximations—such as the local density approximation (LDA), the generalized gradient approximation (GGA), or even more advanced hybrid- and meta-GGA functionals—often fail. Consequently, there is a never-ending search for density functionals (especially better approximations to them).[148]

Throughout this chapter we have seen that machine learning is a powerful tool for finding patterns in high-dimensional data, especially patterns that defy and evade human intuition. Machine learning models can be trained on known good examples to detect a pattern in a new situation or make predictions related to a new situation. Within the context of the present discussion, if solutions are known for a subclass of problems that connect the electron density to a set of properties (e.g., the exchange–correlation energy, electronic kinetic, or total energy), this knowledge can be used to train an machine learning scheme to make predictions of those properties for new instances of the electron density (corresponding to new physical situations).

In recent work, Snyder and coworkers consider a prototypical problem involving N noninteracting spinless fermions confined to a one-dimensional box, $0<x<1$, with hard walls.[11] For continuous potentials $v(x)$, the Schrödinger equation can be solved numerically, the lowest N orbitals occupied, and the kinetic energy T and the electron density $n(x)$ may be computed (the latter is the sum of the squares of the occupied orbitals). For a class of potentials $v(x)$, this procedure provides examples of the relationships between the electron density $n(x)$ and the kinetic energy T, represented here as $T[n]$ (where square brackets represent a functional). The aim of the machine learning approach is then to construct $T[n]$ that first completely bypasses the quantum mechanical procedure (much like the examples provided in the section on "Materials Property Predictions Based on Data from Quantum Mechanical Computations"; cf. Figure 29) and second provides a kinetic energy functional.

In this regard, Snyder et al. consider up to four electrons in a one-dimensional box specified by potentials of the form

$$v(x) = -\sum_{i=1}^{3} a_i e^{\frac{-(x-b_i)^2}{2c_i^2}} \qquad [81]$$

They generated 2000 such potentials, randomly sampling $1<a_i<10$, $0.4<b_i<0.6$, and $0.03<c_i<0.1$. For each potential, numerical solution of the Schrödinger equation yields T and $n(x)$. The $T[n]$ mapping was then established using KRR, using 1000 densities in the test set, M others for training, and 10-fold cross-validation. The authors show that for $M>80$, the mean average error of the predicted kinetic energy is under 1 kcal/mol, that is, "chemical accuracy" has been achieved.

The previous demonstration is gratifying as it heralds a pathway for determining more accurate density functionals. But challenges remain. The authors show that a straightforward application of their procedure does not guarantee that derivatives of the relevant functionals, which are encountered in orbital-free DFT implementations, are always predicted properly. They show this problem can be overcome by using principal components to eliminate arbitrarily close electron densities, which tend to add noise in the training data set. A further point to note is that the predictive capability or domain of machine learning schemes depends on the example (or training) set used. Thus, applying this scheme to molecules and solids will require more realistic and diverse electron densities and properties.

Lattice Models

Many problems in materials science can be expressed in terms of periodic arrays of sites where each site can exist in a variety of different states exemplified, for example, by magnetic ordering where "spin-up" or "spin-down" states are assigned to sites on a lattice (Figure 35a). If we let the variable s_i indicate the state of the ith site, the set of all such variables, denoted by **s**, gives the overall state of the material. A property of the material can then be expressed as a function, $f(\mathbf{s})$. Such functions are known as *lattice models*.

One of the most widely studied lattice models is the Ising model,[149] in which the site states represent the electronic spin at a given site, with "up" spin indicated by $s_i = 1$ and "down" spin indicated by $s_i = -1$. If the interactions between two spin moments are assumed to extend no farther than the nearest neighbor and all sites are symmetrically equivalent, the magnetic energy of the system may be expressed as

$$f(\mathbf{s}) = V_0 + \sum_i V_1 s_i + \sum_{(i,j) \in nn} V_2 s_i s_j \qquad [82]$$

where V_0, V_1, and V_2 are coefficients and "nn" is the set of all pairs of nearest-neighbor sites.

In *cluster expansion* models,[150] the Ising model approach is extended in the following ways (Figure 35):

1. The site variables s_i may represent any state of the site, not just spin. In practice they are commonly used to represent *site occupancy* in materials with substitutional disorder. For example, in a Cu–Ni alloy, $s_i = 1$ would indicate that Cu is present at the ith site, and $s_i = -1$ would indicate the presence of Ni.

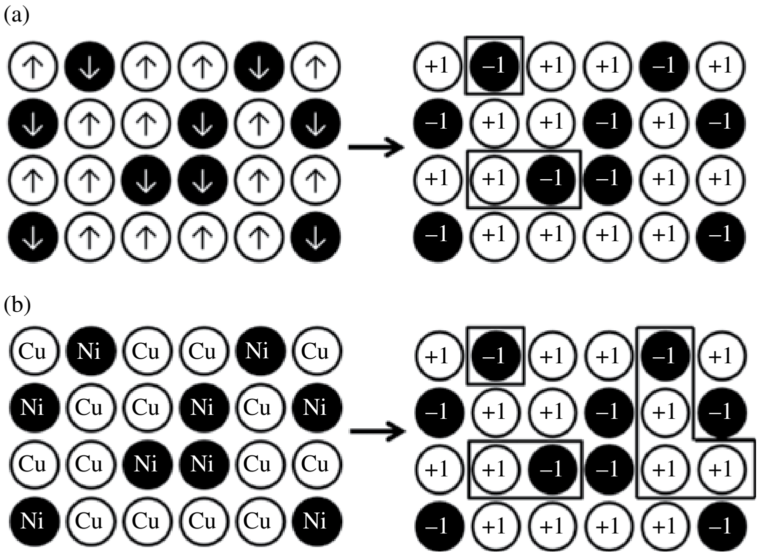


FIGURE 35 Differences between (a) Ising model and (b) a typical cluster expansion. Cluster expansions are applied to a variety of different types of problems but most commonly used to study atomic order. An Ising model includes contributions from individual sites and nearby pairs of sites (outlined in boxes on the right), whereas the cluster expansion can include contributions from larger and more complex clusters of sites.

2. Interactions beyond nearest-neighbor pairs are considered. This includes 3-body, 4-body, etc., interactions, as well as interactions between sites that may be very far apart. Ducastelle et al. showed that if interactions between all possible clusters of sites are considered, then the following expansion is exact for a binary alloy:[150]

$$f(\mathbf{s}) = V_0 + \sum_{\alpha} V_\alpha \prod_{i \in \alpha} s_i \qquad [83]$$

where α represents a cluster of sites, the sum is over all possible clusters of sites, and the product is over all sites in the cluster. V_0 is a constant term, and V_α is the coefficient for cluster α. The number of distinct coefficients in the expansion may be reduced by acknowledging that values of all coefficients for symmetrically equivalent clusters must be the same. The challenge is to determine the values for these unknown coefficients, referred to as *effective cluster interactions* (ECI). Although Eq. [83] is written for binary systems, a similar result can be obtained for higher-order systems.

Cluster expansions are commonly used to study systems exhibiting substitutional disorder, where they are usually parameterized using a training set of DFT calculations. They are typically capable of rapidly calculating the energies of millions of different arrangements of atoms within 10 meV per atom of DFT. This combination of speed and accuracy makes them a useful tool for thermodynamic averaging (e.g., to generate alloy phase diagrams) or searching for structures with optimal property values (e.g., ground state structures). The cost of generating a cluster expansion is primarily the cost of generating the training data used to parameterize the expansion. Hence there is great interest in finding ways to generate accurate cluster expansions with minimal training set size.

To determine the unknown coefficients in a cluster expansion, it is necessary to constrain and/or weight the hypothesis space. Historically, this has been done through *cluster selection*, in which it is assumed that clusters that contain a large number of sites and clusters that contain sites that are very far apart are likely to have small ECI. Accordingly, those ECI can be set to zero with little loss of accuracy. The remaining ECI are typically fit to DFT calculations using a least-squares fit.

To facilitate the cluster selection process, van de Walle and Ceder introduced the concept of cross-validation into cluster expansions.[151] They proposed that the set of clusters to be included in the expansion should be those for which the cross-validation error was lowest. Other groups pursued similar efforts to improve the quality of cluster expansions. For example, Drautz and Diaz-Ortiz proposed introducing a regularization term into the objective function used to determine the ECI, in the form of

$$\sum_{\alpha} {w_\alpha}^2 {V_\alpha}^2 \qquad [84]$$

where the sum is over all clusters in the expansion and w_α is a weight assigned to each set of symmetrically equivalent clusters.[152] The weights could be determined by minimizing the cross-validation score. Similarly, Zunger and coworkers developed a "mixed-basis" cluster expansion, in which the following regularization term was applied to pair clusters:[153,154]

$$\lambda_1 \sum_{\alpha} r^{\lambda_2} V_{\alpha}{}^{2} \qquad [85]$$

where r is the distance between sites in the pair and λ_1 and λ_2 are adjustable parameters that may be determined by cross-validation.

