Franz Martin Rohrhofer, BSc

Machine learning of first-principles energies for carbon and boron crystal structures

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Technical Physics

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Phys. Dr.rer.nat. Wolfgang von der Linden
Institute of Theoretical and Computational Physics

Co-Supervisor

Dipl.-Ing. Dr.techn. Bernhard C. Geiger
Know-Center GmbH Research Center for Data-Driven Business & Big Data Analytics

Graz, May 2019

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date

Signature

Abstract

Machine learning (ML) models have become popular in the field of condensed matter physics, in combination with traditional quantum-mechanical methods, such as Density Functional Theory (DFT). One of the possible applications is to use ML to learn the potential energy landscapes of solids for crystal structure prediction applications. In general, the efficiency and accuracy of a ML application depend on the available data, the learning algorithm and the data representation, which provides relevant information about the system suitable for the learning algorithm. In this work we apply ML techniques to estimate the internal energies of polymorphic mono-elemental crystal structures of carbon and boron, generated by crystal structure prediction techniques. We explore different learning algorithms and develop a physically-motivated data representation, which encodes the structural information of the crystals. We optimize and evaluate the ML performance on datasets containing relaxed and mixed, i.e. relaxed and unrelaxed, crystal structures. Our results show that kernel-based regression methods with the proposed data representation provide promising predictions on energies of mixed structures, with accuracies comparable to DFT. We measure a mean absolute error (MAE) of about 10 meV/atom which suggests that ML models of this type can be used to replace expensive first-principles calculations in cost-intensive applications, such as crystal structure predictions.

Kurzfassung

Maschinelles Lernen ist weit verbreitet auf dem Gebiet der kondensierten Materie, besonders im Zusammenhang mit traditionellen quantenmechanischen Methoden, wie zum Beispiel der Dichtefunktionaltheorie (DFT). Eine mögliche Anwendung ist das Erlernen der Potentialhyperfläche von Festkörpern zur Vorhersage von Kristallstrukturen. Im Allgemeinen ist die Effizienz und Genauigkeit des maschinellen Lernens abhängig von den verfügbaren Daten, dem Lernalgorithmus und der Datendarstellung. Die Datendarstellung ist notwendig um relevante Informationen über das System quantitativ zu erfassen, sodass diese vom Lernalgorithmus verarbeitbar sind. In dieser Arbeit wenden wir unterschiedliche Methoden des maschinellen Lernens an, um die inneren Energien von polymorphen mono-elementaren Kristallstrukturen aus Kohlenstoff und Bor zu erlernen, die zuvor durch Kistallstruktur-Vorhersagen erzeugt wurden. Wir untersuchen unterschiedliche Lernalgorithmen und entwickeln eine physikalisch-motivierte Datendarstellung, welche die Kristallstruktur beschreibt. Wir optimieren und evaluieren die Leistung der Lernalgorithmen an Datensätzen, die relaxierte und gemischte, d.h. relaxierte und unrelaxierte, Kristallstrukturen beinhalten. Unsere Ergebnisse zeigen, dass Kernel-basierende Regressionsverfahren mit der entwickelten Datendarstellung genaue Vorhersagen von Energien gemischter Kristallstrukturen liefern, die mit quantenmechanischen Methoden vergleichbar sind. Mit einem ermittelten mittleren absoluten Fehler (MAE) von ungefähr 10 meV / Atom könnte die entwickelte Methode teure Berechnungen ersetzen, die in kostenintensiven Vorhersagen von Kristallstrukturen benötigt werden.

Contents

Chapter 1

Introduction

Improvements in technology are likely to be driven forward by findings and innovations in materials science. In 1964 Walter Kohn and Pierre Hohenberg published two theorems [1, 2], which set the framework of *Density Function Theory* (DFT); a computational quantum mechanical method, for which Walter Kohn was awarded the Nobel Prize in Chemistry in 1998. DFT, together with often so-called first-principles methods, is capable of solving the fundamental quantum mechanical equations of a system of interacting nuclei and electrons, permitting to calculate the properties of a material computationally, given only its chemical composition and structure. As a consequence, the search for novel materials has been increasingly motivated and accelerated by theoretical investigations on the computer.

Despite their strong predictive power, DFT calculations are still computationally too intensive for applications in which the system under study is large or calculations have to be performed repeatedly for different materials; these applications comes into play in so-called high-throughput screening or crystal structure prediction. In the field of crystal structure prediction, several techniques [3,4] have been developed to systematically search for relevant structures, by exploring the potential energy surface (PES) of a system under given external conditions; usually the energies are computed with first-principles methods. The amount of crystal structures one obtains from these techniques provide a suitable database for training a machine learning (ML) model.

These structure prediction techniques, in turn, vastly benefit from a successful ML application: once trained, ML models are capable of making estimations of material properties with a computational cost which is order of magnitude smaller, compared to first-principles methods. However, a requirement for ML models to be efficiently applied in the field of crystal structure prediction is to achieve a prediction accuracy comparable to the first-principles methods (1 10 meV/atom). This can be difficult since the model performance depends on several components, e.g. the quantity and quality of the data or the learning algorithm used. The most challenging and crucial part, though, is to describe materials through a suitable data representation in order to be processable by the learning algorithm.

The main aim of this thesis is to use different ML algorithms to learn and estimate internal energies of mono-elemental crystal structures. We obtain structures of carbon and boron from structure prediction methods using DFT, and our intent is to achieve ML accuracies, which are comparable to the DFT calculations. ML models of this kind can replace DFT calculations in a structure prediction application, speeding up the searching process considerably.

We have chosen carbon and boron due to their polymorphism: these elements exist in several crystalline phases with different atomic arrangements. On the one hand, this

provides a large database of heterogeneous mono-elemental crystal structures for the ML application. On the other hand, the complex bonding pattern of these structures renders the development of a suitable data representation challenging.

An ideal representation depends on the system under study, and has to fulfill certain properties in order to establish an efficient ML application. In the field of condensed matter physics ML models have been used for different applications. For problems similar to the one treated in this thesis, i.e. learning the potential energy landscape, Coulomb matrices [5,6], bag of bonds [7] and fingerprint techniques [8] have been successfully used to represent molecules and non-periodic materials for ML models. Nevertheless, these representations are not applicable for periodic crystalline solids. Symmetry functions [9,10] and smooth overlap of atomic positions [11] have been developed to individually encode the local environment of atoms in chemical compounds. As a consequence, ML predictions on all kinds of materials can be performed, by disassembling the materials into atomic environments, which are learned separately. Accurate estimations on material properties can be obtained by these methods, even though the training process demands a large amount of data and high cost of computer power. A different approach, which is more relevant for high-throughput applications, is to directly target a specific material property, by making use of descriptors. A descriptor is a single quantity, or a combination of a few quantities, which can easily be extracted from experiments or calculations, and correlates with the desired property. Examples of this approach can be found in [12,13].

In this thesis, we develop a data representation which contains several descriptors that encode relevant information about the geometry of mono-elemental crystal structures. We use physically motivated short- and long-range order descriptors to represent the crystal structures for the application of ridge and lasso regression. Furthermore, we introduce a radial distribution function [14,15], representing the local density, and an angular distribution function [16], which provides additional information about the orientation of chemical compounds in the structure. Both functions are adjusted and optimized on our data, to serve as a suitable data representation for kernel ridge regression and support vector regression. These are kernel-based methods, which in this thesis are implemented with the Gaussian kernel. In this way, we determine which regression method, in combination with our data representation, yields the most promising predictions on the internal energies of our mono-elemental crystal structures.

The structure of the thesis is the following: in chapter 2 a physical background of this thesis is given. Basic concepts of quantum mechanics are used to briefly discuss the fundamentals of density function theory in the beginning of this chapter. The concept of PES is discussed, which is crucial for understanding the ML application analyzed in the thesis. Furthermore, we explain the structure prediction techniques, which we use to obtain the crystal structures, that form our database.

Chapter 3 gives a theoretical framework on machine learning. At first, a general introduction of machine learning can be found, followed by a detailed discussion on linear regression. Linear regression is at the origin of any regression method used in this thesis: based on its concept, the most powerful ML methods have been developed, which are discussed in this section. Support vector regression is explained in the next section. The final section of this chapter contains practical machine learning techniques, which later are used to measure and improve the ML model performance.

Chapter 4 describes the crystal structures used in the thesis, and the data representation. We discuss the common description of crystal structures, in order to emphasize the difficulties in finding a suitable data representation for ML models. The main part of this chapter covers the introduction of our data representation and its components. We

discuss the representation for ridge and lasso regression, as well as the radial distribution function and the angular distribution function. We also give a description of the method, which we use to clean the database from duplicates. In the final section of this chapters we investigate correlations within components of our data representation.

Chapter 5 presents the machine learning application of ridge and lasso regression. First, we use the discussed ML techniques to adjust the data representation and model parameters. This approach provides the optimal parameters, which then are used to analyze the training process. Finally, we use the same settings to measure the prediction accuracy of ridge and lasso regression.

In chapter 6 the machine learning application of kernel-based regression methods is discussed. In the beginning of this chapter we optimize the data representation on our crystal structures. In particular, the radial distribution function and the angular distribution function are discretized. Then, we adjust the model parameters of kernel ridge regression and support vector regression using the optimized data representation. Again, we analyze the training process at optimal settings. Final predictions are performed and discussed by the end of this chapter.

Chapter 7 concludes the obtained results; further applications and possible improvements are discussed as well.

The appendices contain conventions and more theoretical background and derivations, which did not fit into the main body of this thesis. In particular, we describe the parameter estimation in the regression methods from a probabilistic point of view, as well as the application of the kernel trick in the kernel-based methods.

Chapter 2

Density Functional Theory

Density functional theory (DFT) is a powerful and commonly used technique to solve the quantum-mechanical problem for atoms, molecules and crystals. DFT is a first-principles, or ab-initio technique, i.e. it is a theoretical method that rely solely on fundamental laws and natural constants without any additional assumptions. A detailed discussion on DFT would be far beyond the scope of this thesis, however, in this chapter a compressed introduction and some basic concepts of DFT are given, to get a brief overview of how the database used in this thesis was generated. Furthermore, one of the final aims of machine learning approaches to solids, like those we discussed in this thesis, is to replace at least partially computationally expensive DFT calculations, and for this a good understanding of the DFT approach is needed. The following introduction is based on the book by D.S. Sholl and J.A. Steckel [17], as well as on the book by J.P. Perdew and S. Kurth [18].

2.1 Adiabatic Approximation of Many-Body Problems

The fundamental equation of first-principles theories is the Schrödinger equation with the Hamilton operator corresponding to a many-body problem. Typical many-body problems encountered in condensed matter physics comprise many electrons interacting with many nuclei, as in solids or molecules. The total wave function Ψ in real space consequently is a function of electronic and nuclear position coordinates, denoted by $\Psi(\mathbf{r}_1, .., \mathbf{r}_N, \mathbf{R}_1, .., \mathbf{R}_M)$ for a quantum system with N electrons and M nuclei. In this work we do not consider any spin effects; therefore we omit the spin dependency in the wave function and further discussions. Furthermore, we will use Hartree atomic units ($\hbar = e = m_\mathrm{e} = 1/(4\pi\epsilon_0) = 1$) and a short hand notations for the coordinates, $\mathbf{r}_\mathrm{e} \equiv (\mathbf{r}_1, .., \mathbf{r}_N)$ and $\mathbf{R}_\mathrm{n} \equiv (\mathbf{R}_1, .., \mathbf{R}_M)$.

The Hamilton operator is composed of the kinetic energy operator for electrons $\hat{T}_\mathrm{e}$ and nuclei $\hat{T}_\mathrm{n}$, as well as of electron-electron interactions $\hat{U}_\mathrm{ee}$, nucleus-nucleus $\hat{U}_\mathrm{nn}$ and electron-nucleus $\hat{V}_\mathrm{en}$ interactions,

$$\hat{H} = \hat{T}_\mathrm{e} + \hat{T}_\mathrm{n} + \hat{U}_\mathrm{ee} + \hat{U}_\mathrm{nn} + \hat{V}_\mathrm{en} \ . \tag{1}$$

The electron-nucleus interactions couples the nuclear and electronic motion, and thus prevents the separation of electronic and nuclear systems. A common approximation to treat this issue of non-separable electronic and nuclear systems is the *Born-Oppenheimer approximation* [19]. The approximation is based on the observations that atomic nuclei are much heavier than electrons; in fact a proton or neutron has more than 1800 times the mass of a single electron. As a consequence of the large disparity electrons are much more mobile than nuclei, and thus they are more sensitive to changes in their surroundings.

The atomic nuclei, on the other hand, are comparatively unaffected by the motion of the electrons, and hence the nuclear geometry is assumed to be fixed. The infinitesimal changes in the atomic positions equal the conditions of an adiabatic process, why the Born-Oppenheimer approximation also refers to an adiabatic approximation.

In the Born-Oppenheimer approximation we consequently neglect the kinetic term of atomic nuclei $\hat{T}_{\mathrm{n}}$, and assume that the Coulomb potential between the fixed nuclei $\hat{U}_{\mathrm{nn}}$ corresponds to a constant. The electron-nucleus Coulomb interaction $\hat{V}_{\mathrm{en}}$ furthermore is considered as an external potential, i.e. the electrons move in a field of nuclei. This finally permits to decouple the electronic and nuclear motion; hence the total wave function can be broken into its electronic ψ_{e} and nuclear ψ_{n} components

$$\Psi(\mathbf{r}_{\mathrm{e}}, \mathbf{R}_{\mathrm{n}}) = \psi_{\mathrm{e}}(\mathbf{r}_{\mathrm{e}}; \mathbf{R}_{\mathrm{n}}) \otimes \psi_{\mathrm{n}}(\mathbf{R}_{\mathrm{n}}) \; . \tag{2}$$

In further discussions the total electron wave function will be written without subscript and the parametric dependence on the nuclear coordinates will be neglected as well: $\psi(\mathbf{r}_{\mathrm{e}}) \equiv \psi_{\mathrm{e}}(\mathbf{r}_{\mathrm{e}}; \mathbf{R}_{\mathrm{n}})$. The remaining part of the Schrödinger equation for N electrons moving in the field of fixed nuclei in real space is given by

$$\left[-\frac{1}{2} \sum_{i=1}^{N} \nabla_i^2 + \sum_{i=1}^{N} V(\mathbf{r}_i) + \sum_{i=1}^{N-1} \sum_{j<i}^{N} U(\mathbf{r}_i, \mathbf{r}_j) \right] \psi(\mathbf{r}_{\mathrm{e}}) = E \psi(\mathbf{r}_{\mathrm{e}}) \; . \tag{3}$$

$V(\mathbf{r}_i)$ denotes the Coulomb interaction between an electron i and the fixed atomic nuclei, which reads as

$$V(\mathbf{r}_i) = -\sum_{I}^{M} \frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|} \; . \tag{4}$$

$U(\mathbf{r}_i, \mathbf{r}_j)$ represents the repulsive interaction between two electrons

$$U(\mathbf{r}_i, \mathbf{r}_j) = \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \; , \tag{5}$$

and E corresponds to the total energy of the electronic system.

2.2 The Two Hohenberg-Kohn Theorems

The fundamental object in density function theory is the electron density $n(\mathbf{r})$, which describes the probability of finding an electron at position $\mathbf{r}$ in space. The total electron density is given by

$$n(\mathbf{r}) = N \int d^3\mathrm{r}_2 \, ... \int d^3\mathrm{r}_N \; |\psi(\mathbf{r}, \mathbf{r}_2, ..., \mathbf{r}_N)|^2 \; , \tag{6}$$

assuming the wave function to be normalized, $\langle \, \psi \, | \, \psi \, \rangle = 1$. In the mid-1960 *Hohenberg and Kohn* [1] gave the proof, that for a given ground-state density n_0, in principle, it is possible to determine the corresponding ground-state wave function ψ_0. Generally speaking, the wave function is a unique functional of the electron density, $\psi[n]$, and the ground-state wave function is determined by

$$\psi_0 = \psi[n_0] \; . \tag{7}$$

Consequently, also the energy is a functional of the electron density

$$\begin{aligned} E[n] &= \langle\, \psi[n] \,|\, \hat{H} \,|\, \psi[n] \,\rangle \\ &= \langle\, \psi[n] \,|\, \hat{T}_\mathrm{e} + \hat{U}_\mathrm{ee} + \hat{V}_\mathrm{en} \,|\, \psi[n] \,\rangle \;, \end{aligned} \tag{8}$$

where we have used the Hamilton operator in the Born-Oppenheimer approximation. Hohenberg and Kohn have recorded their findings in their first theorem:

> **Theorem 1:** The ground-state energy of a many-electron system is an unique functional of the electron density, $E_0[n]$.

This set down the groundwork for reducing the original problem with $3N$ variables to only three coordinates. We use equation (8) to express the energy functional by its functional components

$$E[n] = T_\mathrm{e}[n] + U_\mathrm{ee}[n] + V_\mathrm{en}[n] \;, \tag{9}$$

where $T_\mathrm{e}[n]$ and $U_\mathrm{ee}[n]$ are called universal functionals. The non-universal functional $V_\mathrm{en}[n]$ depends on the system under study and can be written explicitly in terms of the density

$$V_\mathrm{en}[n] = \int d^3r \; V(\mathbf{r}) n(\mathbf{r}) \;. \tag{10}$$

The second theorem of Hohenberg and Kohn describes an important characteristic of the unique energy functional:

> **Theorem 2:** The functional will give the lowest energy if and only if it is evaluated at the true ground-state electron density.

As a consequence, one has to minimize equation (9) with respect to the electron density n to obtain the ground-state electron density n_0, and hence the ground-state Energy E_0.

$$E_0 = \min_{\psi \to n} \langle\, \psi \,|\, \hat{H} \,|\, \psi \,\rangle = \min_{n} \langle\, \psi[n] \,|\, \hat{H} \,|\, \psi[n] \,\rangle \;. \tag{11}$$

2.3 The Kohn-Sham Equations

The electron-electron interaction $\hat{U}_\mathrm{ee}$ is responsible for the complexity of solving the many-electron system: electron motions are dependent on each other, hence, the problem has be solved considering all electrons simultaneously. To circumvent this issue, we consider an auxiliary system of N non-interacting electrons, which are moving in an external potential $v(\mathbf{r})$. In this case the single electron equations read

$$\left[-\frac{1}{2}\nabla^2 + v(\mathbf{r}) \right] \psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}) \;, \tag{12}$$

where ψ_i denotes a single electron wave function, and ϵ_i the corresponding energy of the single electron quantum system. The electron density for this system is simply given by

$$n(\mathbf{r}) = \sum_{i=1}^{N} |\psi_i(\mathbf{r})|^2 \;. \tag{13}$$

We can express the total kinetic energy of this system in terms of the single electron wave function

$$T_{\mathrm{e}}^{\mathrm{KS}}[n] = \sum_{i=1}^{N} \int d^3r \; \psi_i^*(\mathbf{r}) \left(-\frac{1}{2}\nabla^2 \right) \psi_i(\mathbf{r}) \; . \tag{14}$$

We further introduce an electron-electron interaction energy by the so-called Hartree energy:

$$U_{\mathrm{H}}[n] = \frac{1}{2} \int d^3r \int d^3r' \; \frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \; . \tag{15}$$

The Coulomb interaction between the electrons and the fixed atomic nuclei again can be determined according to equation (10). We express the energy functional (9) of the many-electron system in terms of the introduced functionals of the auxiliary system:

$$\begin{aligned} E[n] &= T_{\mathrm{e}}[n] + U_{\mathrm{ee}}[n] + V_{\mathrm{en}}[n] \\ &= T_{\mathrm{e}}^{\mathrm{KS}}[n] + U_{\mathrm{H}}[n] + V_{\mathrm{en}}[n] + E_{\mathrm{XC}}[n] \; , \end{aligned} \tag{16}$$

where we have defined the so-called exchange-correlation functional $E_{\mathrm{XC}}[n]$. This functional corrects all omitted many-body effects due to the non-interacting auxiliary system:

$$E_{\mathrm{XC}}[n] = (T_{\mathrm{e}}[n] - T_{\mathrm{e}}^{\mathrm{KS}}[n]) + (U_{\mathrm{ee}}[n] - U_{\mathrm{H}}[n]) \; . \tag{17}$$

It can be shown [18] that by the variational problem of equation (11) an expression of the external potential in equation (12) follows, so as to solving the auxiliary system gives rise to the ground-state density. For this the functional derivative of the energy functionals (15) and (17) has to be taken, which yields an expression for the external potential as follows:

$$v_{\mathrm{eff}}[n(\mathbf{r})] = v(\mathbf{r}) + u_{\mathrm{H}}[n(\mathbf{r})] + v_{\mathrm{XC}}[n(\mathbf{r})] \; , \tag{18}$$

with the Hartree potential

$$u_{\mathrm{H}}[n(\mathbf{r})] = \int d^3r' \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \; , \tag{19}$$

and the exchange-correlation potential

$$v_{\mathrm{XC}}[n(\mathbf{r})] = \frac{\delta E_{\mathrm{XC}}[n]}{\delta n(\mathbf{r})} \; . \tag{20}$$

The potential v_{eff} is commonly referred to as effective potential, as the electrons seem to effectively interact with each other through a field caused by the electron density. We use the effective potential, to finally write down the *Kohn-Sham* equations [2]:

$$\left[-\frac{1}{2}\nabla^2 + v_{\mathrm{eff}}[n(\mathbf{r})] \right] \psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}) \; , \tag{21}$$

where the single electron wave functions ψ_i are commonly referred to as Kohn-Sham orbitals. Here it should be mentioned, that until the present day the exact representation of the exchange-correlation functional E_{XC} – except for the free electron gas – has not been found yet. Nevertheless, various approximations to the exchange-correlation functional exist, which achieve accurate results for certain quantum systems. The most commonly used approximations are the local density approximation [20] (LDA) and the generalized gradient approximation [21] (GGA). In the local density approximation the functional

depends only on the density at the position in space where the functional is evaluated:

$$E_{\mathrm{XC}}^{\mathrm{LDA}}[n] = \int d^3r\, n(\mathbf{r})\, \epsilon_{\mathrm{XC}}(n(\mathbf{r}))\ , \tag{22}$$

where ϵ_{XC} is the exchange-correlation term determined for the free electron gas. The generalized gradient approximation additionally includes the gradient of the density at the position in space to account for the non-homogeneity of the true density:

$$E_{\mathrm{XC}}^{\mathrm{GGA}}[n] = \int d^3r\, n(\mathbf{r})\, \epsilon_{\mathrm{XC}}(n(\mathbf{r}), \nabla n(\mathbf{r}))\ . \tag{23}$$

Further terms can be added to achieve more accurate results at a higher computational cost. A commonly known illustration of the collection of exchange-correlation functionals in dependence on simplicity and accuracy of methods is the Jacob's Ladder of DFT [22].

