Development and Application of Machine Learning Methods to Selected Problems of Theoretical Solid State Physics

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)

im Fach: Physik
Spezialisierung: Theoretische Physik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von

M. Sc. Benedikt Hoock

Gutachter/innen:

1. Prof. Dr. Claudia Draxl
2. Prof. Dr. Jan Vybíral
3. Prof. Dr. Ulf Leser

Erst gehören alle Gedanken der Liebe,
dann gehört alle Liebe den Gedanken.
— Albert Einstein

Für meine Eltern.

Summary

Big Data – the raw material of the 21st century – is exploited by methods of machine learning in many branches of scientific research. Also current computational material science already has shifted to the era of data-driven approaches and machine learning methods have proven as useful tools for the prediction of a large number of material properties. They may surrogate very effortful calculations based on density functional theory, provide a better understanding of known materials or even help to discover new materials. Here, an essential role is played by the *descriptor*, a desirably interpretable set of material parameters.

This PhD thesis develops an approach to find descriptors for periodic multi-component systems where also atomic disorder influences the physical characteristics. We process primary features of one-atom, two-atom and tetrahedron clusters by an averaging scheme and combine them further by simple algebraic operations. Several methods of compressed sensing are used to identify an appropriate descriptor out from the matrix of the candidates generated that way. Furthermore, we develop elaborate cross-validation based model selection strategies that may lead to more robust and ideally better generalizing descriptors. Additionally, we study several error measures which estimate the quality of the descriptors with respect to accuracy, complexity of their formulas and the capturing of disorder effects. These generally formulated methods were implemented in a partially parallelized Python program.

Actual learning tasks were studied on the problem of finding models for the lattice constant and the energy of mixing of group-IV ternary compounds in zincblende structure where an accuracy of 0.02 Å and 0.02 eV is reached. We explain the practical preparation steps of data acquisition, analysis and cleaning for the target properties and the primary features, and continue with extensive analyses and the parametrization of the developed methodology on this test case. As an additional application we predict lattice constants and band gaps of octet binary compounds. The respective descriptors are assessed quantitatively by the error measures and, finally, their physical meaning is discussed.

Zusammenfassung

Big Data – der Rohstoff des 21. Jahrhunderts – wird bereits mittels Methoden des maschinellen Lernens in vielen Bereichen der Wissenschaft zu Nutze gemacht. Mittlerweile befindet sich auch die rechnergestützte Festkörperphysik im Zeitalter datengetriebener Forschung und Methoden des maschinellen Lernens haben sich bereits als hilfreiche Werkzeuge zur Vorhersage einer Vielzahl von Materialeigenschaften erwiesen. Somit können aufwendige Berechnungen mittels Dichtefunktionaltheorie umgangen werden und bereits bekannte Materialien besser verstanden oder sogar neuartige entdeckt werden. Eine zentrale Rolle spielt dabei der Deskriptor, ein möglichst interpretierbarer Satz von Materialkenngrößen.

Diese Arbeit entwickelt einen Ansatz zur Auffindung von Deskriptoren für periodische Multikomponentensysteme, deren Eigenschaften auch durch atomare Unordnung beinflusst wird. Primäre Features von Einzel-, Paar- und Tetraederclustern werden über die Superzelle gemittelt und dann weiter algebraisch kombiniert. Aus der Matrix der so erzeugten Kandidaten wird mit verschiedenen Methoden der Dimensionalitätsreduktion ein geeigneter Deskriptor identifiziert. Weiterhin stellt diese Arbeit Strategien vor, wie hier bei der Modellfindung Kreuzvalidierung eingesetzt werden kann, sodass stabilere und idealerweise besser generalisierbare Deskriptoren gefunden werden können. Es werden außerdem verschiedene Fehlermaße untersucht, die die Qualität der Deskriptoren in Hinblick auf Genauigkeit, Komplexität der Formeln und Berücksichtung der Unordnung charakterisieren. Diese allgemein formulierten Methoden wurden in einer teilweise parallelisierten Python-Software implementiert.

Als konkrete Problemstellungen werden Modelle für die Gitterkonstante und die Mischenergie von ternären Gruppe-IV Zinkblende-Legierungen "gelernt", die eine Genauigkeit von 0.02 Å bzw. 0.02 eV erreichen. Die vorbereitenden Schritte der Datenbeschaffung, -analyse, und -bereinigung werden im Hinblick auf die Zielgrößen als auch auf die primären Features erläutert, sodass dann umfassende Analysen und die Parametrisierung der entwickelten Methodik an diesem Testdatensatz durchgeführt werden können. Als weitere Anwendung werden Gitterkonstante und Bandlücken von binären Oktett-Verbindungen vorhergesagt. Die jeweils ermittelten Deskriptoren werden mit den Fehlermaßen quantitativ bewertet und ihre physikalische Relevanz wird abschließend disktutiert.

Ich erkläre, dass ich die Dissertation selbständig und nur unter Verwendung der von mir gemäß § 7 Abs. 3 der Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 42/2018 am 11.07.2018 angegebenen Hilfsmittel angefertigt habe.

München, den 18. Februar 2022

Contents

Acronyms and Symbols

Acronym	Description
CV	cross-validation
DFT	density functional theory
HM	higher moment
KRR	Kernel Ridge Regression
LARS	least angles regression
LASSO	least absolute selection and shrinkage operator
LDA	local density approximation
LLSR	linear-least square regression
LPOCV	leave-p-out cross-validation
MAE	mean absolute error
maxAE	maximum absolute error
ML	machine learning
MP	matching pursuit
MSE	mean square error
OMP	orthogonal matching pursuit
RMSE	root mean square error
SC	"sanity check"

Symbol	Description
$\mathbf{c}$	coefficient vector of a linear model
$\mathscr{D}$	descriptor / model in general
E_{mix}	energy of mixing per atom of the data-set $\mathbf{T}$
$\text{err}^{\text{tr/te}}$	expected / mean training or test error
$\text{err}^{\text{tr/te}}(\mathbf{S}_{\text{tr}})$	individual training / test error
f_k	local primary feature
F_k	global primary feature
$\bar{F}^q$	*raw* higher moment (see Eq. 5.2)
$\tilde{F}^q$	*central* higher moment (see Eq. 5.3)
K	composition (x, y)
$\mathscr{L}$	learning method
$\mathscr{M}$	dimensionality reduction method
N_{f_k}	informational complexity (see Sec. 5.5.2)
N_{op}	operational complexity (see Sec. 5.5.2)
$\mathbf{O}$	*ab-initio* data set of the 82 octet binaries $\mathbf{O}$ of Ref. [1]
$\mathbf{P}$	target property
p	split proportion in LPOCV
$\mathbf{S}_{\text{ho}}$	hold-out set
$\mathscr{S}^{\text{all}}$	model selection strategy using the total data (see Sec. 5.4.1)
$\mathscr{S}^{\text{CV}}_{\text{tr}}, \mathscr{S}^{\text{CV}}_{\text{te}}$	CV strategies using training or test error (see Sec. 5.4.1)
$\mathbf{T}$	*ab-initio* data set of the group-IV zincblende ternary compounds (see Sec. 5.4.2)
$\mathscr{T}$	tier of feature space complexity
$U^{(i)}_{tr}, U^{(i)}_{te}$	test and training sets at CV iteration i
$\mathbf{X}$	data / input matrix of a feature space
$\mathbf{X}_{a,1}, \ldots, \mathbf{X}_{E,4}$	feature spaces for the learning tasks on $\mathbf{T}$ (see Sec. 8.1.1)

Chapter 1

Introduction and Motivation

The highly industrialized society relies on the steady discovery of new materials that are required to fulfill or improve very specific technological functionalities for sustainable energy supply, health applicances or IT hardware [2]. Solid state physics itself has laid the foundations to a rapid increase of computational power over the last 60 years which now allows for replacing expensive experiments of materials science by simulations. Here, in particular software based on density functional theory (DFT) [3, 4] is very successful such that today in every second terabytes of material data are produced around the globe by various such codes. That way, computational materials science already has entered the *era of big data* [5], facing the challenges of how to effectively store, organize and share the data [6]. But big data also offers the opportunity to be distilled to knowledge that allows for gaining a deeper understanding of existing materials and for even predicting new materials of tailored characteristics. For this, very effective methods of data analytics are necessary that use well-understood approaches of machine learning and adapt them to actual material science problems [7].

Over the last decade, progress has been made towards a data-driven materials' science, both by a variety of approaches and for a wide range of properties [7–9]. Some promininent examples here are the classification of crystal structures by a neural network [10], the generation of structure-energy-property landscapes for molecular crystals [11] or the prediction of electronic band gap energies and energies of formation for transparent conducting compounds [12]. They demonstrate that machine learned models may reach high accuracy in property prediction and material classification.

A critical role here plays the numeric representation of the materials: it carries information on their important characteristics in a more or - preferentially - less abstract way and is the base for the machine learning. An approach to generate *interpretable* material representations was recently developed in Refs. [1, 2, 13] that combines symbolic regression [14, 15] with compressed sensing [16, 17]. In these works, simple atomic and dimer features were combined algebraically to a huge data matrix of candidate expressions for a property prediction task on a data-set of octet binary semiconductors. An optimal *descriptor* - a low-dimensional linear combination of such expressions - was extracted by the compressed sensing methods LASSO+ℓ_0 and, as an improvement to that, SISSO.

This thesis directly continues with these works by addressing the following issues:

1. *How can complex multi-component systems effectively be represented by features in the descriptor approach?*
   So far, the descriptors were constructed for the octet binary materials that have a simple, two-atomic unit cell. Hence, a method has to be defined how to featurize materials with several atomic species and a large unit cell such that the algebraic combinations of primary features are still interpretable. In particular, effects of symmetrically invariant atomic arrangements have to be considered by an appropriate materials representation. This concerns also the determination of a set of physically relevant, domain-specific primary features.

2. *How can the methods of feature selection and descriptor assessment be improved?*
   In Refs. [1, 2], the descriptors were identified in a straight-forward manner using the total dataset. This is not optimal with respect to robustness and generalizability of the model and may be improved by selection strategies that exploit the data in a statistical way by cross-validation. Equally, the present scheme to *evaluate* a found descriptor by cross-validation has to be formulated more rigorously. Also there is need to define additional measures that allow to assess also the complexity of the symbolic expressions of the descriptor, and its ability to consider effects of disorder.

3. *What can be learned from applying the methodology to actual material science problems?*

   Generally formulated machine learning methods need to be confined and analyzed on specific and appropriate material test cases. These are an *ab-initio* data-set of group-IV ternaries, materials influenced both by composition and atomic arrangement, and the octet binaries of Ref. [1] for our research questions. We aim at finding descriptors for their lattice constant, energy of mixing and band gap that, at the same time, provide an accurate property prediction and are physically interpretable.

Working along these issues, the manuscript of this thesis is structured the following: in Chapters 2 and 3 we present the background in theory of statistical learning and theoretical solid state physics. An overview of current research on machine learning in computational materials science and the relevant, already existing practical and theoretical concepts are given in Chapter 4. In Chapter 5, we propose the developed methodological contributions, i.e. a feature engineering scheme, cross-validation based selection strategies and additional measures for descriptor quality. Essential aspects on the implementation of the developed Python code are explained in Chapter 6. Chapter 7 presents the steps of generating DFT data for the primary features and the target properties. Analyses of the methodology on actual learning tasks and the resulting descriptors are the topic of Chapter 8. Finally, Chapter 9 presents the conclusions and also limitations of this thesis, and gives an outlook for continuing research.

Theoretical Background Part I

Chapter 2

Background in Statistical Learning Methods

In this work, we apply machine learning methods to quantitatively predict materials properties. This chapter explains the baselines of the underlying mathematical methods, for which the term statistical learning *is more common. After a few introductory remarks and a differentiation of important terms in Sec. 2.1, Sec. 2.2 explains several* linear regression methods. *These are linear least square regression and some closely related extensions to it, as well as some algorithmic linear approaches. In Sec. 2.3 we address kernel ridge regression and symbolic regression, two methods that extend regression to account for non-linearities. Finally, Sec. 2.4 covers the important theoretical background on model evaluation, different error metrics and cross-validation.*

2.1 Introductory Remarks and Terminology

Machine Learning (ML) is currently booming in many fields of technology and science and comprises the wide field of computational methods to *infer knowlegde from data.* It aims at finding statistical models by algorithms based on training data that generalize to samples out of the training data. Examples of its various applications are the automatized recognition of hand-written text [18], the detection of prostate cancer in medical imaging [19], the training of a personalized spam filter [20] or the optimal prediction of a prize in stock trading [21].

The term "machine learning" itself goes back to the computer science pioneer Arthur Samuel at the end of the 1950s, after the earliest developments like the first artificial neural network had happened [22]. These had the main idea to mimic biological learning by experience in brains by computer algorithms, what essentially describes what machine learning is. Further historical landmarks are the research on Bayesian methods in the 1960s, the invention of error-back propagation for neural networks in the 1980s and the win of a machine learned program over human players in the game Go in 2015 [23]. The increased amount of data and the advances in CPU power after the turn of the millennium paved the way to the ongoing boom of machine learning.

Roughly, machine learning algorithms split into two branches [7]: (1) *Supervised learning* which is to derive a relation between the input data and a target. It requires a pairwise structure of the data, $\{\mathrm{x}_i, P_i\}$, where each point in the input space is equipped by a target value. Both *regression* (see below) and *classification* are part of this branch. (2) *Unsupervised learning* aims at exploring the structure hidden in the data, such as different clusters. Here, data comprises just the input points $\{x_i\}$, without a target.

There are several terms that are very closely related to machine learning where it is difficult to draw a hard line between them. *Statistical Learning* is used basically synonymeously with a somewhat stronger accent on the underlying mathematical methods, which is why we prefer it in this Chapter. *Artificial Intelligence* (AI) acts as a broader umbrella term [24] comprising any kind of mimicking intelligent behaviour by technology. It subsumes machine learning as the subcategory of intelligent algorithms. *Data Science* [25] puts the focus on all aspects of data analysis, in particular on data mining, i.e. to gain insight in typically huge data itself. It also includes hard tasks like data gathering and cleaning. *Compressed sensing* [16, 17] are techniques to acquire signals that have a sparse input in a large space of input basis. It is closely related to *dimensionality reduction*, the extraction of a low-dimensional, ideally meaningful representation of the data from a high-dimensional space [26].

2.2 Linear Regression Methods

In this work, our main interest is to find *linear* models $\mathbf{f}(\mathbf{X}) = \mathbf{X}\mathbf{c}$ that predict some target property $\mathbf{P} \in \mathbb{R}^N$. $\mathbf{X} \in \mathbb{R}^{N\times M}$ is called the *input* or *data matrix* and $\mathbf{c} \in \mathbb{R}^M$ is the *coefficient vector* which is to be determined. In general, it is not expected to find a model that does the prediction in an exact manner, so we are seeking for an approximation to $\mathbf{P}$ that is in some sense optimal. The M columns $\mathbf{x}_j$ of the input matrix $\mathbf{X}$ might contain different types of values, each one per material sample. In addition, we always will assume that $\mathbf{X}$ contains a "0th" column full of ones, i.e. $\mathbf{x}_0 = \mathbf{1}$ (hence actually $\mathbf{X} \in \mathbb{R}^{N\times(M+1)}$). The corresponding coefficient c_0 then expresses an absolute offset term, or equivalently, the bias of the linear model.

2.2.1 Linear Least Square Regression

A very common and simple approach to obtain a linear model, i.e. to determine the coefficients $\mathbf{c}$, is *linear least square regression* [27, 28]. This approach finds the $\mathbf{c}$ that minimize the sum of the squared residuals between the target $\mathbf{P}$ and the linear model $\mathbf{X}\mathbf{c}$. Formally, this means to solve

$$\text{argmin}_{\mathbf{c}} L(\mathbf{c}) = \text{argmin}_{\mathbf{c}} \left(\sum_{i=1}^{N} (P_i - \sum_{j=1}^{M} X_{i,j} \cdot c_j)^2 \right) = \text{argmin}_{\mathbf{c}} \|\mathbf{P} - \mathbf{X}\mathbf{c}\|_2^2 . \tag{2.1}$$

The argument of the minimization is the so-called *objective* or *cost function* $L(\mathbf{c})$, and the sum of squared residuals equivalently can be expressed as $\|\ldots\|_2$ using the Euclidean or ℓ_2-norm (cf. Eq. 2.7). It is straightforward to solve this problem by expressing $L(\mathbf{c})$ as

$$L(\mathbf{c}) = (\mathbf{P} - \mathbf{X}\mathbf{c})^{\mathrm{T}} (\mathbf{P} - \mathbf{X}\mathbf{c}). \tag{2.2}$$

Basic calculus gives the two conditions

$$\frac{\partial L}{\partial \mathbf{c}} = -2\mathbf{X}^{\mathrm{T}} (\mathbf{P} - \mathbf{X}\mathbf{c}) = 0 \tag{2.3}$$

and

$$\frac{\partial^2 L}{\partial \mathbf{c} \partial \mathbf{c}^T} = 2\mathbf{X}^{\mathrm{T}}\mathbf{X} > 0 \tag{2.4}$$

on the minimizing $\mathbf{c}$. The second is fulfilled if all the columns of $\mathbf{X}$ are linearly independent or, equivalently, if the *Gramian matrix* $\mathbf{G} = \mathbf{X}^\mathrm{T}\mathbf{X} \in \mathbb{R}^{M\times M}$ is positive definite. From a statistical point of view, this means that $\mathbf{X}$ does not contain any columns that are linearly correlated to each other. In this case, the first condition yields the unique solution

$$\mathbf{c} = (\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{P}. \tag{2.5}$$

There are two important interpretations of this solution. On the one hand, it is minimizing the *mean square error* (*MSE*) that the model makes in predicting $\mathbf{P}$. This can be simply seen from its definition

$$err_\mathrm{MSE} \equiv \frac{1}{N}\left(\sum_{i=1}^{N}(P_i - (\mathbf{f}(\mathbf{X}))_i)^2\right) = \frac{1}{N}L(\mathbf{c}). \tag{2.6}$$

The MSE is probably the most common measure for the quality of the model's prediction, giving relatively higher weight to larger deviations by the squaring. In order to use the same dimension as the target property, usually the *root mean square error* (*RMSE*) is reported which is simply the square root of the MSE. On the other hand, the solution can be interpreted in a geometric way. The columns of the input matrix $\mathbf{X}$ span a subspace of $\mathbb{R}^N$. The model $f(\mathbf{X}) = \mathbf{X}\cdot\mathbf{c} = \mathbf{X}(\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{P} = \mathbf{HP}$ then can be seen as the orthogonal projection (expressed by $\mathbf{H}$) of $\mathbf{P}$ into this subspace. Note further that the minimization problem 2.1 is convex. This is a favourable property as convex optimization problems can be solved for large dimensions in an efficient way, and there exist also a lot of solution algorithms for this type of problems [2].

2.2.2 Regularization

In this work, the data matrices $\mathbf{X}$ typically will have many more columns than target observations, i. e. $M \gg N$. Although being computationally feasible, LLSR has two main drawbacks in this case. These arise from the high number of free parameters c_i, equal to the matrix size M. On the one hand, this may adapt the model too closely to the target data at the cost of its generalizability, i. e. it may lead to *overfitting* (cf. Sec. 2.4.1). On the other hand, the *interpretability* is reduced with every additional degree of freedom. Besides that, if $\mathbf{X}$ is large and even contains highly correlated columns, the Gramian matrix $\mathbf{G}$ might be close to singular, i.e. *ill-conditioned.* This will result in an instable solution of Eq. 2.1 (Ref. [29]).

To overcome these drawbacks one usually makes use of a so-called *regularization* of the minimization problem. In a very general sense, this means to extend the original objective function $L'(\mathbf{c})$ by a regularization term $R(\mathbf{c})$,

$$\mathrm{argmin}_\mathbf{c}\left(L'(\mathbf{c}) + R(\mathbf{c})\right).$$

$R(\mathbf{c})$ penalizes the model parameters $\mathbf{c}$ with the intention to *shrink* some of the parameters c_i towards zero. This will stabilize the solution and make it less prone to be overfitted. If the regularization even achieves $c_i = 0$ for some i, it serves to *select* some of the variables (for which $c_j \neq 0$), which crucially improves the model interpretability.

Commonly, the regularization term $R(\mathbf{c})$ penalizes the size of the parameters by some type of ℓ_p-norm of the parameters. For an arbitrary vector $\mathbf{c}$ and $p \in \mathbb{R}^+$, this is defined as

$$\|\mathbf{c}\|_p \equiv \left(\sum_{j=1}^{n}|c_j|^p\right)^{(1/p)}. \tag{2.7}$$

In the case that $p \geq 1$, the ℓ_p-norm satisfies the triangle inequality

$$\left\|\mathbf{x}+\mathbf{y}\right\|_p \leq \|\mathbf{x}\|_p + \left\|\mathbf{y}\right\|_p \tag{2.8}$$

such that $R(\mathbf{c}) = \|c\|_p$ is *convex*. This is beneficial since, for typical similarly convex choices of $L'(\mathbf{c})$ like the root mean square error, the total optimization problem becomes convex, and there exist efficient solution algorithms to handle this type of problems [28], even for very large matrices and numbers of parameters.

2.2.3 The ℓ_0-problem

As explained above, regularization is able to improve the interpretability if it leads to variable selection such that the linear model depends on a small number k of columns of $\mathbf{X}$ only, i.e. the minimizing $\mathbf{c}$ contains only k non-zero components. The straight-forward choice to achieve such a dimensionality reduction thus is to use the *counting-norm* or ℓ_0-norm for $R(\mathbf{c})$. This is defined by[1]

$$\|\mathbf{c}\|_0 \equiv \operatorname{card}(\left\{j \in \{1,\ldots,N\} : c_j \neq 0\right\}), \tag{2.9}$$

simply counting the number of non-zero coefficients. A vector $\mathbf{c}$ with up to k non-zero elements, i.e. $\|\mathbf{c}\|_0 \leq k$, is called *k-sparse*.

The minimization problem hence turns into the so-called ℓ_0*-problem*

$$\operatorname{argmin}_{\mathbf{c}} S_{\ell_0}(\mathbf{c}) = \operatorname{argmin}_{\mathbf{c}} \left(\|\mathbf{P}-\mathbf{X}\mathbf{c}\|_2^2 + \lambda \|\mathbf{c}\|_0\right) \tag{2.10}$$

that formally will produce a *k-sparse* solution. Here, the hyperparameter $\lambda \in \mathbb{R}_0^+$ balances between the two demands of model accuracy (given by the MSE or ℓ_2 term) and sparsity (given by the ℓ_0 term). The larger λ is chosen, the more components of the minimizing coefficient vector $\mathbf{c}$ will be zero. Obviously, in the limiting case of $\lambda = 0$ the solution is identical to the one from linear least square regression.

For larger λ, the solution to Eq. 2.10 technically is obtained by solving Eq. 2.1 at first for all possible j such that $\|\mathbf{c}\|_0 = 1$, then for all j, k such that $\|\mathbf{c}\|_0 = 2$ etc., and to each time select the linear model of minimal MSE. As the number of combinations grows very quickly with the sparsity, this naive algorithm is computionally very costy. Unfortunately however, it cannot be improved much: there exists a proof that the ℓ_0-problem is NP-hard [16, 30] and thus cannot be solved in an effective way in general.

In this context, it is useful to bring Eq. 2.10 that is in *Lagrangian form* (with the multiplier λ) to the equivalent problem

$$\operatorname{argmin}_{\mathbf{c}} \left(\|\mathbf{P}-\mathbf{X}\mathbf{c}\|_2^2\right) \quad \text{subject to} \quad \|\mathbf{c}\|_0 \leq \tilde{\lambda} \tag{2.11}$$

with the constraint written explicitly. If, for instance, $\tilde{\lambda} = 3$, this expresses the seek for a model with minimal quadratic error under the constraint that up to 3 coefficients are non-zero. From this, the solution algorithm – to perform LLSR of all 1-dimensional, 2-dimensional and 3-dimensional models and then pick the best one – can be seen more clearly. Hence, its computation time has the

[1]Note, that the term is mathematically imprecise since it actually does not satisfy all requirements of a norm.

upper limit

$$T_{\ell_0} \leq N\Omega^2 \cdot \begin{pmatrix} M \\ \Omega \end{pmatrix} \leq \sum_{j=1}^{\Omega} N j^2 \cdot \begin{pmatrix} M \\ j \end{pmatrix} \tag{2.12}$$

where the first factors result from the dominating multiplication $\mathbf{X}^T\mathbf{X}$ in LLSR [31], and the second from the number of combinations where obviously the largest contribution is at $j = \Omega$.

2.2.4 Ridge Regression

To avoid solving the computationally problematic ℓ_0-problem explicitely, it might be approximated by using the ℓ_2-norm for the regularization instead. This approach is called *ridge regression* [27] and is expressed by the minimization problem

$$\text{argmin}_{\mathbf{c}} \left(\|\mathbf{P} - \mathbf{X}\mathbf{c}\|_2^2 + \lambda \, \|\mathbf{c}\|_2^2 \right).$$

or

$$\text{argmin}_{\mathbf{c}} \left(\|\mathbf{P} - \mathbf{X}\mathbf{c}\|_2^2 \right) \quad \text{subject to} \quad \|\mathbf{c}\|_2 \leq \tilde{\lambda}$$

in Lagrangian form. Although the ℓ_2 regularization generally does not provide a sparse solution, it leads to a shrinking of the coefficients towards zero and towards each other – with the larger λ, the smoother and simpler the model. Ridge regression will increase the *bias* of the LLSR model to reduce its *variance* (cf. Sec. 2.4.1). Interestingly, its solution can be expressed in the closed form

$$\mathbf{c} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{P}. \tag{2.13}$$

2.2.5 Least Absolute Selection and Shrinkage Operator (LASSO)

Alternatively, a regularization by the ℓ_1-norm can be used to approximate the ℓ_0-problem (Eq. 2.10). This reads as

$$\text{argmin}_{\mathbf{c}} S_{\ell_1}(\mathbf{c}) = \text{argmin}_{\mathbf{c}} \left(\|\mathbf{P} - \mathbf{X}\mathbf{c}\|_2^2 + \lambda \, \|\mathbf{c}\|_1 \right), \tag{2.14}$$

and is usually referred to as linear least square and regression operator [32], in short *LASSO*. The choice of $p = 1$ plays a special role since, on the one hand, it still guarantees the convexity of the optimization problem as the limiting case of the triangle inequality Eq. 2.8. On the other hand, it also promotes sparse solutions which only is achieved if $p \leq 1$.

This can be understood easily in a geometric interpretation of LASSO. For this, also the minimization problem of LASSO is recast to its Lagrangian form with the constraint

$$\|\mathbf{c}\|_1 \leq \tilde{\lambda}$$

that restricts $\mathbf{c}$ to the square-like ℓ_1-unit-balls. Figure 2.1 on the left illustrates this for a two-dimensional problem with the two free parameters c_1 and c_2, in comparison to ridge regression on the right. Here, the red elliptical lines indicate the contours of the pure mean square error with the linear least square solution $\mathbf{c}_{\text{LLSR}}$ in the middle. The square-like blue area marks the ℓ_1-constraint region of LASSO, the spherical blue area the ℓ_2-constraint region of ridge regression. The point where the concentric ellipses touch the constraint first marks the solution of the two regularized problems. Importantly, only for the square-like LASSO constraint this may happen at one of the axes. In that case, one of the coefficients is zero, i.e. LASSO provides a 1-sparse solution or selects one of the variables.

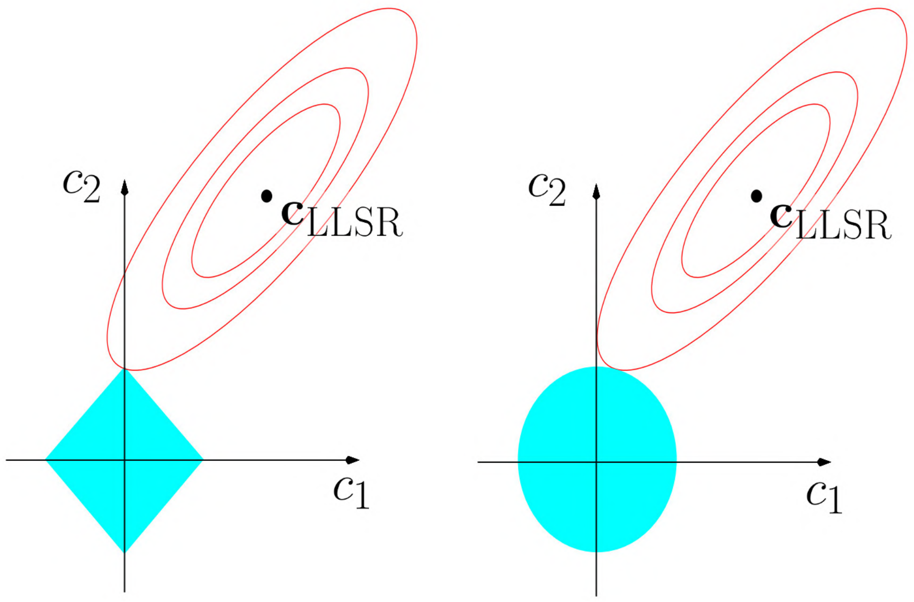


Figure 2.1: Contours of the objective function (the mean square error, red) and the respective constraint regions (blue) in the plane of the coefficients c_1 and c_2 for a two-dimensional problem. On the left, LASSO with the square-like ℓ_0-contraint, on the right, ridge regression with the elliptical ℓ_1-constraint. $\mathbf{c}_{\mathrm{LLSR}}$ marks the solution of a pure linear least square regression, without a constraint. Figure adapted from [27].

LASSO is only an approximation to the exact ℓ_0-problem but it has been studied extensively when the two solutions will coincide. Going into the details is out of the scope of this work, we here refer to [2, 16, 33] for a further reading. As an important result, the research on this question revealed an upper limit of the column number M of $\mathbf{X}$ at which LASSO (Eq. 2.14) yields reasonable solutions to the ℓ_0-problem. This reads as

$$M \leq \exp\left(\frac{N}{C\Omega}\right) \tag{2.15}$$

with a constant $C > 0$ that, in practice, is chosen within $4 \leq C \leq 8$ [16]. M will correspond to the number of generated candidate features in Sec. 5.2 and hence Eq. 2.15 will limit the size of the feature space.

By contrast to ridge regression, the objective function in the minimization problem of LASSO (see Eq. 2.14) is not differentiable due to the ℓ_1-term. Therefore, the problem cannot be solved in a closed form, and also standard solvers including the first derivative are not applicable. As the problem is still convex, it can be tackled by optimization techniques of convex analysis. The two methods being important within the scope of this thesis are *coordinate descent* and *least-angles regression* (*LARS*).

Coordinate Descent: The standard way to solve the LASSO problem is the coordinate descent algorithm [34] which iteratively minimizes its objective function $S_{\ell_1}(\mathbf{c})$. It successively optimizes all directions of $\mathbf{c}$ but only with respect to one c_i at the same time at one step of the iteration. A straightforward implementation is be outlined the following:

1. Start with an initial guess $\mathbf{c}_0$.
2. Determine $\mathbf{c}_1$ by solving

$$c_i^1 = \underset{d \in \mathbb{R}}{\operatorname{argmin}} S_{\ell_1}(c_0^1, \ldots, c_{j-1}^1, d, c_{j+1}^1, \ldots, c_1^k) \tag{2.16}$$

   for each component $d = c_j$ from $j = 1$ to $j = N$ separately and one after each other.

3. Iterate analogously for $i \mapsto i + 1$ to yield $\mathbf{c}_{i+1}$.
4. Stop if sufficient convergence is achieved.

This variant is called *cyclic* as it cycles over the columns at each iteration in step 2 according to their order in **X**. Importantly, it depends on the order of the coefficients **c**, i.e. of the columns of **X**, such that a change in order may lead to a different solution.[2]

Least-angles Regression: Least-angles regression (LARS) [35] is a powerful linear regression method on its own. Similar to OMP (Sec. 2.2.7) is relies on the correlation between features and a residue but does the update of the solution in in a direction *equi-angular* between the features once two or more columns have the same correlation with the residue. Let us outline the steps of the algorithm where it is assumed that the matrix **X** already has been standardized to its mean and standard deviance [27]:

1. Initialize the residual by $\mathbf{R} = \mathbf{P} - \overline{\mathbf{P}}$ and all coefficients by $c_1, \ldots, c_M = 0$.
2. Find the feature $\mathbf{x}_j$ with highest correlation (cf. Eq. 2.17) with the residual **R**.
3. Increase c_j from 0 in the direction of $\operatorname{sign}(\langle \mathbf{R}, \mathbf{x}_j \rangle)$ until another column k has the same correlation with the current residual.
4. Update c_j and c_k in the direction of their LLSR solution until a third column $\mathbf{x}_l$ is equally correlated with the current residual.
5. Iterate until convergence in number of features or accuracy of the model is reached.

The selection of features here is defined by the non-zero coefficients where, if a c_j eventually turns back to zero at step 4, the referring feature is discarded. Importantly – and by contrast to coordinate descent – LASSO-LARS is independent from the order of columns in **X** because the correlation of *all* features is considered at step 2.

2.2.6 Matching Pursuit (MP)

As an alternative approach to solve the ℓ_0 problem, in [36] an iterative scheme called *matching pursuit* (MP) was proposed. The basic idea is to successively decompose the target vector **P** in columns of **X**. Importantly, it is not assumed that **X** is orthogonal, i.e. in other words that it might

[2]Alternatively, the iteration can follow a *random* order at every step which often markedly fastens the convergence – but equally is affected by the column order of **X**.

contain redundant or correlated features. Allover this work, correlation is meant as *linear correlation* which can be expressed by the *Pearson* coefficient of correlation,

$$\rho_{i,j} = \rho(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathrm{cov}(\mathbf{v}_i \mathbf{v}_j)}{\sigma_i \sigma_j} = \frac{\big\langle (\mathbf{v}_i - \mu_i) \cdot (\mathbf{v}_j - \mu_j) \big\rangle}{\sigma_i \sigma_j}. \tag{2.17}$$

for two arbitrary vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ with variance σ_i and σ_j. In the special case that $\mathbf{v}_i$ and $\mathbf{v}_j$ are standardized to have zero mean and variance 1, this definition simply reduces to the standard scalar product $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ which we assume for the rest of this section. Note, that correlation is also investigated graphically in *correlation plots*, plotting the target versus the prediction. Here, the case of a perfect correlation $\rho_{i,j} = 1$ corresponds to all points lying exactly on the bisector, elsewise they are spread to some extent.

The algorithm of MP has a step-wise approach which makes it greedy. It can be sketched by:

1. Initialize the residue $\mathbf{R}_n$ with the target property $\mathbf{P}$ for $n = 1$ and the set of selected column indices to $I_n = \emptyset$.
2. Determine the index $\gamma_n = \underset{j \notin I_n}{\mathrm{argmax}} |\langle \mathbf{R}_n, \mathbf{x}_j \rangle| = \underset{j \notin I_n}{\mathrm{argmax}} |\rho(\mathbf{R}_n, \mathbf{x}_j)|$ and add it to I_n.
3. Update the residue $\mathbf{R}_{n+1} = \mathbf{R}_n - \mathbf{x}_{\gamma_n} c_n$ where $c_n = \langle \mathbf{R}_n, \mathbf{x}_{\gamma_n} \rangle$.
4. Iterate $n \mapsto n+1$ until a convergence criterion (such as a threshold in error or the size of I_n) is reached.

That way an Ω-dimensional linear model $\sum_{i=1}^{\Omega} \mathbf{x}_i c_i$ is determined. Note, that step 3 means to find the vector that is mostly aligned with $\mathbf{R}_n$ from an alternative geometric point of view.

2.2.7 Orthogonal Matching Pursuit (OMP)

MP can be easily extended to the dimensionality reduction method *orthogonal matching pursuit* (OMP) [37, 38]. It modifies the MP algorithm at step 3 such that the coefficients of *all* columns in the current I_n are updated in every iteration. This is done by linear least square regression (cf. Sec. 2.2.1) with a temporary matrix $\mathbf{X}_n$ reduced to the current column selection I_n. The new residual then is obtained by subtracting this fit at each step, i.e. $\mathbf{R}_{n+1} = \mathbf{P} - \sum_{i=1}^{n} \mathbf{x}_i c_i$.

As explained in Sec. 2.2.1, a LLSR fit is equivalent to an orthogonal projection of the target property into the linear subspace spanned by $\mathbf{X}$. In this case, the current residue $\mathbf{R}_n$ is projected in the subspace of $\mathbf{X}_n$. After subtracting the current model, the next residual $\mathbf{R}_{n+1}$ is orthogonal to $\mathbf{X}_n$ – from this reason, this modification is called *orthogonal* matching pursuit. If the matrix $\mathbf{X}$ offers a suitable variety of columns, each next selected column $\mathbf{x}_{n+1}$ will lie mostly aligned with $\mathbf{R}_{n+1}$, and hence almost orthogonal to all previous columns. This can often result in a markedly improved model performance, compared to MP, and reduce the redundancy in the selected columns.

2.2.8 Sure Independence Screening (SIS)

The idea of matching pursuit – to pick columns based on their absolute correlation with the target $\mathbf{P}$ – can be generalized also in another direction. Instead of choosing only a single column, one can instead select a set of the features with highest correlation. This is the idea of *sure independence screening* (SIS, [39]) that builds the basis of the dimensionality reduction method SISSO (cf. 4.4.2).

In a more rigorous way, it can be formulated the following:[3]

- Calculate the absolute correlations $|\rho_j| \equiv |\langle \mathbf{P}, \mathbf{x}_j \rangle|$ for all j.
- Choose a model size $\Omega_{\text{SIS}} = \alpha M$ where $0 < \alpha < 1$ such that $\Omega_{\text{SIS}} \in \mathbb{N}$.
- Determine a model by $\mathscr{D}_{\text{SIS}} = \{1 \leq j \leq M : |\rho_j| \text{ is among the } \Omega_{\text{SIS}} \text{ largest of all } \rho\}$.

Following this approach, the dimensionality can be tackled down from a probably huge size M to a much smaller size Ω_{SIS}. The name of the method derives from the so-called *sure screening property* [39] that guarantees that all important features will be present after the screening with a probability tending to one, i.e.

$$P(\mathscr{D}_{\ell_0} \subset \mathscr{D}_{\text{SIS}}) \to 1 \,\text{for}\, M \to \infty \tag{2.18}$$

where $\mathscr{D}_{\ell_0}$ denotes the exact sparse model from a combinatorial ℓ_0-search (cf. Sec. 2.2.3). Note finally, that the algorithmic complexity to solve SIS is of the order $O(N \cdot M)$, i.e. proportional to the size of $\mathbf{X}$.

2.3 Regression beyond Linearity

2.3.1 Kernel Ridge Regression

All regression methods explained so far considered linear models, $f(\mathbf{X}) = \mathbf{X} \cdot \mathbf{c}$, which often is a too strong restriction on the model structure [40]. One very common method that goes beyond this limitation is *kernel ridge regression* (*KRR*, [27, 40, 41]), that combines ridge regression (see Sec. 2.2.4) with the so-called kernel-trick. It constructs the model by

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i \kappa(\mathbf{x}_i, \mathbf{x}) \tag{2.19}$$

where $\kappa(\mathbf{x}_i, \mathbf{x})$ denotes some type of *kernel function* [42]. A very common choice for κ that we also apply in this work is the *Gaussian kernel* or *radial basis function kernel* (*rbf*),

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \tag{2.20}$$

where $\sigma \in \mathbb{R}$ denotes the width of the kernel.
Analogously to Eq. 2.2.4, kernel ridge regression solves the minimization problem

$$\text{argmin}_{\mathbf{c}} \left(\|\mathbf{P} - \mathbf{f}(\mathbf{X})\|_2^2 + \lambda \sum_{i,j} c_i \kappa(\mathbf{x}_i, \mathbf{x}_j) c_j \right). \tag{2.21}$$

In matrix notation, this reads like

$$\text{argmin}_{\mathbf{c}} \left(\|\mathbf{P} - \mathbf{X}\mathbf{c}\|_2^2 + \lambda \mathbf{c}^{\text{T}} \mathbf{K} \mathbf{c} \right), \tag{2.22}$$

[3]Note, that SIS itself is non-iterative, by contrast to MP, but also has been generalized that way.

where we defined $\mathbf{K} \in \mathbb{R}^{N\times N}$ by $K_{i,j} = \kappa(x_i, x_j)$. Also for kernel ridge regression, a solution exists in closed form, that is given by

$$\mathbf{c} = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{P}. \tag{2.23}$$

Importantly, the matrix to be inverted here is of dimension $N \times N$ while in LLSR it has the dimension $M \times M$. By consequence, while the number of free parameters of LLSR is equal to the *number of features M* ($\mathbf{c} \in \mathbb{R}^M$), KRR has N parameters to be optimized, equal to the *number of samples N* ($\mathbf{c} \in \mathbb{R}^N$).

KRR depends on the parameter λ that balances between the quadratic error and the regularization as well as on the parameter(s) of the used kernel, for instance the Gaussian width σ. In practice, these hyperparameters are often determined through a CV procedure before the actual model training. Note that KRR does not favor sparsity due to the ℓ_2-norm, i.e. they generally depend on the total training data. Additionally, the use of a kernel makes them more abstract than linear methods. This will keep the models in a "black box" that strongly limits their interpretability.

2.3.2 Symbolic Regression

Symbolic regression [7, 14, 43] is another common technique in current artificial intelligence. Similar to standard regression methods it aims at an optimal prediction of the target $\mathbf{P}$ from a set of input variables $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$. Its main idea is to combine these basic variables by algebraic operations to then identify models in the form of simple and human-readable formulas of the most relevant features and their mathematical relation. Symbolic regression could, for instance, select the two variables x_2 and x_5 out from many and connect them algebraically into an expression $\exp(x_2 + 6 \cdot x_5)$ as the model.

Obviously, the number of possible such expressions that could be optimal is infinite in principle. Thus, one typically puts restrictions on their complexity. These could truncate the layers of algebraic combinations up to an understandable level, or reduce the expressions to only physically meaningful ones. However, even with these constraints, symbolic regression is computationally costly.

Often, symbolic regression therefore is used together with *genetic programming* [14] but one can also apply dimensionality reduction methods to find optimal expressions in a potentially huge matrix $\mathbf{X}$ of candidates as done in this work. Balancing between predictive performance and complexity, symbolic regression models typically follow a *Pareto frontier* [44] where the best models at a given complexity follow a monotonically decreasing line.

The algebraic expressions can also be visualized hierarchically as "trees" [7] where terminal nodes represent the considered input variables and interior ones their algebraic combination. Based on this concept, it is common to define quantitative measures for the model complexity [43, 45] such as the number of layers in the tree, the number of terminal nodes, or the length of the tree. The second counts the number of considered basic variables and is also referred to as *information complexity*, the third is known as *expressional complexity*.

2.4 Model Evaluation

2.4.1 Estimating Model Performance

On the one hand, a machine-learned model should give optimal, interpolative fits on the data it is trained on, on the other hand, it also should transfer well to new, unseen data, i.e. have a good generalization ability. The latter is quantified by the *generalization error* or, equivalently, *test error* which formally is defined as [27]

$$err^{te}(U_{tr}) \equiv \mathrm{E}\left[L(\mathbf{P}_{\mathrm{te}}, \mathbf{f}(\mathbf{X}_{\mathrm{te}}))|U_{tr}\right]. \tag{2.24}$$

This is the expectation value of the loss L (e.g. the RMSE) at applying a model $\mathbf{f}$ trained on a fixed training set U_{tr} to input data $\mathbf{X}_{\mathrm{te}}$ and target $\mathbf{P}_{\mathrm{te}}$ of an independent test set U_{te}. Practically, many standard techniques of statistical analysis such as *cross-validation* (see Sec. 2.4.3) or *bootstrapping* [46] instead access the *expected generalization error* or *expected test error*

$$err^{te} \equiv \mathrm{E}\left[L(\mathbf{P}_{\mathrm{te}}, \mathbf{f}(\mathbf{X}_{\mathrm{te}})\right] = \mathrm{E}\left[err^{te}(U_{\mathrm{tr}})\right] \tag{2.25}$$

which averages also over the randomness in choosing the training set U_{tr} that yielded the model. Another important statistical quantity is the *training error*,

$$err^{tr}(U_{\mathrm{tr}}) \equiv \frac{1}{N}\sum_{i=1}^{N} L(P_i, f_i(\mathbf{X})), \tag{2.26}$$

which averages the models' loss over all samples in a specific training set U_{tr}. Similarly to err^{te}, the *expected training error* can be defined as an average over the random choice of the training set by

$$err^{tr} \equiv \mathrm{E}\left[err^{tr}(U_{\mathrm{tr}})\right]. \tag{2.27}$$

Linear least square regression (cf. Sec. 2.2.1) and derived generalized linear methods such as ridge regression (Sec. 2.2.4), LASSO (Sec. 2.2.5) and also SIS (Sec. 2.2.8) rely on the quadratic error in their loss function L. For this choice, a very insightful decomposition of the expected generalization error err^{te} can be derived [27]. Assuming that the target property $\mathbf{P}$ has a normal-distributed error ϵ of variance σ_ϵ, this reads

$$\begin{aligned} err^{te}(x_i) &= \sigma_\epsilon^2 + \left[E\left[f(x_i)\right] - P_i\right]^2 + E\left[f(x_i) - E\left[f(x_i)\right]\right]^2 \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}. \end{aligned} \tag{2.28}$$

for a single data point x_i and a single target value P_i. The *irreducible error* is the error in the property itself (e.g. measurement errors in experiments, the imprecision of computed data etc.) and is a strict lower limit of model performance. This is in contrast to the *reducible error* expressed by the second and the third term that results from the model's imperfectness. The first contribution, (squared) *bias*, is equal to the amount the model prediction deviates from the true value on average and results from the simplification of reality made by assuming a specific form of the model. The second contribution, the *variance*, expresses the squared deviation of the model's prediction around its mean, and is connected to its flexibility and its dependence on the choice of the used training data. Let us illustrate this on a specific set-up in this thesis, to find a linear model for the band gap E_{g}

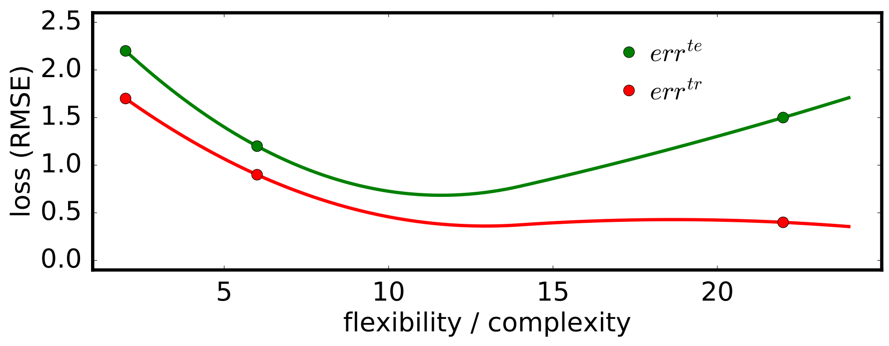


Figure 2.2: Characteristic shape of expected training and test error versus model flexibility or complexity (on an arbitrary scale) for a numeric example. Adapted from [47].

calculated in DFT-LDA approximation. Here, the irreducible error is the difference between the imprecise DFT calculation and the hypothetical true value. The simplification made by assuming that the model is linear and composed of a small number of primary features corresponds to the bias term. The variance of the model is its flexibility in the free coefficients c_i to adapt to the training data.

From Eq. 2.28 it is obvious that a model will generalize well in terms of err^{te} if it simultaneously minimizes the Bias and the Variance term. Practically, the model has to balance between these two by tuning its complexity – the so-called *bias variance trade-off* [27, 47]. Models with a low complexity typically have a high bias but low variance as there is only a few degrees of flexibility. Vice versa, models of higher complexity are equipped with a lower bias but a higher variance as they are much more adaptable to the training data.

An increasing model flexibility or complexity affects the two measures for model performance, err^{tr} and err^{te}, in a different way. Figure 2.2 illustrates the typical shape of their curves for a numeric example. Their relation $err^{te} > err^{tr}$ at any flexibility is not necessary but very common since the model is trained to optimize err^{tr}. Note here the two limiting cases [41, 47, 48]:

- If the model is too simple, both err^{tr} and err^{tr} are large, i.e. the model is *underfitted*.
- If the complexity is too high, err^{tr} still reduces because the model can adapt more and more to the training data. This, however, is at the cost of a high generalization error err^{te} and called *overfitting*.

In practice, ones thus limits the flexibility e.g. by regularization (see Sec. 2.2.2) to achieve an optimal generalizability.

2.4.2 Error Metrics

In this work, we use several different metrics for quantifying the deviance of the model prediction $\mathbf{f}(\mathbf{X})$ from the target $\mathbf{P}$ [41]. Each of them focuses on different aspects and has certain advantages and disadvantages. Naturally, there is no general answer to the question which error metric to use. A very common measure is the *root mean square error* (*RMSE*) that was already defined in Sec. 2.2.1. As it includes a squaring, it gives a relatively high weight to large differences between target and prediction, so in particular *outliers* could dominate the metric. It is a natural choice for the quality assessment of optimization problems involving an ℓ_2-norm of the differences, in particular linear least square regression (Sec. 2.2.1). Compared to the MSE, taking the root ensures that $\mathrm{err}_{\mathrm{RMSE}}$ has the same dimension as the target $\mathbf{P}$.

An alternative error metric is *mean absolute error* (*MAE*),

$$\mathrm{err}_{\mathrm{MAE}} \equiv \frac{1}{N}\sum_{i=1}^{N} |P_i - f_i(\mathbf{X})|, \tag{2.29}$$

that averages the individual absolute differences and hence gives equal, linear weight to them. Equally, it has the same dimensions like the target. Due to its discontinuities, it is rather unusual to use $\mathrm{err}_{\mathrm{MAE}}$ as a loss function.

Often it is interesting to focus on the worst-case prediction a model does. The size of this is quantified by the *maximum absolute error* (*maxAE*)

$$\mathrm{err}_{\mathrm{maxAE}} \equiv \max_{1\le i\le N}(|P_i - f_i(\mathbf{X})|), \tag{2.30}$$

which, obviously, also has the same dimension as $\mathbf{P}$.

These three metrics allow to be compared with the absolute numbers of the target. As an alternative, one also reports *relative errors* where, for instance, the individual residuals r_i are divided by the respective target value P_i

$$r_i' = \frac{r_i}{P_i} = \frac{P_i - f_i(\mathbf{X})}{P_i}, \tag{2.31}$$

before being averaged by one of the definitions above.

2.4.3 Cross-Validation

A standard technique to estimate the generalization error of a model is *cross-validation* (*CV*) [27]. By exploiting the existing data in a statistical way, it yields an approximate for the model generalizability to new data. In cross-validation, the data is split multiple times into training and test sets where the former are used to adapt the model to and the latter to evaluate the generalization capability. Averaging over the splits then yields estimates for expected training and test error.

More rigorously, the procedure is defined the following: Let $\left\{U_{tr}^{(i)}, i = 1,\ldots,N_{\mathrm{CV}}\right\}$ and $\left\{U_{te}^{(i)}, i = 1,\ldots,N_{\mathrm{CV}}\right\}$ be N_{CV} *partitions* of the total data of sizes N_{tr}^i and N_{te}^i each, i.e. for fixed i a data sample s either is in a training set $U_{tr}^{(i)}$ or in a test set $U_{te}^{(i)}$. Indicate by $f(s|U_{tr}^{(i)})$ the prediction for sample s by the model trained on $U_{tr}^{(i)}$, by $P(s)$ the target for s and by L a loss function. Then cross-validation approximates the expected training error by

$$err^{tr} = \frac{1}{N_{\mathrm{CV}}}\sum_{i=1}^{N_{\mathrm{CV}}}\frac{1}{N_{tr}^i}\sum_{s\in U_{tr}^{(i)}} L(P(s), f(s|U_{tr}^{(i)})), \tag{2.32}$$

and the expected generalization or test error by[4]

$$err^{te} = \frac{1}{N_{\text{CV}}} \sum_{i=1}^{N_{\text{CV}}} \frac{1}{N_{te}^{i}} \sum_{s \in U_{te}^{(i)}} L(P(s), f(s|U_{tr}^{(i)})). \tag{2.33}$$

There are several different types of cross-validation that differ in the way the total data is repeatedly split into training and test sets [50]. This might be done in an *exhaustive* or *non-exhaustive*, i.e. *partial* manner, depending on whether all possible partitions or only some of them are considered. The most common CV schemes are:

- **V-fold CV:** a general partition of the total data into $V \in \mathbb{N}$ (approximately) equally sized random splits $\{U_i\}$ is applied. Iteratively, each set once acts as test set and the remainder as training set. Typically, values of $V = 5$ and $V = 10$ are recommended [51]. This partial scheme is fast since it involves only V iterations.
- **Leave-group-out CV (LOGCV)**: in this equally partial CV scheme one or several groups of the data, e.g. based on physical intuition [52], are once left out as test set.
- **Leave-p-out CV (LPOCV):** here, *all* possible subsets of the total data with exactly $p \in \mathbb{N}$ data points form the partition into training and test sets. A special case is leave-one-out CV (*LOOCV*) where each single data point is excluded once for testing. p is often given as percentage proportion of the total data – for example in leave-10%-out CV ($L10\%OCV$), the test sets comprise 10% of the data. As LPOCV exhaustively considers all $\binom{N}{p}$ sets, the number of iterations grows combinatorially with p and hence quickly becomes computationally infeasible. Thus, in practice one approximates this scheme by using a fixed and much smaller number of iterations N_{CV}. The partitioning then uses N_{CV} *randomly chosen* but distinct test and training sets of size $p \in \mathbb{N}$ and $N - p$ each. It has to be ensured that N_{CV} is large enough to provide sufficient convergence in average errors.

For learning problems with a very large space of features one typically uses two-step learning methods [27] that at first do a pre-screening by a dimensionality reduction method $\mathcal{M}$. On the thus obtained much smaller subset of the best features, the model $\mathcal{D}$ is then built by a second method $\mathcal{M}'$. For assessing this approach, it is essential that cross-validation (of whichever type) replicates the *total* model selection $\mathcal{M} \to \mathcal{M}' \to \mathcal{D}^{(i)}$ at every individual iteration i. This means to apply $\mathcal{M} \to \mathcal{M}'$ to the individual training sets $U_{tr}^{(i)}$, yielding models $\mathcal{D}^{(i)}$, and then evaluate them on the respective test set $U_{tr}^{(i)}$. Approximates for the training and test errors err_{tr} and err_{te} are obtained by averaging over all accumulated splits (where eventually $\mathcal{D}^{(i)} \neq \mathcal{D}^{(j)}$).

In a data-rich situation [27], one typically uses additionally a so-called *hold-out* [48] or validation set $\mathbf{S}_{\text{ho}}$.[5] This contains some proportion of the total data that is taken apart in the very beginning for a final estimate of model performance. It is considered neither at model fitting nor at any step of cross-validation – in this case the CV partitions split the remaining data only. This is beneficial as the data used for model selection and for validation is separated more strictly.

[4]This quantity is also frequently referred to as *CV scale* with the abbreviation $CV(f)$ [49].

[5]Unfortunately, the use of the terms "hold-out", "validation" and "test set" is not uniform.

Chapter 3

Background in Theoretical Solid State Physics

This work applies machine learning methods to ab initio materials data. To generate this data, we applied the codes `exciting` *and* `FHIaims` *which both implement density functional theory (DFT). In the following we give a brief overview of the basic principles of DFT in Sec. 3.1, and a summary of the main features of the codes* `exciting` *and* `FHIaims` *in Sec. 3.2. Section 3.3 explains properties of the electronic band structure used in the machine learning and Sec. 3.4 the tight binding model by which also some of the data was calculated.*

3.1 Density Functional Theory

3.1.1 The Hohenberg-Kohn Theorems

A fundamental problem underlying the calculation of solid state systems is the solution of the time-independent Schrödinger equation

$$\hat{H}\Psi = E\Psi \tag{3.1}$$

where $\hat{H}$ is the Hamilton operator, E the energy eigenvalue and Ψ the wave-function depending on the positions of all electrons and nuclei. From Ψ the expectation values of all observables can be obtained. Since Ψ depends on all electronic and nuclear coordinates, solving Eq. 3.1 is demanding and usually even impossible, and hence clever approximations are required to solve it in practice. As a first step, usually the so-called *Born-Oppenheimer approximation* [53] is applied which consists in decoupling the motion of electrons and nuclei. This is possible due to the much smaller mass of the electrons. Additionally, the nuclei can be considered static such that the Schrödinger equation (Eq. 3.1) reduces to an equation for the electronic wave-function, which reads

$$(\hat{T} + \hat{W} + \hat{v}_{\text{ext}})\Psi_{\text{el}} = E\Psi_{\text{el}}. \tag{3.2}$$

Here, $\hat{T}$ is the kinetic energy operator of the electrons, $\hat{W}$ the electron-electron interaction operator and $\hat{v}_{\text{ext}}$ the external potential due to the positively charged nuclei.

In 1964, Hohenberg and Kohn proposed two fundamental theorems which provide a dramatical simplification to this electronic problem [3]. The first theorem states that for a system of N_{el} interacting electrons the external potential $\hat{v}_{\text{ext}}$ is fully determined by the ground state electronic density

$$\rho_0(\mathbf{r}) \equiv N_{\text{el}} \cdot \int \left|\Psi_{\text{el},0}(\mathbf{r},\ldots,\mathbf{r}_{N_{\text{el}}})\right|^2 d\mathbf{r}_2 \cdots d\mathbf{r}_{N_{\text{el}}} \tag{3.3}$$

and vice versa. As a direct consequence from this, $\rho_0(\mathbf{r})$ fully determines the electronic ground state wave-function $\Psi_{\text{el},0}$. Therefore, the expectation value of any observable in the ground state can be expressed as a functional of the density ρ_0.

In particular, this holds for the energy of the system, $E = E[\rho]$, which is important for the second theorem of Hohenberg and Kohn. This transfers the variational principle of the wave-function to the electronic density

$$E_0 = \min_{\Psi_{\text{el}}} E[\Psi_{\text{el}}] \;\mapsto\; E_0 = \min_{\rho'} E[\rho'], \tag{3.4}$$

i.e. the ground state energy E_0 in principle could be obtained by minimizing the energy functional with respect to all trial electronic densities ρ' keeping the number of electrons N_{el} constant. So, the crucial statement of the theory of Hohenberg and Kohn is, that it reduces the electronic problem of Eq. 3.2 depending on all $3N_{\text{el}}$ electronic coordinates to a problem of just 3 spatial coordinates (i.e. those of $\rho(\mathbf{r})$). Unfortunately, however, these two theorems do not have any direct practical implication as the analytical form of the functional $E[\rho]$ is unknown. Nonetheless, very accurate approximations to it have been developed which are presented in the next section.

3.1.2 The Kohn-Sham Scheme

Only one year later, in 1965, Kohn and Sham found an ansatz how to solve the issue of the unknown functional $E[\rho]$ which is the second keystone of density functional theory [4]. The central idea of their approach is that the electronic many-body system can be identified with a much simpler auxiliary system of *non-interacting* electrons that reproduces the ground-state density ρ_0. In consequence, the total energy $E[\rho]$ can be recast as

$$E[\rho] = T_s[\rho] + W_H[\rho] + V_{\text{ext}}[\rho] + E_{\text{XC}}[\rho] \tag{3.5}$$

with the single particle kinetic energy $T_s[\rho]$ and the *Hartree* energy $W_H[\rho]$, the classical electrostatic interaction. The additionally introduced exchange-correlation functional $E_{\text{XC}}[\rho]$ is defined to absorb all further interactions such that the many-body system is resembled exactly, i. e. the remainder in both kinetic energy and electron-electron interaction.

In the Kohn-Sham picture the electronic ground-state density takes the form

$$\rho_0(\mathbf{r}) = \sum_{i=1}^{N} \left|\psi_i(\mathbf{r})\right|^2, \tag{3.6}$$

where the single-particle wave functions ψ_i are found by solving the *Kohn-Sham* equations

$$\left(-\frac{1}{2}\nabla^2 + v_{\text{KS}}(\mathbf{r})\right)\psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}). \tag{3.7}$$

Here, the Kohn-Sham potential is given by [54]

$$v_{\mathrm{KS}}(\mathbf{r}) = v_{\mathrm{ext}}(\mathbf{r}) + v_{\mathrm{H}}(\mathbf{r}) + v_{\mathrm{XC}}(\mathbf{r}) = v_{\mathrm{ext}}(\mathbf{r}) + \int d\mathbf{r}' \frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} + \frac{\delta E_{\mathrm{XC}}[\rho]}{\delta\rho(\mathbf{r})} \tag{3.8}$$

which was derived by a variation of the total energy functional, $\delta E[\rho]/\delta\rho = 0$. Similar to Eq. 3.5, it comprises the external potential $v_{\mathrm{ext}}(\mathbf{r})$, a Hartree term $v_{\mathrm{H}}(\mathbf{r})$ and the exchange-correlation contribution $v_{\mathrm{XC}}(\mathbf{r})$.

As the Kohn-Sham equations depend on the electronic density $\rho(\mathbf{r})$, which by Eq. 3.6 again depends on the Kohn-Sham orbitals $\psi_i(\mathbf{r})$, they need to be solved in a self-consistent manner. From a sufficiently converged solution, the total energy of the ground state is calculated as

$$E[\rho_0] = \sum_{i=1}^{N_{\mathrm{el}}} \epsilon_i - \frac{1}{2}\int v_{\mathrm{H}}(\mathbf{r})\rho_0(\mathbf{r})d\mathbf{r} + E_{\mathrm{XC}}[\rho_0] - \int v_{\mathrm{XC}}(\mathbf{r})\rho_0(\mathbf{r})d\mathbf{r} \tag{3.9}$$

from which the ground state properties are determined. The Kohn-Sham scheme was proven to yield the ground state energy of the many-body system in theory. However, an exact expression of the exchange-correlation potential $E_{\mathrm{XC}}[\rho]$ is not known and hence efficient approximations to it are necessary. There have been many different approximations developed so far, on different levels of accuracy and complexity [55]. The simplest of them, the *local-density approximation*, was used to generate the data in this work and will be outlined in the following.

3.1.3 The Local Density Approximation

This approach to approximate the exchange-correlation potential $E_{\mathrm{XC}}[\rho]$ was already proposed in a seminal work of Kohn and Sham [4]. Here, the electron density $\rho(\mathbf{r})$ is assumed to behave locally like a homogeneous electron gas. In LDA, E_{XC} is split further into its exchange and correlation contributions $E_{\mathrm{X}}(\rho)$ and $E_{\mathrm{C}}(\rho)$.[1] For the former part, an analytical expression is given by [56–58]

$$E_{\mathrm{X}}(\rho) = \int \epsilon_x(\rho(\mathbf{r}))\rho(\mathbf{r})d\mathbf{r} \tag{3.10}$$

where $\epsilon_x(\rho(\mathbf{r}))$ is the exchange energy density of the homogeneous electron gas

$$\epsilon_x(\rho(\mathbf{r})) = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3} \rho(\mathbf{r})^{\frac{1}{3}}. \tag{3.11}$$

By contrast, in the analogous expression of the correlation energy

$$E_{\mathrm{C}}(\rho) = \int \epsilon_c(\rho(\mathbf{r}))\rho(\mathbf{r})d\mathbf{r}, \tag{3.12}$$

there is no analytical expression for the correlation energy density $\epsilon_c(\rho(\mathbf{r}))$. It was estimated numerically with a high precisicion by Monte Carlo simulations [59] from which several analytical parametrizations have been developed [60, 61]). In this work we use the parametrization by Perdew and Wang (*LDA-PW*) [62].

LDA provides reliable approximations for systems with a slowly varying electronic density like metals. Due to the fullfillment of certain sum rules, LDA has given good results also in cases where

[1]These are only functions of ρ now due to the assumption of locality.

the density is not slowly varying [63]. That way, calculations based on LDA are able to reproduce bond lengths and hence the geometries of molecular and solid bulk structures typically with ~ 1% accuracy. While for these systems the *structure* of electronic bands is also reasonable [64], the fundamental band gaps are typically underestimated by ~ 40%, the so-called *band gap problem* of LDA [65]. A prominent example is the semiconducting Ge in diamond structure which LDA even misclassifies as a metal.