Mueller and Ceder demonstrated how all of the aforementioned approaches, and others, could be interpreted in a Bayesian framework, in which the regularization term was the negative log of a prior probability distribution placed over the ECI.[155] In the Bayesian interpretation of the cluster selection approach, the prior probability distributions for clusters excluded from the fit are effectively delta functions centered at zero. For clusters that are included in the fit, an improper uniform prior is implicitly used.

Building off of the Bayesian interpretation, Mueller and Ceder demonstrated that cluster expansions with reliably low prediction error could be generated by explicitly incorporating physical insights into the fitting process. For example, the expected magnitudes of the ECI for pair clusters can be expected to decrease with respect to distance, resulting in a regularization term similar to the one in Eq. [85]. Similarly, the expected magnitudes of the ECI can be expected to decrease as the number of sites in the cluster increases.

The earlier insights can be incorporated into prior distributions for expected ECI values, in which adjustable parameters determine how expected ECI magnitudes decrease as cluster size increases. An example of such prior distributions, and a comparison to the cluster selection approach, can be seen in Figure 36. The parameters in the prior probability distributions can be adjusted to minimize the cross-validation score. It was demonstrated that prior distributions with fewer adjustable parameters tend to produce more accurate cluster expansions than those with many adjustable parameters. In other words, the optimization of the cluster expansion using cross-validation is less likely to overfit the data.

By using a multivariable Gaussian distribution (Eq. [26]) as the prior distribution for the ECI values, the most likely ECI values can be rapidly determined using Eq. [27].[155] The advantage to this approach is that it is in practice only slightly more expensive computationally than a simple least-squares fit, and it allows the inclusion of more distinct ECI in the cluster expansion than there are structures in the training set (something that would lead to an ill-posed problem if a simple least-squares fit were used). It was demonstrated that the off-diagonal terms in the regularization matrix Λ (Eq. [27]) could be used to introduce the concept of *similarity* into cluster expansions.[155] For example, a nearest-neighbor pair three layers below the surface of a material is not symmetrically equivalent to a nearest-neighbor pair four layers below the surface, so the corresponding ECI should not be identical. However it can be expected that they will be similar, and accounting for this expected similarity in the prior distribution can improve the accuracy of cluster expansions for surfaces and nanoparticles significantly. Mueller extended this approach to allow for composition-dependent ECI in a cluster expansion of atomic order in 2 nm Au–Pd nanoparticles,

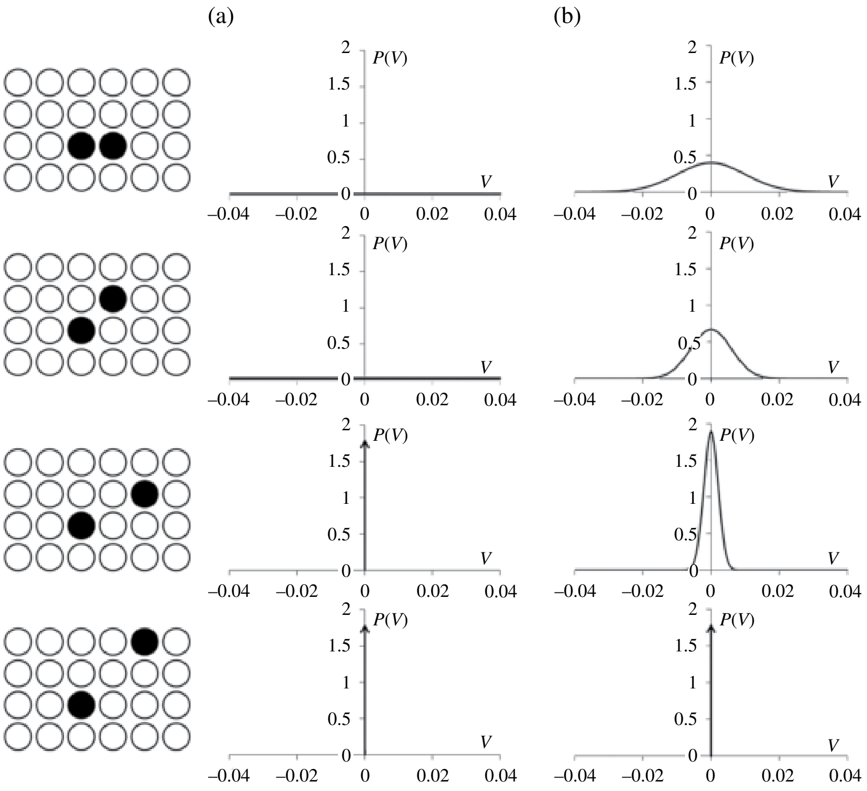


FIGURE 36 Representative prior probability distributions, $P(V)$, for the ECI, V, for pair clusters. In (a) the cluster selection approach is used, with only clusters up to the third nearest neighbor included in the cluster expansion. In (b) a Bayesian approach is used that gradually reduces the width of the prior probability distribution as the distance between sites increases. The sites in the pair clusters are indicated in the left column. The vertical arrows indicate delta functions, and the thick horizontal lines indicate improper uniform priors.

resulting in a cluster expansion capable of evaluating the energies of millions of atomic arrangements per minute with an estimated prediction error of only 1 meV per atom relative to DFT.[156]

A Bayesian approach that uses Laplace prior distribution over ECI values, resulting in ℓ^1 regularization of the ECI, has been demonstrated by Nelson et al.[157] This approach is commonly referred to as "compressive sensing," due to its origins in signal processing. The ℓ^1 norm favors sparse solutions, in which the values of many ECI are set to identically zero. Whether there is any advantages to using an ℓ^1 norm in the context of cluster expansions is not clear,[158] but it is reasonable to expect that this approach should work well for cluster expansions in which there are relatively few ECI that significantly differ from zero. This approach has recently been extended to include the use of hyperpriors on the parameters that define the Laplace distributions.[159]

There have also been a variety of methods proposed to incorporate active learning into the generation of cluster expansions. Zunger et al. proposed building a training set using "quasirandom structures" in which the average interactions among sites in small clusters resemble that of a random alloy.[160] Van de Walle and Ceder developed an active learning approach to determine which structures should be included in the training set of DFT calculations, based on the idea that structures should be chosen in a way that maximizes the expected variance reduction per CPU hour spent generating the training data.[151] Their method is closely related to the A-optimality approach described in the section entitled "The Training Data." Mueller and Ceder have built upon this work to demonstrate that in many cases, it is possible to derive exact expressions for the expected variance reduction in cluster expansions.[161] In parallel, Seko and coworkers have proposed alternative approaches to select structures to be included in cluster expansion training sets.[162,163]

Materials Processing and Complex Materials Behavior

Machine learning methods have enjoyed long-term success in the general areas of automated manufacturing,[164] process control,[165] and in the prediction of materials behavior under complex conditions.[17,166–168] Within the context of manufacturing, precision machining of parts is a critical step. Neural networks, which can effectively map complex nonlinear input/output relationships under these circumstances, have been used extensively. Quantities of interest that are routinely predicted include surface roughness and material removal rate (the outcome, or goals, of the machining process) as a function of a variety of machining conditions. Exploiting learning models such as neural networks thus significantly alleviates the burden on actual time-consuming repetitive experiments, not to mention wastage of parts that do not meet a target requirement.

Neural networks have also been enormously successful in understanding complex materials behavior, such as mechanical behavior (flow stress, hardness, tensile strength, fracture strength, and fatigue behavior) of metal alloys subjected to certain heat treatment and/or deformation procedures, as well as in the prediction of microstructures and phases resulting from heat treatment and/or deformation processes. The most practical way to capture the complex dependence of a desired macroscopic property on the various process parameters is through such learning methods. Figure 37 shows an example of the application of backpropagation neural networks to predict the flow stress of 42CrMo steel (which was subjected to deformation).[164] The inputs to the neural network are the process parameters: deformation temperature, log strain rate, and strain.

More recently, attempts have been made to predict complex, but intrinsic, materials behavior using fundamental atomic-level attributes of the material as the starting point. The friction coefficient is one such complex materials property. In recent work,[78] the tribological properties of 38 ceramic materials were determined using a variety of experimental methods. Sixteen materials properties (including hardness, cation charge, percent ionicity, Madelung constant, melting temperature, density, etc.) that could potentially control the friction coefficient were considered.