The Kohn-Sham equations (21), in combination with the definition of the electron density (13), form a set of non-linear differential equations, which can be far easier solved than the original many-electron system. However, the definition of the external potential includes the electron density, which is obtain from the single electron wave functions. Those, in turn, are determined by the Kohn-Sham equations, which again use the definition of the external potential. Therefore, the differential equations have to be solved self-consistently:

1) An initial trial electron density $n_{\mathrm{init}}(\mathbf{r})$ is defined, for instance as a superposition of atomic densities

2) The defined electron density is used to solve the Kohn-Sham equations, which yields the Kohn-Sham orbitals ψ_i

3) The obtained Kohn-Sham orbitals are used to determine the electron density $n(\mathbf{r})$ according to equation (13)

4) The energies, using the electron densities from step 2 and 3, are compared. If the difference is larger than a defined threshold, the density is updated with a linear mixing scheme of the two density and is used to start again the calculations from step 2. The self-consistent procedure is stopped as the difference is below the threshold

5) The finally obtained density corresponds to the – approximated – true electron density, and thus the determined energy represents the ground-state energy

2.4 Plane Waves and Brillouin Zone Sampling

Solving the Kohn-Sham equations directly is still a complicated and computationally expensive procedure; hence, further approximations are needed to be done in order to solve the Kohn-Sham equations numerically.

In this work we consider crystal structures, which are solid materials with a certain periodicity. We refer the crystal periodicity to a set of translation vectors $\boldsymbol{T}$ and the corresponding reciprocal lattice vector $\boldsymbol{G}$, so that

$$\boldsymbol{T} \cdot \boldsymbol{G} = 2\pi N \quad \text{where } N \in \mathbb{Z}\ , \tag{24}$$

holds for any choice of translation and reciprocal lattice vectors of the crystal structure. The physicist Felix Bloch gave the proof [23] that wave functions for particles in a periodic

potential, as in crystals, must be of the form

$$\psi_{\boldsymbol{k},n}(\mathbf{r}) = e^{i\boldsymbol{k}\cdot\mathbf{r}} u_{\boldsymbol{k},n}(\mathbf{r}) \;, \tag{25}$$

where $e^{i\boldsymbol{k}\cdot\mathbf{r}}$ describes a plane wave and $u_{\boldsymbol{k},n}$ is a function, which has the period of the crystal lattice, i.e. $u_{\boldsymbol{k},n}(\mathbf{r}) = u_{\boldsymbol{k},n}(\mathbf{r} + \boldsymbol{T})$. The wave vector $\boldsymbol{k}$ is also referred to as the Bloch vector. The subscript indicates that for each Bloch vector $\boldsymbol{k}$, there exist multiple solutions to the Schrödinger equation, which are different in energies and labeled by the band index n. Equation (25) shows that the eigenfunctions of the Schrödinger equation for a periodic potential can be expressed by the product of a plane wave $e^{i\boldsymbol{k}\cdot\mathbf{r}}$ times a function $u_{\boldsymbol{k},n}(\mathbf{r})$ with the periodicity of the crystal.

To describe the lattice periodic functions $u_{\boldsymbol{k},n}(\mathbf{r})$, one expands them in terms of basis functions. There are essentially two classes of basis functions: the first class can be referred to atom-centered basis functions, for instance Gaussian-type [24] or muffin-tin orbitals [25]. However, in this thesis we consider the second class of basis functions, which are planes waves [26]. In the plane wave approach the function, invariant under a crystal lattice translation, is expanded as a Fourier series, which forms a complete and orthonormal plane wave basis. For the Kohn-Sham orbitals we thus obtain

$$\psi_{\boldsymbol{k},n}(\mathbf{r}) = e^{i\boldsymbol{k}\cdot\mathbf{r}} \sum_{\boldsymbol{G}} c_{\boldsymbol{k},n}(\boldsymbol{G}) e^{i\boldsymbol{G}\cdot\mathbf{r}} \;, \tag{26}$$

where $c_{\boldsymbol{k},n}(\boldsymbol{G})$ are the expansion coefficients. The corresponding Kohn-Sham energies are $\epsilon_{\boldsymbol{k},n}$, and using the Kohn-Sham orbitals (26), the electron density is given by

$$n(\mathbf{r}) = \sum_{\boldsymbol{k},n} \sum_{\boldsymbol{G},\boldsymbol{G}'} f(\epsilon_{\boldsymbol{k},n}) c^*_{\boldsymbol{k},n}(\boldsymbol{G}') c_{\boldsymbol{k},n}(\boldsymbol{G}) e^{i(\boldsymbol{G}-\boldsymbol{G}')\cdot\mathbf{r}} \;, \tag{27}$$

where the $f(\epsilon_{\boldsymbol{k},n})$ represent the occupation numbers.

The Fourier series in the reciprocal lattice vectors $\boldsymbol{G}$ forms an infinite basis set. For the practical implementation a cut-off energy $E_{\text{cut-off}}$ is defined: wave vectors which do not fulfill $\frac{1}{2}|\boldsymbol{k}+\boldsymbol{G}|^2 < E_{\text{cut-off}}$ are omitted for the calculation. Another computational simplification is to represent the quasi-continuous Bloch vectors $\boldsymbol{k}$ by a discrete set of $\boldsymbol{k}$-points within the first Brillouin-zone. This is commonly realized by generating an equally spaced $\boldsymbol{k}$-mesh. Both approximations are introduced to circumvent the infinite summations in equation (27). Hence, due to the simplifications there are two convergence parameters that have to be adjusted in the practical implementation of the plane wave method: the Brillouin zone sampling and the energy cut-off.

Moreover, wave functions in the vicinity of the atomic cores can oscillate rapidly, for which an extremely large basis set has to be considered. To reduce the high computational effort, pseudopotential are introduced which describe the wave functions around the atomic cores by smooth pseudo-wave functions.

In this thesis we use the Vienna Ab Initio Simulation Package (VASP) [27] to implement the plane wave-pseudopotential codes.

2.5 Potential Energy Surface and Crystal Structure Prediction

The Kohn-Sham approach yields the ground-state energy of the many-electron system. To obtain the ground-state energy of the many-body system, one needs to take account

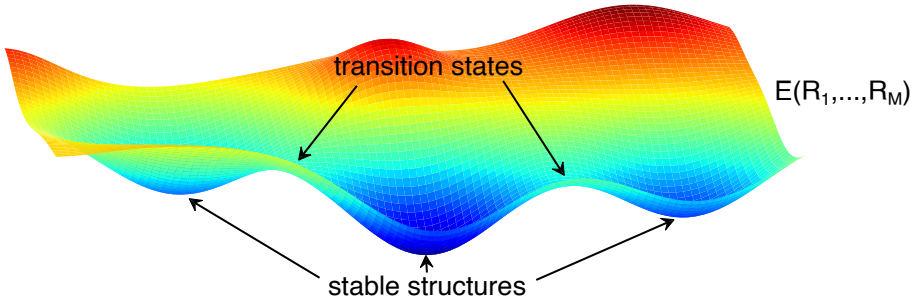


Figure 2.1: Conceptual representation of a potential energy surface, simplified to two dimensions. Local minima on the surface correspond to stable structures. Transition states can be found on saddle points between local minima.

of the repulsive Coulomb term between atomic nuclei, and add to the total energy

$$E_{\mathrm{nn}} = \sum_{\alpha=1}^{M-1} \sum_{\beta>\alpha}^{M} \frac{Z_\alpha Z_\beta}{|\mathbf{R}_\alpha - \mathbf{R}_\beta|} \,, \tag{28}$$

where M is the number of atoms in the system. As a consequence of the Born-Oppenheimer approximation, and the parametric dependence on the nuclear coordinates $\mathbf{R}_1, .., \mathbf{R}_M$, we can consider the ground-state energy of the many-body system as a function of atomic nuclei positions $E(\mathbf{R}_1, .., \mathbf{R}_M)$.

This multivariate function is known as the *potential energy surface* (PES), which for an arbitrary two-dimensional projection is plotted in figure 2.1. Each point on the surface represents a set of atomic positions $\{\mathbf{R}_i\}$. The number of possible configurations in principle is infinite, and an infinitesimal small change in atomic positions causes a small change in the energy; the PES thus is a *continuous* function. Global and local minima on the PES correspond to stable and metastable structures, i.e. configurations for which interatomic distances are relaxed and thus for which all forces within the structures vanish. Relaxation of a structure describes the approach by which its current atomic positions are varied until interatomic distances are in equilibrium. Since forces equal the negative gradient of the PES, minimizing them under relaxation can be seen as moving towards the nearest local minima on the PES; similar to a marble on the surface, falling into the nearest local minimum in the absence of external actions.

In the later course of this thesis, we refer to minima structures as *relaxed* structures. We refer to structures, which are not stable and do not correspond to minima on the PES, as *unrelaxed* structures. Saddle points on the PES correspond to transition states which are the highest energy points on the lowest energy pathway connecting two local minima. The transition states divide the PES into so-called *basins of attraction*. These are regions on the PES, corresponding to a set of configurations which under relaxation converge to the same basin minimum. Multiple basins can form a *funnel*, which describes a collections of basins, where the lowest minimum of the collection can be reached without crossing high-energy barriers.

In materials science the consideration of first-principles methods became essential: by knowing the correct atomic structure of a material, one can determine its physical properties, even before the material is synthesized. In practice, one is interested in determining the physical properties of the thermodynamically stable structures of a system at a given temperature and pressure. Initially, crystal structures of this kind thought to be funda-

mentally unpredictable, but recent development, taking into account the PES, has shown that crystal structure prediction can be successfully performed [28,29]. Common structure prediction approaches, which are based on computational optimizations, are simulated annealing [30], metadynamics [31], basin hopping [32], minima hopping [4] and evolutionary algorithms [3].

In this thesis, we consider random structure generation and minima hopping techniques. Random structure generation is based on the fact that most of the PES corresponds to high-energy structures, in which atoms are much closer or far apart from their equilibrium positions. Minima structure, on the other hand, tend to appear in funnels, located on small areas on the PES. Hence, generating structures from undesired high-energy regions can be circumvent by putting *a priori* constraints on structures of the system under study. For instance, including information from experimental observations or previous searches can reduce the number of possible candidates considerably. Furthermore, the stoichiometry and the number of atoms of structures is fixed for practical reasons. A structure search starts by generating crystal structures with random atomic positions, taking into account the constraints on the bond lengths. Using first-principles calculations, the structures are relaxed to their basin minimum. Performing random displacements of the atomic positions potentially pushed structures into a nearby basin of attraction. In this way, the PES is scanned for global and local minima structures, which has been proven to be a successful and reliable method for small systems.

The minima hopping method is based on a feedback mechanism, which helps to climb out of a funnel in order to explore new regions on the PES. Two essential components form the feedback mechanism: the first part is an inner loop, which performs a certain number of molecular dynamics steps to push the current crystal structure out of its basin. The second part is an outer loop, which relaxes the structure after the molecular dynamics simulation, and checks if the relaxed structure landed in a new basin minimum by comparing the energies and structure fingerprints [33–35]. In case of the structure being the same from which the escape trail started, the inner loop is repeated with an increased number of molecular dynamics steps. However if the structure differs from the initial one, the outer loop determines whether the new minima structure is rejected or accepted, based on a defined energy threshold. The threshold is continuously adjusted after each outer loop step: a rejection of the new minima increases the threshold; an acceptance decreases it. This introduces a mechanism which drives the system to local minima, but also helps to escape from regions where the algorithm got stuck.

Chapter 3

Machine Learning

> 'Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.'

This definition was given by Arthur Samuel in 1959 and coined the term machine learning which is often abbreviated to ML. At that time machine learning had not yet become established as it is nowadays. In fact, machine learning did not start flourishing until the 90's; this led Tom Mitchell to formulate a more precise definition of the field:

> 'A computer program is said to learn from experience E, with some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.'

This definition can be used to express the main aim of this thesis using similar considerations: DFT calculations have been used on crystal structures to obtain their ground-state energies (E). The available data is used to train a machine learning model which consequently performs estimations of ground-state energies on unexplored crystal structures (T). Its predictive performance (P), measured by the deviation of predicted energies from energies obtained using DFT calculations, improves with the quality and the quantity of available data used in the training process. This in particular emphasizes that machine learning approaches have great similarities to methods from statistics and probability.

The following section is used to give an overview of the classification of basic machine learning methods and the commonly used terminology which is adopted in the course of the thesis. Section 3.2 introduces linear regression and its improvements since these approaches provide the basis for modern machine learning models used in this thesis. After a brief section on support vector regression, there is a practical discussion on approaches used to determine and improve model performances. Parts of the sections are based on the book by A. Géron [36].

3.1 Types of Learning Algorithms

With the growth of computational power over the last decades, novel algorithms and methods commonly referred to as machine learning have been developed. Today, already numerous methods exist and state-of-the-art models achieve tremendous performances on highly complex fields of applications, such as image or speech recognition [37, 38]. A general categorization of machine learning methods is given by figure 3.1.

The first main category of machine learning is supervised learning, which uses labeled data to learn and predict discrete or continuous outputs. In other words, the machine

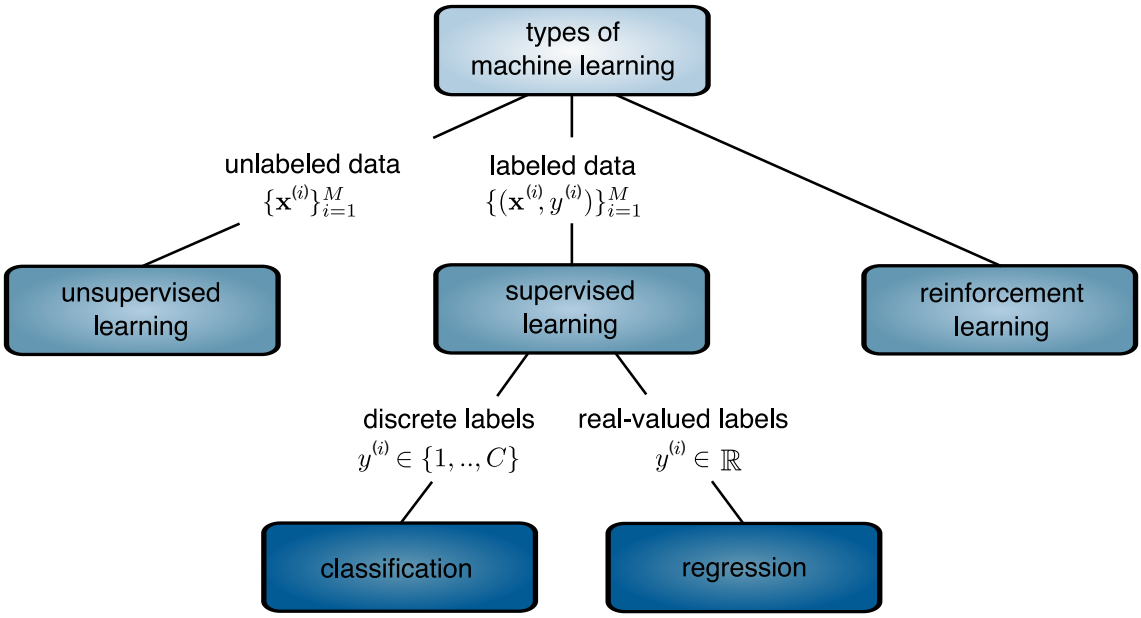


Figure 3.1: A categorization of machine learning algorithms. In this thesis we consider the application of regression methods.

learning model tries to learn a mapping from input variables $\boldsymbol{x}$, called *features*, to output variables y, named *labels*. Alternative definitions for the input variables from statistical perspectives are attributes, covariates, independent variables, explanatory variables, predictor variables or regressors. However, typically in machine learning only the word features is used exclusively. In the field of condensed matter physics the input variables often are referred to as descriptors, as they are used to describe the structures under study. The output variables can also be called target variables, response variables or dependent variables. In this case there is no unique definition used in machine learning, and all these expressions are used equivalently.

In order to learn the mapping from available data, input-output pairs $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$ with a total number of M instances are provided forming the so-called training set. In the previous expression, we have used the bold face for the features, because a single training example usually depends on several features, which are packed into a single feature vector $\boldsymbol{x}$.

Supervised learning can be further divided into two categories, depending on possible values of the labels. If the label is a categorical or nominal variable, i.e. its value can be chosen from a finite set $y^{(i)} \in \{1, .., C\}$ with at maximum of C values, one speaks of a classification problem. In contrast, if labels are real-valued and thus continuous without any restrictions, one speaks of a regression problem.

The second category of machine learning is unsupervised learning or descriptive learning. This method uses solely unlabeled input $\{\boldsymbol{x}^{(i)}\}_{i=1}^{M}$ to cluster the data and detect patterns in it. It is not possible to tell a priori to the model which pattern it should detect, and hence there is no simple way to measure its performance. Some important tasks solved typically by unsupervised learning are clustering, visualization, dimensionality reduction and association rule learning.

In the categorization of figure 3.1 the last and third category of machine learning is known as reinforcement learning. This technique defines a learning system, called agent, which tries to maximizes its cumulative rewards by observing the environment, selecting

and performing actions for which it receives positive or negative rewards based on the main objective. These systems typically make headlines when beating human world champions at strategic games like chess or Go [39].

There exists another common categorization in machine learning based on how the model is trained. In contrast to online learning where the ML model is trained incrementally by feeding data sequentially to improve the performance of the model, the batch learning – sometimes offline learning – uses all the available data at once to learn from them and to make predictions afterwards without learning anymore.

3.2 Linear Regression

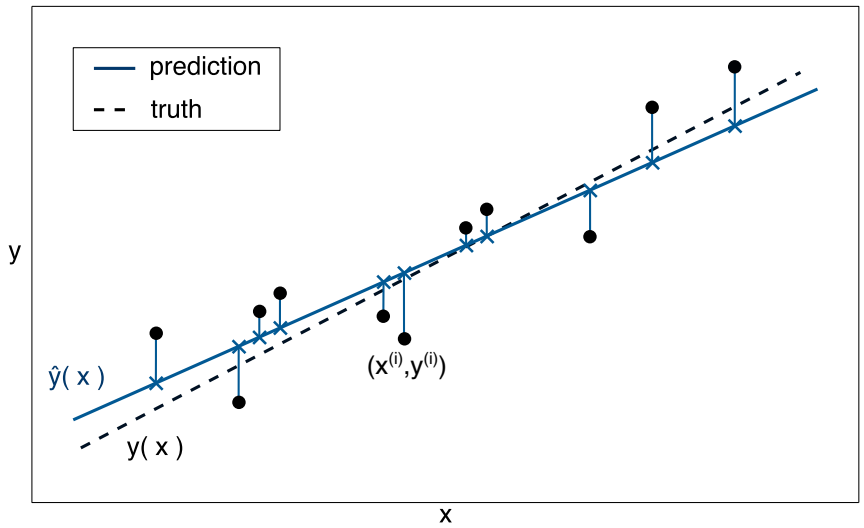


Figure 3.2: The basic concept of linear regression: the underlying linear relationship (dashed line) is estimated by the hypothesis function (solid line) based on given input-output samples (dots) of the underlying relationship.

Linear regression is one of the most fundamental techniques in machine learning since modern algorithms and models, like kernel ridge regression, support vector regression or even neural networks, are based on the concept of linear regression. With modern software packages, such as Scikit-learn [40] or TensorFlow [41], the implementation of high-performance algorithms can be easily achieved with few lines of programming code. Nevertheless, a basic understanding of the underlying concepts and derivations is beneficial for a proper usage.

The fundamental concept of linear regression is, that for a given set of input features $\boldsymbol{x}$ a linear response of the output label y is assumed:

$$y(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{x} + \epsilon = \sum_{j=0}^{N} \beta_j x_j + \epsilon \, . \tag{29}$$

Here, $\boldsymbol{\beta}^T$ denotes the transposed weight vector containing the weights of the function, $N+1$ is the number of features, and ϵ is called the statistical error between the theoretical and the observed outcome. Weights can be also referred to as feature weights, model weights or model coefficients. We identify the feature and weight vectors with column vectors, wherefore the weight vector is assumed to be transposed to represent the algebraic dot or

inner product in equation (29):

$$\boldsymbol{x} \equiv \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^{N+1} \ , \ \boldsymbol{\beta} \equiv \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_N \end{bmatrix} \in \mathbb{R}^{N+1} \ . \tag{30}$$

The indexing in equations (29) and (30) starts from $j = 0$ since by convention the parameter β_0 corresponds to the intercept or *bias* term. In order to leave this parameter independent of any input quantity, we define $x_0 \equiv 1$, so that the original input feature vector $\tilde{\boldsymbol{x}}$ containing N features can be constructed as $\boldsymbol{x} \equiv \begin{bmatrix} 1 \\ \tilde{\boldsymbol{x}} \end{bmatrix}$.

