SIAM J. SCI. COMPUT.
Vol. 33, No. 6, pp. 3381–3401

LEARNING TO PREDICT PHYSICAL PROPERTIES USING SUMS OF SEPARABLE FUNCTIONS*

MAYEUL D'AVEZAC†, RYAN BOTTS‡, MARTIN J. MOHLENKAMP‡, AND ALEX ZUNGER§

Abstract. We present an algorithm for learning the function that maps a material structure to its value on some property, given the value of this function on several structures. We pose this problem as one of learning (regressing) a function of many variables from scattered data. Each structure is first converted to a weighted set of points by a process that removes irrelevant translations and rotations but otherwise retains full information about the structure. Then, incorporating a weighted average for each structure, we construct the multivariate regression function as a sum of separable functions, following the paradigm of separated representations. The algorithm can treat all finite and periodic structures within a common framework, and in particular does not require all structures to lie on a common lattice. We show how the algorithm simplifies when the structures do lie on a common lattice, and we present numerical results for that case.

Key words. separated representation, multivariate regression, optimized materials

AMS subject classifications. 62J02, 62H99, 65D15, 68T05

DOI. 10.1137/100805959

1. Introduction.

Consider some physical property, such as the energy of a given atomic structure or any other property of this structure, which we will denote ρ. This property differs for different material structures σ, and thus there is some function g with $\rho = g(\sigma)$. A numerical method for computing the value of ρ produces a function $\tilde{g}$, which gives some approximation $\tilde{g}(\sigma) \approx g(\sigma)$ for a specific σ. A great deal of effort has gone into developing such numerical methods, and they are now sufficiently accurate for some properties on some classes of structures. As these methods advance, one can image the day when the approximation $\tilde{g}(\sigma) \approx g(\sigma)$ will be sufficiently accurate for any given σ. In this work, we assume that such a sufficiently accurate method is available.

Evaluating $\tilde{g}(\sigma)$ will still be rather expensive, however, and so $\tilde{g}$ must be used sparingly. For example, if one wants to search among a large number of structures for the σ that minimizes ρ, then one can only compute $\tilde{g}(\sigma)$ for relatively few of them. Each time one applies the method to a structure, one generates a data point $(\sigma, \tilde{g}(\sigma)) \approx (\sigma, g(\sigma))$ and so gains some information about g. This information is universal and eternal, so one could collect all such data points from the scientific

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 20, 2010; accepted for publication (in revised form) September 26, 2011; published electronically December 8, 2011.

http://www.siam.org/journals/sisc/33-6/80595.html

†National Renewable Energy Laboratory, 1617 Cole Blvd., Golden, CO 80401 (mayeul.davezac@nrel.gov). This author's work was supported by the U.S. department of Energy, Office of Basic Energy Science and Engineering, under Award DE-AC36-08GO28308, and by the Office of Science, Office of Basic Energy Sciences, Materials Sciences and Engineering Division of the U.S. Department of Energy.

‡Department of Mathematics, Morton Hall 321, 1 Ohio University, Athens, OH 45701 (rb110503@ohio.edu, mohlenka@ohio.edu). The work of these authors was supported by the National Science Foundation under grant DMS-0545895.

§University of Colorado at Boulder, 26 UCB, Boulder, CO 80309-026 (Alex.Zunger@colorado.edu). This author's work was supported by the U.S. Department of Energy, Office of Basic Energy Science and Engineering, under Award DE-AC36-08GO28308.

community into a single database. One can hope that, given enough such data, one can use it to approximate $\tilde{g}(\sigma)$ for σ not in the database without computing $\tilde{g}(\sigma)$ directly. If this approximation is sufficiently accurate and can be computed much faster than computing $\tilde{g}(\sigma)$, then one gains the ability to check ρ on a much larger number of structures. This ability can lead to, for example, optimized materials.

Here we present a framework and algorithm to learn (i.e., regress) g from such data. The foundation is a well-defined way to convert a structure into a weighted set of points in high (formally infinite) dimensions. This conversion has three important characteristics. First, it allows all finite and periodic structures to be considered together, without restrictions such as lying on a common lattice. Second, two structures σ_1 and σ_2 are converted to the same set of points if and only if σ_1 can be obtained as a rotation and translation of σ_2. Third, the resulting set of points can be used within certain multivariate regression algorithms. In this work we adapt the multivariate regression algorithm in [3] to this setting, and we approximate g as a sum of separable functions.

1.1. Formulation of the problem. A *structure* σ is an unordered set of atoms a, with each atom given by a pair $a = (t, \mathbf{r})$, where t is a species type (e.g., $t = \text{Mo}$) and $\mathbf{r}$ is a location in three-dimensional space (e.g., $\mathbf{r} = (x, y, z) = (1, 0, 3.4)$). This set can be infinite, and the only certain constraint is that no two atoms occupy the same location. In our main development we consider structures σ that can be specified by a finite set of atoms $\tilde{\sigma}$ and a periodicity rule. This periodicity rule is usually three linearly independent vectors that specify how to tile the theee-dimensional space with the atoms in $\tilde{\sigma}$. We allow structures with fewer than three vectors, and thus we can have structures that are finite in some directions and periodic in others, or are simply finite. In principle one can allow amorphous structures as well by replacing the algorithm in section 2.1 with a statistical version, but we will not develop that idea.

The property that we are interested in is a function on the set of all structures. We assume that some numerical method has been used to compute its value on some structures, thus giving us a data set from which to learn. We denote this data by

$$D = \{(\sigma_j, \rho_j)\}_{j=1}^{N}, \tag{1.1}$$

where ρ_j is the property of interest. The goal is to approximate this property function, i.e., construct a function f so that $f(\sigma_j) \approx \rho_j$ and $f(\sigma)$ is a good prediction for the property for other σ.

Two structures are *equivalent* if one can be mapped to the other by a translation and/or rotation. We assume that the property of interest is *consistent*, meaning that equivalent structures have the same value. We require the function f that we construct to be consistent. (One could incorporate reflections as well if desired.)

Remark 1.1. The structure σ as described above contains the actual positions of the atoms, and ρ is the property value of σ. Instead, one could have σ be the positions of the atoms before some physical relaxation, while ρ is the property after this relaxation. We use such a strategy in section 4, where σ nominally lies on a specific lattice.

1.2. Representation using sums of separable functions. We will construct a function of the form

$$f([a_1, a_2, \dots]) = \sum_{l=1}^{r} s_l \prod_{i=1}^{\infty} f_i^l(a_i), \tag{1.2}$$

whose domain is the set of ordered lists of atoms. We call r the *separation rank*, following [4, 5]. The functions f_i^l and normalization coefficients s_l are both to be determined, as is the paradigm for *separated representations* [4, 5]. This paradigm is in contrast with tensor product bases, where the functions f_i^l are predetermined, and only the coefficients s_l are to be determined. In [3] a method was presented to construct such functions to solve regression or machine learning problems in high dimensions. In that method the functions f_i^l and coefficients s_l are determined by trying to minimize the least-squares error with the data, possibly subject to regularization.

To remove the infinite list/product over i, we note that there is no rule that we must use all input variables. Indeed, a great many physical properties are determined mostly by the local environment around each atom [1]. As such, we select d variables, where d is chosen a posteriori to maximize the predictive ability of our method (as in Tables 4.3 and 4.5). For notational convenience we choose to use the first d atoms in each list. We thus have

$$f([a_1, a_2, \dots]) := f([a_1, a_2, \dots, a_d]) = \sum_{l=1}^{r} s_l \prod_{i=1}^{d} f_i^l(a_i)\,. \tag{1.3}$$

To enforce the consistency condition, we will define an operator $\mathcal{C}$ that converts such a function f on the set of ordered lists of atoms to a consistent function $\mathcal{C}f$ on the set of structures. We then fit $\mathcal{C}f$ to the data, instead of f itself, and use $\mathcal{C}f$ to obtain predictions as well. This operator allows us to enforce consistency while working on the more tractable set of functions with ordered inputs. A similar idea was used in [5, 6] to incorporate the antisymmetry condition in quantum mechanics.

1.3. Summary of the remainder of the paper. In section 2 we develop the consistency operator $\mathcal{C}$. In section 3 we adapt the algorithm from [3] to include $\mathcal{C}$. In section 4 we specialize the method to the simpler case of Mo and Ta on a body-centered cubic (BCC) lattice in order to give a concrete example including numerical results.

1.4. Prior work. The general idea of using data to obtain an approximation for g is not new. For structures consisting of few (usually two) atom types on a fixed crystal lattice, the cluster expansion method has been used extensively (see, e.g., [9, 8, 1]). For example, in [9] it was used to describe the energy of formation of $\mathrm{Mo}_{1-x}\mathrm{Ta}_x$ alloys on a BCC lattice, where $\tilde{g}(\sigma)$ is obtained from accurate first-principles calculations within the framework of density functional theory. (We will consider this particular system and property within our method in section 4.) An explicit averaging over the lattice symmetry group is used to ensure that structures that are rotations or translations of one another produce the same prediction. The cluster expansions then provide a regression method for this averaged data. The weakness of the cluster expansion method is that it can only handle structures on a fixed lattice.

For structures with fixed atom types but general positions, methods have been developed based on neural networks [2, 7, 11]. The first step is to convert the structure into small number of coordinates, such as inter-atom distances and angles. These coordinates attempt to capture the physically relevant parameters that describe the structure. A neural network is then used to perform the regression with respect to these coordinates. The weaknesses of this approach are that the atom types are fixed and the conversion to a few coordinates is not done in a rotation- and translation-invariant way. Essentially, the approach is appropriate for testing perturbations of a given structure, but not for learning a wide variety of structures.