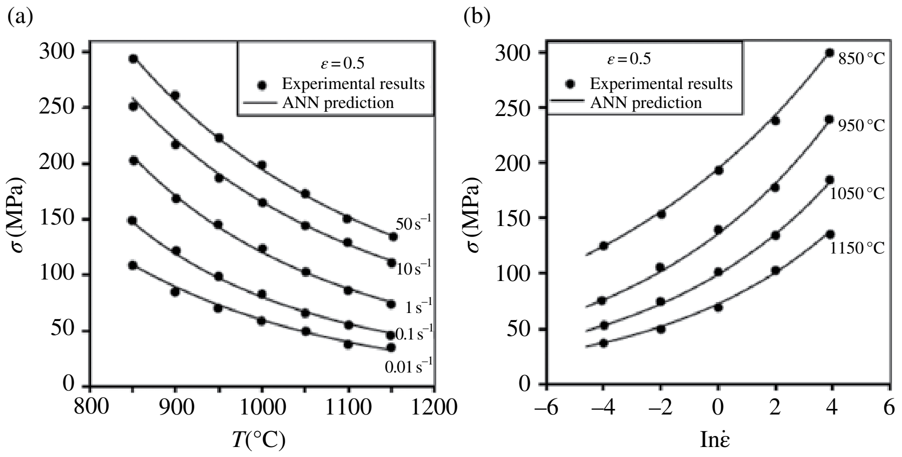


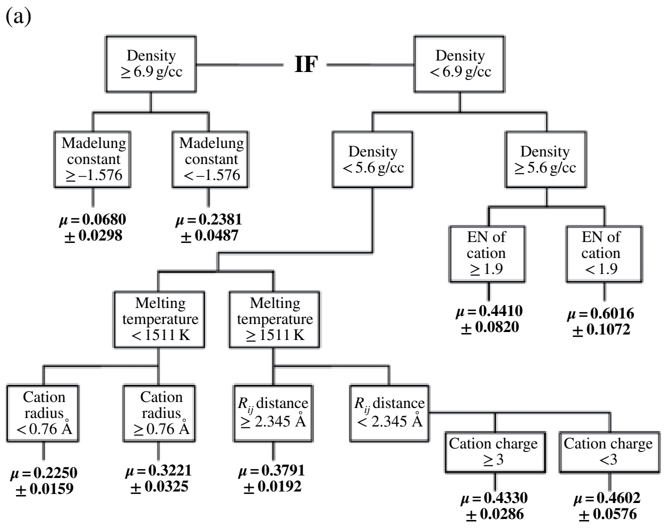
FIGURE 37 Comparison between the experimental and predicted flow stress of 42CrMo steel: (a) effects of the deformation temperature, (b) effects of the deformation strain rates. The predictions were made using a backpropagation artificial neural network (ANN) with the deformation temperature, strain rate, and strain as input. Reprinted from Ref. 164, with permission from Elsevier.

Feature reduction methods (as described in the section on "Latent Variable Analysis") were used to identify the most critical of these properties, which were identified to be as follows: density, cation electronegativity, melting temperature, cation–anion distance (R_{ij}), Madelung constant, and cation radius. Based on these identified features, a dendrogram (or decision tree, as described previously) was created to aid in the prediction of the friction coefficient. Figure 38 shows both the decision tree and a comparison of the predicted versus experimental values of the friction coefficient.

Automated Micrograph Analysis

An existing bottleneck preventing rapid advanced material discovery is the process of turning 2D sectional micrographs into actionable knowledge about material structure. Automation of this process will expedite our understanding of the relationships between (i) material processing history, (ii) microstructure, and (iii) the resultant functional properties. Optimally, the automation process would convert either single micrographs or a series of sectional micrographs into a microstructure representation that could be rapidly stored, searched, and mined for feature similarity. Furthermore, the representation should take into account the statistical nature of the microstructure and provide a means for identifying a representative volume element.

There are currently two dominant methods for automated microstructure identification. The first involves quantifying the spatial arrangement of microstructure "building blocks" such as grains, and the second involves quantifying the distribution of their shapes. As an initial step for both methods, the micrograph is first converted



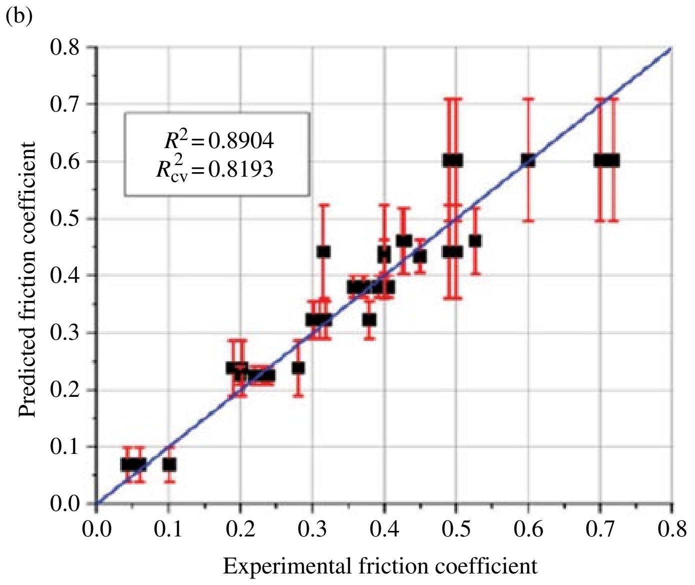


FIGURE 38 (a) Decision tree (or dendrogram) for the prediction of friction coefficients of materials based on six fundamental material level or atomic-level features. (b) Comparison of the predicted versus experimental friction coefficients. Reprinted from Ref. 78 with kind permission from Springer Science and Business Media.

into a collection of grain regions identified by structural properties such as texture and crystal orientation by using feature extraction techniques (discussed in the section "Feature Extraction") including edge/blob detection and neural network and graphical model-based segmentation.[169–173]

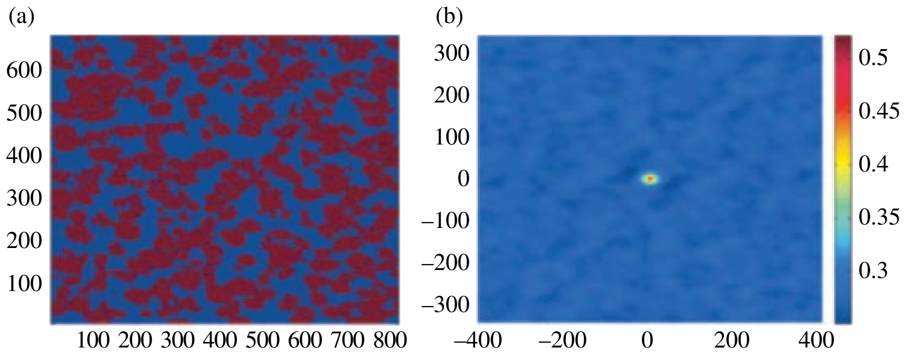


FIGURE 39 (a) Electron backscattered diffraction (EBSD) of a α–β Ti alloy with bimodal microstructure with primary α particles (dark gray; red in color insert) and secondary α particles (light gray; blue in color insert). (b) 2-point statistics for the EBSD micrograph shows that the primary α particles are spatially uncorrelated and have a 0.51 volume fraction. Reproduced from Ref. 169 with kind permission from Springer Science and Business Media. (*See insert for color representation of the figure.*)

Microstructure spatial arrangement can be rapidly computed through the use of *N*-point statistics (described in the section "*N*-Point Cross-Correlations for *N*-Point Statistics"), with accelerated computation through the use of fast Fourier transforms.[174] An example is shown in Figure 39a.[169] Here an α–β Ti alloy with bimodal microstructure is shown using electron backscattered diffraction. The image was preprocessed using a combination of the generalized Hough transform and image segmentation techniques to identify primary α particles (dark gray; red in e-book version) and secondary α particles (light gray; blue in e-book version). Using 2-point statistics (Figure 39b) shows that the primary α particles are spatially uncorrelated and have a volume fraction of 0.51, given by the proportion of the figure area with a 2-point statistic value greater than approximately 0.3.

The *N*-point statistics representation can be reduced to a lower-dimensional representation by multidimensional scaling and latent variable analysis techniques such as principal component analysis. Using the first two or three principal components, the *N*-point statistics can then be visualized in a 2D or 3D space where processing–structure–properties relationships are more easily visualized and understood.[175] An example is drawn from Ref. 169. In Figure 40 a set of segmented backscattered electron micrographs from five different heat treatments of Ti-5553 were analyzed using 2-point statistics and then projected into a 3D space of the first three principal components. In the 3D principal component space, the five groups of materials are visually separable into clusters by heat treatment conditions, shown as different colored hulls.

For microstructure representation based on grain shape, the shapes of the grain regions can be computed using a set of moment invariants (described in the section "Shape Identification with Moment Invariants") that allow for shape identification despite similarity and/or affine transformations. An example is drawn from Ref. 132.