Equation (29) does not represent the prediction of the linear regression model, but the mapping from input to output, which the linear regression model *tries* to find and learn. Figure 3.2 should give a clearer picture: the dashed line represents the theoretical linear relationship between the input features and the output labels - in the figure a one-dimensional case is considered where the output depends on a single feature. The linear relationship – in case of figure 3.2 described by the slope of the dashed line – corresponds to the values of the weights $\boldsymbol{\beta}$. Linear regression estimates this theoretical relation from a representative sample of input-output pairs indicated by dots in figure 3.2. Furthermore the observed output values underlie a certain deviation from their expected values which is described by the statistical error ϵ in equation (29). The extent of the statistical error is determined by inaccuracies in calculations, measurements or natural spread. The common approach is to assume that the distribution of all statistical errors is Gaussian which helps estimating the model parameters. The training data is then used to estimate the weights. Here, several training objectives can come into play: minimizing the mean squared error, the mean absolute error, or maximizing the likelihood of the weights under a certain probabilistic model. In this sense, the estimated weights do not only depend on the true theoretical relation between the observed input and output samples, but also on the training objective and, potentially, on the training algorithm.

Once the model is trained with data, the weights are estimated which subsequently can be used to make predictions on new data according to

$$h_{\hat{\boldsymbol{\beta}}}(\boldsymbol{x}) \equiv \hat{y} = \hat{\boldsymbol{\beta}}^T \boldsymbol{x} \ , \tag{31}$$

where $\hat{\boldsymbol{\beta}}$ here denotes the estimated model parameters, and $\hat{y}$ the prediction using these parameters in the *hypothesis function* $h_{\hat{\boldsymbol{\beta}}}(\boldsymbol{x})$.

3.2.1 Vectorization and Parameter Estimation

The approach to find the optimal weights comes from the Bayesian concept learning and is part of the discussion in appendix B. Assuming the statistical errors in (29) to be Gaussian distributed, the maximum likelihood estimation (MLE) minimizes the mean squared error (MSE) to estimate the weights. The mean squared error is given by

$$J(\hat{\boldsymbol{\beta}}) \equiv \mathrm{MSE}(\hat{\boldsymbol{\beta}}) = \frac{1}{M} \sum_{i=1}^{M} (y^{(i)} - \hat{\boldsymbol{\beta}}^T \boldsymbol{x}^{(i)})^2 \ , \tag{32}$$

where $J(\hat{\boldsymbol{\beta}})$ is a commonly used abbreviation in the context of ML, and often referred to as *cost function*. The quantities $y^{(i)}$ and $\boldsymbol{x}^{(i)}$ denote the output label and the input feature

vector of the i-th training example or, more generally, the given data set $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$ with M training examples.

The quantity in brackets in (32) is identified as the residual errors $\hat{\epsilon}^{(i)} = (y^{(i)} - \hat{\boldsymbol{\beta}}^T \boldsymbol{x}^{(i)}) = y^{(i)} - \hat{y}^{(i)}$. The residual error $\hat{\epsilon}$ is frequently confused with the statistical error ϵ. However, the residual error measures the difference between predicted and observed values, whereas the statistical error corresponds to the deviation between the observed and expected values. In figure 3.2 the residual error is illustrated as vertical lines between observed (dots) and predicted (crosses) values. The statistical error instead is not shown, but would correspond to vertical lines from observed values to the dashed line.

The mean squared error (32) can be rewritten in a vectorized form improving simplicity and efficiency of algorithms. In order to make this possible, we store the observed labels $y^{(i)}$ in the training set with M instances in a single label vector

$$\boldsymbol{y} \equiv \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(M)} \end{bmatrix} \in \mathbb{R}^M \ . \tag{33}$$

Furthermore, we arrange the feature vectors $\boldsymbol{x}^{(i)}$ as the rows into a matrix $\boldsymbol{X}$ which is often called *design matrix*:

$$\boldsymbol{X} \equiv \begin{bmatrix} -(\boldsymbol{x}^{(1)})^T - \\ -(\boldsymbol{x}^{(2)})^T - \\ \vdots \\ -(\boldsymbol{x}^{(M)})^T - \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_N^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \cdots & x_N^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(M)} & x_1^{(M)} & \cdots & x_N^{(M)} \end{bmatrix} \in \mathbb{R}^{M \times (N+1)} \ . \tag{34}$$

Here, $x_j^{(i)}$ denotes the j-th feature of the i-th training example. The features in the first column correspond to the additional feature due to the bias term, hence $x_0^{(i)} \equiv 1$. Using these definitions, we express the mean squared error (32) by

$$J(\hat{\boldsymbol{\beta}}) = \frac{1}{M} \|\boldsymbol{y} - \hat{\boldsymbol{\beta}}^T \boldsymbol{X}\|^2 \ , \tag{35}$$

where $\|\boldsymbol{v}\| = \sqrt{\sum_i v_i^2}$ corresponds to the Euclidean or ℓ^2-Norm of a vector.

The optimal weights $\hat{\boldsymbol{\beta}}$ can be determined analytically (appendix B.1). The equation that yields the optimal weights is known as normal equation and defined as

$$\hat{\boldsymbol{\beta}}^{\mathrm{MLE}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \ , \tag{36}$$

where $\boldsymbol{X}^T \boldsymbol{X} \in \mathbb{R}^{(N+1)\times(N+1)}$ is called the sum of squares matrix, and MLE refers to the *maximum likelihood estimate*.

Determining the inverse of the sum of squares matrix limits the application of the normal equation for obtaining the optimal parameters. This calculation is computationally expensive and its numerical cost increases as $\mathcal{O}(N^3)$, where N is the matrix dimension. Consequently, increasing the number of features considerably increases the computational effort required for solving equation (36). In addition, the matrix has to be invertible which is possible if and only if all rows and columns of the matrix are linearly independent.

When the number of features exceeds the limit of available computational resources, numerical optimization packages find their practical usage. The most widely used numer-

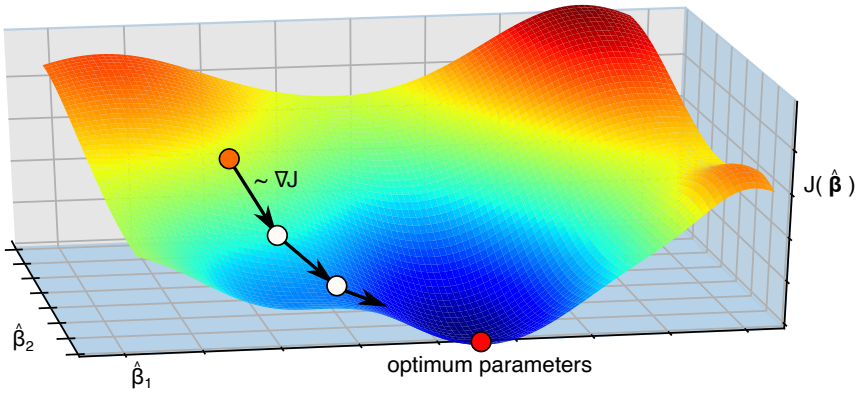


Figure 3.3: The concept of gradient descent: from an initial set of parameters (orange), updates are made according to the steepest descent at the current position on the surface of the cost function. The global minimum (red) corresponds to the optimal set of parameters that minimize the cost function.

ical optimization method is gradient descent.

Gradient descent takes advantage of the fact that the surface of the cost function (35) is convex in the parameter vector $\hat{\boldsymbol{\beta}}$. Hence, the optimal set of parameters that minimizes the total cost corresponds to the global minimum of the cost function.

The basic approach is to randomly choose an initial set of parameters which corresponds to a certain cost. The steepest descent at a given position on the surface, i.e. the direction of the largest reduction in cost regarding the model parameters $\hat{\boldsymbol{\beta}}$, is equal to the negative gradient at this position $\nabla_{\hat{\boldsymbol{\beta}}} J(\hat{\boldsymbol{\beta}})$. Thus, the initial set of parameters is updated according to

$$\hat{\boldsymbol{\beta}} \rightarrow \hat{\boldsymbol{\beta}} - \alpha \cdot \nabla_{\hat{\boldsymbol{\beta}}} J(\hat{\boldsymbol{\beta}}) \,, \tag{37}$$

where α is called the learning rate and determines the step size. Updates are repeated as many times until either a maximum number of iterations is reached or the difference in cost after an update step falls below a certain threshold. Figure 3.3 illustrates the surface of an arbitrary cost function and the path followed by gradient descent steps. Gradient descent has the great advantage that it is not as susceptible to a large number of features as the normal equation is, but has the disadvantage that its numerical implementation and fine-tuning of convergence requires the adjustment of additional parameters. Table 3.1 highlights the basic differences between the two estimation methods.

3.2.2 Expansion on Basis Functions

So far, we have only considered a linear relationship between input features $\boldsymbol{x}$ and output labels y, but the concept of linear regression can be applied to more complex relations than linear functions. In particular the expression *linear regression* refers to the fact that the model is *linear in the weight parameters*. A non-linear relationship between input and output can be obtained by modifying the original features $\boldsymbol{x}$ with a non-linear function of the features $\boldsymbol{\phi}(\boldsymbol{x})$, called *basis function expansion*. In other words, a basis function expansion is a mapping of the original set of features to another feature space:

$$\boldsymbol{\phi} : \mathbb{R}^{N+1} \rightarrow \mathbb{R}^{N'} \,, \tag{38}$$

Normal equations	Gradient descent
analytical	numerical
slow if number of features is large	suitable for high numbers of features
susceptible to redundant features and exact duplicates in examples	no restrictions
no additional parameters	learning rate and iteration steps have to be adjusted

Table 3.1: Property and behavior comparison of the normal equation and gradient descent

where $N+1$ is the original number of features and N' the number of features after the transformation. Any appearance of the original feature vector $\boldsymbol{x}$ can be replaced by the transformed set of features $\boldsymbol{\phi}(\boldsymbol{x})$. Hence, final predictions are made using a modified hypothesis function

$$h_{\hat{\boldsymbol{\beta}}}(\boldsymbol{x}) = \hat{\boldsymbol{\beta}}^T \boldsymbol{\phi}(\boldsymbol{x}) \; . \tag{39}$$

A basis function expansion is a powerful tool, which can be used to model a non-linear relationship between input and output. For instance, the basis function $\boldsymbol{\phi}(x) = [x, x^2, .., x^d]$ transforms the original feature x to a feature space containing the polynomial expansion of x up to degree d. Using the resulting set of features permits to estimate a polynomial relation of degree d. This approach is referred to as polynomial regression. In practice, the exact form of the underlying relationship is unknown; therefore, one determines the optimal polynomial degree among probable values by comparing the predictive performance on new data. This approach is further discussed in section 3.4.4.

Figure 3.4 illustrates estimations of polynomial regression with a basis function expansions of degree $d = 2$ (left) and $d = 10$ (right). The function of polynomial degree $d = 10$, however, overshoots the actual objective of estimating the underlying relationship, since the function would also fit the stochastic noise, i.e. the statistical error, in the data. This should not be considered at all. In machine learning this situation is indicated with the term *overfitting* and its consequences will be discussed in section 3.4.

Considering data with multiple input features stored in the feature vector $\boldsymbol{x}$, the most general polynomial basis function expansion transforms the input features according to

$$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1, x_2, \; ... \; , x_1 x_2, \; ... \; , x_1^2 x_2, \; ... \; , x_1^d, \; ... \; , x_N^d] \; \in \mathbb{R}^{N'} \; . \tag{40}$$

This basis expansion, however, leads to an explosion of $(N'+d)!/(d!N'!)$ features. To understand why a polynomial basis function expansion as defined in equation (40) could be problematic, we can consider the following example: if, for instance, the number of original features is $N = 10$ and a basis function expansion with polynomial degree of $d = 10$ is applied, the original set of features is transformed to a set of approximately $N' = 185\,000$ features. This, in turn, requires estimating 185 000 model parameters which definitely limits the effectiveness of high polynomial basis function expansions when training set is small.

A useful practical approach to avoid overfitting and increase the algorithms computational efficiency, would be to determine the most important terms in the expansion and

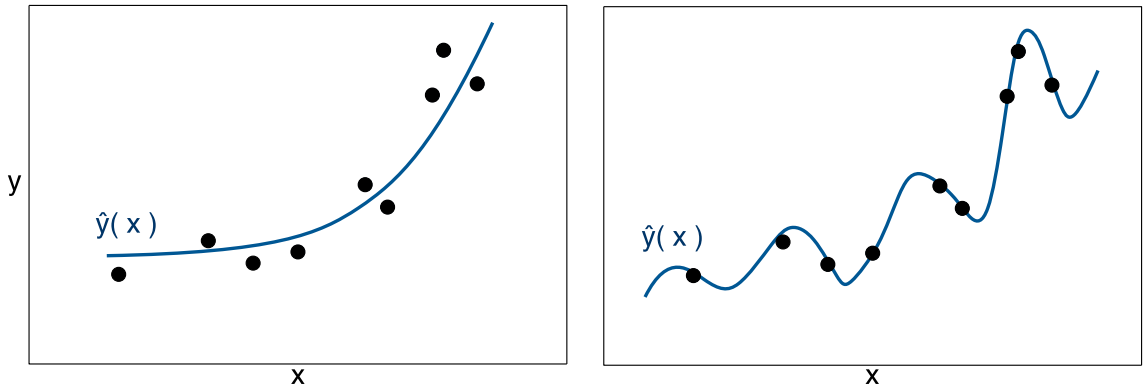


Figure 3.4: With the application of basis function expansions a non-linear relationship between input and output can be estimated. In polynomial regression the original feature is transformed by a basis function expansion, for instance with polynomial degree $d = 2$ (left) or $d = 10$ (right).

to leave out terms that do not improve the model performance. In section 3.2.4 lasso regression is introduced which mathematically realizes this idea, and obtains a sparse solution of optimal model parameters. Moreover, one of the most effective improvements in machine learning to deal with this issue is the concept of kernel functions, discussed in the following section.

3.2.3 Kernel Functions

In modern machine learning kernel functions – in short *kernels* – have achieved great success in improving efficiency and applicability to complex systems; The underlying concept is the kernel trick. The kernel trick permits to transform the original set of N features to a higher or even infinite dimensional feature space $N' \to \infty$, often called hyperspace, without actually carrying out the transformation $\boldsymbol{\phi}(\boldsymbol{x})$. This can be mathematically realized formulating the original problem in such a way, that the kernel trick can be applied. This permits to solve the optimization problem at a reduced computational cost. We will give further details on this topic after a general discussion on kernel functions.

In general a kernel function κ describes a mapping from two input arguments, e.g. two feature vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$, to a single real value, i.e. $\kappa(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}$. In order to make use of the kernel trick a few conditions, better known as the Mercer conditions, have to be fulfilled. A kernel is said to be a Mercer (positive definite) kernel if it is symmetric, i.e. $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}', \boldsymbol{x})$, and non-negative, i.e. $\kappa(\boldsymbol{x}, \boldsymbol{x}') \geq 0$. Under these conditions, Mercer's theorem [42] proves that there exists a basis function expansion $\boldsymbol{\phi}(\boldsymbol{x})$, such that

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}') \; . \tag{41}$$

Conversely, this implies that instead of determining the dot product of two transformed feature vectors, one could simply determine the value of the corresponding kernel function, without ever carrying out the transformation of the original vectors.

This we can show using the polynomial kernel, for instance:

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = (r + \gamma \boldsymbol{x}^T \boldsymbol{x}')^d \; . \tag{42}$$

This kernel fulfills the Mercer conditions. We consider two feature vectors, each of which

contains two features

$$\boldsymbol{x} \equiv \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad , \quad \boldsymbol{x}' \equiv \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} \tag{43}$$

and a polynomial basis function expansion with degree $d = 2$ which maps the original feature vector to a 6-dimensional feature space, according to

$$\boldsymbol{\phi}(\boldsymbol{x}) \equiv [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2] \, . \tag{44}$$

Here, we have included certain scaling prefactors; however, this does not affect the actual optimization of the problem. Subsequently, the dot product of the basic function expansion, applied to both vectors, can be calculated and further rearranged:

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}') &= \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}^T \cdot \begin{pmatrix} 1 \\ \sqrt{2}x_1' \\ \sqrt{2}x_2' \\ x_1'^2 \\ x_2'^2 \\ \sqrt{2}x_1'x_2' \end{pmatrix} \\ &= 1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_1'x_2x_2' \\ &= (1 + x_1x_1' + x_2x_2')^2 \\ &= (1 + \boldsymbol{x}^T\boldsymbol{x}')^2 \, , \end{aligned} \tag{45}$$

which equals the polynomial kernel (42) with degree $d = 2$ and parameters $\gamma = r = 1$. In other words, the dot product of the transformed vectors is equal to the square of the dot product of the original vectors, $\boldsymbol{\phi}(\boldsymbol{x})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}') = (1 + \boldsymbol{x}^T\boldsymbol{x}')^2$. The latter, however, is computationally more efficient, particularly in situations where the basis function expansion transforms the vectors to a even higher dimensional feature space.

A commonly used kernel of great importance is the *Gaussian* radial basis function (RBF) kernel, which is defined by

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \, \|\boldsymbol{x} - \boldsymbol{x}'\|^2) \, , \tag{46}$$

where γ is a free parameter and referred to as the kernel coefficient. The Gaussian kernel fulfills the Mercer conditions which proves the existence of a corresponding basis function expansion. The form of the underlying basis function expansion, though, can not be explicitly represented since the transformed feature map lives in an infinite dimensional space due to the Taylor expansion of the exponential function. Hence, the application of the Gaussian kernel enables access to an infinite dimensional feature space at low computational cost. This makes the Gaussian kernel one of the most powerful tools in modern machine learning techniques.

An alternative perspective on the Gaussian kernel is, that it serves as similarity measure since the kernel will return values close to $\kappa \approx 1$ when two feature vectors are similar and $\kappa \approx 0$ when input vectors differ from each other. The range of similarity, i.e. the characteristic drop from $\kappa \approx 1$ to $\kappa \approx 0$, is determined by the kernel coefficient γ.

Up to this point, the application of the kernel trick has not been discussed. The complete discussion on the kernel trick can be found in appendix C. It can be shown that the original optimization problem can be restated in a different optimization problem, called the *dual problem*. In the dual problem, terms of the inner product $\boldsymbol{\phi}(\boldsymbol{x}^{(i)})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}^{(j)})$ appear, which then can be replaced by the corresponding kernel function $\kappa(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$.

Furthermore, *dual variables* $\hat{\boldsymbol{\alpha}}$ are defined, which finally help to completely kernelize the original problem. By this means, the entire optimization problem can be treated using solely the computational cheaper kernel functions, instead of basis function expansions.

The dual problem yields the hypothesis function

$$h_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}) = \sum_{i=1}^{M} \hat{\alpha}^{(i)} \kappa(\boldsymbol{x}, \boldsymbol{x}^{(i)}) \,, \tag{47}$$

which shows that the prediction on new data depends directly on the data $\boldsymbol{x}^{(i)}$ from the training set. Instead of learning the weights $\hat{\boldsymbol{\beta}}$ for a fixed-sized feature vector, the model remembers each training example and learns a corresponding weight $\hat{\alpha}^{(i)}$ for it.

Specifically for the Gaussian kernel, the original set of features is no longer used to directly estimate the output value, but only serves as a representation that uniquely describes the data in the feature space. This alternative perspective on designing relevant features implies that the prediction on new data is based on the similarity to already learned and known data.

Applying the kernel trick, and treating the optimization in the dual problem, demands a computational cost of order $\mathcal{O}(M^3)$, where M is the number of training examples. This implies that the usage of kernel functions renders the number of original features negligible. Therefore, applying kernel functions, especially the Gaussian kernel, has great advantages over ordinary basis function expansions particularly in situations where

- the number of training examples M is a small or intermediate number and
- the data representation or rather the feature engineering cannot be achieved easily.

The second statement applies especially to situations in which there is no background knowledge about the true relationship between input and output. In other words, designing features and determining the optimal polynomial degree of the basis function expansion can be complicated when the underlying system is complex. The application of the Gaussian kernel, however, simplifies this procedure.

3.2.4 Regularization

Regularization in machine learning is a key concept which is used to avoid fitting the statistical error in the data. This issue typically occurs when the degree of the polynomial of the model used for estimation is larger than the degree of the underlying relationship. On one hand, reducing the polynomial degree may help to simplify the class of the hypothesis function. On the other hand however, regularization provides a versatile and more flexible way to prevent model estimations from fitting the statistical error. Beyond that, a specific definition of regularization yields a sparse solution of model parameters which can be a computational advantage.

In general, regularization can be realized by demanding small weight parameters $\hat{\boldsymbol{\beta}}$ as discussed in appendix B.2. Among different definitions to achieve this, a common approach yields the new cost function

$$J(\hat{\boldsymbol{\beta}}) = \frac{1}{M} \|\boldsymbol{y} - \hat{\boldsymbol{\beta}}^T \boldsymbol{X}\|^2 + \lambda \sum_{j=1}^{N} \hat{\beta}_j^2 \tag{48}$$

with the additional regularization term in the weight parameters $\hat{\beta}_j$. The parameter λ is called the regularization parameter and determines the degree of regularization. The

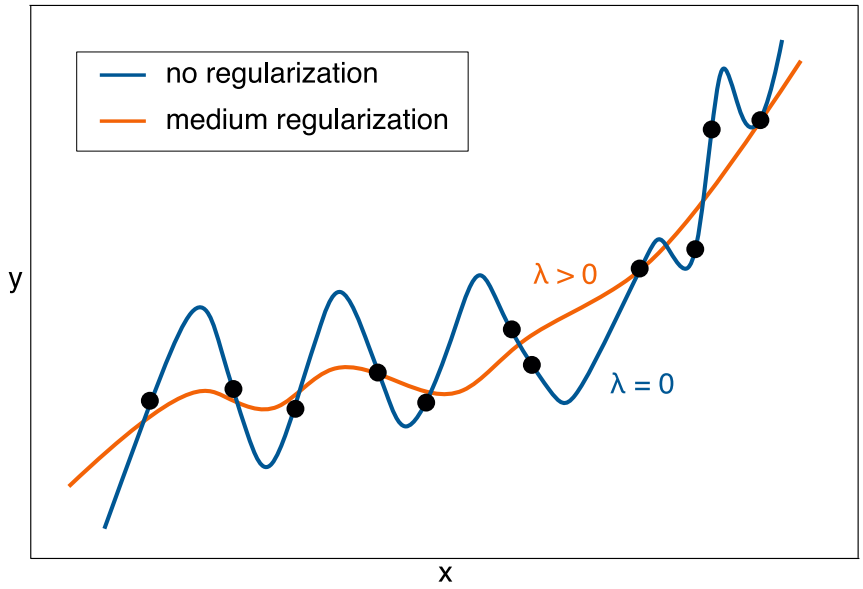


Figure 3.5: Regularization simplifies the class of the hypothesis function; the degree is adjusted with the regularization parameter λ. This helps to avoid fitting the statistical error in the data.

sum in the regularization term is written out explicitly to emphasize that the bias term $\hat{\beta}_0$ should not be regularized since its suppression can cause a large bias error. Considering the weight vector $\hat{\boldsymbol{\beta}}$ without the bias term, the regularization term can be expressed using the squared ℓ^2-norm of the vector $\|\hat{\boldsymbol{\beta}}\|^2$. Therefore, this regularization is called ℓ^2-regularization or Tikhonov regularization [43]. In machine learning the application of the ℓ^2-regularization in the linear regression approach is referred to as ridge regression.