1.5. Alternative regression methods. Many classical regression methods (see, e.g., [12, 10]) do not make sense when the input data are structures. Several methods, such as kernel methods and support-vector machines, rely on a notion of distance between data points, which in our context means distance between structures. We can find no satisfactory definition for $\|\sigma_1 - \sigma_2\|$, and thus we believe such methods cannot be used. Similarly, linear models rely on the notion of inner product, but we can find no satisfactory definition for the inner product of two structures, or of a structure with some sort of dual element.

Other classical methods can be used to replace the function representation (1.3) and thus can be used within our framework with $\mathcal{C}$. The method must be able to accept an ordered list of atoms as input and have internal parameters that can be varied to obtain a (consistent) regression function. In particular, any method based on fitting with a combination of functions of single atoms should be compatible with our approach. This includes the additive model

$$f([a_1, a_2, \ldots, a_d]) = \sum_{i=1}^{d} f_i(a_i) \tag{1.4}$$

and neural networks.

1.6. Flaws and future work. We consider the work presented here as only the beginning of the development of this method. Some aspects have not yet been developed, and others are not satisfactory in their current form.

1. As noted in section 1.5, another regression method could be used in place of (1.3). It is unknown if some other regression method would be better for this application.
2. The conversion method, as described in section 2, includes decisions based on which atom is closest to another. If we move an atom continuously, the closest atom can change suddenly, which could lead to a discontinuity in our prediction. A version of the conversion method where near-ties are dealt with smoothly seems possible but has not yet been developed.
3. If the data $\tilde{g}(\sigma)$ has uncertainty, it could be given as a probability distribution, and the "vector-valued" version of [3] could be used to give the prediction as a probability distribution. Similarly, one may be able to use probability distributions for the positions of atoms in a structure. These ideas have not yet been developed.
4. As noted in Remark 2.6, a conversion method for amorphous materials seems possible but has not yet been developed.
5. In section 3.6 we discuss the choices for the functions of a single atom, but we have not actually tried any of them.
6. The numerical results in section 4.5 are for structures lying on a fixed lattice. In this problem, our numerical results for a fixed lattice are not competitive with existing methods based on cluster expansions. Note, however, that the fixed-lattice problem is not the most interesting application for our method.
7. The numerical results in section 4.5 show good approximation power and good prediction on average but poor prediction for the worst case. The cause of this behavior, and the related issue of how to sample the set of all structures, requires further study.

2. Consistency. In this section we develop a definition and method to compute the consistency operator $\mathcal{C}$. We do not directly construct $\mathcal{C}$ or $\mathcal{C}f$, but instead produce a method to compute $\mathcal{C}f(\sigma)$ for any σ. The basis of the method is an algorithm for

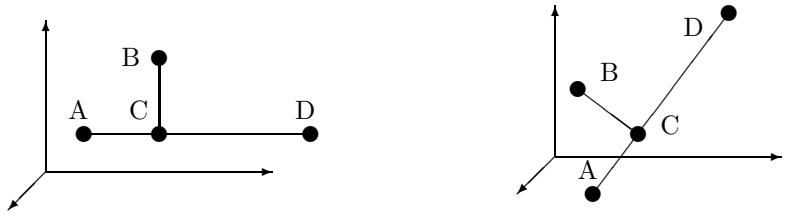

FIG. 2.1. *Two equivalent "toy" structures. The lines connecting the atoms are for visual reference only.*

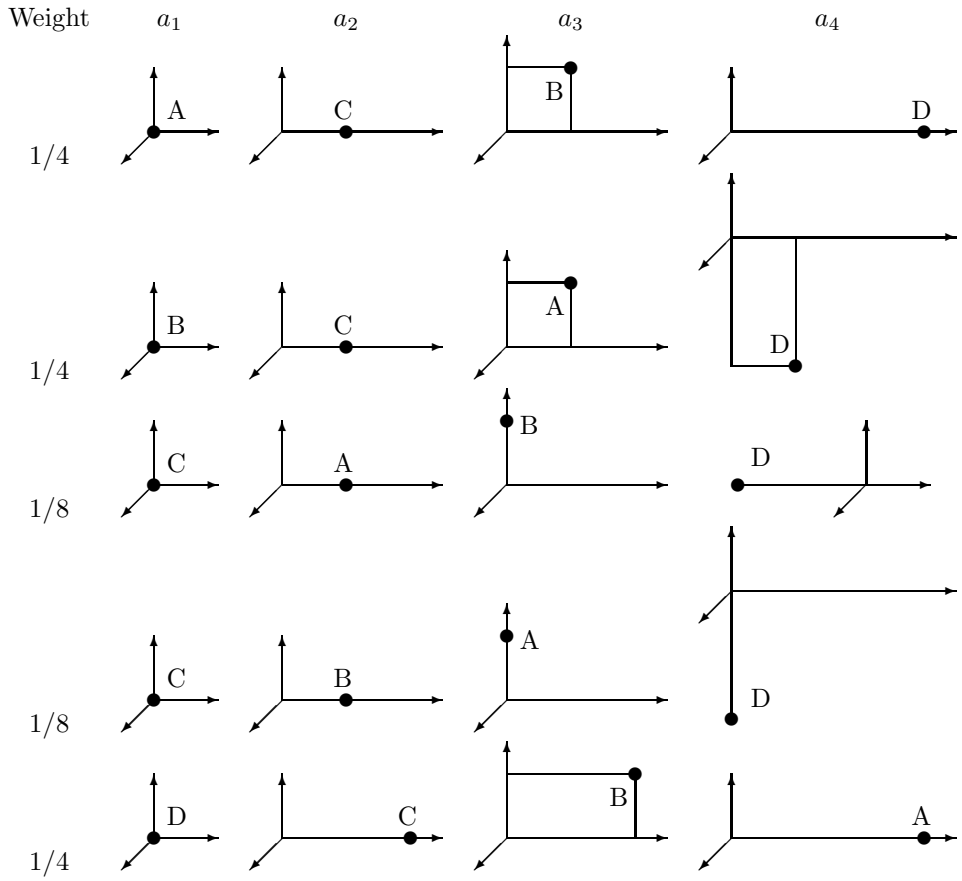

FIG. 2.2. *The weighted views of the structure(s) in Figure* 2.1. *Each row gives a view.*

converting a structure to a set of weighted points. We call these points "views" since they give the perspective that an atom in the structure would have of the entire structure. The weights represent the relative frequencies of the views. For illustrative purposes, in Figure 2.1 we give two equivalent "toy" structures, both of which map to the set of weighted views given in Figure 2.2.

DEFINITION 2.1 (view). *A view is an ordered list of atoms* $[a_1, a_2, \dots]$ *whose coordinates* $\mathbf{r}_j = (x_j, y_j, z_j)$ *satisfy the following orientation conditions*

1. $\mathbf{r}_1 = (0, 0, 0)$*;*
2. $x_2 > 0$*,* $y_2 = 0$*, and* $z_2 = 0$*; and*

3. *for the smallest i such that $y_i \neq 0$, we have $y_i > 0$ and $z_i = 0$;*

and satisfy the distance-ordering condition that either

1. $\|\mathbf{r}_i\| < \|\mathbf{r}_{i+1}\|$ *or*
2. $\|\mathbf{r}_i\| = \|\mathbf{r}_{i+1}\|$ *and either*
   (a) $x_i > x_{i+1}$ *or*
   (b) $x_i = x_{i+1}$ *and either*
      i. $y_i > y_{i+1}$ *or*
      ii. $y_i = y_{i+1}$ *and* $z_i > z_{i+1}$.

DEFINITION 2.2 (relative coordinate system). *A relative coordinate system, denoted* (**oxyz**)*, is specified in a global coordinate system by the vector* **o** *that specifies the origin and the orthonormal set of column vectors* $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ *that gives the x-, y-, and z-axes. To preserve orientation we require* $\mathbf{z} = \mathbf{x} \times \mathbf{y}$*, where $\times$ is the cross product.*

DEFINITION 2.3 (view of a structure). *We say that a view v is a view of a structure σ if there exists a relative coordinate system* (**oxyz**) *such that, after expressing all of the atoms in σ in* (**oxyz**)*, v lists those atoms exactly.*

2.1. Algorithm for constructing all views of a structure. We now present an algorithm for constructing all views of a given structure σ and accounting for their relative frequencies. The inputs of this algorithm are

- the finite set of atoms $\tilde{\sigma}$ that, together with the periodicity rule, specifies σ; and
- a function with arguments a_1 and $[a_1, \ldots, a_j]$ that returns the atom(s) in σ that are nearest to a_1, excluding $[a_1, \ldots, a_j]$.

(In implementations, the function in the second argument depends statically on $\tilde{\sigma}$ and the periodicity rule to define σ.) The output is a finite set

$$V_\sigma = \{(w, v)\} \tag{2.1}$$

with w positive numbers such that $\sum w = 1$, and v views of σ. The algorithm works by growing an empty view recursively into a valid view of σ. When there is more than one possibility for the next atom in the view, the view splits and divides its weight.

We initialize with the set

$$\{(1, ?, [])\}\,, \tag{2.2}$$

whose single element has weight 1, undefined coordinate system, and an empty atom list. This element $(1, ?, [])$ looks at the finite set of atoms $\tilde{\sigma}$ and notes the number of these atoms $n = |\tilde{\sigma}|$. It then splits itself into n elements, so that

$$(1, ?, []) \mapsto \left\{\left(1/n, (\mathbf{o}?)_k, [a_1^k]\right)\right\}_{k=1}^{n}\,, \tag{2.3}$$

where $\{a_1^k\}_{k=1}^n = \tilde{\sigma}$ and $(\mathbf{o}?)_k$ indicates the coordinate system with origin at $\mathbf{o} = \mathbf{r}_1^k$, but which does not yet have axes defined.