3.2 Ab-initio Codes

To generate both the primary features and the target DFT data for the machine learning, the *ab-initio* codes `exciting` [66] and `FHIaims` [67] were applied. Generally, the variety of DFT codes can be distinguished by their way they expand the electronic wave-functions [68]:

1. by localized orbitals,
2. by plane waves,
3. by augmented functions.

In this classification, `FHIaims` belongs to class 1 as it uses numeric atom-centered orbitals. By contrast, `exciting` does the expansion by plane-waves that are augmented by atomic functions and thus belongs to class 3. We give a brief outline of both of them in the following.

3.2.1 `exciting`

The DFT code `exciting` expands the electronic wave-function in terms of *augmented plane waves (APW)*. The basic idea is to divide the space of the calculated material in muffin-tin (MT) spheres centered at the atomic sites $\mathbf{R}_\alpha$, and an interstitial region (I) outside of these spheres. Formally, the real-space Kohn-Sham wave function of the ith state and the wave vector $\mathbf{k}$ is expanded as

$$\psi_{i\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{G}} C_{i\mathbf{G}}^{\mathbf{k}} \phi_{\mathbf{G}+\mathbf{k}}(\mathbf{r}), \tag{3.13}$$

where $C_{i\mathbf{G}}^{\mathbf{k}}$ are expansion coefficients and the summation is over all reciprocal lattice vectors $\mathbf{G}$. The basis functions - the augmented plane waves - are defined by

$$\phi_{\mathbf{G}+\mathbf{k}}(\mathbf{r}) \begin{cases} \sum_{lm} A_{lm\alpha}^{\mathbf{G}+\mathbf{k}} u_{l\alpha}(r_\alpha) Y_{lm}(\hat{\mathbf{r}}_\alpha) & \text{if} \quad r_\alpha \leq R_{\mathrm{MT}} \\ \\ \frac{1}{\sqrt{\Omega}} e^{i(\mathbf{G}+\mathbf{k})} & \text{if} \quad \mathbf{r} \in \mathrm{I} \end{cases} \tag{3.14}$$

In the muffin tins of radius R_{MT} (that depends on the atom species), the wave function is expanded in terms of the radial functions $u_{l\alpha}(r_\alpha)$ and the spherical harmonics $Y_{lm}(\hat{\mathbf{r}}_\alpha)$. In the interstitial space, this is just a plane-wave, normalized by $\frac{1}{\sqrt{\Omega}}$. The coefficients $A_{lm\alpha}^{\mathbf{G}+\mathbf{k}}$ serve to satisfy the continuity of the APW at the borders between the muffin tins and the interstitial region.

For $r_\alpha \leq R_{\mathrm{MT}}$, the Kohn-Sham potential v_{KS} is adequately approximated by its spherical average $v_0(r)$. Thus, the radial wave functions $u_{l\alpha}(r_\alpha)$ have to satisfy the radial Schrödinger equation

$$\left[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + \frac{l(l+1)}{2r^2} + v_0(r) - \epsilon_{i\mathbf{k}}\right](r u_{l\alpha}(r)) = 0. \tag{3.15}$$

with an eigenvalue $\epsilon_{i\mathbf{k}}$. From this, the augmented plane wave basis functions are fully defined, allowing to bring the Kohn-Sham equations (Eq. 3.7) for a specific **k**-vector in the matrix form

$$\mathbf{H}^{\mathbf{k}}\mathbf{C}^{\mathbf{k}} = \epsilon^{\mathbf{k}}\mathbf{S}^{\mathbf{k}}\mathbf{C}^{\mathbf{k}} \tag{3.16}$$

with the coefficient eigenvector $\mathbf{C}^{\mathbf{k}}$ specifying the expansion of Eq. 3.13. Here, $\mathbf{H}^{\mathbf{k}}$ is the Hamiltonian matrix consisting of the elements $\langle\phi_{\mathbf{G}+\mathbf{k}}(\mathbf{r})| - \frac{1}{2}\nabla^2 + v_{KS}(\mathbf{r})|\phi_{\mathbf{G}'+\mathbf{k}}(\mathbf{r})\rangle$, and the overlap matrix $\mathbf{S}^{\mathbf{k}}$ with the elements $\langle\phi_{\mathbf{G}+\mathbf{k}}(\mathbf{r})|\phi_{\mathbf{G}'+\mathbf{k}}(\mathbf{r})\rangle$.

Since the basis functions depend on the eigenenergies $\epsilon_{i\mathbf{k}}$ in the radial part, the eigenvalue problem of Eq. 3.16 is non-linear. In `exciting`, there are two different strategies implemented to make it computationally tractable. On the one hand, the radial functions are expanded by

$$u_{l\alpha}(r_\alpha,\epsilon) \approx u_{l\alpha}(r_\alpha,\epsilon_{l\alpha}) + (\epsilon_{l\alpha} - \epsilon)\dot{u}_{l\alpha}(r_\alpha,\epsilon_{l\alpha}) \tag{3.17}$$

where $\epsilon_{l\alpha}$ is constant and $\dot{u}_{l\alpha}$ the first derivative with respect to the total energy, yielding the classical *linearized augmented plane waves* (LAPW) basis set [66, 69]. On the other hand, the APWs are considered with fixed ("frozen") energy parameters $\epsilon_{l\alpha}$, and additional radial basis functions are introduced by

$$\phi_\mu(\mathbf{r}) = \delta_{\alpha\alpha_\mu}\delta_{ll_\mu}\delta_{mm_\mu}\left[a_\mu u_{l\alpha}(r_\alpha;\epsilon_{l\alpha}) + b_\mu\delta\dot{u}_{l\alpha}(r_\alpha;\epsilon_{l\alpha})\right]Y_{lm}(\hat{\mathbf{r}}_\alpha). \tag{3.18}$$

The ϕ_μ are strictly located inside the MT of one atom and called *local orbitals* (*LO*) therefore. Here, $\mu = (\alpha, l, m)$ denotes a certain l, m orbital of a muffin tin α, and the coefficients a_μ and b_μ are used to satisfy the conditions $\phi_\mu(\mathbf{r}) = 0$ at the MT boundary and $\int_{\mathrm{MT}} |\phi_\mu(\mathbf{r})|^2 d\mathbf{r} = 1$ over the volume of MT. `exciting` allows to freely combine these two concepts and hence has the total familiy of the LAPW methods implemented - often referred to as the gold standard of solving the KS equation due to their very high precision in energy [70].

Equation 3.16 is solved efficiently by restricting the reciprocal lattice vectors **G** by the cut-off condition $|\mathbf{G}+\mathbf{k}| < G_{\mathrm{max}}$. The number of considered **G** vectors that satisfy this condition determines the number of LAPWs and in turn the size of the Hamiltonian (together with the number of local orbitals). The cut-off parameter G_{max} is determined by the dimensionless parameter `rgkmax` $= R_{\mathrm{MT}} \cdot G_{\mathrm{max}}$. Thus, for a given `rgkmax`, larger radii R_{MT} imply a smaller G_{max} and a smaller Hamiltonian, and vice versa. Note that R_{MT} here refers to the smallest value in case of several atomic species. The paramter `rgkmax` determines the precision of the expansion in the region I [70] with typical values between 4 (e.g. for light materials) and 9 (e.g. for a high precision in forces), and even more than 12 (e.g. for heavier elements or d-states).

Two other parameters influence the precision of `exciting`: (1) the mesh size $(n_{\mathbf{k}_1}, n_{\mathbf{k}_2}, n_{\mathbf{k}_3})$ (`ngridk`) of the **k**-point grid for integrations over the Brillouin-zone, (2) the plane-wave cut-off `gmaxvr` $= |\mathbf{G}|$ for the expansion of the interstitial electron density and potential. Large unit cells require a smaller `ngridk`, and, in comparison to insulators and semiconductors, metals typically require a larger ngridk, in order to accurately resolve the Fermi surface. $|\mathbf{G}|$ commonly is set to values around $12\,\mathrm{Bohr}^{-1}$ and $16\mathrm{Bohr}^{-1}$. Proper parametrization is done by convergence tests, for instance, with respect to total energy, balancing between precision and computation time.

A main application field of `exciting` is the calculation of the electronic energy spectrum of periodic systems. Beyond DFT, it supports the many-body perturbation theory method GW in the single-shot approximation G_0W_0 [71]. For further information on `exciting`, in particular its input parameters, we refer to its documentation online [72].

3.2.2 FHIaims

Part of the data used for this thesis was created with `FHIaims` (Fritz Haber Institute *ab initio* molecular simulations package) [67] that can be applied to both periodic bulk structures and isolated molecules. This code uses numeric atom-centered orbitals (NAOs) as basis set for expanding the wave function. These are defined by

$$\phi_i(\mathbf{r}) = \frac{u_i(r)}{r} Y_{lm}(\Omega) \tag{3.19}$$

where i refers to an electron, $Y_{lm}(\Omega)$ denotes the spherical harmonics and $u_i(r)$ the radial functions. The latter are numerically tabulated from the solution of the radial Schrödinger equation on a logarithmic grid with additionally confining them to the region of one atom. Similar to `exciting`, the electronic density is determined in a self-consistent manner. Code efficiency is based on the spatial confinement of the radial functions, the use of effective linear algebra packages and high parallelization. Accuracy is mainly determined by the set-up of the species-dependent parameters regarding the basis set, integration grids and the precision of the Hartree potential. The code provides several sets of default settings from which the option `tight` is used in the data of this work, providing sufficient convergence in the generic case (see the documentation of the code).

3.3 Ionization Potential, Electron Affinity, Band Gap

For a general system with N electrons, the ionization potential and the electron affinity are defined as differences in total energies of the N-, $(N-1)$- and $(N+1)$-electronic system

$$IP = E_{N-1} - E_N \tag{3.20}$$

and

$$EA = E_N - E_{N+1}, \tag{3.21}$$

i.e. as the energy change due to electron removal or insertion [73, 74], respectively. The *fundamental band gap* of the system is then given by

$$E_g = IP - EA = E_{N-1} + E_{N+1} - 2E_N. \tag{3.22}$$

The *Kohn-Sham gap* from DFT calculations actually is the difference

$$E_g^{\mathrm{KS}} = \epsilon_{\mathrm{lu}} - \epsilon_{\mathrm{ho}} \tag{3.23}$$

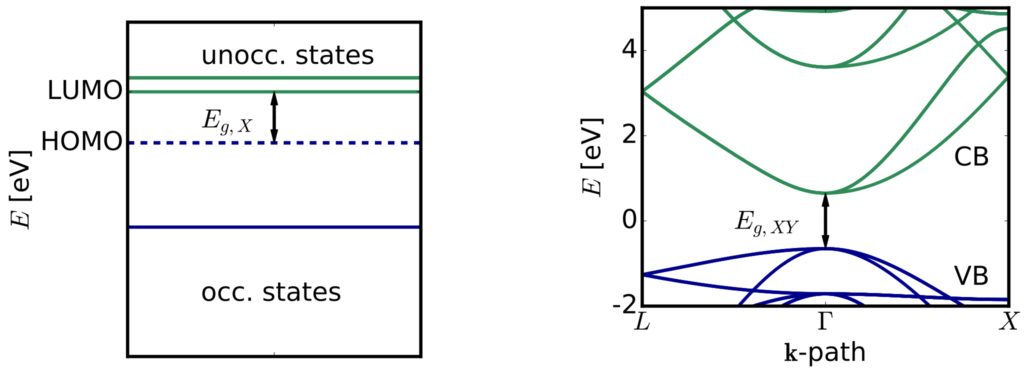


Figure 3.1: Left: occupied (blue) and unoccupied (green) electronic energy states of an isolated carbon atom (from the generated data described in Sec. 7.4.1). The $2p$ states at the HOMO are plotted by a dashed line to indicate the 3-fold degeneracy. The HOMO-LUMO gap, the primary feature $E_{g,X}$ in Tab. 8.2, is indicated by the arrow. Right: band structure along the **k**-path L-Γ-X of SiC (in the binary supercell of Fig. 7.2, from the data described at the end of Sec. 7.2). Valence bands (VB) and conduction bands (CB) are marked in blue and green, respectively. The fundamental band gap is taken at the Γ-point (denoted as $E_{g,XY}$ in Chapter 8).

between the lowest unoccupied (ϵ_{lu}) and the highest occupied state (ϵ_{ho}), i.e. for atoms and molecules the highest occupied and the lowest unoccupied molecular orbit (*HOMO* / *LUMO*). Importantly, E_g^{KS} is not equal to E_g due to the dicontinuity of exchange-correlation effects [73, 75]. However, if we approximate $IP \approx -\epsilon_{\mathrm{ho}}$ and $EA \approx -\epsilon_{\mathrm{lu}}$, the HOMO-LUMO difference approximates E_g [76, 77]. Fig. 3.1 on the left illustrates this for the *ab initio* levels of an isolated carbon atom that we use to derive the atomic HOMO-LUMO gap, the primary feature $E_{g,X}$ in the machine learning input data (see Sec. 7.4.1 and Tab. 8.2).

In the continuous band structure $E(\mathbf{k})$ of bulk materials, the *fundamental gap* E_g is defined as the difference between the conduction band minimum and the valence band maximum, that can be located at different **k**. Instead of this, we consider the *direct gap* at the high-symmetry point $\mathbf{k} = \Gamma$,

$$E_g^{\Gamma} = E_{\mathrm{CB}}^{\Gamma} - E_{\mathrm{VB}}^{\Gamma}, \tag{3.24}$$

as primary feature for the group-IV ternary dataset **T** and as target for the octet binaries **O** in Chapter 8. This definition is indicated in Fig. 3.1 on the right for bulk silicon carbide, obtained as explained at the end of Sec. 7.2.

3.4 The Tight Binding Approach

A common model to calculate the electronic bands is the *tight binding* (*TB*) approach which is computationally much faster than methods based on density functional theory [58, 78, 79]. It is based on superpositions of single-atom wave-functions located at the crystal sites. This effectively keeps the electrons localized at the atoms which is a reasonable assumption for materials like semiconductors and insulators whereas it is less appropriate for metals.

In the TB approach, the Hamiltonian of a crystalline system can be written as [80]

$$\hat{H} = \hat{T} + \sum_{n,v} V_{\text{at}}(\mathbf{r} - \mathbf{r}_{nv}) \tag{3.25}$$

where $\hat{T}$ is the kinetic energy. The second term decomposes the potential energy into the potentials of isolated atoms, where $\mathbf{r}_{nv}$ marks the position of atom v in the unit cell n. The TB approach now makes the ansatz

$$\Theta_{vm}(\mathbf{k},\mathbf{r}) = \frac{1}{\sqrt{N}} \sum_n \exp^{i\mathbf{k}\cdot\mathbf{r}_{nv}} \vartheta_{v,m}(\mathbf{r} - \mathbf{r}_{nv}) \tag{3.26}$$

for the wave-function of atom v in the unit cell n. The sum on the right hand runs over all unit cells with the atomic wave-functions $\vartheta_{v,m}(\mathbf{r} - \mathbf{r}_{nv})$ weighted such that the Bloch condition for periodic crystals is satisfied. $\vartheta_{v,m}(\mathbf{r} - \mathbf{r}_{nv})$ are the solutions to the Hamiltonian of isolated atoms where m indicates the quantum number of the eigenstate. With this ansatz, the total wave-function of the crystal is expanded as

$$\phi_{\mathbf{k}}(\mathbf{r}) = \sum_{v,m} C_{vm} \Theta_{vm}(\mathbf{k},\mathbf{r}). \tag{3.27}$$

In the basis of the wave-functions Θ_{vm}, the Hamiltonian can be brought into matrix form

$$H_{v'm'vm}(\mathbf{k}) = \langle \Theta_{v'm'}(\mathbf{k}) | H | \Theta_{vm}(\mathbf{k}) \rangle. \tag{3.28}$$

It is found by straight-forward calculus that $H_{v'm'vm}(\mathbf{k})$ contains only integrals over the atomic wave functions $\vartheta_{v,m}(\mathbf{r} - \mathbf{r}_{nv})$. The coefficients C_{vm} in Eq. 3.27 which solve the many-body Schrödinger equation are determined by diagonalizing $H_{v'm'vm}$.

Unfortunately, the number of integrals in $H_{v'm'vm}(\mathbf{k})$ might be large and has to be limited in practice. This is usually achieved by restricting the considered atomic wavefunctions to only a few states m, for example to solely s- and p-orbitals. Alternatively, one can limit the spatial range of interactions by, for instance, neglecting all except the nearest-neighbors' contributions. The number of parameters of the TB approach thus can be reduced drastically and, even more, these are commonly fitted to match DFT results [81]. This is important for the scope of this work where the fitting relies on *on-site energies* E_{OS} at single atomic sites as well as semi-empirical *transfer integrals*. In specific, the former is done by atomic on-site energies of s- and p-states from DFT calculations in local density approximation (LDA, cf. Sec. 3.1.3).

Chapter 4

Machine Learning in Computational Material Science

This chapter addresses the required theory in data-driven materials science, a quickly growing and very interdisciplinary field. In Sec. 4.1, we give a brief overview of the current research. Section 4.2 explains the practical steps on the path from raw data to a machine learned model. In Sec. 4.3, we present the theoretical concepts of representing materials numerically which is the basis for successful models. Finally, in Sec. 4.4 we outline the dimensionality reduction methods LASSO+ℓ_0 and SISSO that recently were developed to find symbolic regression-like material descriptors.

4.1 Data-driven Material Science: Status Quo

Although the methods of statistical learning have been applied more to the soft-sciences chemistry, biology and life-science first [9], they also spread to theoretical solid state physics over the last ten to twenty years, in front of the enormous amount of available *ab initio* data (see Fig. 4.1). So far, the variety of successfully applied data-science techniques and studied materials science problems is remarkable, as summarized in the recent reviews of Refs. [7–9, 82]. In this work we concentrate on supervised learning tasks (see Sec. 2.1) for quantitative materials property prediction, i.e. of a real valued scalar target **P**. For this, the prediction of the lattice parameter of binary wurtzite superlattices [83] or of G_0W_0 band gaps of a set of inorganic compounds [84] are anong many other examples [12, 85–87]. The lack for a direct interpretability of the model here is a common problem, even if a high numeric precision is reached. Symbolic regression approaches are promising to improve over this issue as demonstated for instance by the identification of structual features of hydrogenated amorphous silicon or the rediscorvery of the Johnson-Mehl-Avrami-Kolmogorov equation for recrystallization kinetics ([15, 40]).

A too nested and hence incomprehensible model expression as in Ref. [88] can be avoided by combining symbolic regression with compressed sensing, the idea of the work of Ghiringhelli et. al. [1, 2]. They used this approach to identify *comprehensible* material descriptors for predicting the energy difference between zincblende and rocksalt structure of 82 octet binary compounds [89, 90]. Here, simple atomic and dimer features were processed to a large pool of heuristic formulas first, as for instance $\mathrm{EA}(B) - \mathrm{IP}(B)/r_p(A)^2$, consisting of the electron affinity EA and the ionization potential IP of the anion A, and an atomic radius r_p of the cation B. The best *descriptor* $\mathcal{D}$, defined as low-dimensional linear combination of such formulas, was then extracted from the candidate features by the method LASSO+ℓ_0 (see Sec. 4.4.1), reaching an accuracy level of 0.06 eV (RMSE). An upfollowing work of Ouyang et. al. [91] demonstrated that the alternative method *SISSO* (Sec. 4.4.2) is able to reduce the correlation in descriptor features and to push forward the limitations on feasible feature space size, not only for this data set [92].

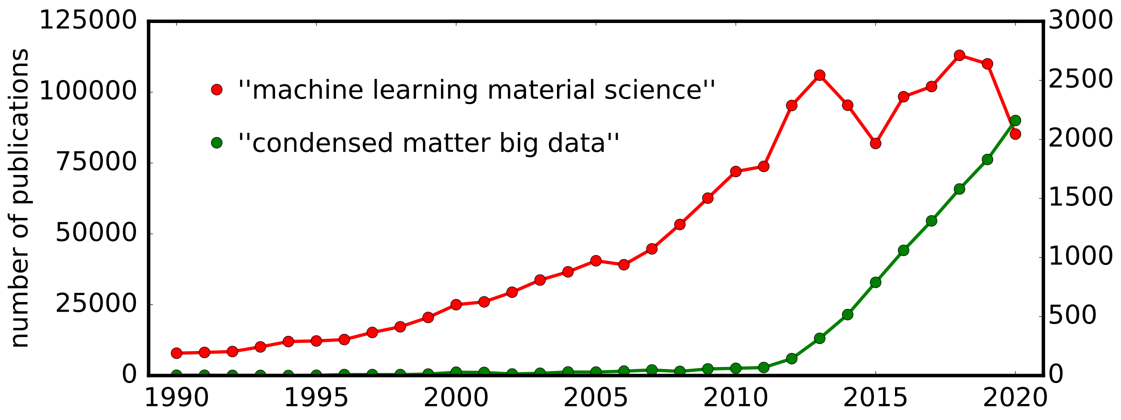


Figure 4.1: Number of publications in data-driven material science, considering the number of results of google scholar searches with the two terms "machine learning material science" (left axis) and "condensed matter big data" (right axis).

Another critical step forward in computational materials science concerns a more rigorous study of learning strategies and how to assess a model's perfomance and generalization capabilities [8, 93]. This means to further transfer established statistical learning methods to material science as done in Ref. [48]. Very recently, the benefits of a cross-validation based feature selection were demonstrated at predicting impurity activation energies by Gaussian Process Regression [52], an approach that also is combinable with symbolic regression. Part of this research area is also the question, how to best exploit data on simpler properties of a material at predicting a more elaborate one. Here, closely related strategies like Δ-*learning* (to learn the difference between a computationally cheap and an expensive one), *multi-fidelity learning* (using low-fidelity data as features for a model of a high-fidelity approximation) or a *hierarchical* descriptor search (to build a feature space for a complex property by descrittors for a simpler one) already have started to emerge [8, 94, 95].

4.2 Steps from Raw Data to a ML Model

The gained experience from the pioneer works in data-driven materials science already led to several similar attempts to outline a generic workflow to find valuable models [8, 9, 82, 96]. These can be summarized to the four basic steps shown in the scheme of Fig. 4.2.

Step 1 comprises all kind of data preparation which typically is the most technical, laborious and time-consuming task. Both input and target data has to be acquired, for instance from public data bases [97, 98], or generated by new calculations. Raw data needs to be preprocessed or parsed to extract the relevant quantities and the data structure (e.g. subcategories of materials). A manual or automated analysis of the parsed data is essential, for instance by graphically examining crystal structures or the bands, or by checking the consistency of the *ab-initio* parametrization.

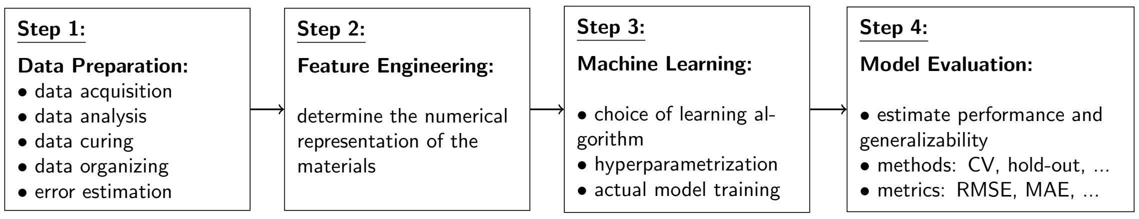


Figure 4.2: The four basic steps of machine learning in computational material science.

Detected outliers or erroneous data may require data cleaning, involving the repetition of some DFT calculations or the exclusion of specific samples. At this step also the error in the data has to be estimated since it determines the irreducible error (see Eq. 2.28) that limits the further model accuracy. In that way, clearly organized input and output data of well understood veracity [5] is obtained.

In regard of the *ab-initio* data itself, one can distinguish the following hierarchical levels (modified from Ref. [96]):

- *level 0 / input data:* the pure chemical and geometric information
- *output level 1:* electronic charge density ρ_{el}, wave-functions ψ_{el}, energy levels $E(\mathbf{k})$,
- *output level 2:* ground state energy E_0 and atomic forces $\{F_k(i)\}$,
- *output level 3:* physical properties derived from level 1 and 2 such as binding energies, energy differences, lattice constants, band gaps or bond lengths.

Note here, that so far machine learned models predominantly have been developed to predict properties on level 2 and 3. In this work, we use data from levels 0, 1 and 3 in the input to learn target properties on level 3 solely, namely lattice constants, energies of mixing and band gaps.

The second step is the feature engineering which means to determine a numerical representation (also called fingerprint) of the materials as the input for the machine learning. This has to include physical domain expertise and strongly depends on the chosen learning algorithm. This processing might consist of determining the atomic building blocks in the materials or involve more complex transformations, but should require by far less cost than the generation of the target data the model will surrogate. For quantitative supervised learning tasks, the feature engineering ends up at a data set of the structure $\{s, \mathbf{x}(s), P(s)\}$, i.e. a label s for the material, a string of input numbers $\mathbf{x}(s)$, and the real-valued target $P(s)$. We address more details on this step in Sec. 4.3.

The third step comprises the actual machine learning by a properly chosen learning algorithm, such as LASSO+ℓ_0 or SISSO in this work. In preparation, the *hyperparameters* of the algorithm need to be determined by numerical tests. These might be [99]: (1) *numerical* hyperparameters of the representation like the width of a Gaussian Kernel, (2) *structural* hyperparameters of the representation, for instance the layers of operations in symbolic regression or the type of the basis functions in KRR, (3) hyperparameters *of the ML method* such as the balancing factor λ of LASSO.

The actual training then yields a model $\mathscr{D}$ for the property prediction, optimal with respect to a specific criterion. Essentially, the model also needs to be evaluated which is step 4 in the workflow. This usually is done by cross-validation (Sec. 2.4.3) or related techniques, or by means of graphical methods, and uses error metrics like the RMSE to check the model performance and generalizability.

type	description	interpretability	accuracy
gross level descriptors	based on simple atomic features (mass, period, radii...) or other general properties (bulk modulus, density), mostly used with linear methods	high	low
molecular-fragment descriptors	based on features of larger building blocks (dimers, organic groups), frequently combined with kernel methods	medium	medium
fine level descriptors	capture fine structure details (< 1Å), approximate electronic density, used with neural networks or KRR	low	high

Table 4.1: Classification of material descriptors based on their granularity. Based on [8].

4.3 Representing Materials for Machine Learning

A crucial step in ML in computational materials science is to numerically represent a material s by a vector of features $\mathbf{x}(s)$ which is the ingredient for any further supervised or unsupervised learning task. The terminology is not that uniform here, but $\mathbf{x}(s)$ can be called *descriptor, fingerprint* or *representation* more or less synonymously. The very fundamental descriptors of a material are position and charge of the nuclei $\{\mathbf{r}_i, Z_i\}$ and the number of electrons N_{el} as they determine any quantum mechanical property via the many-body Hamiltonian [13]. Surrogate ML models will bypass the complex interdependences of the Hamiltonian but similarly will rely on the geometry and / or the electronic properties of the atoms as the base information carried by $\{\mathbf{r}_i, Z_i\}$ and N_{el} (i.e. on data on level 0). How to process these fundamental descriptors and also enrich them by further available information to features $\mathbf{x}(s)$ reaches from simply determining the atomic numbers in a material, over a statistical aggregation of atomic properties [85, 100], up to approximating the local electronic density by radial distribution functions [101] or a smooth overlap of atomic positions [102].

Different types of material descriptors can be classified, for instance with regard to their granularity as in Table 4.1. Here, roughly the finer and elaborate the fingerprints, the higher is the predictive accuracy but the lower may be the human understanding gained from the model. Another criterion is the locality: *local* descriptors represent only some parts of the materials (like building blocks or building blocks in their environment) and are suitable for learning local target properties (forces etc.). If processed further, e.g. averaged over the total material, they can also be applied to global, extensive target properties (like melting temperatures). *Global* descriptors characterize a material s as a whole and hence are used to predict global properties (like band gaps etc.). Finally, representations can be distinguished by the system they are applied to, where the main split is between finite and periodic systems.

One can summarize a set of important requirements for material descriptors [1, 99, 103]:

1. Invariance to transformations that keep the property $\mathbf{P}$. This refers to shifts in indexing the

atoms in the unit cell or to geometric transformations like unit cell rotations.

2. Uniqueness, i.e. any transformation that changes the property $\mathbf{P}$ should also be reflected by a change in the descriptor. Otherwise, "degenerate" fingerprints $\mathbf{x}(s) = \mathbf{x}(s')$ of two systems s, s' that differ in $\mathbf{P}$ would erroneously lead to equal predictions for $\mathbf{P}$.

3. Correlatedness in the sense that small vs. large changes in the descriptor should reflect small vs. large changes in $\mathbf{P}$.

4. Generality, so the descriptor should be able to encode any comparable atomistic system instead of being too specific.

5. Simplicity, considering both computational efficiency in comparison to $\mathbf{P}$, and human interpretability.

Additionally, one may demand continuity and even differentiability with respect to atomic coordinates, typically for the fine level descriptors of Tab. 4.1.[1]

4.4 Dimensionality Reduction Methods

4.4.1 LASSO + ℓ_0 Approach

As already explained before in Sec. 2.2.3, an exact solution of the ℓ_0-problem is not feasible for large matrices $\mathbf{X}$, even for low-dimensional linear combinations, due to the combinatorial explosion of computatione time. A way to overcome this problem was presented in Ref. [1] which combines the LASSO method and an exact solution of the ℓ_0-problem, and hence is called LASSO+ℓ_0 or ℓ_1+ℓ_0 approach. The main idea is to first determine a much smaller preselection I_{ℓ_0} of the most important features by LASSO and to then solve the ℓ_0-problem exactly, being feasible on the smaller set.
More in detail, the approach works the following way:

- As a preparation step, the data matrix $\mathbf{X}$ is standardized columnwise to its means and standard deviations to ensure that the columns approximately have the same weight in the ℓ_1-regularizing term $\|\mathbf{c}\|_1$ in Eq. 2.14.

- The preselection of the $\tilde{M} \ll M$ most relevant features is done by sequentially applying LASSO to $\mathbf{X}$ with decreasing hyperparameter λ. The maximum, λ_m, is chosen such that all coefficients c_j are equal to zero (none of the feature columns is selected), and then reduced on a decreasing sequence $\Lambda = (\lambda_i)_{i\in\mathbb{N}}$ (which is, for instance, a logarithmic grid). That way, more and more c_j will be non-zero and the referring column $\mathbf{x}_j$ will be added to the selection. The procedure stops when the set of selected columns, denoted by the set of indices $I_{\ell_0} = \{j|\exists\lambda_i \in \Lambda : c_j \neq 0\}$, has reached a certain number $\tilde{M}$.

- On this subset of reduced dimensionality, the ℓ_0-problem is solved exactly (being feasible here) to obtain the best $1,\ldots,\Omega_\mathrm{max}$-dimensional descriptors $\{\mathscr{D}_i\}$. As explained in Sec. 2.2.3, this is equivalent to perform LLSR of all $1,\ldots,\Omega_\mathrm{max}$-dimensional combinations of coefficients and to pick each one with the lowest RMSE.

[1]The materials representation is called *exact* in that case [103].

Note, that one also could think of using LASSO alone to approximate the ℓ_0-problem, without the ℓ_0-step. The optimal Ω-dimensional descriptor then simply would be the first linear combination of features with exacly Ω non-zero coefficients at descreasing λ. Unfortunately, it turned out that this straight-forward approach does not work if **X** contains many highly correlated columns (with a Pearson correlation close to 1) [2] which typically is also the case for the pools of candidate features in this work. The combined ℓ_0+ℓ_1 approach however was demonstrated to yield the exact solution of the ℓ_0-problem (at low Ω where the combinatorial solution is feasible) for certain cases, and thus to improve in this issue.

4.4.2 SISSO Approach

A closer investigation of the LASSO+ℓ_0 approach yielded that it nevertheless has problems to identify optimal descriptors. On the one hand, this is due to a limitation of LASSO to matrix sizes of the order of $M = 10^6$ features. Even in disregard of Eq. 2.15, practically these feature space sizes exceed the memory capabilities of standard local workstations with the used implementation of LASSO.[2] On the other hand, the intended improvement over the pure LASSO for highly correlated features in **X** turned out to be quite limited: the yielded descriptors $\{\mathscr{D}\}$ still can consist of highly correlated and hence redundant features in many cases, as is demonstrated in Sec. 8.2.2. Against this background, Ouyang et. al. [91] recently proposed the *sure independence screening and sparsifying operator* (in short, *SISSO*) which could improve over LASSO+ℓ_0 in these two aspects. The name derives from combining a screening for the most relevant feature columns by *sure independence screening* (SIS, cf. Sec. 2.2.8) with an additional dimensionality reduction method, here called *sparsifying operator* SO, in an iterative way. The method is sketched in Fig. 4.3 and can be outlined as follows [13]:

1. **SIS step for $i = 1$:** Rank all columns by their (absolute) correlation $|\rho|$ (see the definition of Eq. 2.17) with the target **P** and select the M_{SIS} ones with highest $|\rho|$ to a first subspace $\mathbf{S}_1$.

2. **SO step for $i = 1$:** Select the column $\mathbf{x}_1$ from $\mathbf{S}_1$ with the highest $|\rho|$ as 1-dimensional descriptor $\mathscr{D}_1$. The coefficient c_1 is yielded from a linear least square regression by Eq. 2.1 to **P**. Evaluate the residual by $\mathbf{R}_1 = \mathbf{P} - c\mathbf{x}_1$.

3. **SIS steps for $i > 1$:** Find the M_{SIS} columns being highest correlated with the previous residual $\mathbf{R}_{i-1}$. The ith subspace $\mathbf{S}_i$ is determined by building the union of this subset of features $\tilde{\mathbf{S}}_i$ with all previous subsets $\tilde{\mathbf{S}}_j$, i.e. by $\mathbf{S}_i = \cup_{j=1}^{i}\tilde{\mathbf{S}}_j$ (here, trivially $\tilde{\mathbf{S}}_1 = \mathbf{S}_1$).

4. **SO steps for $i > 1$:** Apply the sparsifying operator SO to the subspace $\mathbf{S}_i$ to determine the i-dimensional descriptor $\mathscr{D}_i$.

The iteration is stopped if a certain stopping criterion is satisfied, like a threshold in prediction accuracy, or if a chosen maximum of descriptor dimension $\Omega_{\max}$ has been reached. For the sparsifying operator SO, different choices are proposed and discussed in Ref. [91], such as the dimensionality reduction methods LASSO and OMP (see Secs. 2.2.5 and 2.2.7), and also SO = ℓ_0, i.e. an exact combinatorial solution of the ℓ_0-problem Eq. 2.10. The size of the subspaces M_{SIS} is determined by the capabilities of SO, for instance for SO = LASSO, $\cup_{j=1}^{\Omega}\tilde{\mathbf{S}}_i$ could contain up to 10^6 features. For SO = ℓ_0 and $\Omega_{\max} = 5$, difficulties from a combinatorial explosion limit the feasible size of $\cup_{j=1}^{i}\tilde{\mathbf{S}}_i$ to a

[2]Ref. [2] and this work uses the Python package `sklearn.linear_model`, as also addressed in Sec. 6.3.2.

few tens of features. Nevertheless, Ouyang et. al. [91] demonstrated SO=ℓ_0 to surpass the other options.

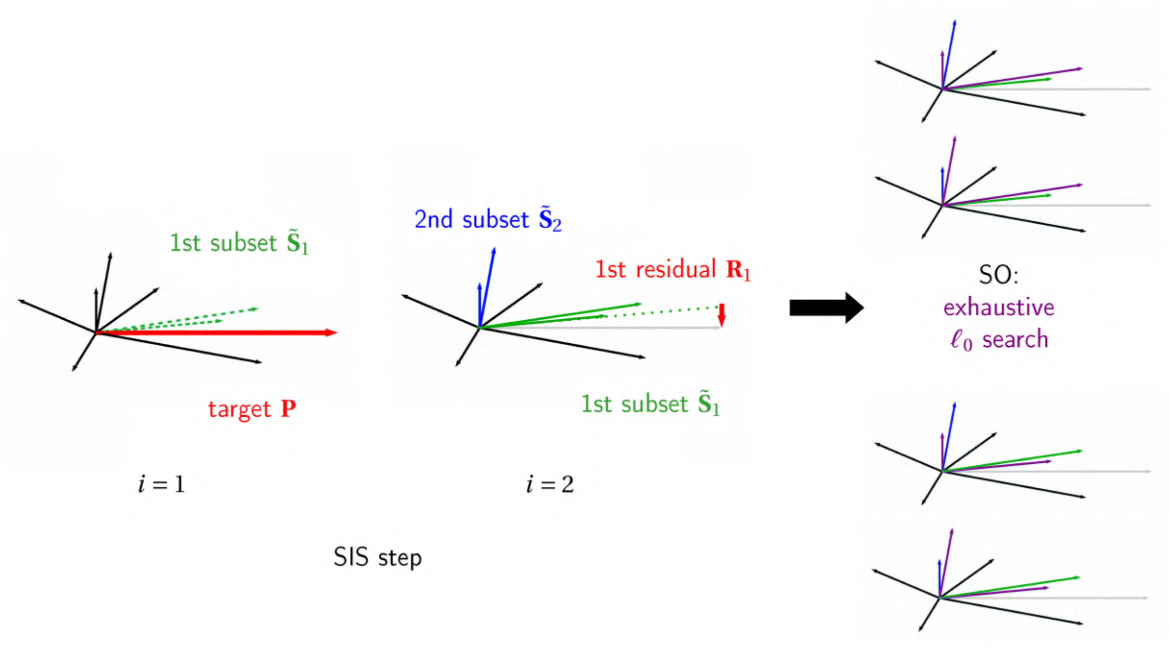

1st subset $\tilde{\mathbf{S}}_1$
target $\mathbf{P}$
$i = 1$
2nd subset $\tilde{\mathbf{S}}_2$
1st residual $\mathbf{R}_1$
1st subset $\tilde{\mathbf{S}}_1$
$i = 2$
SIS step
SO:
exhaustive
ℓ_0 search

Figure 4.3: Schematic visualization of the SISSO approach. Target property $\mathbf{P}$, features of $\mathbf{X}$, the bunch-like subsets $\tilde{\mathbf{S}}_{1/2}$ and the first residual $\mathbf{R}_1$ are visualized in a two-dimensional plane. Left: iterations $i = 1$ and $i = 2$ of the SIS screening, right: combinatorial search for the best 2-dimensional descriptor of SO = ℓ_0. The purple features are combined to a two-dimensional linear combination.

Methods and Implementation Part II

Chapter 5

Machine Learning Methodology

In this chapter, we present the machine learning methodology that was developed in this thesis. In Sec. 5.1 we explain our general approach how to obtain and assess optimal descriptors starting from simple primary features. Section 5.2 presents the scheme to engineer a feature space,. The actual set-up of the methods LASSO+ℓ_0 and SISSO is briefly described in Sec. 5.3. Section 5.4 explains three model selection strategies that employ cross-validation in a different way. In Sec. 5.5 we present quantitative measures to assess specific characteristics of a descriptor beyond its pure numeric performance.

5.1 General Machine Learning Approach

Our general machine learning approach adapts the one of Refs. [1, 2] (see Sec. 4.1) and is sketched in Fig. 5.1. It starts from simple *primary features* $\{f_k\}$ which could comprise atomic data or other properties of the materials' constituents.[1] These features are then *engineered* by two different steps to a feature space that is represented in the data matrix **X**. First, the $\{f_k\}$ are processed to primary features $\{F_k\}$ by an averaging scheme such that they represent a structure as a whole, not only its constituents. Second, these are further modified and combined by various algebraic operations as in symbolic regression (see Sec. 2.3.2). That way, a huge pool **X** of candidates with an analytical formula like, for instance, $(r_s^2 - d_{AB}^2)/2$, is obtained. Note, that we distinguish the primary features $\{f_k\}$ and $\{F_k\}$ in terminology by calling the former *local* as they are specific to a "building block" γ of the material, and the latter *global* since they refer to the total material.

The best descriptor $\mathscr{D}$, a low dimensional linear combination of features, is hidden in **X** like the needle in a haystack. It is extracted by a learning method $\mathscr{L}$ in a selection strategy $\mathscr{S}$. The learning method $\mathscr{L}$ combines a dimensionality reduction method $\mathscr{M}$ (for instance LASSO or OMP, see Sec. 2.2) and an exhaustive ℓ_0-search (see Sec. 2.2.3). The selection strategy $\mathscr{S}$ defines a criterion for what actually makes the descriptor the best. Finally, the so obtained optimal descriptor of low dimension Ω is assessed with respect to its performance and generalizability by a cross-validation procedure.

[1]For some examples see Tables 8.1 and 8.2 and Tab. 8.11 for ML on the the datasets **T** and **O**, respectively.

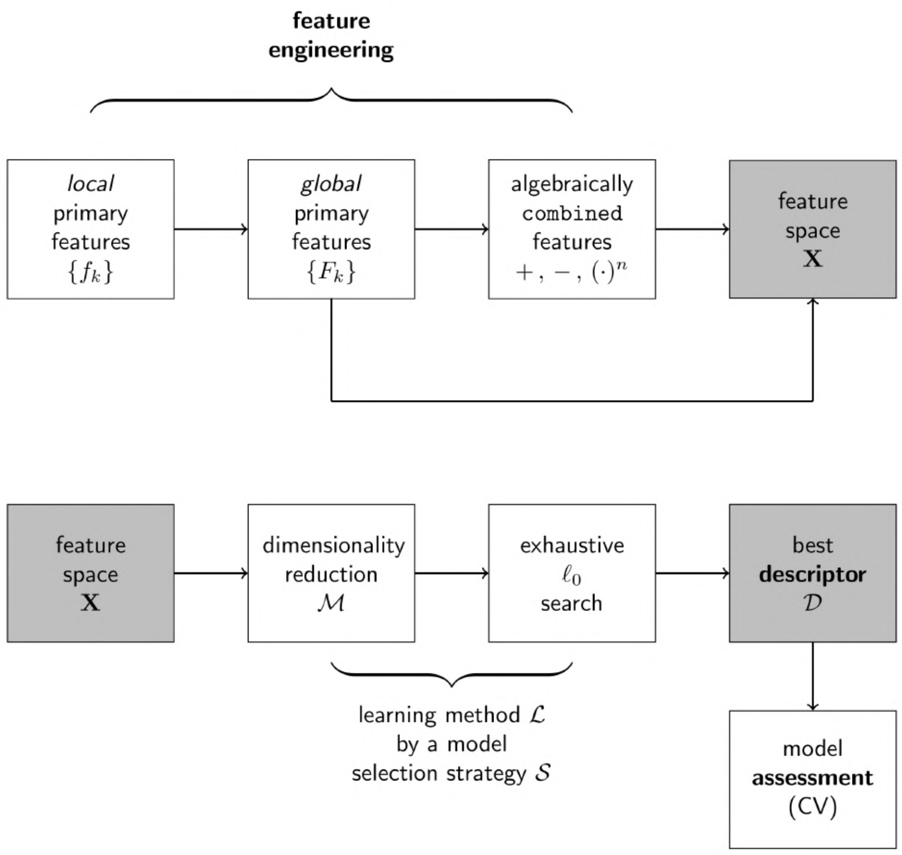

feature engineering
local primary features $\{f_k\}$
global primary features $\{F_k\}$
algebraically combined features $+, -, (\cdot)^n$
feature space $\mathbf{X}$
feature space $\mathbf{X}$
dimensionality reduction $\mathcal{M}$
exhaustive ℓ_0 search
best descriptor $\mathcal{D}$
learning method $\mathcal{L}$ by a model selection strategy $\mathcal{S}$
model assessment (CV)

Figure 5.1: The steps of the machine learning approach of this work: generation of a feature space $\mathbf{X}$ from primary features $\{f_k\}$ by a feature engineering scheme (top), and the actual selection of the best descriptor $\mathscr{D}$ and its assessment (bottom).

5.2 Feature Engineering

5.2.1 Generating Global Features

For very simple materials like the octet binaries of the dataset **O** (see Sec. 8.5.1), a local primary feature $\{f_k\}$ already may suffice to fully characterize a material. For example, the feature d_{XY}, the dimer distance, already represents a binary structure AB in total since there is only one distinct pair of atoms in AB, e.g. d_{GaAs} would already fully characterize GaAs. For more complex systems, it is necessary to process the local quantities $\{f_k(\gamma)\}$ to global features $\{F_k\}$ that equally represent a material by a single scalar number such that a low-dimensional linear model can be constructed from them. This is the puprpose of the scheme explained in the following. It relies on the two *fundamental descriptors* [13] of a material, $\{\mathbf{R}_\alpha, Z_\alpha\}$, the set of atomic positions and atomic numbers.[2] Importantly, we use the atomic positions of the ideal structures as the relaxed ones would already require for costly DFT calculations (see Sec. 7.1) which the ML approach is desired to circumvent.

[2]Ref. [13] actually names position and charge of the nuclei $\{\mathbf{r}_i, Z_i\}$ and the number of electrons N_{el} in the system but we assume $\sum_i Z_i = N_{\text{el}}$ here.

Averaging Scheme: A straightforward approach to generate a global primary feature is by averaging the local primary features $\{f_k(\gamma)\}$ over all the building blocks of one type γ by

$$F_{sk} = \frac{1}{m}\sum_{\gamma\in s} f_k(\gamma). \tag{5.1}$$

m denotes the number of such building blocks in the structure s and serves for normalization. For example, Eq. 5.1 could define the average over all atoms' orbital radii or over all tetrahedral clusters' formation energies. For each building block in the sum, the value of f_k labeling this specific γ will contribute, i.e. the specific orbital radius of the atomic species at a site i or the specific formation energy of the tetrahedral cluster of four atoms at i. This idea of averaging is closely related to Vegard's law (see Eq. 7.8) which approximates the lattice constant a_{AB} of a binary crystal by a linear interpolation between the lattice constants of the respective pristine materials a_{AA} and a_{BB}. If Eq. 5.1 is applied to a_{AA} as a primary feature f_k (i.e. the atoms α in the supercell are labeled by the respective bulk lattice constant) it is even identical to Vegard's law.

Raw and Central Higher Moments: As a next step, we generalize the averaging scheme by taking the q-th *raw moments* $\bar{F}^q$ and the q-th *central moments* $\tilde{F}^q$ of f_k, respectively. This is done by

$$\bar{F}^q_{sk} = \frac{1}{m}\sum_{\gamma\in s}\left(f_k(\gamma)\right)^q \tag{5.2}$$

and

$$\tilde{F}^q_{sk} = \left[\frac{1}{m}\sum_{\gamma\in s}\left(f_k(\gamma) - F_{sk}\right)^q\right]^{1/q}, \tag{5.3}$$

where $q \in \mathbb{N} \geq 2$ defines the order of the moment. For the trivial case $q = 1$, Eq. 5.2 is identical to Eq. 5.1 and Eq. 5.3 equals to zero. The raw moments of Eq. 5.2 define averages of higher powers of the local primary features which may provide insightful physical interpretations: if, for instance, f_k is an atomic radius r, Eq. 5.2 effectively means to average the surfaces of atomic spheres in case of $q = 2$, and their volumes in case of $q = 3$. The features $\bar{r}^2$ and $\bar{r}^3$ hence can be interpreted to cover effects of the surface or the volume of the constituting atoms on the target. The central moments of Eq. 5.3 define averages with respect to the mean F_{sk} and hence carry information on the shape of the distribution of f_k. For $q = 2$, it is equal to the standard deviation of f_k, for $q = 3$ it is related to the skewness of f_k which is a measure for the asymmetry of a distribution. We include the power by $1/q$ in the definition of the central moments to ensure that they have the same dimension as the initial properties. This is convenient in view of the further augmentation of the feature space by mathematical operations (cf. Sec. 5.2.2).

Figures 5.2 and 5.3 additionally illustrate the definitions of Eqs. 5.2 (top parts) and 5.3 (bottom parts) by applying them to an exemplary structure of the group-IV ternary data set **T**. They serve to get an understanding of the magnitude of the contributions and the captured characteristics of the distribution for the different types of averages. The first one is for the case of an atomic local primary feature (atomic radii r_s and r_d, respectively), the second one for a pair feature (dimer distance d_{XY}). The columns indicate the contributions at the individual building blocks γ to the sums of Eqs. 5.2 and 5.3 for $q = 1$ (red), $q = 2$ (blue) and $q = 3$ (green), i.e. at the individual atoms i and for all distinctive pairs (α, β), respectively. The dotted lines mark the resulting averages where the operation $1/q$ is omitted for scaling reasons.

Raw and central higher moments bring two important benefits: on the one hand, they introduce non-linearities into the feature space. This is important because the ML approach is limited to *linear* models $f(\mathbf{X}) = \mathbf{Xc}$ so any non-linearity has to be included in the features themselves. On the other hand, they are similar to a common correction scheme to Vegard's law accounting for *bowing* effects (see Eq. 7.9). While, however, Eqs. 5.2 and 5.3 consider higher powers of the features f_k, these bowing corrections use higher powers of the concentrations.

Nearest-neighbor Differences: In many cases, materials' properties depend on differences in atomic properties, both with respect to geometry and to composition. For instance, differences in the atomic size between the components species were found to be related to deviations from Vegard's law [104, 105]. Stimulated by this work, we augment the pool of candidate features with averages over nearest neighbor differences of the local primary features:

$$(\Delta F)_{sk} \equiv \frac{1}{m} \sum_{\gamma=(i,j)\in s} \left| f_k(i) - f_k(j) \right| \equiv \frac{1}{m} \sum_{\gamma=(i,j)\in s} f_{k'}(\gamma) = F_{sk'}. \tag{5.4}$$

This is an average over all pairs of neighbors $\gamma = (i,j)$ in the fashion of Eq. 5.2, using the special case of pair features $f_{k'}$ that are generated by absolute differences $\left|f_k(i) - f_k(j)\right|$ in a single-atom feature f_k. Similarly to above, we generalize this average also by q-th raw and central moments as in Eqs. 5.2 and 5.3 and denote them with $(\bar{\Delta}F)^q_{sk}$ and $(\tilde{\Delta}F)^q_{sk}$.

We study the properties of averaged nearest-neighbor differences on a few simple examples. First, consider a four-atomic periodic linear chain of a binary mixture AB and a local primary feature f_k with $f_k(A) = 0$ and $f_k(B) = 1$. For varying composition from $N_A = 4$ to $N_B = 4$, Tab. 5.1 illustrates the individual contributions $\left|f_k(i) - f_k(i+1)\right|$ at the bonds $(i, i+1)$ and the resulting value of $(\Delta F)_k$. Several important characteristics of $(\Delta F)_k$ become apparent here, also in regard to the requirements on a representation (see Sec.4.3):

1. $(\Delta F)_k$ is invariant with respect to an exchange of species $A \leftrightarrow B$. This is because $(\Delta F)_k$ is only sensitive to bonds with different species and obviously an exchange $A \leftrightarrow B$ does not affect this characteristics of a bond.
2. $(\Delta F)_k$ is invariant to shifts in atom indexing and translations of the unit cell.
3. $(\Delta F)_k$ is sensitive to differences in atomic arrangement at fixed composition (N_A, N_B).

Importantly, property 1 does not generalize to systems with more than two components. This is illustrated in Fig. 5.4 (a) for a one-dimensional three-component system with a six-atomic unit cell. We assume that $f_k(A) \neq f_k(B) \neq f_k(C)$ and study an exchange of species A (black) and C (green). This changes the total types of bonds and hence $(\Delta F)_k$. For example, the most-right bond AB is replaced by CB in the right chain, a bond motif that is not present in the left chain. Note further, that $(\Delta F)_k$ *may* capture arrangements effects but also *may* miss them. This is shown in Fig. 5.4 (b) for a one-dimensional two-component system of periodicity 6. Here, the two different arrangements of the composition $N_A = 2, N_B = 4$ lead to the same $(\Delta F)_k$ because they have the same types of bonds in total although they change locally. Mathematically, the map from the arrangements of a composition to $(\Delta F)_k$ generally is not injective and hence $(\Delta F)_k$ is not a unique material representation. This lack of uniqueness generalizes to all averages over pairs by Eqs. 5.1, 5.2 and 5.2 which is obvious if the next-neighbor difference is interpreted as a pair features $f_{k'}(i,j)$ in Eq. 5.4.

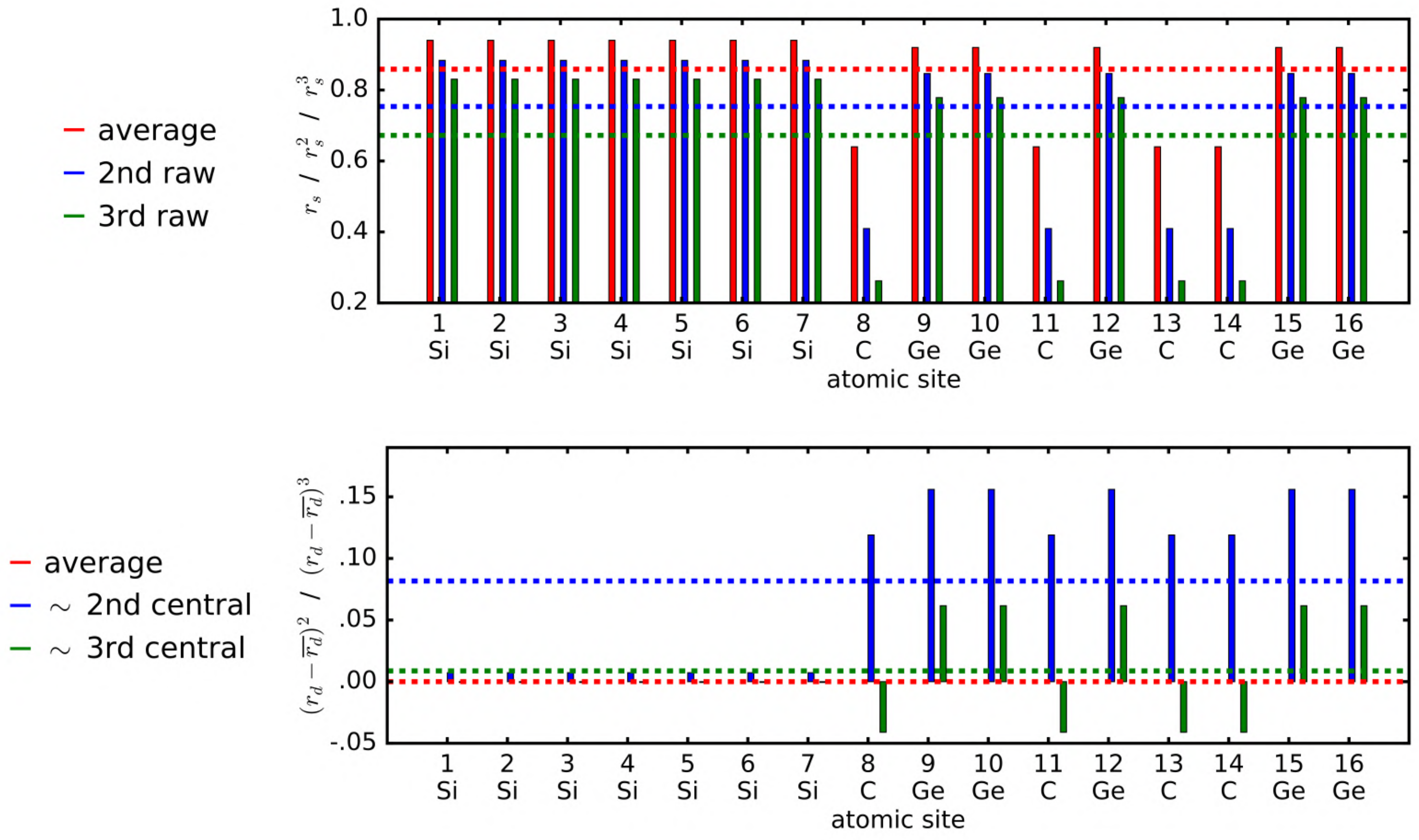


Figure 5.2: Illustration of raw and central moments. Top: the scheme of Eq. 5.2 applied to the atomic local primary feature r_s for $q = 1,2,3$ for an exemplary ternary material of the dataset **T**. Bars indicate the contributions at the individual atomic sites i. Dotted lines mark the resulting *raw* moments (for $q = 1$, this is equal to the average of Eq. 5.1). Bottom: analogously, the contributions to Eq. 5.3 for the feature r_d and $q = 2,3$ and the resulting *central* moments. Note, that the power $1/q$ is not applied here and r_d is used instead of r_s as differences in the contributions are more prominent. The considered sample is $C_4Si_7Ge_5$ in the arrangement 1111111022020022 (see Sec. 7.2).

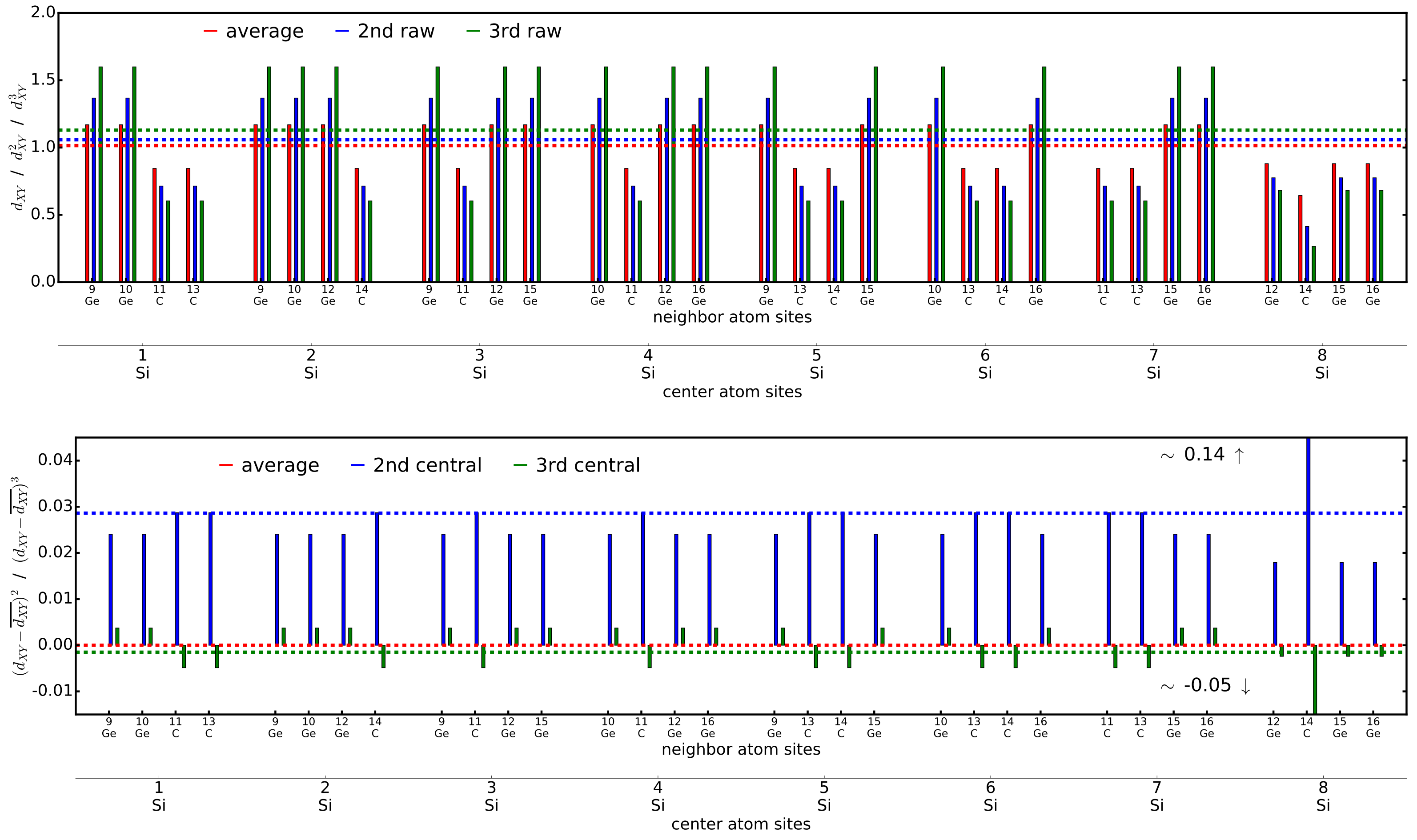


Figure 5.3: Illustration of raw and central moments for the pair local primary feature d_{XY}, similar to Fig. 5.2. Top: contributions and resulting raw moments for $q = 1,2,3$, Bottom: contributions and resulting central moments for $q = 2,3$ (again, without applying $1/q$), grouped by the four neighbors of each atom.

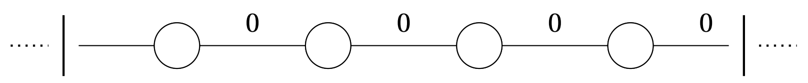
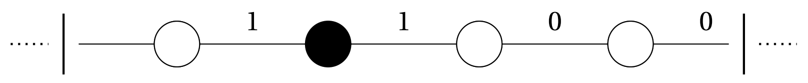
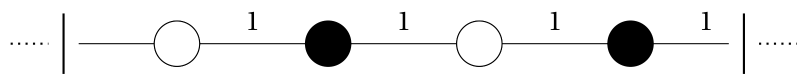
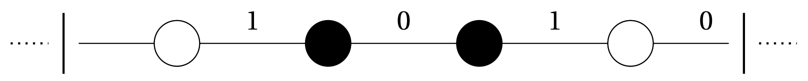
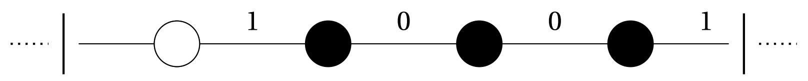
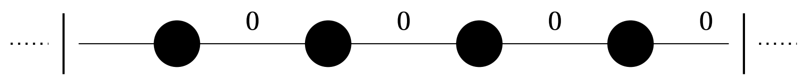
Comp.	Arrangement and contributions	$(\Delta F)_k$
A_4		0
A_3B_1		2
A_2B_2		4
A_2B_2		2
A_1B_3		2
B_4		0

Table 5.1: Illustration of the nearest-neighbor differences from Eq. 5.4 for a binary 4-atomic linear chain with varying composition (first column) and arrangement. The second column visualizes the arrangement in the one-dimensional unit cell with the contributions at the four bonds (considering periodicity), the third contains the resulting value of $(\Delta F)_k$.

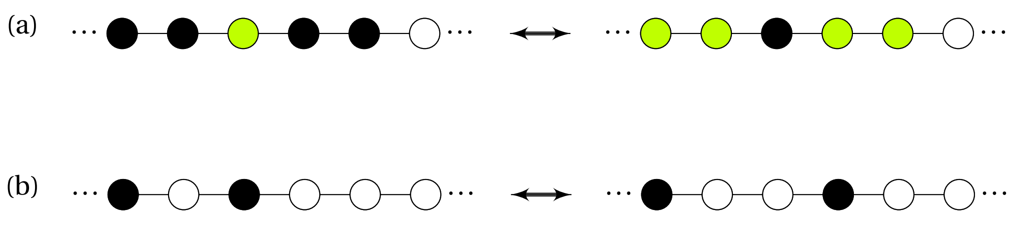


Figure 5.4: Two one-dimensional systems with a 6-atomic unit cell illustrating the limitations of averaged nearest-neighbor differences $(\Delta F)_k$. (a) A three-component system with atoms A, B and C in black, white and green. An exchange $A \leftrightarrow C$ changes the total structure of bonds and hence $(\Delta F)_k$. (b) A two-component system with atoms A and B in black and white. At fixed composition, the arrangement of atoms is varied. It changes the bonds locally but keeps their types in the total system. Thus, $(\Delta F)_k$ does not change.