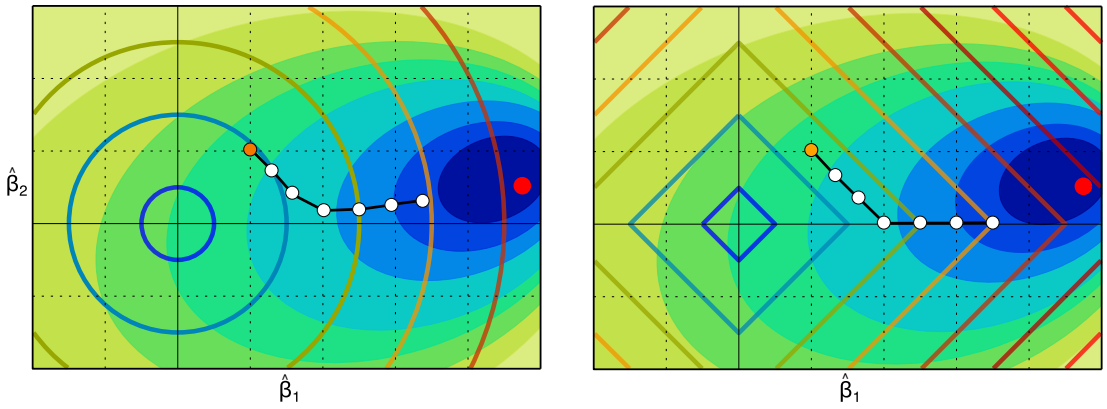
The regularization term causes the weight parameters to be small, and the level of regularization can be adjusted by the regularization parameter λ. With $\lambda = 0$, regularization is removed and (48) simplifies to (35), the unconstrained linear regression. This corresponds to the blue line in figure 3.5, which uses all weights in full extent to fit all training examples. Turning on regularization by choosing $\lambda > 0$ causes the weight parameters to be diminished, which results in a simpler class of the hypothesis function, conforming to the orange line in the figure. Setting the regularization parameter to large values $\lambda \gg 0$, the weight parameters are forced to assume low values. This translates into a simple class of the hypothesis function, where in the extreme case of $\lambda \to \infty$ only the bias term remains.

Including the regularization term in the objective of minimizing the cost function (48) yields a new analytical solution of optimal weight parameters, given by

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\lambda \boldsymbol{I}_{N+1} + \boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{49}$$

where $\boldsymbol{I}_{N+1}$ denotes the identity matrix with matrix dimension $N+1$. Applying the kernel trick to ordinary ridge regression – as discussed in appendix C – one of the most powerful regression methods in machine learning, the kernel ridge regression, is obtained. Kernel ridge regression combines the concept of regularization with that of kernel functions.

In addition to ridge regression, an alternative regularized regression approach exists, the lasso regression [44]. Lasso stands for *least absolute shrinkage and selection operator*



(a) Ridge Regression (b) Lasso Regression

Figure 3.6: The difference of ridge and lasso regression: using numerical optimization methods, model parameters in ridge regression (a) are updated without any preference for sparsity. In lasso regression (b) model parameters of least importance are set to zero during the updates which corresponds to sparse solutions.

and its cost function is defined by

$$J(\hat{\boldsymbol{\beta}}) = \frac{1}{M}\|\boldsymbol{y} - \hat{\boldsymbol{\beta}}^T \boldsymbol{X}\|^2 + \lambda \sum_{j=1}^{N} |\hat{\beta}_j| \ . \tag{50}$$

Again, the regularization term can be expressed by a vector norm if the bias term is not considered. In lasso regression, however, this corresponds to the absolute value or ℓ^1-norm.

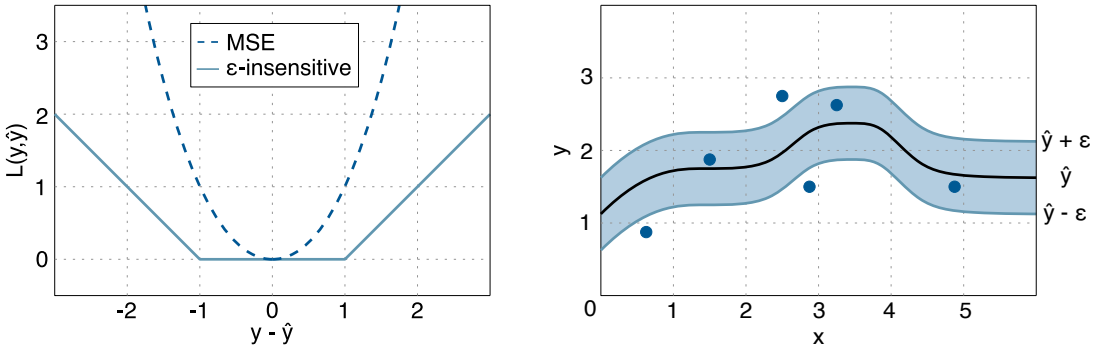
In lasso regression there exists no closed form solution for optimum parameters like the modified normal equation (49) for ridge regression. Due to the absolute value function in equation (50), the regularization term is not differentiable at $\hat{\beta}_j = 0$. Several techniques from convex analysis and optimization theory, for instance subgradient methods [45] or least-angle regression [46], have been developed to numerically estimate optimal parameters.

In the numerical optimization approach the main difference between ridge and lasso regression is revealed: The ℓ^2-regularization in ridge regression causes the weights to be modified simultaneously when $\lambda > 0$. The ℓ^1-regularization in lasso regression, in contrast, tends to completely eliminate weights of least importance first, leaving behind a sparse model, i.e. unimportant weights are set equal to zero as regularization is adjusted.

Considering the application of basis function expansions with high polynomial degrees and hence large feature vectors, lasso regression can achieve better computational performances compared to ridge regression as sparse solutions of model parameters are used and considered. In addition, lasso regression is often used to determine the relative importance of single features.

Whether ridge or lasso regression achieves better predictions depends on the underlying problem, the available data and particularly the choice of the regularization parameters. In the optimal case both methods give the same result.

Summarizing the discussion on the two methods, figure 3.6 shows an illustration of how ℓ^1- and ℓ^2-regularization act on the minimization problem when considering numerical optimization methods. The update of weights are marked by white dots, while the yellow dots represent the arbitrary initial set of parameters. In both pictures the background



(a) Cost (b) Hypothesis Function

Figure 3.7: (a) MSE-cost and the ϵ-insensitive cost where $\epsilon = 1$. (b) Support vector regression tries to fit as many points inside the ϵ-margin while reducing the cost of margin violations. Points inside the margin do not give any cost.

contours indicate the regularized cost function, i.e. the ordinary least square function, and the red dot marks the optimal value of two arbitrary feature parameters $\hat{\beta}_1$ and $\hat{\beta}_2$. In this example the parameter weight $\hat{\beta}_2$ is assumed to be less important than $\hat{\beta}_1$, which is indicated by the lower optimal weight value. Turning on regularization is equivalent to imposing an upper constraint B to possible values of weights, whose value increases as regularization gradually gets turned off, i.e. $B \sim \frac{1}{\lambda}$. The constraint surface of ridge regression due to the regularization term $\hat{\beta}_1^2 + \hat{\beta}_2^2 \leq B$ can be illustrated as concentric circles, whereas considering lasso regression the regularization term $|\hat{\beta}_1| + |\hat{\beta}_2| \leq B$ describes diamond-shaped contours. The larger the constraint contour, the higher the cost for the particular choice of weights; varying the degree of regularization consequently changes the distance between the constraint contours in the figures.

In the actual optimization problem, the final cost function is the superposition of the two contours, the regularized and the regularized. Thus, the resulting contour has a slightly different shape with different optimal feature weights whose values depend on the applied regularization. In particular, when the regularization exceeds a certain degree, the optimal value of the parameter weight $\hat{\beta}_2$ in lasso regression would lie on the horizontal axis, i.e. $\hat{\beta}_2 = 0$. However, in figure 3.6 the unregularized and constraint contours are plotted separately in order to show their exact influence on the actual optimization problem.

The total cost for a set of parameters is lowest where the contours of the ordinary cost function intersects the constraint contours in a single point. Here it becomes evident, that in lasso regression the model tries to update the weights along the corners of the diamond shape since the corners of the regularization contours first intersect the contours of the objective cost function. This corresponds to sparse solutions since, as shown, the parameter $\hat{\beta}_2$ first is set to zero and further updates occur along the axis. In contrast, in ridge regression the constraint contours do not have any sides or corners, so the intersect takes places at any point without any preference for sparsity.

3.3 Support Vector Regression

In machine learning support vector machines (SVM) are powerful models which like kernel ridge regression are capable of performing regression with the application of kernels and regularization. Mostly known for their great performance on complex classification

problems, support vector machines are also widely used as regression models.

In case of kernel ridge regression the application of kernel functions causes the model to estimate the weights $\hat{\alpha}^{(i)}$ of each training example in the dual problem. Support vector machines, however, do not take into account every single training example to make predictions according to equation (47), but only use a sparse estimate of weights $\hat{\alpha}^{(i)}$. This can be achieved defining a new cost function

$$L_\epsilon(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| < \epsilon \\ |y - \hat{y}| - \epsilon & \text{otherwise} \end{cases} . \tag{51}$$

This cost function is called the ϵ-insensitive cost function and by definition any observed output value that lies outside an ϵ-margin around the prediction gives rise to a certain cost, which increases linearly with the residual error as shown in figure 3.7a. Any point lying inside an ϵ-margin around the true value is not penalized and thus the approach of the support vector regression model can be seen to fit as many instances as possible in the margin while trying to reduce the margin violations, i.e. instances off the shaded area, as shown in figure 3.7b. The final objective cost function including regularization is given by

$$J(\hat{\boldsymbol{\beta}}) = C \sum_{i=1}^{N} L_\epsilon(y^{(i)}, \hat{y}^{(i)}) + \frac{1}{2} \sum_{j=1}^{M} \hat{\beta}_j^2 . \tag{52}$$

Using support vector machines the cost function typically is divided by the regularization parameter yielding the new regularization constant $C = 1/2\lambda$. Without proof, the cost function is convex, but not differentiable due to the absolute value function. One common approach is to define slack variables which helps restating the cost function without an absolute value function. The slack variables correspond to the degree to which points lie above or below the ϵ-margin:

$$\begin{aligned} y^{(i)} &\leq \hat{y}^{(i)} + \epsilon + \xi_+^{(i)} \\ y^{(i)} &\geq \hat{y}^{(i)} - \epsilon - \xi_-^{(i)} \\ \xi_+^{(i)}, \xi_-^{(i)} &\geq 0 \end{aligned} \tag{53}$$

Consequently, points above the margin have $\xi_+^{(i)} > 0$ and $\xi_-^{(i)} = 0$. Inside the ϵ-margin $\xi_+^{(i)} = 0$ and $\xi_-^{(i)} = 0$ applies and points below the margin have $\xi_+^{(i)} = 0$ and $\xi_-^{(i)} > 0$. We express the cost function (52) using the slack variables:

$$J(\hat{\boldsymbol{\beta}}) = C \sum_{i=1}^{N} (\xi_+^{(i)} + \xi_-^{(i)}) + \frac{1}{2} \sum_{j=1}^{M} \hat{\beta}_j^2 , \tag{54}$$

which has to be minimized under the constraints of equations (53). One can show, e.g. in ref. [47], that the optimal solution for a given dataset depends on the parameters ϵ and C and that the kernelized hypothesis function is given by

$$h(\boldsymbol{x}) = \hat{\beta}_0 + \sum_i \hat{\alpha}^{(i)} \kappa(\boldsymbol{x}, \boldsymbol{x}^{(i)}) . \tag{55}$$

The solution of optimal dual weights $\hat{\alpha}^{(i)}$, though, is sparse because instances inside the ϵ-margin are disregarded. This results in considerably faster predictions of the SVR algo-

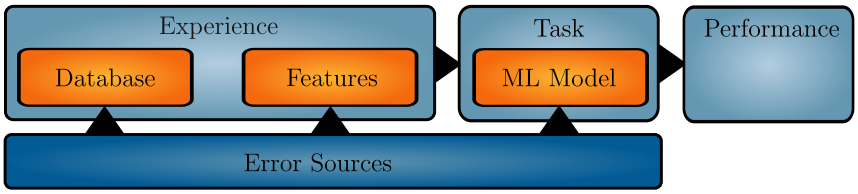


Figure 3.8: Components in machine learning: the model learns the data in the training database, represented by features of the data representation. The performance on the task increases with experience. The data, the data representation and the ML model can be a source of errors, affecting the model performance.

rithm, especially for large datasets. Training examples that lie on or outside the ϵ-tube, i.e. for which $\hat{\alpha}_i > 0$, are called the support vectors.

3.4 Machine Learning Concepts and Methods

In this section we introduce practical approaches and concepts in machine learning, such as scoring parameters and methods providing helpful tools for applying and developing a model. In the beginning of chapter 3 we have discussed the definition of machine learning given by Tom Mitchell. The definition comprises the expressions experience, task and performance; Figure 3.8 gives a graphical overview on how these expressions combine with the instances an entire machine learning application consists of.

Here we want to emphasize, that each component – the data, the features and the ML model – can be a source of error, and thus influences the final model performance; for instance, outliers in the data, inefficient features to describe the data or inadequate model assumptions. How strongly these sources of errors influence the final prediction depends completely on the system under study and cannot be assessed in advance. Methods introduced in this section, though, can be used to detect and minimize the errors.

3.4.1 Scoring Parameters

Scoring parameters are the main instruments available to rate performances of models. Especially in optimization steps, measuring the model performance on the test set, as well as on the training set, is a common approach to monitor the behavior and improvements of the model. Numerous scoring parameters exist; each comes along with advantages and disadvantages.

A commonly used scoring parameter in regression analysis is the mean squared error which already has been introduced as objective quantity that becomes minimized in the least squares approach. Nevertheless, the mean squared error is also applicable in regression methods which do not use the least squares cost function, for instance support vector regression. The fundamental characteristic of the mean squared error, which gives a straightforward interpretation, is that its value always is non-negative and the closer its value is to zero, the better the performance. A value of zero hence corresponds to perfect predictions meaning predicted target values have zero deviation from true values.

Nonetheless, the mean squared error also has drawbacks: the mean squared error is database dependent [48], which is why a direct comparison of model performances and generalized statements on the predictive power should be considered in combination with the used data. Furthermore, the mean squared error is not dimensionless, but its unit is equal to the squared unit of the data. We use the mean squared error in the optimization

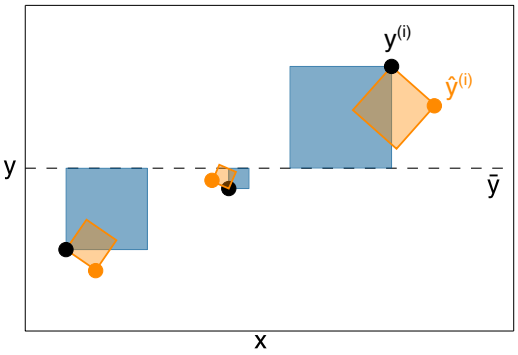


Figure 3.9: The coefficient of determination explains how much variation in the target value y is described by the input features $\boldsymbol{x}$. This is determined by the ratio between the residual sum of squares (orange) and the total sum of squares (blue).

of our machine learning model.

Further performance measures, we consider to rate the final predictive power, are the root mean squared error (RMSE) and the mean absolute error (MAE). Both share the advantage of describing the performance in units of the variable of interest. The root mean squared error can be directly obtained from the mean squared error by $\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$. In contrast, the mean absolute error is defined by

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y^{(i)} - \hat{y}^{(i)}| \,. \tag{56}$$

Compared to the mean absolute error, the root mean squared error gives higher weights to larger errors due to the squaring before the averaging. This fact can be used to characterize the distribution of errors that are made: in case that exactly equal errors are made, the root mean squared error would show the same value as the mean absolute error. The more the errors are dispersed, the larger the difference between both.

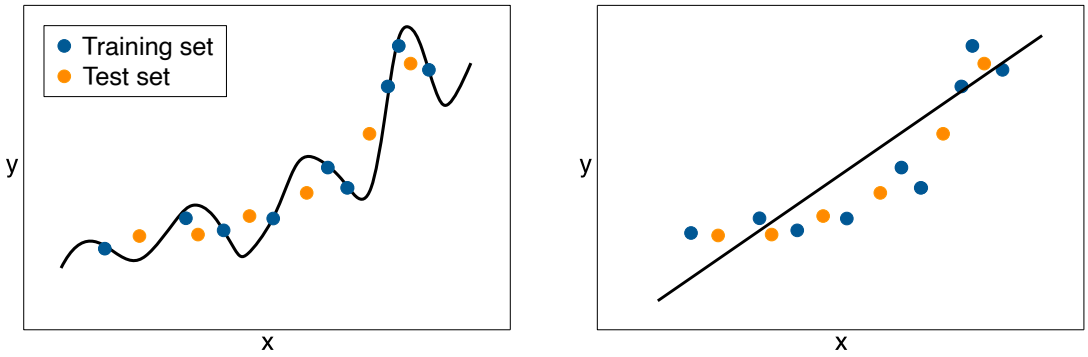
Lastly, we introduce the coefficient of determination, which provides a dimensionless rate of the predictive power. The coefficient of determination is denoted by R^2 and among several, a common definition is given by

$$R^2 = 1 - \frac{RSS}{TSS} \,, \tag{57}$$

where $RSS = \sum_i (y^{(i)} - \hat{y}^{(i)})^2$ is the residual sum of squares and $TSS = \sum_i (y^{(i)} - \bar{y})^2$ the total sum of squares. The coefficient of determination is a measure, based on the ratio of the prediction error and the total variance in the true values, as shown in figure 3.9. In regression analysis the R^2 value is typically seen as a measure of how much variation in the target value y is explained by the features $\boldsymbol{x}$.

By its definition, the R^2-value is dimensionless and always $R^2 \leq 1$. Perfect predictions are $R^2 = 1$ since the residual sum of squares converges to zero in the numerator of the second term in (57). The R^2-value is zero if no variation in y, hence, no relationship between y and $\boldsymbol{x}$ is explained. This corresponds to a situation, where the model always predicts the mean of the target value. Moreover if predictions are worse than this, the value shows $R^2 < 0$.

The coefficient of determination, due to the directly defined total sum of squares, has the drawback that adding data to the validation set possibly increases the R^2-value. This



(a) Overfitting (b) Underfitting

Figure 3.10: (a) The model overfits the data, i.e. it performs well on the training set but generalizes badly on the test set. (b) The class of the hypothesis function is too simple to estimate the underlying relationship, and underfits the data. The performance is bad on the test set and the training set as well.

would be possible since the additional data increases the total sum of squares, whose rise could be larger than the increase of the residual sum of squares. Consequently, R^2-values are only comparable if measured on one and the same database.

A common method to practically optimize and measure the model performance is *cross validation*. Instead of using always the same training and test set, which would focus and optimize the model on that specific training set, all available data points are combined in a single database. The model performance then is measured during multiple runs while randomly choosing at each run the training and test set from the entire data. Usually, 80% or 90% of the data serves as the training set and the complementary part is taken as the test set. This random separation and validation is repeated a certain number of times to reduce variability and to obtain a performance value by the average of the single runs. Additionally, the standard deviation from the sequence is determined to provide an estimate of the stability of predictions.

3.4.2 Overfitting and Underfitting the Data

In supervised learning, *overfitting* and *underfitting* are the most frequent situations which highly affect the model's predictive power. In following discussions we use the mean squared error to characterize the model performance.

Overfitting the data refers to the undesirable situation in which the model performs well on the training data, but shows a bad generalization and performance on the test set. The underlying cause has been already discussed and corresponds to circumstances in which the model also fits the statistical error in the data as shown in figure 3.10a. As a consequence the mean squared error measured on the training set is low, but the model fails to generalize the predictions on the test set, causing high errors on the test set. Using cross-validation the model may randomly take training sets which at this specific choice represents the entire data well, and hence the error on the test set can be low. This leads to high variation in the sequence of performance runs wherefore overfitting is also often referred to as high variance.

In contrast, when the model underfits the data the performance is bad on the test set, but also on the training set. This is caused by the class of the hypothesis function being to simple to estimate the true structure of the relationship, as captured by figure 3.10b. Too

simple features could be a possible explanation for this, as well as a low polynomial degree or a high regularization. Since the prediction error here is independent on the choice and amount of data, underfitting is referred to as high bias.

3.4.3 Learning Curves

Visualizing and detecting overfitting and underfitting issues as shown by figure 3.10 is generally hard; especially in the use of multiple input features or kernel functions. Under circumstances like this, a common approach is to have a look onto the *learning curves*.