For each k we now have an element of the form $(w, (\mathbf{o}?), [a_1])$, which next tries to complete itself by defining its coordinate system and the remainder of its list. To do so it finds the atom(s) in σ that are closest to $\mathbf{o}$, i.e., minimizes $\|\mathbf{o} - \mathbf{r}_i\|$, excluding a_1 itself. If there is only one closest atom a_2, then

$$(w, (\mathbf{o}?), [a_1]) \mapsto (w, (\mathbf{ox}?), [a_1, a_2])\,, \tag{2.4}$$

where (**ox**?) indicates that the x-axis is now determined as $\mathbf{x} = (\mathbf{r}_2 - \mathbf{o})/\|\mathbf{r}_2 - \mathbf{o}\|$. If there are n atoms $\{a_2^k\}_{k=1}^n$ tied for closest, then $(w, (\mathbf{o}?), [a_1])$ splits itself into n

elements by

$$(w, (\mathbf{o}?), [a_1]) \mapsto \left\{(w/n, (\mathbf{ox}?)_k, [a_1, a_2^k])\right\}_{k=1}^{n} . \tag{2.5}$$

For each k we now have an element of the form $(w, (\mathbf{ox}?), [a_1, a_2])$, which next tries to complete itself. It finds the atom(s) in σ that are closest to $\mathbf{o}$, excluding a_1 and a_2. If there are ties, then from these it chooses the atom(s) with greatest x-coordinate in the system $(\mathbf{ox}?)$, or, equivalently, that are closest to a_2. If there is now only one atom a_3, then

$$(w, (\mathbf{ox}?), [a_1, a_2]) \mapsto (w, (\mathbf{oxyz}), [a_1, a_2, a_3]) , \tag{2.6}$$

where $(\mathbf{oxyz})$ indicates that the y-axis is now determined by the projection of the direction $\mathbf{r}_3 - \mathbf{o}$ orthogonal to $\mathbf{x}$, and from this the z-axis is determined by the handedness of the coordinate system via $\mathbf{z} = \mathbf{x} \times \mathbf{y}$. If there are n atoms $\{a_3^k\}_{k=1}^n$ still tied for closest, then

$$(w, (\mathbf{ox}?), [a_1, a_2]) \mapsto \left\{(w/n, (\mathbf{oxyz})_k, [a_1, a_2, a_3^k])\right\}_{k=1}^{n} . \tag{2.7}$$

Once the coordinate system is defined, the rule for selecting the next atom in the list is as follows: closest to the origin, with ties broken by the largest x-coordinate, with remaining ties broken by the largest y-coordinate, and remaining ties broken by the largest z-coordinate. The splitting of elements due to ties is finished once the coordinate system is fixed, since a full coordinate system allows us to break all ties. The coordinate system $(\mathbf{oxyz})$ is determined at the smallest j for which the set of points $\{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_j\}$ is not colinear and is fixed thereafter. There is an initial splitting (2.3) to determine a_1, a possible split due to ties at a_2, and a possible split due to ties at a_j, but no intermediate splits since a tie would contradict colinearity. Thus an element can split at most three times in its history, and the set of views is finite.

Finally, each element $(w, (\mathbf{oxyz}), [a_1, a_2, \ldots])$ expresses the atoms in its list in the local coordinate system $(\mathbf{oxyz})$, and then discards that system to become of the form (w, v), with v a view. Then we check whether any of the views are now identical, in which case we combine the elements and add their weights. Such duplicates can be caused by an inefficient choice of $\tilde{\sigma}$ or by additional (e.g., rotational) symmetries in σ. These final elements become the set V_σ.

In practice the algorithm terminates when the views reach length d or include all atoms of a finite σ. (If finite structures of different lengths are to be compared, null arguments can be used to fill all views to length d.) For an example of this algorithm applied to a particular structure, see section 4.3.

PROPOSITION 2.4. *V_σ contains all views of σ exactly once and no extra views.*

Proof. Since the views in V_σ were constructed from σ, they must all be views of σ. Since we combined duplicate views, each occurs only once. Since $\tilde{\sigma}$ tiles σ, all views of σ can be generated starting from $\tilde{\sigma}$. Since we allowed all choices for generating the lists of atoms consistent with the definition of a view, we must have generated all views. ∎

PROPOSITION 2.5. *The weight w associated to a view v in V_σ is independent of the way σ was specified via $\tilde{\sigma}$ and the periodicity vectors.*

Proof. Since $\tilde{\sigma}$ was used only in the first splitting (2.3), and the periodicity vectors are not explicitly used at all, we need only account for the effect of (2.3) on w. The idea of the argument is that a choice of $\tilde{\sigma}$ that is, e.g., two unit cells, will result

in twice as many elements with half the weight after the splitting (2.3), but these duplicates will be recombined in the final step. Consider the maximal translation group G of σ and some $\tilde{\sigma}_1$ that produces σ under G. For some other $\tilde{\sigma}$, take the group H generated by its periodicity vectors. The group H is a normal subgroup of G and has a quotient group $\tilde{G} = G/H$ with some number of elements n. Choosing a representative in G for each element in $\tilde{G}$, we can map an atom $a \in \tilde{\sigma}_1$ to n distinct elements in σ. By the definition of a quotient group, these elements are equivalent under H to n distinct elements of $\tilde{\sigma}$. Since composing $\tilde{G}$ with H generates all atoms in σ with no duplication, all elements of $\tilde{\sigma}$ can be generated this way. Thus $\tilde{\sigma}$ has an n-fold redundancy, which is removed when the views are consolidated. ∎

Remark 2.6. If one could obtain a meaningful set of weighted views for an amorphous material through some statistical method, then one could apply the rest of our method to those as well.

2.2. The consistency operator.

DEFINITION 2.7 (consistency operator). *For any function f on the set of ordered lists of atoms and finite or periodic structure σ, we define*

$$\mathcal{C}f(\sigma) = \sum_{(w,v)\in V_\sigma} w f(v)\,, \tag{2.8}$$

where V_σ is the set of weighted views constructed above.

PROPOSITION 2.8. *$\mathcal{C}f(\sigma)$ is consistent.*

Proof. By definition, views of a structure are independent of the global coordinate system in which σ sits, and thus $\mathcal{C}f(\sigma)$ is invariant under rotations and translations of σ. ∎

3. The algorithm. We now describe how to fit $\mathcal{C}f$ to the data D, with f of the form (1.3). The algorithm is based closely on the work in [3] but now includes the consistency operator $\mathcal{C}$.

3.1. Data-driven inner product, with consistency. Given a finite collection of data D from (1.1) we define a pseudo–inner product

$$\langle f, p\rangle_D = \frac{1}{N}\sum_{j=1}^{N} \mathcal{C}f(\sigma_j)\mathcal{C}p(\sigma_j)\,. \tag{3.1}$$

This is not a true inner product since for some choices of nonzero f we could have $\|f\|_D^2 = \langle f, f\rangle_D = 0$ depending on the σ in the data set. This definition allows us to take inner products with the data as well,

$$\langle D, f\rangle_D = \langle \{(\sigma_j, \rho_j)\}_{j=1}^N, f\rangle_D = \frac{1}{N}\sum_{j=1}^{N} \rho_j \mathcal{C}f(\sigma_j)\,, \tag{3.2}$$

and thus treat the data as some unknown function. The least-squares error with respect to this inner product is

$$\|D - f\|_D^2 = \frac{1}{N}\sum_{j=1}^{N} \left(\rho_j - \mathcal{C}f(\sigma_j)\right)^2\,. \tag{3.3}$$

For each σ_j in D, in section 2 we defined V_{σ_j} and used it to construct $\mathcal{C}f(\sigma_j)$ via (2.8). Using (2.8), the least-squares error (3.3) can be written

(3.4)
$$\frac{1}{N}\sum_{j=1}^{N}\left(\rho_j - \sum_{v\in V_{\sigma_j}} w_v^j f([a_1^{jv},\dots])\right)^2 = \frac{1}{N}\sum_{j=1}^{N}\left(\rho_j - \sum_{v\in V_{\sigma_j}} w_v^j \sum_{l=1}^{r} s_l \prod_{i=1}^{d} f_i^l(a_i^{jv})\right)^2 .$$

We will attempt to minimize this error.

3.2. Collapse to one-dimensional subproblems. We now assume that an initial f of the form (1.3) is given. We fix the components for all values of i but one, and so collapse to a one-dimensional (i.e., one-atom) problem. For ease of exposition we describe the case $i = 1$, and so fix f_i^l for $i > 1$. For all $j = 1,\dots,N$ and $l = 1,\dots,r$ we define the partial products

$$p_j^{v,l} = s_l \prod_{i=2}^{d} f_i^l(a_i^{jv}) \tag{3.5}$$

for all $v \in V_{\sigma_j}$. The least-squares error (3.3), as expanded out in (3.4), now collapses to

$$\frac{1}{N}\sum_{j=1}^{N}\left(\rho_j - \sum_{v\in V_{\sigma_j}} w_v^j \sum_{l=1}^{r} p_j^{v,l} f_1^l(a_i^{jv})\right)^2 . \tag{3.6}$$

To minimize (3.6) we must solve a one-dimensional least-squares problem for the r functions f_1^l.

3.3. One-dimensional linear least-squares. To represent the functions f_1^l, we choose M linearly independent functions $\{\phi_m\}_{m=1}^M$, so we can express $f_1^l = \sum_{m=1}^M c_m^l \phi_m$, where the c_m^l are real coefficients. We will discuss this choice in section 3.6. Expressed in terms of these coefficients, the error (3.6) becomes

$$\frac{1}{N}\sum_{j=1}^{N}\left(\rho_j - \sum_{v\in V_{\sigma_j}} w_v^j \sum_{l=1}^{r} p_j^{v,l} \sum_{m=1}^{M} c_m^l \phi_m(a_i^{jv})\right)^2 . \tag{3.7}$$

The coefficients c_m^l are the free parameters with respect to which we minimize (3.7).