Averages over Products: As an additional idea, the supercell averages can be generalized further by considering nearest neighbor products, defined by

$$(\Pi F)_{sk} \equiv \frac{1}{m} \sum_{\gamma=(i,j)\in s} \left| f_k(i) \cdot f_k(j) \right| . \tag{5.5}$$

This definition is related to the average *correlation* between nearest neighbors in an atomic property f_k. We test this variant on the octet binaries **O** in Sec. 8.5.

Discussion of the Scheme: The presented scheme of averaging f_k over the supercell is a somehow natural choice to generate a global fingerprint. Its advantages become also clear if one thinks of two different ways to define a feature space based on the local quantities $\{f_k\}$:

- One could directly use the available local primary features $f_k(\gamma)$ as columns of **X**. A row of **X** could be defined by

$$\begin{pmatrix} f_k(\gamma_1) & f_k(\gamma_2) & \dots & f_l(\gamma_1) & f_l(\gamma_2) & \dots \end{pmatrix}, \tag{5.6}$$

  where γ_i are the distinct building blocks that are present in the *total data-set.* If, for instance, f_k is an atomic radius r_s then the columns would refer to r_s(C), r_s(Si), r_s(Zn) etc. if atoms of species C, Si and Zn etc. are in the considered materials set. For a specific structure s, an entry would take the value of $f_k(\gamma_1)$ if $\gamma_1 \in s$ and be equal to 0 if $\gamma_1 \notin s$. Obviously, it is difficult to find interpretable low-dimensional descriptors $\mathcal{D}$ from this, consisting of a few selected columns of **X** only: first, a descriptor component $f_k(\gamma_i)$ would be uninformative to the physics of a structure that does not contain γ_i at all. Second, the selection of a column unfavorably would be dominated by the statistically accidental majority of structures that contain the respective γ_i.

- Alternatively, the columns of **X** could refer to the f_k at *fixed crystal sites.* If the sites are enumerated by $i = 1, 2, 3, \dots$, a row of **X** then could be defined by

$$\begin{pmatrix} f_k(\gamma(1)) & f_k(\gamma(2)) & \dots & f_l(\gamma(1)) & f_l(\gamma(2)) & \dots \end{pmatrix}, \tag{5.7}$$

  where f_k and f_l are different primary features and $\gamma(i)$ marks the building block at the site i. A low -dimensional descriptor $\mathcal{D}$ would select a few sites i which is only physically meaningful if the sites play a distinct role. Naturally, $\mathcal{D}$ would hardly generalize to materials of a different material class.

Note, that our averages are either gross level or molecular- fragment descriptors by the classification scheme of Tab. 4.1, depending on whether atomic and bulk properties, or properties of small, molecule-like clusters like dimers and tetrahedrons are considered. We further remark that our feature engineering scheme mostly satisfies the requirements from Sec. 4.3: (1) As the averages run over all atom sites, invariance with respect to the atom indexing is satisfied. Invariance with respect to unit cell translations and rotations is satisfied as well since these do not affect the pattern of building blocks γ. (2) Supercell averages of *atomic* properties lack uniqueness because they do not consider atomic positions $\{\mathbf{r}_i\}$. If, for instance, two ternary materials of equal composition K differ in atomic arrangement σ, they will be described by the same global feature, although the target **P** might vary. As already discussed, averages over next-neighbor differences and similarly over pair features account for such arrangement effects only partially, so they are also not strictly unique. In Sec. 8.1.1 it is explained that averages over tetrahedron features neither are unique. (3) Single features are not necessarily correlated to the target **P**, which follows from their lack of uniqueness. The total descriptor $\mathscr{D}$ however will be correlated to **P** by construction as a fitted *linear* model (if it is useful).[3] (4) The scheme is generalizable to any periodic system. (5) Computational simplicity is reached, because the calculation of the local primary features and their processing are less effortful than calculations of a total material. Simplicity in regard to human understanding is satisfied by ordinary averages ($q = 1$) as they have a simple expression. Higher moments obviously might be harder to interpret.[4]

5.2.2 Augmenting the Feature Space

In analogy to [1, 2] we progressively enlarged the feature space **X** in the manner of symbolic regression. Global primary features are modified and combined to *derived* features by applying clearly defined sets of mathematical operations $\mathscr{O}_i$. This augmentation of the feature space includes non-linearities to the feature space and, as addressed in Sec. 2.3.2, considers relations between the primary features. We proceed in a number of combinations of increasing complexity, leading to several *tiers* $\mathscr{T}_i$ of derived features. A scheme for this is the following:

1. First, the global primary features $\{F_k\}$ are generated from the local primary features $\{f_k\}$ by Eqs. 5.1, 5.2, 5.3, 5.4. We call these tier 0, $\mathscr{T}_0$, and build the first columns of **X** by them.
2. Next, **X** is extended by derived features following two substeps:
   (a) Binary operations from $\mathscr{O}_{2a} = \{(\cdot + \cdot), |\cdot - \cdot|\}$ are applied to all (allowed) feature pairs from $\mathscr{T}_0$.
   (b) Unary operators from $\mathscr{O}_{2b} = \{(\cdot)^n, \exp(\cdot), 1/\exp(\cdot), \log(|\cdot|), 1/\log(|\cdot|)\}$, $n \in \{\pm 1/3, \pm 1/2, \pm 2, \pm 3\}$ are applied to features from $\mathscr{T}_0$ and those generated in step 2 (a). These features yield tier 1, $\mathscr{T}_1$, which is added to **X**.
3. Further, all pairs of features from $\mathscr{T}_0$ and $\mathscr{T}_1$ up to now are combined by products, $\mathscr{O}_3 = \{(\cdot * \cdot)\}$. This way, tier $\mathscr{T}_2$ is formed and also added to **X**.

Thus, the final feature space **X** is obtained from the union of all considered tiers. We add here various remarks on the details of this scheme: (1) For simple materials where the map from local to global primary features is not necessary, $\mathscr{T}_0$ consists of the primary features $\{f_k\}$. (2) Step 2 (a) is intermediate and we do not include its features to **X**. The reason for this is that these are linear combinations of two primary features already represented by two-dimensional descriptors with features from $\mathscr{T}_0$.[5] (3) At step 2 (a) only physically meaningful combinations are considered, for example, we include sums of two energies but exclude sums of an energy and a distance. (4) Operations containing exp and log require dimensionless arguments. When applying them to the

[3] We point here at the connection between minimizing the linear correlation $\rho_{i,j}$ and the ℓ_2-norm.

[4] For completeness, we note here that continuity and differentiability with respect to atomic coordinates are not given from the trivial reason that atomic coordinates are not considered explicitly by the features.

[5] Of course, the two-dimensional descriptor is more general: it contains also the coefficients c_1 and c_2 that typically do not satisfy $c_1 = c_2$ or $c_1 = -c_2$ equivalent to $(\cdot + \cdot)$ and $|\cdot - \cdot|$, respectively.

features as they are, we implicitly assume a normalization constant hence (see Sec. 8.1.2). (5) The absolute values of differences at substep 2 make these combinations symmetric with respect to an exchange of subtrahend and minuend. (6) For operations requiring positive arguments (log, $(\cdot)^{1/2}$ etc.) the absolute value is taken before. For $(\cdot)^{1/3}$, we actually consider $\text{sign}(\cdot)(|\cdot|)^{1/3}$. (7) In step 3, the product combinations can quickly lead to a combinatorial explosion such that the size of **X** becomes too large to be computationally tractable. To avoid this problem, we define a smaller restricted tier $\mathscr{T}_2^r$ of products where one factor is taken from $\mathscr{T}_0$ and the other from $\mathscr{T}_1$.

Table 5.2 additionally illustrates this augmentation scheme by examples for the ML on the group-IV ternaries **T**. These rely on the local primary features which are further explained in Tables 8.1 and 8.2. Finally, note that we generally will use the symbols $\mathscr{T}_i$ as a shorthand notation for feature spaces made up from tiers *up to* $\mathscr{T}_i$.

Tier	Examples
$\mathscr{T}_0$	$\Delta r_s,\ \tilde{d}^2_{XY}$
$\mathscr{T}_1$	$\frac{1}{\sqrt{\|a_{XX}-d_{XY}\|}},\ \log(r_p + \tilde{r}_s^2)$
$\mathscr{T}_2$	$Z \cdot r_d,\ \frac{\exp(r_s)}{\|E_{m,b}-E_{m,t}\|}$

Table 5.2: Examples for features included in the different tiers, for the ML on the dataset **T**. Note, that the division in the second listed feature on $\mathscr{T}_2$ derives from combining $(\cdot * \cdot)$ and $(\cdot)^n$ in the second factor.

5.3 Set-Up of LASSO and SISSO

In this work, we apply several dimensionality reduction methods $\mathscr{M}$ to identify a pre-selection I_{ℓ_0} of the most relevant features. Most of our analyses either use the elaborate methods LASSO or SISSO that were explained in Secs. 2.2.5, 4.4.1 and 4.4.2. Their actual set-up in this work is described in the following.

5.3.1 Set-Up of LASSO

Similar to Ref. [1], LASSO is applied stepwise at decreasing hyperparameter λ. It is varied on the sequence

$$\Lambda = \left\{2^{i/10} | i = i_\text{m}, i_\text{m} - 1, i_\text{m} - 2 \ldots\right\},\ i \in \mathbb{N} \tag{5.8}$$

where the maximum $\lambda_\text{m} = 2^{i_\text{m}/10}$ (at which all c_i have to be zero) is determined empirically, depending on the learning task. The iteration is stopped when the subset I_{ℓ_0} has reached the size $\tilde{M} = 30$. For this value an exact solution of the ℓ_0-problem is still computationally feasible for descriptors up to the dimension $\Omega = 5$. Note, that it might happen at the decrease of λ that coefficients c_j that already were non-zero for larger λ flip back to zero at smaller λ. By our definition, the indices j will be kept in the preselected set I_{ℓ_0} in that case. Figure 5.5 illustrates the progression of the coefficients c_j at the decrease of λ. In this example, energy of mixing E_{mix} of the data-set $\mathbf{T}$ was learned by LASSO-LARS by descriptors $\{\mathscr{D}\}$ of dimension $\Omega \leq 5$. c_i of features that are selected to the $\{\mathscr{D}\}$ in the subsequent exhaustive ℓ_0 search are indicated by colored lines. Three interesting characteristics can be seen in this figure:

- As said above, some of the coefficients turn back to zero for decreasing λ. This, for instance, can be observed at the yellow line for a feature considered in $\{\mathscr{D}\}$ and at the green-gray line for a feature that is just selected to I_{ℓ_0}.
- The descriptors $\{\mathscr{D}\}$ are not made up from the Ω features selected to I_{ℓ_0} first. For instance, the feature appearing as the second at $\log_2(\lambda) = -2$ (green-gray line) is not considered in any of $\{\mathscr{D}\}$.
- The coefficient magnitude indicates the amount the corresponding feature contributes to the prediction at a given λ. This could be used to conclude on the relevance of the features, similar to the concept of *factor loading* in explanatory factor analysis [106], a common method in social sciences.

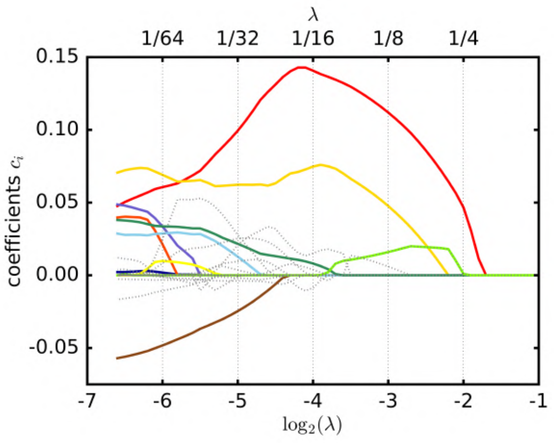


Figure 5.5: Example for the progression of the coefficients c_i at feature selection with LASSO. The hyperparameter λ is decreased sequentially on the grid Λ, i. e. from right to left. For scaling reasons, $\log_2(\lambda)$ is used on the x-axis, the corresponding values of λ are indicated in the top of the figure. Colored lines indicate coefficients of features contained in the descriptors $\{\mathscr{D}\}$ up to $\Omega = 5$. Gray lines refer to features selected to I_{ℓ_0} but not contained in $\{\mathscr{D}\}$. The yellow and green-gray coefficients "vanish" at decreasing λ. Learning task: E_{mix} of $\mathbf{T}$ with feature space $\mathbf{X}_{E,2}$ on complexity $\mathscr{T}_2$, using LASSO-LARS.

5.3.2 Set-Up of SISSO

SISSO is employed only with the sparsifying operator SO= ℓ_0 in this work. Importantly, we use SISSO as a dimensionality reduction method $\mathscr{M}$ to obtain a subset of the most important features I_{ℓ_0}, i.e. as an alternative to LASSO. This is conceptually different to the original approach of Ref.

[91] where SISSO is used as a learning method $\mathscr{L}$ to directly obtain optimal descriptors. To be consistent with $\mathscr{M}$=LASSO, the size of I_{ℓ_0} is $\tilde{M} = 30$. To provide this, SISSO is iterated until $i = \Omega = 5$ where the individual subsets $\tilde{\mathbf{S}}_i$ comprise $N_{\text{SIS}} = 6$ features. From this I_{ℓ_0} is obtained by taking the *union* of subsets, i.e. I_{ℓ_0} is equal to the fifth subspace of SISSO, $I_{\ell_0} = \mathbf{S}_5 = \cup_{j=1}^{5} \tilde{\mathbf{S}}_j$.[6] As an alternative approach, we keep the subsets $\tilde{\mathbf{S}}_j$ separated. This means that in the subsequent exhaustive ℓ_0 search only features from different $\tilde{\mathbf{S}}_j$ are combined, i.e. for dimension Ω all index tuples from $\tilde{\mathbf{S}}_1 \times \cdots \times \tilde{\mathbf{S}}_\Omega$ are considered as candidate descriptors. While the developed code has both of these approaches implemented (see Sec. 6.3.3), in the analyses of Chapter 8 descriptors are always found from the union of subsets $\cup_{j=1}^{5} \tilde{\mathbf{S}}_j$. Finally note that we discard the up to Ω-dimensional descriptors that are found at the iteration of SISSO. We only utilize the subspace $\mathbf{S}_5$ or the subsets $\{\tilde{\mathbf{S}}_j\}$ for the subsequent ℓ_0 search, completely analogous to the other dimensionality reduction methods $\mathscr{M}$.[7]

5.4 Model Selection Strategies

Developing robust approaches to define what makes a model actually the best and how to assess it is a critical step. In this section, we first describe the model selection strategy of Ref. [1] and explain in which way its model evaluation by cross-validation can be improved. We then will develop two more elaborate alternative strategies that include cross-validation already at model selection in order to improve the model robustness and generalizability. Although these strategies can be combined with any learning method $\mathscr{L}$, we present them in the context of either $\mathscr{L}$ =LASSO+ℓ_0 or $\mathscr{L}$ =SISSO+ℓ_0. For brevity, we will often omit the descriptor dimension Ω in this section: $\mathscr{D}$ has to be understood as the set of descriptors $\{\mathscr{D}_1, \ldots, \mathscr{D}_\Omega\}$.

5.4.1 Cross-Validation for Model Validation

In the approach of Ref. [1], the optimal descriptor $\mathscr{D}^{\text{all}}$ ($\Omega = 1, \ldots, 5$) is found by applying the learning method $\mathscr{L}$ =LASSO+ℓ_0 to the total data set, including both model selection and fitting of the coefficients (see Fig. 5.6). In a subsequent CV procedure, the LASSO+ℓ_0 approach is repeatedly applied in exactly the same manner to the individual training sets $U_{\text{tr}}^{(i)}$, i.e. again employing *both* model selection and coefficient fitting. This yields a set of optimal descriptors $\{\mathscr{D}^{(i)}\}_{i=1}^{N_{\text{CV}}}$ that generally will differ between the iterations and, in particular, will not coincide with $\mathscr{D}^{\text{all}}$ from the total set. A statistics of the iterations i satisfying $\mathscr{D}^{\text{all}} = \mathscr{D}^{(i)}$ is used as an indicator for the stability of $\mathscr{D}^{\text{all}}$, i.e. its robustness with respect to the randomness in data used for training: if it was selected in most of the iterations, $\mathscr{D}^{\text{all}}$ is stable. If, conversely, it is selected in few iterations, there are many alternative models of similar predictive performance and hence $\mathscr{D}^{\text{all}}$ is unstable. Table 5.3 gives an example for this procedure for predicting the lattice constant a of the data set $\mathbf{T}$ with the feature space $\mathbf{X}_a^2$ on the high complexity level $\mathscr{T}_2^{\text{r}}$. The 1-dimensional descriptor $\mathscr{D}_1^{\text{all}}$ is found in 47 of the 50 iterations and hence quite stable whereas the rather unstable $\mathscr{D}_4^{\text{all}}$ is only found in 14 iterations, closely followed by an alternative $\mathscr{D}_4^{'}$ that is the best in 12 iterations.

This CV procedure is also used to estimate the expected training error err^{tr} (Eq. 2.27) and generalization error err^{te} (Eq. 2.25) for the descriptor $\mathscr{D}^{\text{all}}$. In Ref. [1], this is done by averaging the individual training and test errors of the iterations that satisfy $\mathscr{D}^{\text{all}} = \mathscr{D}^{(i)}$ as only in these cases the

[6]Note, that occasionally a feature is considered in several subsets $\tilde{\mathbf{S}}_i$. I_{ℓ_0} comprises less than $\tilde{M} = 30$ in that cases.

[7]In a very strict terminology, this approach hence is $\mathscr{M} + \mathscr{L} = \text{SISSO}(\ell_0) + \ell_0$.

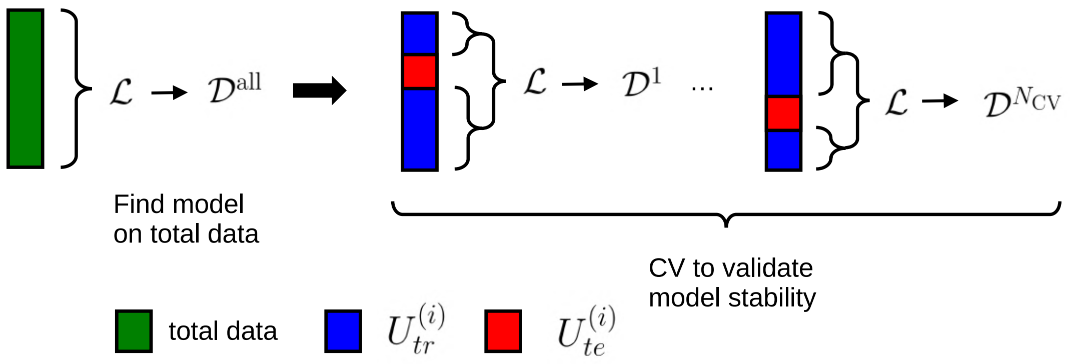

$\mathcal{L} \to \mathcal{D}^{\mathrm{all}}$
$\mathcal{L} \to \mathcal{D}^{1}$ …
$\mathcal{L} \to \mathcal{D}^{N_{\mathrm{CV}}}$
Find model on total data
CV to validate model stability
total data
$U_{tr}^{(i)}$
$U_{te}^{(i)}$

Figure 5.6: Model selection and validation in the scheme of Ref. [1]: first, an optimal model $\mathscr{D}^{\mathrm{all}}$ is found by applying a learning method $\mathscr{L}$ to the total data set (indicated by the green box). Second, a cross-validation run with N_{CV} iterations is performed (marked by the brace). In every iteration, $\mathscr{L}$ is applied to the training split (marked by the blue regions). This yields the models $\mathscr{D}^1, \ldots, \mathscr{D}^{N_{CV}}$ and their individual training and test errors (the latter obtained on the respective test split, marked by the red box). From this CV run, the stability of $\mathscr{D}^{\mathrm{all}}$ can be checked and its average errors can be estimated.

errors refer to $\mathscr{D}^{\mathrm{all}}$. However, this gives *biased* values of err^{tr} and err^{te} because the averages are calculated under the condition that $\mathscr{D}^{\mathrm{all}}$ is the best model on the respective training set $U_{\mathrm{tr}}^{(i)}$. For the 4-dimensional descriptor in Table 5.3, for example, only 14 of the total 50 iterations contribute to the estimate. Typically, the expected training error err^{tr} will be underestimated that way.

The two following approaches could improve the estimates of expected errors by:

1. The average runs over all iterations i, regardless if $\mathscr{D}^{\mathrm{all}}$ or a different descriptor of the same dimension Ω was selected. Mathematically, this number is defined by

$$err^{\mathrm{tr}}(\mathscr{D}_\Omega) \equiv \frac{1}{N_{\mathrm{CV}}} \sum_{i=1}^{N_{CV}} err^{\mathrm{tr}}(\mathscr{D}_\Omega^{(i)}, U_{\mathrm{tr}}^{(i)}), \tag{5.9}$$

   and analogously for err^{te}. However, this will rather estimate how far in performance a certain feature space can get with an *arbitrary* Ω-dimensional descriptor instead of quantifying $\mathscr{D}^{\mathrm{all}}$ with its specific components.

2. The average runs over all iterations i, always considering $\mathscr{D}^{\mathrm{all}}$ from the total data set. In mathematical terms, this reads

$$e\bar{r}r^{\mathrm{tr}}(\mathscr{D}_{\mathrm{all}}) \equiv \frac{1}{N_{\mathrm{CV}}} \sum_{i=1}^{N_{CV}} err^{\mathrm{tr}}(\mathscr{D}_{\mathrm{all}}, U_{\mathrm{tr}}^{(i)}) \tag{5.10}$$

   where $\mathscr{D}^{\mathrm{all}}$ always is fitted to the respective training split $U_{\mathrm{tr}}^{(i)}$.

Let us make several comments on these approaches:

- The relation between the first and the second estimate is $err^{\mathrm{tr}}(\mathscr{D}_\Omega) < e\bar{r}r^{\mathrm{tr}}(\mathscr{D}_{\mathrm{all}})$. This is

Ω	Descriptor	#	training RMSE[mÅ]	test RMSE[mÅ]
1	$\mathscr{D}_1^{\text{all}}$	47	37.6	38.4
	$\mathscr{D}_1^{'}$	3	37.7	45.6
2	$\mathscr{D}_2^{\text{all}}$	30	33.9	32.6
	$\mathscr{D}_2^{'}$	9	33.5	38.7
	$\mathscr{D}_2^{''}$	5	33.9	34.6
3	$\mathscr{D}_3^{\text{all}}$	17	31.4	27.5
	$\mathscr{D}_3^{'}$	7	31.2	33.0
	$\mathscr{D}_3^{''}$	6	30.4	36.2
4	$\mathscr{D}_4^{\text{all}}$	14	29.3	25.8
	$\mathscr{D}_4^{'}$	12	28.4	31.2
	$\mathscr{D}_4^{''}$	6	28.9	32.8
5	$\mathscr{D}_5^{\text{all}}$	8	28.3	22.7
	$\mathscr{D}_5^{'}$	<5	27.6	28.9

Table 5.3: Stability analysis from CV (L10%OCV with $N_{\text{CV}} = 50$) for predicting a of the dataset **T** (feature space $\mathbf{X}_a^2$ on complexity $\mathscr{T}_2^r$, $\mathscr{M}$=LASSO-LARS). The table shows the results for descriptors up to $\Omega = 5$. Listed are the number of CV iterations in which a descriptor was obtained as the best and the average training and test RMSE. $\mathscr{D}_i^{\text{all}}$ indicates the descriptors that were found by $\mathscr{S}_{\text{all}}$ on the total data-set, $\mathscr{D}_\Omega^{'}$ and $\mathscr{D}_\Omega^{''}$ mark the most frequent alternative descriptors. Descriptors with fewer than 5 occurrences are not listed. Average errors are calculated by averaging over the cases where a descriptor was the best, i.e., for instance, over 14 CV iterations for $\mathscr{D}_4^{\text{all}}$.

because in the iterations i where $\mathscr{D}^{(i)} \neq \mathscr{D}^{\text{all}}$ is selected, obviously

$$err^{\text{tr}}(\mathscr{D}^{(i)}, U_{tr}^{(i)}) < err^{\text{tr}}(\mathscr{D}^{\text{all}}, U_{tr}^{(i)}) \tag{5.11}$$

holds. Thus, some of the terms in Eq. 5.9 are smaller than those in Eq. 5.10 which directly implies the relation.

- The second approach estimates the expected test error similar to Eq. 5.10. This number $\bar{err}^{\text{te}}(\mathscr{D}_{\text{all}})$ has to be interpreted with caution, however: $\mathscr{D}_{\text{all}}$ is fixed a priori in all iterations, so this procedure does not mimic the full learning by $\mathscr{L}$ only on $U_{tr}^{(i)}$ (see Sec. 2.4.3 on CV of two-step approaches). Moreover, the choice of $\mathscr{D}_{\text{all}}$ is already based on the total data-set, i.e. it already has used the information in the test splits $U_{tr}^{(i)}$. Hence, $\bar{err}^{\text{te}}(\mathscr{D}_{\text{all}})$ will not provide a clean estimate of the generalization error.

- This concerns also the terminology: taking $\mathscr{D}_{\text{all}}$ fixed actually is not a cross-validation of the learning method $\mathscr{L}$ but some kind of bootstrapping [107]. From this reason, we will refer to the procedure of calculating $e\bar{r}r^{\text{te}}$ and $e\bar{r}r^{\text{tr}}$ by Eq. 5.10 as *sanity check* (SC).
- Technically, in this CV procedure the test error is calculated only for the best model $\mathscr{D}^{(i)}$ at a iteration i. Thus, the contributions $err^{\text{te}}(\mathscr{D}_{\text{all}}, U_{\text{tr}}^{(i)})$ required to calculate $e\bar{r}r^{\text{te}}(\mathscr{D}_{\text{all}})$ by the second approach (Eq. 5.10) are missing if $\mathscr{D}^{(i)} \neq \mathscr{D}_{\text{all}}$. These could be simply "patched" by fitting $\mathscr{D}_{\text{all}}$ to $U_{\text{tr}}^{(i)}$ and validating it on $U_{\text{te}}^{(i)}$ for that cases. Equivalently, $e\bar{r}r^{\text{te}}(\mathscr{D}_{\text{all}})$ can be obtained using N_{CV} new partitions.

From this background, we decided to modify the approach of Ref. [1] to the following model selection strategy $\mathscr{S}^{\text{all}}$:

1. An optimal model $\mathscr{D}^{\text{all}}$ is determined by applying the learning method $\mathscr{L}$ to the total data set.
2. The stability is evaluated by the frequency of $\mathscr{D}^{(i)} = \mathscr{D}^{\text{all}}$ in a CV procedure where $\mathscr{L}$ is applied to the respective training sets $U_{tr}^{(i)}$.
3. Expected training and test errors of $\mathscr{D}^{\text{all}}$ are estimated by the second approach of Eq. 5.10 (i.e. as sanity check).
4. Additionally, $\mathscr{D}^{\text{all}}$ is evaluated on a *hold-out set* $\mathbf{S}_{\text{ho}}$ that was excluded from the total data set in previous.

5.4.2 Cross-Validation for Model Selection

By contrast to the above, our new strategies $\mathscr{S}_{\text{tr}}^{\text{CV}}$ and $\mathscr{S}_{\text{te}}^{\text{CV}}$ use cross-validation for the model *selection.* Both of them consist of the same three steps that are also illustrated in Fig. 5.7:

1. Determine a pre-selection of candidate models $\{\mathscr{D}^{(j)}\}$, for instance by taking the best models at the individual iterations i from a CV procedure.
2. Calculate estimates of expected training and test error, $e\bar{r}r^{\text{tr}}$ and $e\bar{r}r^{\text{te}}$ by Eq. 5.10 for all candidates $\{\mathscr{D}^{(j)}\}$.
3. Select the best candidate from $\{\mathscr{D}^{(j)}\}$ that either minimizes $e\bar{r}r^{\text{tr}}$ for strategy $\mathscr{S}_{\text{tr}}^{\text{CV}}$ or $e\bar{r}r^{\text{tr}}$ for strategy $\mathscr{S}_{\text{te}}^{\text{CV}}$, i.e. determine

$$\underset{\mathscr{D}^{(j)}}{\operatorname{argmin}}\, \mathrm{e\bar{r}r}_{\text{tr}}(\mathscr{D}^{(j)}) \quad \text{or} \quad \underset{\mathscr{D}^{(j)}}{\operatorname{argmin}}\, \mathrm{e\bar{r}r}_{\text{te}}(\mathscr{D}^{(j)}). \tag{5.12}$$

Let us explain the three steps in more detail: the first step is necessary if the number of possible models is large or even huge (such as for the typical candidate matrices $\mathbf{X}$ in this work). By restricting to a set $\{\mathscr{D}^{(j)}\}$ of a manageable amount, the computational effort for the subsequent fitting step is reduced drastically. Although the preselection could be specified manually, e. g. based on physical intuition, here we propose to determine it by repeatedly applying the learning method $\mathscr{L}$ to the random training splits $U_{tr}^{(i)}$. This is equivalent to the CV procedure for model assessment in Ref. [1] and strategy $\mathscr{S}_{\text{all}}$ and yields a best model $\mathscr{D}^{(i)}$ in every iteration i - best in terms of training error for $\mathscr{S}_{\text{tr}}^{\text{CV}}$ and in terms of test error for $\mathscr{S}_{\text{te}}^{\text{CV}}$. The candidates $\{\mathscr{D}^{(j)}\}$ are then defined as the set of

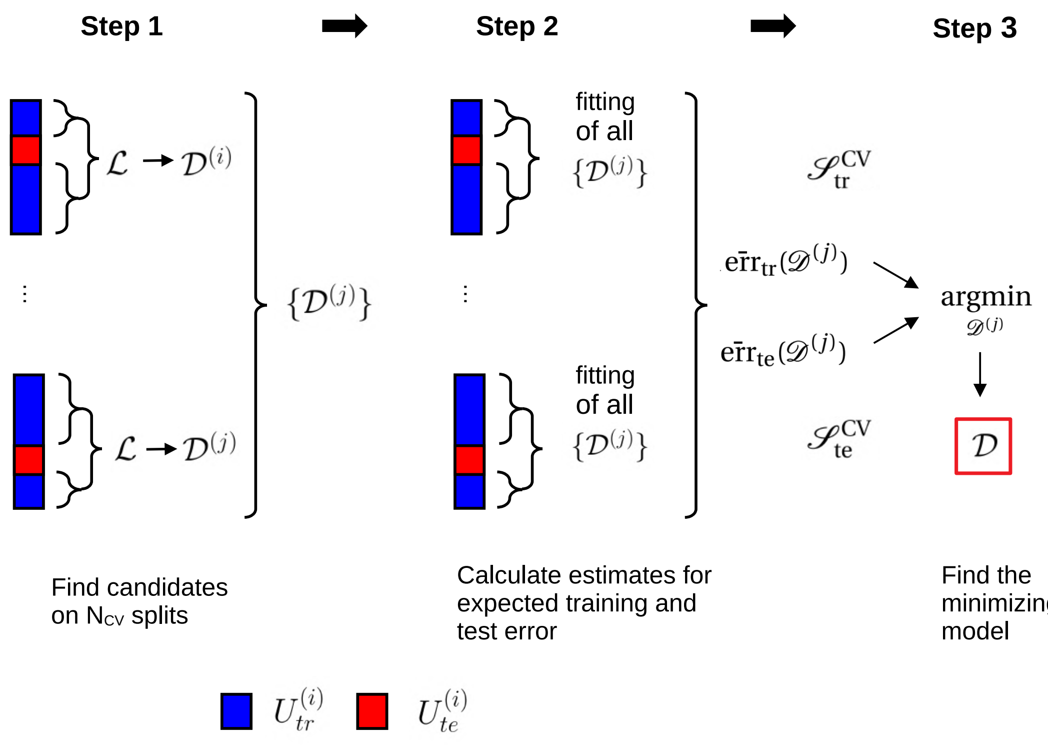

Step 1
Step 2
Step 3
$\mathcal{L} \rightarrow \mathcal{D}^{(i)}$
$\mathcal{L} \rightarrow \mathcal{D}^{(j)}$
$\{\mathcal{D}^{(j)}\}$
fitting of all $\{\mathcal{D}^{(j)}\}$
fitting of all $\{\mathcal{D}^{(j)}\}$
$\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr}}$
$\bar{\mathrm{err}}_{\mathrm{tr}}(\mathscr{D}^{(j)})$
$\bar{\mathrm{err}}_{\mathrm{te}}(\mathscr{D}^{(j)})$
$\mathscr{S}^{\mathrm{CV}}_{\mathrm{te}}$
argmin $\mathscr{D}^{(j)}$
$\mathcal{D}$
Find candidates on N_{CV} splits
Calculate estimates for expected training and test error
Find the minimizing model
$U^{(i)}_{tr}$
$U^{(i)}_{te}$

Figure 5.7: Scheme of the model selection strategies $\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr}}$ and $\mathscr{S}^{\mathrm{CV}}_{\mathrm{te}}$ (analogously to Fig. 5.6). In the first step, iteratively a set of candidate models $\{\mathscr{D}^{(j)}\}$ is determined. In each iteration, the learning method $\mathscr{L}$ is applied to the respective training set. In step 2, estimates of expected training and test errors $e\bar{r}r^{\mathrm{tr}}$ and $e\bar{r}r^{\mathrm{te}}$ of all candidates $\{\mathscr{D}^{(j)}\}$ are calculated by repeatedly fitting the $\{\mathscr{D}^{(j)}\}$ to the training sets, and evaluating them on the test sets. In step 3, the candidate minimizing the average training ($\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr}}$, upper path) or average test error ($\mathscr{S}^{\mathrm{CV}}_{\mathrm{te}}$, lower path) is determined as the best model $\mathscr{D}$.

most frequent models. Note that strictly speaking this does not serve to *validate* a model, so the term *resampling* [47, 107] procedure instead of cross-validation is better suited here. The second step is done by again iterating over N_{CV} splits where all candidates are fitted to the training set $U^{(i)}_{tr}$ by LLSR and evaluated on the test set $U^{(i)}_{te}$ in every iteration. As already addressed above, we prefer to call this procedure sanity check (SC) as the models are kept fixed in regard to the selected components. The decisive difference between strategies $\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr}}$ and $\mathscr{S}^{\mathrm{CV}}_{\mathrm{te}}$ is the use of training or test error, respectively, in the minimization of step 3.

To sum up, the three described model selection strategies differ in how they define what makes a model the best one: in $\mathscr{S}_{\mathrm{all}}$ "best" means "best fit to a single training set"; in $\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr}}$ it means "average best fit on several training sets"; and in $\mathscr{S}^{\mathrm{CV}}_{\mathrm{te}}$ it means "average best generalizability to several test sets". By construction, $\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr}}$ leads to more robust models compared to $\mathscr{S}^{\mathrm{all}}$ as it considers several fits. It is intended to reduce the influence of the actual choice of the training set, i.e. the variance term in Eq. 2.28 [47]. $S^{\mathrm{CV}}_{\mathrm{te}}$ puts the focus on generalizability as being based on test errors, the estimates for the generalization error. We aim at a better understanding of the differences between these three definitions in Sec. 8.3.

5.5 Methods for Model Assessment

The following presents two approaches that go beyond purely assessing the quality of a model fit in terms of loss: Sec. 5.5.1 presents a measure to quantify the models' ability to capture effects of the atomic configuration in the supercell, and Sec. 5.5.2 proposes measures that consider the informational and algebraic complexity of the descriptors. We point out also to Sec. A.2.3 where we study several numeric examples concerning the behavior of different error metrics at cross-validation.

5.5.1 Capturing Arrangement Effects

For materials with a derivative supercell [108] such as the group-IV ternaries **T**, atomic compositions K can be realized by several, symmetrically inequivalent arrangements s. In order to measure how well a model captures the effects of the arrangement, we define a figure of merit by[8]

$$\mathrm{err}_{\mathrm{MAE,arr}} \equiv \frac{1}{N_K}\sum_K \frac{1}{N_s(K)}\sum_{s\in K}\left|\Delta_K P(s) - \Delta_K f(\mathbf{x}_s)\right|. \tag{5.13}$$

The outer averaging is over the total composition space, the inner over all different atomic arrangements of a fixed composition. Here, N_K denotes the total number of compositions in the data set and $N_s(K)$ the number of arrangements for that K. The summands are the absolute differences between target and prediction, taken with respect to their within-composition mean

$$\Delta_K P(s) \equiv P(s) - \frac{1}{N_{s'}}\sum_{s'\in K} P(s) \tag{5.14}$$

for the target and analogously for the prediction $f(\mathbf{x}_s)$. The idea behind this is to capture only how well $f(\mathbf{x})$ matches the shape of $\mathbf{P}$. Note, that all K that comprise only one arrangement s do not contribute to $\mathrm{err}_{\mathrm{MAE,arr}}$. This happens frequently in the hold-out set $\mathbf{S}_{\mathrm{ho}}$ due to its smaller amount of data. Thus, $\mathrm{err}_{\mathrm{MAE,arr}}(\mathbf{S}_{\mathrm{ho}})$ is not comparable to the value on the training data from statistical reasons.

5.5.2 Measuring Descriptor Complexity

To assess the interpretability and simplicity of the descriptors' symbolic representation, we introduce two measures of complexity (cf. Sec. 2.3.2):

- *Informational complexity* N_{f_k}: the total number of distinct local primary features f_k in $\mathscr{D}$,
- *Algebraic complexity* N_{op}: the total number of operations in $\mathscr{D}$, combining the primary features.

As a natural choice to measure the *total complexity*, we use the sum $\Sigma \equiv N_{f_k} + N_{op}$. Table 5.4 gives an example of these definitions for the 1D and 2D descriptors of Ref. [2] on the octet binaries **O**. Note, that by our convention N_{op} considers the sum operations due to linearly combining descriptor components but neglects the products by the coefficients c_i. Every rise in Ω trivially gives one additional product so their contributions do not add any information to N_{op}.

[8]For the sake of readability, we will often use the abbreviation $\mathrm{MAE}_{\mathrm{arr}}$.

Ω	components			N_{f_k}	N_{op}	Σ
1	$\frac{\text{EA(B)}-\text{IP(B)}}{r_p(\text{A})^2}$			3	3	6
2	$\frac{\text{EA(B)}-\text{IP(B)}}{r_p(\text{A})^2}$	$+$	$\frac{r_s(\text{A})-r_p(\text{B})}{\exp(r_s(\text{A}))}$	4	7	11

Table 5.4: Complexity measures N_{f_k} and N_{op} for the 1D and 2D descriptors reported in [2] to predict the energy difference between rocksalt and zincblende structure of the octet binary data-set **O**.

It is not that straightforward to calculate N_{op} for the descriptors constructed by the scheme of Sec. 5.2. In particular, this regards the averaging from local to global primary features, $f_K \mapsto F_K$, and the distinction between simple averages ($q = 1$), higher moments ($q > 1$) and nearest-neighbor differences. We come to a more precise definition of N_{op} that counts:

- all operations that combine *global* primary features F_k, i.e. all augmenting operations in the sets $\mathcal{O}_i$,
- the sum operations from linear combinations,
- all supercell averages if they involve a raw or central higher moment, i.e. $q \geq 2$,
- the nearest-neighbor differences.

Importantly, the pure averaging step $f_k \mapsto F_k$ itself (i.e. $q = 1$) is ignored as this is done in any case. Only if $q \geq 2$ an implicit further step of operations is performed which needs to be captured by N_{op}. In Sec. 8.1.2 we apply this definition to learning tasks on the ternary materials **T** where it can be understood more clearly.[9]

[9]We studied also tree representations of symbolic regression for the analytical expressions of the descriptors which provide another useful perspective on N_{f_k} and N_{op}. As this goes beyond the scope of this manuscript we only point at the example in the online material [109].

Chapter 6

Implementation of the Machine Learning Program

A main part of this PhD project was to develop a Python machine learning program that implements the methodology from Chapter 5 and is applied to specific problems in Chapter 8. Here, we explain its overall architecture and some general remarks in Sec. 6.1. Important technical details on the implementation of the feature engineering are addressed in 6.2. The implementation of the learning process is described in Sec. 6.3. Additional details are given in Appendix C. Parts of the program are available online [109] where some learning runs on the data-set $\mathbf{T}$ *can be re-run and a further documentation can be found.*

6.1 Program Structure and Workflow

The developed machine learning code implements the proposed methodology in a set of Python modules. It can be run both on local machines and on high-end computing clusters, where in the latter case it is able to deal with huge data matrices comprising hundreds of millions of features in parallel. The code's workflow is visualized in Fig. 6.1 which basically follows from the ML approach sketched in Fig. 5.1.

The most important program modules and their functionality are, in the order of their call:

- `main.py`: main module defining the main variables and sequentially calling the submodules below.
- `definitions.py`: imports the external modules the code depends on, and defines all internal functions and constants.
- `settings.py`: specifies the user-defined general preferences, settings concerning the model selection and assessment, the set-up of the dimensionality reduction and output options.
- `data.py`: loads the already parsed external raw data of the primary features $\{f_k\}$ and the target $\mathbf{P}$ and further prepares it, like generating dictionaries or converting the numeric values.
- `make_samples.py`: prepares and organizes the data-set of material samples $\{s\}$ by generating unique material labels, sorting and categorizing the samples and assigning samples to the hold-out set $\mathbf{S}_{\text{ho}}$.

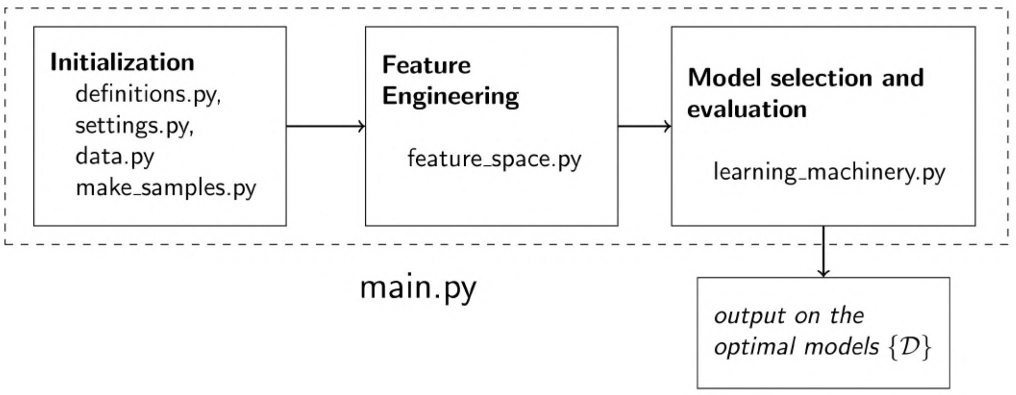


Figure 6.1: Flow diagram and structure of the developed Python code. The main script is `main.py` which calls the described subroutines.

- `feature_space_*.py`: generates the feature space represented by **X** in a pre-defined setup (this is indicated by `*`), regarding the included local primary features, their processing to global primary features and the scheme of algebraic augmentation.
- `learning_machinery.py`: performs the model selection and validation steps and generates the numeric, graphical and LaTeX output of the optimal descriptors $\mathscr{D}$ and competing alternative models.

6.2 Implementation of the Feature Engineering Approach

6.2.1 Generation of Global Primary Features

Within `feature_space_*.py`, the parsed primary features $\{f_k\}$ are loaded to dictionaries $\{\gamma : f_k(\gamma)\}$, mapping all building blocks γ of a certain type to the respective features. For example, the dictionary `dimer_data` contains the distances, binding energies, etc. of the calculated dimers, such as `dimer_data[('C','Ge')] = (1.6,-5.1e5,...)` for the dimer γ=(C,Ge).

In order to calculate the averages by Eq. 5.1, 5.2 and 5.3, one first must identify all building blocks γ in a structure s. Identifying all atomic building blocks $\gamma = \{\alpha\}$ in s is trivial. To find all pair clusters $\gamma = \{(\alpha,\beta)\}$ in s, we in previous determined a list of all nearest neighboring sites in the *ideal* lattice, i.e. independent of a specific material. These were obtained by a brute-force search [110] iterating over all sites i. All other sites j with the nearest neighbor distance R_{NN} to i were kept in the list. Periodic boundary conditions were taken into account by replicas and a shell of a small width θ and of radius R_{NN} was considered to provide stable numerical comparisons. At the actual run of the code, the pair clusters $\gamma = \{(\alpha,\beta)\}$ in a *specific* material s are determined by occupying the sites in the list with the respective species. If, for instance, the list species the sites $(1,2,3,5)$ as neighbors of site 9, then `(Ge, Ge, Ge, Si)` would be identified as the neighboring species of atom `Sn` in a specific structure.

For the tetrahedral clusters used for the data-set **T**, similarly a list of all clusters $\{(\alpha,\beta_1,\beta_2,\beta_3,\beta_4)\}$ in the ideal structure was derived. It simply groups a center atom α and its four nearest neighbors

```
1  def ave_feature(fk_dict,lbl,q1,q2):
2    # input: dictionary of primary features of one type,
3    # structure label lbl, raw and central HM order q1, q2
4    # output: supercell average / HM
5
6    # get the building blocks in the material by a map
7    # considering the ideal structure
8    gammas = map_phi (lbl)
9
10   F = np.mean([fk_dict[gamma]**q1 for gamma in s])
11   if q2 == 1:
12     return F
13   else:
14     F = np.mean([(f_k - F)**q2 for gamma in s])
15     F = np.sign(F) * np.abs(F)**(1./q2)
16     return F
```

Listing 6.1: Python code to calculate the supercell averages and raw or central higher moments.

β_i. A specific structure then is equipped by a list of look-up keys for the dictionary of features like `('C','Si','Si','Ge','Ge')`, in the convention to put the center atom first and then the surrounding atoms, sorted by nuclear number.

The function `ave_feature` shown in Listing 6.1 uses a dictionary `fk_dict` = $\{\gamma : f_k(\gamma)\}$ of the primary features and a unique label `lbl` that encodes information on the structure s, to actually calculate the supercell averages by Eqs. 5.1, 5.2 and 5.3.[1] It calculates ordinary averages as well as raw and central moments. The order of the moments is defined by the arguments q_1 (raw) and q_2 (central) where the choice $q_1 = 1$ and $q_2 = 1$ returns the simple average, $q_2 = 1$ and any $q_1 > 1$ a *raw* moment, and $q_1 = 1$ and any $q_2 > 1$ a *central* moment. As we do not combine raw and central moments, $q_2 > 1$ implies $q_1 = 1$. `F` in line 10 hence automatically provides the ordinary average F_{sk} of f_k that is required for the centralization of the central moment in line 14. The absolute value in line 15 ensures that all values are real after the operation `**(1./q2)` for odd values of q_2. The sign function in the same line is applied by our convention to capture the sign before `**(1./q2)` in that case.

Averages of nearest-neighbor differences by Eq. 5.4 follow the same algorithm. It is just required to generate a dictionary of "pseudo-pair" primary features first, by calculating the absolute differences of all pairs $\{(\alpha,\beta) : f_{k'(\alpha,\beta)}(\alpha,\beta)\} = \{(\alpha,\beta) : |f_k(\alpha) - f_k(\beta)|\}$. For example, `delta_r_d = {('Ge','Sn'):(r_d['Ge']- r_d['Sn']),...}` derives the dictionary of nearest-neighbor differences in atomic radius r_d from the dictionary `r_d`.

6.2.2 Feature Space Augmentation

The feature space augmentation as described in Sec. 5.2.2 is implemented in two separate functions: (1) the function `create_features` generating a single row of $\mathbf{X}$ up to complexity tier $\mathscr{T}_1$, (2) the function `calculate_prods` yielding the features of $\mathscr{T}_2$, i.e. the combinations by products. In the primer, the global primary features $\{F_k(s)\}$ are calculated and assigned to lists according to their

[1] For an example of a label `lbl` of a ternary material, see Fig. 7.7.

```
1  # define a 1D shareable array for multiprocessing
2  x_mp = mp.Array(ctypes.c_double,(N*(M+1))))
3
4  # split samples into N_p chunks
5  ids_split = split_list(range(len(N)),N_p)
6
7  # calculate tier 0 and 1 in parallel
8  pool = mp.Pool(processes=N_p)        # define pool
9  pool.map(create_features,ids_split)  # work through samples
10 pool.close(); del(pool)              # close & delete the pool
11
12 # construct the 2D feature matrix as an np.array
13 x_all = np.frombuffer(x_mp.get_obj()).reshape((N,M+1))
```

Listing 6.2: Parallelization of the feature space augmentation up to tier $\mathscr{T}_1$.

physical dimensions, e. g.

$$\texttt{numbers1}, \texttt{numbers2}, \texttt{energies1}, \texttt{energies2}, \ldots$$

This considers also the higher powers of the physical dimensions that are introduced by the raw higher moments by separate lists. Only features of the same list are then combined by the binary operations $\mathscr{O}_{2a}$ to generate well-defined and physically meaningful expressions.[2] After applying also $\mathscr{O}_{2b}$, features up to $\mathscr{T}_1$ are written to a slice of the shareable array `x_mp` required for the parallelization (see below). If product features are included, the function `calculate_prods` writes the values of the product combinations for one sample s to the respective slice of `x_mp`. Internally, it iterates over a list of the index pairs (k,l) that determines the features to be combined, depending on whether $\mathscr{T}_2$ or $\mathscr{T}_2^r$ is considered (see Sec. 5.2.2). Irrelevant combinations (k,l), such like $\exp(f_k) \cdot \exp(f_k)^{-1}$, are excluded.

Listing 6.2 shows how the generation of $\mathbf{X}$ is parallelized. It relies on the Python package `multiprocessing` (`mp`, [111]) that overcomes certain burdens of the design of Python to be efficiently run on multi-core machines but brings some specialties on the sharing and exchanging of data between several processes. This is why first the special 1D array `x_mp` from the `ctypes` library is assigned, with the size `N*(M+1)` of the flattened data matrix (line 2). Next, in line 5 all material samples are split into N_p chunks, according to the degree of parallelism. A pool of working processes, the central structure of the `mp` package, is defined, and the function `create_features` is executed by `pool.map`, handling through the chunks of samples in parallel (lines 7 - 10). After completion, the 2D array `x_all` of the shape `(N,M+1)`, referring to $\mathbf{X}$, is constructed from `x_mp` (line 13). Here, `np.frombuffer(...)` serves to retrieve the data from `x_mp` in a memory efficient way without double-allocation. The parallelization of `calculate_prods` that is run for tier $\mathscr{T}_2$ or $\mathscr{T}_2^r$ works analogously.

[2]Although all powers of dimensionless numbers could be combined in regard to their physical dimension, we do not intermix them.

6.3 Implementation of Model Selection and Evaluation

The second core part of the code comprises all steps of model selection and evaluation, i. e. the bottom part of Fig. 5.1. As illustrated in the flow diagram of the module `learning_machinery.py` in Fig. 6.2, first the data matrix for the chosen learning method $\mathscr{L}$ is prepared. Next, optimal descriptors are identified by the quick model selection strategy $\mathscr{S}^{\text{all}}$ (see Sec. 5.4), and then evaluated by a sanity check and on the hold-out set. Afterwards, the computationally more costly strategies $\mathscr{S}_{\text{tr}}^{\text{CV}}$ and $\mathscr{S}_{\text{te}}^{\text{CV}}$ are run, providing also the data for the stability analysis of $\mathscr{S}^{\text{all}}$. Finally, the resulting candidates and optimal models are evaluated in a comprehensive way. In the following we will explain the important technical details of these steps.

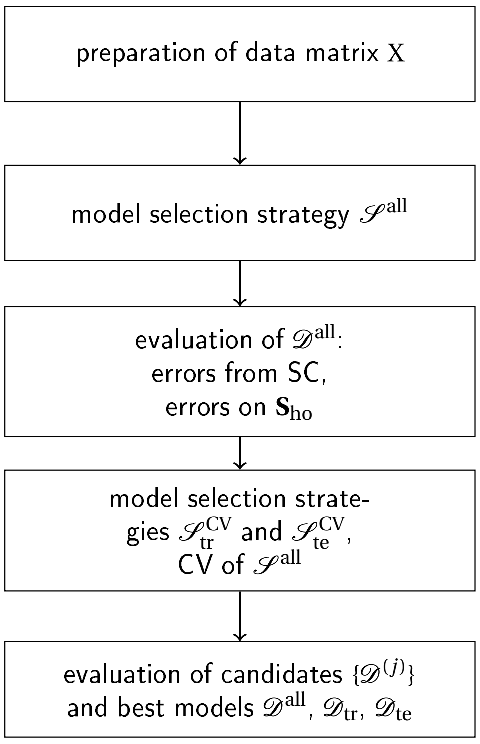

preparation of data matrix X
model selection strategy $\mathscr{S}^{\text{all}}$
evaluation of $\mathscr{D}^{\text{all}}$: errors from SC, errors on $\mathbf{S}_{\text{ho}}$
model selection strategies $\mathscr{S}_{\text{tr}}^{\text{CV}}$ and $\mathscr{S}_{\text{te}}^{\text{CV}}$, CV of $\mathscr{S}^{\text{all}}$
evaluation of candidates $\{\mathscr{D}^{(j)}\}$ and best models $\mathscr{D}^{\text{all}}$, $\mathscr{D}_{\text{tr}}$, $\mathscr{D}_{\text{te}}$

Figure 6.2: Flow diagram of the submodule `learning_machinery`, implementing the steps of model selection and evaluation.

6.3.1 Matrix Preparation

For all learning methods $\mathscr{L}$ except for SISSO, a check-up of the matrix `x_all` is performed in before. This discards all columns with non-finite entries as well as constant, i.e. non-informative, columns. Additionally, also features that take non-finite values on $\mathbf{S}_{\text{ho}}$ are discarded because they are not transferable. If the check-up is passed, mean μ_i and standard deviation σ_i of the respective column are calculated to perform the standardization by

$$\tilde{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mu_i}{\sigma_i}. \tag{6.1}$$

Data-points in $\mathbf{S}_{\text{ho}}$ are not considered in μ_i and σ_i. Note, that these preparation step are parallelized by columns of $\mathbf{X}$.

6.3.2 Implementation of LASSO

The dimensionality reduction by $\mathscr{M}$ =LASSO following the steps described in Secs. 4.4.1 and 5.3.1 is implemented to the function `find_best_ids_LASSO_LARS` (see Listing 6.3). It determines the subset I_{ℓ_0} of the $\tilde{M}$ most relevant features as a list using the LARS solution algorithm (see Sec. 2.2.5). For every $\lambda \in \Lambda$, LASSO-LARS is run in its implementation of the machine learning library *Scikit-learn* [112] with the parametrization denoted in Tab. 6.1 (lines 7 and 8). Features whose coefficients turned to a non-zero value the first time are added to I_{ℓ_0} in lines 16 - 18. Note here three important exceptions: (1) If the fit yields already non-zero c_i at λ_{max}, this parameter was chosen too small and the iteration will break (line 12,13). (2) It might happen that two or more c_i appear at the same λ so that eventually I_{ℓ_0} exceeds the desired $\tilde{M}$. For this case, only the $\tilde{M}$ first entries of I_{ℓ_0} are kept (line 21), although this convention adds some arbitrariness to the scheme since smaller column indices will be favored. (3) The loop will stop at a λ_{min}, even if $\tilde{M}$ is not reached yet, in order to limit the run time.

```
1  def find_best_ids_LASSO_LARS(x,P,**kwargs):
2    cs, I_l0 = [[],[],[]] # coefficients cs, subset I_l0
3
4    # iterate at decreasing hyperparameter
5    for lambda_ in lambdas:
6      if len(set(I_l0)) >= M: break # target M reached
7      clf = linear_model.LassoLars(alpha=2*N*lambda_,kwargs)
8      clf.fit(x,P)
9      cs = clf.coef_[0]
10
11     # maximum lambda too small
12     if lambda_ == lambdas[0] and cs.any() != 0.:
13       return 1
14
15     # add new index to selection
16     for j in range(len(cs)):
17       if cs[j] != 0.0 and j not in I_l0:
18         I_l0.append(j)
19
20   # reduce I_l0 if M is exceeded
21   if len(I_l0) > M: I_l0 = I_l0[0:M]
22
23   return I_l0
```

Listing 6.3: Implementation of the dimensionality reduction method $\mathscr{M}$ =LASSO. The code refers to LASSO-LARS, altering line 7 to `linear_model.Lasso` changes to the coordinate descent algorithm. The variables `I_l0`, `lambda_`, `lambdas` and `M` in the code refer to I_{ℓ_0}, λ, Λ and $\tilde{M}$. The hyperparameter `alpha` of `sklearn.LassoLars` corresponds to $2 \cdot N \cdot \lambda$ in our convention (see line 7). Details on the keyword arguments can be found in the code documentation.

`fit_intercept`	`False`	`max_iter`	3000	`fit_path`	`False`
`normalize`	`False`	`eps`	$2.22 \cdot 10^{-16}$	`verbose`	`True`
`precompute`	`'auto'`	`copy_X`	`True`	`positive`	`False`

Table 6.1: Parametrization of `LassoLars` from the subpackage `sklearn.linear_model` used for the learning tasks of this work. A documentation of the parameters is available online [113].

6.3.3 Implementation of SISSO

The SISSO approach (see Sec. 4.4.2 and also Sec. 5.3.2) is originally implemented in a parallelized Fortran 90 program [114]. It consists of the two modules *FC* for "feature construction", i.e. the feature engineering in a approach similar to Sec. 5.2, and *DI* for "descriptor identification", i.e. the model learning. At least for simple learning tasks, this code could be integrated in our Python program by exporting the feature matrix $\mathbf{X}$ to a file and then run the *DI* module on it. At more difficult tasks, $\mathbf{X}$ may exceed the size to be written in total to hard-disk and thus its highest tier must be constructed by the *FC* module.[3] SISSO's subsets $\tilde{\mathbf{S}}_j$ that determine the most relevant features I_{ℓ_0} in our set-up (see Sec. 5.3.2) can be re-imported in our code to then determine the best models. For our CV based strategies this has to be repeated frequently, bringing a significant overhead due to the slow interface between the codes. Besides that, several other technical differences lead to a cumbersome integration of the Fortran program into our code.

Hence, we developed an independent Python implementation of SISSO. It is realized in the module `SISSO_core.py` with the requirements to be memory efficient, fast and parallelized such that huge data matrices can be handled on computing clusters. This is first reached by calculating the data matrix $\mathbf{X}$ only up to complexity $\mathscr{T}_1$. A standardization of the matrix is not necessary because this implicitly is considered at calculating Pearson correlations (see Eq. 2.17) in the subspace update. If the product tier $\mathscr{T}_2$ / $\mathscr{T}_2^{\mathrm{r}}$ is considered - which drastically increases the size of $\mathbf{X}$ - the features $\mathrm{x}_k \circ \mathrm{x}_l$ are calculated on the fly. The implementation repeatedly calculates their absolute correlation $|\rho((k,l))|$ with the residue, required for the update of each subset $\tilde{\mathbf{S}}_j$, and directly discards them afterwards. Only the lower tiers $\mathscr{T}_0$ and $\mathscr{T}_1$ are stored permanently in the RAM.

Our version of SISSO is parallelized (1) at the iterative search of the subsets $\tilde{\mathbf{S}}_j$, (2) at the ℓ_0-search of the sparsifying operator and (3) at cross-validation. We outline these procedures here and refer to Sections C.1 and C.2 in the appendix for further details on (1) and (3).

1. In the search of the subsets $\tilde{\mathbf{S}}_j$, the absolute Pearson correlations $|\rho|$ between features and the target are calculated in parallel. This is done by distributing matrix columns i (for tiers $\mathscr{T}_{0/1}$) or pairs of columns (k,l) (for product tiers $\mathscr{T}_2$ / $\mathscr{T}_2^{\mathrm{r}}$) to several processes.

2. Our code allows to choose whether the subsets $\tilde{\mathbf{S}}_j$ are united or kept separated for the subsequent ℓ_0-step of the sparsifying operator. Only in the first option the huge number of feature combinations requires a parallelization, namely by distributing all Ω-dimensional combinations of features to the N_{p} processes. At a single task, a LLSR fit is performed for one combination $(i,j,\ldots)$, such that a list of all quadratic losses is generated. The optimal

[3]The lower tiers cannot be calculated by the *FC* module because it does not support for all operations of our feature engineering approach.

Ω-dimensional descriptor then is the combination of minimum loss.

3. Also at cross-validation, our SISSO implementation calculates the correlations $|\rho|$ in parallel. Here, all splits of the CV iteration are treated *simultaneously* which effectively extends the learning task by one dimension. The parallelization is then done by feature index i, i. e. by columns of $\mathbf{X}$.

6.3.4 Implementation of the ℓ_0-step

Having determined the subset I_{ℓ_0} by a method $\mathscr{M}$, the optimal descriptors are obtained by an explicit ℓ_0 step. Its implementation is straight-forward: for a fixed Ω, the loss from LLSR fits is sampled over the grid of all Ω-combinations of indices in I_{ℓ_0} first. Here, the lists `loss_grid` and `ids_grid` are constructed alike, by appending the respective loss and the corresponding combination of matrix indices $(i, j, \ldots)$. At every iteration a temporary reduced matrix `x_2` is generated that comprises only the features $(i, j, \ldots)$ and an additional column full of ones to account for the offset. The position of the minimal loss in the list `loss_grid` is determined by the function `np.argmin`. Finally, it is mapped back to the best Ω-dimensional combination of column indices, i.e. the components of the optimal descriptor $\mathscr{D}_\Omega$.

6.3.5 CV and Model Selection Strategies

The model validation and selection strategies presented in Sec. 5.4 essentially rely on cross-validation, realized as non-exhaustive LPOCV in the applications. The required N_{CV} partitions are managed by an instance of the class `CV_ids_class`. At its initialization, all training splits $U_{tr}^{(z)}$ are drawn randomly to the attribute lists `self.ids_tr`. To ensure that all $U_{tr}^{(z)}$ are different, a new split is generated if the present one is already contained in `self.ids_tr`. The corresponding test set $U_{te}^{(z)}$ is determined by building the complement set. To account for an eventually failed CV iteration and to guarantee for code stability in that case, a re-draw of the partition is implemented.

Our model selection and validation scheme employs three subsequent CV procedures: (1) the sanity check SC of the descriptors from $\mathscr{S}^{all}$, (2) the extensive CV including model selection required for the stability analysis in $\mathscr{S}^{all}$ and the identification of candidates in $\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr/te}}$, and (3) the SC to calculate average errors of the candidates $\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr/te}}$ (see Sec. 5.4). The implementation uses one instance of CV partitions for step (2). A second one is used in step (1) and re-used in (3) to ensure that the average errors of $\mathscr{D}_{\mathrm{all}}$ and the alternative candidates $\{\mathscr{D}^{(j)}\}$ are comparable and not distorted by the effects of randomness and split re-drawing.

Sanity checks at fixed model are efficiently performed in series. The CV procedure including model selection is parallelized, either by material sample s if $\mathscr{M}$ =SISSO or by iteration z if $\mathscr{M}$ $\neq$SISSO (see Sec. C.2 in the appendix). For SCs, the CV and for an evalution on $\mathbf{S}_{\mathrm{ho}}$, i.e. whenever splitted data is used, it is important that the matrix standardization is done with respect to the mean and standard deviation of the training samples in the respective split $U_{tr}^{(z)}$ only [2]. As the magnitude of the trained coefficients c_i depends on that standardization, test and hold-out data similarly must be standardized by the mean and standard deviation of the training data before the model is evaluated on them.

The results from the extensive CV (2) are grouped to categories of the same descriptor and their frequency is counted (recall Tab. 5.3). Descriptors occurring more often than a pre-defined threshold `N_min_CV` are added to the set of candidates $\{\mathscr{D}^{(j)}\}$ of strategies $\mathscr{S}_{\text{tr}}^{\text{CV}}$ and $\mathscr{S}_{\text{te}}^{\text{CV}}$. Individual error data is averaged over the occurrences for a first model assessment (this was problematized in Sec. 5.4.1). Training and test errors $e\bar{r}r^{\text{tr}}$ and $e\bar{r}r^{\text{te}}$ by Eq. 5.10 are generated for the candidates and optimal descriptors of $\mathscr{S}_{\text{tr}}^{\text{CV}}$ and $\mathscr{S}_{\text{te}}^{\text{CV}}$ afterwards, by the mentioned sanity check (3).