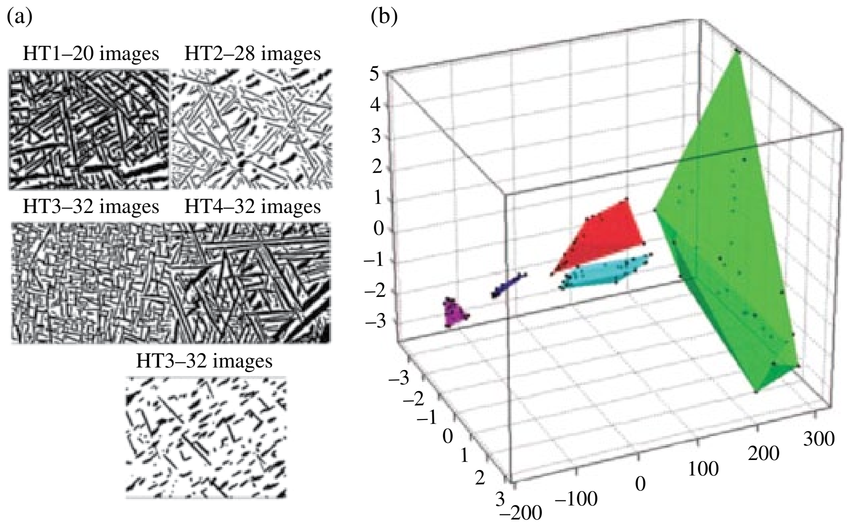


FIGURE 40 (a) Segmented backscattered electron micrographs for five different heat treatments of Ti-5553 (b) 3D vector space of the first three principal components of the 2-point micrograph statistics. The five groups of materials are clustered by heat treatment type. Clusters are indicated by colored hulls (color available in e-book version). Reproduced from Ref. 165 with kind permission from Springer Science and Business Media.

In Figure 41, a 2D phase field simulation for a Ni–Al binary is shown as the upper left inset. The grains are then identified for shape through their projection into the (ω_1, ω_2) moment invariant space. The distribution of grain shapes in (ω_1, ω_2) space can also be used as a "fingerprint" to identify the most likely 3D particles observed in the 2D micrograph.[115] As an example drawn from Ref. 115, Figure 42 shows three unique shape distributions for cube, octahedron, and tetrahedron particles. For series micrographs, the particle shape distribution can also be tracked as a function of processing parameters.[132]

The image processing techniques discussed are only a small subset of the broad and rapidly growing area of computer vision. There are myriad of research opportunities available in the application area of automated micrograph analysis.

Structure–Property Relationships in Amorphous Materials

It is now possible to use atomic-scale calculations to predict many properties of crystalline materials accurately, but amorphous materials pose a greater challenge due to the lack of symmetry in their atomic structure. To make it computationally feasible to calculate the properties of an amorphous material, the material may be approximated as having a pseudo-amorphous structure, in which a large amorphous unit cell is repeated periodically (Figure 43). However the amorphous structure of atoms within this unit cell is not well defined—in many cases, a single pseudo-amorphous structure will not be sufficiently representative of the many possible atomic arrangements that might occur in an amorphous material. In such cases, it is preferable to

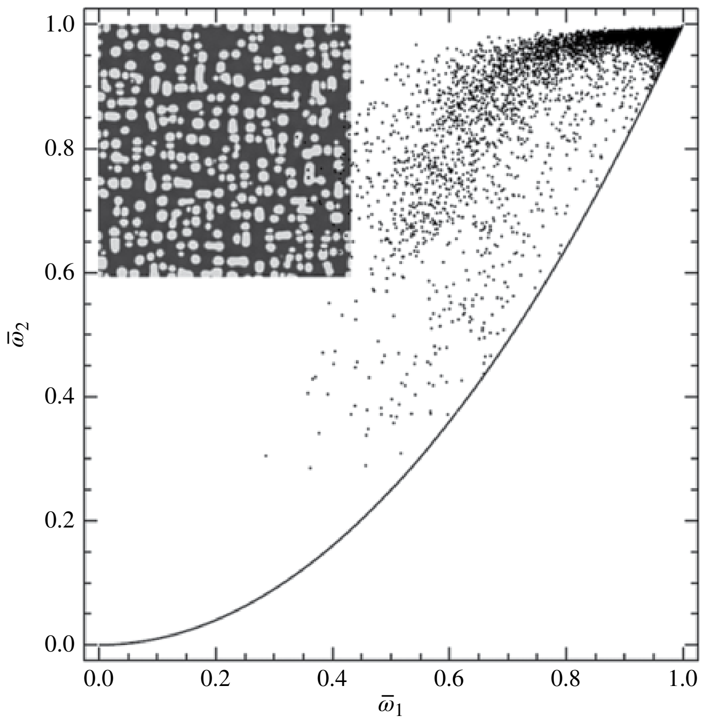


FIGURE 41 Inset of 2D phase field simulation for a Ni–Al binary. The grain shapes from the micrograph are plotted in the ω_1–ω_2 vector space for shape identification. Reprinted from Ref. 132, with permission from Elsevier.

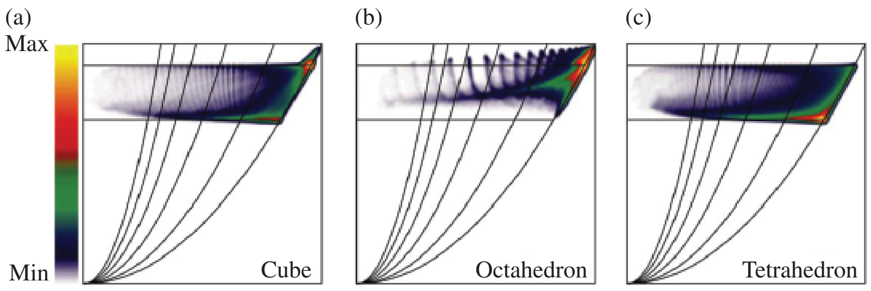


FIGURE 42 Characteristic micrograph shape distributions in the ω_1–ω_2 vector space for (a) cubic particles, (b) octahedron particles, and (c) tetrahedron particles (color available in e-book version). Reproduced with permission from Ref. 115. © IOP Publishing. All rights reserved.

calculate the properties of an ensemble of pseudo-amorphous structures. Analyzing the resulting data to determine the relationship between the atomic structure of amorphous materials and their properties is a task that is well suited for machine learning.

The utility of machine learning to study structure–property relationships in amorphous materials was demonstrated by Mueller et al., in a study of

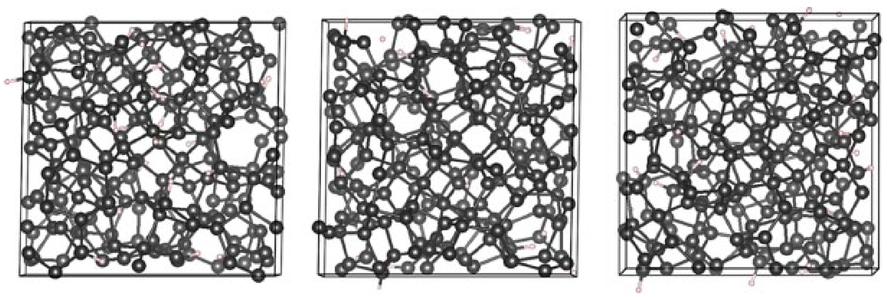


FIGURE 43 Unit cells for three different model nc-Si:H structures. Gray spheres represent silicon, and white spheres represent hydrogen. In Refs. 106 and 176, properties for thousands of such structures were calculated.

hydrogenated nanocrystalline (nc-Si:H) and amorphous (a-Si:H) silicon.[106] Hydrogenated amorphous silicon is a widely used thin-film photovoltaic material, but the efficiency of a-Si:H solar cells is limited by the low hole mobility in this material.[177] The efficiency of a-Si:H solar cells can be improved by using nc-Si:H, which consists of small crystalline Si particles surrounded by a-Si:H. To identify the structural features contributing to hole traps in nc-Si:H, Mueller et al. used density functional theory to calculate hole trap depths in an ensemble of more than 1000 nc-Si:H structures; the approach was based on that used in Ref. 176. For each sample, they also calculated 242 descriptors of the atomic structure (e.g., unit cell volume, smallest Si–H bond length, etc.). They then used genetic programming (described previously) to identify functions of the descriptors that best predicted the hole trap depths.

By identifying a set of functions that relate the descriptors to hole trap depths, the authors were able to determine which descriptors most influence hole trap depths, even if the relationship is nonlinear. The best functions identified by the genetic programming algorithm consistently contained descriptors that fell into one of three classes: fivefold coordinated silicon (floating bonds), regions of dense silicon, and Si–H–Si (bridge) bonds. Taken together, these results indicated that holes were more likely to be trapped in the amorphous regions of the material. The first two classes of structural features had been identified in previous computational studies of hole traps in a-Si:H,[176,178–181] but the third (bridge bonds) had not.