Taking the gradient of (3.7) with respect to the c_m^l and setting it equal to zero produces the usual linear normal equations

$$\mathbf{Az} = \mathbf{b}\,. \tag{3.8}$$

The matrix $\mathbf{A}$ has entries defined by

$$\mathbf{A}(m,l;m',l') = \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{v\in V_{\sigma_j}} w_v^j p_j^{v,l}\phi_m(a_i^{jv})\right)\left(\sum_{v\in V_{\sigma_j}} w_v^j p_j^{v,l'}\phi_{m'}(a_i^{jv})\right), \tag{3.9}$$

with the combined index (m,l) acting as the row index and the combined index (m',l') acting as the column index. The vector $\mathbf{b}$ has entries defined by

$$b(m,l) = \frac{1}{N}\sum_{j=1}^{N}\rho_j \sum_{v\in V_{\sigma_j}} w_v^j p_j^{v,l}\phi_m(a_i^{jv})\,, \tag{3.10}$$

with the combined index (m, l) acting as the row index. Solving the linear system (3.8) using, e.g., the conjugate gradient method, yields a vector $\mathbf{z}$. Setting $c_m^l = z(m, l)$ then minimizes (3.7).

3.4. Iterative improvement. Since we can solve the one-dimensional subproblems, we can iteratively solve such problems to reduce the error (3.4). One strategy for ordering the iteration is to loop through the directions $i = 1, \ldots, d$. This alternating least-squares (ALS) approach is well known and was used in [3]. One then repeats this process and monitors the change in error to detect convergence. It is certainly possible to hit local minima, in which case one would need to restart with a different guess or increase r. Even when we approach the true minimum, we have no reason to expect any better than linear convergence.

3.5. Computational cost. The computational cost of this procedure depends on several parameters:

- r—the separation rank in (1.3).
- d—the number of variables used in (1.3).
- $N_{\mathcal{C}d}$—the number of data points, inflated by their number of views of length d. This number is bounded by the total number of views

$$N_{\mathcal{C}} = \sum_{j=1}^{N} |V_{\sigma_j}|\,, \tag{3.11}$$

  but we consolidate views of a structure that are identical when truncated to d sites.
- M—the number of functions $\{\phi_m\}_{m=1}^M$ in section 3.3.
- M_f—the cost to evaluate a single $f_i^l(a_i)$.
- K—the number of ALS iterations used in section 3.4.
- S—the number of conjugate gradient iterations needed to solve the system (3.8). In theory this could be as large as rM, but after a few ALS iterations we should have a very good starting guess, so S should become quite small.

The cost for computing all of the $p_j^{v,l}$ in (3.5) is $\mathcal{O}(rdN_{\mathcal{C}d}M_f)$. When we switch directions during the ALS, we do not need to recompute $p_j^{v,l}$ from scratch but can update it instead. For example, when switching from direction 1 to direction 2, we would multiply $p_j^{v,l}$ by $f_1^l(a_1^{jv})/f_2^l(a_2^{jv})$. Our incremental cost for computing all $p_j^{v,l}$ is thus $\mathcal{O}(rN_{\mathcal{C}d}M_f)$. Given the $p_j^{v,l}$, to compute all entries in $\mathbf{A}$ (3.9) costs $\mathcal{O}(r^2M^2N_{\mathcal{C}d})$, and to compute all entries in $\mathbf{b}$ (3.10) costs $\mathcal{O}(rMN_{\mathcal{C}d})$. The cost for solving the normal equations (3.8) using conjugate gradient is $\mathcal{O}(r^2M^2S)$. Assuming $M_f < rM^2$, our combined cost is thus

$$\mathcal{O}\left(Kdr^2M^2(N_{\mathcal{C}d} + S)\right). \tag{3.12}$$

3.6. The single-atom function. In section 3.3 we deferred the issue of choosing the basis $\{\phi_m\}_{m=1}^M$ to use for the single-atom functions f_i^l. The input a is an atom, consisting of a species t and location $\mathbf{r}$ in the local coordinate system. The species variable t is discrete, so we can span in that variable using the orthonormal basis of unit normal vectors $\{\mathbf{e}_n\}_{n=1}^T$, where T is the number of species considered. For each value of t, $f_i^l(t, \mathbf{r})$ can be an independent function of $\mathbf{r}$. The spanning set could depend on t, but for simplicity we assume that it does not, and is given by $\{\tilde{\phi}_m\}_{m=1}^{\tilde{M}}$.

The function is then given by

$$f_i^l((t,\mathbf{r})) = \sum_{n=1}^{T} \sum_{m=1}^{\tilde{M}} c_{nm} \mathbf{e}_n(t) \tilde{\phi}_m(\mathbf{r})\,. \tag{3.13}$$

The coefficients c_{nm} depend on i and l, and in principle $\tilde{M}$ and $\{\tilde{\phi}_m\}_{m=1}^{\tilde{M}}$ could as well. In terms of our formulation in section 3.3, we have mapped the index m to the tuple (n,m) and the number of elements M to $T\tilde{M}$.

The representation (3.13) still leaves quite a lot of freedom, from which we have little basis to decide. We instead sketch a few considerations and suggest likely candidates to try. When $i=1$ we notice that we always have $\mathbf{r}_1 = (0,0,0)$, so we should choose f_1^l to be constant in $\mathbf{r}$. When $i=2$, we have $\mathbf{r}_2 = (x,0,0)$ with $x>0$, so f_2^l should only depend on x in space. When $i=3$, we have $\mathbf{r}_3 = (x,y,0)$ with $y \geq 0$, so f_3^l should depend only on x and y in space. For $i>3$, $\mathbf{r}_i$ depends on all three coordinates. The likely candidates for $\{\tilde{\phi}_m\}_{m=1}^{\tilde{M}}$ are low-degree spaces of spherical harmonics. For large i, we expect a_i to have a simpler interaction with a_1, so we can perhaps decrease the degree of the representation.

3.7. Avoiding overfitting. A regression algorithm is supposed to both fit the available data and provide useful predictions for other inputs. *Overfitting* is when the regression method improves the fitting error (often by a small amount) at the expense of degradation of the predictive value (often by a large amount). As an extreme example of overfitting in one dimension, one could use the function

$$f(x) = \sum_{j=1}^{N} \rho_j \exp(-c(x-x_j)^2) \tag{3.14}$$

to represent the data $\{(x_j,\rho_j)\}_j$. In the limit $c\to\infty$, this function would match the given data exactly but predict 0 for other values of x.

There are two standard approaches for avoiding overfitting. In parametric methods, f is constrained to be of a certain form, with only a few free parameters to determine. If the model for f is correct, then this approach will work very well, but if the model is incorrect, then it may not be able to fit the data sufficiently well. In nonparametric methods, f is chosen from a much wider class of functions, with some mechanism encouraging the choice of a nice (smooth) function. The wider class of functions allows the method to fit the data well, while the "regularization" mechanism attempts to prevent overfitting. If these two interests can be balanced in the method, then it can indeed both fit the data and provide useful predictions. The amount of regularization is parametrized by some $\lambda>0$, which in general must be determined empirically. One common strategy is to split the data into two parts. Using the first part, generally 2/3 of the data, one runs the algorithm using several different values of λ. One then tests the resulting regression functions on the remaining 1/3 of the data to determine which value of λ performed best. Using this value of λ, one then runs the algorithm again using all of the data.

With respect to r, we recommend a parametric approach, i.e., keep r small. For the one-directional functions $f_i^l(a)$ we recommend a nonparametric approach. The basic approach is to penalize by the square of the L^2 norm of the gradient. (The approach described here is an improvement over that in [3].) For each l, we formally have

$$\lambda \int \sum_{t=1}^{T} \left(\nabla_{(t,\mathbf{r})} f_1^l((t,\mathbf{r}))\right)^2 d\mathbf{r}\,. \tag{3.15}$$

Substituting the form (3.13) in for f_1^l, our penalty becomes

$$\lambda \int \sum_{t=1}^{T} \left(\nabla_{(t,\mathbf{r})} \sum_{n=1}^{T} \sum_{m=1}^{\tilde{M}} c_{nm} \mathbf{e}_n(t) \tilde{\phi}_m(\mathbf{r}) \right)^2 d\mathbf{r} \,. \tag{3.16}$$

Differentiating with respect to c_{pq} yields

$$2\lambda \sum_{n=1}^{T} \sum_{m=1}^{\tilde{M}} c_{nm} \int \sum_{t=1}^{T} \left[\left(\nabla_{(t,\mathbf{r})} \mathbf{e}_n(t) \tilde{\phi}_m(\mathbf{r}) \right) \cdot \left(\nabla_{(t,\mathbf{r})} \mathbf{e}_p(t) \tilde{\phi}_q(\mathbf{r}) \right) \right] d\mathbf{r} \,. \tag{3.17}$$

Splitting off the discrete part of the gradient, we have

$$2\lambda \sum_{n=1}^{T} \sum_{m=1}^{\tilde{M}} c_{nm} \left[\sum_{t=1}^{T} \nabla_t \mathbf{e}_n(t) \nabla_t \mathbf{e}_p(t) \int \tilde{\phi}_m(\mathbf{r}) \tilde{\phi}_q(\mathbf{r}) d\mathbf{r} + \delta_{np} \int \nabla_{\mathbf{r}} \tilde{\phi}_m(\mathbf{r}) \cdot \nabla_{\mathbf{r}} \tilde{\phi}_q(\mathbf{r}) d\mathbf{r} \right] , \tag{3.18}$$

where δ_{np} is the Kronecker delta. We define

$$\nabla_t \mathbf{v}(t) = \mu \left(v(t) - \frac{1}{T} \sum_{k=1}^{T} v(k) \right) \tag{3.19}$$

for some positive scalar μ that determines the weight that this discrete gradient is given relative to the continuous gradient. As with λ, the parameter μ must usually be determined empirically. We then have

$$2\lambda \sum_{n=1}^{T} \sum_{m=1}^{\tilde{M}} c_{nm} \left[\mu \left(\delta_{np} - \frac{1}{T} \right) \int \tilde{\phi}_m(\mathbf{r}) \tilde{\phi}_q(\mathbf{r}) d\mathbf{r} + \delta_{np} \int \nabla_{\mathbf{r}} \tilde{\phi}_m(\mathbf{r}) \cdot \nabla_{\mathbf{r}} \tilde{\phi}_q(\mathbf{r}) d\mathbf{r} \right] . \tag{3.20}$$

To minimize (3.7) + (3.15), the matrix in the normal equations (3.8) is modified by adding a matrix to the diagonal ($l = l'$) blocks. In (3.9) we defined $\mathbf{A}$ using a single m index to index the basis, but now we use the double index nm. In terms of this index, we add the matrix with $(nm, n'm')$ entry given by

$$2\lambda \left[\mu \left(\delta_{nn'} - \frac{1}{T} \right) \int \tilde{\phi}_m(\mathbf{r}) \tilde{\phi}_{m'}(\mathbf{r}) d\mathbf{r} + \delta_{nn'} \int \nabla_{\mathbf{r}} \tilde{\phi}_m(\mathbf{r}) \cdot \nabla_{\mathbf{r}} \tilde{\phi}_{m'}(\mathbf{r}) d\mathbf{r} \right] . \tag{3.21}$$

In the limit $\lambda \to \infty$ this additional matrix forces f_1^l to be constant.