6.3.6 Program Output

Extensive output is generated on the console, as image files and as additional text files. The first gives information on the progress in the program run and on the timings of its main tasks. The most important part is the information on the identified models $\{\mathscr{D}\}$ and their performance. This is printed in the following form

```
Ids in descriptor | MSE (a/r) | MAE (a/r) | maxAE (a/r) | RMSE (a/r)

1395       |  0.009 19.946 | 0.077 1.141 | 0.365  55.026 | 0.097 4.466
1395 9057  |  0.005 49.968 | 0.053 1.264 | 0.254 104.681 | 0.070 7.069
...
```

where the lines refer to descriptor dimensions $\Omega = 1, 2$ etc. The first numbers are the feature indices of the descriptor components, the following entries list the error metrics MSE, MAE, maxAE and RMSE (see Sec. 2.4.2). These are reported both in absolute and relative values, indicated by "a" and "r" in the table. This format is used for the results of all selection strategies. Its verbosity optionally may be reduced. For the ternaries **T**, additionally MAE_{arr} (see Sec. 5.5.1) is printed.
A dictionary like

```
Mini dictionary:

1395 (\langle E_{g,p} \rangle_{2}+\langle E_{g,XY} \rangle_{2})^3
881 (\langle a^{XY} \rangle_{2}+\langle a^{XY} \rangle_{3})^3
...
```

on the console informs on the symbolic expressions of the components in LaTeX style.
Information of the CV stability analysis as in the example of Tab. 5.3 is printed as

```
2D
Ids in descriptor        |        counts

 692 2193                              10
2193 2764                               9
2206 2257                               9
2194 2257                               8
...
```

on the console. It lists the component indices and the occurrence of, in this example, all 2D descriptor candidates in descending order of counts. Ideally, the assessed descriptor should be listed on top here, indicating the stability of its selection.

Optionally, further statistical information on the error distributions from the CV or the sanity check is displayed on the console as

```
RMSE
dim | min | 1% pctl | 25% pctl | med | 75% pctl | 99% pctl | max

1D:  0.038 | 0.038 | 0.040 | 0.040 | 0.041 | 0.042 | 0.042
2D:  0.026 | 0.027 | 0.028 | 0.029 | 0.029 | 0.030 | 0.030
...
```

Here, minimum, median and maximum values as well as the 1%, 25%, 75% and 99% percentile of the error distributions are printed.
Graphical output for the found models is generated in three forms:

- *correlation plots* between the descriptor prediction and the target (see Chap. 8 for many examples)
- *heat maps* of the pairwise absolute correlation $|\rho_{i,j}|$ for the descriptor components as well as for the features in the preselection I_{ℓ_0} (e.g. Fig. 8.4),
- *"data plots"* showing target and predicted values versus sample number (e.g. Fig. 8.16).

More details on the output including a list of all produced text files can be found in the code documentation.

6.3.7 Implementation of Complexity Measures

The implementation of the two complexity measures N_{f_k} and N_{op} defined in Sec. 5.5.2 relies on the list `features_dict` which contains LaTeX expressions of the total feature space. First, Listing 6.4 illustrates how informational complexity N_{f_k} is determined for an Ω-dimensional descriptor. It retrieves the symbolic expressions from `features_dict` for the component column indices i considered in the descriptor and concatenates them to the string `temp_str` (lines 1 - 3). By iterating over the list of all possible primary features `primary_features = ['r_s','r_p','r_{XX}',...]` the primary features included in this descriptor are detected. Whenever one of them is present once or several times, `N_fk` is raised by one (lines 5 - 7).

```
1 temp_str=''
2 for i in components:  # iterate over component indices
3   temp_str += features_dicts[i] # concatenate the strings
4
5 for fk in primary_features:  # filter out the primary features
6   if fk in temp_str:
7     N_fk += 1 # raise by one if feature is present
```

Listing 6.4: Python code to calculate informational complexity N_{f_k}.

The implementation of operational complexity N_{op}, as defined to work with the feature engineering scheme of Sec. 5.2, is a cumbersome programming task. We only illustrate it by Listing 6.5 that refers to the fifth component of the descriptor in Tab. 8.5 on the left, $\tilde{d}_{XY}^3/\sqrt[3]{\tilde{r}_s^2+\tilde{r}_s^3}$. Step by step, the different operations are detected in the string of its LaTeX expression. First, the higher moments are identified (lines 5-7), then the operation `'\sqrt'` (lines 13 - 15), next the `'+'` (lines 17 - 19) and finally `'\frac'` (lines 21 - 23). `N_op[4]` is raised accordingly each time (the index 4 marks the descriptor component). After a type of operations has been identified, the corresponding symbols are removed from the string of the LaTeX expression. Lines 9 - 11 show that also further string operations, such as removing the averaging symbols or other decorations, is necessary. A final summation over the contributions of the descriptor components yields the actual value of N_{op} as we defined it (not shown in the example).

```
 1  # initial string
 2  '\frac{\tilde{d}_{XY}^{3}}{\sqrt[3]{\tilde{r}_s^{2}+\tilde{r}_s^{3}}}'
 3  >> N_op[4] = 0
 4
 5  # identify higher moments
 6  '\frac{\tilde{d}_{XY} }{\sqrt[3]{\tilde{r}_s +\tilde{r}_s } }'
 7  >> N_op[4] = 3
 8
 9  # remove averaging symbols and other decorations
10  '\frac{ {d} }{\sqrt[3]{ {r}_s + {r}_s }}'
11  >> N_op[4] = 3
12
13  # identify '\sqrt'
14  '\frac{ {d} }{ {r}_s + {r}_s }'
15  >> N_op[4] = 4
16
17  # identify '+'
18  '\frac{ {d} }{ {r}_s  {r}_s}}'
19  >> N_op[4] = 5
20
21  # identify '\frac'
22  '{d} { {r}_s  {r}_s}'
23  >> N_op[4] = 6
```

Listing 6.5: Example for determining operational complexity `N_op[4]` for the fifth component of the descriptor in Tab. 8.5 on the left.

Chapter 7

Generating Ab-initio Data

Providing reliable input and output data of an understood level of accuracy is the prerequisite for building any machine learned model. This is the topic of this Chapter that focuses on the generation of the ab-initio *data of the group-IV ternary data set* **T***, the main test set of the developed ML methodology, and the required primary features. We first explain the computation, analysis, cleaning and organizing of the data set* **T***, and second continue with the pool of primary features for the feature engineering. Both of these parts start with a methodological background (Secs. 7.1 and 7.3), and then explain the actual data production and analysis (Secs. 7.2 and 7.4).*

7.1 Methods for Calculating Bulk Materials

Structure Relaxation Scheme: For every structure s in the group IV zincblende ternaries data set **T**, a *full* structure relaxation was performed. This yields the ground-state energy E_0 from which the two target values, the lattice constant a and the energy of mixing E_{mix} are derived. A full structure relaxation consists of both relaxing the crystal cell dimensions and the atomic positions. Fig. 7.1 shows the steps of this procedure.

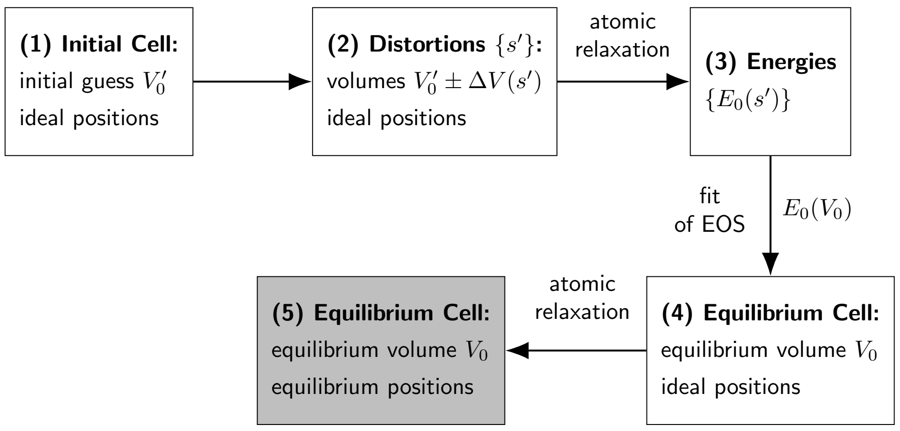


Figure 7.1: Full structure relaxation procedure to determine the equilibrium volume V_0.

First, a unit cell is set up with an initial crystal volume V_0' and ideal atomic positions. Several

]

distorted structures $\{s'\}$ are obtained by altering the volume V by $V_0' \pm \Delta V(s')$. For these, atomic positions are then relaxed from their ideal positions until a zero force configuration is reached. This produces a set of ground-state energies $\{E_0(s')\}$ to which an *equation of state* $E_0(V_0)$ (*EOS*) is fitted to determine the equilibrium volume V_0. Finally, again atomic positions are relaxed in this equilibrium cell. The accuracy of the thus obtained physical properties is influenced by the parametrization (i) of the ground-state calculation at fixed atomic positions and (ii) of the atomic relaxation module, (iii) as well as by the quality of the fit to the EOS.

Murnaghan Equation of State: For the group-IV ternaries, we use the *Murnaghan* equation of state [115]

$$P(V) = \frac{B_0}{B_0'}\left[\left(\frac{V_0}{V}\right)^{B_0'} - 1\right]. \tag{7.1}$$

It relates the pressure P and the unit cell volume V and derives from a linear expansion of the bulk modulus in P, $B = B_0 + B_0' \cdot P$. Using $P(V) = dE/dV$ and integrating over V leads to the expression

$$E_0(V) = E_0 + \frac{B_0 V_0}{B_0'}\left[\frac{1}{B_0'-1}\left(\frac{V_0}{V}\right)^{B_0'-1} + \frac{V}{V_0} - \frac{B_0'}{B_0'-1}\right] \tag{7.2}$$

for the total energy. It depends on four adjustable parameters, B_0, B_0', E_0 and V_0 that is to be determined. Hence, the fitting requires at least four supporting points $\{E_0(s')\}$.

Atomic Relaxation: For the atomic relaxation we use the optimization method *Broyden-Fletcher-Goldfarb-Shannon* (*BFGS*, [66, 116, 117]) in its limited memory and box variant as implemented in `exciting`. It determines the atomic configuration of zero-forces by restricting the three coordinates $r_k(i)$ of each atom i to a box by

$$r_k(i) - \tau_{\text{bfgs}} \leq r_k(i) \leq r_k(i) + \tau_{\text{bfgs}}. \tag{7.3}$$

The parameter τ_{bfgs} (in `exciting` `taubfgs`) determines the box width in which the coordinates are allowed to vary at an iteration step, typically chosen in the order of 1 Bohr. It relies on high accuracy in total energy in individual DFT steps, therefore in `exciting` it is recommended to use an energy convergence threshold `epsengy` two orders of magnitude smaller than the force convergence threshold `epsforces` [118].

7.2 The Group-IV Ternary Compounds Data Set T

The data-set **T** comprises materials composed out of three of the four elements carbon, silicon, germanium, and tin. In this work, we focus on the materials $C_xSi_yGe_{1-x-y}$, $C_xSi_ySn_{1-x-y}$, $C_xGe_ySn_{1-x-y}$ and $Si_xGe_ySn_{1-x-y}$ in the compositional range $0.0 < x, y \leq 0.5$ and $x + y < 1$ with a 16-atomic zincblende-like supercell.

From a technological point of view, group-IV ternary materials are of interest because their physical properties can be tailored by changing the relative concentrations. In particular the band gap and the lattice constant are decoupled [119], i.e. they can be tuned independently. For example, this allows to change the gap although the lattice constant is fixed to match an underlying buffer layer the ternary is grown on. Group-IV ternaries are hence promising candidates for electronic, photoelectronic and photovoltaic applications. Specifically, materials of the carbon-free type SiGeSn already are used for modulators and multi-quantum-well photodiodes [120, 121]. Experimentally, they are synthesized by chemical vapor deposition [122] or molecular beam epitaxy [123], typically as a random mixture of several compositions. The mean composition can be measured by Rutherford Backscattering Spectroscopy, and lattice constants are often measured by X-ray diffraction. For the carbon-free type SiGeSn, lattice constants are accessible in a range between 5.4 Å and 6.5 Å which however has not been covered experimentally in total yet [124]. As a specific example with a mean concentration similar to materials in **T**, for $Si_{41.9}Ge_{48.9}Sn_{9.2}$ a lattice constant of 5.631 Å is reported [80].[1] Computationally, *ab-initio* calculations of these materials are quite costly, requiring several DFT calculations per sample as explained in Sec. 7.1. It hence is of interest to find less demanding surrogate machine learning models.

Supercell and Atomic Occupation: The studied group-IV ternary compounds can be described by a 16-atom supercell which derives from the two-atomic zincblende parent lattice with the unit cell

$$\mathbf{u}_x = \frac{a}{2}(0,1,1),\ \mathbf{u}_y = \frac{a}{2}(1,0,1),\ \mathbf{u}_z = \frac{a}{2}(1,1,0) \tag{7.4}$$

with the atomic positions

$$\mathbf{x}_\mathrm{A} = (0,0,0),\ \mathbf{x}_\mathrm{B} = \frac{1}{4}(1,1,1) \tag{7.5}$$

for atoms of species A and B. The ternary materials are constructed by doubling this binary parent lattice in each direction and then exchanging up to four atoms by ones of type C. Due to this construction scheme, the numbers of atoms N_A, N_B, N_C of the types A, B and C, respectively, i.e. the composition, are restricted by

$$4 \le N_A \le 8,\ 4 \le N_B \le 8,\ 1 \le N_C \le 4, \tag{7.6}$$

or, equivalently, the concentrations $(x, y, z = 1 - x - y)$ by

$$\frac{1}{4} \le x \le \frac{1}{2},\ \frac{1}{4} \le y \le \frac{1}{2},\ \frac{1}{16} \le z \le \frac{1}{4}. \tag{7.7}$$

The atomic positions in the supercell are denoted in Tab. 7.1. Here, the two columns refer to sites occupied by A or B atoms in the original binary parent lattice. This parent structure is also shown in Fig. 7.2 on the left, the enumeration of sites in the supercell is sketched on the right. Based on this enumeration scheme, the individual derivative structures can be labeled by a 16-digit *occupation vector* σ [125, 126]. This is explained by Fig. 7.3 and plays an important role for the implementation of feature engineering (see Sec. 6.2.1). Here, atom types A, B and C are represented by the digits 0, 1 and 2 that refer to Ge, C and Si atoms in the shown example.

An essential characteristic of the ternaries is that a certain composition $K = (N_A, N_B, N_C)$ can be realized by *several symmetrically inequivalent* atomic arrangements σ, i.e. K does not define a structure s uniquely. This is illustrated by Fig. 7.4 showing the supercells of the four different arrangements of the same composition $(6,7,3)$. Apparently, not only the composition but also the arrangement influences the physical properties. Special attention has to be paid hence at the construction of the candidate features. For a given compound ABC, all different symmetrically

[1] These values need to be doubled to refer to a 16-atomic supercell.

invariant structures that satisfy Eq. 7.6 usually are found algorithmically. We used the tool *enumlib* [108, 127] to identify in total 52 possible structures of ABC, where some have equal K but differ in arrangement. All these structures are listed in Tables B.1, B.2 and B.3. Considering all combinations of the elements C, Ge, Si and Sn in **T**, the number of hypothetical structures hence is $12 \cdot 52 = 648$.

A positions					B positions			
u_1	u_2	u_3	#		u_1	u_2	u_3	#
0	0	0	1		1/8	1/8	1/8	9
0	0	1/2	2		1/8	1/8	5/8	10
0	1/2	0	3		1/8	5/8	1/8	11
0	1/2	1/2	4		1/8	5/8	5/8	12
1/2	0	0	5		5/8	1/8	1/8	13
1/2	0	1/2	6		5/8	1/8	5/8	14
1/2	1/2	0	7		5/8	5/8	1/8	15
1/2	1/2	1/2	8		5/8	5/8	5/8	16

Table 7.1: Atomic positions in the 16-atomic supercell spanned by the unit vectors of Eq. 7.4 (i.e. in lattice coordinates). Here, A and B positions refer to the parent binary structure with A and B atoms in alternating stacking. The fourth columns define the enumeration of the sites.

Interpolation and Vegard's law: Physical properties P of compound alloys are often linearly interpolated between the values $P_A, P_B, \ldots$ of the constituent pristine structures [128]. For ternary compounds with concentrations $(x, y, 1-x-y)$, this is expressed by

$$P_{\text{lin}}(x, y) = x \cdot P_A + y \cdot P_B + (1-x-y) \cdot P_C. \tag{7.8}$$

In the specific case of the lattice constant a this relation is known as *Vegard's law* [104, 129]. Similarily, the property P can be interpolated between the constituent binary components with properties P_{AB}, P_{AC} and P_{BC} [128]. Deviations from this very idealized linear scheme can be corrected by so-called *bowing terms* where the concentrations enter in quadratic or even higher order [128]. For instance, the lattice constant of the ternary alloy GeSiSn is approximated via [130]

$$a_{\text{GeSiSn}}(x, y) = a_{\text{Ge}} + (a_{\text{Si}} - a_{\text{Ge}})x + \Theta_{\text{SiGe}}x(1-x) + (a_{\text{Sn}} - a_{\text{Ge}})y + \Theta_{\text{SnGe}}y(1-y) \tag{7.9}$$

where Θ_{SiGe} and Θ_{GeSn} are empiric bowing coefficients. We adapt these linear and non-linear approximation schemes in the developed feature engineering method of this work (see Sec. 5.2.1). Note, that effects of the atomic arrangement, beyond the composition K, obviously are missed by these schemes.

Volume, Lattice Constant, Energy of Mixing E_{mix}: The unit cell volume V and the lattice constant a of these materials are connected by

$$a = \sqrt[3]{4 \cdot V}. \tag{7.10}$$

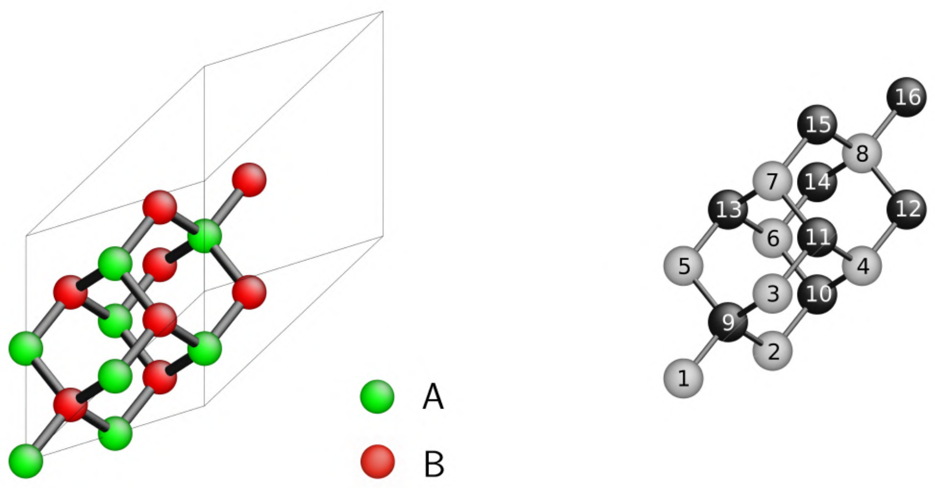


Figure 7.2: Left: Supercell of the underlying parent binary structure A_8B_8 with alternating packing of atoms A in red and B in green (showing the conventional cell). It is labeled by the occupation vector σ ='`0000000011111111`'. Right: enumeration of atomic sites in the ternary supercell defining the digits in the occupation vector σ.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Sites
`0`	`0`	`0`	`2`	`2`	`0`	`0`	`0`	`1`	`1`	`1`	`1`	`1`	`1`	`1`	`2`	Occup. vector
Ge	Ge	Ge	Si	Si	Ge	Ge	Ge	C	C	C	C	C	C	C	Si	Actual occup.

Figure 7.3: Labeling scheme of the atomic arrangements of the ternary materials **T**. The top line is the site enumeration (see Fig. 7.2 on the right) and the middle line the occupation vector σ where the digits `0`, `1` and `2` represent the general atom types A, B, C. The bottom line shows an example of the actual atomic occupation by the species for the structure $\mathrm{Ge_6C_7Si_3}$.

This is used to derive the values of a from the equilibrium volume V_0 obtained from fitting the Murnaghan equation of states (see Sec. 7.1). A final atomic relaxation of the equilibrium supercell (step (5) in Fig. 7.1) yields the ground-state energy $E_0(s)$ for a structure a. The energy of mixing per atom is then obtained by

$$E_{\text{mix}} = \frac{E_0(s) - E_{\text{lin}}(x, y)}{16} \tag{7.11}$$

where $E_{\text{lin}}(x, y) = x \cdot E_A + y \cdot E_B + (1 - x - y)E_C$ is the linearly interpolated ground-state energy according to Eq. 7.8.

Computational Details on the DFT Data: We started from an existing data set data set of the group IV zincblende ternaries that was created with `exciting` (see Sec. 3.2.1, applied release:

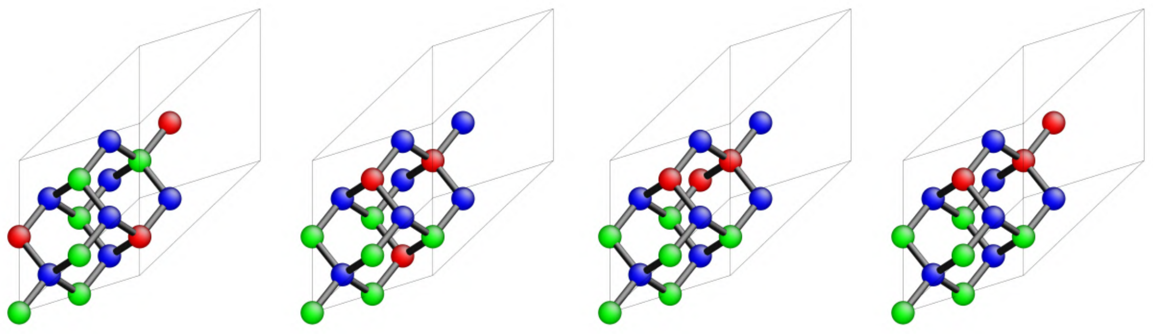


Figure 7.4: The four different atomic arrangements of the fixed composition $A_6B_7C_3$. Arrangements are labeled by the occupation vectors `0002200011111112`, `0000002212111111`, `0000002211111211`, and `0000002211111112` (from left to bottom), atoms A are plotted in green, B in blue and C in red.

beryllium [131]) and is online accessible at the NOMAD Repository [132]. Calculations were generated by the full structure relaxation scheme explained in Sec. 7.1. For each material, the lattice constant a was varied by ±2% and ±4% around an initial guess a_0 from Vegard's law (Eq. 7.8). Atomic relaxation was performed with the parameters `epsforce` = 10^{-4} Ha/Bohr and `taubfgs` = 0.3 with some exceptions addressed below. Fitting to the Murnaghan EOS was done by Powell's method [133] within **exciting**'s internal `OPTIMIZE` scripts, yielding E_0, V_0 and, by Eq. 7.10, a. For the equilibrium structures, the band structure was calculated along the path $L - \Gamma - X$ sampled by 200 steps. The parametrization of the ground-state module is presented in Tab. 7.2 in the top part, the applied muffin tin radii R_{MT} are in the bottom part. Atomic species were used from the `standard` set for the distortions and of the `smaller_core_set` for the final runs[2]. Since the two paths $L - \Gamma$ and $\Gamma - X$ are calculated separately properties of the relaxed structures are present twice in the data set. We extracted all target properties from the file `INFO.OUT` referring to $L - \Gamma$.

ngridk	rgkmax	xctype	nempty	epsengy
$4 \times 4 \times 4$	7	LDA_PW	9	10^{-6} Ha

species	C	Si	Ge	Sn
R_{MT} [Bohr]	1.45	1.65	1.8	1.9

Table 7.2: Parametrization (top) and muffin tin radii (bottom) used in the `groundstate` module to generate the dataset **T**.

Analysis of the Initial Data Set: Before the actual ML tasks were tackled, an in-depth analysis of the existing data set was performed to check for its accuracy and consistency. This initial data set comprises 448 of the possible 648 structures that split into the 12 different compounds as listed in Tab. B.4. For each of these, the 42 arrangements of Tables B.1 and B.2 are considered where for some of them are no data was available. Statistically, the under-representation of materials of the

[2]The latter are available at [109].

type GeSnC might introduce biases at the machine learning.

Next, we screened the data for consistency and correctness in the DFT calculations, of both the distorted and the equilibrium structures (see Fig. 7.1). The data presented some variability with regard to several parameters which mostly affect the target properties only in a minor way. The variability in the energy convergence threshold `epsengy` (see Tab. B.5) affects the Murnaghan fits which was further investigated (see below).

As another part of the analysis we graphically examined the convergence in atomic forces both of the distorted and the equilibrium structures (see Fig. B.2 in the Appendix). For some of the distorted structures, the convergence threshold `epsforce` of 10^{-4} Ha/Bohr is not met, and for other structures the convergence could not be checked due to missing data on the forces.

Consequently, we also analyzed the fits of the Murnaghan EOS explicitly. For some materials these were found to be very inconsistent. An example for this is shown in the left panel of Fig. 7.5 that is problematic from two reasons. First, energies of the distortions exhibit a "jump" of $\Delta E_0 \sim 100$ mHa between the second and the third value. Second, the value of a used in the final structure (red dot) deviates by ~ 0.7Å from the actual fitted minimum (green).[3]

As a last step, the atomic positions of all materials were examined visually. For several structures, we found the displacement of the atoms from their ideal positions unreasonably large. The consistency of the relaxation was checked for that cases. Generally, we observed the high diversity in bonding motifs in the ternary materials in that step: a fixed type of bond (e.g. Si-C) varies between materials, center atoms and neighboring atoms (see the two example structures in Fig. 7.6). This provided important insights on which characteristics the material representation for the machine learning needs to consider, or, vice versa, might miss.

Curing of the Data: Based on the data analysis, it was decided to cure the data by performing new *ab-initio* relaxations for the missing and erroneous material samples.[4] High care was taken to resemble the original parametrization as close as possible. Unfortunately, the missing arrangements similarly could not be relaxed fully. It can be concluded that these are hypothetical, energetically very unfavorable arrangements. Most of the structures with faulty relaxations could be cured like in the example of Fig. 7.5 on the right. Only for the three structures $C_6Si_6Sn_4$ (`0002200011111122`), $C_5Ge_7Sn_4$ (`0000022211111121`) and $C_5Si_7Sn_4$ (`0000022211111121`), it was not possible to obtain reliable values of a and E_0. These were excluded from **T** hence such that the final data set has the size N =445. Lastly, the actual used lattice constants were then determined for all samples by new fits to the Murnaghan EOS with an external optimizer (`scipy.optimize.curve_fit`, the non-linear least squares optimizer from the SciPy library [134]).

Organizing the Final Data Set: All materials samples were organized by unique material labels defined by

$$\texttt{lbl} = (A, B, C, N_A, N_B, N_C, \sigma). \tag{7.12}$$

[3]Besides that, also smaller discontinuities in the $E_0(V)$ curves were identified. We found that these result from a changing total number of **G**-vectors at the different $V(s')$ and hence a changing size of the Hamiltonian matrix. This could be "healed" by rescaling the parameter $\texttt{rgkmax} = R_{\mathrm{MT}}^{\min} \cdot |\mathbf{G}+\mathbf{k}|_{\max}$ via $\texttt{rgkmax} \mapsto \texttt{rgkmax} \cdot [a_0/a_0(s')]$ which will adapt the number of **G**-vectors accordingly to keep the size of the Hamiltonian. Since this affects a by less than 1 mÅ and E_0 by less than 1 meV, significantly smaller than the reachable accuracy of the ML approach, this step is not necessary.

[4]Still, only the 42 structures per compound from Tab. B.4 were considered.

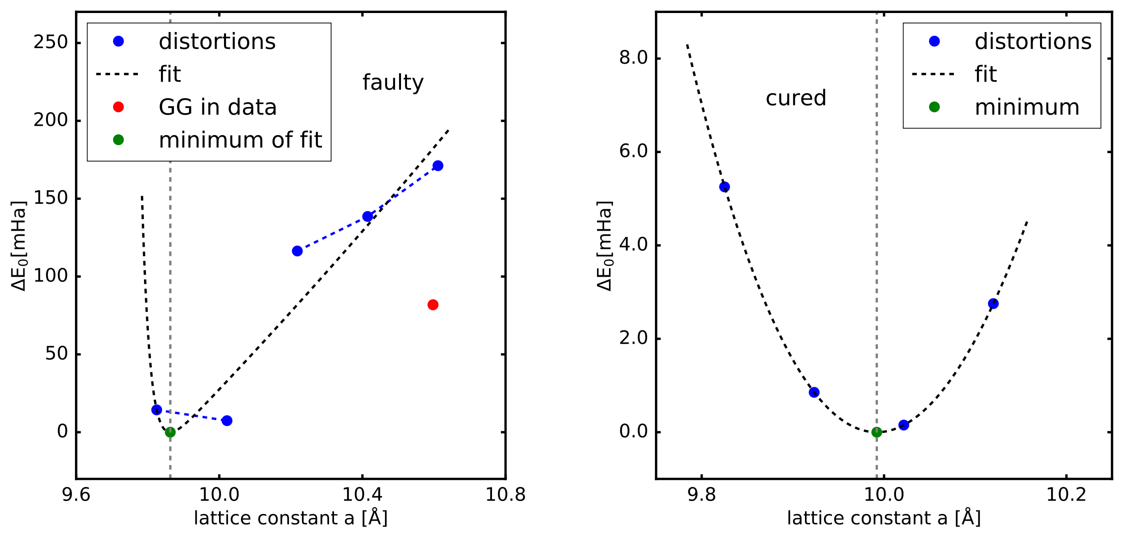


Figure 7.5: Total energy versus lattice constant a for the structure $C_4Si_6Ge_6$ in the arrangement $\sigma = 1111110022202022$. Blue dots indicate distorted structures s', curves show the Murnaghan fit, green dots mark the minimum of the fit. Energies are plotted as differences with respect to the energy of the minimum of the fit. Left: the initial calculations with a jump in energy of $\Delta E_0 \sim 100$ mHa between the second and the third data point. The red dot marks the lattice constant present in the equilibrium calculation in the data that markedly deviates from the fitted minimum. Right: updated calculations, showing that a was overestimated by ~ 0.6Å initially. Note the narrower scales for the energy and the lattice constant a for the cured relaxation.

This list contains the atomic species A, B, C, their composition (N_A, N_B, N_C) and their arrangement by the occupation vector σ and is required for the feature engineering.[5] The order of the atom species in `lbl` follows the convention to put them in ascending order of element number which differs from the convention in the initial data-set. Accordingly, the restrictions on compositional space and concentrations of Eqs. 7.6 and 7.7 change to $0 < N_A, N_B, N_C \leq 8$ and $0 < x, y, z \leq \frac{1}{2}$, respectively. An example for a label `lbl` and its transformation is given in Fig. 7.7.

We further organized the total data set by sorting all materials in ascending order of their labels `lbl`, defining a systematic indexing scheme. Here, each entry in the list `lbl` is compared after the other where the species are compared based on element number and the occupations σ are sorted in alphabetic order (without a physical meaning). This steps brings the 12 different combinations of four atoms to ternaries to the four branches $C_xSi_yGe_{1-x-y}$, $C_xSi_ySn_{1-x-y}$, $C_xGe_ySn_{1-x-y}$ and $Si_xGe_ySn_{1-x-y}$ and groups all arrangements of equal composition K together. The indexing scheme is used in Fig. 7.8 that shows the two target properties a and E_{mix} of the total data set. In the inset figures arrangements of equal composition are connected by lines, illustrating the property variability with respect to the arrangement. Additionally, the hold-out sets $\mathbf{S}_{\text{ho}}$ used in the ML tasks are indicated. They comprise each ~10% of the total data and were chosen manually to represent the two property spaces adequately. A summary of the main characteristics of the data-set $\mathbf{T}$ is given in Tab. 7.3.

[5]Note, that N_A, N_B, N_C are denoted in `lbl` for convenience although they could be derived from σ.

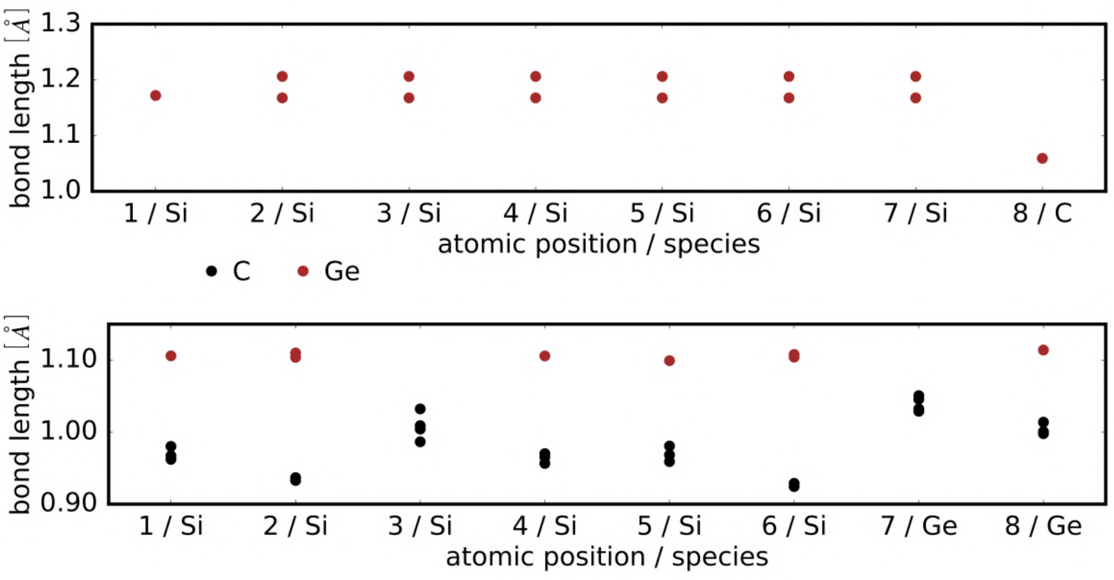


Figure 7.6: Bonding motifs for the samples $C_1Si_7Ge_8$, σ = 1111111022222222 (top), and $C_6Si_6Ge_4$, σ = 1111112202000200 (bottom), of the ternary data set **T**. Shown are the each four bond lengths at the eight atoms at the A positions of the parent lattice (see Fig. 7.2 and Tab 7.1). The species of the center atoms are indicated on the axes, the species of the outer atoms by colors.

```
('Ge', 'C', 'Si', '6', '7', '3', '0000002212111111')
                          ↓
('C', 'Si', 'Ge', '7', '3', '6', '2222221101000000')
```

Figure 7.7: Example for a material label `lbl` of a ternary sample and its transformation from the original convention (top line) to the convention used in this work (bottom line). The species are brought in the order of their element number which switches N_A, N_B, N_C and modifies σ.

Estimating the Data Accuracy: In order to determine the accuracy of the target properties a and E_{mix}, we consider the magnitude of the three contributions (i), (ii) and (iii) described in Sec. 7.1. The accuracy in the groundstate energy E_0 at fixed atomic positions (i) was estimated to be ~ 20 mHa.[6] The contribution of the atomic relaxation (ii) is ~ 0.2 mHa and hence negligible.[7] Also the contribution of the fitting (iii) since the optimizer was run in a tight parametrization. An analysis on the EOS curves[8] yielded that this leads to an uncertainty Δa of a few ~ 0.01 Å in the equilibrium lattice constant. This uncertainty again affects the final value of the total energy E_0. From the fits to the Murnaghan EOS, we estimate that a variation of a by $\pm$0.01 Å on average influences E_0 by ~ 1 mHa. This is negligible compared to the mentioned uncertainty due to the DFT calculation

[6]From the relevant parameters `rgkmax`, `ngridk` and `epsengy`, the contribution of `rgkmax` here is the largest. The number was obtained by varying this parameter between 7.0, 7.5, 8.0 and 8.5 for a representative test material.

[7]It was checked that the additional convergence threshold `epsforce` does not reduce the accuracy of E_0. For this, we determined that E_0 does not change by more than ~ 0.2 mHa right before convergence of forces at the relaxation.

[8]On several representative materials the standard deviance in a was determined by each 100 fits to the EOS with noise uniformly distributed in the interval $[-10\,\text{Ha}, 10\,\text{Ha}]$ added to the values $E_0(s')$.

itself. We assume that the uncertainty in the ground state energies of the pristine materials $E_{0,XX}$ equally is ~ 20 mHa (see the paragraph right below) and derive by basic rules of error analysis and Eq. 7.11 that the uncertainty in the energy of mixing per atom is $\Delta E_{\text{mix}} \approx 1/8 \cdot 20\,\text{mHa} \approx 68\,\text{meV}$ or less. For the machine learning, these values for Δa and ΔE_{mix} serve as approximates for the irreducible error in the target property (see Eq. 2.28).

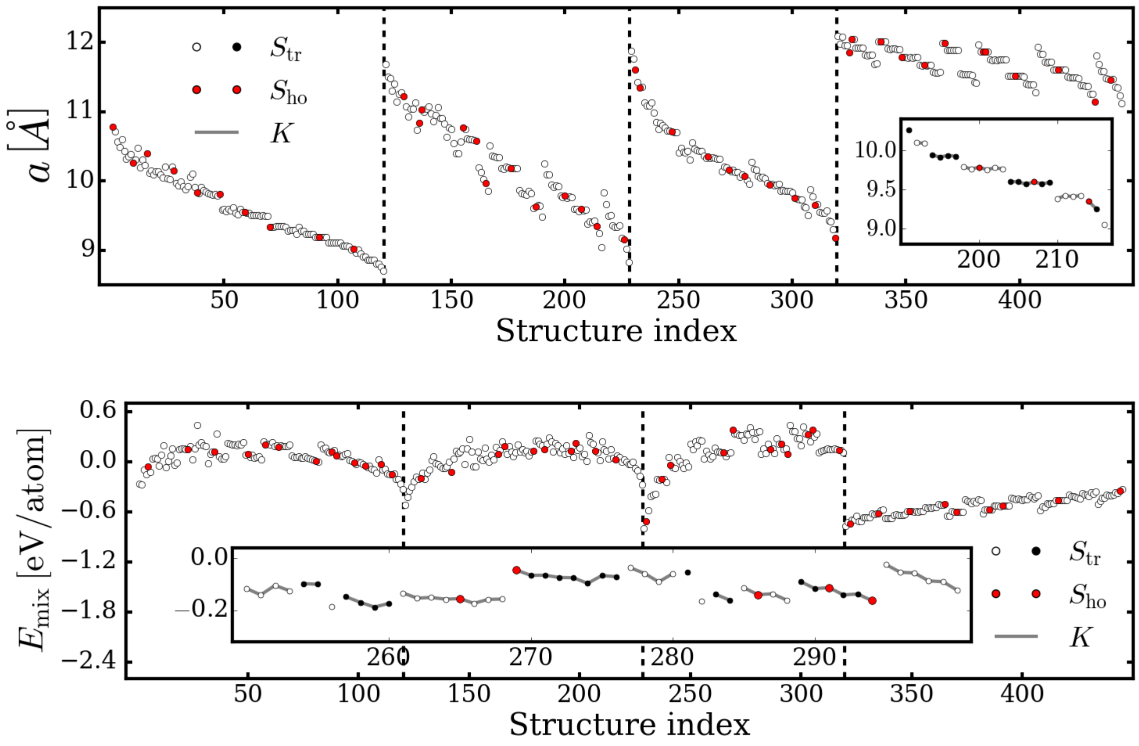


Figure 7.8: Target properties a (top panel) and E_{mix} (bottom panel) versus structure index of the data-set of zincblende group-IV ternaries **T** (the sorting scheme is explained in the text). Data points used for model training (and in CV) are plotted in white or black, those in the hold-out set $\mathbf{S}_{\text{ho}}$ in red. Compositions K are linked by grey lines. Vertical dotted lines separate between the compounds $C_xSi_yGe_{1-x-y}$, $C_xSi_ySn_{1-x-y}$, $C_xGe_ySn_{1-x-y}$ and $Si_xGe_ySn_{1-x-y}$. Insets show some parts of the data to illustrate influences of the arrangement σ within one K.

Bulk Properties: The feature spaces for the ML on the data set **T** include also properties of pristine and binary bulk properties. To be consistent, we newly generated the former by the same steps as for the ternary materials, also with a 16-atomic supercell. The *ab-initio* calculations differed from the parametrization of Tab. 7.2 by using $R_{MT} = 1.35$ Bohr for carbon, applying the `standard` species preferences and setting `nempty` to 5 instead.

All binary materials AB have the supercell of Fig. 7.2 on the right and were already included in the initial ternary dataset. Hence, the data relies on exactly the same preferences of `exciting` as used for **T**. Lattice constants a_{XY} stem from new fits to the EOS performed by us, ground-state energies $E_{0,XY}$ and the gap at Γ $E_{g,XY}$ were taken from the relaxed initial output.

P	level	N	N_{ho}	min	max	sdv	Δ
a	3	445	44	8.708 Å	12.093 Å	0.949 Å	~ 0.03 Å
E_{mix}	3	445	43	-0.799 eV	0.440 eV	0.314 eV	~ 0.068 eV

Table 7.3: Summary of the characteristics of the data-set of zincblende group-IV ternaries **T**. Listed are the levels in the hierarchy of *ab-initio* data (see Sec. 4.2), the sizes N of the total data-set and the hold-out sets, the minimum and maximum value, the standard deviation and the estimated uncertainty Δ for the two target properties a and E_{mix}.

7.3 Methods for Calculating Non-Periodic Structures

7.3.1 Using `exciting` for Isolated Molecules

The LAPW code **`exciting`** (cf. Sec. 3.2.1) is designed for periodic solid structures, i.e. it always assumes a three-dimensional periodicity of a unit cell. Isolated molecule-like clusters like the single atoms, pairs and tetrahedrons used in this work, are embedded in a sufficiently large box of vacuum to avoid the interaction with the replica in the periodic cells. A proper choice of the box dimensions $x_{\text{vac}}, y_{\text{vac}}, z_{\text{vac}}$ is determined by numerical tests in preparatory. The groundstate calculations have to be run with a **k**-grid of $1 \times 1 \times 1$ in that case which does not allow for a parallel execution. Instead, the computational effort that quickly grows with the box size is reduced by changing from the standard Hamiltonian solver to the *Davidson algorithm* ([135], see the run-time test in Appendix B.2).

7.3.2 Optimization Scheme

At the feature engineering for the data set **T**, we considered isolated dimers and regular tetrahedron clusters at their energetic equilibrium. These structures depend on the single degrees of freedom $x = d_{XY}$ or d_t, respectively (see Fig. 7.10), for which the total energy E_0 of the groundstate has to be minimized. This was done by combining DFT calculations at fixed geometry with an external iterative optimizer, as illustrated by Fig. 7.9.

At each iteration i of the procedure, **`exciting`** yields the total energy $E_0(i)$ which is read in by a Python script. The structural update then is performed by the *Nelder-Mead algorithm* [136], using the implementation of the SciPy optimization sublibrary [134]. Updated values of d_{XY} or d_t are written to a new `input.xml` file that **`exciting`** takes to calculate the following $E_0(i+1)$. The procedure is stopped if the convergence criteria of the Nelder-Mead algorithm are satisfied. A final *ab-initio* run yields the equilibrium physical properties for the feature engineering which we parse from the file `INFO.OUT`.

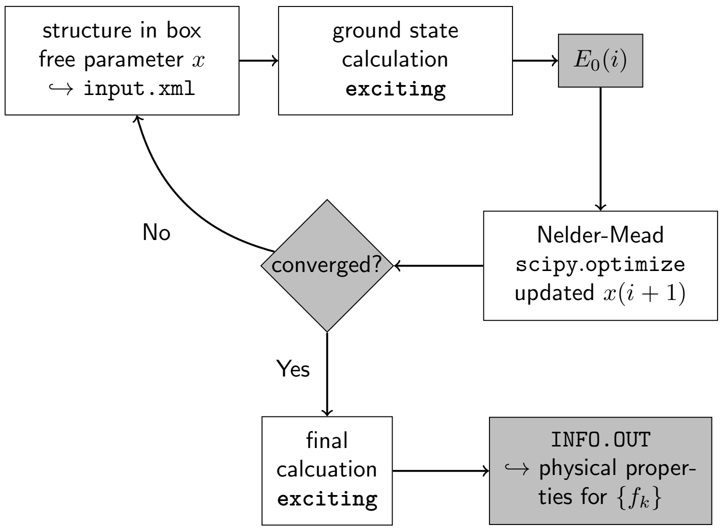


Figure 7.9: Flow diagram of the structural optimization of dimer and tetrahedron clusters. Single runs of `exciting` are combined with the external Nelder-Mead algorithm.

7.4 Generation of Local Primary Features

7.4.1 Atomic Properties

For the most basic building block - a single atom - DFT basic data was generated by `exciting` calculations of the isolated atoms in a cubic box of vacuum (cf. Sec. 7.3.1). From these, the local primary features $E_{0,X}$, i.e. the ground state energy, and $E_{g,X}$, i. e. the HOMO-LUMO gap as defined in Sec. 3.3, were extracted. Atomic species settings from the `smallercore` set were applied (see [109]) and muffin tin radii were equal to the values used to calculate the ternary data set **T** (see Tab. 7.2). `rgkmax` was increased to 10.0 which ensures an accuracy level at least equivalent to **T** where the accuracy is determined by the smallest atom C with `rgkmax` = 7 and $R_{\mathrm{MT}} = 1.45$ (cf. Sec. 3.2.1). A cubic box of edge length $s = 20.$ Bohr = 10.58 Å was used. Convergence of the properties $E_{0,X}$ and $E_{g,X}$ with respect to s was tested for a Sn atom as this is the largest and hence most demanding element required for this thesis (see Sec. B.3 in the Appendix). In that case E_0 is converged by ~ 5 meV and $E_{g,X}$ by ~ 0.1 eV. This rather low accuracy has to be considered for the predictive performance of ML models based on these features.

Table B.6 in the Appendix lists $E_{0,X}$ and $E_{g,X}$ for the elements C, Si, Ge and Si contained in **T**. For these atoms, also ionization potential IP and electron affinity EA as well as radii of the s-, p- and d-orbitals were included to the local primary features. These radii are defined as maximum of the valence radial probability density of s-, p- and d-wavefunctions and were calculated with the code FHIaims. The values were taken from [1] and were performed with the LDA-PW exchange-correlation functional [62], `tight` grid settings and atomic basis level `tier3`.

7.4.2 Dimer Properties

For pair building blocks, a pool of dimer features was calculated with `exciting` for all required combinations of two atoms X, Y (comprising both homo-atomic combinations, XX, and heteroatomic ones, XY). Following the scheme described above, the isolated dimers were embedded in a box of vacuum and structurally relaxed. That way, the equilibrium dimer distance d_{XY}, the ground-state energy $E_{0,XY}$ and the HOMO-LUMO gap $E_{g,XY}$ (cf. Sec. 3.3) were determined. d_{XY} was taken from the relaxation algorithm, $E_{0,XY}$ from `INFO.OUT` and $E_{g,XY}$ by calculating the difference between the last at least partially occupied eigenvalue and the first fully unoccupied one from `EIGVAL.OUT`. The DFT parametrization was chosen as a compromise between accuracy and efficiency as the relaxation may involve many single runs. Atomic settings from the `standard` set were applied and `rgkmax` was set to 7. Generally, muffin tin radii of the ternary data set **T**, were used (see Tab. 7.2). For some combinations XY however, smaller values of R_{MT} were applied to avoid an overlap of muffin tin regions. The Nelder-Mead algorithm was run with the parameters `xtol`$= 10^{-5}$ and `maxiter=100`. The size of the cubic box of vacuum was $s = 15\,\mathrm{Bohr} = 7.94\,\mathrm{Å}$ for all dimers. Convergence of the determined primary features was investigated on the computationally most demanding dimer Sn-Sn (see Sec. B.3 in the Appendix).

d_{XY}
$d_t/\sqrt{3}$

Figure 7.10: Examples of a dimer XY with dimer distance d_{XY} (left) and a tetrahedron cluster with bond length $d_t/\sqrt{3}$ which is equal for all bonds (right).

7.4.3 Tetrahedron Properties

Additionally, a pool of 120 tetrahedrons was generated that considers all combinations of one center atom and four surrounding ones, required for the data-set **T**. The geometry was assumed to be symmetric, i.e. atomic positions were $(0,0,0)$, $(1,1,-1)$, $(-1,-1,-1)$, $(-1,1,1)$ and $(1,-1,1)$, multiplied by the scaling factor d_t (see Fig. 7.10). The same set-up of **`exciting`** as for the dimers was applied, also with eventually adapted muffin tin radii R_{MT}. An `input.xml` file for one of the tetrahedrons is listed in the Appendix in Listing B.1. From the relaxed clusters, we took the properties d_t from the optimization as well as the ground-state energy $E_{0,t}$ and the HOMO-LUMO gap $E_{g,t}$ from `INFO.OUT`. These values are reported in Tabs. B.8 - B.13 in the Appendix. The external optimization was run with `xtol`$= 10^{-5}$ and `maxiter`$= 50$. Convergence tests were performed on the heaviest tetrahedron Sn-Sn-Sn-Sn-Sn as benchmarking case which is reported in Sec. B.3 in the Appendix.

Note that the isolated five-atomic tetrahedron clusters are highly artificial objects from a chemist's perspective because of their unfavourable electronic configuration. The obtained properties hence were assumed to capture the local environment in the ternary compounds too roughly. As an improvement, we tried to calculate the clusters with the outer atoms saturated by three hydrogen atoms each [137], leading to stable molecules of the type of tetramethylsilane $Si(CH_3)_4$. The increase in number of atoms and size – and hence the required box size – however resulted in an unreasonable computational effort.

Analyses and Results Part III

Chapter 8

Applications: Analysis and Results

In this chapter, we apply the developed machine learning methodology to several property prediction problems. The main analyses are done on the group-IV ternary data-set $\mathbf{T}$. *We first investigate the construction of the feature space, in regard of the different types of local primary features (Sec. 8.1.1), and the feature engineering steps (Sec. 8.1.2). In Sec. 8.2 the different dimensionality reduction methods* $\mathscr{M}$ *are analyzed where in particular LASSO and SISSO are compared. The developed model selection strategies* $\mathscr{S}^{\mathrm{all}}$, $\mathscr{S}_{\mathrm{tr}}^{\mathrm{CV}}$ *and* $\mathscr{S}_{\mathrm{te}}^{\mathrm{CV}}$ *are tested and evaluated in Sec. 8.3. In Sec. 8.4 we present a set of optimal descriptors for the lattice constant and energy of mixing of the ternaries* $\mathbf{T}$ *that was identified with the confined methodology. As an additional application, in Sec. 8.5 we study several learning tasks for the lattice constant, the TB and the LDA band gap on the data set of the octet binary semiconductors* $\mathbf{O}$.

8.1 Analysis of the Feature Space Construction

8.1.1 Analysis of the Selection of Primary Features

In this work, we use four different types of physical quantities to generate the candidate features for the ternary dataset $\mathbf{T}$. The corresponding lists of these local primary features are given in Tables 8.1 and 8.2 for the two different target properties, the equilibrium lattice constant a and the energy of mixing E_{mix}. They contain physical properties of single atoms, molecular dimers, tetrahedral clusters, and pristine and binary bulk solids for all chemical species or combinations of chemical species present in the dataset. We treat the bulk properties in the same way as single-atom and pair properties, because they only depend on a single atom type or a pair of atoms, respectively. Global primary features are then calculated by Eqs. 5.1, 5.2, 5.3, and 5.4, averaging over all atoms, pairs or tetrahedrons in the supercell of a structure s. The normalization factor m is equal to 16 for atomic and atom-centered tetrahedral building blocks for these 16-atom supercells. For pair properties, m is equal to 32, the total number of pairs in these structures.

Dimer bond lengths $d_{XX/XY}$, tetrahedral scaling factors d_t and lattice constants $a_{XX/XY}$ were scaled to radius equivalents. This was done to allow for a better interpretation of features that algebraically combine these properties. Specifically, the dimer lengths were divided by 2, assuming touching atomic spheres, and tetrahedron scaling factors d_t by $2 \cdot \sqrt{3}$ which derives from the tetrahedron shape (see Fig. 7.10).[1] Bulk lattice constants accordingly were scaled by $\sqrt{3}/16$ considering also touching spheres in the 16-atomic crystal structure.

The included energies are of two types: energies of formation and band-gap energies. Energies of formation are defined as difference in ground state energy of the respective building blocks γ and all constituent, isolated atoms, $E_{f,\gamma} = E_{0,\gamma} - \sum_{i\in\gamma} E_{0,\mathrm{X}}(i)$.[2] We use per-atom values, to improve the interpretability of algebraic combinations. For isolated atoms, dimers and tetrahedrons, we use the HOMO-LUMO or KS band gap by Eq. 3.23 while for pristine and binary materials we use the band gap at Γ by Eq. 3.24.

Feature (f_k)	Description	$X_{a,1}$	$X_{a,2}$	$X_{a,3}$	$X_{a,4}$
Single-atom data					
Z	Atomic number	×	×		×
P	Period (in periodic table)	×	×		×
r_s	s-orbital radius	×	×		×
r_p	p-orbital radius	×	×		×
r_d	d-orbital radius	×	×		×
a_{XX}	Lattice constant of pristine material		×		×
d_{XX}	Bond length of homoatomic dimer		×		×
Pair data					
a_{XY}	Lattice constant of binary material		×		×
d_{XY}	Bond length of heteroatomic dimer		×		×
Tetrahedral cluster data					
d_t	Bond length in tetrahedron			×	×

Table 8.1: The local primary features f_k and feature spaces $\mathbf{X}_{a,i}$, $i = 1-4$ used for learning the lattice constant a of the ternaries $\mathbf{T}$. The crosses indicate that a quantity is included in $\mathbf{X}_{a,i}$. Note, that $\mathbf{X}_{a,1}$ contains only pure atomic properties and only the computationally cheapest of them. The more expensive properties a_{XX} and d_{XX} are excluded although they are formally treated as atomic features in Eq. 5.1.

From the local primary features listed in Tables 8.1 and 8.2, several feature spaces $\mathbf{X}_{a,i}$ and $\mathbf{X}_{E,i}$ ($i = 1,\dots,4$) of different complexity are constructed for the two learning problems in an analogous manner: $\mathbf{X}_{a/E,1}$ contains the single-atom features only, i.e. any average in Eq. 5.1 will be over atomic building blocks only ($\gamma = i$). We do not consider bulk or dimer features here in order to include only the computationally cheapest features. $\mathbf{X}_{a/E,2}$ additionally contains pair features which are averaged over all pair building blocks $\gamma = (\alpha,\beta)$. $\mathbf{X}_{a/E,3}$ solely consists of tetrahedron features, with the aim to study the predictive power of such building blocks. $\mathbf{X}_{a/E,4}$ finally combines single-atom, pair and tetrahedron data.

[1] We will refer to d_t also as tetrahedral bond length which apparently transforms to the same radius equivalent.

[2] As, obviously, isolated atoms do not have an energy of formation, we consider only the band gap for these building blocks.

Feature	Description	$X_{E,1}$	$X_{E,2}$	$X_{E,3}$	$X_{E,4}$
Single-atom data					
Z	Atomic number	×	×		×
P	Period	×	×		×
r_s	s-orbital radius	×	×		×
$E_{g,X}$	HOMO-LUMO gap of isolated atoms	×	×		×
$E_{f,p}$	Energy of formation of pristine material		×		×
$E_{g,p}$	Gap of pristine material at the Γ-point		×		×
$E_{f,XX}$	Energy of formation of homoatomic dimer		×		×
$E_{g,XX}$	HOMO-LUMO gap of homoatomic dimer		×		×
Pair data					
$E_{f,b}$	Energy of formation of binary material		×		×
$E_{g,b}$	Gap of binary material at the Γ-point		×		×
d_{XY}	Bond length of heteroatomic dimer		×		×
$E_{f,XY}$	Energy of formation of heteroatomic dimer		×		×
$E_{g,XY}$	HOMO-LUMO gap of heteroatomic dimer		×		×
Tetrahedral cluster data					
d_t	Bond length in tetrahedra			×	×
$E_{f,t}$	Energy of formation of tetrahedra			×	×
$E_{g,t}$	HOMO-LUMO gap of tetrahedra			×	×

Table 8.2: Local primary features f_k and feature spaces $\mathbf{X}_{E,i}$, $i = 1-4$ used for learning the energy of mixing, E_{mix}, analogous to Tab. 8.1. The quantities Z, P, r_s, d_{XY} and d_t are the same as for learning a. In addition, band-gap energies and energies of formation are considered here. Similar to $\mathbf{X}_{a,1}$, $\mathbf{X}_{E,1}$ contains only the computationally cheapest pure atomic features.

For predicting the lattice constants, most of the considered local primary features are distances, except from the atomic number Z and the period P. Since d_t is the only tetrahedron feature included here $\mathbf{X}_{a,3}$ is based on a single local primary feature. For E_{mix}, various energies are dominating the feature space. For the following discussion, we augment the features up to complexity tier $\mathscr{T}_1$, without the higher moments of Eqs. 5.2 and 5.3. Descriptors are identified by strategy $\mathscr{S}^{\text{all}}$ up to dimension $\Omega = 5$, and the data matrices are pre-processed as described in Sec. 6.3.1.

Analysis at learning the lattice constant: The left panel of Fig. 8.1 shows training errors (RMSE) and MAE_{arr} (see Eq. 5.13) versus Ω for predicting lattice constants using the four feature spaces $\mathbf{X}_{a,1-4}$. Additionally, the interpolations between pristine materials a_{XX}, i.e. Vegard's law (Eq. 7.8), and binary materials a_{XY} (by Eq. 5.1, considering pairs), as well as the average tetrahedral bond length $\bar{d}_t$ are indicated as reference lines. In the lower left figure of MAE_{arr}, a_{XX} indicates

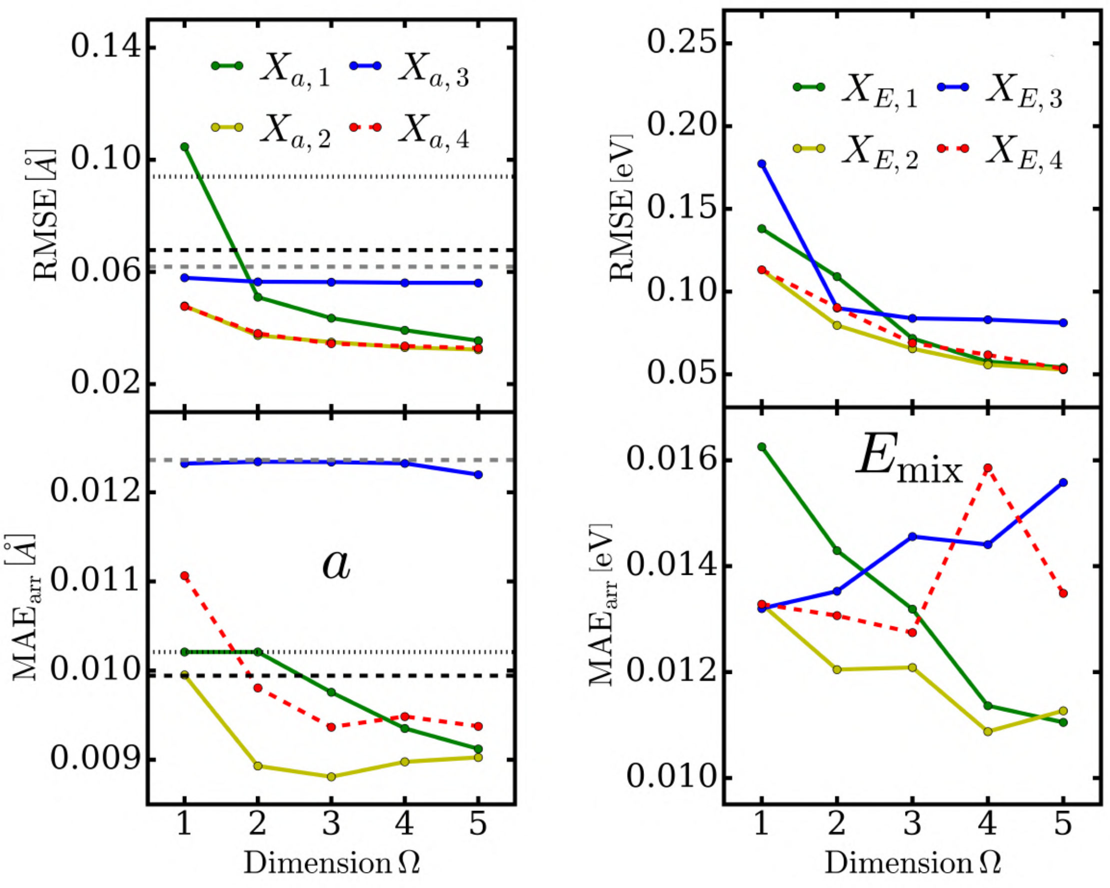


Figure 8.1: Performance of the feature spaces $\mathbf{X}_{a/E,1-4}$ with different primary features for learning a (on the left) and E_{mix} (on the right) of $\mathbf{T}$. Top plots: training error (RMSE from $\mathscr{S}^{\mathrm{all}}$) versus descriptor dimension Ω. Bottom plots: MAE$_{\mathrm{arr}}$ (cf. Eq. 5.13) vs. Ω. In the left plot, black dotted lines refer to the interpolation between pristine materials a_{XX} (Vegard's law, Eq. 7.8) and black dashed lines to the interpolation between binary materials a_{XY}. Gray dashed lines mark a fit of averaged tetrahedron bond length, i.e. the one-dimensional descriptor $\mathscr{D}_{\mathrm{1D}} = d_t$.

the baseline of a blindness to effects of the atomic arrangement, since it depends only on the composition. a_{XY} captures them to some degree, i.e. it lies below, since different structures of the same composition often differ in some bond types (this was discussed in Sec. 5.2.1 on p. 40 and following).

In the upper panel, for all four feature spaces an increase in dimension Ω naturally lowers the training error, which is most distinctive for $\mathbf{X}_{a,1}$ and least for $\mathbf{X}_{a,3}$. Except for the 1D descriptor of $\mathbf{X}_{a,1}$ ($\mathscr{D}_{\mathrm{1D}} = 1/\sqrt[3]{r_p}$), all models outperform Vegard's law a_{XX} as well as the binaries' average a_{XY}. Including pair quantities in addition to pure atomic features, i.e. going from $\mathbf{X}_{a,1}$ to $\mathbf{X}_{a,2}$, reduces the RMSE drastically. Adding also the tetrahedron features, by contrast, has a negligible effect on the training error. Yet, the descriptors emerging from $\mathbf{X}_{a,4}$ are not identical to those from $\mathbf{X}_{a,2}$ but contain also the tetrahedron feature d_t. Looking at the 1D descriptors, the pure tetrahedron feature space $\mathbf{X}_{a,3}$ has a much lower training error than $\mathbf{X}_{a,1}$, however still larger than that of $\mathbf{X}_{a,2/4}$. At higher dimension, the RMSE improves only little such that the descriptors from $\mathbf{X}_{a,3}$ cannot compete with the others. The fitting error also remains close to the benchmarking case $\mathscr{D}_{1D} = d_t$. As d_t is the only local primary feature in this case, the higher mathematical complexity of these descriptors barely improves the descriptor performance.