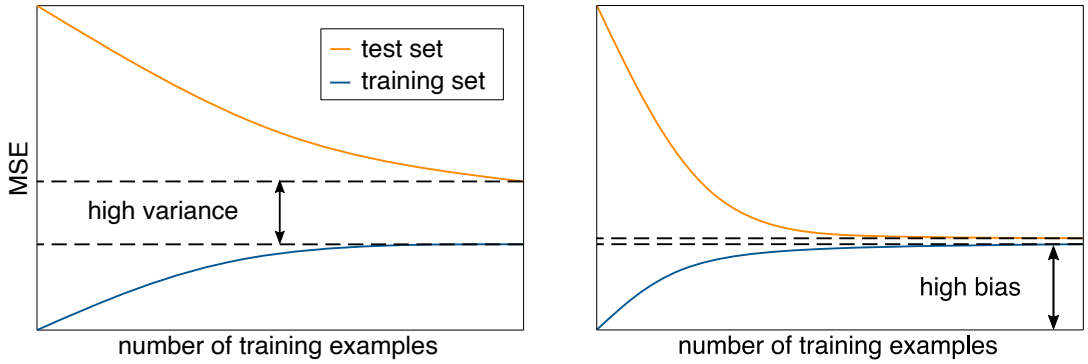
The concept of learning curves is to visualize overfitting and underfitting by means of the performance difference between training and test set. In practice, this is realized by incrementally increasing the number of data used for training the model, starting with solely two training examples and increasing the training set size up to the maximum available data. At each stage the performance is measured on the used training set and the test set, which unlike the training set is kept at fixed size during the entire process.

The performances on the training and test set then are plotted against the number of training examples as shown in figure 3.11. Starting with training solely two examples, the model will be able to fit and predict the two examples perfectly by a simple straight line, while the predictions on the test set will completely miss their true values. Thus, the prediction error is low on the training set and high on the test set. As soon as more training data is introduced, the model cannot fit every single training example anymore, which is shown by an increase in the training set prediction error as new instances are included. The prediction error on the test set, though, is decreased since the model better generalizes on the data. Both error curves approach each other as more data is included in the training process.

In the event of underfitting issues, where the model's hypothesis function is too simple as shown in figure 3.10b, the model makes equal prediction errors on the training and test set at the maximum available data. Hence, both prediction curves converge to the same error value, and the absence of any gap between both curves can be the indication of underfitting issues. This is shown by figure 3.11b. Furthermore, the final error value corresponds to the bias error caused by the class of the hypothesis function being to simple.

Overfitting, in contrast, can be shown by a comparably slow increase of the prediction error on the training set, as the complex class of the hypothesis function is able to accurately learn and describe more training examples. Going back to figure 3.10a, the model also learns the statistical error – even at the maximum available data – why the prediction error on the test set will be higher than on the training set due to the bad generalization. This is indicated by a distinct gap between the prediction curves of the training and the test set, which is the typical proof of overfitting issues. Furthermore, in the use of cross validation the final performance on the test set would show variations in the sequence of performance runs, and hence instability in the performance. Figure 3.11a represents the corresponding situation of overfitting the data.

A good fit of the learning algorithm exists between an overfit and underfit model. In this case the training and the test set performance decrease to a point of stability with a minimal gap between the two final prediction errors. The prediction error on the training set will still be slightly lower than on the test set; the corresponding final gap between the two curves is referred to as generalization gap.



(a) Overfitting/high variance (b) Underfitting/high bias

Figure 3.11: (a) The distinct gap between the MSE on the training set and test set indicates high variance issues. (b) The absence of a gap can indicate that the class of the hypothesis function is to simple; the model suffers from high bias issues.

3.4.4 Model Selection and Regularization Path

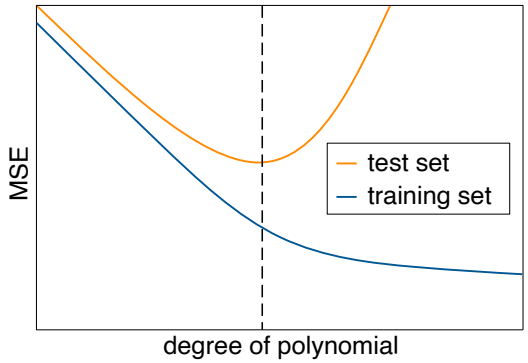
Visualizing the learning curves gives information whether the model suffers from overfitting or underfitting. Overfitting and underfitting can be the consequence of a non-optimal degree of the basis function expansion, or an inappropriately adjusted regularization. Here, the concepts of *model selection* and *regularization path* are helpful tools, which determine the optimal degree of polynomial expansion and regularization. While the model selection method will be used in the application of ridge and lasso regression, the regularization path technique will be also applied in kernel-based regression. By the end of this subsection we introduce practical approaches to avoid overfitting or underfitting.

Model selection is based on the model being prone to overfit or underfit the data as non-optimal degrees of polynomial are chosen. The model performance is determined on the training and the test set for incrementally increased degrees of the polynomial basis function expansion. Consequently the prediction errors are plotted against the polynomial degree as shown in figure 3.12a.

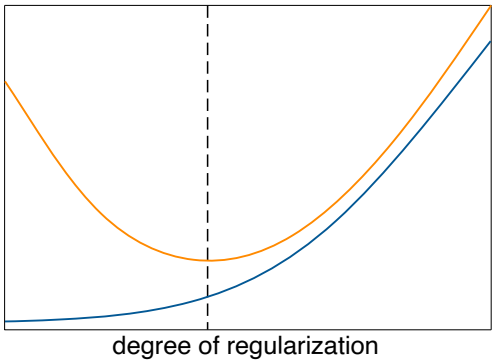
In the case of a polynomial degree being lower than the optimal value, the model underfits the data. This is shown by a high prediction error on the test set and the training set as well. Increasing the polynomial degree, increases the complexity of the class of the hypothesis function, which reduces the bias error, and hence improves the prediction on both datasets. The optimal degree of the polynomial corresponds to the minimum of the prediction error on the test set. As the polynomial degree exceeds the optimal value, the model tends to overfit the data, which is indicated by a further decrease of the prediction error on the training set, while the prediction error on the test set rises.

The regularization path approach is similar to that of the model selection: the prediction errors on the training and the test set are measured for various parameters of the regularization. The model performance then is visualized as a function of the incrementally increased regularization as given by figure 3.12b.

In the case of no regularization the model is more likely to overfit the data, if the class of the hypothesis function is complex enough. Consequently, the prediction error is low on the training set and high on the test set at a low degree of regularization. Increasing the regularization parameter in a step-wise manner reduces the complexity of the hypothesis function, which increases the error on the training set; the error on the test set, though, decreases. The optimal choice of regularization, again, is determined by the minimum of



(a) At low polynomial degrees the model is more prone to underfit the data; at high degrees the model can overfit the data. The optimal degree is in between, where the prediction error on the test set shows a minimum.



(b) Without regularization the model is more prone to overfit the data; high regularization causes the class of the hypothesis function to be simple, causing underfitting. The optimal degree of regularization is determined by the minimal prediction error on the test set.

Figure 3.12: The concept of model selection and regularization path

the prediction error on the test set. Continuing increasing regularization further simplifies the class of the hypothesis function, which causes the errors to rise again on both datasets.

In practice, regularization will affect the optimal degree of polynomial and vice versa. We will treat this two-dimensional optimization problem by performing the model selection at different degrees of polynomial and consequently use the best polynomial degrees to determine the regularization path.

If a model is more prone to overfit the data, getting more training examples is likely to help, since adding new training instances causes the model to better generalize the underlying relation. This in turn reduces the prediction error on the test set. This idea can be also seen in the learning curves in figure 3.11a: introducing more data continues the graph to the right, where the gap between training set and test set eventually narrows down. Unfortunately this can not be used in the presence of high bias issues, since adding new instances will not help to reduce the bias error caused by a simple class of the hypothesis function. Here, adding additional features or increasing the polynomial degree can cause a more complex class of the hypothesis function, which could help avoiding the high bias issues. Choosing a smaller set of features or reducing the polynomial degree, in turn, again could fix high variance problems. Finally, the regularization parameter can be adjusted to force the model using simpler or more complex class of the hypothesis function. There exists no universal approach, though, on how to proceed when a model suffers from high bias or high variance issues, but this has to be chosen individually depending on the actual situation. We summarize these final ideas in table 3.2.

Avoid Overfitting	Avoid Underfitting
getting more training data	-
reducing the set of features	adding more features
reducing the complexity of the class of hypothesis functions	increasing the complexity of the class of hypothesis functions
increasing regularization	reducing regularization

Table 3.2: Approaches which could help to avoid overfitting or underfitting

Chapter 4

Representing and Preparing the Data

Data is the most valuable resource for machine learning; however, it can also be the cause of failures in the machine learning application, as emphasized by figure 3.8 where we have shown the possible sources of error in a machine learning application. Providing and preparing data, hence, is of great importance. This is why we decided to discuss this issue in a separate chapter. In the following part of the thesis we train and test the machine learning model using features, which are obtained directly from the geometry of crystal structures.

The common description of the geometry of crystal structures, also used in standard DFT calculations, is in terms of atomic positions and lattice vectors. This description is discussed in more detail in section 4.1. While DFT codes are able to directly process this information, this crystal structure description cannot be used for the machine learning model, as argued in the beginning of section 4.2. In the same section we discuss the alternative data representation, i.e. the input features, which we use for the machine learning model. In particular, we clarify the features considered for the application of ridge and lasso regression, and the kernel-based methods. In section 4.3 a discussion on the crystal structures is given, followed by the introduction of a database cleaning process. In the final section 4.4 we visually examine the data and investigate correlations in the features.

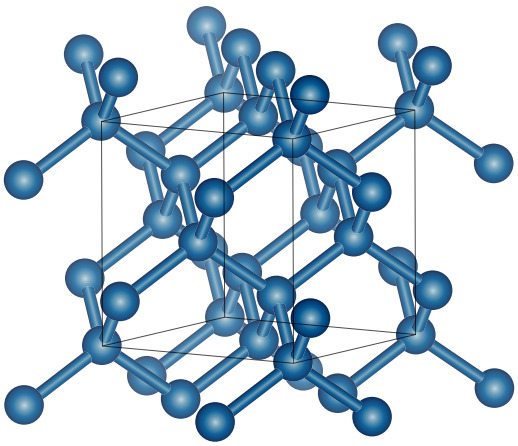


Figure 4.1: A conventional unit cell of the crystal structure of diamond; the unit cell contains eight atoms.

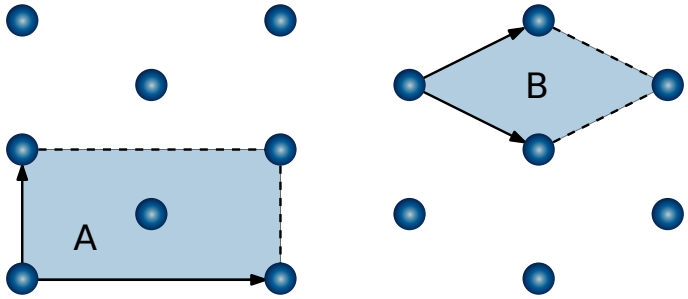


Figure 4.2: Two equivalent representations of the same crystal structure. Unit cell **A** contains two lattice points, whereas unit cell **B** contains a single lattice point. The latter is referred to as the *primitive* unit cell.

4.1 The Crystal Structures

Crystal structures are defined by a crystal lattice and a basis containing one or more atoms, ions or molecules. The crystal lattice is described by the lattice, or basis, vectors: $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{a}_3$. Any point on the crystal lattice can be described by a linear combination of the lattice vectors

$$\boldsymbol{T} = n_1\boldsymbol{a}_1 + n_1\boldsymbol{a}_2 + n_1\boldsymbol{a}_3 \ , \tag{58}$$

where $\boldsymbol{T}$ describes the translational vector pointing to a lattice point. We have already used this notation in the discussion of Bloch's theorem in section 2.4. The parallelepiped, spanned by the three lattice vectors, is called the *unit cell*. In the three-dimensional case, all crystal structures can be reduced to 14 crystal lattices, which are known as the Bravais lattices [23].

Nevertheless, there are no restrictions on the definition of the lattice vectors, as long as their translation covers the entire crystal. In figure 4.2 we show a graphical representation of a two-dimensional crystal lattice. Here, the dots correspond to lattice points, which could be any combination of atoms, ions or molecules. The two definitions **A** and **B** are both valid, as the repetitive translation of their lattice vectors generates the entire crystal lattice. However, definition **B** contains a single lattice point, and its corresponding unit cell (shaded area) is referred to as the *primitive* unit cell. On the other hand, the unit cell **A** is identified as a non-primitive, or conventional unit cell, and includes more than one lattice point of the crystal lattice. A common conventional unit cell of the crystal structure of diamond, containing eight atoms, is shown in figure 4.1.

The position of the atoms inside the unit cell are described by either absolute Cartesian coordinates, or fractional coordinates of the lattice vectors. The latter defines the atomic positions according to

$$\mathbf{R} = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2 + x_3\boldsymbol{a}_3 \ , \tag{59}$$

where x_1, x_2, x_2 are the fractional coordinates, and $0 \leq x_1, x_2, x_2 \leq 1$. A crystal structure is thus described by $3N+9$ coordinates: nine coordinates are needed to define the unit cell by the lattice vectors, and three additional coordinates are used to describe the position of each of the N atoms inside the unit cell.

4.2 Feature Engineering

In this section we introduce the features used in our data representation, starting with a discussion on the general properties required for a crystal structure representation. In the previous section we have explained the common description of crystal structures in terms of unit cells. The $3N + 9$ coordinates, though, are not suitable to serve as features for a machine learning model. This can be clarified by the first and most important property of an ideal data representation for crystal structures:

Property i) invariance with respect to the choice of the unit cell.

In the previous section we have shown that the definition of the unit cell is not unique, i.e. there are infinite ways to describe one and the same crystal structure with different unit cells. The machine learning model would interpret the various descriptions incorrectly and consider them as different structures. Hence, property i) emphasizes that any crystal structure should be reduced to features which always have the same values, independently of the unit cell. This ensures that predictions on new crystal structures are actually improved by training data.

The second important property of an optimal data representation is:

Property ii) translational and rotational invariance.

This is a direct consequence of an obvious physical considerations: translating and rotating a crystal should leave its properties unchanged. This means that the features should exhibit the same value independently on the origin and on the rotation of the unit cell.

In addition,

Property iii) uniqueness,

should be fulfilled, i.e. two different crystal structures should not yield the same feature values.

In particular for the use of the Gaussian kernel the ideal data representation should also satisfy

Property iv) continuity.

For machine learning algorithms based on the Gaussian kernel it is beneficial that crystal structures which have similar energies, have small distances in the feature space. Therefore, two structures i and j which have similar features $\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\| \to 0$, should also show a similar energy $|E^{(i)} - E^{(j)}| \to 0$. Furthermore, small deviations in the geometry of a crystal structure should cause small changes in its features, which can be actually motivated by the continuity of the PES.

Finally, for the data representation used in this work we require

Property v) independence on the structure and system size.

The data representation should be designed to be applicable to various structures and crystal sizes. Consequently, neither the number of features, nor the values of the features should scale with the number of atoms inside the unit cell. This limits the feature engineering to a mean value approach, but considerably increases the number of crystal structures which can be used in the same machine learning application.

The data representation used in this thesis is based on these properties. In addition, we also try to use features which contain physical and chemical information.

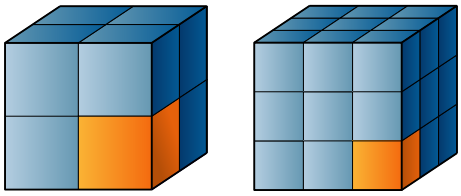


Figure 4.3: The number density determines how many atoms are within a unit volume.

From the available atomic positions $\mathbf{R}_i$ we obtain the radial interatomic distances:

$$d_{ij} = \|\mathbf{R}_i - \mathbf{R}_j\| \, . \tag{60}$$

These quantities provide a useful basis for the further feature engineering, as they directly fulfill the required properties i) and ii). In order to take the crystal periodicity into account, we additionally determine interatomic distances from atoms inside the unit cell to atoms in the neighboring unit cells. For further discussions, we use the index i to denote atoms inside the unit cell and the index j is used for atoms in the super-cell.

In the following subsections we introduce short-range and long-range order features, which will be used for ridge and lasso regression, as well as for the kernel-based methods. We also present a radial distribution function and an angular distribution function, which will be only considered for the kernel-based regression methods.

4.2.1 Number and Packing Density

We use the number density and the packing fraction as a system scale and long-range order measure. Both quantities are intensive quantities, i.e. they are independent on the system size.

The **number density**, here ρ, is a measure for the concentration of atoms in physical space, and is determined by the number of atoms inside the unit cell and the volume of the unit cell:

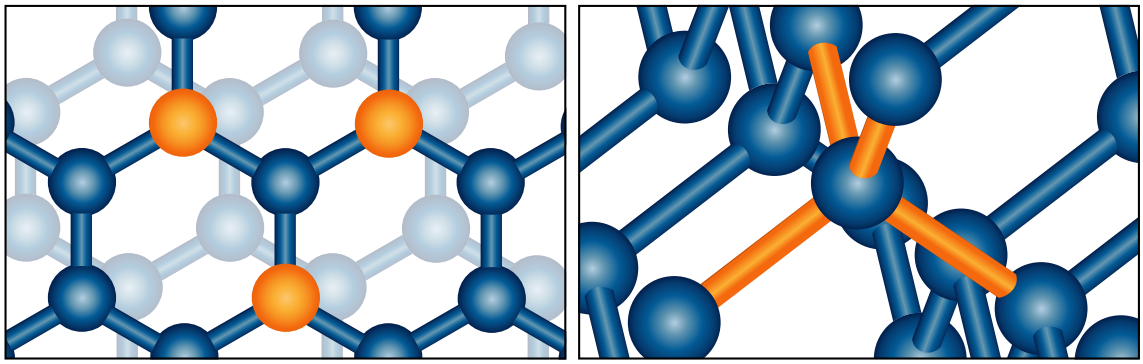
$$\rho = \frac{N}{V_{\text{unit cell}}} \, , \tag{61}$$

where the volume of the unit cell is obtained using the triple product of the lattice vectors

$$V_{\text{unit cell}} = \boldsymbol{a}_1 \cdot (\boldsymbol{a}_2 \times \boldsymbol{a}_3) \, . \tag{62}$$

By the definition, the number density is an isotropic and homogeneous quantity, and represents a measures of how many atoms are within a unit volume, assuming atoms to be of equal size (figure 4.3).

The **packing fraction** is a measure of how much of the unit cell volume is occupied by the volume of the constituent atoms. In its commonly used definition, and assuming equal-sized atoms, the packing fraction again is an isotropic and homogeneous measure, which thus would highly correlate with the number density introduced before. However, we do not assume equal-sized atoms for the packing fraction, in order to reduce the correlation to the number density. In particular, here we consider that the volume of an atom in the unit cell is determined by the distance to the nearest neighbors. We obtain the atomic radii and volumes of atoms from interatomic distances, by successively determining the maximum extent each atom can take in its environment. This approach yields individual atomic



(a) Coordination number (b) Bond length

Figure 4.4: The coordination number determines the number of nearest neighbors in the chemical environment of an atom; the bond length corresponds to the distance to these neighbors.

volumes V_i, which are used to determine the packing fraction using following definition:

$$PF \equiv \frac{\sum_{i=1}^{N} V_i}{V_{\text{unit cell}}} \,. \tag{63}$$

One intention, using the packing fraction in addition to the number density, is to create a broad measure of the density distribution inside the structure: while the number density is independent on the actual atomic positions inside the unit cell, the packing fraction takes different values depending on how closely the atoms are arranged. We assume that this measures are particularly important for unrelaxed structures, where interatomic distances, within the unit cell, can differ considerably.

4.2.2 Coordination Number and Bond Length

The coordination number and the bond length represent the most essential features for ridge and lasso regression. The coordination number and the bond length include short-range information on the chemical environment of atoms at low computational cost.

The coordination number of an atom is equal to the number of neighboring atoms bound to it; the bond length specifies the distance to these atoms. Determining the coordination number in crystals consistently, though, can be challenging due to varying bond lengths in the environment of atoms. Here, we use a self-consistent method, introduced in ref. [49], which yields the effective coordination number and average bond length.

The effective coordination number is a fractional number of nearest neighbors, which can be seen as a measure of how many atoms effectively are located in the environment of an atom. In the following, we omit the expression effective for the coordination number.

The algorithm starts with an initial guess of the average bond length $\bar{d}_i$ and iteratively updates the quantity to the actual average bond length of the i-th atom. Using the interatomic distances d_{ij} from (60), we update the average bond length of the i-th atom according to

$$\bar{d}_i = \sum_j d_{ij} p_{ij}, \quad p_{ij} = \frac{e^{f(d_{ij})}}{\sum_j e^{f(d_{ij})}}, \quad f(d_{ij}) = \left[1 - \left(\frac{d_{ij}}{\bar{d}_i}\right)^6\right] \,. \tag{64}$$

Once the average bond length has converged, we use it to determine the coordination

number of the i-th atom using

$$C_i = \sum_j e^{f(d_{ij})} \, . \tag{65}$$

This procedure is performed on the N atoms in the unit cell. To ensure the independence on the system size, we average over the obtain quantities:

$$\bar{d} = \frac{1}{N} \sum_{i=1}^{N} \bar{d}_i, \quad \bar{C} = \frac{1}{N} \sum_{i=1}^{N} C_i, \quad \sigma_C = \sqrt{\frac{\sum_{i=1}^{N} (C_i - \bar{C})^2}{N}} \, . \tag{66}$$

We additionally determine the standard deviation of the coordination number to characterize its distribution in the structure.

Furthermore, we use the same algorithm to determine the effective number and average distance of second nearest neighbors in the structure. This is achieved by omitting the interatomic distances d_{ij}, which correspond to first nearest neighbor distances.

4.2.3 Radial Distribution Function

Various definitions of a radial distribution function [14, 15] have been used as data representation for kernel-based methods. In this subsection we introduce a radial distribution function for the machine learning application on mono-elemental crystal structures.