4. Example: Mo and Ta on a BCC lattice. In this section we consider a simpler version of our general problem. This example will allow us to clarify several concepts by giving concrete realizations. It will also allow us to present numerical experiments with real data.

4.1. The BCC lattice and its symmetry group. The BCC lattice is represented as an infinite number of sites in three dimensions with all coordinates either even or odd, i.e., with locations

$$\{(2i, 2j, 2k)\}_{i,j,k} \cup \{(2i+1, 2j+1, 2k+1)\}_{i,j,k} \quad \text{for all } i, j, k \in \mathbf{Z}. \tag{4.1}$$

In the BCC lattice all sites are equivalent, so the symmetry group includes translations of any site to another. We can account for these translations by considering the site that is translated to the origin. The remaining rotational group for the BCC lattice contains 24 elements. Normally these are counted by noting the six possible directions to which the x-axis can be rotated and then the four possible directions to which the y-axis can be rotated once the x-axis is fixed. An alternative way of counting is to note the eight possible locations to which $(1,1,1)$ can be rotated and then the three possible locations to which $(-1,1,1)$ can be rotated once $(1,1,1)$ is fixed. Specifically, $(1,1,1)$ can be rotated to the locations

$$\begin{aligned}\{&(1,1,1),(-1,1,1)(1,-1,1),(1,1,-1),(-1,-1,1),\\ &(-1,1,-1),(1,-1,-1),(-1,-1,-1)\}.\end{aligned} \tag{4.2}$$

If, e.g., $(1,1,1)$ is left fixed at $(1,1,1)$, then $(-1,1,1)$ can be rotated to the locations

$$\{(-1,1,1),(1,-1,1),(1,1,-1)\}. \tag{4.3}$$

In section 4.3 we show how to construct all of the views of a structure on a BCC lattice. This construction will illustrate how the consistency operator captures the lattice symmetries. It is the alternative way of counting above that will appear and account for the $8 \times 3 = 6 \times 4 = 24$ elements in the rotation group of the BCC lattice.

4.2. Structures composed of Mo and Ta. We consider structures consisting of Molybdenum (Mo) and Tantalum (Ta). In constructing f we take the atoms to be located on the BCC lattice, even if the physical property that we are fitting is computed after strain relaxation. We assume that all structures are periodic, but we do not assume that they have the same period or periods that are multiples of each other.

One such structure σ is given by the atoms

$$\tilde{\sigma} = \{(\text{Mo},(0,0,0)),(\text{Mo},(1,1,1)),(\text{Ta},(0,0,2)),(\text{Ta},(1,1,3))\} \tag{4.4}$$

and the periodicity defined by the vectors

$$\{(2,0,0),(0,2,0),(0,0,4)\}. \tag{4.5}$$

From this we may determine the atom type at any location. For example, there will be an atom of Mo at any site having the form

$$(1,1,1)+i(2,0,0)+j(0,2,0)+k(0,0,4) \quad \text{for any } i,j,k \in \mathbf{Z} \tag{4.6}$$

since $(1,1,1)$ contains Mo.

4.3. Construction of all weighted views. In this section we describe how to construct the set V_σ using the algorithm in section 2.1 for the structure σ specified by (4.4) and (4.5).

Since there are $n = 4$ elements in $\tilde{\sigma}$, our first splitting (2.3) yields

$$\begin{aligned}\{&(1/4,((0,0,0)?),[(\text{Mo},(0,0,0))]),(1/4,((1,1,1)?),[(\text{Mo},(1,1,1))]),\\ &(1/4,((0,0,2)?),[(\text{Ta},(0,0,2))]),(1/4,((1,1,3)?),[(\text{Ta},(1,1,3))])\}.\end{aligned} \tag{4.7}$$

Our initial splitting (4.7) accounts for the translational symmetry in the BCC lattice for our given structure, since taking any other atom to the origin would result in a duplicate.

For this example, we choose to follow the first element in (4.7), so we now fix the origin at $\mathbf{o} = (0,0,0)$ and the first list element $a_1 = (\mathrm{Mo}, (0,0,0))$. The closest atoms to the origin in σ are

$$\begin{aligned}(4.8)\qquad &\{(\mathrm{Mo},(1,1,1)),(\mathrm{Mo},(-1,1,1)),(\mathrm{Mo},(1,-1,1)),(\mathrm{Ta},(1,1,-1)),\\ &(\mathrm{Mo},(-1,-1,1)),(\mathrm{Ta},(-1,1,-1)),(\mathrm{Ta},(1,-1,-1)),(\mathrm{Ta},(-1,-1,-1))\}.\end{aligned}$$

Our second splitting (2.5) thus yields

$$\begin{aligned}(4.9)\qquad &\{(1/32,(\mathbf{o}(1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})?),[a_1,(\mathrm{Mo},(1,1,1))]),\\ &(1/32,(\mathbf{o}(-1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})?),[a_1,(\mathrm{Mo},(-1,1,1))]),\\ &(1/32,(\mathbf{o}(1/\sqrt{3},-1/\sqrt{3},1/\sqrt{3})?),[a_1,(\mathrm{Mo},(1,-1,1))]),\\ &(1/32,(\mathbf{o}(1/\sqrt{3},1/\sqrt{3},-1/\sqrt{3})?),[a_1,(\mathrm{Ta},(1,1,-1))]),\\ &(1/32,(\mathbf{o}(-1/\sqrt{3},-1/\sqrt{3},1/\sqrt{3})?),[a_1,(\mathrm{Mo},(-1,-1,1))]),\\ &(1/32,(\mathbf{o}(-1/\sqrt{3},1/\sqrt{3},-1/\sqrt{3})?),[a_1,(\mathrm{Ta},(-1,1,-1))]),\\ &(1/32,(\mathbf{o}(1/\sqrt{3},-1/\sqrt{3},-1/\sqrt{3})?),[a_1,(\mathrm{Ta},(1,-1,-1))]),\\ &(1/32,(\mathbf{o}(-1/\sqrt{3},-1/\sqrt{3},-1/\sqrt{3})?),[a_1,(\mathrm{Ta},(-1,-1,-1))])\},\end{aligned}$$

where we indicate the x-axis by the unit vector in its direction. The locations that appear in (4.9) are the eight possible locations to which $(1,1,1)$ could be rotated, as given in (4.2). Thus we can see a portion of the rotation group of the BCC lattice.

We now choose to follow the first element in (4.9) and so can fix the x-axis $\mathbf{x} = (1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})$ and the second atom $a_2 = (\mathrm{Mo},(1,1,1))$. The closest atoms to the origin are

$$\begin{aligned}(4.10)\qquad &\{(\mathrm{Mo},(-1,1,1)),(\mathrm{Mo},(1,-1,1)),(\mathrm{Ta},(1,1,-1)),(\mathrm{Mo},(-1,-1,1)),\\ &(\mathrm{Ta},(-1,1,-1)),(\mathrm{Ta},(1,-1,-1)),(\mathrm{Ta},(-1,-1,-1))\}.\end{aligned}$$

To break the tie, we find the largest x-coordinate along the x-axis $(1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})$, and find that

$$(4.11)\qquad \{(\mathrm{Mo},(-1,1,1)),(\mathrm{Mo},(1,-1,1)),(\mathrm{Ta},(1,1,-1))\}$$

are still tied, with x-coordinates of $1/\sqrt{3}$. Our third splitting (2.7) thus yields

$$\begin{aligned}(4.12)\qquad &\{(1/96,(\mathbf{ox}(-2/\sqrt{6},1/\sqrt{6},1/\sqrt{6})(0,-1/\sqrt{2},1/\sqrt{2})),[a_1,a_2,(\mathrm{Mo},(-1,1,1))]),\\ &(1/96,(\mathbf{ox}(1/\sqrt{6},-2/\sqrt{6},1/\sqrt{6})(-1/\sqrt{2},0,1/\sqrt{2})),[a_1,a_2,(\mathrm{Mo},(1,-1,1))]),\\ &(1/96,(\mathbf{ox}(1/\sqrt{6},1/\sqrt{6},-2/\sqrt{6})(-1/\sqrt{2},1/\sqrt{2},0)),[a_1,a_2,(\mathrm{Ta},(1,1,-1))])\}.\end{aligned}$$

To compute the y-axis we orthogonalized to the x-axis, so for $(-1,1,1)$ for example, we compute

$$\begin{aligned}(4.13)\qquad &(\mathbf{r}_3-\mathbf{r}_1)-\left\langle(\mathbf{r}_3-\mathbf{r}_1),\frac{\mathbf{r}_2-\mathbf{r}_1}{\|\mathbf{r}_2-\mathbf{r}_1\|}\right\rangle\frac{\mathbf{r}_2-\mathbf{r}_1}{\|\mathbf{r}_2-\mathbf{r}_1\|}\\ &\quad=(-1,1,1)-\left\langle(-1,1,1),(1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})\right\rangle(1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})\\ &\quad=(-1,1,1)-(1/3,1/3,1/3)=(-4/3,2/3,2/3)\,,\end{aligned}$$

and then normalize by $\sqrt{3/8}$ to obtain $\mathbf{y} = (-2/\sqrt{6}, 1/\sqrt{6}, 1/\sqrt{6})$. To compute the z-axis we use the x- and y-axes and the handedness rule, so by using the cross product we obtain

(4.14)
$$\mathbf{z} = \mathbf{x}\times\mathbf{y} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ -2/\sqrt{6} & 1/\sqrt{6} & 1/\sqrt{6} \end{vmatrix} = 0\mathbf{i} - (1/\sqrt{2})\mathbf{j} + (1/\sqrt{2})\mathbf{k} = (0, -1/\sqrt{2}, 1/\sqrt{2}).$$

The locations that appear in (4.12) are the three possible locations to which $(-1, 1, 1)$ could be rotated, as given in (4.3). Thus we see the second portion of the rotation group of the BCC lattice, and we have accounted for all 24 rotations.