MAE$_{\mathrm{arr}}$ (lower panel of Fig. 8.1) also decreases with higher Ω, except for $\mathbf{X}_{a,2}$ and $\mathbf{X}_{a,4}$ where it has a shallow minimum at $\Omega = 3$. Interestingly, already descriptors from the pure atomic feature space $\mathbf{X}_{a,1}$ are able to go beyond the arrangement-agnostic a_{XX}. This is the case for $\Omega \geq 3$ where the descriptors include nearest-neighbor differences by Eq. 5.4. If this type of features is not included, like in the 1D and 2D descriptors, the descriptors predict a constant value for all arrangements of fixed composition which implies that they lie on the dotted line of Vegard's law. Minimal MAE$_{\mathrm{arr}}$, however, is achieved by combined atomic and pair features ($\mathbf{X}_{a,2}$) for all descriptor dimensions considered. Pure tetrahedron descriptors from $\mathbf{X}_{a,3}$ give even worse capturing of arrangement effects than Vegard's law a_{XX}. Here, MAE$_{\mathrm{arr}}$ is almost equal for all Ω and very close to $\mathscr{D}_{1D} = d_t$. Interestingly, this baseline descriptor itself captures arrangement effects quite poorly, much worse than a_{XX} and a_{XY} (which is in contrast to the RMSE). If one finally adds the tetrahedron features to the atomic and pair ones ($\mathbf{X}_{a,2} \rightarrow \mathbf{X}_{a,4}$), MAE$_{\mathrm{arr}}$ is found to increase, i.e. $\mathbf{X}_{a,4}$ only gives a good training error but does not capture the arrangement effects in an optimal way.[3] To conclude, it turns out that optimal feature spaces for predicting the lattice constants of this data set should be based on the combination of atomic and pair features. The most elementary atomic features alone do not suffice as ingredients for $\mathscr{D}$, and the tetrahedron data can be discarded.

Analysis at learning E_{mix}: The right part of Fig. 8.1 shows RMSE and MAE$_{\mathrm{arr}}$ for learning E_{mix}.[4] As expected, also in this case the fitting error decreases with increasing Ω for all $\mathbf{X}_{E,i}$. For $\Omega = 1$ and 2, the descriptors constructed from basic atomic features ($\mathbf{X}_{E,1}$) exhibit larger values of RMSE than the ones from $\mathbf{X}_{E,2}$ and $\mathbf{X}_{E,4}$. At $\Omega \geq 3$, they are on the same level as $\mathbf{X}_{E,2}$ and $\mathbf{X}_{E,4}$, i.e. considering more complex physical quantities here does not improve the RMSE anymore. The training errors of pure tetrahedron descriptors ($\mathbf{X}_{E,3}$) are the largest except for $\Omega = 2$, where $\mathbf{X}_{E,3}$ is in line with $\mathbf{X}_{E,2}$ and $\mathbf{X}_{E,4}$. Although the effect is very small, adding tetrahedron features to atomic and pair features ($\mathbf{X}_{E,2} \rightarrow \mathbf{X}_{E,4}$) increases the RMSE, except for $\Omega = 1$ where $\mathbf{X}_{E,2}$ and $\mathbf{X}_{E,4}$ lead to identical descriptors. The rise in RMSE at adding some columns to $\mathbf{X}$ is noteworthy. One would expect that a larger feature space would at least yield the old solution that is still present in $\mathbf{X}$, if it does not find an improving combination. However, this holds only for solving the ℓ_0-problem in an exact manner. Approximating it by LASSO + ℓ_0 may indeed raise the error. This is because it can happen that LASSO selects some of the new columns to the preselection I_{ℓ_0}. These will replace some features that were part of the previous descriptors since I_{ℓ_0} has a fixed size, $\tilde{M}$. The new features might improve the preselection but there is no guarantee that the descriptors from the new subspace will be better as well.[5]

The trends of the arrangement capturing by MAE$_{\mathrm{arr}}$ in the lower right figure are less clear than for a which we relate to the higher complexity of the target E_{mix} (see Fig. 7.8). For $\mathbf{X}_{E,1}$, MAE$_{\mathrm{arr}}$ improves at increasing dimension Ω. As for learning a by $\mathbf{X}_{a,1}$, this is achieved by arrangement-

[3]This is because some features from $\mathbf{X}_{a,2}$ might be replaced by tetrahedron features that improve the RMSE although they are rather poor with respect to MAE$_{\mathrm{arr}}$.

[4]In its definition (Eq. 7.11), E_{mix} already includes a linear interpolation as a reference. We do not use additional references here therefore.

[5]As described in Sec. 2.2.5, LASSO finds the exact solution only with high probability. Recall also Fig. 5.5: it shows that the descriptors from the subsequent ℓ_0-step are not trivially constructed from the first features selected to I_{ℓ_0}, and thus demonstrates the difference between the solutions of LASSO and LASSO+ℓ_0.

sensitive nearest-neighbor differences by Eq. 5.4 at higher Ω. Descriptors from $\mathbf{X}_{E,2}$, including also pair features, generally give the best MAE_{arr} (the difference to $\mathbf{X}_{E,1}$ at $\Omega = 5$ is negligible). Adding tetrahedron features ($\mathbf{X}_{E,2} \to \mathbf{X}_{E,4}$) never improves MAE_{arr}. Pure tetrahedron features ($\mathbf{X}_{E,3}$) are on the same level as $\mathbf{X}_{E,2/4}$ only for $\Omega = 1$ whereas increasing Ω raises MAE_{arr}. Thus, analogously to the lattice constant, tetrahedron features seem to be unsuited for achieving good capturing of arrangement effects on E_{mix}. In conclusion, also for the energy of mixing of this data set, optimal feature spaces prove to be based on the combination of atomic and pair features, although the most basic atomic features are doing quite well here. Again, tetrahedron quantities appear to be dispensable.

Conclusions: In view of these results, we will focus on feature spaces combining atomic and pair features ($\mathbf{X}_{a,2}$ and $\mathbf{X}_{E,2}$) in the following sections. Let us nevertheless add some remarks on the tetrahedron features which may be of interest for similar machine-learning problems. The large RMSE for predicting a mainly seems to result from constructing $\mathbf{X}_{a,3}$ only from a single local primary feature, so the comparison to the other feature spaces appears rather unfair. However, the fact that the reference descriptor $\mathscr{D}_{1\text{D}} = d_t$ improves over a_{XX} and a_{XY} proves that this type of features *does* carry some important physical information. This similarly can be seen from the relatively good performance of the pure tetrahedron descriptors $\mathbf{X}_{E,3}$ for E_{mix}. A reason for the unsatisfactory capturing of arrangement effects by tetrahedron features might be that the geometry of the local environments in the data set often differs markedly from the very idealized regular tetrahedron shape which is assumed for the calculated tetrahedron properties. Conceptually, one might ask if tetrahedron-based features are unique representations of a material (see Sec. 4.3). As we found distinct materials s, s' with the same type of tetrahedral clusters in their supercell, and hence the same global feature by Eqs. 5.1, 5.2 and 5.2, they obviously are not.

8.1.2 Analysis of the Mathematical Operations

We now study how the mathematical operations used to construct the feature space $\mathbf{X}$ impact the performance of the resulting descriptors. On the one hand, we analyze the effect of the tiers $\mathscr{T}_i$, $i = 0, 1, 2$ as defined in Sec. 5.2.2. On the other hand, we investigate the effect of specific operations: higher moments by Eqs. 5.2 and 5.3, nearest-neighbor differences by Eq. 5.4 and the operations exp, 1/ exp, log and 1/ log. We restrict the analysis to the learning of E_{mix} because the learning of a leads to very similar results.

Analysis of Predictive Performance: Figure 8.2 shows the training error, the error on the hold-out set and MAE_{arr} (on the training data) of the resulting descriptors for this learning task. We use feature spaces based on single atom and pair primary features ($\mathbf{X}_{E,2}$) on the three complexity levels $\mathscr{T}_0$, $\mathscr{T}_1$ and $\mathscr{T}_2^{\text{r}}$ (we use the restricted $\mathscr{T}_2^{\text{r}}$ explained on p. 46 due to computational limitations in combination with the dimensionality reduction method $\mathscr{M}$ = LASSO). To study the particular effect of the mentioned operations, we skip either the higher moments, the nearest-neighbor differences or exp, 1/ exp, log and 1/ log from a feature space of complexity $\mathscr{T}_1$ which we indicate by $\mathscr{T}_1^A$, $\mathscr{T}_1^B$ and $\mathscr{T}_1^C$, respectively.

For all dimensions Ω, an increase of complexity in terms of tiers reduces the fitting error. This does not yield overfitted models because at the same time the hold-out error (top right part of figure 8.2) reduces as well. For $\mathscr{T}_1$ and $\mathscr{T}_2^{\text{r}}$, the training error converges to $\sim 0.03\,\text{eV}$. This means that for $\mathscr{T}_1$ at

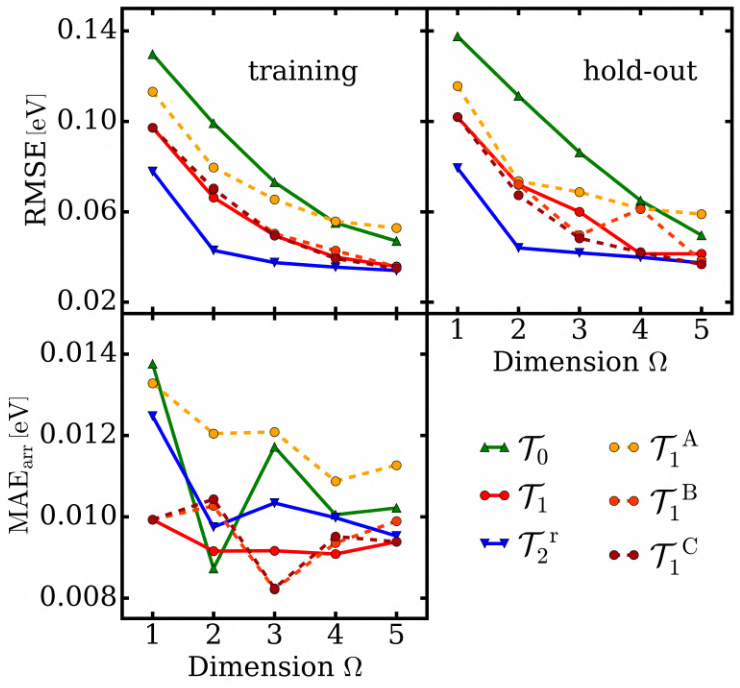


Figure 8.2: Training errors (top left), hold-out errors (top right) and MAE$_{\text{arr}}$ (bottom) at different complexity of mathematical operations for predicting E_{mix}. $\mathcal{T}_0$, $\mathcal{T}_1$ and $\mathcal{T}_2^{\text{r}}$ mark feature spaces that include features of a complexity up to tier i as defined in section 5.2. The "r" indicates that combinations by products are included in a restrictive way. $\mathcal{T}_1^{\text{A}}$, $\mathcal{T}_1^{\text{B}}$ and $\mathcal{T}_1^{\text{C}}$ mark feature spaces of complexity up to tier 1 with excluding either higher moments (A) or nearest-neighbor differences (B) or the operations exp, 1/exp, log and 1/log (C), respectively. All feature spaces are based on atomic and pair properties ($\mathbf{X}_{E,2}$).

$\Omega = 5$ and for $\mathcal{T}_2^{\text{r}}$ already at $\Omega = 3$ a good model dimension is reached. For $\mathcal{T}_0$, a further increase of Ω could attain even lower training errors while for $\mathcal{T}_0$ the expectedly increasing curve of the hold-out error indicates that models tend to be overfitted in that case. Arrangement effects (lower part of Fig. 8.2) are, surprisingly, captured best by medium complexity tier $\mathcal{T}_1$, except from $\Omega = 2$ where even the simplest $\mathcal{T}_0$ does best. However, apart from $\Omega = 1$ and, to some extent, for $\Omega = 3$, the differences in MAE$_{\text{arr}}$ between $\mathcal{T}_0$, $\mathcal{T}_1$ and $\mathcal{T}_2^{\text{r}}$ are rather small.

Figure 8.2 also shows the errors of the descriptors from $\mathcal{T}_1^{\text{A}}$, $\mathcal{T}_1^{\text{B}}$ and $\mathcal{T}_1^{\text{C}}$ that omit specific mathematical operations. For all three types of errors plotted here, both $\mathcal{T}_1^{\text{B}}$ and $\mathcal{T}_1^{\text{C}}$ do not differ much from $\mathcal{T}_1$ in general. By contrast, $\mathcal{T}_1^{\text{A}}$ has markedly higher training RMSE, hold-out RMSE and MAE$_{\text{arr}}$. First, one could argue that this is due to the fact that the feature space of $\mathcal{T}_1^{\text{A}}$ contains significantly fewer candidates compared to $\mathcal{T}_1^{\text{B}}$, $\mathcal{T}_1^{\text{C}}$ and $\mathcal{T}_1$ (see Tab. 8.3 for the feature space sizes). However, the feature space size is quite unimportant as long as the relevant features are contained at all. Hence, compared to $\mathcal{T}_1$, $\mathcal{T}_1^{\text{A}}$ is obviously missing some important candidate features, i.e. the higher moments accounting for non-linearities turn out to be an essential ingredient to the feature space in this setup. Nearest-neighbor differences as well as operations containing exp and log, in turn, can safely be disregarded in the feature space here.

Let us add two remarks to this conclusion: first, this is a quite specific finding because, for instance, nearest-neighbor differences have proven to be beneficial when using atomic features only, in

$\mathscr{T}_i$	M	$\mathscr{T}_i$	M
$\mathscr{T}_0$	105	$\mathscr{T}_1^{\mathrm{A}}$	2890
$\mathscr{T}_1$	30803	$\mathscr{T}_1^{\mathrm{B}}$	12612
$\mathscr{T}_2^{\mathrm{r}}$	3254085	$\mathscr{T}_1^{\mathrm{C}}$	21284

Table 8.3: Size $M(\mathscr{T}_i)$ of the feature spaces up to certain tiers $\mathscr{T}_i$. Note that the largest M is still below the upper boundary of Eq. 2.15 ($C = 4, N = 401, \Omega = 5$).

particular for capturing the effects of atomic arrangement (cf. Sec. 8.1.1). Here, however, pair properties like d_{XY}, which also carry information on the bonds, seem to replace them. Second, operations involving exp and log functions are to some extent problematic in a general sense as they implicitly assume some scaling constants to make their arguments dimensionless, e.g. some u' in $\exp(u/u')$. These are different from the coefficients c_i of the linear model and thus actually would require for an additional learning step to be determined systematically. As a result from this analysis, only generalized averages will be included in the feature spaces for all following tasks on **T**, while neither the operations exp, log etc. nor nearest-neighbor differences will be considered.

Complexity Analysis: Table 8.4 and Fig. 8.3 serve to analyze the complexity of the descriptors on $\mathscr{T}_0$, $\mathscr{T}_1$ and $\mathscr{T}_2^{\mathrm{r}}$ also by the complexity measures defined in Sec. 5.5.2. As can be seen, descriptors of equal Ω but different tier expectedly differ in the number of considered local primary features and hence in informational complexity N_{f_k}. For instance, at $\Omega = 1$ the descriptors are based on $N_{f_k} = 1$, 2 or 3 local primary features for $\mathscr{T}_0$, $\mathscr{T}_1$ and $\mathscr{T}_2^{\mathrm{r}}$, respectively, and on $N_{f_k} = 2$, 3 or 5 for $\Omega = 2$ (cf. Tab. 8.4). Their training error (RMSE, left part of Fig. 8.3) decreases monotonically with N_{f_k}, i.e. the more basic information is included the better the prediction. Interestingly, the curves of the three tiers roughly follow a common line. From this we conclude that it is rather irrelevant whether the descriptors combine the basic quantities in a simple linear ($\mathscr{T}_0$) or more complicated, also non-linear way ($\mathscr{T}_1$, $\mathscr{T}_2^{\mathrm{r}}$), as long as the same amount of basic information is considered.[6] As the RMSE improves only slightly after $N_{f_k} = 5$, the consideration of five primary features seems to be sufficient for this learning task.

The RMSE as a function of algebraic complexity N_{op} (middle part of Fig. 8.3) decreases substantially first and then, from $N_{\mathrm{op}} \sim 15$, basically takes a converged value. Again, a common shape of the different tiers can be observed which can be interpreted in the way that the exact type of operations, be it only linear combinations as in $\mathscr{T}_0$ or also non-linear ones, is rather irrelevant as long as their number is increasing. As an exception, the RMSE differs markedly between the tiers at $N_{\mathrm{op}} \approx 5$ where the descriptor from $\mathscr{T}_0$ is best. We relate this to the difference in free parameters as the 4-

[6]This can be understood the following way: suppose two primary features u and v which are close to linear around their means u_0 and v_0 (this is a common behavior of the used candidate features). Let these primary features be further combined to a feature $f(u, v) = \exp(u + v)$. Then, for small ranges of u and v around u_0 and v_0, $f(u, v)$ can be linearly approximated by the expansion $f(u, v) \propto (\partial_u f)(u_0, v_0) \cdot u + (\partial_v f)(u_0, v_0) \cdot v = c_1 \cdot u + c_2 \cdot v = g(u, v)$. So, a rather complex single feature – that could form a 1D descriptor – is equivalently expressed by a two-dimensional linear combination of simpler features, i.e. a 2D descriptor. This can be called a trade-off between descriptor dimensionality and feature complexity.

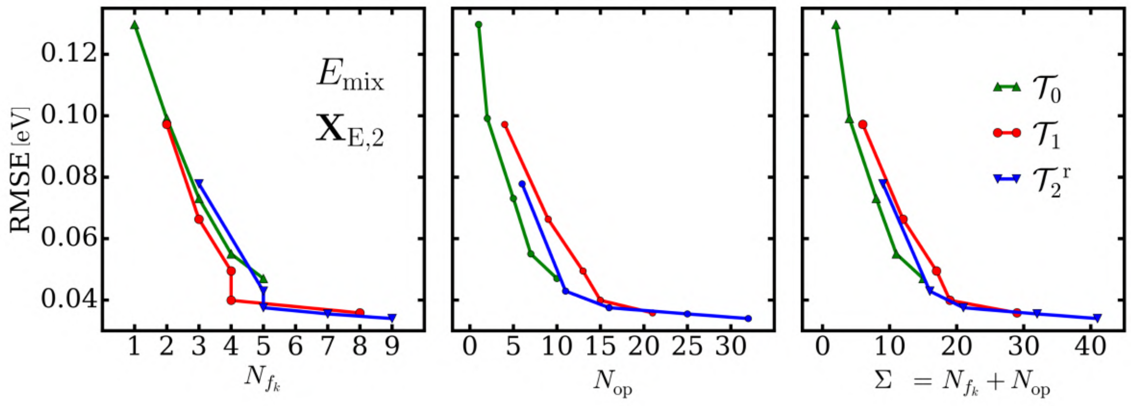


Figure 8.3: Training error vs. descriptor complexity for the three tiers $\mathscr{T}_0$, $\mathscr{T}_1$ and $\mathscr{T}_2{}^{\mathrm{r}}$. Left: RMSE vs. informational complexity N_{f_k}. Middle: RMSE vs. algebraic complexity N_{op}. Right: RMSE vs. total complexity $\Sigma = N_{f_k} + N_{\mathrm{op}}$. The curves go from $\Omega = 1$ (most left) to $\Omega = 5$ (most right) dot by dot.

dimensional descriptor of $\mathscr{T}_0$ includes already four fitted constants $c_{1,\ldots,4}$ whereas the corresponding one-dimensional descriptors of $\mathscr{T}_1$ and $\mathscr{T}_2{}^{\mathrm{r}}$ contain just one fitted constant c_1.

Also for Σ (right part of Fig. 8.3) the training errors of the three tiers approximately follow a common curve. The common line decreases substantially first and then, from $\Sigma \sim 15$, it flattens, i. e. an increase of total complexity is most beneficial at low total complexity. This uncovers a general dependency of the fitting error on total complexity where the different tiers cover different regions of complexity (given that the Ω range is limited). Such a behavior is typical for models in symbolic regression and matches the Pareto frontier explained in Sec. 2.3.2. Finally, while most of the results presented here are very similar for learning the lattice constants, the patterns of the RMSE as a function of N_{f_k} differ where only the curves of $\mathscr{T}_0$ and $\mathscr{T}_1$ share their shape. The one of $\mathscr{T}_2{}^{\mathrm{r}}$, however, lies below those two, i.e. including the product combinations brings some additional benefit here.

8.2 Analysis of Dimensionality Reduction Methods

8.2.1 Comparing LASSO's Solver Algorithms

In this project, initially the coordinate algorithm in its cyclic variant was used as solver for LASSO (see Sec. 2.2.5), as in Ref. [2]. Unfortunately, we found that in this case the feature selection of LASSO depends on the column order of the feature matrix **X**. To demonstrate this explicitly, we did a numeric test on the learning task to predict E_{mix} of the data-set **T** with feature space $\mathbf{X}_{E,2}$. The feature selection by LASSO at decreasing hyperparameter λ was run two times independently where the order of columns in **X** was exchanged randomly. In both runs the first seven features selected to I_{ℓ_0} were absolutely identical though their coefficient profiles $c_i(\lambda)$ differed. Features 8 however indeed represented the different physical parameters $\sqrt{|E_{f,p} - E_{f,b}|}$ and $1/\sqrt[3]{|E_{g,b}|}$ $(f_k \neq f_k')$. Obviously, this dependency of feature selection on the order of columns is disadvantageous. The applied cyclic iteration variant might even prefer small column indices which refer to the simpler candidate features according to our feature engineering scheme. As explained in Sec. 2.2.5, the alternative

$\mathscr{T}_i$	1D			N_{f_k}	N_{op}	Σ
$\mathscr{T}_0$	$\tilde{E}^2_{g,p}$			1	1	2
$\mathscr{T}_1$	$(\tilde{E}^2_{g,p} + \tilde{E}^2_{g,XX})^3$			2	4	6
$\mathscr{T}_2^{\text{r}}$	$(\bar{E}^2_{f,XX}) \cdot \sqrt{\lvert \bar{r}^2_s - \bar{d}^2_{XY} \rvert}$			3	6	9
$\mathscr{T}_i$	2D			N_{f_k}	N_{op}	Σ
$\mathscr{T}_0$	$\tilde{d}^2_{XY}$	+	$E_{g,b}$	2	2	4
$\mathscr{T}_1$	$(\tilde{E}^2_{g,p} + \tilde{E}^2_{g,XY})^3$	+	$\exp(\Delta E_{g,p} + \Delta E_{g,XX})$	3	9	12
$\mathscr{T}_2^{\text{r}}$	$\tilde{d}^2_{XY} \cdot \left(\bar{\Delta} E^2_{g,X} + \bar{E}^2_{g,b}\right)^2$	+	$E_{m,p} \cdot \sqrt[3]{\lvert r_s - d_{XY} \rvert}$	5	11	16

Table 8.4: Complexity measures for 1D and 2D descriptors resulting from feature spaces $\mathbf{X}_{E,2}$ on complexity $\mathscr{T}_0$, $\mathscr{T}_1$ and $\mathscr{T}_2^{\text{r}}$ for predicting E_{mix} of the ternaries **T**. Listed are the components of the descriptors, the informational complexity N_{f_k}, the algebraic complexity N_{op}, and the total complexity Σ.

algorithm LASSO-LARS should overcome this issue. Working on the same task as above, we indeed could not observe an influence of column order. From this background, we changed to work with LASSO-LARS which additionally brings significant benefits in computation time (see Tab. C.2 in the Appendix).

8.2.2 Comparison between LASSO+ℓ_0 and SISSO+ℓ_0

As described in Sec. 4.4.2, two main drawbacks of LASSO+ℓ_0 are its problems to find optimal descriptors from a highly correlated feature space and a rather strong limitation on matrix size which SISSO, as an alternative to LASSO, has been proven to overcome. Hence, it is interesting to compare these two feature selection methods at identifying the descriptors. For this, we compare first the preselected subsets I_{ℓ_0} and then the resulting 5D descriptors. We apply strategy $\mathscr{S}_{\text{all}}$ in this subsection and study the learning of a and E_{mix} with complexity $\mathscr{T}_1$, $\mathscr{T}_2^{\text{r}}$ and $\mathscr{T}_2$ (the size of the feature spaces is denoted in Tab. 8.6). For predicting the lattice constant a on $\mathscr{T}_2^{\text{r}}$, the left part of Fig. 8.4 shows heat maps of pairwise correlation, with LASSO in the lower-left triangle and SISSO in the upper-right one.

Note here, that the subset I_{ℓ_0} obtained with SISSO contains only 29 instead of 30 features. This can happen when certain features are repeatedly selected to different subsets $\tilde{\mathbf{S}}_j$. The used feature space complexity $\mathscr{T}_2^{\text{r}}$ poses a challenging case for LASSO because it provides many highly correlated candidate features. Indeed, highly correlated features dominate the LASSO subspace as is evident from the predominance of yellow in the lower-left triangle. Importantly, there are only four pairs weakly correlated as indicated by the four red bars in the triangle of LASSO in Fig. 8.4 (left). Thus, all possible 5D descriptors from this subspace will inevitably contain at least one highly correlated feature pair (see below). By contrast, in the preselection obtained with SISSO, most of identified features are low correlated (notice the predominance of red). High correlation basically is observed only in the small yellow triangles in the heat-map. These correspond to the bunch-like subsets $\tilde{\mathbf{S}}_j$ which, by construction of SISSO (cf. 4.4.2), contain strongly correlated features.

The right part of Fig. 8.4 compares *mean* absolute correlations $\overline{|\rho_{i,j}|}$ averaged over the total preselected sets I_{ℓ_0} versus feature space complexity. This is done with the smaller ($\mathscr{T}_1$) and the larger feature matrix ($\mathscr{T}_2^{\text{r}}$), respectively, for both predicting a and E_{mix}. Clearly, in all cases the overall correlation $\overline{|\rho_{i,j}|}$ is higher for subsets obtained with LASSO compared to SISSO. Moreover, at increasing matrix size, the mean correlation increases for LASSO and decreases for SISSO. For the more difficult learning task of predicting E_{mix}, the overall correlation is lower compared to the simpler task of learning a when using LASSO, while for SISSO it is vice versa. As a general conclusion, SISSO selects subsets I_{ℓ_0} with less redundancy in terms of $|\rho_{i,j}|$, in particular for large and highly correlated feature spaces. This is a prerequisite for the following ℓ_0-step to select low-correlated descriptors.

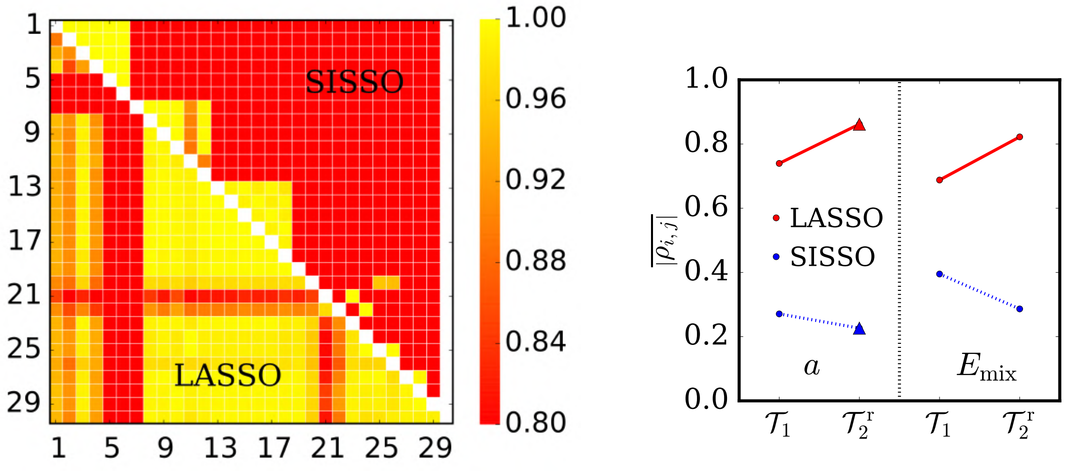


Figure 8.4: Comparison of preselected subspaces obtained by LASSO and SISSO. Left: heat map of the absolute correlation $|\rho_{i,j}|$ between the features in the preselected subsets I_{ℓ_0} for learning a with feature space complexity $\mathscr{T}_2^{\text{r}}$. The plotted range of $|\rho_{i,j}|$ is between 0.8 or less (red) and 1.0 (yellow) and the axes refer to the feature index in I_{ℓ_0}. Lower triangle: subset from LASSO, upper triangle: subset from SISSO. The preselection from SISSO contains only 29 instead of 30 features. Right: $|\rho_{i,j}|$ averaged over the total subselection versus feature space complexity for predicting a (left) and E_{mix} (right). Triangles indicate the learning of a with $\mathscr{T}_2^{\text{r}}$ that is shown in the left plot.

In Figures 8.5 and 8.6, the correlations of the 5D *descriptors* are exemplarily compared after applying the ℓ_0-step to either the LASSO or the SISSO preselection. Table 8.5 lists the formulas of the descriptor components. In the heat map in Fig. 8.5, pairwise correlations are compared for predicting a on complexity level $\mathscr{T}_2^{\text{r}}$. For LASSO, note the high correlation between components 1, 2 and 3 indicated by the yellow and orange squares in the lower triangle. Here, components 1 and 2 are particularly highly correlated ($|\rho_{1,2}| = 0.999$), and also appear to have a similar algebraic form (see Tab. 8.5). Their numerators are identical and their denominators are almost proportional to each other. This is first because of $\sqrt{d_{XX}}$ being almost proportional to d_{XX}, due to the narrow range of d_{XX}, and second because $d_{XX} \approx a_{XX}$ holds from physical reasons. So, components 1 and 2 are virtually equal after standardization and, hence, one of them can be considered as redundant. Also for components 1 and 3 as well as 2 and 3 $|\rho_{i,j}|$ takes high values ($|\rho_{1,3}| = 0.935$ and $|\rho_{2,3}| = 0.929$). However, they seem to carry different information as can be taken from the larger dispersion in the middle correlation plot in Fig. 8.6 (here, only the correlation for components 1 and 3 are shown - the analogous figure for components 2 and 3 is almost identical due to the high correlation of components 1 and 2). In addition, the right part of Fig. 8.6, showing the correlation of components 1 and 4 ($|\rho_{2,3}| = 0.816$), gives an example of relatively uncorrelated features. Let us remark that interestingly only one pair is really strongly correlated which, as said above, is a necessary consequence of the structure in the pre-selection. The ℓ_0-step itself obviously makes its best out of the offered features. By contrast to the LASSO descriptor, the 5D descriptor from SISSO does not contain highly correlated components.

The right part of Fig. 8.5 compares the mean correlation $\overline{|\rho_{i,j}|}$ of the 5D descriptors, similar to Fig. 8.4. For LASSO, increasing feature space complexity rises (in case of E_{mix}) or slightly drops (in case of a) $\overline{|\rho_{i,j}|}$. For SISSO, $\overline{|\rho_{i,j}|}$ always decreases at increasing complexity. Most importantly, $\overline{|\rho_{i,j}|}$ of a SISSO descriptor is always lower than for the LASSO counterpart. In these cases SISSO is able to identify less intercorrelated descriptors, even for large and correlated feature spaces.

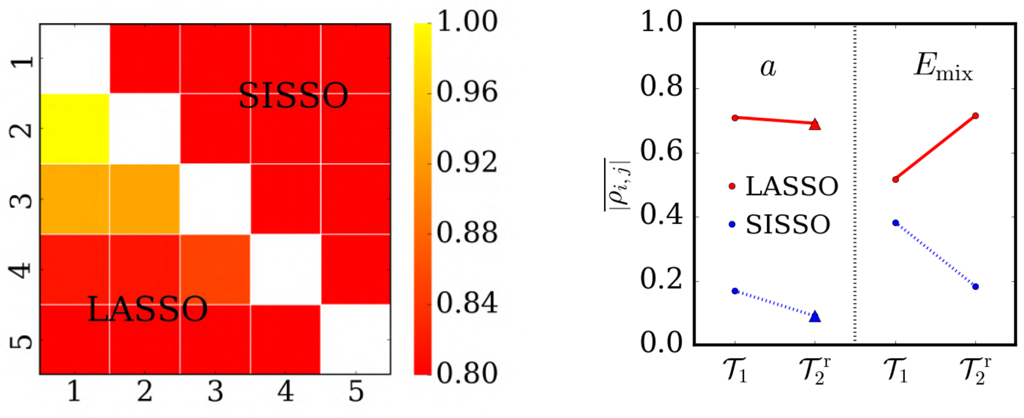


Figure 8.5: Comparison of 5D descriptors obtained by LASSO and SISSO. Left: heat map of the absolute correlation $|\rho_{i,j}|$ of the components of the 5D descriptors for learning a with feature space complexity $\mathcal{T}_2^{\mathrm{r}}$. Lower triangle: LASSO+$\ell_0$, upper triangle: SISSO+$\ell_0$. Right: $|\rho_{i,j}|$ averaged over 5D descriptor component pairs versus feature space complexity for predicting a and E_{mix}.

It is also interesting to compare the predictive power of descriptors from LASSO and SISSO. For the task of predicting a, this is shown in the left part of Fig. 8.7 with fitting error in the top panel and hold-out error in the bottom one. Both LASSO and SISSO were applied to feature spaces with complexity $\mathcal{T}_1$ and $\mathcal{T}_2^{\mathrm{r}}$. In these cases, the fitting error does not differ much between LASSO and SISSO. The hold-out error indicates that for larger Ω, SISSO descriptors may be overfitted. Additionally, results from $\mathcal{T}_2$ including *all* feature combinations by products are shown. This case was run only with SISSO since LASSO is limited due to its high demands in terms of memory. Fitting errors are further reduced with $\mathcal{T}_2$ but descriptors are overfitted for $\Omega \geq 3$. It seems that ~ 0.03 Å is a limit in predictive performance which is in the order of our estimation of the irreducible error (see Tab. 7.3).

Similarly, in the right panel of Fig. 8.7 the predictive power is compared for E_{mix}. Fitting errors of LASSO and SISSO are close to each other for $\mathcal{T}_1$ and $\mathcal{T}_2^{\mathrm{r}}$ and approach at ~ 0.04 eV at increasing Ω. The hold-out errors indicate that even for $\Omega = 5$ there is no overfitting. For $\mathcal{T}_1$ and $\Omega \geq 3$, LASSO descriptors are slightly better than the SISSO counterparts. Also for E_{mix}, the feature space

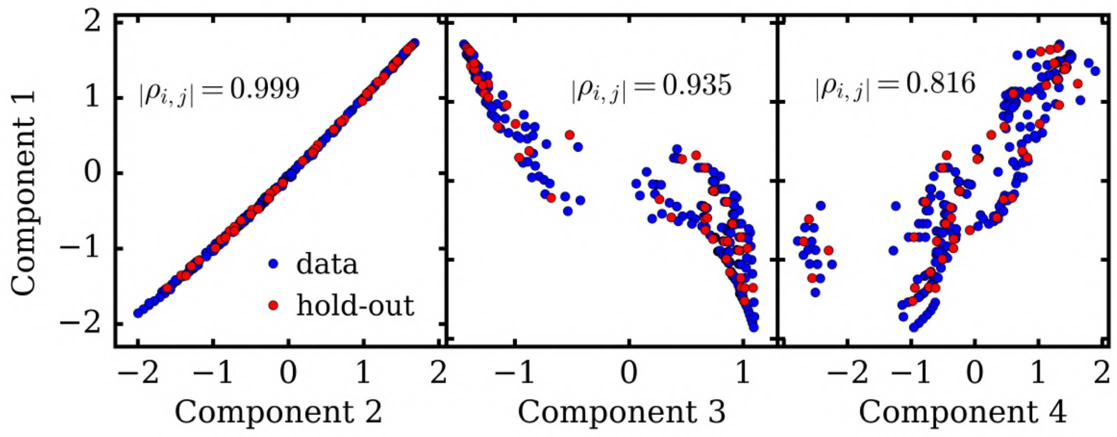


Figure 8.6: Correlation plots for the pairs of components (1,2), (1,3) and (1,4) of the 5D descriptor from LASSO+ℓ_0 on $\mathscr{T}_2^{\,\mathrm{r}}$ for predicting a (cf. Tab. 8.5). These pairs give examples of high, medium and low correlation in terms of the color bar in Fig. 8.5.

No.	LASSO	SISSO
1	$\frac{\bar{a}_{XY}^2}{\sqrt{d_{XX}}}$	$\frac{\bar{a}_{XY}^2}{\sqrt{\bar{r}_p^2+\bar{a}_{XY}^2}}$
2	$\frac{\bar{a}_{XY}^2}{a_{XX}}$	$\frac{\tilde{Z}^3}{\bar{r}_p^2+\bar{r}_p^3}$
3	$\sqrt[3]{\lvert\bar{r}_p^2-\bar{d}_{XY}^2\rvert}$	$\bar{r}_d^2\cdot\sqrt{\tilde{a}_{XY}^3}$
4	$\bar{d}_{XY}^3\cdot\sqrt[3]{\lvert\tilde{r}_d^2-\tilde{d}_{XY}^3\rvert}$	$\tilde{a}_{XY}^2\cdot(\tilde{d}_{XX}^2-\tilde{a}_{XY}^3)^3$
5	$\frac{\tilde{d}_{XY}^3}{\sqrt[3]{\tilde{r}_s^2+\tilde{r}_s^3}}$	$\frac{\tilde{P}^3}{\tilde{r}_s^2+\tilde{r}_d^3}$

Table 8.5: Components of the 5D descriptors for predicting a with feature space complexity $\mathscr{T}_2^{\,\mathrm{r}}$, obtained by LASSO+$\ell_0$ or SISSO+ℓ_0, respectively. The components are sorted by correlation $|\rho|$ with a in decreasing order.

complexity $\mathscr{T}_2$ (again only run with SISSO) reduces the fitting error without leading to overfitted descriptors (the hold-out error seems to reach a plateau).

To summarize, in the studied cases the descriptors from LASSO and SISSO are very similar with respect to predictive power. As expected, descriptors from SISSO benefit from less redundant components and could handle feature matrices being too big for LASSO. Note, finally, that the one-dimensional descriptors from LASSO and SISSO always coincide which follows by construction

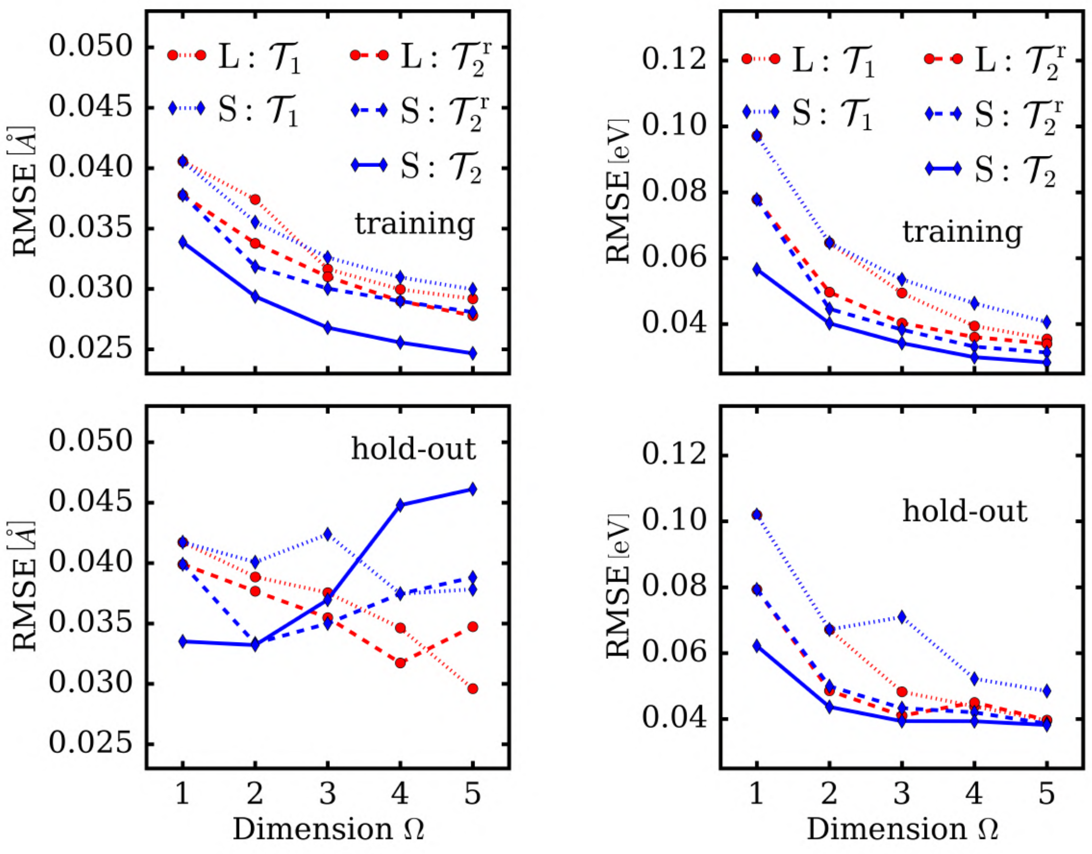


Figure 8.7: Comparison of predictive performance of descriptors up to $\Omega = 5$ from LASSO (red) and SISSO (blue) at varying complexity of feature space for predicting a (left) and E_{mix} (right). Top panels: training error vs. descriptor dimension. Bottom panels: hold-out error versus descriptor dimension.

a		E_{mix}	
$\mathcal{T}_i$	$M(\mathcal{T}_i)$	$\mathcal{T}_i$	$M(\mathcal{T}_i)$
$\mathcal{T}_1$	5257	$\mathcal{T}_1$	8736
$\mathcal{T}_2^r$	239797	$\mathcal{T}_2^r$	572351
$\mathcal{T}_2$	13820653	$\mathcal{T}_2$	38163216

Table 8.6: Size $M(\mathcal{T}_i)$ of the feature spaces used to compare LASSO and SISSO as well as the model selection strategies. Note that the largest size does not exceed the upper limit of Eq. 2.15 $(C = 4, N = 401, \Omega = 5)$.

of these two methods.[7]

[7] Both LASSO and SISSO add the feature of highest correlation to I_{ℓ_0} first and LLSR will identify this as one-dimensional solution in the ℓ_0-step (also see the discussion of Sec. 8.2.3).

8.2.3 Further Dimensionality Reduction Methods

LASSO and SISSO are advanced dimensionality reduction methods, improving over earlier and simpler methods. Examples for older techniques are matching pursuit (Sec. 2.2.6), orthogonal matching pursuit (Sec. 2.2.7) and SIS (Sec. 2.2.8) that, in a way, can be thought of as SISSO's precursors. For comparison, these were implemented as well in the ML code of this thesis. Here, we test them on the learning task of above, to predict E_{mix} of $\mathbf{T}$ by descriptors up to $\Omega = 5$ extracted from a feature space of complexity level $\mathscr{T}_1$.

Following the general workflow of Fig. 5.1, MP, OMP and SIS were used as methods $\mathscr{M}$ to find a subset I_{ℓ_0} of the most relevant features. Also here, the subset size was set to $\tilde{M} = 30$ and the feature matrix $\mathbf{X}$ was standardized to its mean and standard deviance before. Optimal descriptors then were found by applying the ℓ_0-search to the training data (strategy $\mathscr{S}^{\text{all}}$). Validation was performed on the general hold-out set $\mathbf{S}_{\text{ho}}$.

Resulting fitting and hold-out errors are shown in Fig. 8.8 where we additionally plotted the numbers of LASSO and SISSO from the previous discussion. As can be seen, the 1D solution is on the same level of performance for all methods since it coincides. This follows from the construction of the methods that equally select the feature of highest correlation with the target $\mathbf{P}$ to the subset first. Once it is present in the subset, the subsequent ℓ_0-search will prefer this feature over all others and result it as 1D descriptor.[8]

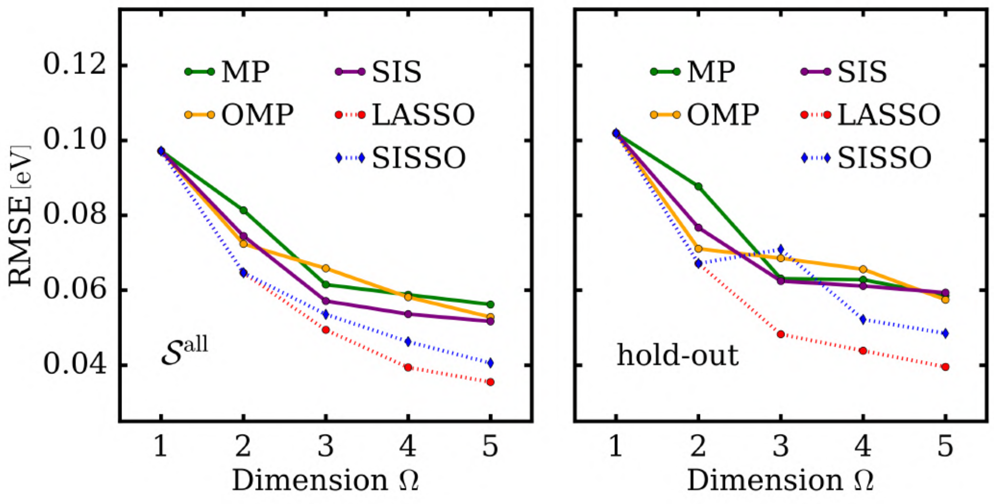


Figure 8.8: Model performance at using the dimensionality reduction methods MP, OMP and SIS. RMSE from $\mathscr{S}^{\text{all}}$ vs. Ω (left plot) and hold-out RMSE (right plot) E_{mix} of $\mathbf{T}$, complexity $\mathscr{T}_1$. Performance of LASSO and SISSO from Fig. 8.7 (top right panel) is added for comparison. The underlying numbers are reported in Tab. 8.7.

For higher Ω, descriptors from MP, OMP and SIS subsets differ. Although differences in training and hold-out RMSE are small, SIS generally gives the best results. Importantly, the descriptors from the LASSO and SISSO subsets achieve better performance (except for SISSO at $\Omega = 3$ on $\mathbf{S}_{\text{ho}}$). Hence,

[8]This also can be understood from a geometric perspective: the feature of highest correlation $\mathbf{x}_i$ has - by definition of ρ - the smallest angle to $\mathbf{P}$. The ℓ_0-search for $\Omega = 1$ will yield coefficients c_k that project all features $\mathbf{x}_k$ to the ℓ_2-sphere containing $\mathbf{P}$. Obviously, the model $c_j\mathbf{x}_j$ on this sphere with the lowest ℓ_2-distance to $\mathbf{P}$ has the smallest angle to $\mathbf{P}$, and hence $i = j$.

these two newer approaches indeed prove as advancement over their precursors in this application. It is also interesting, to study the (pairwise) correlation of features in the subsets I_{ℓ_0} from MP, OMP and SIS as a lower number of highly correlated features is a precondition for non-redundant descriptor components. Thus, the mean Pearson correlation $\overline{|\rho_{i,j}|}$ was calculated for the three methods which is shown in Fig. 8.9. Obviously, the preselected subsets from both MP and SIS contain on average very highly correlated features. This is an expectable result as the construction principles of both techniques do not focus on finding orthogonal features. Accordingly, OMP improves over both of them and even over LASSO that, as already addressed, may have problems with a highly correlated feature space. From all methods, however, the best is SISSO, as an extension of OMP by the selecting bunches of features in every iteration.

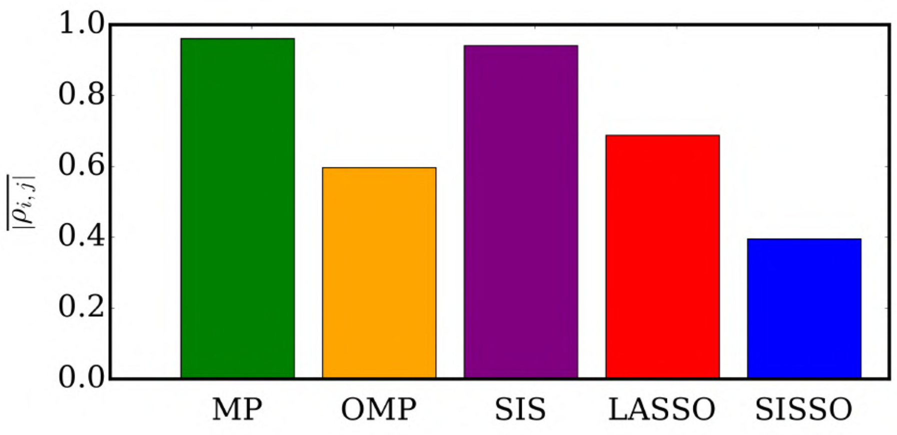


Figure 8.9: Comparison of mean correlation $\overline{|\rho_{i,j}|}$ for the subsets I_{ℓ_0} obtained by $\mathscr{M}$ = MP, OMP and SIS on the learning task of Fig. 8.8. Values for LASSO and SISSO are equivalent to Fig. 8.4 (right panel).

With respect to the computational effort, any improvement in feature selection comes along with the cost of a larger execution time, as shown in Fig. 8.10. In the same learning task as above, the trends between the methods are just the other way around, compared to the mean correlation of Fig. 8.9. The best achieved feature selection is achieved by the most elaborate method SISSO and accordingly requires the maximum in time. Importantly, the three simple methods and the "higher level" ones LASSO and SISSO differ in mean execution time by more than one order of magnitude (note the different scales in Fig. 8.10).

8.2.4 Comparison with KRR

For an additional comparison, we performed also a kernel ridge regression on the same ML task as above. As explained in Sec. 2.3.1, KRR is a conceptually different method that does not include a dimensionality reduction or feature selection. Here, we use it as a full learning method $\mathscr{L}$ by the scheme of Fig. 5.1. As it is a non-linear technique itself, we constructed the feature space $\mathbf{X}$ solely using the global primary primary features as defined by Eq. 5.1. A further augmentation in the manner of symbolic regression, also by higher moments, was not applied. We did this since KRR neither would identify special low-dimensional expressions (that could be interpreted) nor needs for an additional consideration of non-linearity. Specifically, $\mathbf{X}$ was constructed from the 13

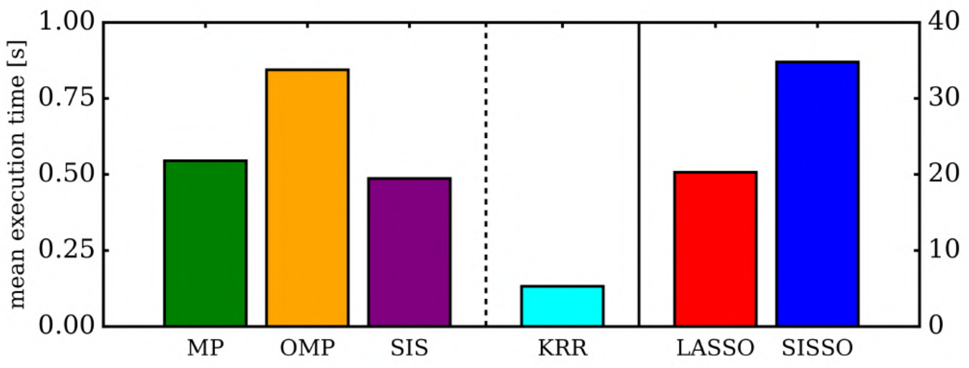


Figure 8.10: Profiling of the dimensionality reduction methods MP, OMP, SIS, LASSO-LARS and SISSO, and of the learning method KRR. Plotted is the mean execution time in s averaged over 10 runs of subset selection ($\tilde{M}$=30, for KRR: total training time). Learning task was E_{mix} from a complexity $\mathscr{T}_1$ feature space. Execution times of LASSO and SISSO are plotted with respect to the right axis. Runs were performed on a quadcore machine with Intel i5 1.6 GHz processors and 8 GB RAM without parallelization.

local primary features listed in Tab. 8.2. KRR was used with a Gaussian kernel as defined in Sec. 2.3.1. The model was obtained by the function `sklearn.kernel_ridge` of the Scikit-learn package ([112, 138]). Its regularization strength `alpha` (α) corresponds to λ in Eq. 2.21, and the parameter `gamma` (Γ) of the Gaussian kernel transforms to the width σ by $\Gamma = 1/2 \cdot \sigma^{-2}$.

Preliminary to the actual learning, hyperparameters α and σ were determined by minimizing $\overline{\mathrm{err}}_{\mathrm{te}}$ (MSE, equivalent to RMSE) from a CV procedure. For this, α and σ were varied on a 21×21 logarithmic grid defined by $10^{-25} \leq \alpha \leq 1$ and $0.1 \leq \sigma \leq 10^5$. At each grid point, $\overline{\mathrm{err}}_{\mathrm{te}}$ was determined by L10%OCV with $N_{\mathrm{CV}} = 50$, i.e. by 50 individual KRR models trained for a given (α, σ). Analogously to the other learning methods, $\mathbf{X}$ was standardized columnwise to the mean and standard deviation. The logarithmic heat map in Fig. 8.11 shows the results of this procedure. The minimum in $\overline{\mathrm{err}}_{\mathrm{te}}$ is taken at $\alpha \approx 1.7 \cdot 10^{-4}$ and $\sigma \approx 1.6$ which is used in the following.

Fig. 8.12 compares the KRR model trained with $\mathscr{S}^{\mathrm{all}}$ with the 5-dimensional descriptors from LASSO (on complexity $\mathscr{T}_1$) and SISSO ($\mathscr{T}_2^{r}$) with regard to target-prediction correlation at learning E_{mix}. Very interestingly, KRR performs significantly better than LASSO and SISSO, both on the training data and the hold out set ($\mathbf{S}_{\mathrm{ho}}$), i.e. this improvement is not at the cost of overfitting. As can be seen from Tab. 8.7, this superiority is equally present for other error metrics - in particular MAE_{arr} - and, naturally, is observed at comparing with the dimensionality reduction methods MP, OMP and SIS as well. Finally, also with respect to the computation time, KRR "succeeds" over all other approaches.

However, it has to kept in mind that this comparison is unfair in a threefold manner: (1) KRR is a conceptually much simpler method because it directly fits the model without a time-consuming iterative selection of features. (2) As addressed in Sec. 2.3.1, the number of adaptable parameters in KRR is equal to the number of samples N while the sparse Ω-dimensional descriptors, after the dimensionality reduction and the ℓ_0-step, rely on $\Omega + 1$ parameters only (including the offset). (3) KRR here yields *black box* models while the other approaches unveil interpretable descriptors, at the cost of higher errors.

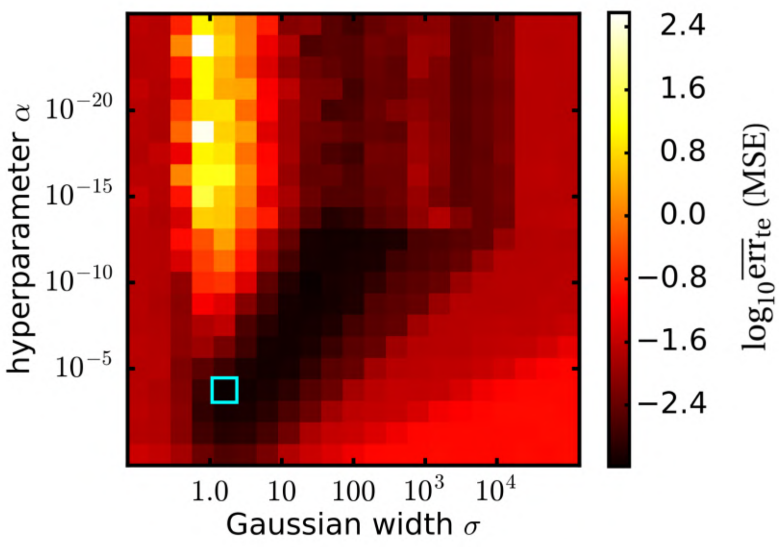


Figure 8.11: Heat map of logarithmic average test error $\log_{10}\overline{\text{err}}_{\text{te}}$ (MSE) on the grid of the two hyperparameters α and σ from learning by KRR (E_{mix} of **T**). The minimum value is indicated by the blue square.

	metric	KRR	LASSO	SISSO	MP	OMP	SIS
$\mathscr{S}^{\text{all}}$	RMSE	0.015	0.036	0.041	0.056	0.053	0.052
	MAE	0.008	0.026	0.029	0.043	0.039	0.040
	maxAE	0.089	0.153	0.179	0.184	0.221	0.176
	MAE_{arr}	0.005	0.015	0.011	0.010	0.011	0.010
S_{ho}	RMSE	0.031	0.040	0.048	0.059	0.057	0.059
	MAE	0.017	0.029	0.037	0.045	0.041	0.046
	maxAE	0.078	0.097	0.133	0.149	0.185	0.157
	MAE_{arr}	$2\cdot10^{-4}$	0.042	0.001	0.001	0.001	0.001

Table 8.7: Performance of the different learning methods. Learning task: E_{mix} of **T**. KRR is based on primary features of complexity $\mathscr{T}_0$, all other values refer to 5D descriptors, obtained by an ℓ_0-step. For further details, see the previous figures and Sec. 8.2.3.

8.3 Analysis of Model Selection Strategies

In this section we apply the model selection strategies $\mathscr{S}_{\text{all}}$, $\mathscr{S}_{\text{CV}}^{\text{tr}}$ and $\mathscr{S}_{\text{CV}}^{\text{te}}$, described in Sec. 5.4, to the task of predicting the lattice constant a of the ternaries data-set **T**. We apply LASSO-LARS only and, again, use feature spaces of complexity $\mathscr{T}_1$ and $\mathscr{T}_2^{\text{r}}$ that differ in size and degree of feature correlation. The results are shown in Fig. 8.13 where the top part refers to $\mathscr{T}_1$ and the bottom to $\mathscr{T}_2^{\text{r}}$. In both cases, the distribution of training errors (left), the distribution of test errors (middle), and

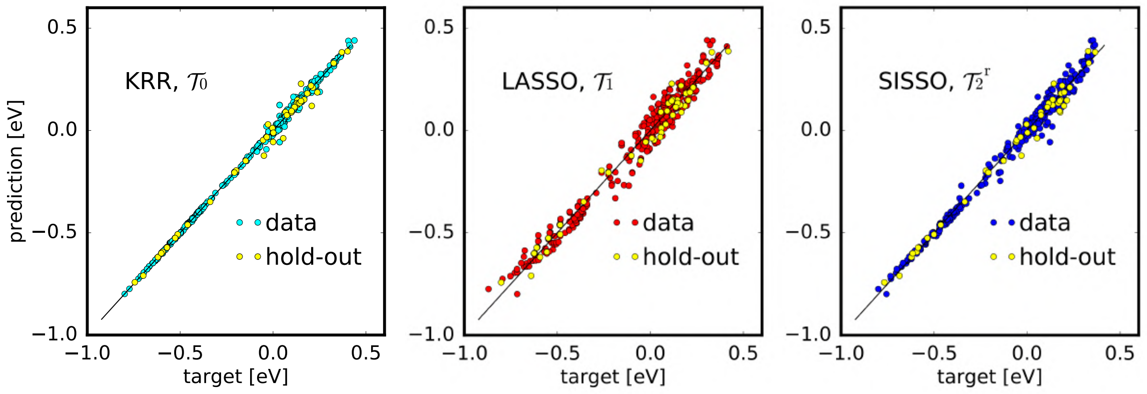


Figure 8.12: Comparison of correlation for models trained by KRR (left), LASSO (middle) and SISSO. Learning task: E_{mix} of **T**. KRR is based on primary features of complexity $\mathcal{T}_0$, 5D descriptors of LASSO and SISSO on $\mathcal{T}_1$ and $\mathcal{T}_2^{\mathrm{r}}$, respectively. Models were obtained by $\mathscr{S}^{\mathrm{all}}$ and applied to the hold-out set.

the hold-out errors (on the right) are depicted. The error distributions result from sanity checks (SC), i.e. from CV runs with the descriptors obtained by the three strategies kept fixed. Additionally, the averages $e\bar{r}r^{\mathrm{tr}}$ and $e\bar{r}r^{\mathrm{te}}$ by Eq. 5.10 are indicated. For comparison, note the errors of the baseline 1D descriptor a_{XX} (including fitting constants c_0 and c_1) of approximately 0.07 eV equally on training, test and hold-out data.

First we observe that the strategies throughout yield different models, except for $\Omega = 1$ at $\mathcal{T}_1$ where identical models are found that obviously have identical errors. The training errors of all three strategies (left panels) are not found to have a high dispersion at both complexities. This is in accordance with our expectation since all three strategies include a fitting to one or several training sets at some point in their construction: even in $\mathscr{S}_{\mathrm{CV}}^{\mathrm{te}}$ the test errors are calculated for models fitted to a training split. Mostly, the strategies $\mathscr{S}_{\mathrm{all}}$ and $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ are very close in average training errors $e\bar{r}r^{\mathrm{tr}}$, with $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ having slightly smaller values. For complexity $\mathcal{T}_2^{\mathrm{r}}$, $\mathscr{S}_{\mathrm{CV}}^{\mathrm{te}}$ is very similar as well, for $\mathcal{T}_1$ and some descriptor dimensions Ω it is markedly worse, however. Also these observations are intuitive: descriptors of $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ have, by their definition, lowest training errors in many CV iterations and hence should be best here. The descriptors from $\mathscr{S}_{\mathrm{all}}$ are fitted to the full training data once, so they should not perform significantly worse on the partitioned training data. Models from $\mathscr{S}_{\mathrm{CV}}^{\mathrm{te}}$ are selected by individual and average minimum test errors where they eventually accept higher training errors.

Test errors (middle panels) generally have a narrow distribution at the simple feature space on $\mathcal{T}_1$ (top figure) and a wide distribution at the complex feature space on $\mathcal{T}_2^{\mathrm{r}}$ (bottom figure). This increase of variance is a typical behavior at rising model complexity (see Sec. 2.4.1 and Ref. [27]). Between the strategies no clear difference in error distributions and average errors $e\bar{r}r^{\mathrm{tr}}$ is apparent. In particular, we do not find less variance at $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ as a sign for an enhancement in model robustness. The tendency of the descriptors from $\mathscr{S}_{\mathrm{CV}}^{\mathrm{te}}$ to have slightly lower test errors is intuitive as they are selected on this type of error. We do not observe a superiority of $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ over $\mathscr{S}_{\mathrm{all}}$ that would indicate its better generalization we aimed at.