To confirm the relationship between bridge bonds and the DFT-calculated hole trap depths, the authors used the same methodology on an ensemble of 2700 a-Si:H calculations that had been generated for a previous publication.[176] Extensive prior examinations of this data set, using methods other than machine learning, had not revealed a relationship between hole trap depths and bridge bonds. However, within minutes, the genetic programming algorithm was able to identify a similar relationship to the one it had found in the nc-Si:H samples. Inspections of the locations of deep hole traps confirmed that they are located near bridge bonds, further supporting this relationship.

As statistical sampling methods become more common in materials research, the large volumes of data generated will become increasingly difficult for humans to analyze using traditional methods. The examples presented in this section and previous sections demonstrate how machine learning can play an important role in facilitating the analysis of such large data sets, providing insights into complex materials that elude human researchers.

ADDITIONAL RESOURCES

There are many comprehensive resources available[1] for readers interested in learning more about machine learning algorithms and analysis. Two in particular are:

1. T. Hastie, J. Friedman, and R. Tibshirani, *The Elements of Statistical Learning*, Springer, New York, 2009.
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2007.

Open-source machine learning packages exist for all the algorithms described here. The Python package scikit-learn (available at http://scikit-learn.org/) has a library of methods with appropriate documentation. The commercial product MATLAB also has a library of methods as part of the toolboxes such as Statistics, Optimization, Neural Networks, Signal Processing, Image Processing, and Computer Vision. These methods come with extensive documentation including examples. For methods specific to image and video processing, the package OpenCV (http://opencv.org/) is extensive but may require background knowledge in machine learning. The commercial software package Eureqa (http://www.nutonian.com/products/eureqa/) provides a straightforward interface for genetic programming.

SUMMARY

This chapter addresses the role that data-driven approaches, especially machine learning methods, are expected to play in materials research in the immediate future. Machine learning, an important part of artificial intelligence, has already made monumental contributions to areas outside materials science (ranging from commerce to gaming to search engines to drug design). These methods come in many flavors under many names with a copious amount of jargon. Keeping these aspects in mind, this chapter first provided the necessary mathematical background (at a basic and unified level) to allow a materials researcher entering this field to

[1] Certain commercial equipment, instruments, or materials are identified in this publication for informational purposes only. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

use these methods most effectively. The chapter then provided an assortment of examples of recent machine learning applications within materials science and discussed a range of emerging efforts, including high-throughput phase diagram and crystal structure determination methods, accelerated prediction of materials properties, development of interatomic potentials and functionals for accelerating materials simulations, and efficient and low-cost methods for materials process control.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Kenny Lipkowitz of the Office of Naval Research for the opportunity and invitation to create this manuscript. RR acknowledges financial support from the Office of Naval Research, which was instrumental in initiating research related to machine learning in his group. TM acknowledges financial support from the National Science Foundation under award number 1352373.

REFERENCES

1. R. LeSar, *Stat. Anal. Data Min.*, **1**, 372 (2009). Materials Informatics: An Emerging Technology for Materials Development.
2. K. Rajan, *Mater. Today*, **8**, 38 (2005). Materials Informatics.
3. C. J. Long, J. Hattrick-Simpers, M. Murakami, R. C. Srivastava, I. Takeuchi, V. L. Karen, and X. Li, *Rev. Sci. Instrum.*, **78**, 072217 (2007). Rapid Structural Mapping of Ternary Metallic Alloy Systems Using the Combinatorial Approach and Cluster Analysis.
4. G. Hautier, C. C. Fischer, A. Jain, T. Mueller, and G. Ceder, *Chem. Mater.*, **22**, 3762 (2010). Finding Nature's Missing Ternary Oxide Compounds Using Machine Learning and Density Functional Theory.
5. D. Morgan, S. Curtarolo, K. Persson, J. Rodgers, and G. Ceder, *Phys. Rev. Lett.*, **91**, 135503 (2003). Predicting Crystal Structures with Data Mining of Quantum Calculations.
6. G. Pilania, C. Wang, X. Jiang, S. Rajasekaran, and R. Ramprasad, *Sci. Rep.*, **3**, 1 (2013). Accelerating Materials Property Predictions Using Machine Learning.
7. K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, *J. Chem. Theory Comput.*, **9**, 3404 (2013). Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies.
8. K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, *J. Phys. Chem. Lett.*, **6**, 2326 (2015). Machine Learning Predictions of Molecular Properties: Accurate Many-Body Potentials and Nonlocality in Chemical Space.
9. T. D. Huan, A. Mannodi-Kanakkithodi, and R. Ramprasad, *Phys. Rev. B*, **92**, 014106 (2015). Accelerated Materials Property Predictions and Design Using Motif-Based Fingerprints.

10. T. Morawietz and J. Behler, *J. Phys. Chem. A*, **117**, 7356 (2013). A Density-Functional Theory-Based Neural Network Potential for Water Clusters Including van der Waals Corrections.
11. J. Behler, *Phys. Chem. Chem. Phys.*, **13**, 17930 (2011). Neural Network Potential-Energy Surfaces in Chemistry: A Tool for Large-Scale Simulations.
12. A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, *Phys. Rev. Lett.*, **104**, 136403 (2010). Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, Without the Electrons.
13. V. Botu and R. Ramprasad, *Phys. Rev. B*, **92**, 094306 (2015). Learning Scheme to Predict Atomic Forces and Accelerate Materials Simulations.
14. V. Botu and R. Ramprasad, *Int. J. Quantum Chem.*, **115**, 1074 (2015). Adaptive Machine Learning Framework to Accelerate Ab Initio Molecular Dynamics.
15. J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, *Phys. Rev. Lett.*, **108**, 253002 (2012). Finding Density Functionals with Machine Learning.
16. A. G. Kusne, T. Gao, A. Mehta, L. Ke, M. C. Nguyen, K.-M. Ho, V. P. Antropov, C.-Z. Wang, M. J. Kramer, C. J. Long, and I. Takeuchi, *Sci. Rep.*, **4**, Article number: 6367 (2014). On-the-Fly Machine-Learning for High-Throughput Experiments: Search for Rare-Earth Free Permanent Magnets.
17. H. K. D. H. Bhadeshia, *Stat. Anal. Data Min.*, **1**, 296 (2009). Neural Networks and Information in Materials Science.
18. T. Bayes and R. Price, *Philos. Trans. (1683–1775)*, **53**, 370 (1763). An Essay Towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S.
19. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, John Wiley & Sons, Washington, DC, 1977.
20. R. E. Kass and L. Wasserman, *J. Am. Stat. Assoc.*, **91**, 1343 (1996). The Selection of Prior Distributions by Formal Rules.
21. E. T. Jaynes, *Phys. Rev.*, **106**, 620 (1957). Information Theory and Statistical Mechanics.
22. G. Schwarz, *Ann. Stat.*, **6**, 461 (1978). Estimating the Dimension of a Model.
23. H. Akaike, *IEEE Trans. Autom. Control*, **19**, 716 (1974). A New Look at the Statistical Model Identification.
24. J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, John Wiley & Sons, Inc., Hoboken, NJ, 2005.
25. R. Horst, P. M. Pardalos, and N. V. Thoai, *Introduction to Global Optimization*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2000.
26. R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, Ltd, Chichester, 2013.
27. I. Guyon, G. Cawley, G. Dror, and V. Lemaire, Results of the Active Learning Challenge, in *JMLR: Workshop and Conference Proceedings,* 2011, vol 16, pp. 19–45. Workshop on Active Learning and Experimental Design.
28. B. Settles, University of Wisconsin, Madison, (2010). Active Learning Literature Survey.
29. D. C. Montgomery, *Design and Analysis of Experiments*, 8th Edition, John Wiley & Sons, Inc., New York, 2012.
30. D. D. Lewis and W. A. Gale, in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Springer-Verlag, Inc., New York, 1994. A Sequential Algorithm for Training Text Classifiers.