We now choose to follow the first element. We fix the y-axis $\mathbf{y} = (-2/\sqrt{6}, 1/\sqrt{6}, 1/\sqrt{6})$, the z-axis $\mathbf{z} = (0, -1/\sqrt{2}, 1/\sqrt{2})$, and the third atom $a_3 = (\text{Mo}, (-1, 1, 1))$. The closest atoms to the origin are

$$\begin{aligned} &\{(\text{Mo}, (1, -1, 1)), (\text{Ta}, (1, 1, -1)), (\text{Mo}, (-1, -1, 1)), \\ &(\text{Ta}, (-1, 1, -1)), (\text{Ta}, (1, -1, -1)), (\text{Ta}, (-1, -1, -1))\}, \end{aligned} \tag{4.15}$$

and of these, the largest x-coordinates are $1/\sqrt{3}$, held by

$$\{(\text{Mo}, (1, -1, 1)), (\text{Ta}, (1, 1, -1))\}. \tag{4.16}$$

With respect to $\mathbf{y}$ these atoms both have coordinate $-2/\sqrt{6}$ and so remain tied. With respect to $\mathbf{z}$, however, $(\text{Mo}, (1, -1, 1))$ has the larger coordinate of $\sqrt{2}$. Our fourth step is thus

$$(1/96, (\mathbf{oxyz}), [a_1, a_2, a_3]) \mapsto (1/96, (\mathbf{oxyz}), [a_1, a_2, a_3, (\text{Mo}, (1, -1, 1))]). \tag{4.17}$$

At this point we continue the process to define the list, and we have no more splitting. Due to the symmetries in this example, the weight tells us that we have 96 total elements, which is 24, the size of the rotation group of the lattice, times 4, the size of the specification $\tilde{\sigma}$ in (4.4). In each element the locations of the atoms are then expressed in the relative coordinate systems. Finally, any duplicate views are consolidated, and we obtain V_σ.

4.4. The single-atom functions and avoiding overfitting. Since geometrically all views of a structure on a BCC lattice are the same, for any fixed i the a_i from all views have the same $\mathbf{r}_i$. Thus the functions f_i^l should depend only on the species type and can be considered as vectors of length T, where T is the number of species considered.

The method to reduce overfitting from section 3.7 also simplifies considerably. The entries (3.21) in the matrix added to the diagonal blocks of $\mathbf{A}$ become

$$2\lambda\mu\left(\delta_{nn'} - \frac{1}{T}\right), \tag{4.18}$$

and we can set $\mu = 1$. For our current example of Mo and Ta we have $T = 2$, so the matrix is

$$\lambda \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix}. \tag{4.19}$$

4.5. Numerical results. In this section we present numerical results from using our method on one data set. We caution the reader, however, that very little can be extrapolated from a single test such as this. It is not known, and is probably unknowable, how well this data set captures the true property function. If, for example, the structures are all very similar, then we may learn and predict well on this data set, but would do poorly on some dissimilar structure. On the other hand, if the structures contain independent information, then learning on some subset may not allow good predictions on the remaining structures, and so we may confuse poor performance of the method with inadequate data.

The data set consists of $N = 57$ structures on a BCC lattice, with atom types in $\{\mathrm{Mo}, \mathrm{Ta}\}$; it was produced for and used in [9]. The structures are given in the format in the example (4.4), (4.5). The largest structure has 16 sites and the longest periodicity vector is $(1, 1, 6)$. The distribution of structure sizes is [(1, 2), (2, 2), (3, 6), (4, 14), (5, 8), (7, 8), (8, 4), (9, 2), (10, 1), (12, 1), (13, 1), (14, 1), (16, 7)], in the format (size, count). The dependent variable ρ is the formation enthalpy, from which the long-range elastic interactions have been removed as in [9]. The formation enthalpy is the energy released when forming (or the energy required to form) a single compound containing both Mo and Ta starting from the pure compounds containing only Mo and only Ta. As a result, the structure of all Mo and the structure of all Ta have $\rho = 0$. The resulting ρ_j lie in the range $[-204.8, 0.0]$ and are given with one digit after the decimal. They have mean

$$\overline{\rho} = \frac{1}{N}\sum_{j=1}^{N} \rho_j = -119.16\,. \tag{4.20}$$

The "null" predictor for ρ_j is simply the mean $\overline{\rho}$. This predictor gives a mean squared error (MSE) of

$$\mathrm{MSE}_0 = \frac{1}{N}\sum_{j=1}^{N} (\rho_j - \overline{\rho})^2 = 2377.53\,, \tag{4.21}$$

and a maximum error

$$\mathrm{MAX}_0 = \max |\rho_j - \overline{\rho}| = 119.16\,. \tag{4.22}$$

To assess the error in our approximation f, we can measure

$$\text{absolute MSE} = \frac{1}{N}\sum_{j=1}^{N} (\rho_j - \mathcal{C}f(\sigma_j))^2\,, \tag{4.23}$$

$$\text{relative MSE} = \frac{1}{\mathrm{MSE}_0}\frac{1}{N}\sum_{j=1}^{N} (\rho_j - \mathcal{C}f(\sigma_j))^2\,, \tag{4.24}$$

$$\text{absolute maximum error} = \max_j |\rho_j - \mathcal{C}f(\sigma_j)|\,, \quad \text{and} \tag{4.25}$$

$$\text{relative maximum error} = \frac{1}{\mathrm{MAX}_0}\max_j |\rho_j - \mathcal{C}f(\sigma_j)|\,. \tag{4.26}$$

Since the data is given with one digit after the decimal, the smallest the absolute maximum error in the data could be expected to be on average is 0.05, which would give a relative maximum error of 4.2e-4. We can roughly calibrate the maximum error

TABLE 4.1

Comparison table for different ways of measuring error. We give the approximate data error and then a selection of values.

Absolute max error	0.05	0.01	0.1	0.5	1.0	5.0	10
Relative max error	4.2e-4	8.4e-5	8.4e-4	4.2e-3	8.4e-3	4.2e-2	8.4e-2
$-\log_{10}$(relative max error)	3.38	4.08	3.08	2.38	2.08	1.38	1.08
Absolute MSE	8.3e-4	3.3e-5	3.3e-3	8.3e-2	3.3e-1	8.3e+0	3.3e+1
Relative MSE	3.5e-7	1.4e-8	1.4e-6	3.5e-5	1.4e-4	3.5e-3	1.4e-2
$-\log_{10}$(relative MSE)	6.46	7.85	5.85	4.46	3.85	2.46	1.85

m and MSE s if we assume the error is uniformly distributed between $\pm m$. We would then have $s = \int_{-m}^{m} x^2 dx/(2m) = m^2/3$, and so absolute MSE 8.3e-4 and relative MSE 3.5e-7. These errors act as a baseline for our fitting errors, but the actual error in the data is likely higher. Since we are generally only interested in the order of magnitude of our errors, in the tables we will display $-\log_{10}$ of the relative errors, which roughly gives the number of correct digits. To aid the reader in mentally comparing these various errors, in Table 4.1 we give the equivalent numbers using the various measures.

We select some number of atoms d to use in our views. For each structure σ_j we generate V_{σ_j} and associate to it the property value ρ_j. We can then evaluate

$$\mathcal{C}f(\sigma_j) = \sum_{(w,v)\in V_{\sigma_j}} w \sum_{l=1}^{r} s_l \prod_{i=1}^{d} f_i^l(\mathbf{r}_i^v)\,. \tag{4.27}$$

Recall from section 3.5 that $N_{\mathcal{C}d}$ is the total number of views when truncated to d sites. We can compare this data quantity with the number of free parameters in our model (1.3). Each f_i^l has M parameters, but the normalization $\|f_i^l\| = 1$ means that only $M-1$ are free. The scalar s_l adds another free parameter, so the total number of free parameters in (1.3) is $r(d(M-1)+1)$. Since here we have $M = 2$, this simplifies to $r(d+1)$. We remark that since the free parameters interact nonlinearly, the number of actual degrees of freedom may be less than the number of free parameters.

We first test the approximation power of the method and report the results in Table 4.2. Since we are not testing predictive power, we turn off regularization by setting $\lambda = 0$. We present $d = 2$ and then the values of d that add the next complete shell of atoms. For each d we first report $N_{\mathcal{C}d}$ and the maximum (radial) distance $\|\mathbf{r}_d\|$ in such a view, both in absolute units and relative to the diagonal length $\sqrt{3}$ and side length 2 of the BCC unit cube. Then, for each $r = 1, \ldots, 7$ we report the number of free parameters and two measures of the fitting error. Since we are measuring only the approximation power of the method, we allow ourselves 10 tries using different random starting guesses and choose the result with the smallest MSE. As expected, the errors generally decrease with increasing r and d. There is a large improvement in performance as r changes from 2 to 3 at larger d, as well as a large improvement as d changes from 27 to 51 at larger r. By the lower right of Table 4.2 the approximation is more precise than the data itself.