In the right panels of Fig. 8.13, generally $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ yields the smallest hold-out errors and $\mathscr{S}_{\mathrm{CV}}^{\mathrm{te}}$ tends to yield the largest. If this quantity is used as criterion for model generalizability, hence $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ indeed improves over $\mathscr{S}_{\mathrm{all}}$ while $\mathscr{S}_{\mathrm{CV}}^{\mathrm{te}}$ does not. Analogous results for predicting E_{mix}, which are not shown

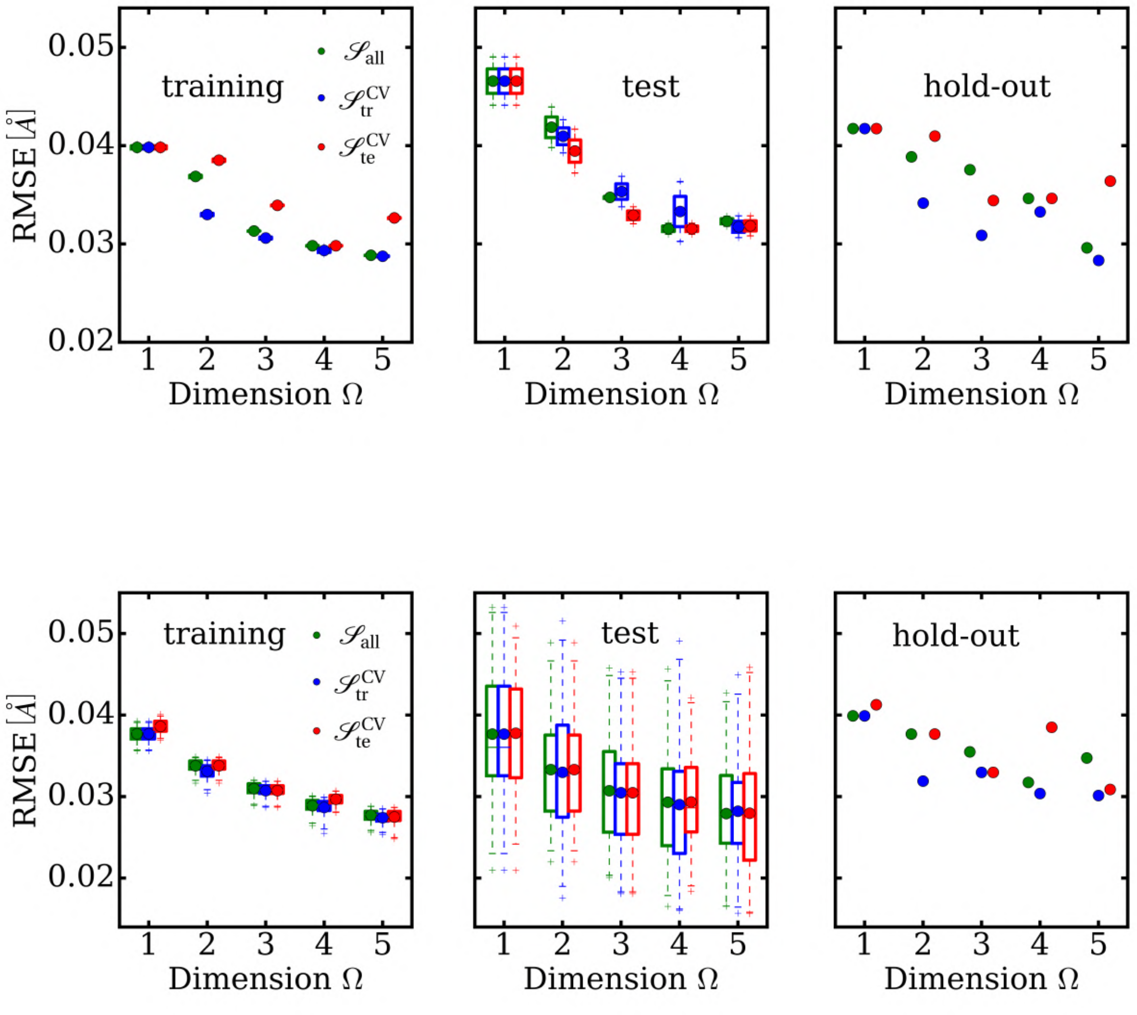


Figure 8.13: Training (left) and test (middle) errors from CV, and hold-out errors (right) vs. descriptor dimension Ω for the different model selection strategies. Green dots refer to $\mathscr{S}_{\mathrm{all}}$, blue to $\mathscr{S}_{\mathrm{tr}}^{\mathrm{CV}}$, and red to $\mathscr{S}_{\mathrm{te}}^{\mathrm{CV}}$. Learning task: a of the ternaries **T**, with features of complexity $\mathscr{T}_1$ (top plots) and $\mathscr{T}_2^{\mathrm{r}}$ (bottom plots). The left and middle plots show the error distributions from SC procedures ($N_{\mathrm{CV}} = 50$) at fixed descriptor components as boxplots. Here, the central box marks the interquartile range between the 1st and the 3rd quartile, the line in the middle the median, the whiskers the range between the 1- and 99-percentile and the crosses the minimum and the maximum of the distribution. The dots mark the averages $e\bar{r}r^{\mathrm{tr}}$ and $e\bar{r}r^{\mathrm{te}}$ by Eq. 5.10. In the right plots, errors on $\mathbf{S}_{\mathrm{ho}}$ are plotted. Note, for comparison, the RMSE of ~ 0.07 eV of the the baseline 1D descriptor a_{XX}, equally for training, test and hold-out data.

explicitly here, do not unveil marked differences between the strategies for *any* of training, test and hold-out error.

As a conclusion, the results give some evidence for our assumption that strategy $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ (minimizing the average training error) can improve descriptor generalizability, without worsening the performance on the training data. We emphasize that the differences between the models are small here because the used data set is very homogeneous (see Fig. 7.8). Thus, testing the strategies on more diverse data-sets should be an issue in future research, for example on materials differing in space group as in Ref. [12], or with a larger variety of elements. Clearly, however, a disadvantage of the CV-based strategies $\mathscr{S}_{\mathrm{CV}}^{\mathrm{tr}}$ and $\mathscr{S}_{\mathrm{CV}}^{\mathrm{te}}$ is their higher computational effort compared to the simple $\mathscr{S}_{\mathrm{all}}$, roughly scaling with N_{CV}.

8.4 Machine Learning Results on the Group-IV Ternaries T

Based on the previous analyses, we now report a set of optimal descriptors for the two targets a and E_{mix}. Very clearly, the term *optimal* should not be taken too literally in the sense of a final solution to a mathematical equation. As should have become clear at this point, there are plenty of ways to construct candidate features which are always guided by some goal with some degree of arbitrariness. This also holds for model hyperparameters like the descriptor dimension Ω. One would, for example, select a small value of Ω if simplicity is the main goal, at the expense of accuracy (see Sec. 2.3.2). Moreover, when model complexity increases the learning methods yield more and more competing models of virtually the same predictive power - which itself can be defined in different ways, as was shown before. Nevertheless, we will now present some exemplary descriptors: for a, we have selected the 3D descriptor on the very simple complexity level $\mathscr{T}_0$ and the 2D descriptor on the rather complex $\mathscr{T}_2$, for E_{mix} the 4D descriptor on $\mathscr{T}_0$ and the 2D descriptor on $\mathscr{T}_2$. All of these were identified by SISSO using strategy $\mathscr{S}_{\mathrm{all}}$.

The actual components of the descriptors are reported in Tab. 8.8. In the two upper cases in the table, double arrows indicate models which are virtually indistinguishable in performance when exchanging one by the other component. The two right components, respectively, have the advantage of reducing the computational effort to calculate the descriptors for new data because they do not include dimer properties that derive from a different *ab-initio* calculation.

Complexity measures, training and test mean MAE from a sanity check, MAE and maxAE on the hold-out set as well as $\mathrm{MAE}_{\mathrm{arr}}$ on the total data are listed in Tab. 8.9. Errors and distributions of errors are also visualized in Fig. 8.14, and Fig. 8.15 shows the correlation plots for the targets a and E_{mix}, respectively. Before analyzing the individual descriptors, we make two general remarks: (1) From the previous analyses we checked that the models are not markedly overfitted. (2) The observed lower variance in training errors compared to test errors is an expected behavior as the coefficients c_i are fitted to the individual training sets in the SC procedure.

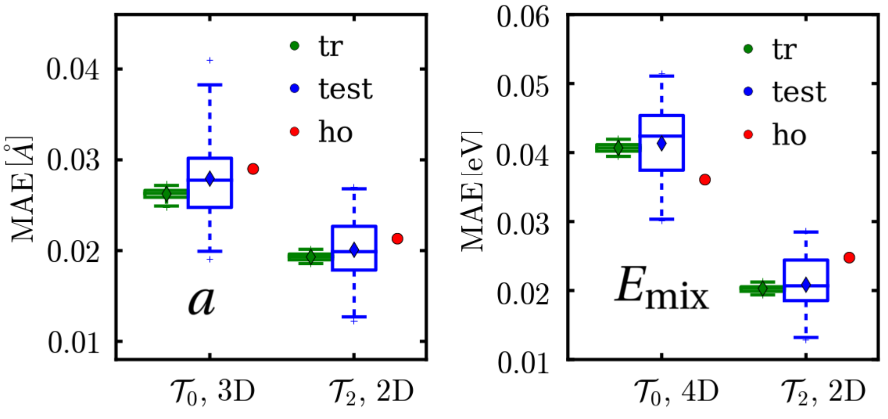


Figure 8.14: Performance of the optimal descriptors for predicting a (left) and E_{mix} (right) of **T** obtained by descriptors of complexity $\mathscr{T}_0$ and $\mathscr{T}_2$. The boxplots (cf. the caption of Fig. 8.13) show the MAE distributions on training sets (green) and on test sets (blue) from a SC ($N_{\mathrm{CV}} = 50$), single dots mark the hold-out errors (red).

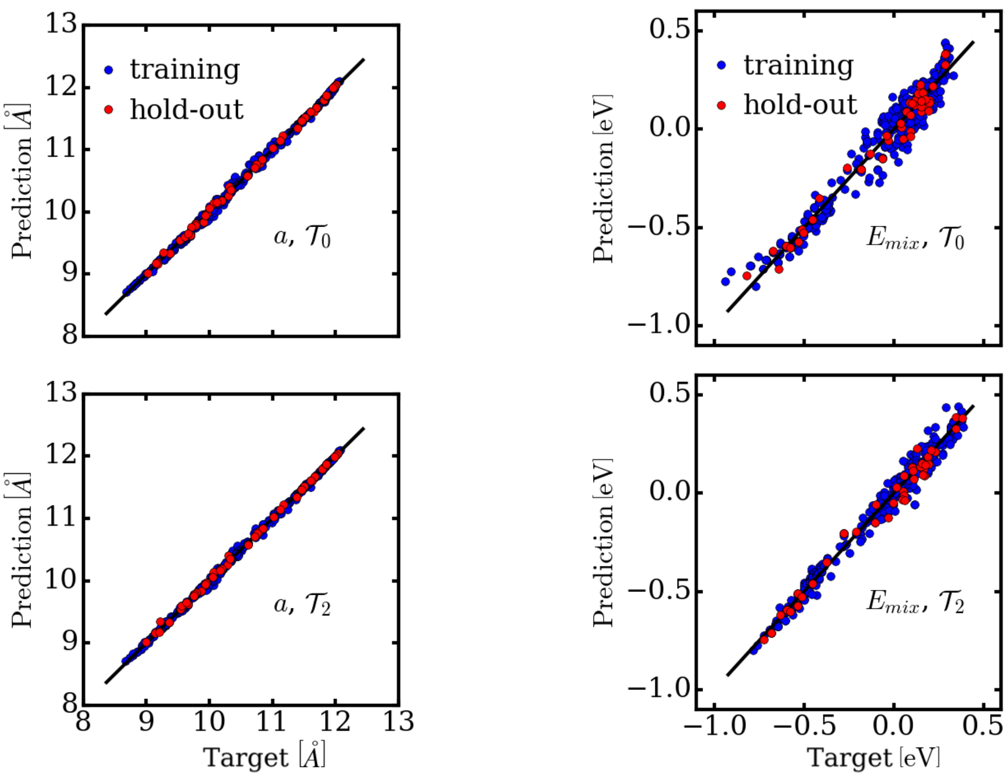


Figure 8.15: Correlation plots (prediction vs. calculation) of the optimal descriptors. Results for learning of a are shown on the left, for E_{mix} on the right. In the two upper plots, feature space complexity $\mathscr{T}_0$ was used, in the two lower ones complexity $\mathscr{T}_2$.

$a, \mathscr{T}_0$		$E_{\mathrm{mix}}, \mathscr{T}_0$	
1	a_{XY}	1	$\bar{Z}^3$
2	$\tilde{d}^2_{XY} \leftrightarrow \tilde{a}^2_{XY}$	2	$\tilde{P}^2$
3	$\tilde{a}^3_{XY}$	3	$\tilde{E}^2_{g,b}$
		4	$\tilde{d}^3_{XY} \leftrightarrow \tilde{a}^3_{XY}$

$a, \mathscr{T}_2$		$E_{\mathrm{mix}}, \mathscr{T}_2$	
1	$\frac{\sqrt{\bar{a}^2_{XX}+\bar{a}^2_{XY}}}{\sqrt[3]{\bar{r}^2_p+\bar{d}^2_{XX}}}$	1	$\frac{1}{E_{f,XX}+\tilde{E}^2_{g,X}} \cdot \frac{1}{\sqrt{E_{f,XX}+E_{g,X}}}$
2	$\frac{\sqrt[3]{\lvert\tilde{r}^2_d-\tilde{d}^3_{XY}\rvert}}{(r_d+\tilde{r}^3_d)^3}$	2	$(E_{g,XX}+E_{g,b})^2 \cdot \sqrt{\lvert\bar{r}^3_s-\bar{d}^3_{XY}\rvert}$

Table 8.8: Components of the optimal descriptors for predicting a and E_{mix} on complexity levels $\mathscr{T}_0$ and $\mathscr{T}_2$. The double arrows indicate components that, when exchanged by each other, are virtually indistinguishable in performance while requiring more (left) or less (right) computational effort to generate the required local primary features.

Results for learning the lattice constant: We start with the 3D descriptor for a on complexity level $\mathscr{T}_0$ where a_{XY} gives essentially the same model as with d_{XY} in the second component, with a negligible rise in error. This is due to the high correlation of a_{XY} and d_{XY}, and is favorable because when being based on the bulk binary lattice constant a_{XY} as a single feature only, informational complexity N_{f_k} and also the computational effort to calculate the descriptor is reduced. The descriptor takes a very elegant form in the style of an expansion by a simple average and the second and third central moments of a_{XY}. This effectively augments a pure interpolation between binaries by some additional terms similar to bowing corrections (see Eq. 7.9). As a_{XY} was made equivalent to an atomic radius it refers to an atomic ball with the mean radius of species A and B. The second and third moment its radius can be interpreted to cover effects of atomic surface and volume (cf. Sec. 5.2.1). Being based on the second and third power of a length, the higher moments can be interpreted to account for effects of atomic surfaces and volumes. MAE_{arr} for this model is lower than the benchmark of arrangement blind Vegard's rule ($\text{MAE}_{\text{arr}} = 0.010\,\text{Å}$). This is because a_{XY} (and its higher moments) include an average over pairs and thus capture the local bonding motifs (cf. Secs. 5.2.1 and 8.1.2). Finally, we note that the three individual components are not redundant as they do not exhibit a high pairwise correlation (maximum $|\rho_{i,j}| = 0.49$).

The second presented descriptor for a, the 2D descriptor on $\mathscr{T}_2$, has much higher informational (N_{f_k}) and algebraic (N_{op}) complexity. It uses all basic properties of length dimension from our pool of local primary features $\{f_k\}$, except from r_s. On the one hand, the increased complexity results in an improved predictive performance, on the other hand, the descriptor seems to lack an easy interpretation. One just might argue that the first component that is based on second moments only, again accounts for atomic surface effects. Note also here that the two components are very little correlated ($|\rho_{1,2}| = 0.14$), i.e. they are not redundant.

As can be seen in Fig. 8.15, both presented descriptors for a give pretty good correlations without any outliers. So, apart from the fact that the descriptor on complexity $\mathscr{T}_2$ is able to drop the errors slightly, it hence seems to be already sufficient to stay with the descriptor on $\mathscr{T}_0$.

Results for learning the energy of mixing: Going over to E_{mix}, in the 4D descriptor on complexity $\mathscr{T}_0$ an exchange of a_{XY} and d_{XY} in the fourth component virtually keeps the predictive performance equal. Choosing here a_{XY}, again, has the benefit that it reduces the computational effort to calculate the descriptor because a_{XY} and $E_{g,b}$ result from the same *ab-initio* calculation. Alternatively replacing the binary gap $E_{g,b}$ by the dimer gap $E_{g,XY}$ such that the total descriptor does not include binary properties markedly worsens the predictive power, by the way. So, it is best to stay with the version without dimer properties. Here, the binary lattice constant and band gap involve an average over pairs and hence enable the descriptor, up to some extent, to capture the effects of atomic arrangements. It additionally includes the very simple properties Z and P. Besides that it interpolates between binaries, it is hard to find an interpretation of the descriptor, in particular to the types and exponents of the higher moments. Also here, there is no high redundancy in components (maximum $|\rho_{i,j}| = 0.38$).

Similar to a, the 2D descriptor on the higher complexity tier $\mathscr{T}_2$ improves predictive performance, rising both in algebraic and informational complexity. As can be taken from Tab. 8.8, the descriptor relies on band gap and formation energies, on the atomic radius r_s, and on the dimer distance d_{XY}. At first glance, one might ask if an alternative descriptor with the first component simplified to $(E_{f,XX} + \tilde{E}_{g,X}^2)^{3/2}$ or $(E_{f,XX} + \tilde{E}_{g,X})^{3/2}$ would have equal predictive power. Unfortunately, this is not the case because the simply averaged $E_{g,X}$ and the second higher moment $\tilde{E}^2_{g,X}$ are not interchangeable (their $|\rho_{i,j}|$ of 0.92 is too low). The argument of the square root in the second component, $\bar{r}_s^3 - \bar{d}^3_{XY}$, probably expresses an effect of volume differences between isolated and bounded atoms due to the third higher moments. Besides this, however, no easy interpretation of the descriptor is eye-catching. Also here, the two components are almost uncorrelated ($|\rho_{1,2}| = 0.13$).

The larger dispersion in the right plots of Fig. 8.15 demonstrates that the learning of E_{mix} is more difficult than of a and requires higher descriptor complexity (this is expectable in the light of Fig. 7.8 where the pattern of a is more regular). Finally, Fig. 8.16 shows that the 2D descriptor for E_{mix} captures the main features of arrangement dependency.

Conclusions on the data set T: As a general remark, descriptors on a high complexity level, as on $\mathscr{T}_2$ here, appear quite technical. Mainly, we see an interpretability of the selected primary features itself and the supercell averaging that generalizes interpolations in the style of Vegard's law. The further algebraic combinations rather seem to provide a good fitting by accounting for non-linearities, instead of uncovering intelligible physical relations for these specific learning tasks.

As an extension of our research on the ternary materials, it would be interesting to find also models for several other physical properties. Data for the *bulk modulus* B_0 already has been calculated at the Murnaghan fits on **T** and we expect that a feature space similar to $\mathbf{X}_{a,2}$ for the lattice constant will yield useful descriptors for B_0 as well. Further hints on how to improve upon this are provided in the review of Ref. [9]. First work also has been done to learn *band gaps* at the high-symmetry point Γ, both in LDA and TB approximation (the latter from additional calculations). The TB band gap is highly correlated to a ($\rho = 0.987$) so the first searches for descriptors were promising. The prediction of LDA band gaps of **T** is challenging due to their more irregular behavior, compared to a and E_{mix}, with a large influence of the atomic arrangement (see Fig. B.1). For future research, we recommend a deeper data analysis that also considers effects like band crossing and, in line with the findings for the LDA band gap of data set **O** (see Sec. 8.5.3), a different material representation.

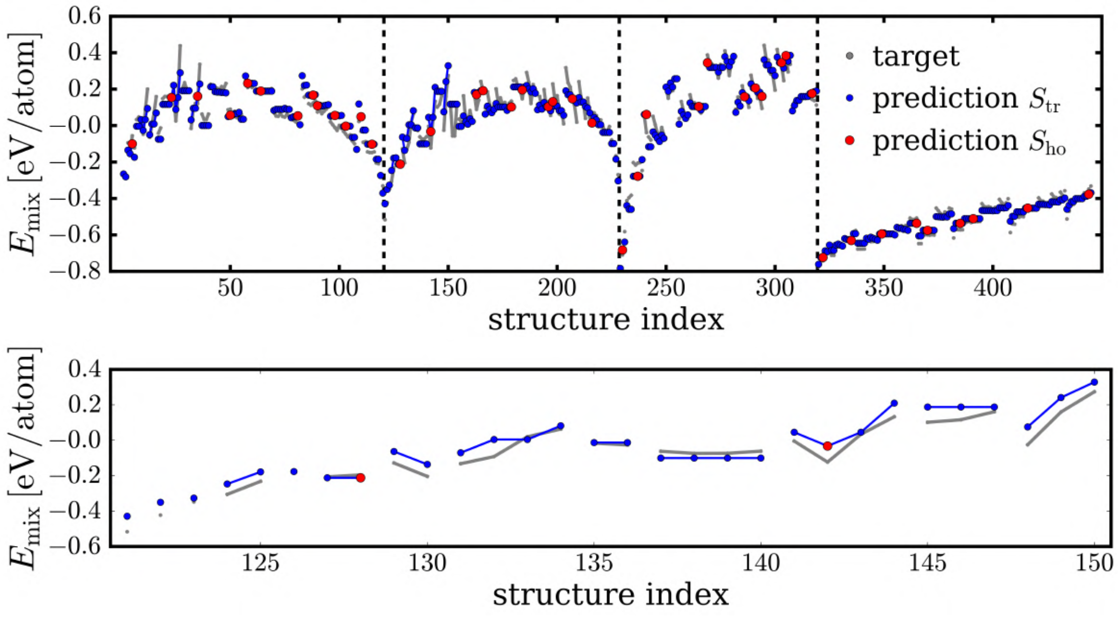


Figure 8.16: Target property E_{mix} of **T** and prediction by the optimal 2D descriptor on complexity $\mathscr{T}_2$ for all structures in the top panel. Samples within the same composition K are connected by grey lines. The lower panel shows some part of the data where the capturing of arrangment effects can be seen more clearly.

target	complexity	Ω	N_{f_k}	N_{op}	training MAE	test MAE	hold-out MAE	hold-out maxAE	$\mathrm{MAE_{arr}}$
a [Å]	$\mathscr{T}_0$	3	1	4	0.026	0.028	0.029	0.093	0.009
	$\mathscr{T}_2$	2	6	18	0.019	0.020	0.021	0.133	0.009
E_{mix} [eV]	$\mathscr{T}_0$	4	4	7	0.051	0.051	0.046	0.130	0.011
	$\mathscr{T}_2$	2	6	15	0.030	0.031	0.035	0.099	0.010

Table 8.9: Performance of the optimal descriptors for predicting a and E_{mix}. Reported are the characteristics dimension Ω, complexity measures N_{f_k} and N_{op}, mean training and test MAE from a SC procedure ($N_{\mathrm{CV}} = 50$), MAE and maxAE on the hold-out set as well as $\mathrm{MAE_{arr}}$ on the total data (except $\mathbf{S}_{\mathrm{ho}}$).

8.5 Machine Learning on the Octet Binary Compounds O

In the following, we extend the work of Refs. [1, 2] on the 82 zincblende binary compounds **O** by finding also models for the equilibrium lattice constant a_{ZB} as well as the TB and LDA band gap energy E_{g}^{Γ} at the Γ-point of these materials [139]. We apply the developed ML methodology from Chap. 5, tailored to this data-set. Learning E_{g}^{Γ} on two different approximation levels allows us to compare differences in relevant primary features and to study a multi-fidelity learning strategy (see Sec. 4.1). All learning tasks on **O** are summarized in Tab. 8.10.

target	primary features	complexity	name of feature space
a_{ZB}	features from Tab. 8.11, only basic data and distances	$\mathcal{T}_0$	$\mathbf{X}_a$
$E_{\mathrm{g}}^{\Gamma,\mathrm{TB}}$	all features from Tab. 8.11	$\mathcal{T}_0$, $\mathcal{T}_1$	$\mathbf{X}_{g,1}$, $\mathbf{X}_{g,2}$
$E_{\mathrm{g}}^{\Gamma,\mathrm{LDA}}$	all features from Tab. 8.11	$\mathcal{T}_0$, $\mathcal{T}_1$	$\mathbf{X}_{g,1}$, $\mathbf{X}_{g,2}$
$E_{\mathrm{g}}^{\Gamma,\mathrm{LDA}}$	VB and CB energies from TB	$\mathcal{T}_1$	$\mathbf{X}_{TB}$

Table 8.10: Table of the machine learning tasks performed on the octet binary compounds **O**.

8.5.1 General Remarks on the Data and the Methodology

Lattice constant a_{ZB} and LDA gap energy $E_{\mathrm{g}}^{\Gamma,\mathrm{LDA}}$ of the materials in **O** were taken from Ref. [1] were their comupation is reported. The calculations of this data set, generated by the code `FHIaims` [67], is also online accessible at [140]. Additionally, TB gaps $E_{\mathrm{g}}^{\Gamma,\mathrm{TB}}$ were calculated by a different code [141], using the relaxed structures from **O**. The materials and the target properties $E_{\mathrm{g}}^{\Gamma,\mathrm{TB}}$ and $E_{\mathrm{g}}^{\Gamma,\mathrm{LDA}}$ are listed in Tables B.14 - B.17 in the appendix. The band gaps are also plotted in Fig. 8.17 in the order of Tables B.14 - B.17, and their correlation is shown in Fig. 8.18. As can bee seen, the two approximations of E_{g}^{Γ} are quite uncorrelated – specifically, $E_{\mathrm{g}}^{\Gamma,\mathrm{TB}}$ covers the range between 0 and 11 eV where LDA predicts a zero band gap.

The used local primary features $\{f_k\}$ are reported in Tab. 8.11, comprising (1) basic atomic data from the periodic table, (2) distances characterizing the isolated atoms and dimers and (3) energies of isolated atoms and dimers. All distances and energies were calculated with the code `FHIaims` and are taken from Ref. [1] where the details on the data is documented. The on-site energies E_{OS}^{s} and E_{OS}^{p} of this data were also used as parameters for the TB calculations. The feature space $\mathbf{X}_{\mathrm{TB}}$ is constructed on valence band maximum and conduction band minimum energies from the same TB calculation to generate $E_{\mathrm{g}}^{\Gamma,\mathrm{TB}}$ that are also reported in Tables B.14 - B.17.

For all the following ML tasks, we identified descriptors up to $\Omega = 5$ by strategy $\mathcal{S}_{\mathrm{all}}$ with the learning method $\mathcal{L}$ =LASSO-LARS+ℓ_0. Feature matrices were standardized and screened for ill-defined entries before the learning (see Sec. 6.3.1). The subset I_{ℓ_0} of preselected features was of size $\tilde{M} = 30$. If descriptor assessment is reported in the following, the stability is checked by means of counts of selection at an extensive CV. Model performance and over- or underfitting are studied by average errors from a sanity check (SC). All CV procedures use the set-up of L10%OCV with $N_{\mathrm{CV}} = 50$. A general hold-out set $\mathbf{S}_{\mathrm{ho}}$ to investigate the generalizability was not applied, by contrast to **T**.

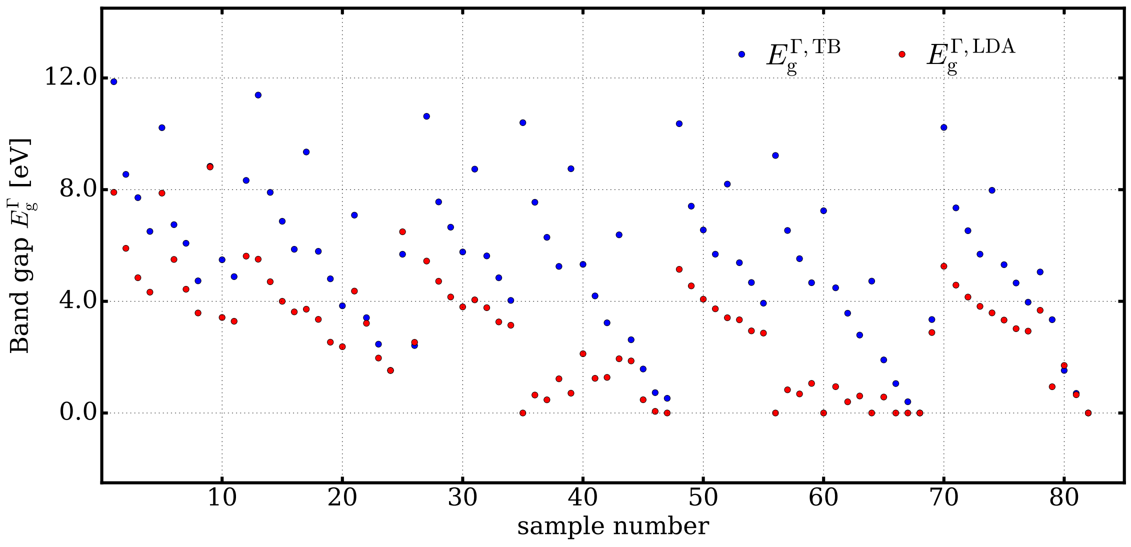


Figure 8.17: Tight binding and LDA band gaps at Γ ($E_g^{\Gamma,\mathrm{TB}}$ and $E_g^{\Gamma,\mathrm{LDA}}$) of the octet binary data set **O** versus sample number (see Tables B.14 - B.17 for the order of samples).

8.5.2 Prediction of Lattice Constants

The feature space $\mathbf{X}_a$ for learning of a_{ZB} considers only the computationally cheap primary features $\{Z, r_s, r_p, r_d, d_{XX}, d_{XY}\}$. $\mathbf{X}_a$ was constructed in a simplified version of the feature engineering scheme of Sec. 5.2: (1) the primary features build tier $\mathcal{T}_0$, (2) $\mathcal{T}_1$ combines all distances by the binary operations $\{(\cdot + \cdot), |\cdot - \cdot|\}$, (3) the unary operations $\{(\cdot)^n, \exp(\cdot), 1/\exp(\cdot), \log(|\cdot|), 1/\log(|\cdot|)\}$, $n \in \{\pm 1/3, \pm 1/2, \pm 2, \pm 3\}$ applied to all features of $\mathcal{T}_0$ and $\mathcal{T}_1$ form $\mathcal{T}_2$. In total, $M = 170$ candidate features were generated that way. Note, that local primary features were included directly, without processing them to global features. Features hence refer to specific atomic sites A or B, and are not symmetric with respect to component switch $A \leftrightarrow B$ but depend on the binaries' chemical naming convention AB (i.e. they do not satisfy requirement 1 from Sec. 4.3).

The obtained descriptors are listed in Tab. D.1 while RMSE, MAE, maxAE from $\mathcal{S}_{\mathrm{all}}$ and SC, and CV counts are presented in Tab. D.2. Their performance in terms of RMSE is shown in Fig. 8.19 on the left and their correlation with a_{ZB} on the right (for $\Omega = 1$ and $\Omega = 5$). Complexity measures N_{f_k} and N_{op} are reported in Tab. D.3.

For increasing Ω, errors decrease as expected and converge to ~ 0.05Å. Correlation improves from $\rho = 0.975$ ($\Omega = 1$) to $\rho = 0.998$. The identified 1D descriptor d_{XY} is intuitive because it is obviously related to a_{ZB}. Training and test errors from SC are close to the fitting errors from $\mathcal{S}_{\mathrm{all}}$ with a low variance and thus do not indicate overfitting. Model stability in terms of counts decreases with Ω due to the rise in number of possible models. It is noteworthy that the found 3D descriptor from $\mathcal{S}_{\mathrm{all}}$ is not selected in any of the CV iterations. Instead, a very similar expression is found frequently which has the second component $\sqrt[3]{r_s(A) + r_s(B)}$ replaced by $\sqrt{r_s(A) + r_s(B)}$.

Either way, both of these 3D descriptors reach maximum informational complexity $N_{f_k} = 5$ (within all Ω) and have an elegant algebraic form that is symmetric with respect to species exchange $A \leftrightarrow B$. The second component contains the sum $r_s(A) + r_s(B)$ that is intuitively related to the

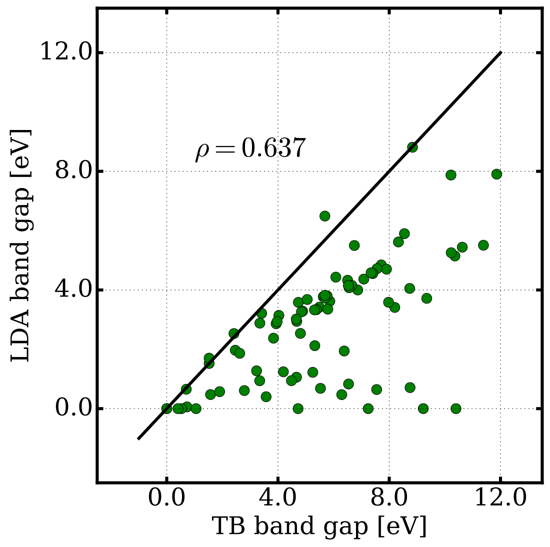


Figure 8.18: Correlation plot of $E_{\mathrm{g}}^{\Gamma,\mathrm{TB}}$ and $E_{\mathrm{g}}^{\Gamma,\mathrm{LDA}}$ of the octet binaries **O** including their Pearson correlation ρ.

lattice constant in a model of touching atomic spheres. The third component $1/(Z(A)+Z(B))$ can be understood as the reciprocal value of the materials' total mass. 4D and 5D descriptors loose the symmetry of species exchange and become less interpretable as also indicated by a higher N_{op}. Note, finally, that none of the descriptors consist of highly correlated components.

8.5.3 Prediction of Band Gaps

Predicting TB Band Gaps: Learning of the TB gap $E_{\mathrm{g}}^{\Gamma,\mathrm{TB}}$ was done using all primary features listed in Tab. 8.11. The feature spaces $\mathbf{X}_{\mathrm{g},1/2}$ were constructed by the scheme of Sec. 5.2. Here, higher moments were considered up to $q = 3$. As a modification, also nearest-neighbor *products* (Eq. 5.5) were included at Step 1. $\mathbf{X}_{\mathrm{g},1}$ includes features of complexity tier $\mathscr{T}_0$, $\mathbf{X}_{\mathrm{g},2}$ up to $\mathscr{T}_1$. The processing from local to global primary features was considered here. Even for the simple materials in **O** this is beneficial to predict a global property like the band gap (see Sec.4.3). For their only two-atomic unit cell, the averaging by Eqs. 5.1, 5.2, 5.3 and 5.4 reduces to the contribution of the single pair i, j. For the heteroatomic dimer properties (d_{XY}, E^{g}_{XY}, E^{b}_{XY}) all central moments obviously are equal to zero.

The found descriptors of $\mathbf{X}_{\mathrm{g},1}$ and $\mathbf{X}_{\mathrm{g},2}$ are listed in Tables D.4 and D.5. MAE from $\mathscr{S}_{\mathrm{all}}$ and MAE distribution from a SC are presented in Fig. D.1 and Fig. D.2 on the left, correlation of the 1D and 5D descriptor with $E_{\mathrm{gap}}^{\Gamma,TB}$ on the right. Per construction, the descriptors from the smaller $\mathbf{X}_{\mathrm{g},1}$ have simple expressions. The atomic on-site energy E_s^{OS} is present for $\Omega = 1,2,3$ and 5, which is interesting as it is an essential parameter of the TB calculation. Vice versa, features of the type distance are not considered at all. A closer look on the 5D descriptor that (as expected) has the best performance unveils further important primary features. As a possible interpretation, we connect the ionization energy IP to the gap by Eq. 3.22. Also the the dimer binding energy E^{0}_{XY} might be related to the gap which would be in line with Ref. [142] that propose a linear relation between

Feature f_k	Description	Feature f_k	Description
Basic properties from the periodic table		Atomic / dimer energies (DFT)	
Z	atomic number^a	IP	atomic ionization potential
g	chemical group $(1,\ldots,8)$	EA	atomic electron affinity
P	chemical period $(2,\ldots,5)$	H, L	atomic Kohn-Sham levels
Atomic / dimer distances (DFT)		$E^s_{\mathrm{OS}}, E^p_{\mathrm{OS}}$	atomic on-site energies
r_s	s-orbital radius^a	$E^{\mathrm{g}}_{XX}, E^{\mathrm{g}}_{XY}$	HOMO-LUMO gap of dimers
r_p	p-orbital radius^a	E^0_{XX}, E^0_{XY}	binding energies of dimers
r_d	d-orbital radius^a		
d_{XX}	bond length homoatomic dimer^{a,b}		
d_{XY}	bond length heteroatomic dimer^{a,b}		

Table 8.11: The local primary features $\{f_k\}$ used for the machine learning tasks on **O** (except for the feature space $\mathbf{X}_{\mathrm{TB}}$). Superscript a: prediction of a_{ZB} uses these features only. Superscript b: dimer distances $d_{XX/XY}$ were not transformed to radius equivalents here.

single bond energy and the gap. Furthermore, we find it interesting to see the very cheap properties P and g. Overall, the model performance reaches from 1.11 eV at $\Omega = 1$ to 0.41 eV at $\Omega = 5$ in terms of MAE from $\mathscr{S}^{\mathrm{all}}$, and the correlation to the target from $\rho = 0.836$ to 0.983 (see Tab. 8.12).

Also for $\mathbf{X}_{\mathrm{g},2}$, IP plays an important role. For instance, the 1D descriptor consists of IP only. At higher Ω, also the energies E_s^{OS}, E^0_{XY} and the Kohn-Sham level H contribute whereas similarly none of the distances is selected. For $\Omega > 1$, the expression $\left(\langle E_s^{OS,2}\rangle + E^{0,2}_{XY}\right)^{-3}$ is always present, although it is hard to find an interpretation to it. Additionally, $\langle \Pi P\rangle + \langle \Pi g\rangle$ plays an important role which is noteworthy as it does not require any computational cost to determine the features P and g from the periodic table. The predictive power ranges from 0.70 eV ($\Omega = 1$) to 0.24 eV ($\Omega = 5$) in terms of MAE ($\mathscr{S}^{\mathrm{all}}$, see Fig. D.2 on the left) and from $\rho = 0.944$ to 0.994 (see the correlation in Fig. D.2 on the right, and Tab. 8.12).

Predicting LDA Band Gaps from Basic Atomic and Dimer Properties: By contrast to $E^{\Gamma,TB}_{\mathrm{gap}}$, the calculation of the LDA gap $E^{\Gamma,LDA}_{\mathrm{gap}}$ is much more time-consuming, and hence identifying effective surrogate machine-learning models is more desirable here. First, we worked analogously to $E^{\Gamma,TB}_{\mathrm{gap}}$ with the same feature spaces $\mathbf{X}_{\mathrm{g},1}$ and $\mathbf{X}_{\mathrm{g},2}$ that consider only atomic and dimer properties.

For $\mathrm{X}_{\mathrm{g},1}$ we reach MAEs from 1.32 eV at $\Omega = 1$ to 0.52 eV at $\Omega = 5$ and correlations with the target from $\rho = 0.657$ to 0.942 (see Fig. D.3). The expressions of the descriptor components are listed in Tab. D.6. Here, the dominance of energies, in particular again of the dimer binding energy E^0_{XY} is noteworthy [142]. The 2D descriptor does not excel in regard to model performance but we find its very simple expression interesting. Furthermore, pair products appear frequently, and the feature $\bar{\Delta} d^3_{XX}$ (present at $\Omega = 4$ and $\Omega = 5$) might correspond to a difference in atomic volumes as d_{XX} is strongly related to the atomic radius.

Working with the larger feature matrix $\mathbf{X}_{\mathrm{g},2}$, MAEs between 1.16 eV ($\Omega = 1$) and 0.41 eV ($\Omega = 5$) were achieved where the highest gain by $\sim 50\%$ is reached from $\Omega = 1$ to 2 (see Fig. D.4 on the left).

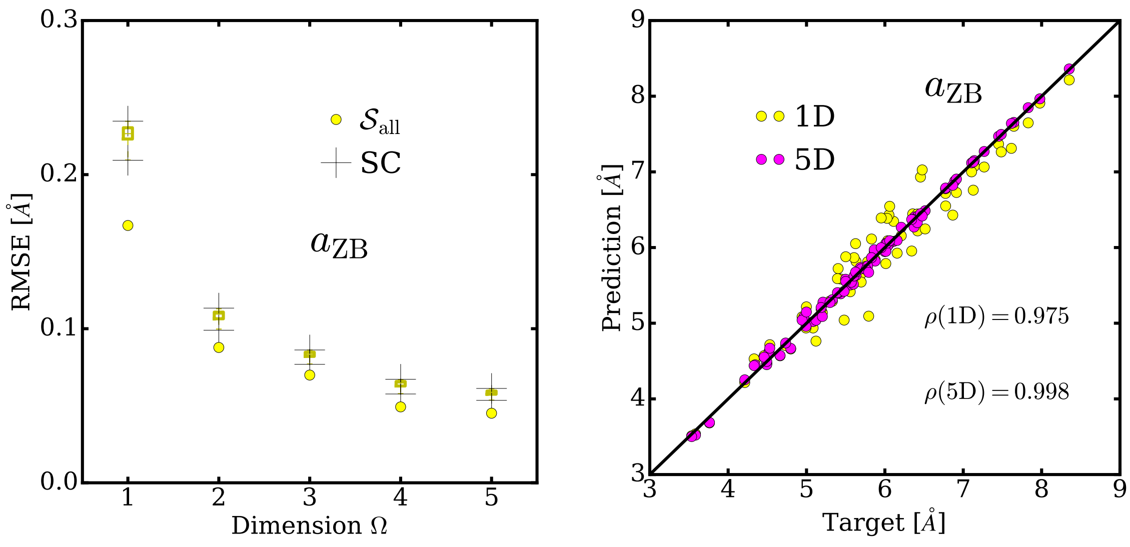


Figure 8.19: Performance of the descriptors for a_{ZB} of the data set **O**. Left: RMSE versus descriptor dimension Ω. Circles mark the training RMSE from $\mathscr{S}_{\text{all}}$, the boxplot (see the caption of Fig. 8.13) shows the distribution of RMSE from a SC (which obviously is very narrow). Right: correlation plot (prediction vs. target) for the 1D and the 5D descriptor including the Pearson correlation with the target.

Correlation ranges from $\rho = 0.719$ to $\rho = 0.964$ (Fig. D.4 on the right), which expectedly improves over $\mathbf{X}_{\text{g},1}$. The algebraic form of the descriptor components in Tab. D.7 appears complex and hard to interpret. Interestingly, the simple feature $1/\sqrt{|\Pi Z - \Pi P|}$ seems to be responsible for the large drop in errors, and the components at $\Omega = 2$ are almost always present for all higher Ω. Notably, nearest-neighbor products are present in many expressions. With respect to the selected primary features, we observe that - except from Z and P - these comprise only energies, presumably more closely related to the gap energy than distances.

Predicting LDA Gaps from TB Data: An alternative approach was to learn $E_{\text{gap}}^{\Gamma,\text{LDA}}$ solely from tight-binding data as primary features. This way we investigate a *multi-fidelity* learning approach from the easy-obtainable TB band structure to the computationally more elaborate LDA band gap. The feature space $\mathbf{X}_{\text{TB}}$ considers the primary features $\{E_{\text{VBM}}, E_{\text{CBM}}, E_{\text{VB}}^{Z}, E_{\text{CB}}^{Z}\}$, comprising the global valence band maximum E_{VBM} and conduction band minimum E_{CBM} as well as the energy of the highest valence band and the lowest conduction band at the high-symmetry points $Z = L, \Gamma, X$ and the points U/K, belonging to the same point group [143]. These global primary features do not require being averaged over supercells. Instead of considering the higher moments as above, we simply raise the primary features by the powers of $q = 2$ and $q = 3$. The further augmentation applies the operations of Scheme 5.2 up to tier $\mathscr{T}_1$. We stress that $\mathbf{X}_{\text{TB}}$ includes expressions of the type $|E_{\text{VB}}^{Y} - E_{\text{CB}}^{Z}|$, i.e. direct and indirect band gaps.

The performance of the descriptors for $\Omega \leq 5$ in terms of MAE and target-prediction correlation is shown in Fig. D.5, the components of the descriptors are listed in Tab. D.8 in the appendix. By construction, the algebraic expressions are simple, as for instance the frequently selected feature $E_{\text{CB}}^{L} + E_{\text{CB}}^{X}$. Notably, the combination $|E_{\text{VB}}^{\Gamma} - E_{\text{CB}}^{\Gamma}|$, i.e. the direct TB gap at Γ $E_{\text{g}}^{\Gamma,\text{TB}}$, is not considered.

Model performance ranges from 0.75 eV at $\Omega = 1$ to 0.49 eV at $\Omega = 5$ (MAE from $\mathscr{S}_{\text{all}}$), correlation from $\rho = 0.858$ to $\rho = 0.950$. By contrast to the learning by atomic and pair features, we observe a high dispersion of the errors.

Conclusions on Learning TB and LDA gaps: A summary of the achieved performance of the learning tasks for the gaps $E_{\text{gap}}^{\Gamma,TB}$ and $E_{\text{gap}}^{\Gamma,LDA}$ is presented in Tab. 8.12, listing MAEs from $\mathscr{S}^{\text{all}}$ for $\Omega = 1,\ldots,5$ and the correlations with the target ρ for $\Omega = 1$ and $\Omega = 5$. From this, we draw the following conclusions:

- For both types of gaps, the comparison of $\mathbf{X}_{g,1}$ and $\mathbf{X}_{g,2}$ demonstrates that model performance is improved by an increase in feature complexity, with the cost of a lower interpretability.
- The frequently selected nearest-neighbor products by Eq. 5.5 proved to be a helpful ingredient at augmenting the feature space for this learning tasks.
- With equal feature spaces $\mathbf{X}_{g,1}$ or $\mathbf{X}_{g,2}$, better predictions are reached for $E_{\text{gap}}^{\Gamma,\text{TB}}$ compared to $E_{\text{gap}}^{\Gamma,\text{LDA}}$. This is remarkable, as both TB and LDA related primary features, such like on-site energies or dimer gaps, respectively, are present in $\mathbf{X}_{g,1}$ and $\mathbf{X}_{g,2}$. It might result from the conceptually simpler level of approximation that TB does, i.e. indicate that TB gaps per se are easier to learn. We further note, that we do not observe a close similarity of the actual features contributing to the descriptors for $E_{\text{gap}}^{\Gamma,\text{TB}}$ and $E_{\text{gap}}^{\Gamma,\text{LDA}}$ which is expectable considering the low correlation between the two targets (see Fig. 8.18).
- Comparing the learning of $E_{\text{gap}}^{\Gamma,\text{LDA}}$ from pair and dimer data ($\mathbf{X}_{g,2}$) and from TB data ($\mathbf{X}_{TB}$, both on complexity $\mathscr{T}_1$) does not yield a clear preference of one of the feature sets here: whereas TB data yields better performance for $\Omega = 1$, it is slightly worse for larger Ω. We relate the unfavorable high dispersion in the MAE distribution for $\mathbf{X}_{TB}$ to the lower number of considered local primary features and the smaller size of the feature space (3959 vs. 49121 for $\mathbf{X}_{g,2}$).
- Additional CV analyses (not shown) indicate that increasing the descriptor dimension Ω beyond 5 leads to a decrease in model stability and to the onset of overfitting. This is in line with the limit by Eq. 2.15 that does not make larger Ω reasonable, assuming $C = 4$ and a feature space size of $M \sim 10000$.
- Overall, the best achieved accuracy is 0.24 eV for $E_{\text{gap}}^{\Gamma,\text{TB}}$ and 0.41 eV for $E_{\text{gap}}^{\Gamma,\text{LDA}}$ in terms of MAE from $\mathscr{S}^{\text{all}}$. It would be desirable to reduce that error, in particular for small-band-gap materials ($E_{\text{gap}}^{\Gamma} \sim 1\,\text{eV}$). A possible way to achieve this is the inclusion of primary features like polarizabilities or using a representation that considers characteristics of the local electronic density of states, such as in the c/BOP approach of Ref. [12], which we leave as an outlook.

		$E_g^{\Gamma,\mathrm{TB}}$ [eV]		$E_g^{\Gamma,\mathrm{LDA}}$ [eV]		
	Ω	$\mathbf{X}_{g,1}$	$\mathbf{X}_{g,2}$	$\mathbf{X}_{g,1}$	$\mathbf{X}_{g,2}$	$\mathbf{X}_{TB}$
MAE	1	1.11	0.70	1.32	1.16	0.75
	2	0.64	0.50	0.96	0.58	0.63
	3	0.56	0.33	0.76	0.49	0.56
	4	0.45	0.27	0.64	0.44	0.54
	5	0.41	0.24	0.52	0.41	0.49
ρ	1	0.836	0.944	0.657	0.719	0.858
	5	0.983	0.994	0.942	0.964	0.950

Table 8.12: Summary of predictive performance for learning band gaps of **O** by the different learning tasks. Reported are the MAE of descriptors from $\mathscr{S}_{\mathrm{all}}$ ($\Omega \leq 5$) as well as the resulting target-prediction Pearson correlation ($\Omega = 1$ and 5).

Chapter 9

Conclusions and Outlook

9.1 Conclusions

In this thesis, we developed machine learning methods for material science problems, implemented them in a Python program and found predictive descriptors for material properties of group IV ternary and octet binary compounds. The roadmap towards these models followed the steps of data production and processing, feature engineering, the actual machine learning, and model assessment.

First, we generated a well-analyzed *ab-initio* data-set of the group IV ternary materials and the required data for the primary features. We then proposed a feature engineering scheme based on symbolic regression that averages simple local primary features over the supercell and further combines them algebraically. It hence is able to represent crystal structures with complex unit cells where physical properties are affected both by composition and atomic arrangement.

We showed that LASSO-LARS achieves a feature selection that is independent from the column order and could demonstrate that SISSO reduces the redundancy in descriptor components. Our defined cross-validation based selection strategy choosing the model by mean training error could improve the generalizability of the descriptor in a first application. For assessing the descriptor formulas quantitatively, we proposed measures for the algebraic and informational complexity. A newly-defined error metric is able to quantify the degree of the capturing of arrangement effects.

Data parsing, analysis and the actual learning was computationally solved in Python. The core code provides a flexible and extensive statistical output and handles huge candidate feature matrices efficiently. It is partially parallelized, in particular in its SISSO module.

At applying the methodology to the group-IV zincblende compounds, tetrahedron primary features and next-neighbor differences in atomic features were found to be omittable at feature engineering. We identified descriptors that effectively predict the equilibrium lattice constant and the energy of mixing by a fractional cost of *ab-initio* calculations. With these, we achieved an accuracy of MAE= 0.02 Å and MAE=0.02 eV, respectively, and discussed the meaning of their analytical formulas. The zincblende lattice constant and the TB band-gap of octet binary materials could be learned by descriptors that reach a precision of MAE=0.07 Å and MAE=0.24 eV.

Finally, we add three general conclusions: (1) A proper understanding of uncertainties of the target is essential to estimate the required accuracy in the primary features and the reachable model performance. (2) The symbolic regression inspired descriptor approach worked best for geometric and rather simple learning tasks like lattice constants but got to its limits at more complex target properties with many underlying physical mechanisms such as band gaps. Here, the desired identification of simple formulas might be too optimistic [9] and other less coarse material representations are better suited. (3) Physical domain knowledge is an essential prerequisite to develop *meaningful* models. Without that, machine learning barely can go beyond uncovering correlative dependencies or reaching objectives in numerical performance.

9.2 Outlook

Due its high homogeneity, the data set **T**, in particular the target a, puts limitations on analyzing the CV based selection strategies. Hence, it is necessary to apply these to materials of higher variability, for example the binary alloys of Ref. [144] or the TCOs of Ref. [12]. Here, it also is interesting to transfer our feature engineering approach to those structures and to compare it with the ones used in the mentioned works.

We see two open directions to improve the capturing of arrangement effects of the ternaries: (1) The occupation vector σ could be simply processed semantically, for instance by counting differences between neighboring digits or calculating the average lengths of clusters of equal digits. This is generalizable to the three-dimensional unit cell and will be similar to the *n-gram* representation of Ref. [12]. (2) Local geometric information on the bond length d_b could be included to some of the primary features f_k, i.e. they could be transformed to $f_k(d_b)$. The functional form of $f_k(d_b)$ can be derived from calculations at different d_b, as already done e.g. for the total energy of the tetrahedral clusters $E_{0,t}$ in Fig. B.6. The dimer distances d_{XY} at a certain bond could be used to approximate its length at which $f_k(d_b)$ will be evaluated. Alternatively, machine-learned bond-lengths may serve as estimates for d_b.

Regarding the search through the space of algebraic combinations, efficiency could be improved by basing this on genetic programming [14]. That way, the costly explicit calculation of the total data matrices **X** and the step of dimensionality reduction could be avoided.

Finally, it would be interesting to modify the minimization problem in three directions: (1) One of our complexity measures could be added to the regularization term such that a high complexity will be penalized, directly favoring compact descriptors [7, 45]. (2) Similarly, a regularization by MAE_{arr} might improve the models' arrangement capturing. (3) As already done in the implementation of SISSO, our CV selection strategies could be formulated also for other dimensionality reduction methods as multi-task learning problems.

Appendix Part IV

Appendix A

Parametrization and Numerical Tests of the Machine Learning Methods

A.1 Convergence Test for Descriptor Dimensionality Ω

The model performance with respect to descriptor dimension Ω was studied for learning both target properties a and E_{mix} of the data set $\mathbf{T}$. For this, we used feature spaces comprising atomic and pair primary features on complexity level $\mathscr{T}_1$ with matrix sizes $\|\mathbf{X}_a\| = 5257$ and $\|\mathbf{X}_E\| = 8736$. Due to the combinatorial explosion of the NP-hard ℓ_0-problem at increasing Ω (cf. Sec. 2.2.3), the descriptors were not obtained by $\mathscr{L}$ =LASSO+ℓ_0. Instead, we used here LASSO as a single learning method and directly take its Ω-dimensional linear combinations up to $\Omega_{\text{max}} = 30$ as they appear at decreasing hyperparameter λ, although this is especially prone to yield highly correlated features in the descriptor components [2, 145]. Cross-validation was performed with $N_{CV} = 50$, and from this average test and training errors were calculated. Here, $\overline{err}(\Omega)$ averages over all descriptors $\{\mathscr{D}^{(i)}\}_{i=1}^{N_{\text{CV}}}$ with that dimension Ω, which generally do not coincide between the CV iterations. At some increments of λ, LASSO might select more than one variable. For instance, it could directly "jump" from an Ω-combination to an $\Omega+2$-combination such that no model of dimension $\Omega+1$ is available for that CV iteration. The average errors for the skipped $\Omega+1$ then consider then less than N_{CV} runs. For high dimensions this can happen very frequently. We do not investigate the thus poorly sampled averages in that cases.

The results of this convergence test are presented in Fig. A.1 for a on the left and E_{mix} on the right, showing both individual and averaged training and test errors (RMSE) as functions of Ω. For a, errors decrease rapidly up to $\Omega \sim 10$, for higher Ω performance improves only little. The average test RMSE is rising for $\Omega \gtrsim 22$, indicating the onset of overfitting. But even before, from $\Omega \sim 10$ the high dispersion of individual test errors marks a larger instability in model selection (see Sec. 2.4.1). Also for E_{mix}, both training and test errors decrease quickly upto $\Omega \sim 10$ and seem to be converged at $\Omega \sim 20$. From $\Omega \gtrsim 27$, the overfitting regime starts. By contrast to a, the generally low scattering of test errors indicate a high stability of the descriptors, even at large Ω. Although LASSO was used here without the ℓ_0-step and feature spaces were on low complexity $\mathscr{T}_1$, we conclude that at the maximum descriptor dimension of $\Omega_{\text{max}} = 5$ considered in this work the models from LASSO+ℓ_0 generally will not be overfit. The performance could improved by a rise in Ω but for the sake of model interpretability and due to the drop in stability, going beyond $\Omega_{\text{max}} = 5$ is not reasonable.

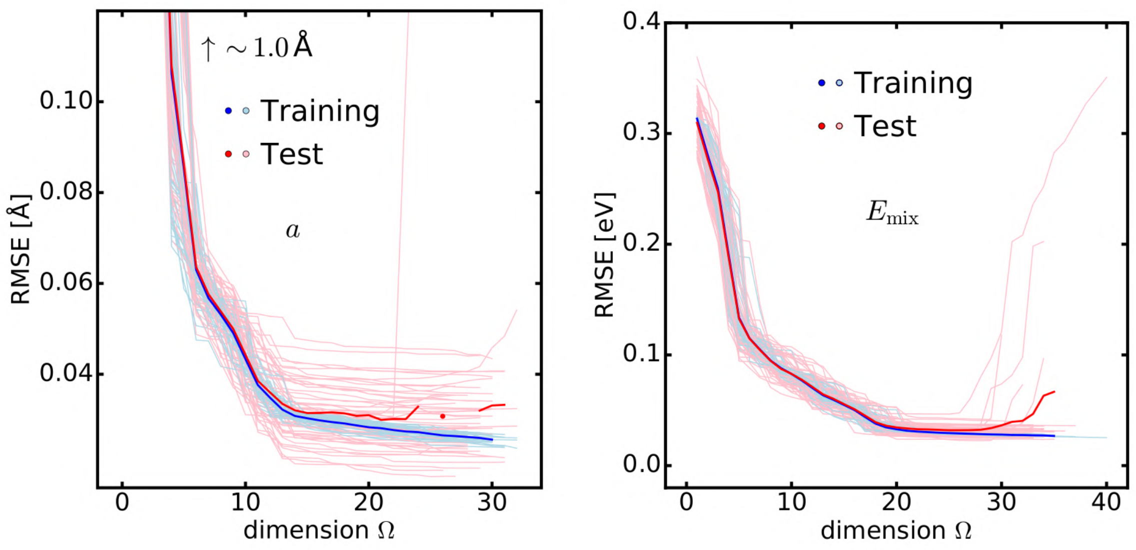


Figure A.1: Convergence of test and training RMSE with respect to descriptor dimension Ω. Shown are individual (light colors) and averaged (dark colors) training and test RMSE from L10%O-CV for learning a (left) and E_{mix} (right) of **T** by LASSO *without* a subsequent ℓ_0 search. In both cases feature spaces with atomic and pair primary features on complexity $\mathscr{T}_1$ are used.

A.2 Tests for Cross-Validation

A.2.1 Convergence Test for the Number of CV Iterations N_{CV}

Here, we analyze the convergence of mean training and test errors from cross-validation with respect to the number of iterations N_{CV}. For the tests, 500 iterations of L10%O-CV were performed at fixed models from strategy $\mathscr{S}^{\text{all}}$, i.e. as SC, and test and training errors in the metrics RMSE, MAE and maxAE were considered. The convergence was studied on *moving averages*

$$\overline{err}(N_{\text{CV}}) \equiv \frac{1}{N_{\text{CV}}} \sum_{i=1}^{N_{\text{CV}}} err(i) \tag{A.1}$$

that average the errors $err(i)$ at the individual iterations i up to N_{CV}. These were analyzed for predicting a and E_{mix} of the data-set **T** with the feature spaces $\mathbf{X}_{a,2}$ and $\mathbf{X}_{E,2}$ on complexity $\mathscr{T}_0$.

Figure A.2 shows the convergence of test RMSE for descriptors of dimension $\Omega = 1$ and $\Omega = 5$ for E_{mix} as an example. Apparently, in both cases $\overline{err}^{\text{te}}_{\text{RMSE}}$ fluctuates markedly until $N_{\text{CV}} \approx 15$ and reaches convergence at $N_{\text{CV}} \approx 200$. In all learning tasks of this work, $N_{\text{CV}} = 50$ was applied as a feasible choice. We estimate the level of convergence at this value by comparison with the reference value at $N_{CV} = 500$, i.e. the difference of the two horizontal lines in the figures. That way, all upper limits in error accuracy listed in Tab. A.1 were derived considering the respective maximum within $\Omega = 1, \ldots, 5$. Importantly, mean RMSE and mean MAE of both training and test data lie below the irreducible error in the targets of ~ 0.03 Å for a and ~ 68 meV for E_{mix} (see Tab. 7.3). An increase of N_{CV} beyond 50 hence is not profitable.

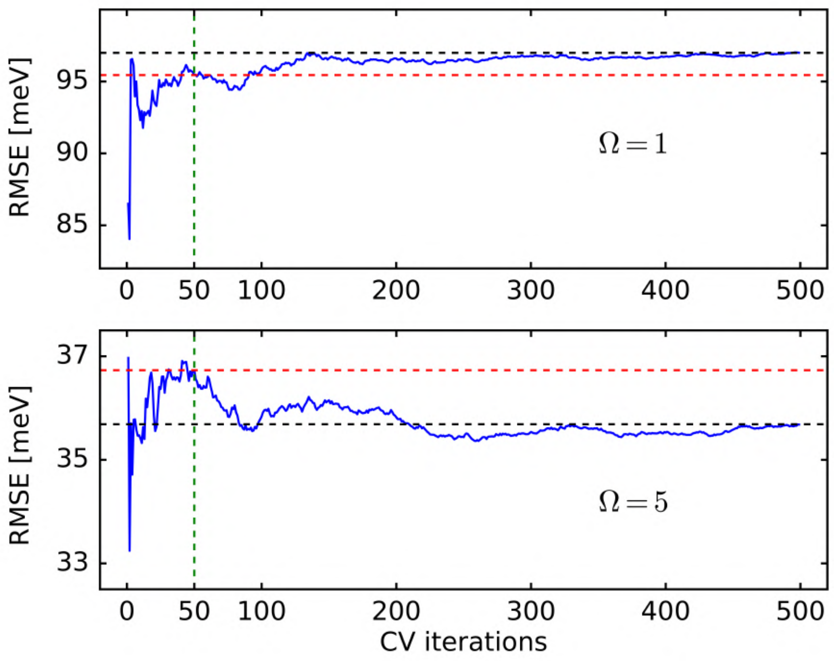


Figure A.2: Moving average of test RMSE $\overline{err}^{\mathrm{te}}_{\mathrm{RMSE}}(N_{\mathrm{CV}})$ versus cross-validation iterations for descriptors of dimension $\Omega = 1$ (top) and $\Omega = 5$ (bottom). The CV was done as L10%O-CV at fixed descriptor components (SC). The considered learning task is E_{mix} of the data-set **T** with feature space $\mathbf{X}_{E,2}$ on complexity $\mathscr{T}_0$. The green vertical lines mark $N_{\mathrm{CV}} = 50$ that was used in this work, the red horizontal lines indicate the corresponding value of $\overline{err}^{\mathrm{te}}_{\mathrm{RMSE}}$, the black horizontal lines the converged value at $N_{\mathrm{CV}} = 500$.

target	type	RMSE	MAE	maxAE
a	training	0.1 mÅ	0.1 mÅ	0.3 mÅ
	test	1.2 mÅ	0.5 mÅ	4.4 mÅ
E_{mix}	training	0.2 meV	0.2 meV	3.9 meV
	test	1.7 meV	1.8 meV	6.1 meV

Table A.1: Convergence levels of training and test errors from L10%O-CV (SC) for predicting a and E_{mix} of the data-set **T**. Reported are the absolute differences at $N_{\mathrm{CV}} = 50$ with respect to the reference at $N_{\mathrm{CV}} = 500$ for training and test $\overline{err}_{\mathrm{RMSE}}$, $\overline{err}_{\mathrm{MAE}}$ and $\overline{err}_{\mathrm{maxAE}}$. For comparison, note the uncertainties of 0.03 Å in a and 68 meV in E_{mix}.

A.2.2 Influence of Split Size p

Besides N_{CV}, also the influence of the CV split size p on training and test errors was studied. The tests were done with the non-exhaustive CV scheme applied in this work with $N_{CV} = 50$ iterations at fixed descriptors, i.e. as SC. The number of data points in the test sets was varied between 1 and 2 (LOOCV and L2OCV) as well as $p = 5\%$, 10%, 25% and 50%. As in the previous, we consider the learning of E_{mix} of data-set **T** with the feature space $\mathbf{X}_{E,2}$ on complexity $\mathscr{T}_1$. The distribution of the resulting test and training errors is shown in the boxplots of Fig. A.3.

We observe that training errors have a low variance for all split sizes p, and that their mean at a fixed Ω is not affected by the various p. By contrast, test errors have a large variance for small test splits that decreases with increasing split size. Also their mean is affected by p at fixed Ω with the largest difference between LOOCV and L2OCV. It is additionally noteworthy that at LOOCV mean test errors lie below the mean training errors, i.e. $\overline{\text{err}}^{\text{te}} < \overline{\text{err}}^{\text{tr}}$.

The independence of training errors from split size p is a consequence of the high homogeneity of the data set $\mathbf{T}$: if the data-points are very similar even a skip of half of the data at p=50% will yield very similar models. The decrease of test error variance as a function of p, or equivalently at decreasing training split size $1-p$, is a usual behaviour for this CV scheme [50]. An argument why mean test errors may lie below training errors at $p = 1$ (and rise with p) is given in Sec. A.2.3 below. We conclude from this analysis that the generally recommended and in this work applied value of $p = 10\%$ is a proper choice. It avoids the high variance in test errors and the suspicious relation $\overline{\text{err}}^{\text{te}} < \overline{\text{err}}^{\text{tr}}$ at the smaller p. The choices $p = 25\%$ and $p = 50\%$ could work as alternatives on this homogeneous data-set but are unusual.