31. H. S. Seung, M. Opper, and H. Sompolinsky, in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ACM, New York, 1992. Query by Committee.
32. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, New York, 2009.
33. H. Chernoff, *Ann. Math. Stat.*, **24**, 586 (1953). Locally Optimal Designs for Estimating Parameters.
34. A. Wald, *Ann. Math. Stat.*, **14**, 134 (1943). On the Efficient Design of Statistical Investigations.
35. J. Kiefer and J. Wolfowitz, *Ann. Math. Stat.*, **30**, 271 (1959). Optimum Designs in Regression Problems.
36. E. A. Rady, M. M. E. Abd El-Monsef, and M. M. Seyam, *InterStat*, **247**, 1 (2009). Relationships Among Several Optimality Criteria.
37. K. Chaloner and I. Verdinelli, *Stat. Sci.*, **10**, 273 (1995). Bayesian Experimental Design: A Review.
38. A. DasGupta, in *Handbook of Statistics*, S. Ghosh and C. R. Rao, Editors, Elsevier, Amsterdam, 1996, pp. 1099–1147. Review of Optimal Bayes Designs.
39. A. K. Kurtz, *Pers. Psychol.*, **1**, 41 (1948). A Research Test of the Rorschach Test.
40. C. I. Mosier, *Educ. Psychol. Meas.*, **11**, 5 (1951). The Need and Means of Cross Validation. I. Problems and Designs of Cross-Validation.
41. S. Geisser, *J. Am. Stat. Assoc.*, **70**, 320 (1975). The Predictive Sample Reuse Method with Applications.
42. M. Stone, *J. R. Stat. Soc. Ser. B Methodol.*, **36**, 111 (1974). Cross-Validatory Choice and Assessment of Statistical Predictions.
43. B. Efron, *Ann. Stat.*, **7**, 1 (1979). Bootstrap Methods: Another Look at the Jackknife.
44. B. Efron and R. Tibshirani, *J. Am. Stat. Assoc.*, **92**, 548 (1997). Improvements on Cross-Validation: The .632+ Bootstrap Method.
45. B. Efron, *J. Am. Stat. Assoc.*, **99**, 619–642 (2004). The Estimation of Prediction Error: Covariance Penalties and Cross-Validation [With comments by P. Burman, L. Denby, J. M. Landwehr, C. L. Mallows, X. Shen, H.-C. Huang, J. Ye, and C. Zhang].
46. R. Kohavi, in *IJCAI'95 Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1995, vol 2, pp. 1137–1143. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection.
47. M. W. Browne, *J. Math. Psychol.*, **44**, 108 (2000). Cross-Validation Methods.
48. S. Arlot and A. Celisse, *Stat. Surv.*, **4**, 40 (2010). A Survey of Cross-Validation Procedures for Model Selection.
49. B. Efron and G. Gong, *Am. Stat.*, **37**, 36 (1983). A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation.
50. S. Borra and A. Di Ciaccio, *Comput. Stat. Data Anal.*, **54**, 2976 (2010). Measuring the Prediction Error. A Comparison of Cross-Validation, Bootstrap and Covariance Penalty Methods.
51. A. E. Hoerl and R. W. Kennard, *Technometrics*, **12**, 55 (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems.
52. R. Tibshirani, *J. R. Stat. Soc. Ser. B Methodol.*, **58**, 267 (1996). Regression Shrinkage and Selection via the Lasso.

53. D. L. Donoho, *IEEE Trans. Inf. Theory*, **52**, 1289 (2006). Compressed Sensing.
54. T. Goldstein and S. Osher, *SIAM J. Imaging Sci.*, **2**, 323 (2009). The Split Bregman Method for L1-Regularized Problems.
55. C. Cortes and V. Vapnik, *Mach. Learn.*, **20**, 273 (1995). Support-Vector Networks.
56. C. J. C. Burges, *Data Min. Knowl. Disc.*, **2**, 121 (1998). A Tutorial on Support Vector Machines for Pattern Recognition.
57. R. G. Brereton and G. R. Lloyd, *Analyst*, **135**, 230 (2010). Support Vector Machines for Classification and Regression.
58. P. Wolfe, *Econometrica*, **27**, 382 (1959). The Simplex Method for Quadratic Programming.
59. M. Frank and P. Wolfe, *Nav. Res. Logist. Q.*, **3**, 95 (1956). An Algorithm for Quadratic Programming.
60. J. A. K. Suykens and J. Vandewalle, *Neural Process. Lett.*, **9**, 293 (1999). Least Squares Support Vector Machine Classifiers.
61. M. Aizerman, E. Braverman, and L. Rozoner, *Autom. Remote Control*, **25**, 821 (1964). Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning.
62. J. Mercer, *Philos. Trans. R. S. Lond. A*, **209**, 415 (1909). Functions of Positive and Negative Type, and Their Connection with the Theory of Integral Equations.
63. N. Aronszajn, *Trans. Am. Math. Soc.*, **68**, 337 (1950). Theory of Reproducing Kernels.
64. B. Schölkopf, R. Herbrich, and A. J. Smola. in *Computational Learning Theory*, Springer-Verlag, Berlin, 2001, pp. 416. A Generalized Representer Theorem.
65. G. S. Kimeldore and G. Wahba, *Ann. Math. Stat.*, **41**, 495 (1970). A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines.
66. C. Saunders, A. Gammerman, and V. Vovk, in *(ICML-1998) Proceedings of the 15th International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1998, pp. 515–521. Ridge Regression Learning Algorithm in Dual Variables.
67. W. S. McCulloch and W. Pitts, *Bull. Math. Biophys.*, **5**, 115 (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity.
68. F. Rosenblatt, *Psychol. Rev.*, **65**, 386 (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.
69. P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Harvard University, Cambridge, MA, 1974.
70. K.-I. Funahashi, *Neural Netw.*, **2**, 183 (1989). On the Approximate Realization of Continuous Mappings by Neural Networks.
71. D. F. Specht, *IEEE Trans. Neural Netw.*, **2**, 568 (1991). A General Regression Neural Network.
72. B. Widrow and M. A. Lehr, *Proc. IEEE*, **78**, 1415 (1990). 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation.
73. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall PTR, Upper Saddle River, NJ, 1998.
74. C. Kingsford and S. L. Salzberg, *Nat. Biotechnol.*, **26**, 1011 (2008). What Are Decision Trees?
75. S. K. Murthy, *Data Min. Knowl. Disc.*, **2**, 345 (1998). Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey.

76. S. R. Safavian and D. Landgrebe, *IEEE Trans. Syst. Man Cybern.*, **21**, 660 (1991). A Survey of Decision Tree Classifier Methodology.
77. W.-Y. Loh, *Wiley Interdiscip. Rev. Data Min. Knowl. Disc.*, **1**, 14 (2011). Classification and Regression Trees.
78. E. W. Bucholz, C. S. Kong, K. R. Marchman, W. G. Sawyer, S. R. Phillpot, S. B. Sinnott, and K. Rajan, *Tribol. Lett.*, **47**, 211 (2012). Data-Driven Model for Estimation of Friction Coefficient via Informatics Methods.
79. D. A. Carr, M. Lach-Hab, S. J. Yang, I. I. Vaisman, and E. Blaisten-Barojas, *Microporous Mesoporous Mater.*, **117**, 339 (2009). Machine Learning Approach for Structure-Based Zeolite Classification.
80. L. Hyafil and R. L. Rivest, *Inf. Process. Lett.*, **5**, 15 (1976). Constructing Optimal Binary Decision Trees Is NP-Complete.
81. J. Ross Quinlan, *Mach. Learn.*, **1**, 81 (1986). Induction of Decision Trees.
82. L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*, CRC Press, Boca Raton, FL, 1984.
83. Y. Wang and I. H. Witten, in *Proceedings of the Ninth European Conference on Machine Learning*, Springer-Verlag, London, 1997, pp. 128. Inducing Model Trees for Continuous Classes.
84. J. Mingers, *Mach. Learn.*, **4**, 227 (1989). An Empirical Comparison of Pruning Methods for Decision Tree Induction.
85. J. R. Quinlan, *Int. J. Man Mach. Stud.*, **27**, 221 (1987). Simplifying Decision Trees.
86. L. Breiman, *Mach. Learn.*, **45**, 5 (2001). Random Forests.
87. R. Polikar, *IEEE Circuits Syst. Mag.*, **6**, 21 (2006). Ensemble Based Systems in Decision Making.
88. L. Rokach, *Artif. Intell. Rev.*, **33**, 1 (2010). Ensemble-Based Classifiers.
89. R. Maclin and D. Opitz, arXiv preprint arXiv:1106.0257 (2011). Popular Ensemble Methods: An Empirical Study.
90. L. Breiman, *Mach. Learn.*, **24**, 123 (1996). Bagging Predictors.
91. R. E. Schapire, *Mach. Learn.*, **5**, 197 (1990). The Strength of Weak Learnability.
92. Y. Freund and R. E. Schapire, *J. Comput. Syst. Sci.*, **55**, 119 (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.
93. P. M. Long and R. A. Servedio, *Mach. Learn.*, **78**, 287 (2010). Random Classification Noise Defeats All Convex Potential Boosters.
94. N. L. Cramer, in *Proceedings of the First International Conference on Genetic Algorithms*, L. Erlbaum Associates Inc., Hillsdale, NJ, 1985, pp. 183. A Representation for the Adaptive Generation of Simple Sequential Programs.
95. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.
96. J. R. Koza, in *IJCAI'89 Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1989, vol 1, pp. 768. Hierarchical Genetic Algorithms Operating on Populations of Computer Programs.
97. B.-T. Zhang and H. Mühlenbein, *Evol. Comput.*, **3**, 17 (1995). Balancing Accuracy and Parsimony in Genetic Programming.