We next test the predictive power of the method by training on a portion of the data and testing on the remaining data, and we report the results in Table 4.3. A table of this sort can be used to empirically determine appropriate d and r for a given problem. Since we have only $N = 57$ data points, we choose to train on $N - 1 = 56$ points and test on the remaining point. To reduce the influence of the random starting guess for f, we train five times with different f, select the one that performs best on the training data, and use that for testing. We obtain an MSE for

TABLE 4.2
Results on the approximation power of the method.

	d	2	9	15	27	51	59	65	89	113
	# shells		1	2	3	4	5	6	7	8
	$N_{\mathcal{C}d}$	181	779	1172	1365	1568	1568	1605	1605	1638
	$\|\mathbf{r}_d\|$	1.73	1.73	2.00	2.83	3.32	3.46	4.00	4.36	4.47
	$\|\mathbf{r}_d\|/\sqrt{3}$	1.00	1.00	1.15	1.63	1.91	2.00	2.31	2.52	2.58
r	$\|\mathbf{r}_d\|/2$	0.87	0.87	1.00	1.41	1.66	1.73	2.00	2.18	2.24
1	# parameters	3	10	16	28	52	60	66	90	114
	$-\log_{10}$(rel. MSE)	1.2	1.5	2.0	2.3	2.3	2.7	2.1	2.3	2.1
	$-\log_{10}$(rel. max)	0.6	0.6	0.8	1.1	1.2	1.2	0.8	1.0	0.8
2	# parameters	6	20	32	56	104	120	132	180	228
	$-\log_{10}$(rel. MSE)	1.2	1.9	2.7	3.0	3.5	3.8	3.6	4.2	4.0
	$-\log_{10}$(rel. max)	0.6	0.9	1.4	1.3	1.7	1.7	1.5	1.9	1.7
3	# parameters	9	30	48	84	156	180	198	270	342
	$-\log_{10}$(rel. MSE)	1.2	2.0	2.9	3.5	5.3	4.8	6.8	8.8	10.9
	$-\log_{10}$(rel. max)	0.6	0.9	1.5	1.7	2.5	2.0	3.1	4.2	5.2
4	# parameters	12	40	64	112	208	240	264	360	456
	$-\log_{10}$(rel. MSE)	1.2	2.0	3.0	3.9	8.2	8.2	12.2	12.2	12.5
	$-\log_{10}$(rel. max)	0.6	0.9	1.4	1.7	3.7	3.7	6.0	5.9	6.0
5	# parameters	15	50	80	140	260	300	330	450	570
	$-\log_{10}$(rel. MSE)	1.2	2.0	3.3	3.9	9.9	11.3	13.1	13.0	13.0
	$-\log_{10}$(rel. max)	0.6	0.9	1.5	1.7	4.5	5.4	6.5	6.3	6.3
6	# parameters	18	60	96	168	312	360	396	540	684
	$-\log_{10}$(rel. MSE)	1.2	2.0	3.3	3.9	11.4	12.3	13.4	13.3	13.3
	$-\log_{10}$(rel. max)	0.6	0.9	1.5	1.7	5.4	5.9	6.6	6.5	6.5
7	# parameters	21	70	112	196	364	420	462	630	798
	$-\log_{10}$(rel. MSE)	1.2	2.0	3.3	3.9	11.9	12.9	13.5	13.5	13.4
	$-\log_{10}$(rel. max)	0.6	0.9	1.5	1.7	5.6	6.2	6.6	6.4	6.6

the training data and another for the testing data, both of which we divide by MSE_0 from (4.21), which uses the full data set. Similarly, we obtain maximum errors, which we divide by MAX_0 from (4.22). We perform this test for all 57 possibilities for the test point, and thus obtain the full leave-one-out cross-validation result possible for this data set. We then compute the mean, standard deviation, median, interquartile range, and maximum. We report the results for the cases $d = 51$, $r = 1, \ldots, 5$, and $\lambda = 0$, 10, 100, and 1000. For $\lambda = 0$ we see that the training error decreases with r, whereas the testing error remains large, especially when we consider the maximum over the 57 trials. The training errors were expected to increase as λ is increased, and do so. The testing errors were expected to decrease with λ initially, but eventually increase when λ is too large. We see this behavior for the $r = 1, 2, 3$ cases, whereas for $r = 4, 5$ the error may still be decreasing.

4.5.1. Assessment. For applications, the most important test of the method is its ability to predict. In the results in Table 4.3 the mean and median statistics for the testing error are much better than the maximum statistic. The predictions for some structures are unacceptably large. For comparison, we applied the cluster expansion method of [9] to our data set and obtained the cross-validation results on the left side of Table 4.4. At this level of the maximum statistic our method is not competitive with existing methods based on cluster expansions. This assessment should be tempered by the fact that the cluster expansion methods require a fixed lattice, whereas our methods do not.

At present, we cannot adequately explain why some structures are poorly predicted, but we can suggest some possibilities. First, it may be that some specific struc-

TABLE 4.3

Analysis of the leave-one-out cross-validation results for $d = 51$. We give the mean (with standard deviation), median (with interquartile range), and maximum of the relative MSE and relative maximum errors. We display $-\log_{10}$ of the value, which gives the number of correct digits.

			$\lambda = 0$		$\lambda = 10.0$		$\lambda = 100.0$		$\lambda = 1000.0$	
r	Error	Stat	Train	Test	Train	Test	Train	Test	Train	Test
1	MSE	mean	2.3(2.6)	1.0(0.6)	2.2(2.5)	1.1(0.6)	2.1(2.4)	1.1(0.7)	0.7(1.8)	0.5(0.3)
		med	2.3(2.7)	1.8(1.0)	2.3(2.4)	1.8(1.4)	2.2(2.6)	1.8(1.4)	0.7(1.9)	0.9(0.4)
		max	1.8	−0.2	1.8	−0.2	1.7	−0.0	0.7	−0.5
	MAX	mean	1.1(1.4)	1.1(1.0)	1.0(1.3)	1.1(1.1)	1.0(1.3)	1.1(1.1)	0.2(1.6)	0.7(0.8)
		med	1.1(1.5)	1.3(0.9)	1.0(1.2)	1.3(1.2)	1.1(1.5)	1.3(1.3)	0.2(1.4)	0.9(0.7)
		max	0.7	0.3	0.6	0.3	0.5	0.4	0.2	0.1
2	MSE	mean	3.4(3.8)	0.6(-0.1)	2.8(3.4)	1.4(1.1)	2.5(3.3)	1.7(1.5)	1.8(2.3)	0.9(0.6)
		med	3.4(3.7)	1.9(0.9)	2.8(3.3)	2.0(1.4)	2.5(3.2)	2.0(1.8)	1.8(2.3)	1.6(1.0)
		max	3.1	−1.0	2.6	0.4	2.4	0.8	1.4	−0.3
	MAX	mean	1.6(2.2)	1.0(0.8)	1.3(2.1)	1.2(1.2)	1.2(2.0)	1.4(1.5)	0.9(1.4)	1.0(1.0)
		med	1.6(2.1)	1.3(0.9)	1.4(2.0)	1.4(1.2)	1.2(1.9)	1.4(1.5)	0.9(1.4)	1.2(1.0)
		max	1.4	−0.1	1.1	0.6	1.1	0.8	0.5	0.3
3	MSE	mean	4.6(4.7)	0.4(-0.2)	3.2(3.9)	1.5(1.3)	2.7(3.6)	1.9(1.7)	2.4(3.2)	1.7(1.4)
		med	4.7(4.6)	1.4(1.1)	3.2(3.8)	1.9(1.7)	2.7(3.4)	2.4(1.9)	2.4(3.1)	2.3(1.7)
		max	4.0	−1.1	3.0	0.6	2.6	1.1	2.2	0.7
	MAX	mean	2.1(2.4)	0.9(0.7)	1.5(2.4)	1.3(1.3)	1.3(2.2)	1.5(1.5)	1.1(1.8)	1.4(1.4)
		med	2.1(2.2)	1.1(1.1)	1.5(2.2)	1.4(1.3)	1.3(2.2)	1.6(1.4)	1.1(1.6)	1.5(1.3)
		max	1.7	−0.1	1.4	0.7	1.2	0.9	1.0	0.7
4	MSE	mean	6.8(6.5)	0.9(0.6)	3.4(4.1)	1.5(1.3)	2.9(3.6)	1.7(1.4)	2.6(3.6)	1.8(1.4)
		med	7.6(7.0)	1.3(0.8)	3.4(3.9)	2.1(1.6)	2.9(3.5)	2.4(1.8)	2.6(3.4)	2.4(2.0)
		max	5.8	−0.2	3.2	0.6	2.7	0.7	2.5	0.6
	MAX	mean	3.3(3.2)	1.0(1.0)	1.6(2.4)	1.3(1.3)	1.4(2.2)	1.4(1.4)	1.3(2.1)	1.4(1.4)
		med	3.5(3.2)	1.1(0.8)	1.6(2.2)	1.4(1.3)	1.4(2.1)	1.6(1.4)	1.3(2.0)	1.6(1.5)
		max	2.6	0.3	1.5	0.7	1.3	0.8	1.1	0.7
5	MSE	mean	7.8(7.0)	0.6(0.1)	3.6(4.2)	1.6(1.4)	3.0(3.7)	1.5(1.2)	2.7(3.6)	1.9(1.7)
		med	10.6(9.5)	1.3(0.7)	3.6(4.0)	2.1(1.6)	3.0(3.7)	2.1(1.6)	2.7(3.5)	2.4(1.8)
		max	6.2	−0.7	3.3	0.7	2.8	0.4	2.6	0.8
	MAX	mean	4.1(3.5)	0.9(0.8)	1.7(2.4)	1.3(1.4)	1.5(2.4)	1.3(1.3)	1.3(2.2)	1.5(1.5)
		med	5.1(4.6)	1.0(0.8)	1.7(2.2)	1.5(1.3)	1.5(2.2)	1.4(1.3)	1.3(2.0)	1.6(1.4)
		max	2.7	0.0	1.5	0.7	1.4	0.6	1.2	0.8