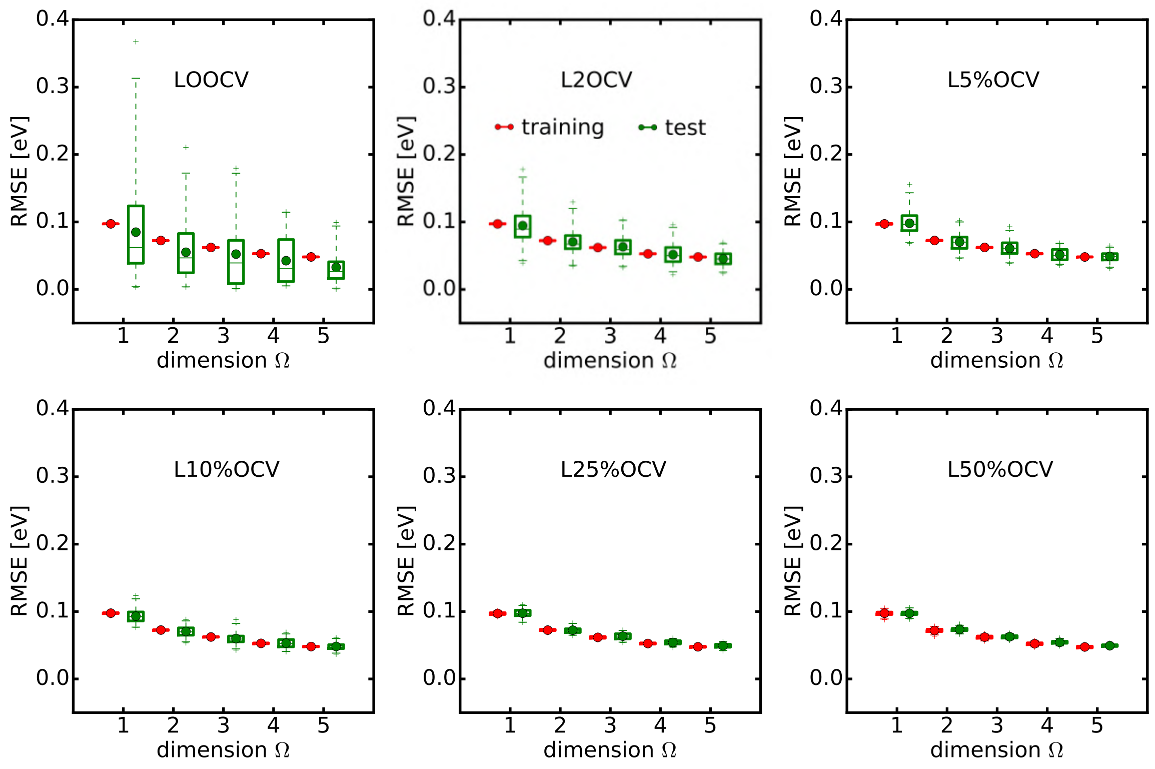


Figure A.3: Box plots (see the caption of Fig. 8.13) of training (red) and test (green) RMSE from LPOCV as SC, at various split size p. Mean errors are indicated by filled circles. Underlying learning task: prediction of E_{mix} with feature space $\mathbf{X}_{E,2}$ on complexity $\mathscr{T}_1$.

A.2.3 Considerations on Error Metrics and CV

As addressed, mean test errors are observed to lie below mean training errors, i.e. $\overline{\text{err}}^{\text{te}} < \overline{\text{err}}^{\text{tr}}$, for $p = 1$ in Fig. A.3. This conflicts with the expectation that a model should perform better on the training data it is fitted to, than on the test data (see Sec. 2.4.1). One might assign this to the small number of iterations $N_{\mathrm{CV}} = 50$ in this non-exhaustive CV scheme. The random one-point test sets together comprise only 50 of the 402 data-points and hence an accidental unrepresentative sampling is likely. The findings in Fig. A.4 however contradict with this assumption. Here, the non-exhaustive LOOCV run from Fig. A.3 is compared to a second, independent non-exhaustive LOOCV run, and an exhaustive LOOCV ($N_{CV} = 402$). Average test errors indeed are affected by the different random partitions as the green curves do not coincide between the two partial runs. As however $\overline{\mathrm{err}}^{\mathrm{te}} < \overline{\mathrm{err}}^{\mathrm{tr}}$ is observed in all three runs, this relation cannot follow from an adverse split drawing.

Instead, the relation $\overline{err}^{\mathrm{te}}_{\mathrm{RMSE}} < \overline{err}^{\mathrm{tr}}_{\mathrm{RMSE}}$, or more general a p-dependence of the mean $\overline{err}^{\mathrm{te}}_{\mathrm{RMSE}}$, is connected to the error metric RMSE. By its definition as square of Eq. 2.6, it depends by $1/\sqrt{N}$ on the number of data-points considered in the average. The individual errors $err^{\mathrm{te}}_{\mathrm{RMSE}}(i)$ at the CV iterations i and hence their average $\overline{err}^{\mathrm{te}}_{\mathrm{RMSE}}$ depend by $1/\sqrt{p}$ on split size for RMSE. This unfavorable behavior of $\overline{err}^{\mathrm{te}}_{\mathrm{RMSE}}$ as a function of p obviously is most pronounced at small split sizes p.[1] We argue hence that the combination of small split size p and the metric RMSE should be avoided by either increasing p or using one of the alternative metrics MSE or MAE instead. Although we found a simple numeric example, showing the strong dependence of mean RMSE and maxAE on p, by contrast to mean MSE and MAE, providing strong proofs would be out of the scope of this thesis. For future research, we suggest to generally favor the report of MAE because it has the same dimension as the target and reflects only the average amount of error magnitude. By contrast, RMSE "measures" three characteristics of a distribution of errors in a confounded way: the average amount of error magnitude, the variance of error magnitude and the number of single error contributions [146].

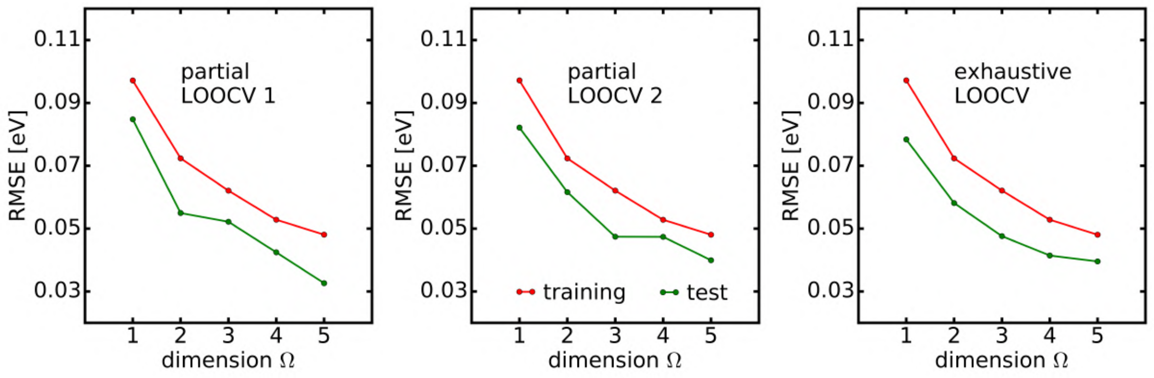


Figure A.4: Mean training and test errors vs. dimension Ω from two independent *non-exhaustive* LOOCV runs (left and middle, $N_{CV} = 50$) and an *exhaustive* LOOCV run ($N_{\mathrm{CV}} = N = 402$, right). Errors are from SC with models from $\mathscr{S}^{all}$. Learning task: E_{mix} of $\mathbf{T}$ with $\mathbf{X}_{E,2}$ on $\mathscr{T}_1$, as in Fig. A.3.

[1]We neglect the influence of the squaring of residuals $(P(s) - f(\mathbf{x}_s))^2$ here.

Appendix B

Details on DFT Calculations and the Materials

B.1 Group IV Ternaries Data Set T

No.	N_A	N_B	N_C	occupation σ
01	4	8	4	0002022211111111
02	4	8	4	0002202211111111
03	5	7	4	0000022211111112
04	5	7	4	0002200211111112
05	5	7	4	0000022211111121
06	5	7	4	0002200211112111
07	5	7	4	0000022211121111
08	5	7	4	0000022221111111
09	5	8	3	0000022211111111
10	5	8	3	0002200211111111
11	6	6	4	0002200011111122
12	6	6	4	0000002211111212
13	6	6	4	0002200011111221
14	6	6	4	0000002211121211
15	6	6	4	0000002211122111
16	6	6	4	0000002212111112
17	6	6	4	0000002212111121
18	6	6	4	0000002212111211
19	6	7	3	0000002211111112
20	6	7	3	0002200011111112
21	6	7	3	0000002211111211
22	6	7	3	0000002212111111

Table B.1: Table of all structures for one ternary type (A, B, C) that are considered in the data set **T** (Part I). (N_A, N_B, N_C) denotes the composition, the occupation vector σ labels the arrangement of the atoms in the supercell. Structures of the same composition are linked by brackets.

No.	N_A	N_B	N_C	occupation σ
23	7	5	4	0000000211111222
24	7	5	4	0000000211112122
25	7	5	4	0000000211122112
26	7	5	4	0000000211212112
27	7	5	4	0000000211212211
28	7	5	4	0000000212212111
29	7	6	3	0000000211111122
30	7	6	3	0000000211112112
31	7	6	3	0000000211122111
32	7	6	3	0000000211212111
33	7	7	2	0000000211111112
34	7	7	2	0000000211112111
35	7	8	1	0000000211111111
36	8	4	4	0000000011121222
37	8	4	4	0000000011122122
38	8	5	3	0000000011111222
39	8	5	3	0000000011122112
40	8	7	1	0000000011111112
41	8	6	2	0000000011111122
42	6	8	2	0000002211111111

Table B.2: Table of all structures for one ternary type (A, B, C) that are considered in the data set **T** (Part II). Structures 40, 41 and 42 are realized in one arrangement only.

No.	N_A	N_B	N_C	**occupation** σ
43	4	8	4	`0022220011111111`
44	4	8	4	`0000222211111111`
45	6	6	4	`0000002211111122`
46	6	6	4	`0000002211112211`
47	6	6	4	`0000002222111111`
48	6	8	2	`0002200011111111`
49	6	6	2	`0002200011122111`
50	8	4	4	`0000000011112222`
51	8	4	4	`0000000011222211`
52	8	6	2	`0000000011122111`

Table B.3: Table of the ternary structures generated by enumlib [127] that were not considered in the initial data set.

GeSnSi	SiSnGe	CSiSn	GeCSi	SiCGe	SiGeSn
42	42	39 (37)	42	42	42
SiGeC	GeSnC	SiSnC	CGeSn	CSnSi	CSnGe
36	9	30	41 (40)	41	42

Table B.4: The 448 structures present in the initial data set **T**, separated by compounds. For GeSnC, only data for 9 of the considered 42 arrangements is available. Numbers in brackets indicate the three samples excluded from the final data set.

ϵ_{F} [10^{-4}]	0.5	1.0	1.2	1.3	1.5	2.0	2.2	3.0
N	164	1419	79	1	17	30	1	10

Table B.5: Statistics of ϵ_{F} / `epsforce` of the ternary data set **T**. Considered are the distorted structures of the initial data if the file `INFO.OUT` reports the atomic forces.

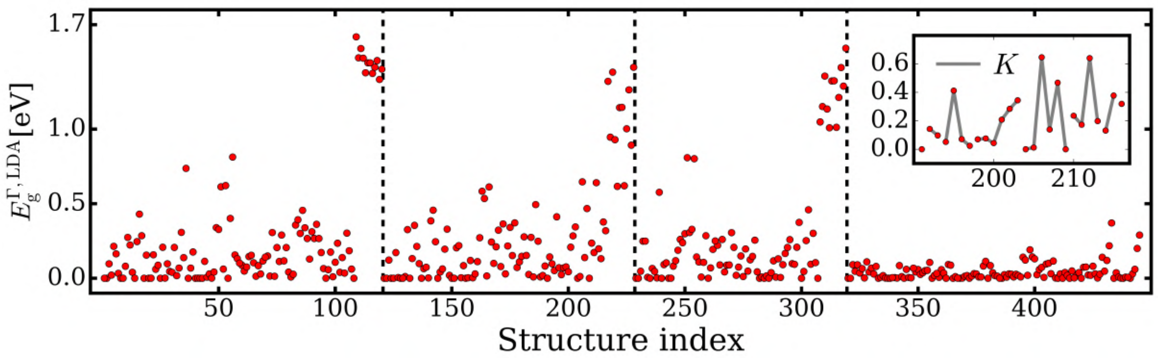


Figure B.1: Band gap $E_g^{\Gamma,\mathrm{LDA}}$ at Γ of the ternary data-set **T**, similar to Fig. 7.8. In the inlet, the remarkable dependence on the arrangement is shown.

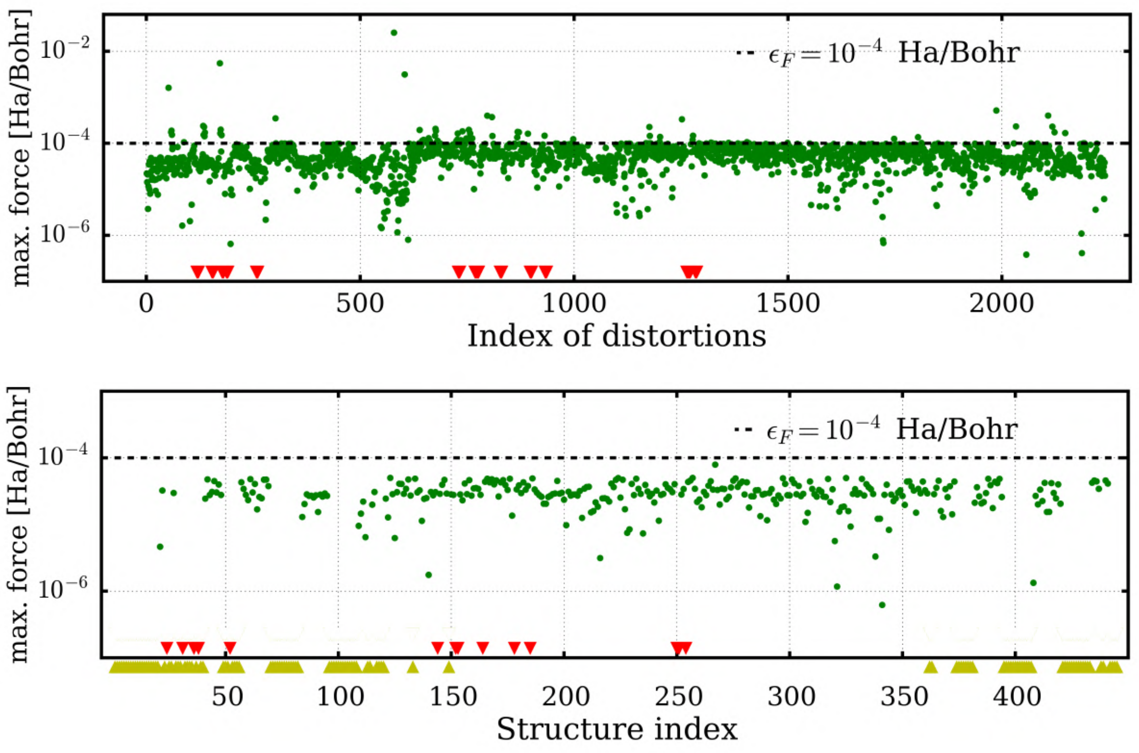


Figure B.2: Convergence analysis of atomic forces for the distorted and relaxed structures of the ternary data set **T**. Maximum component of corrected atomic force (Hellman-Feynman forces including incomplete basis set correction and core correction terms [66]) of the distortions in the initial data set (top) and for the equilibrium structures after data-curing (bottom). The horizontal line indicates the target vale $\epsilon_F = 10^{-4}$ Ha/Bohr, being exceeded several times obviously. Red symbols mark the indices of inconsistent data that could be corrected, yellow symbols structures where the initial data set misses information on the atomic forces (but was kept in the data set).

B.2 Comparison of Hamiltonian Solvers for Isolated Structures

Relaxations of dimers and tetrahedral clusters with the Nelder-Mead algorithm may involve many single runs of `exciting` that are performed efficiently by altering the Hamiltonian solver to the Davidson algorithm ([135], see Sec. 7.3). For this, the speed-up was determined for an isolated tin atom at different box sizes s as a test system. This is illustrated by Fig. B.3 that compares total computation time t of the default and the Davidson algorithm on a standard computer. In both cases, t scales approximately exponentially with s. Fitted exponents are $\kappa = 0.23$ (default solver) and $\kappa = 0.1$ (Davidson solver). For $s = 20\,\text{Bohr}$ which was chosen to calculate the atomic data (see Sec. B.4) t is reduced from $\sim 64{,}000\,\text{s}$ to $\sim 274\,\text{s}$, i.e. by more than two orders of magnitude.

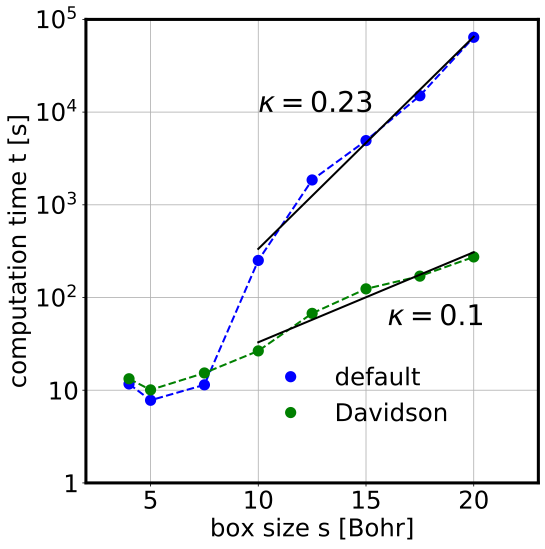
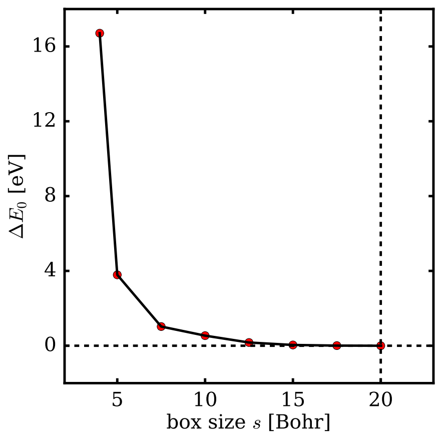


Figure B.3: Scaling of computation time t with respect to box size s using the standard vs. the Davidson solver for Hamiltonian diagonalization. Test system is an isolated Sn atom (B.4). Calculation was run on a standard computer in serial run. Lines indicate exponential fits for $s > 10\,\text{Bohr}$.

B.3 Convergence tests for isolated atoms, dimers and tetrahedral clusters

Convergence with respect to box size s for a Sn atom

For single atom properties (cf. Sec. 7.4.1), convergence of the total energy $E_{0,X}$ and the HOMO-LUMO gap $E_{g,X}$ with respect to box size s was tested on an isolated Sn atom. These tests are shown in Fig. B.4 were s was varied between 4.0 Bohr and 20.0 Bohr. We estimate from the difference with respect to $s = 17.5$ Bohr that E_0 is converged by ~ 5 meV at $s = 20$ Bohr, the size that was applied for all atoms. Similarly comparing 17.5 Bohr and 20. Bohr, $E_{g,X}$ is converged by ~ 0.1 eV. This rather low accuracy has to be considered for the predictive performance of ML models based on these features.



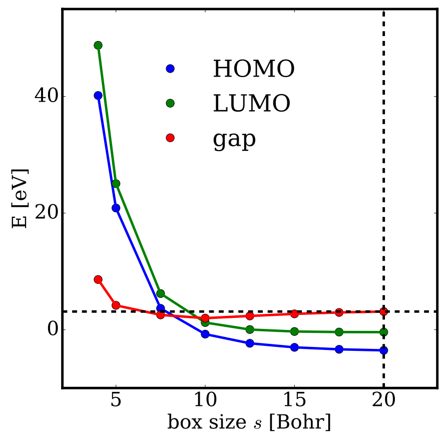


Figure B.4: Influence of box size s for calculating an isolated Sn atom with `exciting`. Left: difference in total energy $E_{0,X}$ with respect to the minimum value vs. box size. Right: HOMO, LUMO and HOMO-LUMO gap $E_{g,X}$ vs. box size. Lines mark the values at the chosen box size $s = 20$ Bohr = 10.58 Å.

Convergence with respect to box size s for the Sn-Sn dimer

The influence of s on the dimer primary features d_{XY}, $E_{f,XY}$ and $E_{g,XY}$ was tested on the dimer Sn-Sn at box sizes of 10, 15, 20 and 25 Bohr, as shown in Fig. B.5. For the equilibrium dimer distance d_{XY} (left panel), at each value of s 5 DFT calculations were performed, varying d_{XY} by $\pm 5\%$ and $\pm 10\%$ around an initial guess. From comparing the minima of quadratic fits of the total energy E_0, the uncertainty in d_{XY} is estimated by ~ 0.13 Bohr (i.e. 0.07 Å or $\pm 4.5\%$) at the chosen value $s = 15$ Bohr. The difference in total energy E_0 (middle panel) differs by 0.7 eV between $s = 15$ Bohr and $s = 25$ Bohr at a guessed value for d_{SnSn}. We could not determine how this affects the actual primary feature $E_{f,XY}$, the energy of formation (see Sec. 8.1.1). For the same set-up the HOMO-LUMO gap $E_{g,XY}$ varies by 0.16 eV between the applied value of s and $s = 25$ Bohr (right panel).

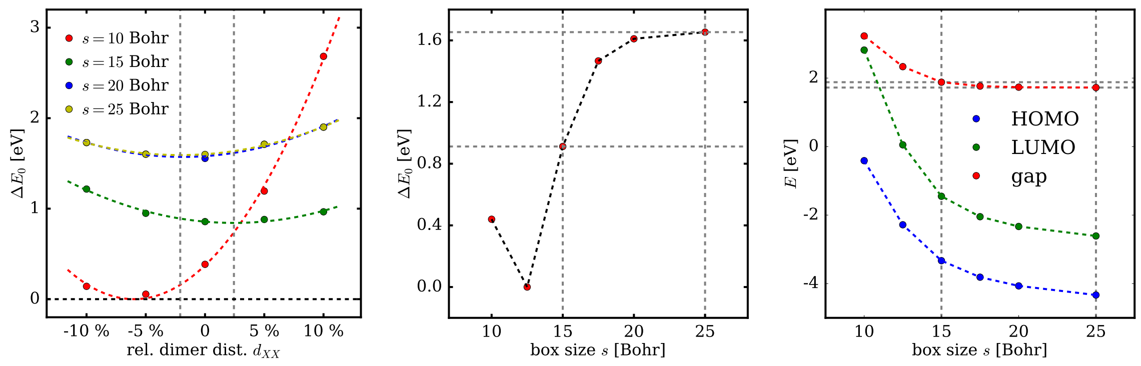


Figure B.5: Tests for the influence of the box size s for an isolated Sn-Sn dimer with `exciting`. Left: total energy vs. relative dimer distance at different box sizes (10, 15, 20, and 25 Bohr). Dimer distance is varied by $\pm 10\%$ around an initial guess for the equilibrium value. Additionally, quadratic fits are plotted. Energy is measured with respect to the fitted minimum (horizontal line) of $s = 10$ Bohr. Vertical lines mark the minima at $s = 25$ Bohr and $s = 10$ Bohr used to determine the uncertainty in d_{XX}. Middle: total energy E_0 (with respect to the minimum) versus box size s at a guessed value of $d_{\mathrm{SnSn}} = 2.75$ Å. The lines indicate the applied size $s = 15$ Bohr $= 7.94$ Å and the maximum $s = 25$ Bohr, and the corresponding values of E_0. Right: HOMO, LUMO and HOMO-LUMO gap $E_{g,XY}$ vs. box size s. Lines again indicate $s = 15$ Bohr and $s = 25$ Bohr and the values of $E_{g,XY}$.

Convergence with respect to box size s for the tetrahedron Sn-Sn-Sn-Sn-Sn

Figure B.6 shows $E_{0,t}$ and $E_{g,t}$ as a function of scaling factor d_t at the used box size of $s = 15$ Bohr and at $s = 30$ Bohr for the tetrahedron Sn-Sn-Sn-Sn-Sn as benchmarking case. In the left part, $E_{0,t}$ is fitted to a *Morse potential* [147] that obviously captures the behavior over d_t. The difference between the minima of the two fits is 0.014 Å illustrating that d_t is determined accurately using $s = 15$ Bohr. Considering the effects of both d_t and s, the uncertainty in $E_{0,t}$ is in the order of a few eV. It is hard, however, to estimate how this transfers to the primary feature $E_{f,t}$ where the reference to the atomic energies introduces many more sources of uncertainty. From the right part it can be be seen that $E_{g,t}$ is largely influenced by both d_t and s. In particular, the two minima of the considered s differ by 0.70 eV from each other. This may limit the predictive power of models including $E_{g,t}$ as a primary feature.

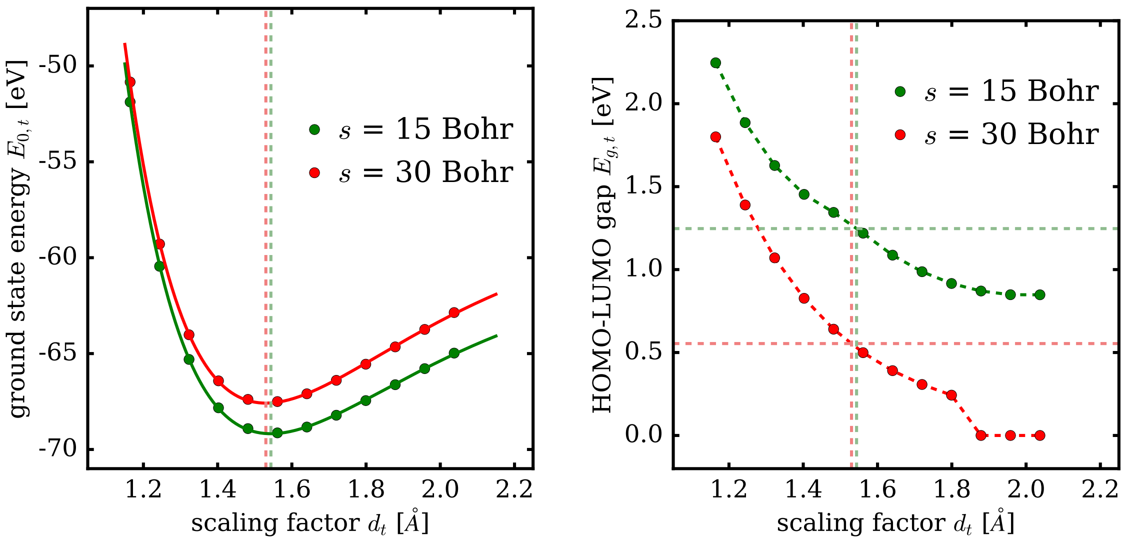


Figure B.6: Influence of the box size s for the tetrahedron Sn-Sn-Sn-Sn-Sn. Left: total energy $E_{0,t}$ (offset 839900 eV) versus scaling factor d_t for $s = 15$ Bohr (green) and $s = 30$ Bohr (red). Data points are fitted to a Morse potential of which the respective minimima in d_t are indicated by the vertical lines of corresponding color. Right: gap energy $E_{g,t}$ versus d_t for the two box sizes s. The lines mark the minima in d_t from the right panel, and the corresponding values.

B.4 Single Atom Data

A	Z	P	$E_{g,X}$	$E_{0,X}$	IP*	EA*	r_s*	r_p*	r_d*
C	6	2	4.99	-1018.81	-10.85	-0.87	0.64	0.63	1.63
Si	14	3	1.97	-6435.56	-7.76	-0.99	0.94	1.13	1.89
Ge	32	4	2.51	-53370.75	-7.57	-0.95	0.92	1.16	2.37
Sn	50	5	3.12	-167992.10	-7.04	-1.04	1.06	1.34	2.03

Table B.6: List of the used single atom data. Z and P are the nuclear number and the period from the table of elements, $E_{g,X}$ and $E_{0,X}$ the HOMO-LUMO gap and the total energy (in eV) which were generated with the code **`exciting`**. The ionization potential IP, the electron affinity EA (in eV) and the s-, p- and d-orbital radii (in Å) result from `FHIaims` calculations and were taken from the supplementary information of [1] (this is indicated by "*").

B.5 Dimer Data

A	B	d_{XY}	$E_{g,XY}$	$E_{0,XY}$
C	C	1.29	6.63	-2047.23
C	Si	1.69	3.89	-8884.71
C	Ge	1.76	3.79	-58060.05
C	Sn	1.99	3.00	-169016.50
Si	Si	2.26	2.71	-15723.49
Si	Ge	2.34	2.56	-64898.94
Si	Sn	2.57	2.18	-175855.79
Ge	Ge	2.42	2.40	-114073.75
Ge	Sn	2.62	2.10	-225030.50
Sn	Sn	2.83	1.88	-335986.26

Table B.7: List of the dimers calculated for this work and the respective primary features d_{XY} (dimer distance in Å), $E_{g,XY}$ (HOMO-LUMO gap in eV) and $E_{0,XY}$ (ground-state energy in eV). Details on the calculations are described in Sec. 7.4.2.

B.6 Tetrahedron Data

A	B	C	D	E	d_t	$E_{0,t}$	$E_{g,t}$
C	C	C	C	Ge	0.79	-61134.40	0.97
C	C	C	C	Si	0.78	-11959.61	1.05
C	C	C	C	Sn	0.85	-172089.11	0.79
C	C	C	Ge	Ge	0.86	-117149.06	0.82
C	C	C	Ge	Sn	0.90	-228104.74	0.73
C	C	C	Si	Ge	0.85	-67973.93	0.85
C	C	C	Si	Si	0.83	-18798.87	0.90
C	C	C	Si	Sn	0.89	-178929.42	0.75
C	C	C	Sn	Sn	0.94	-339060.77	0.67
C	C	Ge	Ge	Ge	0.91	-173164.71	0.92
C	C	Ge	Ge	Sn	0.95	-284120.92	0.88
C	C	Ge	Sn	Sn	0.98	-395077.41	0.87
C	C	Si	Ge	Ge	0.90	-123989.41	0.94
C	C	Si	Si	Ge	0.89	-74814.14	0.96
C	C	Si	Si	Si	0.87	-25638.93	0.99
C	C	Si	Si	Sn	0.93	-185770.10	0.90
C	C	Si	Sn	Sn	0.97	-345901.91	0.87
C	C	Sn	Sn	Sn	1.01	-506034.12	0.88
C	Ge	Ge	Ge	Ge	0.95	-229181.05	1.29
C	Ge	Ge	Ge	Sn	0.98	-340137.58	1.25
C	Ge	Ge	Sn	Sn	1.01	-451094.30	1.24
C	Ge	Sn	Sn	Sn	1.03	-562051.19	1.29

Table B.8: List of the calculated tetrahedron clusters used for the ternary data set **T**, and the physical properties d_t (bond length in Å, scaled to a radius equivalent), $E_{0,t}$ (ground-state energy in eV) and $E_{g,t}$ (HOMO-LUMO gap in eV). Details on the data generation are given in Sec. 7.4.3. - Part I.

A	B	C	D	E	d_t	$E_{0,t}$	$E_{g,t}$
C	Si	Ge	Ge	Ge	0.94	-180005.61	1.29
C	Si	Si	Ge	Ge	0.93	-130830.20	1.30
C	Si	Si	Si	Ge	0.92	-81654.82	1.32
C	Si	Si	Si	Si	0.91	-32479.48	1.34
C	Si	Si	Si	Sn	0.96	-192611.09	1.25
C	Si	Si	Sn	Sn	1.00	-352743.17	1.22
C	Si	Sn	Sn	Sn	1.03	-512875.57	1.28
C	Sn	Sn	Sn	Sn	1.06	-673008.18	1.40
Ge	C	C	C	C	0.93	-61128.39	0.65
Ge	C	C	C	Ge	1.01	-117143.02	0.52
Ge	C	C	C	Si	1.00	-67967.55	0.54
Ge	C	C	C	Sn	1.05	-228099.07	0.46
Ge	C	C	Ge	Ge	1.07	-173158.66	0.50
Ge	C	C	Ge	Sn	1.10	-284115.15	0.48
Ge	C	C	Si	Ge	1.06	-123983.08	0.50
Ge	C	C	Si	Si	1.05	-74807.51	0.51
Ge	C	C	Sn	Sn	1.14	-395071.85	0.46
Ge	C	Ge	Ge	Ge	1.12	-229174.88	0.65
Ge	C	Ge	Ge	Sn	1.15	-340131.63	0.66
Ge	C	Ge	Sn	Sn	1.18	-451088.53	0.68
Ge	C	Si	Ge	Ge	1.11	-179999.23	0.65
Ge	C	Si	Si	Ge	1.11	-130823.59	0.65

Table B.9: Tetrahedron Data - Part II.

A	B	C	D	E	d_t	$E_{0,t}$	$E_{g,t}$
Ge	C	Si	Si	Si	1.10	-81647.96	0.65
Ge	C	Sn	Sn	Sn	1.21	-562045.53	0.73
Ge	Ge	Ge	Ge	Sn	1.19	-396145.58	1.09
Ge	Ge	Ge	Sn	Sn	1.21	-507102.41	1.11
Ge	Ge	Sn	Sn	Sn	1.23	-618059.31	1.20
Ge	Si	Ge	Ge	Ge	1.15	-236014.98	1.05
Ge	Si	Ge	Ge	Sn	1.18	-346971.85	1.07
Ge	Si	Ge	Sn	Sn	1.20	-457928.81	1.09
Ge	Si	Si	Ge	Ge	1.15	-186839.49	1.05
Ge	Si	Si	Ge	Sn	1.17	-297796.33	1.05
Ge	Si	Si	Si	Ge	1.14	-137664.00	1.05
Ge	Si	Si	Si	Si	1.13	-88488.52	1.05
Ge	Si	Si	Si	Sn	1.17	-248620.81	1.05
Ge	Si	Si	Sn	Sn	1.20	-408753.27	1.07
Ge	Si	Sn	Sn	Sn	1.23	-568885.86	1.18
Ge	Sn	Sn	Sn	Sn	1.25	-729016.28	1.34
Si	C	C	C	C	0.89	-11954.73	0.66
Si	C	C	C	Ge	0.97	-67968.72	0.52
Si	C	C	C	Si	0.96	-18793.28	0.54
Si	C	C	C	Sn	1.02	-178924.52	0.46
Si	C	C	Ge	Ge	1.04	-123984.02	0.48
Si	C	C	Si	Ge	1.03	-74808.47	0.49

Table B.10: Tetrahedron Data - Part III.

A	B	C	D	E	d_t	$E_{0,t}$	$E_{g,t}$
Si	C	C	Si	Si	1.02	-25632.93	0.49
Si	C	C	Si	Sn	1.07	-185764.75	0.46
Si	C	C	Sn	Sn	1.11	-345896.93	0.45
Si	C	Ge	Ge	Ge	1.09	-180000.05	0.62
Si	C	Si	Ge	Ge	1.09	-130824.42	0.62
Si	C	Si	Si	Ge	1.08	-81648.80	0.62
Si	C	Si	Si	Si	1.07	-32473.20	0.62
Si	C	Si	Si	Sn	1.11	-192605.39	0.62
Si	C	Si	Sn	Sn	1.15	-352737.83	0.64
Si	C	Sn	Sn	Sn	1.19	-512870.46	0.68
Si	Ge	Ge	Ge	Ge	1.14	-236015.73	1.05
Si	Ge	Ge	Ge	Sn	1.16	-346972.56	1.06
Si	Ge	Ge	Sn	Sn	1.19	-457929.47	1.08
Si	Ge	Sn	Sn	Sn	1.21	-568886.49	1.16
Si	Si	Ge	Ge	Ge	1.13	-186840.26	1.03
Si	Si	Ge	Ge	Sn	1.16	-297797.04	1.04
Si	Si	Ge	Sn	Sn	1.18	-408753.95	1.06
Si	Si	Si	Ge	Ge	1.12	-137664.78	1.02
Si	Si	Si	Ge	Sn	1.15	-248621.54	1.02
Si	Si	Si	Si	Ge	1.12	-88489.31	1.03
Si	Si	Si	Si	Sn	1.15	-199446.05	1.02
Si	Si	Si	Sn	Sn	1.18	-359578.43	1.04

Table B.11: Tetrahedron Data - Part IV.

A	B	C	D	E	d_t	$E_{0,t}$	$E_{g,t}$
Si	Si	Sn	Sn	Sn	1.20	-519710.95	1.14
Si	Sn	Sn	Sn	Sn	1.23	-679843.58	1.30
Sn	C	C	C	C	1.04	-172082.95	0.48
Sn	C	C	C	Ge	1.11	-228098.15	0.41
Sn	C	C	C	Si	1.10	-178922.55	0.42
Sn	C	C	C	Sn	1.15	-339054.54	0.38
Sn	C	C	Ge	Ge	1.17	-284114.05	0.42
Sn	C	C	Ge	Sn	1.21	-395070.75	0.41
Sn	C	C	Si	Si	1.15	-185762.73	0.43
Sn	C	C	Si	Sn	1.20	-345895.05	0.42
Sn	C	C	Sn	Sn	1.24	-506027.60	0.41
Sn	C	Ge	Ge	Ge	1.22	-340130.38	0.61
Sn	C	Ge	Ge	Sn	1.25	-451087.29	0.62
Sn	C	Ge	Sn	Sn	1.28	-562044.29	0.64
Sn	C	Si	Si	Si	1.20	-192603.27	0.59
Sn	C	Si	Si	Sn	1.24	-352735.83	0.60
Sn	C	Si	Sn	Sn	1.27	-512868.54	0.63
Sn	C	Sn	Sn	Sn	1.30	-673001.38	0.71
Sn	Ge	Ge	Ge	Ge	1.26	-396144.20	0.99
Sn	Ge	Ge	Ge	Sn	1.28	-507101.04	1.01
Sn	Ge	Ge	Sn	Sn	1.30	-618057.95	1.03
Sn	Ge	Sn	Sn	Sn	1.32	-729014.92	1.13

Table B.12: Tetrahedron Data - Part V.

A	B	C	D	E	d_t	$E_{0,t}$	$E_{g,t}$
Sn	Si	Ge	Ge	Ge	1.25	-346970.45	0.97
Sn	Si	Ge	Ge	Sn	1.27	-457927.42	0.99
Sn	Si	Ge	Sn	Sn	1.30	-568884.48	1.00
Sn	Si	Si	Ge	Ge	1.24	-297794.90	0.95
Sn	Si	Si	Ge	Sn	1.27	-408751.86	0.97
Sn	Si	Si	Si	Ge	1.24	-248619.37	0.95
Sn	Si	Si	Si	Si	1.23	-199443.83	0.94
Sn	Si	Si	Si	Sn	1.26	-359576.31	0.96
Sn	Si	Si	Sn	Sn	1.29	-519708.90	0.98
Sn	Si	Sn	Sn	Sn	1.32	-679841.59	1.10

Table B.13: Tetrahedron Data - Part VI.

Listing B.1: `input.xml` for relaxation of a tetrahedron cluster. *s* denotes the box size and *d* the tetrahedron scale (lines 10 to 13 and 16) which are modified by the external script. The central atom of the cluster is at the center of the box, the outer atoms are shifted such that the shape is regular.

```
 1 <input>
 2   <title>Sn-Ge-Ge-Ge-Ge relaxation</title>
 3   <structure speciespath="../species/" cartesian="true">
 4     <crystal>
 5       <basevect>15.0 0.0 0.0</basevect> <!--box size-->
 6       <basevect>0.0 15.0 0.0</basevect> <!--box size-->
 7       <basevect>0.0 0.0 15.0</basevect> <!--box size-->
 8     </crystal>
 9     <species speciesfile="Ge.xml" rmt="1.80"> <!--outer atom-->
10       <atom coord="s/2 + d,  s/2 + d, s/2 - d"/> <!--outer atom-->
11       <atom coord="s/2 - d, s/2 - d, s/2 - d"/> <!--outer atom-->
12       <atom coord="s/2 - d, s/2 + d, s/2 + d"/> <!--outer atom-->
13       <atom coord="s/2 + d, s/2 - d, s/2 + d"/> <!--outer atom-->
14     </species>
15     <species speciesfile="Sn.xml" rmt="1.90"> <!--inner atom-->
16       <atom coord="s/2, s/2, s2"/> <!--inner atom at center-->
17     </species>
18   </structure>
19   <groundstate
20     do="fromscratch"
21     rgkmax="7.0"
22     gmaxvr="16"
23     ngridk="1 1 1"
24     xctype="LDA_PW"
25     epsengy="0.0001"
26     >
27   <solver type="Davidson" constructHS="false" />
28   </groundstate>
29 </input>
```

B.7 Details on the Octet Binaries O - Tight Binding Data and LDA Gaps

A	B	$E_{\text{gap}}^{\Gamma,\text{TB}}$	$E_{\text{gap}}^{\Gamma,\text{LDA}}$	E_{VB}^{L}	E_{CB}^{L}	E_{VB}^{Γ}	E_{CB}^{Γ}	E_{VB}^{X}	E_{CB}^{X}	$E_{\text{VB}}^{U/K}$	$E_{\text{CB}}^{U/K}$
Li	F	11.866	11.2375	-13.91	0.35	-12.07	-0.20	-15.70	0.87	-15.54	0.70
Li	Cl	8.5464	7.1724	-10.14	-0.81	-9.11	-0.57	-11.31	-0.26	-11.08	-0.43
Li	Br	7.713	6.3326	-9.23	-1.12	-8.35	-0.64	-10.29	-0.59	-10.05	-0.72
Li	I	6.5035	5.3314	-8.23	-1.37	-7.51	-1.01	-9.14	-0.97	-8.89	-1.04
Be	O	10.2192	9.4537	-13.76	0.58	-10.76	-0.54	-16.99	1.42	-16.12	1.11
Be	S	6.7458	5.6132	-9.74	-1.57	-7.98	-1.23	-11.76	-0.52	-11.16	-0.85
Be	Se	6.0793	5.1952	-8.97	-2.10	-7.42	-1.34	-10.77	-0.94	-10.23	-1.26
Be	Te	4.7332	4.5769	-8.00	-2.51	-6.72	-1.99	-9.49	-1.53	-9.04	-1.74
B	N	8.8369	9.4373	-13.71	0.38	-9.90	-1.06	-17.63	1.65	-16.48	1.12
B	P	5.4884	5.4587	-9.86	-2.29	-7.40	-1.91	-12.37	-0.62	-11.63	-1.20
B	As	4.8841	5.4978	-9.21	-3.07	-6.99	-2.10	-11.47	-1.11	-10.80	-1.76
C	C	8.3303	11.2296	-13.68	0.29	-9.58	-1.25	-17.78	1.91	-16.58	1.12
Na	F	11.3874	9.981	-12.75	-0.83	-11.70	-0.31	-13.93	-0.19	-13.75	-0.36
Na	Cl	7.9016	7.0256	-9.61	-1.37	-8.95	-1.05	-10.46	-1.06	-10.24	-1.11
Na	Br	6.8668	6.2667	-8.80	-1.55	-8.22	-1.35	-9.59	-1.23	-9.36	-1.28
Na	I	5.8642	5.3459	-7.89	-1.70	-7.42	-1.55	-8.58	-1.46	-8.36	-1.49
Mg	O	9.3487	6.923	-11.67	-1.16	-9.95	-0.60	-13.80	-0.10	-13.15	-0.41
Mg	S	5.7907	4.7234	-8.60	-2.21	-7.55	-1.76	-9.94	-1.54	-9.53	-1.64
Mg	Se	4.8065	4.4046	-8.00	-2.47	-7.05	-2.25	-9.20	-1.75	-8.84	-1.86
Mg	Te	3.84	3.9125	-7.21	-2.71	-6.43	-2.59	-8.22	-2.10	-7.91	-2.18

Table B.14: Tight binding band gap $E_{\text{gap}}^{\Gamma,\text{TB}}$ and LDA band gap $E_{\text{gap}}^{\Gamma,\text{LDA}}$ at the high-symmetry point Γ, as well as energies of the highest valence band E_{VB} and the lowest conduction band E_{CB} at the high-symmetry points L, Γ, X and U/K, for the octet binary data set **O**. - Part I

A	B	$E_{\text{gap}}^{\Gamma,\text{TB}}$	$E_{\text{gap}}^{\Gamma,\text{LDA}}$	E_{VB}^{L}	E_{CB}^{L}	E_{VB}^{Γ}	E_{Γ}^{Z}	E_{VB}^{X}	E_{CB}^{X}	$E_{\text{VB}}^{U/K}$	$E_{\text{CB}}^{U/K}$
Al	N	7.0863	5.0483	-10.92	-2.01	-8.55	-1.47	-13.49	-0.47	-12.73	-0.95
Al	P	3.4136	3.5667	-7.96	-3.44	-6.44	-3.03	-9.60	-2.07	-9.11	-2.33
Al	As	2.4652	3.5286	-7.56	-3.76	-6.14	-3.68	-9.10	-2.20	-8.64	-2.50
Al	Sb	1.5251	3.3522	-6.86	-3.99	-5.66	-4.14	-8.17	-2.57	-7.78	-2.80
Si	C	5.6862	4.5259	-10.33	-2.34	-7.63	-1.95	-13.05	-0.54	-12.25	-1.16
Si	Si	2.4169	3.4718	-7.70	-3.87	-5.94	-3.53	-9.45	-1.85	-8.94	-2.46
K	F	10.6259	8.2658	-12.18	-1.18	-11.53	-0.91	-12.90	-0.82	-12.82	-0.88
K	Cl	7.5602	6.4702	-9.29	-1.47	-8.86	-1.30	-9.84	-1.29	-9.72	-1.31
K	Br	6.6546	5.8766	-8.52	-1.58	-8.14	-1.49	-9.03	-1.39	-8.91	-1.42
K	I	5.7679	5.1195	-7.68	-1.66	-7.36	-1.59	-8.14	-1.52	-8.01	-1.53
Ca	O	8.735	4.8303	-10.84	-1.55	-9.67	-0.94	-12.27	-0.79	-11.89	-0.99
Ca	S	5.6265	3.6001	-8.20	-2.10	-7.42	-1.79	-9.23	-1.61	-8.91	-1.69
Ca	Se	4.8432	3.3751	-7.66	-2.27	-6.94	-2.10	-8.61	-1.74	-8.31	-1.82
Ca	Te	4.0321	2.9933	-6.95	-2.42	-6.35	-2.32	-7.76	-1.98	-7.51	-2.04
Cu	F	10.3991	5.066	-13.10	-1.93	-11.81	-1.41	-14.85	-1.15	-14.30	-1.27
Cu	Cl	7.5477	4.7088	-10.21	-2.08	-9.13	-1.59	-11.64	-1.51	-11.20	-1.59
Cu	Br	6.292	4.4431	-9.34	-2.36	-8.38	-2.09	-10.62	-1.75	-10.22	-1.83
Cu	I	5.2507	4.3307	-8.40	-2.52	-7.57	-2.32	-9.51	-1.99	-9.17	-2.05
Zn	O	8.7487	7.5577	-11.69	-2.04	-9.95	-1.21	-13.83	-0.94	-13.18	-1.14
Zn	S	5.3238	5.5213	-8.87	-2.73	-7.64	-2.31	-10.39	-1.89	-9.93	-2.01

Table B.15: Continuing Tab. B.14. - Part II

A	B	$E_{\text{gap}}^{\Gamma,\text{TB}}$	$E_{\text{gap}}^{\Gamma,\text{LDA}}$	E_{VB}^{L}	E_{CB}^{L}	E_{VB}^{Γ}	E_{Γ}^{Z}	E_{VB}^{X}	E_{CB}^{X}	$E_{\text{VB}}^{U/K}$	$E_{\text{CB}}^{U/K}$
Zn	Se	4.1937	5.0872	-8.23	-3.06	-7.13	-2.93	-9.60	-2.14	-9.18	-2.26
Zn	Te	3.2306	4.4883	-7.43	-3.27	-6.50	-3.27	-8.60	-2.46	-8.24	-2.55
Ga	N	6.3814	6.1068	-10.64	-3.06	-8.44	-2.06	-13.07	-1.43	-12.35	-1.78
Ga	P	2.6237	4.2522	-7.99	-3.98	-6.45	-3.83	-9.66	-2.47	-9.16	-2.71
Ga	As	1.5756	4.0772	-7.57	-4.31	-6.14	-4.57	-9.12	-2.60	-8.66	-2.87
Ga	Sb	0.7292	3.4876	-6.90	-4.45	-5.68	-4.95	-8.23	-2.88	-7.83	-3.10
Ge	Ge	0.5263	3.7739	-7.31	-4.67	-5.69	-5.16	-8.93	-2.22	-8.45	-2.88
Rb	F	10.3618	7.461	-12.02	-1.32	-11.49	-1.12	-12.62	-1.03	-12.55	-1.08
Rb	Cl	7.4083	6.2946	-9.19	-1.55	-8.83	-1.42	-9.66	-1.41	-9.56	-1.42
Rb	Br	6.5556	5.76	-8.45	-1.63	-8.12	-1.56	-8.88	-1.48	-8.78	-1.49
Rb	I	5.6878	5.0612	-7.62	-1.70	-7.34	-1.65	-8.01	-1.58	-7.90	-1.60
Sr	O	8.1988	4.4849	-10.48	-1.77	-9.56	-1.36	-11.65	-1.18	-11.34	-1.30
Sr	S	5.3835	3.5067	-7.99	-2.17	-7.35	-1.97	-8.85	-1.79	-8.58	-1.84
Sr	Se	4.6738	3.2933	-7.47	-2.30	-6.88	-2.21	-8.27	-1.88	-8.02	-1.93
Sr	Te	3.9334	2.9438	-6.80	-2.41	-6.30	-2.37	-7.49	-2.07	-7.27	-2.11
Ag	F	9.2237	6.2688	-12.50	-2.58	-11.63	-2.40	-13.75	-2.08	-13.36	-2.14
Ag	Cl	6.5372	5.9989	-9.73	-2.64	-8.99	-2.45	-10.78	-2.24	-10.45	-2.29
Ag	Br	5.5274	5.6812	-8.95	-2.78	-8.26	-2.74	-9.91	-2.34	-9.61	-2.39
Ag	I	4.6662	5.2786	-8.09	-2.84	-7.47	-2.81	-8.96	-2.44	-8.69	-2.48
Cd	O	7.244	6.7342	-10.97	-2.85	-9.71	-2.47	-12.60	-1.99	-12.10	-2.11
Cd	S	4.4844	5.3078	-8.46	-3.18	-7.50	-3.02	-9.69	-2.47	-9.31	-2.56

Table B.16: Continuing Tab. B.14. - Part III

A	B	$E_{\text{gap}}^{\Gamma,\text{TB}}$	$E_{\text{gap}}^{\Gamma,\text{LDA}}$	E_{VB}^{L}	E_{CB}^{L}	E_{VB}^{Γ}	E_{Γ}^{Z}	E_{VB}^{X}	E_{CB}^{X}	$E_{\text{VB}}^{U/K}$	$E_{\text{CB}}^{U/K}$
Cd	Se	3.574	4.8865	-7.89	-3.38	-7.02	-3.44	-9.02	-2.60	-8.67	-2.69
Cd	Te	2.791	4.3822	-7.18	-3.50	-6.42	-3.63	-8.16	-2.79	-7.86	-2.87
In	N	4.7233	5.2631	-9.76	-3.81	-8.08	-3.36	-11.68	-2.37	-11.11	-2.62
In	P	1.9018	3.9868	-7.51	-4.24	-6.25	-4.35	-8.92	-2.86	-8.50	-3.06
In	As	1.0542	3.7837	-7.15	-4.45	-5.97	-4.91	-8.47	-2.92	-8.08	-3.14
In	Sb	0.4028	3.5449	-6.57	-4.50	-5.54	-5.14	-7.72	-3.09	-7.38	-3.27
Sn	Sn	0.0	3.2327	-6.33	-4.76	-5.11	-5.11	-7.55	-2.63	-7.19	-3.18
B	Sb	3.348	4.4638	-8.24	-3.61	-6.39	-3.04	-10.13	-1.77	-9.58	-2.29
Cs	F	10.2284	6.2993	-11.88	-1.34	-11.45	-1.22	-12.38	-1.14	-12.32	-1.17
Cs	Cl	7.3473	5.887	-9.10	-1.54	-8.80	-1.46	-9.48	-1.44	-9.40	-1.45
Cs	Br	6.5294	5.446	-8.36	-1.60	-8.10	-1.57	-8.73	-1.49	-8.64	-1.51
Cs	I	5.6888	4.8576	-7.55	-1.66	-7.32	-1.63	-7.88	-1.58	-7.79	-1.58
Ba	O	7.9759	3.9811	-10.23	-1.79	-9.48	-1.51	-11.20	-1.34	-10.95	-1.42
Ba	S	5.3121	3.2461	-7.82	-2.13	-7.30	-1.99	-8.55	-1.83	-8.32	-1.87
Ba	Se	4.657	3.0409	-7.32	-2.22	-6.84	-2.18	-8.00	-1.90	-7.78	-1.94
Ba	Te	3.9688	2.7295	-6.68	-2.32	-6.27	-2.30	-7.26	-2.05	-7.08	-2.08
Ge	C	5.0512	5.2772	-9.81	-3.37	-7.36	-2.30	-12.28	-1.41	-11.56	-1.91
Sn	C	3.3423	4.4081	-8.84	-4.05	-6.86	-3.52	-10.87	-2.14	-10.27	-2.57
Ge	Si	1.5248	3.5339	-7.52	-4.28	-5.82	-4.30	-9.21	-2.18	-8.71	-2.69
Sn	Si	0.7029	3.3758	-6.93	-4.47	-5.49	-4.79	-8.37	-2.47	-7.95	-2.94
Sn	Ge	0.0	3.2708	-6.77	-4.76	-5.38	-5.38	-8.16	-2.48	-7.75	-3.07

Table B.17: Continuing Tab. B.14. - Part IV

atom type	E_{s}^{OS}	E_{p}^{OS}
Li	-2.87372	-0.97849
Be	-5.60006	-2.09804
B	-9.38334	-3.71510
C	-13.63871	-5.41560
N	-18.42010	-7.23881
O	-23.75188	-9.19678
F	-29.64662	-11.29406
Na	-2.81912	-0.7176
Mg	-4.78163	-1.35823
Al	-7.82944	-2.78373
Si	-10.87810	-4.16254
P	-14.01512	-5.59628
S	-17.27899	-7.10616
Cl	-20.68981	-8.70047
K	-2.42615	-0.69742
Ca	-3.86382	-1.41468
Cu	-4.85627	-0.64114
Zn	-6.21748	-1.19409
Ga	-9.16739	-2.73241
Ge	-11.94245	-4.04600
As	-14.69575	-5.34066
Se	-17.47846	-6.65393
Br	-20.31439	-8.00053
Rb	-2.36004	-0.70502
Sr	-3.64054	-1.27859
Ag	-4.70978	-0.47947
Cd	-5.95194	-1.30926
In	-8.47784	-2.69690
Sn	-10.80594	-3.86649
Sb	-13.07980	-4.99107
Te	-15.35110	-6.10902
I	-17.64440	-7.23614
Cs	-2.22029	-0.54790
Ba	-3.34602	-1.11536

Table B.18: Atomic on-site energies E_s^{OS} and E_p^{OS} of s- and p-state used for the tight binding calculations. The values were obtained by the code `FHIaims` and provided by [148].

Appendix C

Additional Material on the Implementation

C.1 Parallelization of the Subspace Update in SISSO

The iterative search of the subsets $\tilde{\mathbf{S}}_j$ in our implementation of SISSO is illustrated in the Scheme of Fig. C.1. Here, the two branches refer to tiers $\mathscr{T}_{0/1}$ and the product tier, respectively, and proceed analogously. On the left, first the column indices are grouped to N_p chunks and distributed to the processes. A parallel execution of the function `updater_1` by `pool.map` then yields all absolute correlations $|\rho(i)|$ to the residue. These are used to determine a ranking I_ρ of the columns from which the best M_{SIS} are added to the subset $\tilde{\mathbf{S}}_j$. In the right part, illustrating the subset update for the product tier $\mathscr{T}_2$ / $\mathscr{T}_2^{\text{r}}$, first N_p groups I_Π^m of index pairs (k, l) are determined. These are distributed to the worker processes that both calculate the product feature $\text{x}_k \circ \text{x}_l$ and its correlation $\rho(k, l)$ to the residue. Similarly, the M_{SIS} best features are found which - if they improve over the features of lower tiers already present in the subset - are added to the respective $\tilde{\mathbf{S}}_j$. The procedure stops after Ω iterations have been performed (cf. Sec. 5.3.2).

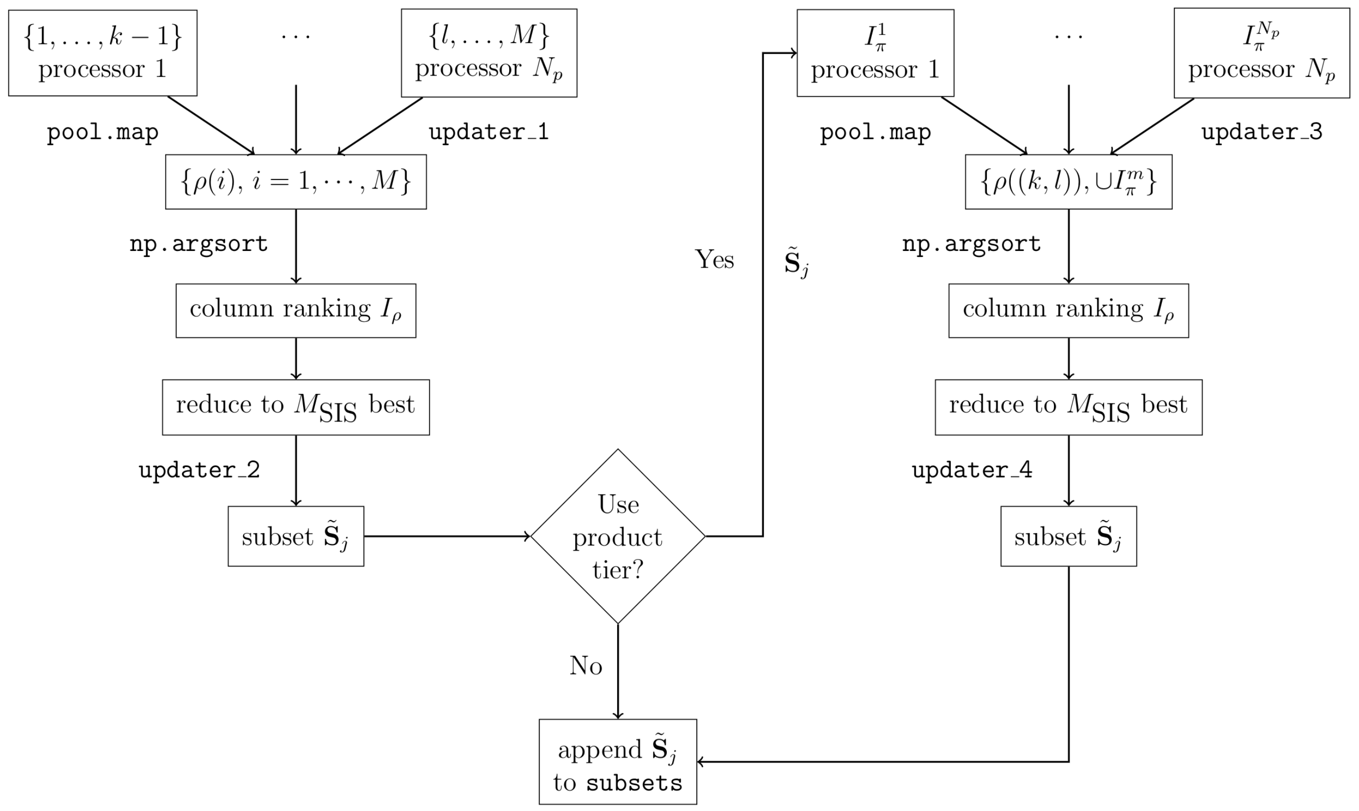

$\{1,\ldots,k-1\}$ processor 1
$\cdots$
$\{l,\ldots,M\}$ processor N_p
pool.map
updater_1
$\{\rho(i),\, i=1,\cdots,M\}$
np.argsort
column ranking I_ρ
reduce to M_{SIS} best
updater_2
subset $\tilde{\mathbf{S}}_j$
Use product tier?
Yes
$\tilde{\mathbf{S}}_j$
No
append $\tilde{\mathbf{S}}_j$ to subsets
I_π^1 processor 1
$\cdots$
$I_\pi^{N_p}$ processor N_p
pool.map
updater_3
$\{\rho((k,l)), \cup I_\pi^m\}$
np.argsort
column ranking I_ρ
reduce to M_{SIS} best
updater_4
subset $\tilde{\mathbf{S}}_j$

Figure C.1: Flow diagram of the subspace update in the developed SISSO implementation. The left branch refers to features up to $\mathscr{T}_1$, the right branch to features of $\mathscr{T}_2$ or $\mathscr{T}_2^{\mathrm{r}}$, considering product combinations. For the parallelization, feature indices or index pairs are distributed in groups to the N_p processes, `pool.map()` is used to execute the parallelized functions `updater_1` and `updater_3`. The former calculates the absolute correlation $|\rho(i)|$ of a column $\mathbf{x}_i$ to the target, the latter calculates a product feature $\mathbf{x}_k \circ \mathbf{x}_l$ on the fly, determines its correlation $|\rho((k,l))|$ and discards it afterwards. The functions `updater_2` and `updater_4` handle through the subset update by comparing the correlations. The list `subsets` contains all subsets to yield the pre-selection I_{ℓ_0} of the most relevant features.

C.2 Implementation of Cross-Validation in SISSO

At cross-validation, the developed implementation of SISSO basically resembles the same steps as executed on the total data. The N_{CV} CV iterations z however extend the learning task by one-dimension. The target $\mathbf{P}$ thus transforms to a two-dimensional matrix $\{P_{z,k}\}$ and the data matrix into a three-dimensional matrix $\{x_{z,k,i}\}$ (here, k denotes a sample index from the respective training split $U_{tr}^{(z)}$). The calculation of the correlations ρ is then executed parallelized *by feature index* i, i. e. by columns of $\mathbf{X}$, while the CV splits are treated *simultaneously*. That way, the learning task turns into a *multi-task* problem $\mathscr{L}^{\mathrm{mt}}$[149] to learn N_{CV} models $\mathscr{D}^{(z)}$ at the same time. Figure C.2 on the right illustrates this strategy of parallelization, in contrast to the other dimensionality reduction methods $\mathscr{M}$ on the left - these are parallelized by CV iteration z.

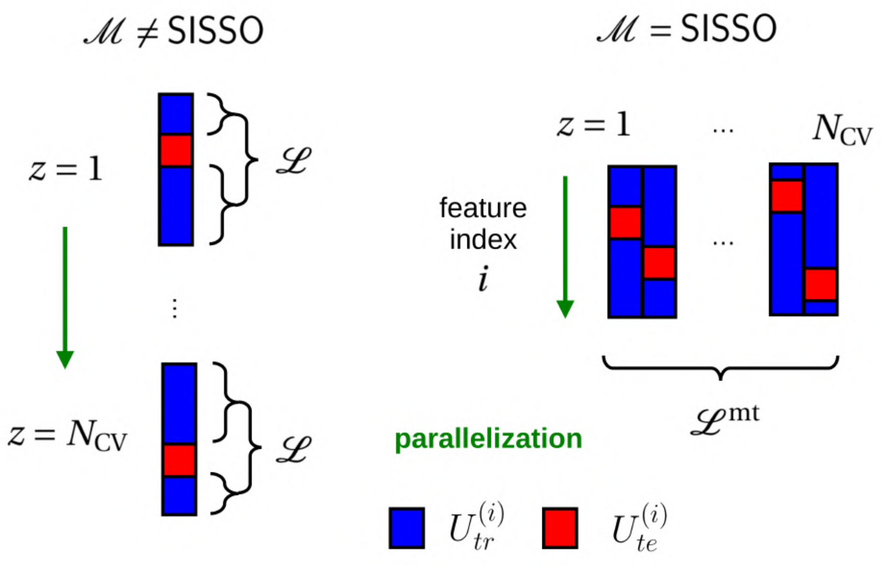


Figure C.2: Comparison of the parallelization of CV in SISSO and the other dimensionality reduction methods $\mathscr{M}$. The scheme is analogous to Figs. 5.6 and 5.7, the green arrow marks the direction of the parallelization. Left: if $\mathscr{M} \neq$SISSO, the CV iterations z are handled separately, possibly assigned to different processes. Right: the SISSO implementation treats all CV splits, referring to the columns in the scheme, simultaneously instead. Parallelization is realized by feature index i, i.e. by rows. This transforms the learning to a multi-task problem $\mathscr{L}^{\mathrm{mt}}$.