In general, the radial distribution function determines the ratio between the local number density $\rho(r)$, at the distance r from a reference atom, and the bulk number density ρ (61),

$$g(r) = \frac{\rho(r)}{\rho} \, . \tag{67}$$

The local number density $\rho(r)$ is determined by the number of atoms within a spherical shell of radius r and infinitesimal thickness dr, centered at a reference atom. The picture on the left-hand side of figure 4.5 visually illustrates this definition. For increasingly large distances r, the local number density converges to the bulk number density, hence, $\lim_{r\to\infty} g(r) = 1$.

The local number density with respect to atom i can also be obtained from the interatomic distances d_{ij}, using

$$\rho_i(r) = \frac{1}{V_r} \sum_j \theta(d_{ij} - r)\theta(r + dr - d_{ij}) \, , \tag{68}$$

where dr denotes the thickness and V_r the volume of the sphere shell. Averaging over the atoms and using the definition of the number density (61), we obtain the expression:

$$g(r) = \frac{V_{\text{unit cell}}}{N^2} \frac{1}{V_r} \sum_{i=1}^{N} \sum_j \theta(d_{ij} - r)\theta(r + dr - d_{ij}) \, . \tag{69}$$

This is a continuous function, which has to be discretized in order to be used as an input feature. Extending the infinitesimal thickness of the sphere to a finite thickness $dr \to \Delta r$, corresponding to the bin size, simply achieves the desired intent. Furthermore, for the practical use of the radial distribution function a cut-off distance $r_{\text{cut-off}}$ has to be used. Both adjustable quantities, the bin size Δr and the cut-off distance $r_{\text{cut-off}}$, have to optimized directly in the machine learning application, as their optimal value depends on the crystal structures under study.

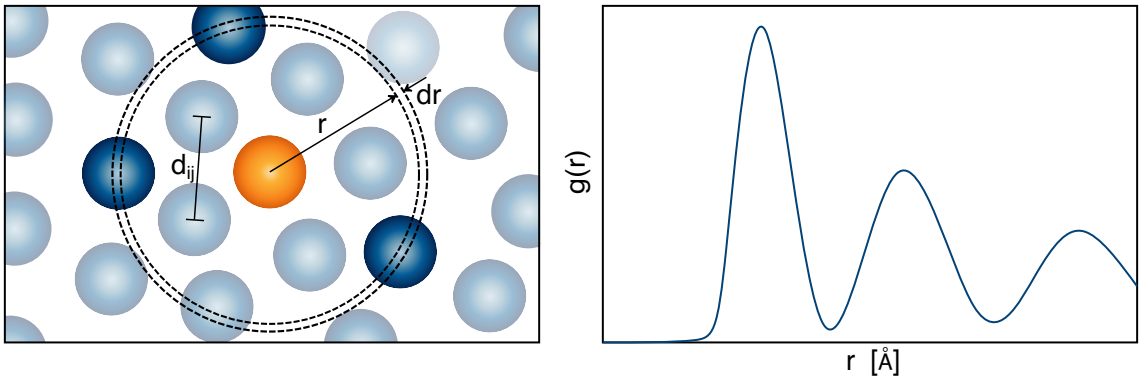


Figure 4.5: The radial distribution function determines the ratio between the local number density $\rho(r)$ and the bulk number density ρ. The local number density corresponds to the number of atoms within a spherical shell of infinitesimal thickness dr. In crystal structures the radial distribution function shows distinct peaks the nearest neighbor distances.

Crystal structures are highly ordered structures, hence, their radial distribution function shows distinct peaks at the distance of nearest neighbors. Due to the Heaviside step functions in equation (69) this characteristic violates the desired continuity of the radial distribution function.

We apply a Gaussian smoothing on interatomic distances, $\delta(d - d_{ij}) \to \mathcal{N}(d_{ij}, \sigma^2_{\text{rdf}})$ with free smoothing parameter σ_{rdf}. As a consequence, the Heaviside step functions can be replaced by an integral over the bin ranges. We arrive at the final expression for the radial distribution function used in our machine learning application:

$$g(r) = \frac{V_{\text{unit cell}}}{N^2} \frac{1}{V_r} \sum_{i=1}^{N} \sum_{j} \int_{r}^{r+\Delta r} \frac{1}{\sqrt{2\pi\sigma^2_{\text{rdf}}}} \exp\left(-\frac{(r - d_{ij})^2}{2\sigma^2_{\text{rdf}}} \right) . \tag{70}$$

The optimal degree of smoothing, again, has to be adjusted in the machine learning application.

4.2.4 Angular Distribution Function

In this subsection we introduce an angular distribution function that provides information on the bond angles in crystal structures. In a system with N atoms the total number of angles between any three of them is $N(N-1)(N-2)/2$. However, for our machine learning application we only consider angles for nearest neighbors, as these angles are sufficient to determine the structure of a system.

The angle formed by atom j and atom k, measured at central atom i, is determined by the the dot product

$$\vartheta_{jik} = \frac{180°}{\pi} \cos^{-1}\left(\frac{\mathbf{r}_{ij} \cdot \mathbf{r}_{ik}}{\|\mathbf{r}_{ij}\| \|\mathbf{r}_{ik}\|} \right) , \tag{71}$$

where $\mathbf{r}_{ij}$ and $\mathbf{r}_{ik}$ denote the position vectors, pointing from atom i towards atom j and k. To simplify further discussions on the angular distribution function, we express angles in degrees.

Based on the angular distribution function introduced in ref. [16], we additionally

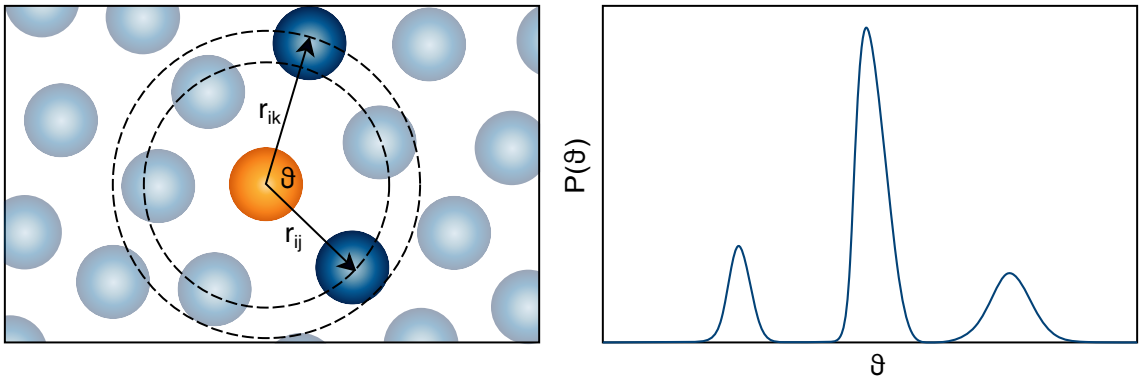


Figure 4.6: The angular distribution function determines the bond angles between nearest neighbors. Here, the angular distribution function is described by a probability density $P(\vartheta)$.

introduce a weighting factor for angles

$$w(\vartheta_{jik}) = \frac{1}{d_{ij}^4 d_{ik}^4} \,, \tag{72}$$

where d_{ij} are the corresponding interatomic distances, or the lengths of the position vectors $d_{ij} = \|\mathbf{r}_{ij}\|$. This definition adds a weight to angles, which is higher for close-range atoms.

Again, we use a Gaussian smoothing on angles, i.e. $\delta(\vartheta - \vartheta_{jik}) \to \mathcal{N}(\vartheta_{jik}, \sigma_{\text{adf}}^2)$, in order to ensure continuity of the angular distribution function. We discretize the angular distribution function by binning the angles into a histogram, using a similar approach as for the radial distribution function. By definition (71), angles occur in the range of $0° \geq \vartheta \geq 180°$; therefore, we apply periodic boundary conditions, in order to do not neglect any contributions outside the range, induced by the Gaussian smoothing process. Furthermore, by using a weighting factor and considering only angles between nearest neighbors, the amplitude of the resulting distribution does not have any meaning, hence, we normalize the distribution for consistency. Our angular distribution function, thus, equals a weighted probability density, given by

$$P(\vartheta) = \frac{1}{A} \sum_{i=1}^{N} \sum_{\substack{<j,k>_{nn} \\ j \neq k \neq i}} w(\vartheta_{jik}) \int_{\vartheta}^{\vartheta+\Delta\vartheta} \frac{1}{\sqrt{2\pi\sigma_{\text{adf}}^2}} \exp\left(-\frac{(\vartheta - \vartheta_{jik})^2}{2\sigma_{\text{adf}}^2}\right) \,, \tag{73}$$

where A refers to the normalization factor and $< j, k >_{nn}$ denotes all possible nearest neighbor pairs of atom i. The angular distribution function, as the radial distribution function, has to be optimized in the machine learning application. The angular distribution function, though, has to be solely optimized in terms of the bin size $\Delta\vartheta$ and the smoothing parameter σ_{adf}, and does not have any cut-off quantities due to its periodicity.

4.3 Structure Generation and Database Cleaning

In this thesis we consider mono-elemental crystal structures of carbon and boron. The carbon and boron crystal structures, though, are considered separately for the machine learning applications. Carbon and boron are well-known for their polymorphism, i.e. the ability to exist in more than one crystal structure. Carbon has this behavior due to orbital hybridization, while boron is able to form multiple bonds due to electron deficiency. As

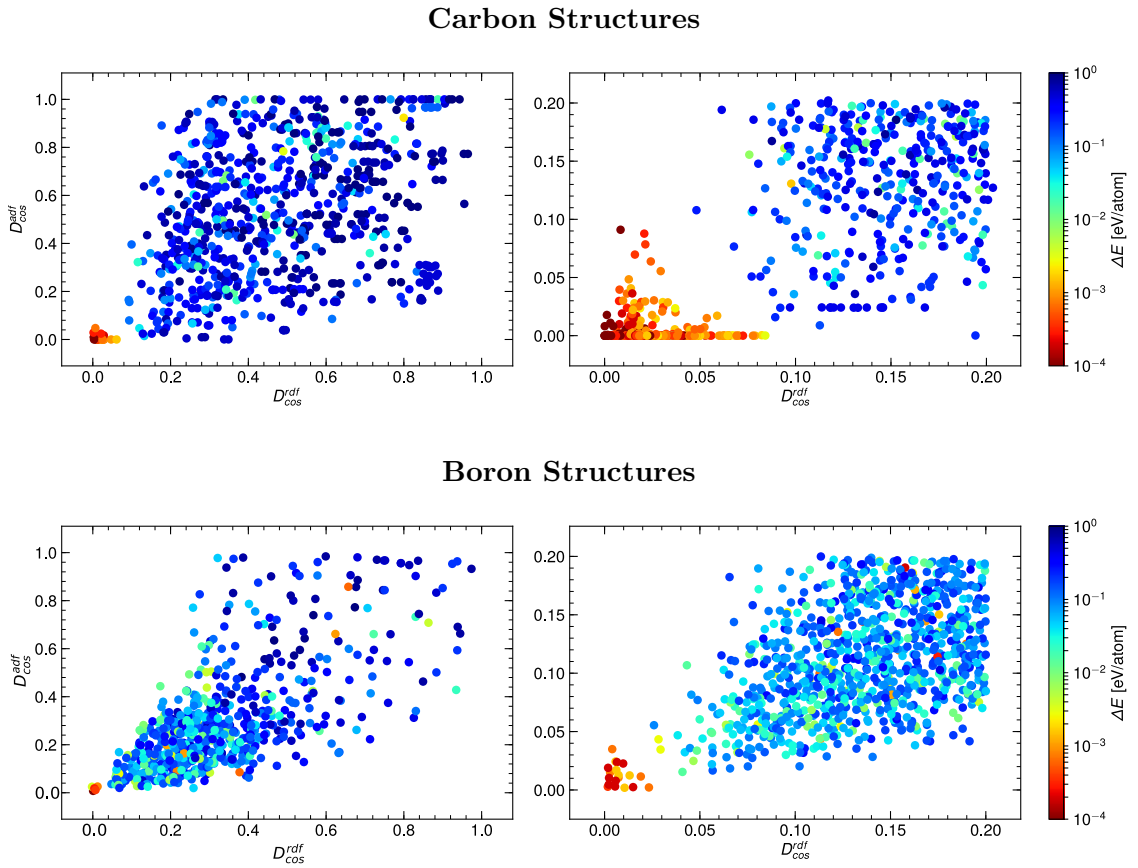


Figure 4.7: Fingerprint distances using the radial distribution function and the angular distribution function. Points in the lower left corner correspond to similar crystal structures. We discard duplicates to improve the quality of the database.

a consequence, the two crystalline systems provide a large number of complex crystal structures, which at the same time are an ideal application and test case for our data representation. In fact, their ability to form clusters with low internal symmetry renders the energy prediction on these structures challenging.

As discussed in section 2.5, we have used random structure generation and minima hopping [4] methods to generate our crystal structures for carbon and boron. From both methods we have obtained relaxed structures, as well as unrelaxed structures. The crystal structures are of varying system sizes, starting from two atoms per unit cell to 24 atoms per unit cell.

We have repeated DFT calculations with a certain choice of the energy cut-off and the Brillouin zone sampling, after we had obtained the data from the two structure prediction methods. This has reduced any deviations in the internal energies due to differently used DFT settings in the crystal structure prediction methods. Furthermore, we have performed the final DFT calculations with an estimated error of 1-10 meV/atom. With our machine learning model we want to achieve comparable prediction accuracies.

The crystal structures have been obtained from multiple generation runs, hence, the same crystal structure may appear more than once in the database, i.e. the database contains numerous duplicates. Here we want to emphasize again, that the machine learning performance strongly depends on the quality of the data. Hence, the data should be as good in quality as possible, hence, a typical approach is to clean the available database, and discard bad data points, for instance outliers and duplicates. This process involves either manually checking for bad data or the implementation of an algorithm, which automatically filters and cleans the data. The latter is used in this thesis and will be introduced in the following discussion.

Measuring the similarity between crystal structures has always been a concern in materials science, and yet no clearly defined method exists. Several fingerprint measures [33,34] have been developed, to determine whether two crystal structures can be considered equal or not. In this thesis, though, we use the simple definition of cosine distances [35].

First, we construct two fingerprint vectors $\boldsymbol{F}_{\text{rdf}}$ and $\boldsymbol{F}_{\text{adf}}$ for each structure. We use the radial distribution function (70) and the angular distribution function (73) to construct the vectors according to

$$\boldsymbol{F}_{\text{rdf}} = \begin{pmatrix} g(r_1) \\ g(r_2) \\ \vdots \end{pmatrix} \quad , \quad \boldsymbol{F}_{\text{adf}} = \begin{pmatrix} P(\vartheta_1) \\ P(\vartheta_2) \\ \vdots \end{pmatrix} \ , \tag{74}$$

where investigations have shown that for our purpose the optimal choice of bin sizes and free parameters is:

$$\Delta r = 0.05\text{Å} \ , \quad r_{\text{cut-off}} = 10\text{Å} \ , \quad \sigma_{\text{rdf}} = 0.025\text{Å} \ , \quad \Delta\vartheta = 5^\circ \ , \quad \sigma_{\text{adf}} = 2.5^\circ \ . \tag{75}$$

We define the cosine function, which measures the similarity between fingerprint vectors of structure i and j, by

$$D_{\cos}(\boldsymbol{F}^{(i)}, \boldsymbol{F}^{(j)}) \equiv 1 - \frac{\boldsymbol{F}^{(i)} \cdot \boldsymbol{F}^{(j)}}{\|\boldsymbol{F}^{(i)}\| \|\boldsymbol{F}^{(j)}\|} \ . \tag{76}$$

According to this definition, two similar structures yield a value close to $D_{\cos} = 0$, while structures that completely differ from each other have $D_{\cos} = 1$.

We determine the $D_{\cos}$ values for all structure pairs; the results are shown in the

Database	Number of structures
Carbon − Minima	1400
Carbon − Mixed	7500
Boron − Minima	3500
Boron − Mixed	7500

Table 4.1: Databases for the machine learning application. The *Minima* databases contain relaxed structures; the *Mixed* databases additionally include unrelaxed structures.

scatter plot 4.7, where the difference in energies $\Delta E = |E^{(i)} - E^{(j)}|$ is additionally shown as color code. We identify structure pairs on the lower left corner as duplicates and set the following threshold for duplicates: if a structure pair shows

$$\text{Carbon: } (D_{\cos}^{\text{rdf}} < 0.10) \text{ AND } (D_{\cos}^{\text{rdf}} < 0.10) \text{ AND } (\Delta E < 10 \text{ meV/atom})$$

$$\text{Boron: } (D_{\cos}^{\text{rdf}} < 0.05) \text{ AND } (D_{\cos}^{\text{rdf}} < 0.05) \text{ AND } (\Delta E < 10 \text{ meV/atom})$$

we consider one of the two structures to be a duplicate and discard the structure from the database.

Using this definition and data cleaning algorithm we discard all duplicates from the database, to increase the quality of the data. To visualize the effect of this cleaning process, we plot the energy distribution of structures in the database before and after the data cleaning. The two situations are shown in figure 4.8 The energy distribution before the data cleaning clearly suffered from skewness, i.e. the distribution is asymmetric, which gradually tapers at one side and falls off rapidly at the other side. A characteristic of this kind should be avoided in database, since it negatively affects the model weights in the training process, resulting in a bad model performance.

Finally, we want to study the effect of including relaxed and unrelaxed crystal structures on the machine learning application. For this, we additionally create a second database for the carbon and boron structures, which only contains the relaxed, i.e. the minima, structures. We label this database as *Minima*. The database containing the relaxed and the unrelaxed structure we label as *Mixed*. In table 4.1 we give an overview of the datasets with the final number of crystal structures included.

4.4 Data Monitoring

Data visualization is an useful concept to gain insights into large datasets. However, these concepts are more effective for the use of ridge and lasso regression, than for the kernel-based methods: ridge and lasso regression directly use the features in the hypothesis function (31), while kernel-based methods use the features for determining the dual variables in the hypothesis function (47). The former approach allows a direct visualization of correlations between features and the target value. In the case of kernel-based methods, there is basically no point in doing so, since feature correlations have less effect on the model performance.

We use scatter plots to visualize the correlation of the features considered for ridge and lasso regression. For this specific purpose we only use the mixed databases as the inclusion of unrelaxed structures gives a better representation of the data point distribution. Furthermore, we only choose a reduced number of randomly chosen data points for the

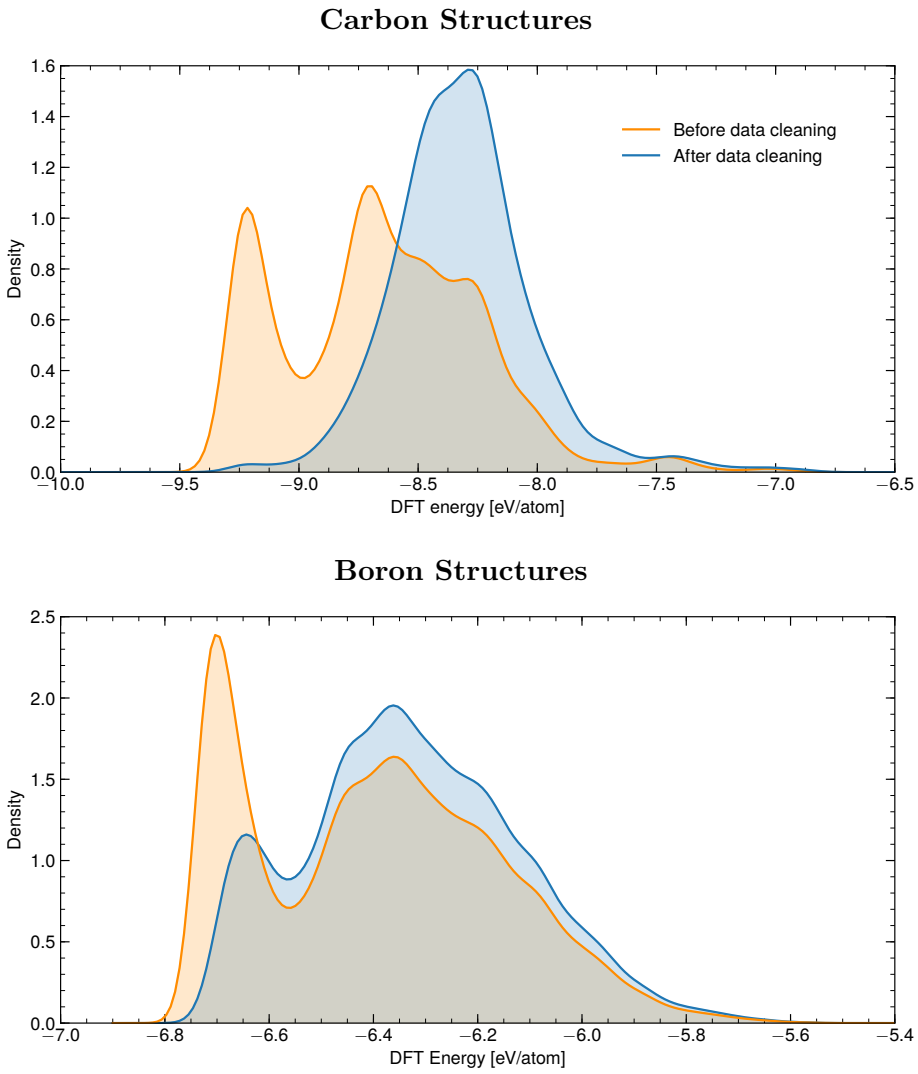


Figure 4.8: The energy distribution in the datbase before and after we have cleaned it. The distribution containing duplicates shows a skewed behavior, which is reduced by the data cleaning process.