TABLE 4.4

Leave-one-out cross-validation results using the cluster expansion method of [9]. *The "All Data" column tests on the full data set and the "Excluding Small" column requires that the* 24 *structures with size less than* 5 *always be in the training set.*

		All Data		Excluding Small	
Error	Stat	Train	Test	Train	Test
MSE	mean	2.7(4.5)	2.4(2.3)	2.7(4.3)	2.5(2.5)
	med	2.7(4.7)	2.5(2.3)	2.7(4.2)	2.6(2.3)
	max	2.7	1.5	2.7	1.9
MAX	mean	1.4(2.9)	1.6(1.8)	1.4(2.9)	1.7(1.9)
	med	1.4(3.2)	1.6(1.7)	1.4(3.2)	1.7(1.7)
	max	1.4	1.2	1.4	1.3

ture(s) contain independent information that cannot be predicted from the remaining structures. In examining the structures that were poorly predicted, we noticed they were often, but not always, very simple structures such as the solid Mo and solid Ta structures. It seems reasonable to require that these structures always be in the training set, but exactly which should be considered simple enough is unclear. This issue is related to the problem of how one should sample the set of all structures, which

TABLE 4.5

Leave-one-out cross-validation results similar to Table 4.3*, but requiring that the* 24 *structures with size less than* 5 *always be in the training set.*

			$\lambda = 0$		$\lambda = 10.0$		$\lambda = 100.0$		$\lambda = 1000.0$	
r	Error	Stat	Train	Test	Train	Test	Train	Test	Train	Test
1	MSE	mean	2.3(2.7)	1.6(1.4)	2.2(2.5)	1.8(1.7)	2.2(2.6)	1.7(1.5)	0.7(1.7)	0.8(0.7)
		med	2.3(2.7)	2.3(1.5)	2.3(2.5)	1.9(1.7)	2.2(2.7)	1.9(1.6)	0.7(1.8)	1.0(0.6)
		max	1.9	0.9	1.8	1.2	1.7	0.8	0.7	0.1
	MAX	mean	1.1(1.6)	1.3(1.4)	1.0(1.4)	1.4(1.5)	1.0(1.3)	1.3(1.4)	0.2(1.5)	0.9(1.0)
		med	1.1(1.7)	1.5(1.3)	1.1(1.2)	1.3(1.3)	1.1(1.6)	1.4(1.3)	0.2(1.3)	0.9(0.8)
		max	0.7	0.8	0.7	1.0	0.5	0.8	0.2	0.4
2	MSE	mean	3.4(3.8)	1.5(1.2)	2.8(3.5)	1.8(1.6)	2.5(3.3)	1.8(1.5)	1.8(2.3)	1.2(0.9)
		med	3.4(3.7)	2.2(1.7)	2.8(3.4)	2.3(1.8)	2.5(3.3)	2.1(1.9)	1.8(2.4)	1.7(1.4)
		max	3.1	0.7	2.6	0.8	2.4	0.8	1.4	0.3
	MAX	mean	1.6(2.2)	1.3(1.3)	1.3(2.2)	1.4(1.5)	1.2(2.0)	1.4(1.5)	0.9(1.6)	1.1(1.1)
		med	1.6(2.1)	1.5(1.3)	1.4(2.2)	1.5(1.4)	1.2(1.9)	1.4(1.5)	0.9(1.4)	1.2(1.2)
		max	1.4	0.7	1.2	0.8	1.1	0.8	0.7	0.5
3	MSE	mean	4.7(4.7)	1.4(1.0)	3.1(3.8)	1.7(1.4)	2.7(3.5)	2.0(1.8)	2.4(3.2)	1.9(1.6)
		med	4.8(4.7)	2.0(1.3)	3.2(3.7)	1.9(1.7)	2.7(3.4)	2.3(2.0)	2.4(3.0)	2.3(1.9)
		max	4.1	0.3	3.0	0.8	2.6	1.1	2.2	0.9
	MAX	mean	2.1(2.4)	1.2(1.2)	1.5(2.4)	1.4(1.4)	1.3(2.2)	1.5(1.6)	1.1(1.9)	1.5(1.5)
		med	2.2(2.3)	1.4(1.2)	1.5(2.2)	1.4(1.3)	1.3(2.0)	1.6(1.4)	1.1(1.6)	1.6(1.4)
		max	1.7	0.5	1.4	0.8	1.2	0.9	1.0	0.8
4	MSE	mean	6.8(6.4)	1.1(0.9)	3.4(4.1)	1.7(1.5)	2.9(3.6)	2.0(1.5)	2.6(3.7)	2.0(1.8)
		med	7.7(7.1)	1.8(1.2)	3.4(4.0)	2.3(1.6)	2.9(3.5)	2.6(2.1)	2.6(3.7)	2.4(2.0)
		max	5.8	0.3	3.2	0.8	2.7	0.7	2.5	1.1
	MAX	mean	3.3(3.1)	1.1(1.1)	1.6(2.4)	1.3(1.4)	1.4(2.2)	1.5(1.5)	1.3(2.1)	1.5(1.6)
		med	3.5(3.3)	1.3(1.1)	1.6(2.1)	1.5(1.3)	1.4(2.1)	1.7(1.6)	1.3(2.1)	1.6(1.5)
		max	2.6	0.5	1.5	0.8	1.3	0.8	1.1	0.9
5	MSE	mean	7.6(6.9)	1.0(0.9)	3.6(4.2)	1.6(1.4)	3.0(3.8)	1.8(1.4)	2.7(3.7)	2.2(2.2)
		med	10.7(9.8)	1.6(1.1)	3.6(4.1)	2.2(1.7)	3.0(3.7)	2.1(1.8)	2.7(3.6)	2.5(2.1)
		max	6.2	0.3	3.3	0.7	2.9	0.7	2.6	1.6
	MAX	mean	4.0(3.4)	1.0(1.1)	1.7(2.3)	1.3(1.4)	1.5(2.5)	1.4(1.4)	1.3(2.2)	1.6(1.7)
		med	5.1(4.8)	1.2(1.1)	1.6(2.2)	1.5(1.4)	1.5(2.3)	1.5(1.4)	1.3(2.2)	1.6(1.6)
		max	2.7	0.5	1.5	0.7	1.4	0.7	1.2	1.2

is beyond the scope of our work. We do present, in Table 4.5, the cross-validation results when one requires the 24 structures with size less than 5 to be always in the training set. For $r = 5$ and $\lambda = 1000$ we obtain results comparable to those for the cluster expansion method in Table 4.4.

Second, it may be that the predictions of the method have instability. We have only a rudimentary understanding of how sums of separable functions approximate other functions in general, so when we include the consistency operator and fit a poorly understood property function, our understanding is almost nil. We speculate that the sum of separable functions model may have too much freedom in trying to approximate the data, and some of these approximations will predict poorly. The regularization in section 3.7 helps significantly but is not sufficient, and more work is needed. We tested the additive model (1.4), which has much less freedom, but found it has essentially no approximation power and so was useless for prediction.

In other numerical experiments not reported in detail here, we attempted to use the method to obtain an optimal structure with respect to some property. A small data set was used to generate predictions for a large set of structures. Those structures with smallest predicted property value were then added to the data set and the process was repeated. We found the predictions to be insufficiently accurate to determine useful new data points in this way.

Acknowledgment. We would like to thank Mingwei Ding, who did research and software development for this project while a graduate student at Ohio University.

REFERENCES

[1] S. Barabesh, V. Blum, S. Müller, and A. Zunger, *Prediction of unusual stable ordered structures of Au-Pd alloys via a first-principles cluster expansion*, Phys. Rev. B (3), 74 (2006), 035108.

[2] J. Behler and M. Parrinello, *Generalized neural-network representation of high-dimensional potential-energy surfaces*, Phys. Rev. Lett., 98 (2007), 146401.

[3] G. Beylkin, J. Garcke, and M. J. Mohlenkamp, *Multivariate regression and machine learning with sums of separable functions*, SIAM J. Sci. Comput., 31 (2009), pp. 1840–1857.

[4] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Natl. Acad. Sci. USA, 99 (2002), pp. 10246–10251.

[5] G. Beylkin and M. J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.

[6] G. Beylkin, M. J. Mohlenkamp, and F. Pérez, *Approximating a wavefunction as an unconstrained sum of Slater determinants*, J. Math. Phys., 49 (2008), 032107.

[7] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, *Neural network models of potential energy surfaces*, J. Chem. Phys., 103 (1995), pp. 4129–4137.

[8] V. Blum, G. L. W. Hart, M. J. Walorski, and A. Zunger, *Using genetic algorithms to map first-principles results to model Hamiltonians: Applications to the generalized Ising model for alloys*, Phys. Rev. B (3), 72 (2005), 165113.

[9] V. Blum and A. Zunger, *Prediction of ordered structures in the bcc binary systems of Mo, Nb, Ta, and W from first-principles search of approximately* 3,000,000 *possible configurations*, Phys. Rev. B, 72 (2005), 020104.

[10] A. W. Bowman and A. Azzalini, *Applied Smoothing Techniques for Data Analysis*, Oxford University Press, London, 1997.

[11] S. Lorenz, M. Scheffler, and A. Gross, *Descriptions of surface chemical reactions using a neural network representation of the potential-energy surface*, Phys. Rev. B, 73 (2006), 115341.

[12] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*, John Wiley, New York, 1992.