C.3 Consistency Check for the SISSO Implementation

The consistency of our Python implementation with the original Fortran version of SISSO was checked for learning the equilibrium lattice constant a of $\mathbf{T}$. The used feature space was $\mathbf{X}_{a,2}$ (see Table 8.1) up to complexity $\mathscr{T}_1$, including also next-neighbor differences, HMs and the mathematical operations containing exp and log, leading to in total $M = 22960$ features. No hold-out data was considered, i.e. $N = 445$. The feature matrix and the target a were exported to a file `train.dat` (size 121 MiB) by our code and then read-in by the Fortran version. No further augmentation of the feature space by the FC module was performed. The DI module was run with the sparsifying operator SO=ℓ_0 (`method='L0'`), model selection was based on the error metric RMSE (`metric='LS_RMSE'`).

Single subset size was set to 6 (`subs_sis=6`), any other parameters were kept to their standard values.

In a correspondent way, descriptors up to $\Omega = 5$ were identified by both codes on a standard machine. Resulting component indices, RMSE and maxAE are reported in Table C.1. For up to $\Omega = 4$, identical descriptors are obtained. Only at $\Omega = 5$, solutions differ in 3 components. Calculated model performances both in terms of RMSE and maxAE are equal up to the level of mÅ. These minor differences seem to result from the 6 decimals limit in number precision in the transfer file `train.dat`. The results thus demonstrate that our implementation is consistent with the original version.

Python

Ω	components					RMSE	maxAE
1	6567					40.65	153.07
2	6567	15252				35.75	151.22
3	6566	8195	15252			32.19	149.28
4	6566	8195	15258	21624		30.20	154.96
5	6567	8195	15252	15258	19827	29.43	137.19

Fortran

Ω	components					RMSE	maxAE
1	6567					40.63	152.82
2	6567	15252				35.73	151.64
3	6566	8195	15252			32.20	149.54
4	6566	8195	15258	21624		30.21	154.92
5	6566	8195	15258	19591	21624	28.98	128.38

Table C.1: Consistency check of the Python SISSO implementation (top) with the original Fortran code (bottom). Reported are component indices of descriptors up to Ω=5, RMSE and maxAE (in mÅ). Learning task: a of $\mathbf{T}$ with $M = 22960$ features up to complexity $\mathscr{T}_1$.

C.4 Profiling of Parallelization

In order to analyze the benefit made by the parallelized subroutines of the developed code we denote these by

- P_1**:** generation of the feature space represented by $\mathbf{X}$,
- P_2**:** preparation of data matrices $\mathbf{X}$ (standardization and check-up for ill-defined columns),
- P_3**:** CV procedure to find the candidate descriptors $\{\mathscr{D}^{(j)}\}$ by $\mathscr{S}^{\mathrm{CV}}_{\mathrm{tr/te}}$,
- P_4**:** calculation of the correlations ρ in the SISSO approach.

For these, tests were run for which we analyzed, on the one hand, the *speed-up*

$$\eta \equiv \frac{T_{\mathrm{ex}}(1)}{T_{\mathrm{ex}}(N_{\mathrm{p}})} \tag{C.1}$$

where $T_{\mathrm{ex}}(N_{\mathrm{p}})$ is the recorded wall clock time using N_{p} processes. On the other hand, we investigated the *overhead*

$$o \equiv \frac{N_{\mathrm{p}} \cdot T_{\mathrm{ex}}(N_{\mathrm{p}})}{T_{\mathrm{ex}}(1)} \tag{C.2}$$

that quantifies the increased computational effort introduced by the parallelization itself.

In the tests, the number of processes N_{p} was varied between 1 and 8 where the available number of processors, 4, was exceeded to check if a single process is requiring the full resources of one core (disregarding other limitations such as RAM or speed of hard disk) - if so, the speed-up η will decrease after $N_{\mathrm{p}} = 4$ when two processes share a processor. The results of these tests are shown in Fig. C.3 where in the top part η is plotted vs. N_{p} and in the bottom part o vs. N_{p}. Subroutines P_2 and P_4 reach maximum speed-up for $N_{\mathrm{p}} = 4$, P_1 and P_3 stop increasing after that value. As explained above, this indicates that P_2 and P_4 mainly require CPU resources. In any case, the parallelization is able to improve the efficiency by factors between $\eta = 2.5$ and $\eta = 3$. The parallelization overhead o increases as expected markedly above $N_{\mathrm{proc}} = 4$ because in this case a single processor must manage several processes.

Figure C.4 visualizes the speed-up η vs. N_{p} for the total run of the code, i.e. parallelized and serial parts, for the same benchmarking task. Either the dimensionality reduction method $\mathscr{M}$=LASSO-LARS (red line) or SISSO (green line) was used for an additional comparison. For both methods, the maximum η is reached at $N_{\mathrm{p}} = 4$ (dashed lines). We find that the learning by SISSO improves the speed-up significantly from $\eta \approx 4/3$ at LASSO-LARS to $\eta \approx 2.0$.

It is also interesting to ask for the code's portion p that can benefit from parallelization. This is estimated through Amdahl's law [150]

$$\eta \to \frac{1}{1-p}. \tag{C.3}$$

Here, p is measured in the execution time the parallelizable subroutines require on a single processor, and an infinite parallelization is assumed, i.e. $N_{\mathrm{p}} \to \infty$. In approximation to that, p can be estimated from the maximum values of η in Fig. C.3, 4/3 and 2. This yields $p = 1/4$ for LASSO-LARS and $p = 1/2$ for SISSO, respectively, which is as expected since only SISSO intrinsically is parallelized.

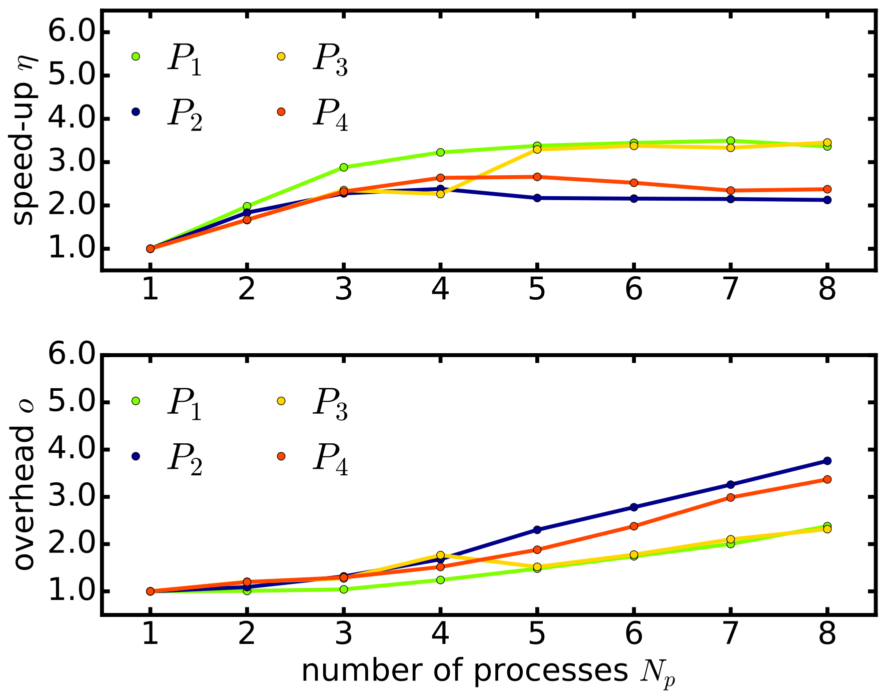


Figure C.3: Profiling of the parallelized subroutines P_1, P_2, P_3 and P_4. The top plot shows speed-up η versus number of processes N_p, the bottom one overhead o versus N_p. Tests were run on a quadcore machine with Intel Xeon 3.30 GHz processors and 16 GB RAM. The considered learning task is the prediction of E_mix of the ternaries $\mathbf{T}$ by the feature space $\mathbf{X}_{E,2}$ of complexity $\mathscr{T}_1$ (8726 features), using $\mathscr{M}$ =SISSO (implying that the subroutine P_3 is parallelized by feature index, as described in Sec. 6.3.3).

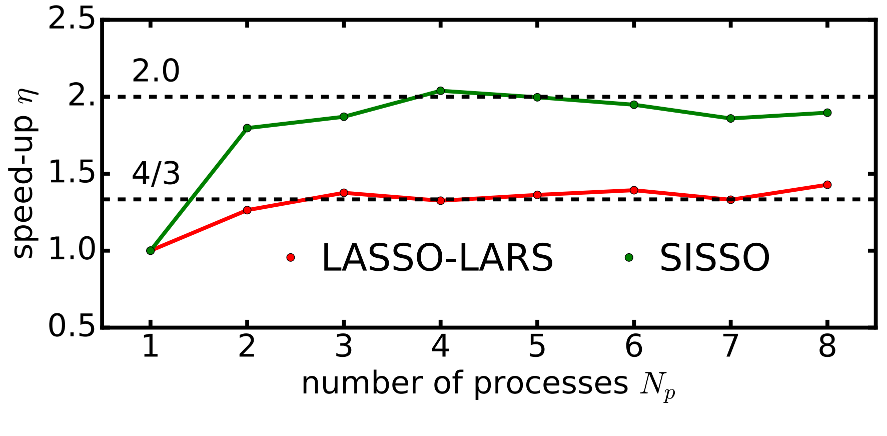


Figure C.4: Speed-up η of the total code vs. number of processes N_p. Feature selection is either performed by LASSO-LARS (red) or by SISSO (green). Black lines mark the achievable improvement in performance for $N_\mathrm{procs} = 4$, $\eta \approx 4/3$ for LASSO-LARS and $\eta \approx 2$ for SISSO. The tests were run on a quadcore machine with Intel i5 1.6 GHz processors and 8 GB RAM (different from above). Details on the learning task are given in the caption of Fig. C.3.

C.5 Runtimes of the two LASSO solvers

Learning Task	Size of **X**	coord. desc.	LARS
E_{mix}, $\mathrm{X}_{e,2}$	8726	94.04 s	6.06 s
a, $\mathrm{X}_{a,2}$	5257	311.74 s	11.43 s

Table C.2: Comparison of run times (in s) of feature selection with LASSO coordinate descent and LASSO-LARS. Runs were performed on a standard notebook computer, and the two different algorithms use the parametrization generally applied in this project.

Appendix D

Additional Information for ML on the Dataset O

Learning of Lattice Constant a_{ZB}

	components				
Ω	1	2	3	4	5
1	d_{XY}				
2	$r_s(A)+d_{XY}$	$\sqrt{r_s(B)+d_{XY}}$			
3	d_{XY}^2	$\sqrt[3]{r_s(A)+r_s(B)}$	$\frac{1}{Z(A)+Z(B)}$		
4	$r_s(A)+d_{XY}$	$r_s(B)+d_{XY}$	$\log(\lvert Z(A)+Z(B)\rvert)$	$\lvert r_s(A)-d_{XY}\rvert^3$	
5	$r_s(A)+d_{XY}$	$r_s(B)+d_{XY}$	$\log(\lvert Z(A)+Z(B)\rvert)$	$\exp(-Z(A))$	$\lvert r_s(A)-d_{XY}\rvert^3$

Table D.1: Components for descriptors up to $\Omega = 5$ for predicting the lattice constant a_{ZB} of the octet binary data-set. Descriptors are obtained on the total dataset by $\mathscr{L}$ =LASSO-LARS+ℓ_0 with $\mathscr{S}_{\text{all}}$.

	RMSE [Å]			MAE [Å]			maxAE [Å]			CV
Ω	$\mathscr{S}_{\text{all}}$	SC_{tr}	SC_{te}	$\mathscr{S}_{\text{all}}$	SC_{tr}	SC_{te}	all	SC_{tr}	SC_{te}	#
1	0.225	0.226	0.217	0.166	0.167	0.165	0.700	0.669	0.451	50
2	0.109	0.109	0.109	0.088	0.088	0.088	0.312	0.302	0.214	38
3	0.083	0.083	0.083	0.071	0.070	0.070	0.154	0.219	0.155	0
4	0.065	0.064	0.072	0.050	0.049	0.056	0.176	0.173	0.139	34
5	0.059	0.058	0.068	0.046	0.045	0.053	0.150	0.148	0.133	9

Table D.2: Performance measures for the descriptors for $a_{\mathbf{ZB}}$ from Tab. D.1. Error metrics RMSE, MAE and maxAE are reported for the fit ($\mathscr{S}_{\text{all}}$) as well as averages from a sanity check (SC, see Sec. 5.4.1, $N_{\text{CV}} = 50$). Additionally, the last column lists the counts from a L10%OCV with $N_{\text{CV}} = 50$ in which these descriptors are identified (by minimum training error). Lower values of test maxAE are expectable as explained in Sec. A.2.3.

Ω	1	2	3	4	5
N_{f_k}	1	3	5	5	5
N_{op}	0	4	7	9	11

Table D.3: Complexity measures N_{f_k} (informational) and N_{op} (algebraic) of the descriptors for predicting the lattice constant a_{ZB} of the binary data set **O** (for the definition of the complexity measures see Sec. 5.5.2).

Learning of TB Gap at Γ, $E_g^{\Gamma,\mathrm{TB}}$ from Feature Space $\mathbf{X}_{g,1}$

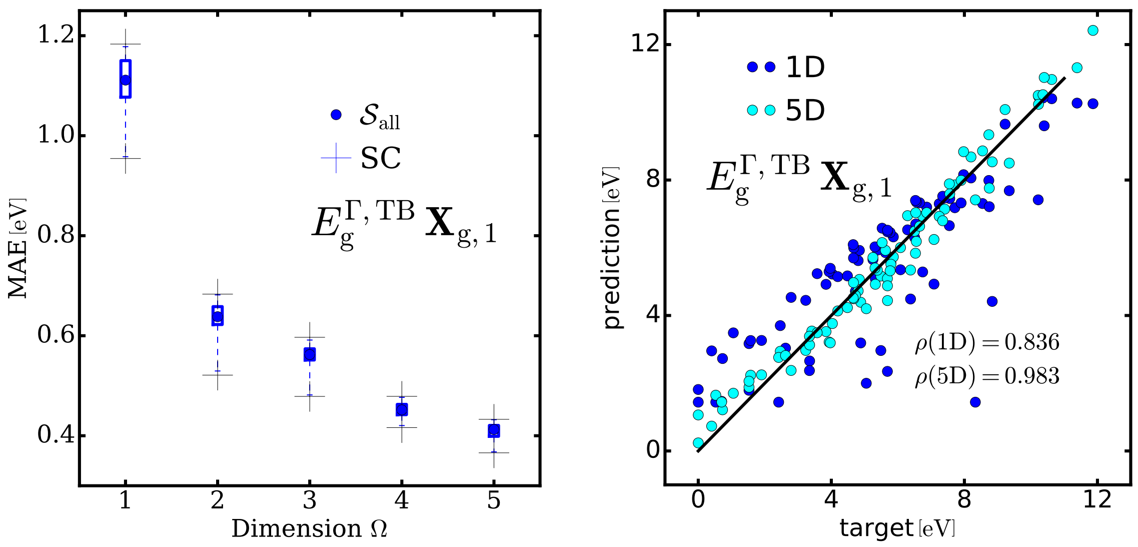


Figure D.1: Performance of the descriptors for predicting the TB band gap of the binary data set with $\mathbf{X}_{g,1}$, analogous to Fig. 8.19. Left: MAE versus descriptor dimension Ω. Right: correlation plot (prediction vs. target) for the 1D and the 5D descriptor including the Pearson correlation with the target.

	components				
Ω	1	2	3	4	5
1	$\bar{E}_s^{OS,2}$				
2	ΔE_s^{OS}	$\tilde{P}^3$			
3	ΔE_s^{OS}	$\tilde{P}^3$	E_{XY}^0		
4	$\tilde{Z}^3$	$\bar{\Delta} g^2$	IP	E_{XY}^0	
5	ΔE_s^{OS}	$\tilde{P}^3$	$\tilde{IP}^2$	E_{XY}^0	Πg

Table D.4: Components for descriptors up to $\Omega = 5$ for predicting TB band gaps of the octet binary data-set using $\mathbf{X}_{g,1}$ ($N = 138$, used learning method $\mathscr{L}$ =LASSO-LARS+ℓ_0, used strategy $\mathscr{S}_{\mathrm{all}}$).

Learning of TB Gap at Γ, $E_g^{\Gamma,\mathrm{TB}}$ from Feature Space $\mathbf{X}_{g,2}$

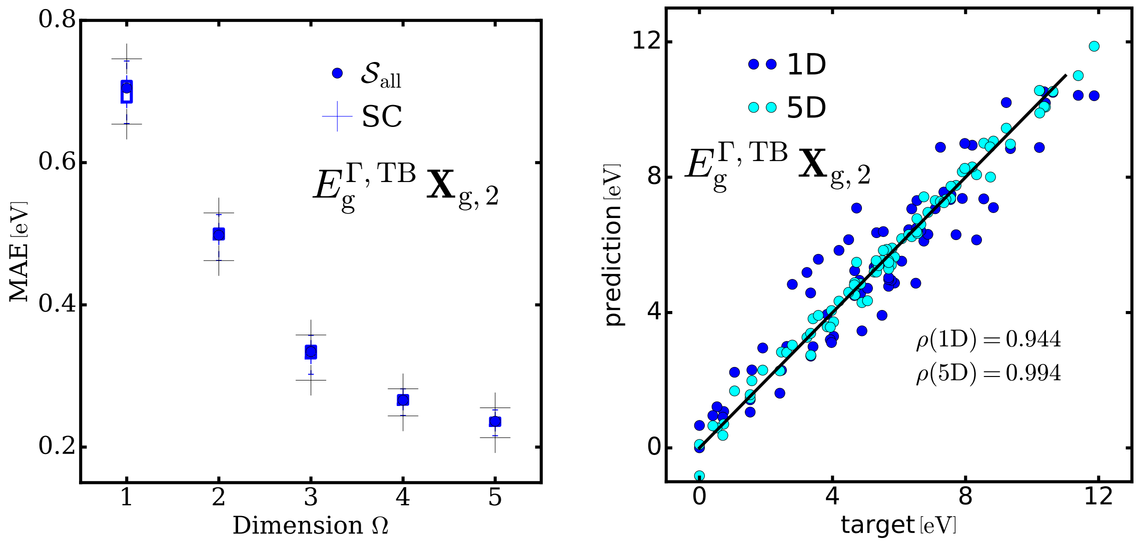


Figure D.2: Performance of the descriptors for predicting the TB band gap of the binary data set with $\mathbf{X}_{g,2}$, analogous to Fig. 8.19. Left: MAE versus descriptor dimension Ω. Right: correlation plot (prediction vs. target) for the 1D and the 5D descriptor and Pearson correlation.

	components				
Ω	1	2	3	4	5
1	$\frac{1}{\sqrt{\bar{\Delta} IP^2 + \Pi IP}}$				
2	$\frac{1}{\left(\bar{E}_s^{OS,2} + \bar{E}_{XY}^{0,2}\right)^3}$	$\sqrt{\bar{\Delta} H^2 + \Pi IP}$			
3	$\frac{1}{\left(\bar{E}_s^{OS,2} + \bar{E}_{XY}^{0,2}\right)^3}$	$\log^{-1}\left\lvert\bar{\Delta} IP^2 + \Pi IP\right\rvert$	$\frac{1}{\Pi P + \Pi g}$		
4	$\frac{1}{\left(\bar{E}_s^{OS,2} + \bar{E}_{XY}^{0,2}\right)^3}$	$\log\left\lvert\bar{\Delta} H^2 + \Pi H\right\rvert$	$\frac{1}{\sqrt[3]{\Pi P + \Pi g}}$	$(P - g)^2$	
5	$\frac{1}{\left(\bar{E}_s^{OS,2} + \bar{E}_{XY}^{0,2}\right)^3}$	$\log\left\lvert\bar{\Delta} H^2 + \Pi H\right\rvert$	$\frac{1}{\Pi P + \Pi g}$	$(P - g)^2$	$\frac{1}{\sqrt{\Pi P + \Pi g}}$

Table D.5: Components for descriptors up to $\Omega = 5$ for predicting TB band gaps of the octet binary data-set using $\mathbf{X}_{g,2}$ ($N = 49121$, used learning method $\mathscr{L}$ =LASSO-LARS+ℓ_0, used strategy $\mathscr{S}_{\mathrm{all}}$).

Learning of LDA Gap at Γ, $E_{\mathrm{g}}^{\Gamma,\mathrm{LDA}}$, from Atomic and Pair Data $\mathbf{X}_{\mathrm{g},1}$ (Complexity $\mathscr{T}_0$)

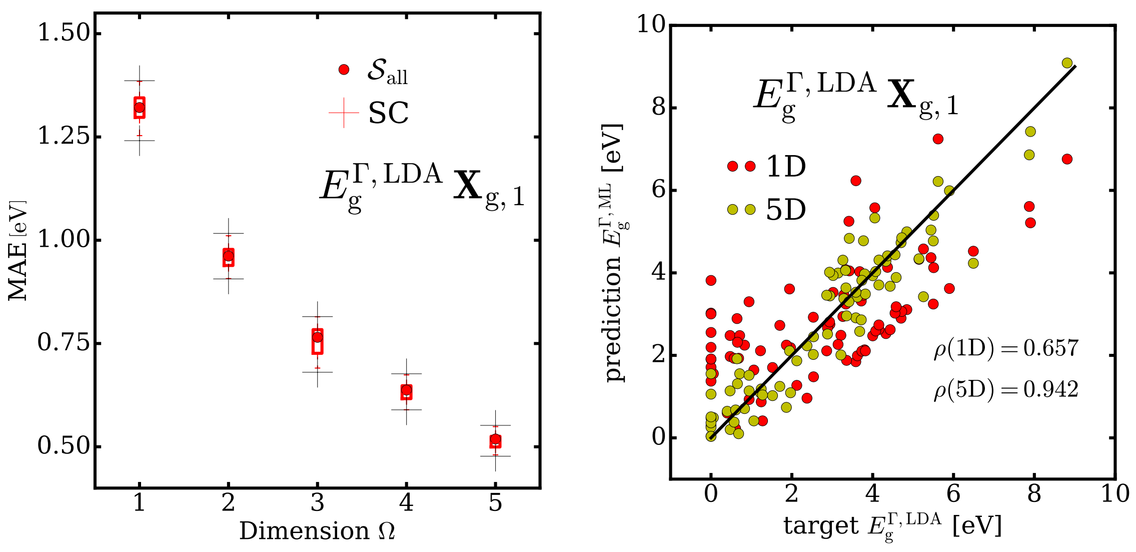


Figure D.3: Performance of the descriptors for predicting the LDA band gap $E_{\mathrm{gap}}^{\Gamma,LDA}$ of the binary data set with $\mathbf{X}_{\mathrm{g},1}$, similar to Fig. 8.19.

	components				
Ω	1	2	3	4	5
1	E^0_{XY}				
2	Πr_s	$\tilde{Z}^2$			
3	E^0_{XY}	$\tilde{Z}^2$	ΠE^s_{OS}		
4	E^0_{XY}	$\bar{\Delta} d^3_{XX}$	ΠE^s_{OS}	ΠEA	
5	E^0_{XY}	$\bar{\Delta} d^3_{XX}$	ΠE^s_{OS}	ΠEA	$\bar{E}^{g,2}_{XX}$

Table D.6: Components for descriptors up to $\Omega = 5$ for predicting the LDA band gap $E_{\mathrm{gap}}^{\Gamma,LDA}$ of $\mathbf{O}$ from $\mathbf{X}_{\mathrm{g},1}$ ($N = 138$, used learning method $\mathscr{L}$ =LASSO-LARS+ℓ_0, used strategy $\mathscr{S}_{\mathrm{all}}$).

Learning of LDA Gap at Γ, $E_{\mathrm{g}}^{\Gamma,\mathrm{LDA}}$, from Atomic and Pair Data $\mathbf{X}_{\mathrm{g},2}$ (Complexity $\mathscr{T}_1$)

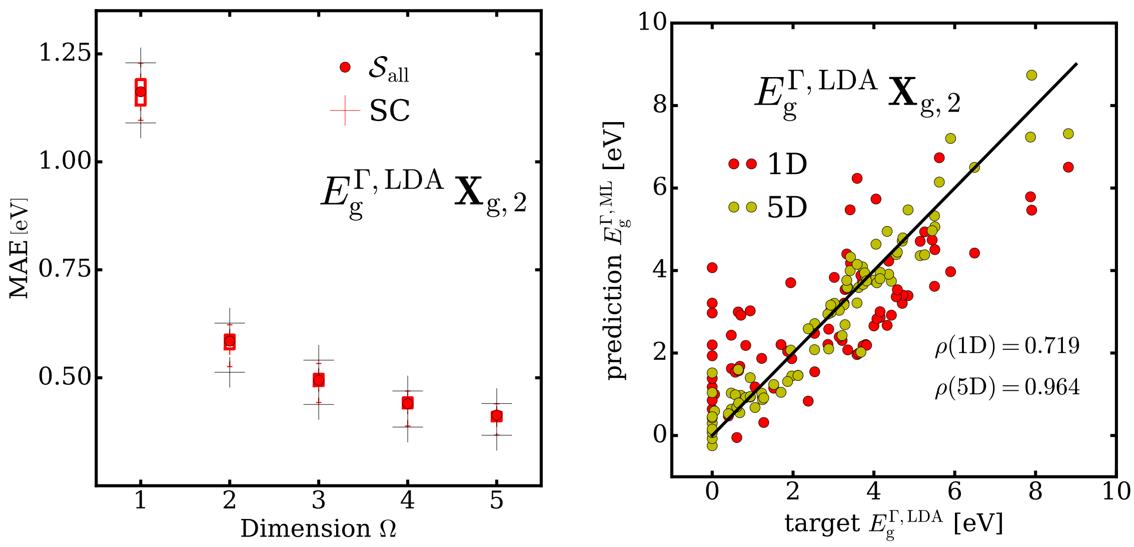


Figure D.4: Performance of the descriptors for predicting the LDA band gap $E_{\mathrm{gap}}^{\Gamma,\mathrm{LDA}}$ of the binary data set with $\mathbf{X}_{\mathrm{g},2}$, similar to Fig. 8.19.

components				
1	2	3	4	5
$\log(\bar{E}A^2 + E_{XY}^{0,2})$				
$\frac{1}{\sqrt{\lvert\Pi Z - \Pi P\rvert}}$	$\frac{1}{\sqrt{\bar{\Delta} E_{XX}^{g,2} - \Pi E_{OS}^{s}}}$			
$\frac{1}{\sqrt{\lvert\Pi Z - \Pi P\rvert}}$	$\frac{1}{\sqrt[3]{\bar{\Delta} E_{XX}^{g,2} - \Pi E_{OS}^{s}}}$	$\log(\bar{E}A^2 + E_{XY}^{0,2})$		
$\frac{1}{\sqrt{\lvert\Pi Z - \Pi P\rvert}}$	$\frac{1}{\sqrt[3]{\bar{\Delta} E_{XX}^{g,2} - \Pi E_{OS}^{s}}}$	$\log(\bar{E}A^2 + E_{XY}^{0,2})$	$(\Pi H - \Pi E_{OS}^{p})^3$	
$\frac{1}{\sqrt{\lvert\Pi Z - \Pi P\rvert}}$	$\frac{1}{\sqrt[3]{\bar{\Delta} E_{XX}^{g,2} - \Pi E_{OS}^{s}}}$	$\sqrt{\bar{\Delta} EA^2 + E_{XY}^{0,2}}$	$\exp(E_{XY}^{0} - H)$	$\exp(\bar{\Delta} E_{XX}^{g,3} - \bar{\Delta} EA^3)$

Table D.7: Components for descriptors up to $\Omega = 5$ for predicting the LDA band gap $E_{\mathrm{gap}}^{\Gamma,LDA}$ of **O** from $\mathbf{X}_{\mathrm{g},2}$ ($N = 49121$, used learning method $\mathscr{L}$ =LASSO-LARS+ℓ_0, used strategy $\mathscr{S}_{\mathrm{all}}$).

Learning of LDA Gap at Γ, $E_g^{\Gamma,\mathrm{LDA}}$, from TB Data

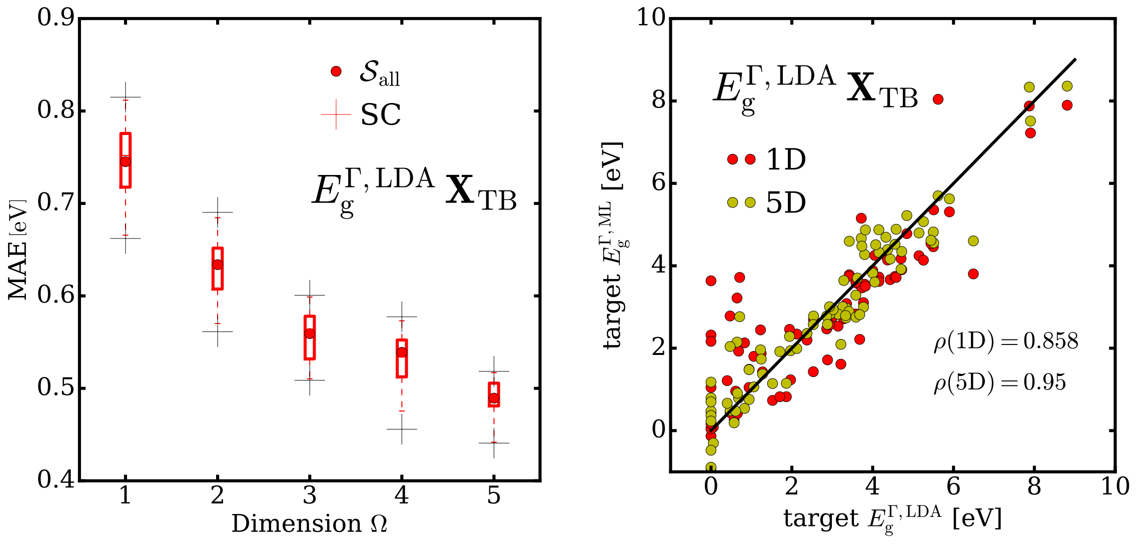


Figure D.5: Performance of the descriptors for predicting the LDA band gap $E_{\mathrm{gap}}^{\Gamma,LDA}$ of **O** from pure TB data $\mathbf{X}_{\mathrm{TB}}$, showing the MAE and its statistics from $\mathscr{S}_{\mathrm{all}}$ (left) and the correlation plot (right).

	components				
Ω	1	2	3	4	5
1	$E_{\mathrm{CB}}^{\mathrm{L}} + E_{\mathrm{CB}}^{\mathrm{X}}$				
2	$E_{\mathrm{CB}}^{\mathrm{L}} + E_{\mathrm{CB}}^{\mathrm{X}}$	$\exp\left(-E_{\mathrm{VB}}^{\mathrm{L}} - E_{\mathrm{CB}}^{\mathrm{L}}\right)$			
3	$\frac{1}{(E_{\mathrm{VB}}^{\Gamma}+E_{\mathrm{CB}}^{\Gamma})^2}$	$E_{\mathrm{CB}}^{\mathrm{X}}$	$\frac{1}{E_{\mathrm{VB}}^{\mathrm{X}}-E_{\mathrm{VB}}^{\mathrm{U/K}}}$		
4	$\frac{1}{(E_{\mathrm{VB}}^{\Gamma}+E_{\mathrm{CB}}^{\Gamma})^2}$	$E_{\mathrm{CB}}^{\mathrm{X}}$	$\frac{1}{E_{\mathrm{VB}}^{\mathrm{X}}-E_{\mathrm{VB}}^{\mathrm{U/K}}}$	$\frac{1}{E_{\mathrm{CBM}}+E_{\mathrm{CB}}^{\mathrm{U/K}}}$	
5	$E_{\mathrm{CB}}^{\mathrm{L}} + E_{\mathrm{CB}}^{\mathrm{X}}$	$E_{\mathrm{CB}}^{\mathrm{X}}$	$\frac{1}{E_{\mathrm{VB}}^{\mathrm{X}}-E_{\mathrm{VB}}^{\mathrm{U/K}}}$	$*\sqrt[3]{E_{\mathrm{CBM}}^{3} + E_{\mathrm{CB}}^{\mathrm{U/K},3}}$	$(E_{\mathrm{VB}}^{\Gamma} + E_{\mathrm{CB}}^{\mathrm{L}})^2$

Table D.8: Components for descriptors up to $\Omega = 5$ for predicting the LDA band gap $E_{\mathrm{gap}}^{\Gamma,LDA}$ of **O** from pure TB data, $\mathbf{X}_{\mathrm{TB}}$ ($N = 3959$, used learning method $\mathscr{L}$ =LASSO-LARS+ℓ_0, used strategy $\mathscr{S}_{\mathrm{all}}$).

Bibliography

[1] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler. Big Data of Materials Science: Critical Role of the Descriptor. *Phys. Rev. Lett.*, 114, 105503, 2015.

[2] L. M. Ghiringhelli, J. Vybiral, E. Ahmetcik, R. Ouyang, S. V. Levchenko, C. Draxl, and M. Scheffler. Learning Physical Descriptors for Materials Science by Compressed Sensing. *New J. Phys.*, 19, 023017, 2017.

[3] P. Hohenberg and W. Kohn. Inhomogeneous Electron Gas. *Phys. Rev.*, 136, B864, 1964.

[4] W. Kohn and L. J. Sham. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys. Rev.*, 140, A1133, 1965.

[5] C. Draxl and M. Scheffler. NOMAD: The FAIR Concept for Big Data-Driven Materials Science. *MRS BULLETIN*, 43, 2018.

[6] C. Draxl and M. Scheffler. Big-Data-Driven Materials Science and its FAIR Data Infrastructure. In *Handbook of Materials Modeling*, pages 1–25. Springer, 2019. doi:10.1007/978-3-319-42913-7_104-1.

[7] T. Mueller, A. G. Kusne, and R. Ramprasad. Machine Learning in Materials Science: Recent Progress and Emerging Applications. *Rev. Comput. Chem.*, 29, 186–273, 2016.

[8] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim. Machine Learning in Materials Informatics: Recent Applications and Prospects. *npj Comput. Mater.*, 3, 1–13, 2017.

[9] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques. Recent Advances and Applications of Machine Learning in Solid-State Materials Science. *npj Comput. Mater.*, 5, 1–36, 2019.

[10] A. Ziletti, D. Kumar, M. Scheffler, and L. M Ghiringhelli. Insightful Classification of Crystal Structures Using Deep Learning. *Nat. Commun.*, 9, 1–10, 2018.

[11] F. Musil, S. De, J. Yang, J. E. Campbell, G. M. Day, and M. Ceriotti. Machine Learning for the Structure-Energy-Property Landscapes of Molecular Crystals. *Chem. Sci.*, 9, 1289–1300, 2018.

[12] C. Sutton, L. M Ghiringhelli, T. Yamamoto, Y. Lysogorskiy, L. Blumenthal, T. Hammerschmidt, J. R. Golebiowski, X. Liu, A. Ziletti, and M. Scheffler. Crowd-Sourcing Materials-Science Challenges with the NOMAD 2018 Kaggle Competition. *npj Comput. Mater.*, 5, 1–11, 2019.

[13] R. Ouyang, E. Ahmetcik, C. Carbogno, M. Scheffler, and L. M Ghiringhelli. Simultaneous Learning of Several Materials Properties from Incomplete Databases with Multi-Task SISSO. *J. Phys.: Mater.*, 2, 024002, 2019.

[14] J. R. Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, volume 1. MIT press, 1992.

[15] Y. Wang, N. Wagner, and J. M. Rondinelli. Symbolic Regression in Materials Science. *MRS Communications*, 9, 793–805, 2019.

[16] H. Boche, R. Calderbank, G. Kutyniok, and J. Vybíral. A Survey of Compressed Sensing. In *Compressed Sensing and its Applications*, pages 1–39. Springer, 2015.

[17] L. J. Nelson, G. L. W. Hart, F. Zhou, V. Ozoliņš, et al. Compressive Sensing as a Paradigm for Building Physics Models. *Phys. Rev. B*, 87, 035125, 2013.

[18] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, 2, 1989.

[19] B.-L. Adam, Y. Qu, J. W. Davis, M. D Ward, M. A. Clements, L. H. Cazares, O. J. Semmes, P. F. Schellhammer, Y. Yasui, Z. Feng, et al. Serum Protein Fingerprinting Coupled With a Pattern-Matching Algorithm Distinguishes Prostate Cancer from Benign Prostate Hyperplasia and Healthy Men. *Cancer Research*, 62, 3609–3614, 2002.

[20] T. S. Guzella and W. M. Caminhas. A Review of Machine Learning Approaches to Spam Filtering. *Expert Systems with Applications*, 36, 10206–10222, 2009.

[21] B. Huang, Y. Huan, L. D. Xu, L. Zheng, and Z. Zou. Automated Trading Systems Statistical and Machine Learning Methods and Hardware Implementation: a Survey. *Enterprise Information Systems*, 13, 132–144, 2019.

[22] A. L. Fradkov. Early History of Machine Learning. *IFAC-PapersOnLine*, 53, 1385–1390, 2020.

[23] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *nature*, 529, 484–489, 2016.

[24] M. Erdmann, J. Glombitza, G. Kasieczka, and U. Klemradt. *Deep Learning for Physics Research*. World Scientific, 2021.

[25] C. Hayashi. What is Data Science? Fundamental Concepts and a Heuristic Example. In *Data Science, Classification, and Related Methods*, pages 40–51. Springer, 1998. doi:10.1007/978-4-431-65950-1_3.

[26] L. Van Der Maaten, E. Postma, J. Van den Herik, et al. Dimensionality Reduction: a Comparative Review. *J. Mach. Learn. Res.*, 10, 13, 2009.

[27] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer series in statistics New York, 2001.

[28] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[29] J. E. Gentle. *Matrix Algebra*, volume 10. Springer, 2007.

[30] S. Arora and B. Barak. *Computational Complexity: a Modern Approach.* Cambridge University Press, 2009.

[31] Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, 1996.

[32] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *J. R. Stat. Soc. Series B*, 58, 267–288, 1996.

[33] A. Cohen, W. Dahmen, and R. DeVore. Compressed Sensing and Best k-term Approximation. *J. Am. Math. Soc.*, 22, 211–231, 2009.

[34] S. J. Wright. Coordinate Descent Algorithms. *Mathematical Programming*, 151, 3–34, 2015.

[35] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least Angle Regression. *Ann. Statist.*, 32, 407–499, 2004.

[36] S. G. Mallat and Z. Zhang. Matching Pursuits with Time-Frequency Dictionaries. *IEEE Transactions on Signal Processing*, 41, 3397–3415, 1993.

[37] Y. Chandra Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993. doi:10.1109/ACSSC.1993.342465.

[38] J. A. Tropp and A. C. Gilbert. Signal Recovery from Random Measurements via Orthogonal Matching Pursuit. *IEEE Transactions on information theory*, 53, 4655–4666, 2007.

[39] J. Fan and J. Lv. Sure Independence Screening for Ultrahigh Dimensional Feature Space. *J. R. Stat. Soc. B*, 70, 849–911, 2008.

[40] T. Mueller, E. Johlin, and J. C. Grossman. Origins of Hole Traps in Hydrogenated Nanocrystalline and Amorphous Silicon Revealed Through Machine Learning. *Phys. Rev. B*, 89, 115202, 2014.

[41] M. Rupp. Machine Learning for Quantum Mechanics in a Nutshell. *International Journal of Quantum Chemistry*, 115, 1058–1073, 2015.

[42] J. Mercer. Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations. *Philosophical Transactions of the Royal Society of London A*, 209, 415–446, 1909.

[43] M. Kommenda. *Local Optimization and Complexity Control for Symbolic Regression.* PhD thesis, Universität Linz, 2018. URL https://epub.jku.at/obvulihs/content/pageview/2724901.

[44] K. Rodriguez-Vazquez, C. M. Fonseca, and P. J. Fleming. Multiobjective Genetic Programming: A Nonlinear System Identification Application. In *Late Breaking Papers at the 1997 Genetic Programming Conference*, pages 207–212. Morgan Kaufmann Publishers, 1997. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.47.99&rep=rep1&type=pdf.

[45] E. J. Vladislavleva, G. F. Smits, and D. Den Hertog. Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming. *IEEE Transactions on Evolutionary Computation*, 13, 333–349, 2008.

[46] R. J. Tibshirani and B. Efron. *An Introduction to the Bootstrap*, volume 57. Chapman and Hall New York, 1993.

[47] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*, volume 112. Springer, 2013.

[48] K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. Von Lilienfeld, A. Tkatchenko, and K.-R. Müller. Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies. *J. Chem. Theory Comput.*, 9, 3404–3419, 2013.

[49] J. M. Sanchez, F. Ducastelle, and D. Gratias. Generalized Cluster Description of Multicomponent Systems. *Physica A*, 128, 334–350, 1984.

[50] S. Arlot and A. Celisse. A Survey of Cross-Validation Procedures for Model Selection. *Statistics Surveys*, 4, 40–79, 2010.

[51] L. Breiman and P. Spector. Submodel Selection and Evaluation in Regression. The X-random Case. *International Statistical Review*, 60, 291–319, 1992.

[52] H.-J. Lu, N. Zou, R. Jacobs, B. Afflerbach, X.-G. Lu, and D. Morgan. Error Assessment and Optimal Cross-Validation Approaches in Machine Learning Applied to Impurity Diffusion. *Comput. Mater. Sci.*, 169, 109075, 2019.

[53] M. Born and R. Oppenheimer. Zur Quantentheorie der Molekeln. *Annalen der Physik*, 389, 457–484, 1927.

[54] K. Burke. Perspective on Density Functional Theory. *The Journal of Chemical Physics*, 136, 150901, 2012.

[55] J. P. Perdew and K. Schmidt. Jacob's Ladder of Density Functional Approximations for the Exchange-Correlation Energy. In *AIP Conference Proceedings*, volume 577, pages 1–20. American Institute of Physics, 2001. doi:10.1063/1.1390175.

[56] R. Parr and Y. Weitao. *Density Functional Theory of Atoms and Molecules*, volume 2. Oxford University Press, 1994.

[57] P. A. M. Dirac. Note on Exchange Phenomena in the Thomas Atom. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 26, pages 376–385. Cambridge University Press, 1930. doi:10.1017/S0305004100016108.

[58] N. W. Ashcroft, N. D. Mermin, et al. *Solid State Physics*. New York: Holt, Rinehart and Winston,, 1976.

[59] D. M. Ceperley and B. J. Alder. Ground State of the Electron Gas by a Stochastic Method. *Phys. Rev. Lett.*, 45, 566, 1980.

[60] S. H. Vosko, L. Wilk, and M. Nusair. Accurate Spin-Dependent Electron Liquid Correlation Energies for Local Spin Density Calculations: a Critical Analysis. *Canadian Journal of Physics*, 58, 1200–1211, 1980.

[61] J. P. Perdew and A. Zunger. Self-interaction Correction to Density-Functional Approximations for Many-Electron Systems. *Phys. Rev. B*, 23, 5048–5079, May 1981.

[62] J. P. Perdew and Y. Wang. Accurate and Simple Analytic Representation of the Electron-Gas Correlation Energy. *Phys. Rev. B*, 45, 13244, 1992.

[63] W. Kohn. Nobel Lecture: Electronic Structure of Matter—Wave Functions and Density Functionals. *Rev. Mod. Phys.*, 71, 1253, 1999.

[64] R. M. Martin, L. Reining, and D. M. Ceperley. *Interacting Electrons*. Cambridge University Press, 2016.

[65] F. Giustino. *Materials Modelling Using Density Functional Theory: Properties and Predictions*. Oxford University Press, 2014.

[66] A. Gulans, S. Kontur, C. Meisenbichler, D. Nabok, P. Pavone, S. Rigamonti, S. Sagmeister, U. Werner, and C. Draxl. Exciting: a Full-Potential All-Electron Package Implementing Density-Functional Theory and Many-Body Perturbation Theory. *J. Phys. Condens. Matter*, 26, 363202, 2014.

[67] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, and M. Scheffler. Ab Initio Molecular Simulations With Numeric Atom-Centered Orbitals. *Comp. Phys. Commun.*, 180, 2175–2196, 2009.

[68] R. M. Martin and R. M. Martin. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004.

[69] O. K. Andersen. Linear Methods in Band Theory. *Phys. Rev. B*, 12, 3060, 1975.

[70] A. Gulans, A. Kozhevnikov, and C. Draxl. Microhartree Precision in Density Functional Theory Calculations. *Phys. Rev. B*, 97, 161105, 2018.

[71] D. Nabok, A. Gulans, and C. Draxl. Accurate All-Electron G_0W_0 Quasiparticle Energies Employing the Full-Potential Augmented Plane-Wave Method. *Phys. Rev. B*, 94, 035118, 2016.

[72] Online Documentation of `exciting`, 2021. URL http://exciting.wikidot.com/documentation.

[73] J. P. Perdew, R. G. Parr, M. Levy, and J. L. Balduz Jr. Density-Functional Theory for Fractional Particle Number: Derivative Discontinuities of the Energy. *Phys. Rev. Lett.*, 49, 1691-1694, 1982.

[74] E. Trushin, M. Betzinger, S. Blügel, and A. Görling. Band Gaps, Ionization Potentials, and Electron Affinities of Periodic Electron Systems via the Adiabatic-Connection Fluctuation-Dissipation Theorem. *Phys. Rev. B*, 94, 075123, 2016.

[75] J. P. Perdew and M. Levy. Physical Content of the Exact Kohn-Sham Orbital Energies: Band-Gaps and Derivative Discontinuities. *Phys. Rev. Letters*, 51, 1884-1887, 1983.

[76] J. P. Perdew and M. Levy. Comment on “Significance of the Highest Occupied Kohn-Sham Eigenvalue”. *Phys. Rev. B*, 56, 16021, 1997.

[77] P. Politzer and F. Abu-Awwad. A Comparative Analysis of Hartree-Fock and Kohn-Sham Orbital Energies. *Theoretical Chemistry Accounts*, 99, 83–87, 1998.

[78] J.-M. Jancu, R. Scholz, F. Beltram, and F. Bassani. Empirical spds* Tight-Binding Calculation for Cubic Semiconductors: General Method and Material Parameters. *Phys. Rev. B*, 57, 6493, 1998.

[79] A. T. Paxton et al. An Introduction to the Tight Binding Approximation - Implementation by Diagonalisation. *NIC Series*, 42, 145–176, 2009.

[80] T. Wendav. *Modeling of SiGeSn-based Semiconductor Heterostructures for Optoelectronic Applications.* PhD thesis, 2017.

[81] A. Urban. *Environment-Dependent Crystal-Field Tight-Binding Based on Density-Functional Theory.* PhD thesis, 2012. URL https://inis.iaea.org/collection/NCLCollectionStore/_Public/45/031/45031624.pdf.

[82] Y. Liu, T. Zhao, W. Ju, and S. Shi. Materials Discovery and Design Using Machine Learning. *Journal of Materiomics*, 3, 159–177, 2017.

[83] G. Pilania and X.-Y. Liu. Machine Learning Properties of Binary Wurtzite Superlattices. *J. Mat. Sci.*, 53, 6652–6664, 2018.

[84] J. Lee, A. Seko, K. Shitara, K. Nakayama, and I. Tanaka. Prediction Model of Band Gap for Inorganic Compounds by Combination of Density Functional Theory Calculations and Machine Learning Techniques. *Phys. Rev. B*, 93, 115104, 2016.

[85] F. Legrain, J. Carrete, A. van Roekeghem, S. Curtarolo, and N. Mingo. How Chemical Composition alone can predict Vibrational Free Energies and Entropies of Solids. *Chem. Mat.*, 29, 6220–6227, 2017.

[86] G. Pilania, C. Wang, X. Jiang, S. Rajasekaran, and R. Ramprasad. Accelerating Materials Property Predictions Using Machine Learning. *Sci. Rep.*, 3, 2013.

[87] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.*, 108, 058301, 2012.

[88] M. Brezocnik, M. Kovacic, and M. Ficko. Prediction of Surface Roughness with Genetic Programming. *Journal of Materials Processing Technology*, 157, 28–36, 2004.

[89] J. A. Van Vechten. Quantum Dielectric Theory of Electronegativity in Covalent Systems. I. Electronic Dielectric Constant. *Phys. Rev.*, 182, 891, 1969.

[90] J. C. Phillips. Ionicity of the Chemical Bond in Crystals. *Rev. Mod. Phys.*, 42, 317, 1970.

[91] R. Ouyang, S. Curtarolo, E. Ahmetcik, M. Scheffler, and L. M. Ghiringhelli. SISSO: A Compressed-Sensing Method for Identifying the Best Low-Dimensional Descriptor in an Immensity of Offered Candidates. *Phys. Rev. Materials*, 2, 083802, Aug 2018.

[92] C. M. Acosta, R. Ouyang, A. Fazzio, M. Scheffler, L. M Ghiringhelli, and C. Carbogno. Analysis of Topological Transitions in Two-Dimensional Materials by Compressed Sensing. *arXiv preprint arXiv:1805.10950*, 2018.

[93] B. Meredig, E. Antono, C. Church, M. Hutchinson, J. Ling, S. Paradiso, B. Blaiszik, I. Foster, B. Gibbons, J. Hattrick-Simpers, et al. Can Machine Learning Identify the Next High-Temperature Superconductor? Examining Extrapolation Performance for Materials Discovery. *Molecular Systems Design & Engineering*, 3, 819–825, 2018.

[94] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld. Big Data Meets Quantum Chemistry Approximations: the Δ-Machine Learning Approach. *J. Chem. Theory Comput.*, 11, 2087–2096, 2015.

[95] L. Foppa, M. Scheffler, and L. M Ghiringhelli. Materials-Genome (Descriptor) Identification by the Hierarchical SISSO Approach. In *Conference on a FAIR Data Infrastructure for Materials Genomics Conference on a FAIR Data Infrastructure for Materials Genomics*. FAIR-DI e.V., 2020. URL https://th.fhi-berlin.mpg.de/meetings/fairdi2020/custom/download.php?G=Meeting&F=poster_41.pdf.

[96] A. Chandrasekaran, D. Kamal, R. Batra, C. Kim, L. Chen, and R. Ramprasad. Solving The Electronic Structure Problem With Machine Learning. *npj Comput. Mater.*, 5, 1–7, 2019.

[97] C. Draxl and M. Scheffler. The NOMAD Laboratory: from Data Sharing to Artificial Intelligence. *Journal of Physics: Materials*, 2, 036001, 2019.

[98] J. J. De Pablo, N. E. Jackson, M. A Webb, L.-Q. Chen, J. E. Moore, D. Morgan, R. Jacobs, T. Pollock, D. G. Schlom, E. S Toberer, et al. New Frontiers for the Materials Genome Initiative. *npj Comput. Mater.*, 5, 1–23, 2019.

[99] M. F. Langer, A. Goeßmann, and M. Rupp. Representations of Molecules and Materials for Interpolation of Quantum-Mechanical Simulations via Machine Learning. *arXiv preprint arXiv:2003.12081*, 2020.

[100] A. F. Bialon, T. Hammerschmidt, and R. Drautz. Three-Parameter Crystal-Structure Prediction for sp-d-valent Compounds. *Chem. Mat.*, 28, 2550–2556, 2016.

[101] K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K.-R. Müller, and E. K. U. Gross. How to Represent Crystal Structures for Machine Learning: Towards Fast Prediction of Electronic Properties. *Phys. Rev. B*, 89, 205118, 2014.

[102] A. P. Bartók, R. Kondor, and G. Csányi. On Representing Chemical Environments. *Phys. Rev. B*, 87, 184115, 2013.

[103] H. Huo and M. Rupp. Unified Representation of Molecules and Crystals for Machine Learning. *arXiv preprint arXiv:1704.06439*, 2017.

[104] A. R. Denton and N. W. Ashcroft. Vegard's Law. *Phys. Rev. A*, 43, 3161–3164, Mar 1991.

[105] S. T. Murphy, A. Chroneos, C. Jiang, U. Schwingenschlögl, and R. W. Grimes. Deviations from Vegard's law in Ternary III-V Alloys. *Phys. Rev. B*, 82, 073201, 2010.

[106] D. J. Bartholomew, F. Steele, and I. Moustaki. *Analysis of Multivariate Social Science Data*. CRC press, 2008.

[107] B. Efron. *The Jackknife, the Bootstrap and other Resampling Plans*. SIAM, 1982.

[108] G. L. W. Hart, L. J. Nelson, and R. W. Forcade. Generating Derivative Structures at a Fixed Concentration. *Comput. Mater. Sci.*, 59, 101–107, 2012.

[109] B. Hoock. The supplementary data and the code are accessible and maintained online at this git repository, 2022. URL https://git.physik.hu-berlin.de/hoock/benedikt-hoock-phd-project-machine-learning.

[110] J. L. Bentley. A Survey of Techniques for Fixed Radius Near Neighbor Searching. Technical report, Stanford Linear Accelerator Center, Calif.(USA), 1975. URL https://inspirehep.net/files/4513dde7569f063490f9e610b911589b.

[111] Documentation of `multiprocessing`, 2021. URL https://docs.python.org/2.7/library/multiprocessing.html.

[112] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12, 2825–2830, 2011.

[113] Documentation of `sklearn.linear_model.LassoLars`, 2021. URL https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoLars.html#sklearn.linear_model.LassoLars.

[114] R. Ouyang, E. Ahmetcik, C. Carbogno, M. Scheffler, and L. M Ghiringhelli. Source Code of SISSO, 2018. URL github.com/rouyang2017/SISSO.

[115] F. D. Murnaghan. The Compressibility of Media Under Extreme Pressures. *Proc. Natl. Acad. Sci. U.S.A.*, 30, 244, 1944.

[116] R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

[117] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM J. Sci. Comput.*, 16, 1190–1208, 1995.

[118] Input Reference of the `exciting` Code, 2021. URL http://exciting.wikidot.com/ref:input.

[119] V. R. D'Costa, Y.-Y. Fang, J. Tolle, J. Kouvetakis, and J. Menendez. Tunable Optical Gap at a Fixed Lattice Constant in Group-IV Semiconductor Alloys. *Phys. Rev. Lett.*, 102, 107403, 2009.

[120] I. A. Fischer, T. Wendav, L. Augel, S. Jitpakdeebodin, F. Oliveira, A. Benedetti, S. S.v, S. Chiussi, G. Capellini, and K. Busch. Growth and Characterization of SiGeSn Quantum Well Photodiodes. *Optics Express*, 23, 25048–25057, 2015.

[121] C. I. Ventura, J. D. Querales Flores, J. D. Fuhr, and R. A. Barrio. Electronic Structure of $\mathrm{Ge}_{1-x-y}\mathrm{Si}_x\mathrm{Sn}_y$ Ternary Alloys for Multijunction Solar Cells. *Progress in Photovoltaics: Research and Applications*, 23, 112–118, 2015.

[122] S. Wirths, D. Buca, Z. Ikonic, P. Harrison, A. T. Tiedemann, B. Holländer, T. Stoica, G. Mussler, U. Breuer, J. M. Hartmann, et al. SiGeSn Growth Studies Using Reduced Pressure Chemical Vapor Deposition Towards Optoelectronic Applications. *Thin Solid Films*, 557, 183–187, 2014.

[123] Yamaha, T. and Nakatsuka, O. and Takeuchi, S. and Takeuchi, W. and Taoka, N. and Araki, K. and Izunome, K. and Zaima, S. Growth and characterization of heteroepitaxial layers of gesisn ternary alloy. *ECS Transactions*, 50, 907, 2013.

[124] J. Kouvetakis, J. Menendez, and A. V. G. Chizmeshya. Tin-Based Group IV Semiconductors: New Platforms for Opto- and Microelectronics on Silicon. *Annu. Rev. Mater. Res.*, 36, 497–554, 2006.

[125] G. L. W. Hart and R. W. Forcade. Algorithm for Generating Derivative Structures. *Phys. Rev. B*, 77, 224115, 2008.

[126] M. Troppenz, S. Rigamonti, and C.laudia Draxl. Predicting Ground-State Configurations and Electronic Properties of the Thermoelectric Clathrates $Ba_8Al_xSi_{46-x}$ and $Sr_8Al_xSi_{46-x}$. *Chem. Mat.*, 29, 2414–2424, 2017.

[127] W. S. Morgan, G. L. W. Hart, and R. W. Forcade. Generating Derivative Superstructures for Systems with High Configurational Freedom. *Comput. Mater. Sci.*, 136, 144–149, 2017.

[128] S. Adachi. *Properties of Semiconductor Alloys: Group-IV, III-V and II-VI Semiconductors*, volume 28. John Wiley & Sons, 2009.

[129] L. Vegard. Die Konstitution der Mischkristalle und die Raumfüllung der Atome. *Zeitschrift für Physik*, 5, 17–26, 1921.

[130] Aella, P. and Cook, C. and Tolle, J. and Zollner, S. and Chizmeshya, A. V. G. and Kouvetakis, J. Optical and Structural Properties of Si_x Sn_y Ge_{1-x-y} Alloys. *Appl. Phys. Lett.*, 84, 888–890, 2004.

[131] Online Ressource of `exciting` Beryllium, 2013. URL exciting.wdfiles.com/local--files/beryllium/exciting.beryllium.tar.gz.

[132] Ute Werner and C. Draxl. NOMAD Dataset of Calculations for 16-atomic Group-IV Zincblende Ternaries, 2015. URL https://dx.doi.org/10.17172/NOMAD/2021.03.02-1.

[133] M. J. D. Powell. An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives. *Comput. J.*, 7, 155–162, 1964.

[134] T. E. Oliphant. Python for Scientific Computing. *Computing in Science & Engineering*, 9, 10–20, 2007.

[135] E. R. Davidson. The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-Symmetric Matrices. *J. Comput. Phys.*, 17, 87 - 94, 1975. ISSN 0021-9991.

[136] J. A Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7, 308–313, 1965.

[137] A. V. G. Chizmeshya, M. R. Bauer, and J. Kouvetakis. Experimental and Theoretical Study of Deviations from Vegard's Law in the Sn_x Ge_{1-x} System. *Chem. Mat.*, 15, 2511–2519, 2003.

[138] Documentation of `sklearn.linear_model.LassoLars`, 2021. URL https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html#sklearn.kernel_ridge.KernelRidge.

[139] Werner U. Hannewald K. Ghiringhelli L. M. Scheffler M. Hoock, B. and C. Draxl. Machine Learning of Structural and Electronic Properties of Semiconductors. In *DPG Fr2016 - Verhandlungen*. Deutsche Physikalische Gesellschaft, 2016.

[140] NOMAD Dataset of Calculations for Octet Binary Materials, 2015. URL https://dx.doi.org/10.17172/NOMAD/2016.05.17-1.

[141] K. Hannewald. Private communications, 2015.

[142] P. Manca. A Relation Between the Binding Energy and the Band-Gap Energy in Semiconductors of Diamond or Zinc-Blende Structure. *Journal of Physics and Chemistry of Solids*, 20, 268–273, 1961.

[143] Brillioun Zone of the fcc Lattice - Applet in the Lecture Scriptum of Hadley, P., TU Graz, 2022. URL http://lampx.tugraz.at/~hadley/ss1/bzones/fcc.php.

[144] C. Nyshadham, M. Rupp, B. Bekker, A. V. Shapeev, T. Mueller, C. W. Rosenbrock, G. Csányi, D. W. Wingate, and G. L. W. Hart. Machine-Learned Multi-System Surrogate Models for Materials Prediction. *npj Comput. Mater.*, 5, 1–6, 2019.

[145] L. Freijeiro-González, M. Febrero-Bande, and W. González-Manteiga. A Critical Review of LASSO and Its Derivatives for Variable Selection Under Dependence Among Covariates. *International Statistical Review*, 2021.

[146] C. J. Willmott and K. Matsuura. Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance. *Climate Research*, 30, 79–82, 2005.

[147] W. Demtröder. *Molecular Physics: Theoretical Principles and Experimental Methods*. John Wiley & Sons, 2008.

[148] L. M. Ghiringhelli. Additional Data from the Calculations of Ghiringhelli et. al., 2015, Phys. Rev. Lett., 2015.

[149] K.-H. Thung and C.-Y. Wee. A Brief Review on Multi-Task Learning. *Multimedia Tools and Applications*, 77, 29705–29725, 2018.

[150] G. M. Amdahl. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, pages 483–485, 1967.

[151] B. Steininger, P. Pavone, and D. Strauch. Theoretical Investigation of the Lattice Dynamics of GaAlSb Superlattices. *Comp. Mat. Sci.*, 20, 376–380, 2001.

List of Figures

List of Tables

Acknowledgements

At first, I want to thank my main supervisor Prof. Claudia Draxl for her support and the patience with my growing craziness over the time in Berlin. She provided the financial support of this work by her Max-Planck fellowship and gave me the opportunities to join many workshops and conferences which connected me to the large material science community. It was a great pleasure to have a home-base at her group at the Humboldt University.

Second, my thank goes to Prof. Matthias Scheffler from the Fritz-Haber Institute of the Max-Planck Society as a co-supervisor. I very appreciate his deep expertise in discussions that set some main directions of this thesis. It was a great help to use the computing facilities of the MPG, and being often a guest at his group did not only make me cross the Berlin wall but also massively widened my scientific horizon.

I am also very thankful to Santiago Rigamonti as first responsible supervisor for the big support in every-day research. Although it was a hard school, he gave me a grounding in *ab-initio* methods - especially the use of `exciting` - and scientific writing. Many ideas of this thesis came up together with him, and in particular the CV strategies were his suggestion. Having a critical mind for scientific discussions sharpened the project really a lot and often reminded me that science is not a solo path.

Many thanks go to Luca Ghiringhelli that equally supported the actual research in a very profound way. His deep understanding of statistical learning led to many fruitful discussions and I hope that I could push his symbolic regression approach further.

I am grateful that Jan Vybíral and Ulf Leser agreed to offer their time as reviewers of this thesis.

My further thank goes to Martin Kuban that was a patient office colleague - the long research evenings made feel us like monks sometimes. From Claudia's group, I also explicitly thank Karsten Hannewald and Ute Werner for the initial support, the room mates Ronaldo Rodriguez Pela and Wahib Aggoune, Pasquale Pavone for the support with LaTeX, Andris Gulans for his assistence with `exciting`, the Secretary Helen Jurscha, and the PhD colleagues Olga Turkina, Sebastian Tillack and Maria Troppenz. You made me feel home in Adlershof.

From the FHI, I want also say my thank to Runhai Ouyang for the discussions on SISSO, Emre Ahmetcik from the Machine Learning subgroup, Matthias Rupp, Christopher Sutton and Daniel Speckhard for inspiring discussions, and Luigi Sbailo and Anne Hodgson for their writing guidelines. It was a great honor to be in touch with the very front level of research at the FHI and I also appreciate that I could stay connected to the FHI in the virtual coffee breaks during the crisis.

Important to mention are also the school managements of the Maximilians-von-Montgelas Gymnasium Vilsbiburg, the Maximiliansgymnasium München and the Johannes-Hofmiller Gymnasium Freising, in particular Bernhard Steininger [151], that financially helped to finish the thesis in remote by providing me the helping teacher positions. Combining research and teaching is the Humboldtian ideal and the sudden experience to teach curious sixth-graders was a great compensation at some writing deeps.

Additionally, my thank goes to Felix Kempf of the group of Prof. Frey in Munich and the library of the University of Applied Sciences in my hometown Landshut that in an uncomplicated way offered me workspaces after my leave of Berlin. Although the thesis was part of the NOMAD project and performed in the style of a nomad, I felt to have some further academic home-bases there.

Finally, and most importantly, my thank goes to my parents: for your almost unlimited patience with a water-kettle addicted lock-down student and for your steady personal support in the background over six hard years!