98. K. Rodriguez-Vazquez, C. M. Fonseca, and P. J. Fleming, in *Late Breaking Papers at the 1997 Genetic Programming Conference*, Morgan Kaufmann Publishers, San Francisco, 1997. Multiobjective Genetic Programming: a Nonlinear System Identification Application.
99. A. H. Gandomi, A. H. Alavi, and M. G. Sahab, *Mater. Struct.*, **43**, 963 (2010). New Formulation for Compressive Strength of CFRP Confined Concrete Cylinders Using Linear Genetic Programming.
100. E. Ozbay, M. Gesoglu, and E. Guneyisi, *Construct. Build Mater.*, **22**, 1831 (2008). Empirical Modeling of Fresh and Hardened Properties of Self-Compacting Concretes by Genetic Programming.
101. A. Baykasoglu, T. Dereli, and S. Tanis, *Cem. Concr. Res.*, **34**, 2083 (2004). Prediction of Cement Strength Using Soft Computing Techniques.
102. A. H. Alavi, M. Ameri, A. H. Gandomi, and M. R. Mirzahosseini, *Construct. Build Mater.*, **25**, 1338 (2011). Formulation of Flow Number of Asphalt Mixes Using a Hybrid Computational Method.
103. M. Kovacic, P. Uratnik, M. Brezocnik, and R. Turk, *Mater. Manuf. Processes*, **22**, 634 (2007). Prediction of the Bending Capability of Rolled Metal Sheet by Genetic Programming.
104. M. Brezocnik, M. Kovacic, and M. Ficko, *J. Mater. Process. Technol.*, **157**, (2004). Prediction of Surface Roughness with Genetic Programming.
105. M. Brezocnik and M. Kovacic, *Mater. Manuf. Processes*, **18**, 475 (2003). Integrated Genetic Programming and Genetic Algorithm Approach to Predict Surface Roughness.
106. T. Mueller, E. Johlin, and J. C. Grossman, *Phys. Rev. B*, **89**, 115 (2014). Origins of Hole Traps in Hydrogenated Nanocrystalline and Amorphous Silicon Revealed Through Machine Learning.
107. R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A Field Guide to Genetic Programming*, Lulu.com, 2008.
108. I. J. Good, *Philos. Sci.*, **50**, 283 (1983). The Philosophy of Exploratory Data Analysis.
109. L. A. Baumes, M. Moliner, and A. Corma, *Chem. Eur. J.*, **15**, 4258 (2009). Design of a Full-Profile-Matching Solution for High-Throughput Analysis of Multiphase Samples Through Powder X-Ray Diffraction.
110. S. Kullback, *Information Theory and Statistics*, Courier Dover Publications, Dover, 2012.
111. S. Kullback and R. A. Leibler, *Ann. Math. Stat.*, **22**, 79 (1951). On Information and Sufficiency.
112. H. Jeffreys, *Proc. R. Soc. Lond. A Math. Phys. Sci.*, **186**, 453(1946). An Invariant Form for the Prior Probability in Estimation Problems.
113. J. Puzicha, T. Hofmann, and J. M. Buhmann, in *Computer Vision and Pattern Recognition, 1997. Proceedings, 1997 IEEE Computer Society Conference on,* IEEE, New York, 1997. Non-parametric Similarity Measures for Unsupervised Texture Segmentation and Image Retrieval.
114. D. Comaniciu, V. Ramesh, and P. Meer, *IEEE Trans. Pattern Anal. Mach. Intell.*, **25**, 564 (2003). Kernel-Based Object Tracking.
115. P. G. Callahan, J. P. Simmons, and M. De Graef, *Model. Simul. Mater. Sci. Eng.*, **21**, 015003 (2013). A Quantitative Description of the Morphological Aspects of Materials Structures Suitable for Quantitative Comparisons of 3D Microstructures.

116. C. W. Niblack, R. Barber, W. Equitz, M. D. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, in *IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology*. International Society for Optics and Photonics, Bellingham WA, 1993. QBIC Project: Querying Images by Content, Using Color, Texture, and Shape.
117. H. Sakoe and S. Chiba, *IEEE Trans. Acoust. Speech Signal Process.*, **26**, 43 (1978). Dynamic Programming Algorithm Optimization for Spoken Word Recognition.
118. Y. Rubner, C. Tomasi, and L. J. Guibas, in *Sixth International Conference on Computer Vision*. IEEE, New York, 1998, pp. 59–66. A Metric for Distributions with Applications to Image Databases.
119. Y. Rubner, C. Tomasi, and L. J. Guibas, *Int. J. Comput. Vis.*, **40**, 99 (2000). The Earth Mover's Distance as a Metric for Image Retrieval.
120. G. Al-Naymat, S. Chawla, and J. Taheri, in *Proceedings of the Eighth Australasian Data Mining Conference*. Australian Computer Society, Inc., Sydney, 2009, vol 101, pp. 117. SparseDTW: a Novel Approach to Speed up Dynamic Time Warping.
121. S. Salvador and P. Chan, *Intell. Anal.*, **11**, 561 (2007). Toward Accurate Dynamic Time Warping in Linear Time and Space.
122. O. Pele and M. Werman, in *2009 IEEE 12th International Conference on Computer Vision*, IEEE, New York, 2009, pp. 460. Fast and Robust Earth Mover's Distances.
123. D. Comaniciu and P. Meer, *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**, 603 (2002). Mean Shift: A Robust Approach Toward Feature Space Analysis.
124. P. Macnaughton-Smith, W. T. Williams, M. B. Dale, and L. G. Mockett, *Nature*, **202**, 1034 (1964). Dissimilarity Analysis: A New Technique of Hierarchical Sub-Division.
125. U. Von Luxburg, *Clustering Stability*, Now Publishers Inc, Hanover, MA, 2010.
126. C. Goutte, L. K. Hansen, M. G. Liptrot, and E. Rostrup, *Hum. Brain Mapp.*, **13**, 165 (2001). Feature-Space Clustering for fMRI Meta-Analysis.
127. C. A. Sugar and G. M. James, *J. Am. Stat. Assoc.*, **98**, 750 (2003). Finding the Number of Clusters in a Dataset.
128. A. Niederliński and Wydawnictwo Pracownia Komputerowa Jacka Skalmierskiego, *A Quick and Gentle Guide to Constraint Logic Programming via ECLiPSe*, Jacek Skalmierski Computer Studio, Gliwice, 2011.
129. Y. K. Yoo, Q. Xue, Y. S. Chu, S. Xu, U. Hangen, H.-C. Lee, W. Stein, and X.-D. Xiang, *Intermetallics*, **14**, 241 (2006). Identification of Amorphous Phases in the Fe–Ni–Co Ternary Alloy System Using Continuous Phase Diagram Material Chips.
130. A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
131. J. S. Lim and A. V. Oppenheim, *Advanced Topics in Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1987.
132. J. P. MacSleyne, J. P. Simmons, and M. De Graef, *Acta Mater.*, **56**, 427 (2008). On the Use of 2-D Moment Invariants for the Automated Classification of Particle Shapes.
133. M. L. Green, I. Takeuchi, and J. R. Hattrick-Simpers, *J. Appl. Phys.*, **113**, 231101 (2013). Applications of High Throughput (Combinatorial) Methodologies to Electronic, Magnetic, Optical, and Energy-Related Materials.
134. C. J. Long, D. Bunker, X. Li, V. L. Karen, and I. Takeuchi, *Rev. Sci. Instrum.*, **80**, 103902 (2009). Rapid Identification of Structural Phases in Combinatorial Thin-Film Libraries Using x-Ray Diffraction and Non-negative Matrix Factorization.

135. R. LeBras, T. Damoulas, J. M. Gregoire, A. Sabharwal, C. P. Gomes, and R. Bruce Van Dover, in *Principles and Practice of Constraint Programming–CP 2011*, Springer, Berlin, 2011, pp. 508–522. Constraint Reasoning and Kernel Clustering for Pattern Decomposition with Scaling.

136. M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, *Phys. Rev. Lett.*, **108**, 058301 (2012). Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning.

137. W. Kohn and L. J. Sham, *Phys. Rev.*, **140**, A1133 (1965). Self-Consistent Equations Including Exchange and Correlation Effects.