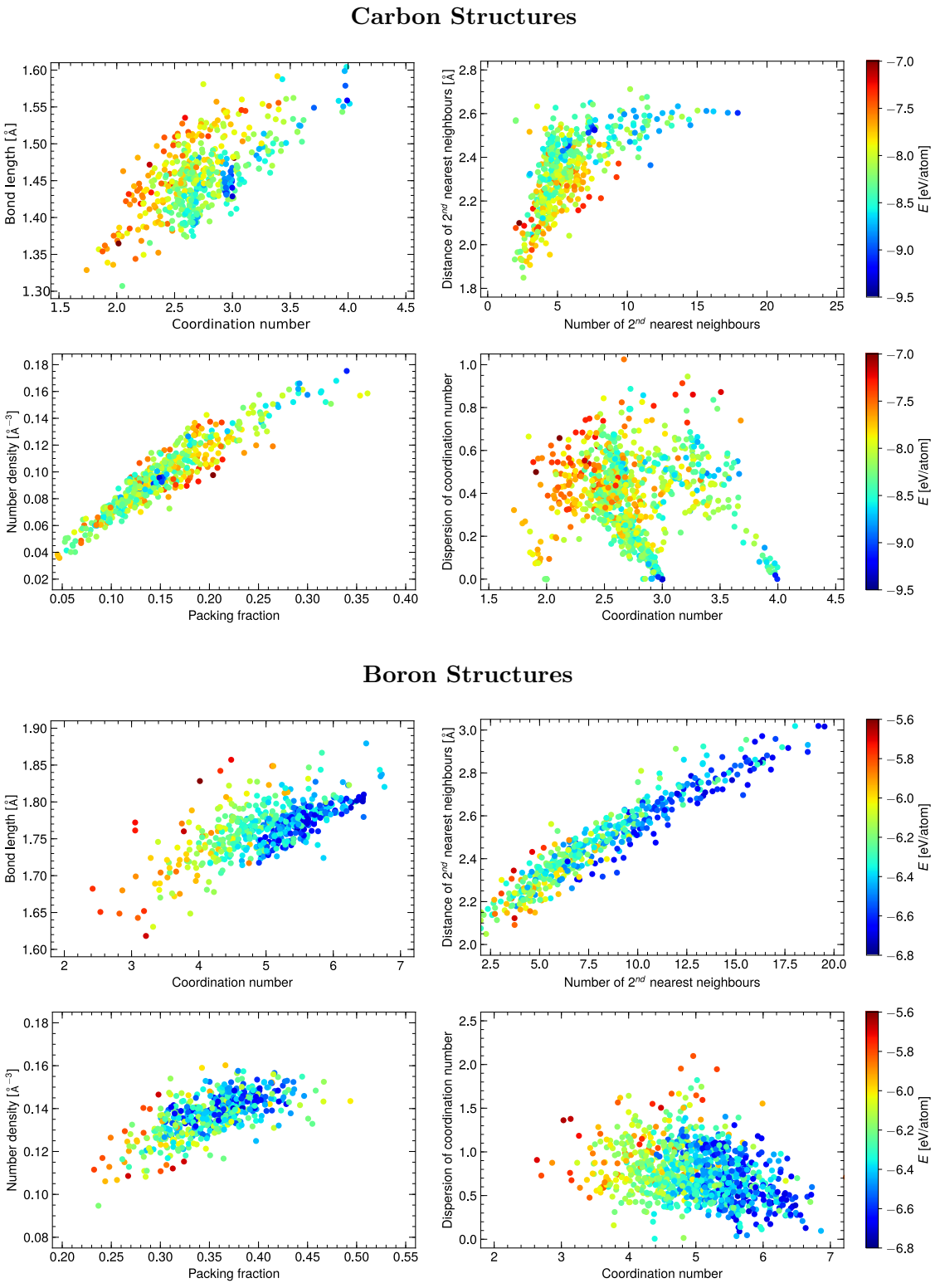


Figure 4.9: Visualizing correlations of features among each other, and with the internal energy. Scatter plots of features, where points are aligned diagonally, usually indicate a high correlation. Our features highly correlate with the internal energies.

visualization. Although any combination of two features can be used to create a scatter plot, we use a certain selection of feature pairs, shown in figure 4.9.

From the plots it is evident, that the coordination number and the bond length (upper left panel) contain good information on the energies of carbon structures and boron structures as well. The coordination number in combination with the dispersion in the coordination number (lower right panel) also provides a useful measure for the energies. We see that the packing fraction and the number density (lower left panel) correlate more in carbon structures than in boron structures. This can be seen because the points tend to align along the plot diagonal. Likewise the number and distance of second nearest neighbors (upper right panel) seem to be more correlated for boron structures than for carbon structures. Nevertheless, we assume the packing fraction, the number density, as well as the number and distance of second nearest neighbors to be still useful for the machine learning application, as their specific patterns clearly show a correlation to the energies.

Chapter 5

Learning Energies with Ridge and Lasso Regression

In this chapter the crystal structure datasets are used to train a machine learning model on the internal energies using ridge and lasso regression. We have discussed and preprocessed the data in chapter 4. In particular, we have discarded duplicates and obtained features from the crystal structures to serve as an input for the machine learning model. Table 5.1 gives a list of the features used for ridge and lasso regression. The crystal structure datasets are considered separately in order to study the performance of the machine learning model with regard to relaxed and mixed – i.e. relaxed and unrelaxed – structures, as well as in dependence on carbon and boron structures. Furthermore, we use ridge regression and lasso regression to study the effect of ℓ^1- and ℓ^2-regularization.

In the first half of this chapter we improve the performance of the machine learning model using basis function expansions and regularization. The optimization includes the adjustment of two parameters: the degree of polynomial of the basis function expansion and the regularization parameter. In section 5.1 we use the model selection approach to optimize the degree of polynomial. The regularization path method is used in section 5.2 to obtain the optimal regularization parameter. We perform the model selection at three different levels of regularization, since the degree of polynomial and regularization affect over- and underfitting simultaneously. Subsequently, three degrees of polynomial are used in the regularization path approach.

In the second half of this chapter we use the model at optimal settings to obtain the learning curves in section 5.3. As final measurements the model's predictive power is rated in section 5.4.

Features used
Number density
Packing fraction
Coordination number
Dispersion in the coordination number
Bond length
Number of second nearest neighbors
Distance of second nearest neighbors

Table 5.1: Features used for the application of ridge and lasso regression.

Polynomial degree d	Number of features N
1	8
2	36
3	120
4	330
5	792
6	1716
7	3432

Table 5.2: Number of features N in the ML application at polynomial degree d. Lasso regression may find a sparse selection of features effectively used.

The machine learning algorithms are implemented using the `sci-kit learn` library for python [40]. The features are scaled using the `StandardScaler` and basis function expansions are applied using the `PolynomialFeatures` function. We use the mean squared error to compare the model performance during the optimization process and the learning curves. In the last section of this chapter we use various scoring parameters to analyze the predictive power of the model.

5.1 Model Selection

The model selection approach has been introduced in section 3.4.4. We apply a polynomial basis function expansion on the features to provide a more complex class of the hypothesis function, which the model can use to estimate the energies. Polynomial degrees higher than the optimal cause overfitting; degrees lower make the class of the function too simple, which results in underfitting issues. High polynomial degrees, in addition, cause an explosion of the features, which is illustrated for our specific application in table 5.2.

We stepwisely increase the degree of polynomial to search for the transition from underfitting to overfitting and hence to find the optimal degree. The performance is measured at each step using cross-validation with 50 repetitions. As soon as the model clearly suffers from overfitting further measurements at higher degrees are omitted. Here, we determine overfitting by an increased error on the test set and furthermore by a high variance in the sequence of performances. The influence of regularization is taken into account by performing the model selection at three different degrees of regularization. The levels of regularization are chosen according to low, intermediate and high regularization.

The model selection curves using ridge regression are shown in figure 5.1. The results from the application of lasso regression are given by figure 5.2. The plots outline the measured mean squared error as a function of the polynomial degree. The lowest prediction error, measured on the test set, is indicated by a horizontal line to comparably show the best performance at each regularization level. It is evident that the significant rise in the number of features causes a rapid transition to overfitting, especially using ridge regression. Lasso regression is able to partly suppress this behavior. In the case of minima structures, optimal performances are obtained at $d = 2$ and $d = 3$. These values are consistent for carbon and boron structures, as well as for both regression methods. In contrast, the hypothesis function for the mixed structures has to be more complex, shown by greater optimal degrees of the polynomial. Here, $d = 4$ and $d = 5$ achieve the lowest errors. For the regularization path approach in the next section we additionally consider $d = 4$ in the

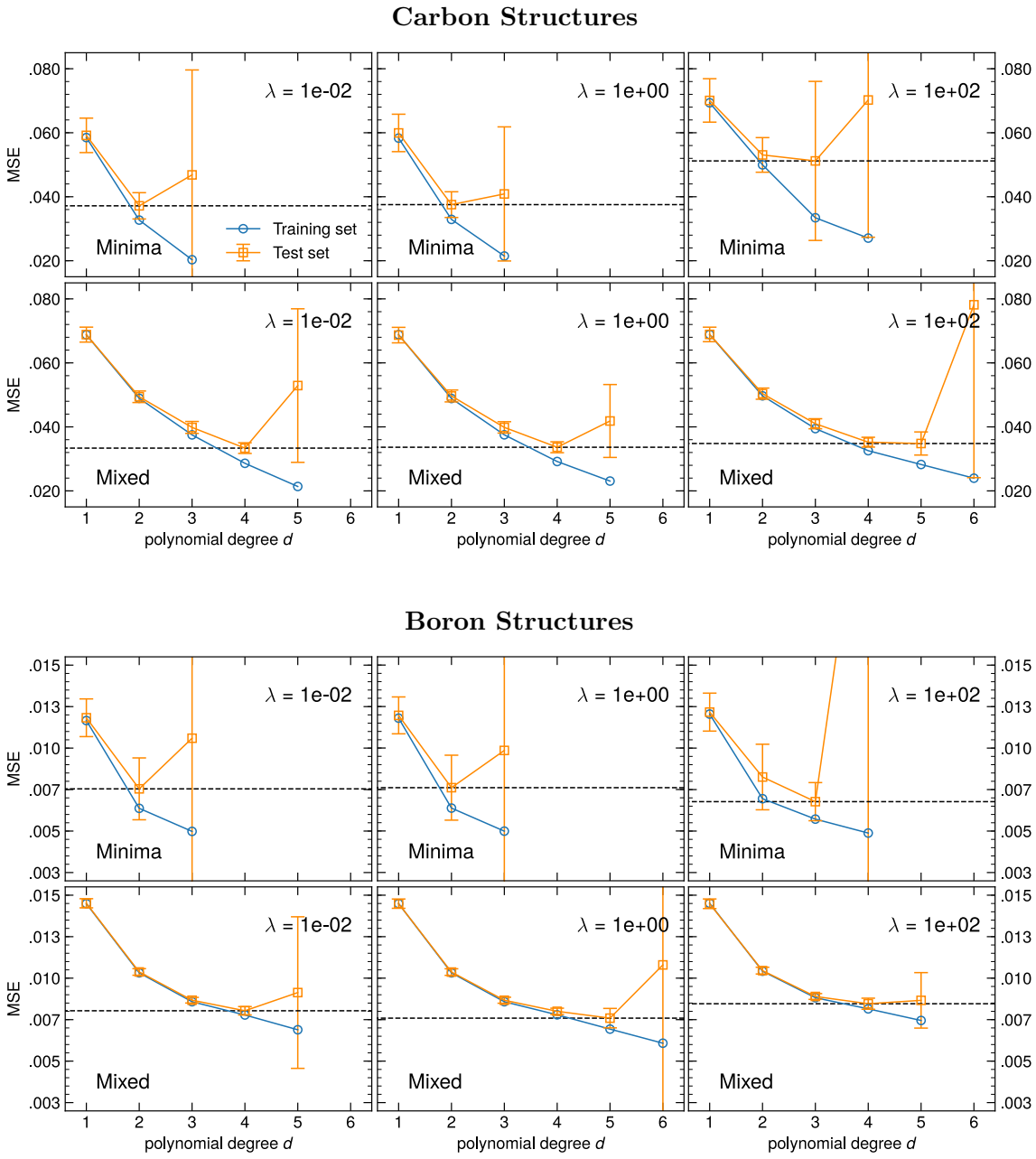


Figure 5.1: Model selection using ridge regression at low (λ = 1e−02), intermediate (λ = 1e+00) and high (λ = 1e+02) regularization. The lowest prediction error determined on the test set is highlighted by a horizontal line.

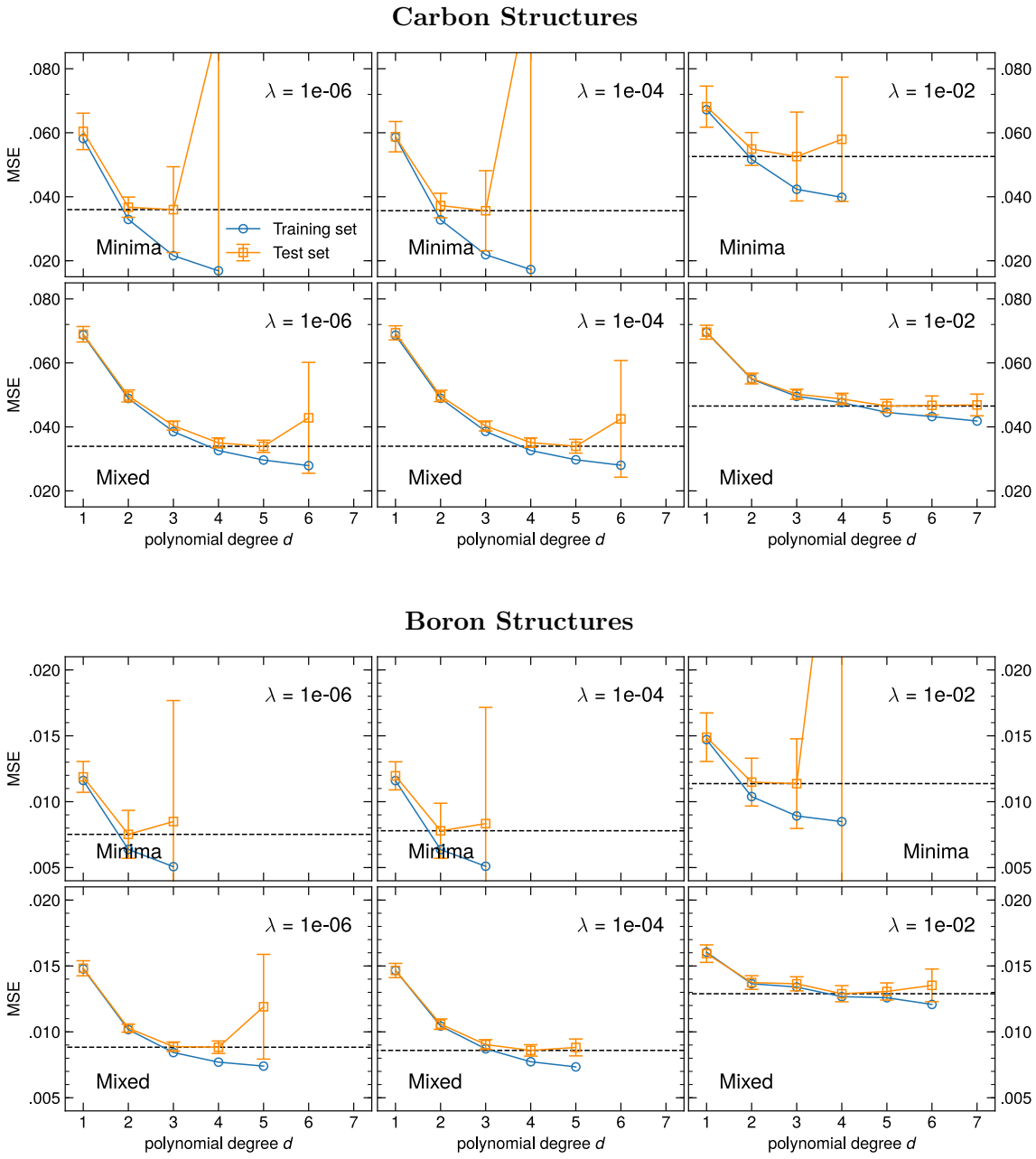


Figure 5.2: Model selection using lasso regression at low (λ = 1e−06), intermediate (λ = 1e−04) and high (λ = 1e−02) regularization. The lowest prediction error determined on the test set is highlighted by a horizontal line.

Database	Regression method	d	λ
Carbon − Minima	Ridge	2	1e+00
	Lasso	2	1e−03
Carbon − Mixed	Ridge	4	1e+00
	Lasso	5	1e−04
Boron − Minima	Ridge	2	1e+00
	Lasso	2	1e−03
Boron − Mixed	Ridge	4	1e+00
	Lasso	5	1e−04

Table 5.3: Optimal values of the polynomial degree d and the regularization parameter λ using ridge and lasso regression.

case of minima structures and $d = 6$ for the mixed case, as high regularization may reduce the overfitting issue.

5.2 Regularization Path

Regularization affects over- and underfitting at a more adjustable level, compared to the choice of discrete degrees of polynomial. This is possible because the regularization parameter is adjustable at a continuous range of values.

We have obtained three optimal degrees of polynomial in the previous section, which are used for the regularization path approach in this section. Cross-validation with 50 repetitions is used and regularization is chosen in the transition range from high regularization to low regularization. Reducing regularization is stopped as the model clearly suffers from overfitting issues.

The regularization paths are given by figure 5.3 for ridge regression and 5.4 for lasso regression. Here, the positive x-direction corresponds to the transition from low regularization to high regularization. The lowest error achieved on the test set, again, is marked with a horizontal line. At first it becomes apparent that regularization could not fix the overfitting issues at $d = 4$ in the minima and $d = 6$ in the mixed case. This is shown by high and unstable prediction errors on the test set. Overfitting due to numerous features, hence, overwhelms the effect of regularization. Neither can ℓ^1-regularization reduce the number of features in lasso regression, and thus suppress the overfitting issues. In terms of ridge regression, the model seems to achieve the lowest prediction errors at a polynomial degree of $d = 2$ for the minima and $d = 4$ for the mixed structures. Accordingly, we determine these degrees in combination with a regularization $\lambda = 1e{+}00$ as optimal settings. Lasso regression, in contrast, achieves the best performances on mixed structures at $d = 5$ and regularization $\lambda = 1e{-}04$; consequently this parameters are determined as optimal settings. In the event of minima structures we determine the optimal settings with parameters $d = 2$ and $\lambda = 1e{-}03$, since the model suffers from overfitting at $d = 3$. We summarize the optimal settings in table 5.3.

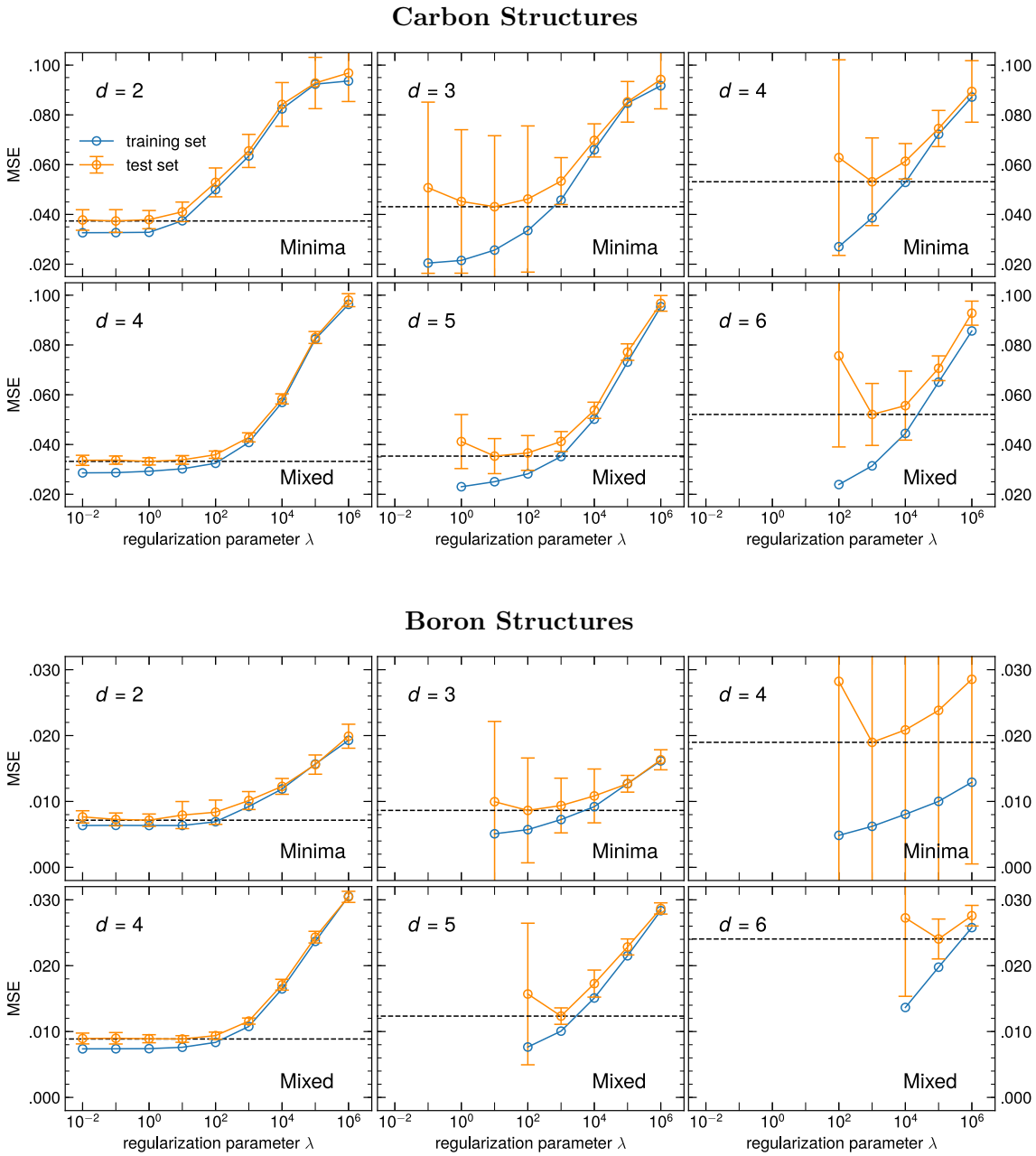


Figure 5.3: Regularization path using ridge regression at different polynomial degrees. The lowest prediction error determined on the test set is highlighted by a horizontal line.