138. P. Hohenberg and W. Kohn, *Phys. Rev.*, **136**, 864 (1964). Inhomogeneous Electron Gas.

139. W. Sidney, *J. Chem. Educ.*, **42**, 502 (1965). Benson, III—Bond Energies.

140. J. J. P. Stewart, *J. Mol. Model.*, **13**, 1173 (2007). Optimization of Parameters for Semiempirical Methods V: Modification of NDDO Approximations and Application to 70 Elements.

141. I. Torrens, *Interatomic Potentials*, Elsevier, Amsterdam, 2012.

142. J. Behler, *J. Chem. Phys.*, **134**, 074106 (2011). The Atom-Centered Symmetry Functions for Constructing High-Dimensional Neural Network Potentials.

143. J. Behler and M. Parrinello, *Phys. Rev. Lett.*, **98**, 146401 (2007). Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces.

144. S. M. Woodley and R. Catlow, *Nat. Mater.*, **7**, 937 (2008). Crystal Structure Prediction from First Principles.

145. J. Maddox, *Nature*, **335**, 201 (1988). Crystals from First-Principles.

146. C. C. Fischer, K. J. Tibbetts, D. Morgan, and G. Ceder, *Nat. Mater.*, **5**, 641 (2006). Predicting Crystal Structure by Merging Data Mining with Quantum Mechanics.

147. Fiz Karlsruhe, *Inorganic crystal structure database* (http://www.fiz-karlsruhe.de/icsd.html, accessed October 9, 2015).

148. K. Burke, *J. Chem. Phys.*, **136**, 150901 (2012). Perspective on Density Functional Theory.

149. E. Ising, *Zeitschrift für Physik*, **31**, 253 (1925). Beitrag zur Theorie des Ferromagnetismus.

150. J. M. Sanchez, F. Ducastelle, and D. Gratias, *Physica*, **128A**, 334 (1984). Generalized Cluster Description of Multicomponent Systems.

151. A. van de Walle and G. Ceder, *J. Phase Equilib.*, **23**, 348 (2002). Automating First-Principles Phase Diagram Calculations.

152. R. Drautz and A. Diaz-Ortiz, *Phys. Rev. B*, **73**, 224207 (2006). Obtaining Cluster Expansion Coefficients in Ab Initio Thermodynamics of Multicomponent Lattice-Gas Systems.

153. D. B. Laks, L. G. Ferreira, S. Froyen, and A. Zunger, *Phys. Rev. B*, **46**, 12587 (1992). Efficient Cluster Expansion for Substitutional Systems.

154. V. Blum and A. Zunger, *Phys. Rev. B*, **70**, 115108 (2004). Mixed-Basis Cluster Expansion for Thermodynamics of bcc Alloys.

155. T. Mueller and G. Ceder, *Phys. Rev. B*, **80**, 024103 (2009). Bayesian Approach to Cluster Expansions.

156. T. Mueller, *Phys. Rev. B*, **86**, 144201 (2012). Ab Initio Determination of Structure–Property Relationships in Alloy Nanoparticles.

157. L. J. Nelson, G. L. W. Hart, F. Zhou, and V. Ozolins, *Phys. Rev. B*, **87**, 035125 (2013). Compressive Sensing as a Paradigm for Building Physics Models.

158. T. Mueller, in *Department of Materials Science and Engineering*, Massachusetts Institute of Technology, Cambridge, MA, 2007, p. 199. Computational Studies of Hydrogen Storage Materials and the Development of Related Methods.

159. L. J. Nelson, V. Ozoliņš, C. Shane Reese, F. Zhou, and G. L. W. Hart, *Phys. Rev. B*, **88**, 155105 (2013). Cluster Expansion Made Easy with Bayesian Compressive Sensing.

160. A. Zunger, S. H. Wei, L. G. Ferreira, and J. E. Bernard, *Phys. Rev. Lett.*, **65**, 353–356 (1990). Special Quasirandom Structures.

161. T. Mueller and G. Ceder, *Phys. Rev. B*, **82**, 184107 (2010). Exact Expressions for Structure Selection in Cluster Expansions.

162. A. Seko, Y. Koyama, and I. Tanaka, *Phys. Rev. B*, **80**, 165122 (2009). Cluster Expansion Method for Multicomponent Systems Based on Optimal Selection of Structures for Density-Functional Theory Calculations.

163. A. Seko and I. Tanaka, *Phys. Rev. B*, **83**, 224111 (2011). Grouping of Structures for Cluster Expansion of Multicomponent Systems with Controlled Accuracy.

164. Y. C. Lin, J. Zhang, and J. Zhong, *Comput. Mater. Sci.*, **43**, 752 (2008). Application of Neural Networks to Predict the Elevated Temperature Flow Behavior of a Low Alloy Steel.

165. R. K. Jain, V. K. Jain, and P. K. Kalra, *Wear*, **231**, 242 (1999). Modelling of Abrasive Flow Machining Process: A Neural Network Approach.

166. T. Sourmail, H. K. D. H. Bhadeshia, and D. J. C. MacKay, *Mater. Sci. Technol.*, **18**, 655 (2002). Neural Network Model of Creep Strength of Austenitic Stainless Steels.

167. M. A. Yescas, H. K. D. H. Bhadeshia, and D. J. MacKay, *Mater. Sci. Eng. A*, **311**, 162 (2001). Estimation of the Amount of Retained Austenite in Austempered Ductile Irons Using Neural Networks.

168. E. A. Metzbower, J. J. DeLoach, S. H. Lalam, and H. K. D. H. Bhadeshia, *Sci. Technol. Weld. Joining*, **6**, 116 (2001). Neural Network Analysis of Strength and Ductility of Welding Alloys for High Strength Low Alloy Shipbuilding Steels.

169. S. R. Kalidindi, S. R. Niezgoda, and A. A. Salem, *JOM*, **63**, 34 (2011). Microstructure Informatics Using Higher-Order Statistics and Efficient Data-Mining Protocols.

170. V. H. C. de Albuquerque, A. R. de Alexandria, P. C. Cortez, and J. M. R. S. Tavares, *NDT&E Int.*, **42**, 644 (2009). Evaluation of Multilayer Perceptron and Self-Organizing Map Neural Network Topologies Applied on Microstructure Segmentation from Metallographic Images.

171. J. P. Simmons, P. Chuang, M. Comer, J. E. Spowart, M. D. Uchic, and M. De Graef, *Model. Simul. Mater. Sci. Eng.*, **17**, 025002 (2009). Application and Further Development of Advanced Image Processing Algorithms for Automated Analysis of Serial Section Image Data.

172. J. Waggoner, Y. Zhou, J. Simmons, M. De Graef, and S. Wang, *IEEE Trans. Image Process.*, **22**(12), 5282 (2013). 3D Materials Image Segmentation by 2D Propagation: A Graph-Cut Approach Considering Homomorphism.

173. L. Huffman, J. Simmons, M. De Graef, and I. Pollak, in *Proceedings of IEEE Statistical Signal Processing Workshop*, 2011, pp. 28–30. Shape Priors for MAP Segmentation of Alloy Micrographs Using Graph Cuts.

174. D. T. Fullwood, S. R. Kalidindi, S. R. Niezgoda, A. Fast, and N. Hampson, *Mater. Sci. Eng. A*, **494**, 68 (2008). Gradient-Based Microstructure Reconstructions from Distributions Using Fast Fourier Transforms.

175. K. Rajan, *Informatics for Materials Science and Engineering: Data-Driven Discovery for Accelerated Experimentation and Application*, Butterworth-Heinemann, Waltham, MA, 2013.

176. E. Johlin, L. K. Wagner, T. Buonassisi, and J. C. Grossman, *Phys. Rev. Lett.*, **110**, 146805 (2013). Origins of Structural Hole Traps in Hydrogenated Amorphous Silicon.

177. V. Avrutin, N. Izyumskaya, and H. Morkoç, *Superlattices Microstruct.*, **49**, 337 (2011). Semiconductor Solar Cells: Recent Progress in Terrestrial Applications.

178. D. A. Drabold, Y. Li, B. Cai, and M. Zhang, *Phys. Rev. B*, **83**, 045201 (2011). Urbach Tails of Amorphous Silicon.

179. Y. Pan, F. Inam, M. Zhang, and D. A. Drabold, *Phys. Rev. Lett.*, **100**, 206403 (2008). Atomistic Origin of Urbach Tails in Amorphous Silicon.

180. Y. Pan, M. Zhang, and D. A. Drabold, *J. Non Cryst. Solids*, **354**, 3480 (2008). Topological and Topological-Electronic Correlations in Amorphous Silicon.

181. P. A. Fedders, D. A. Drabold, and S. Nakhmanson, *Phys. Rev. B*, **58**, 15624 (1998). Theoretical Study on the Nature of Band-Tail States in Amorphous Si.