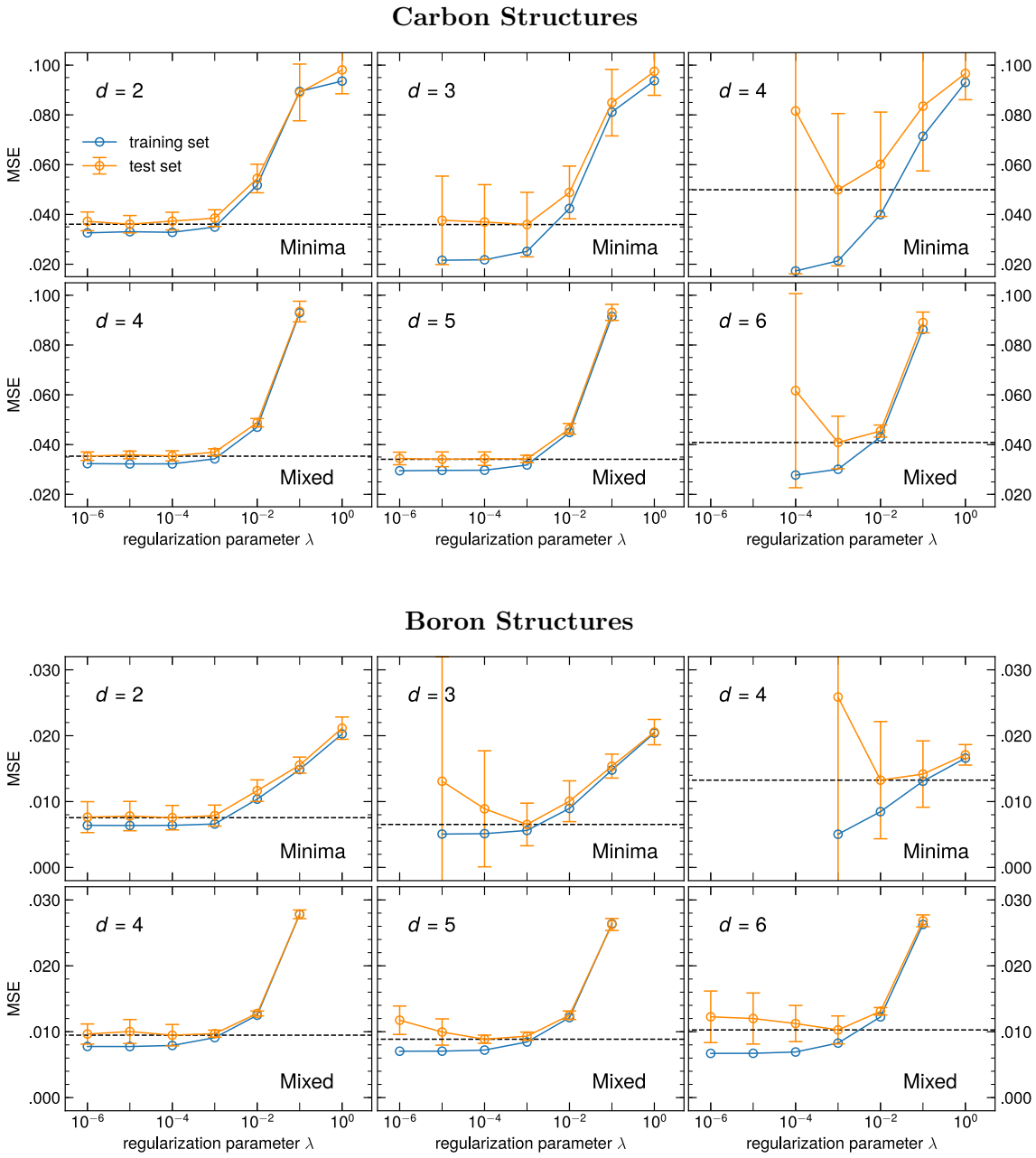


Figure 5.4: Regularization path using lasso regression at different polynomial degrees. The lowest prediction error determined on the test set is highlighted by a horizontal line.

5.3 Learning Curves

The learning curve method is a widely used diagnostic tool, which has been introduced in section 3.4.3. Learning from a training dataset incrementally yields useful information on the training process and the final model performance. Here it should be mentioned that the course of the learning curves is clearly defined by our specific choice of parameters. We determine the model curves at settings listed in table 5.3 using again cross-validation with 50 repetitions.

The obtained learning curves are given by figure 5.5, in which results using ridge regression can be found at the left-hand side and the learning curves from lasso regression are shown on the right. Horizontal lines indicate the final prediction errors on the test set and the training set, highlighting the gap between the errors at the maximum available data. In the training process no distinct difference between ridge regression and lasso regression can be observed. This is consistent with the mentioned statement, that ridge and lasso regression yields equal results, when both methods are optimized in the parameters. Furthermore, this validates that we have determined the optimal parameters for the two methods. In the case of carbon structures, the model performance on the test and the training set converges moderately towards equal prediction errors within a reasonable error range. This indicates that the model suffers neither from overfitting, nor underfitting. It can be observed that boron structures yield a similar performance The learning curves in case of the minima boron structures, though, show a different behavior: the model learns more rapidly and errors flatten out towards larger training sets, which would allow an improvement of the complexity of the class of the hypothesis function. In the previous section neither increasing the polynomial degree, nor reducing the regularization could improve the performance on the test set; therefore we assume that the features in table 5.1 are missing information, which is needed to learn and estimate the energies of minima boron crystals more accurately and consistently.

5.4 Model Performance and Discussion

The application of ridge and lasso regression has been optimized and analyzed in previous sections. The optimization was done with respect to the datasets and the regression methods. In this last section of the chapter using ridge and lasso regression, the optimal parameters are used to make final predictions. Here, the model performance is measured only on the test set and rated using multiple scoring parameters. For the final predictions we use the model parameters listed in table 5.3 and cross-validation with 100 repetitions to measure the performance.

Figure 5.6 gives a visual representation of predicted machine learning energies, compared to energies obtained from density function theory. The corresponding scoring parameters can be found in table 5.4. Contours in the background represent the model's prediction tendency at that certain energy domain. The dots correspond to sample predictions; their density in the case of the mixed structures is consistent with the distribution given in figure 4.8. Thin lines, above and below the solid line, correspond to a 2%- and 5%-deviation from the DFT energies. The inset shows a normalized distribution of the residual errors

Again, neither the prediction plots, nor the scoring parameters show a distinct difference between ridge and lasso. This is why we assume that an optimized ℓ^1- and ℓ^2-regularization yields the same prediction performance on our crystal structures and none of both has an advantage over the other in terms of prediction accuracy.

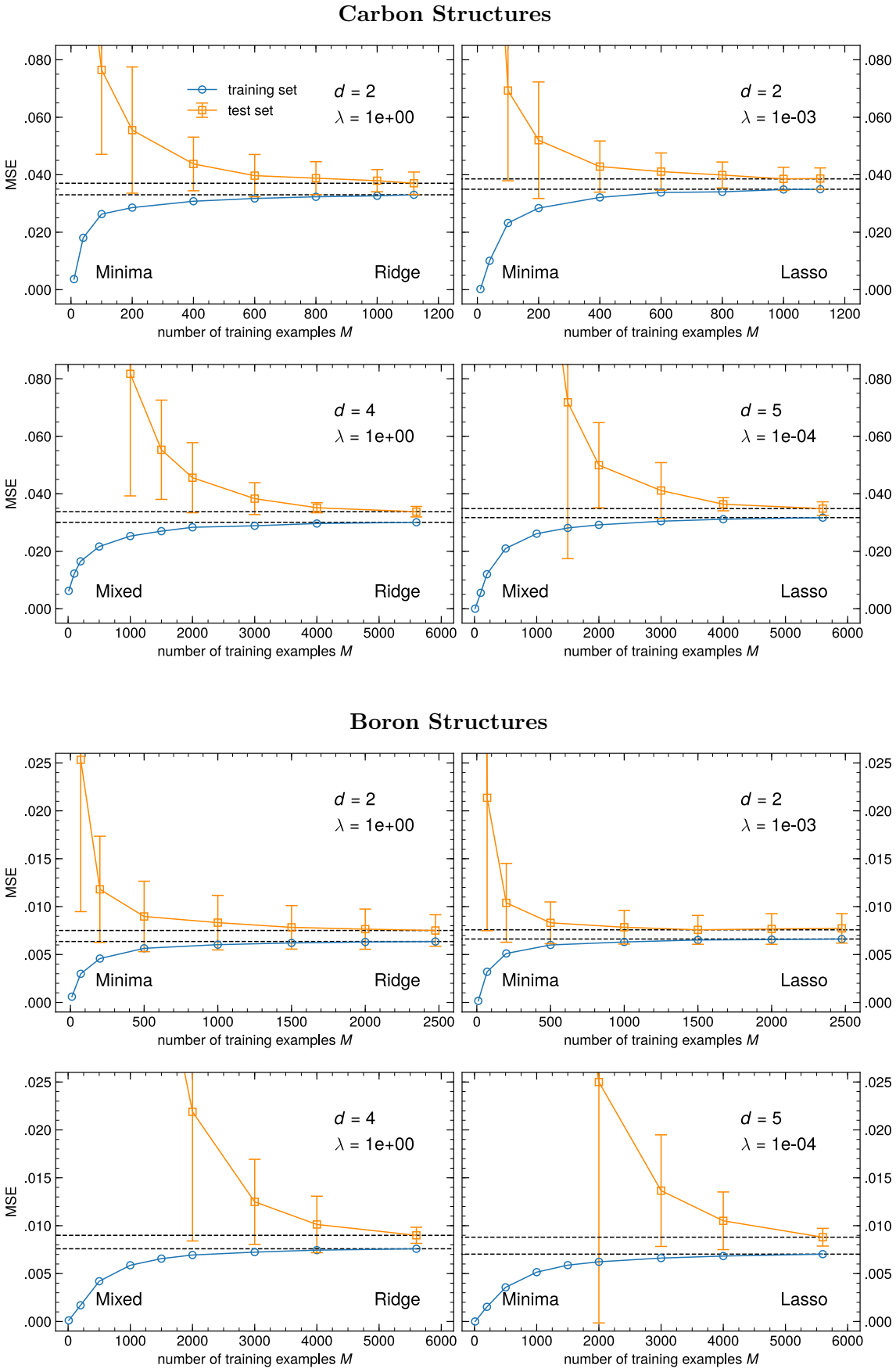


Figure 5.5: Learning Curves at optimal model parameters. The horizontal lines highlight the final prediction errors determined on the test and training set.

Database	Regression method	R^2 [%]	RMSE [meV/atom]	MAE [meV/atom]
Carbon − Minima	Ridge	60 ± 5	194 ± 11	146 ± 7
	Lasso	61 ± 5	196 ± 10	151 ± 8
Carbon − Mixed	Ridge	75 ± 2	182 ± 4	134 ± 3
	Lasso	74 ± 2	184 ± 4	135 ± 3
Boron − Minima	Ridge	62 ± 9	88 ± 11	62 ± 3
	Lasso	63 ± 8	86 ± 8	62 ± 2
Boron − Mixed	Ridge	82 ± 1	90 ± 1	68 ± 1
	Lasso	82 ± 1	90 ± 1	68 ± 1

Table 5.4: The model performance measured by the coefficient of determination R^2, the root mean squared error (RMSE) and the mean absolute error (MAE).

The error values in table 5.4 do not show a distinct improvement of the model performance as also unstable structures are included in the database. In case of the boron structures, the regression on mixed structures yields a higher prediction error than on the minima structures. However, the performance on mixed structures, caught by figure 5.6, seems to be better with regard to overall predictions measured on the entire energy range. This is verified by the listed R^2-values, which is a measure of how well the variance in the energy values is explained by the model. Here, the mixed structures clearly show an improvement. We assume that this behavior is explained by the distributions of energies, which in the case of the minima structures is denser accumulated within a certain energy domain, as it can be seen by the dots in the figure. This, in turn, induces the machine learning model to adjust its weights according to the accumulated structures, which again yields a comparably low prediction error, when these structures appear in the test set due to cross-validation. Therefore, we suggest training a machine learning model, performing ridge or lasso regression on energies of minima crystal structures, with a more uniformly distributed training set.

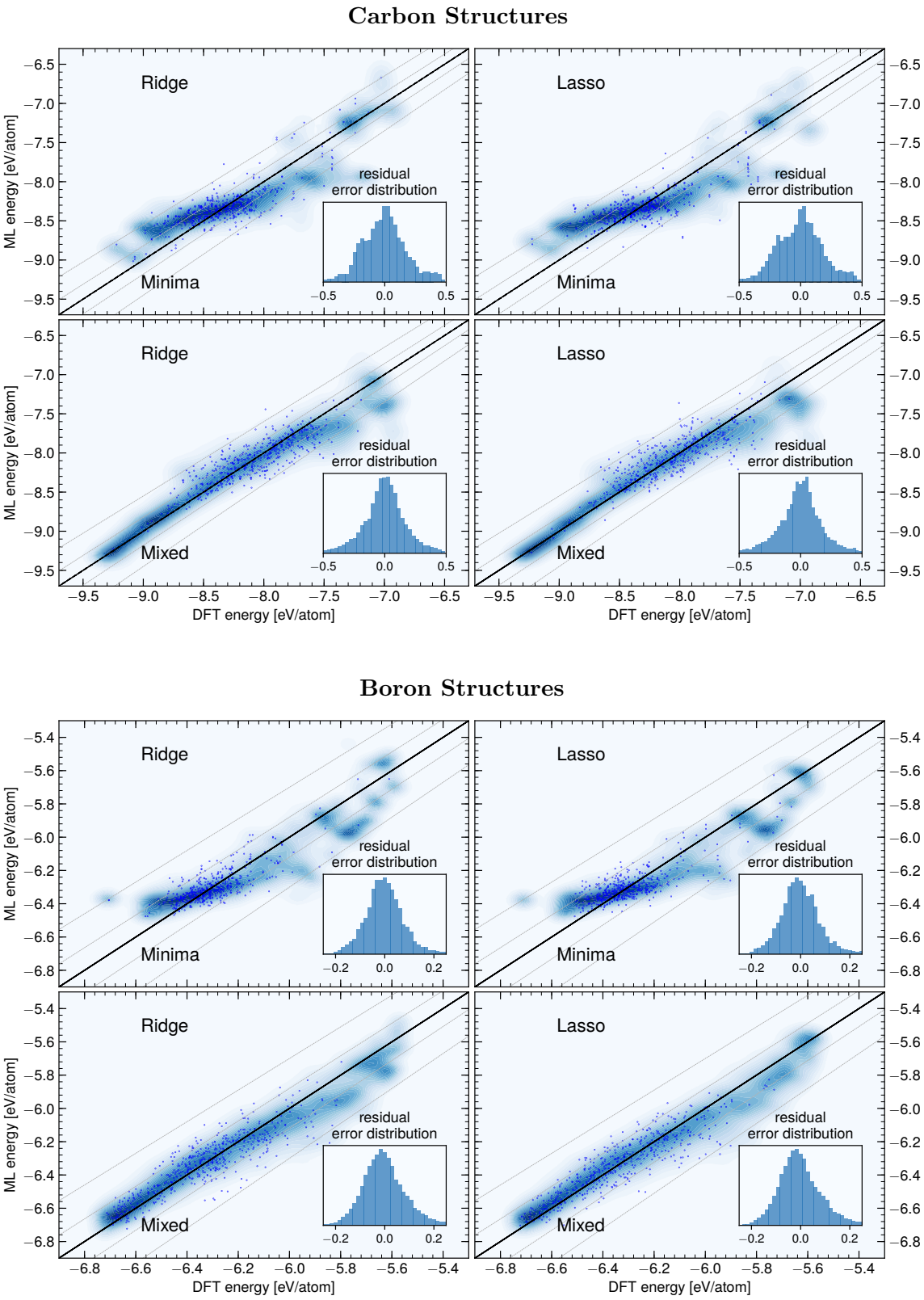


Figure 5.6: Model predictions: a comparison between predicted ML energies and DFT energies. The background contours represent the prediction tendency of the model; dots are sample predictions. The inset axis shows the distribution of the residual errors. The thin lines correspond to a 2%- and 5%-deviation from DFT energies.

Chapter 6

Learning Energies with Kernel-Based Regression

In this chapter the application and optimization of kernel-based regression methods are discussed. In the previous chapter we have used ridge and lasso regression, where initial features have been transformed to a higher-dimensional feature space using polynomial basis function expansions. Kernel-based methods, though, usually provide a more feasible application of a complex class of the hypothesis function: the kernel trick enables access to a high- or infinite-dimensional features space at low computational cost. In this case, limiting the total number of used features has a negligible effect on the performance; hence, we can use any number of features to serve as data representation for the special use of the Gaussian kernel.

On that account, we have introduced a radial distribution function (section 4.2.3) and an angular distribution function (section 4.2.4). In combination with the features from the previous chapter, these features provide an unique data representation for kernel-based regression methods. Nevertheless, the continuous functions have to be described by a discrete set of features, which in turn demands the adjustment of additional parameters, such as the bin size, the cut-off distance and the smoothing parameter.

With figure 3.8 we have given a brief overview on the components of a machine learning application and its sources of errors existing in the data, the data representation and the model itself. Here we note that the optimal parameters of the machine learning model – especially the kernel coefficient – are affected by the specific choice of parameters used in the discretization of the data representation. Hence, we carefully conduct the model optimization step by step, starting with the discretization of the radial distribution function in section 6.1. The binning of the angular distribution function is treated in section 6.2. Kernel ridge regression with default model parameters is used to optimize the data representation.

We additionally use support vector regression from section 6.3 onwards, to study the model performance using the ϵ-insensitive cost function. Here, we study the model performance in terms of prediction accuracy and do not compare computational performances of the different regression methods. In section 6.3 the kernel coefficient is adjusted, as well as model parameters of kernel ridge regression and support vector regression. The optimal settings are used in section 6.4 and 6.5 to obtain the learning curves and final predictions. We again measure the model performance throughout the optimization process by the mean squared error, using cross-validation with 50 repetitions.

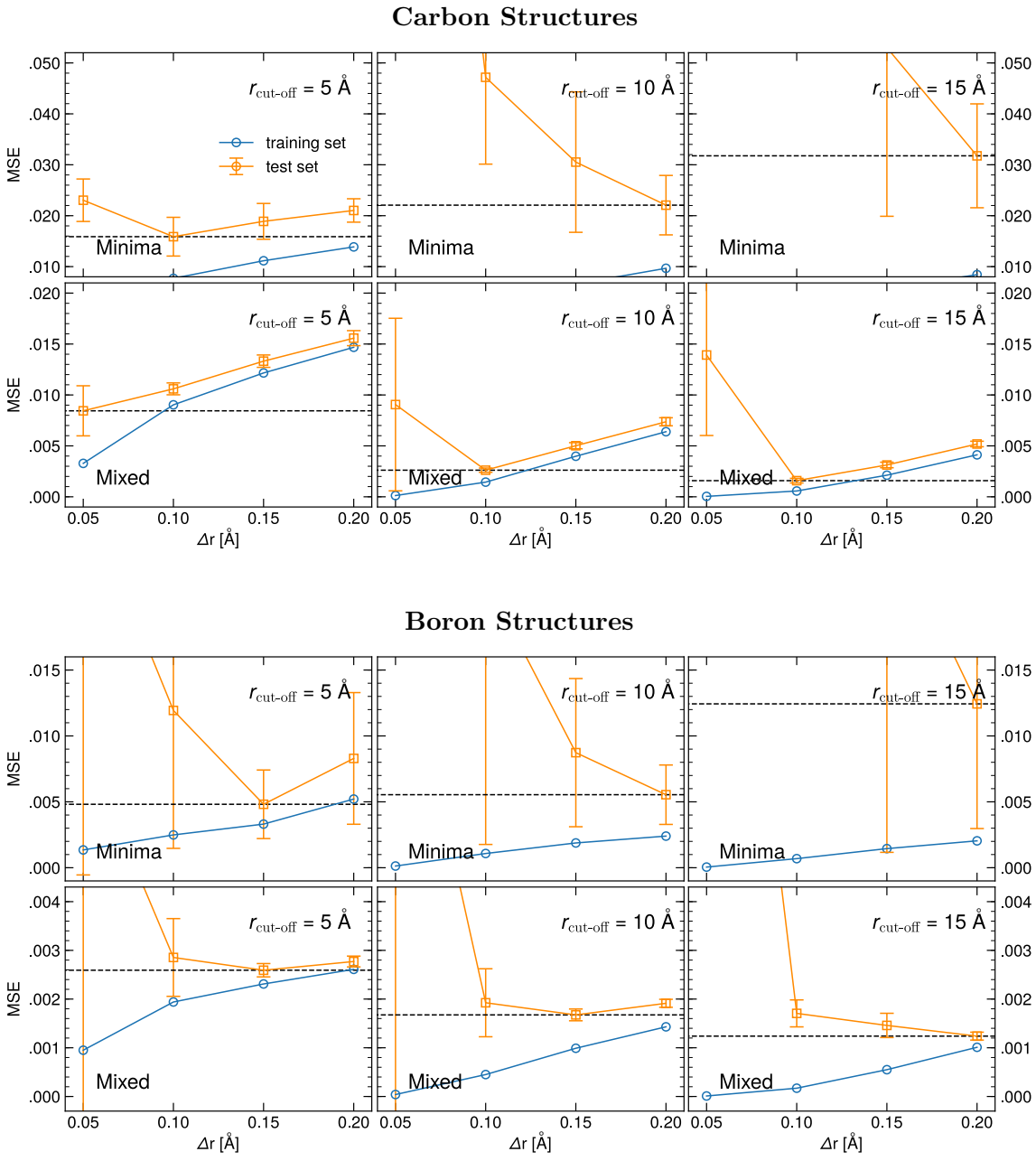


Figure 6.1: Optimizing the RDF. The bin size and the cut-off distance determine the resolution of the discretization of the function. The lowest prediction error obtained on the test set is highlighted by a horizontal line.

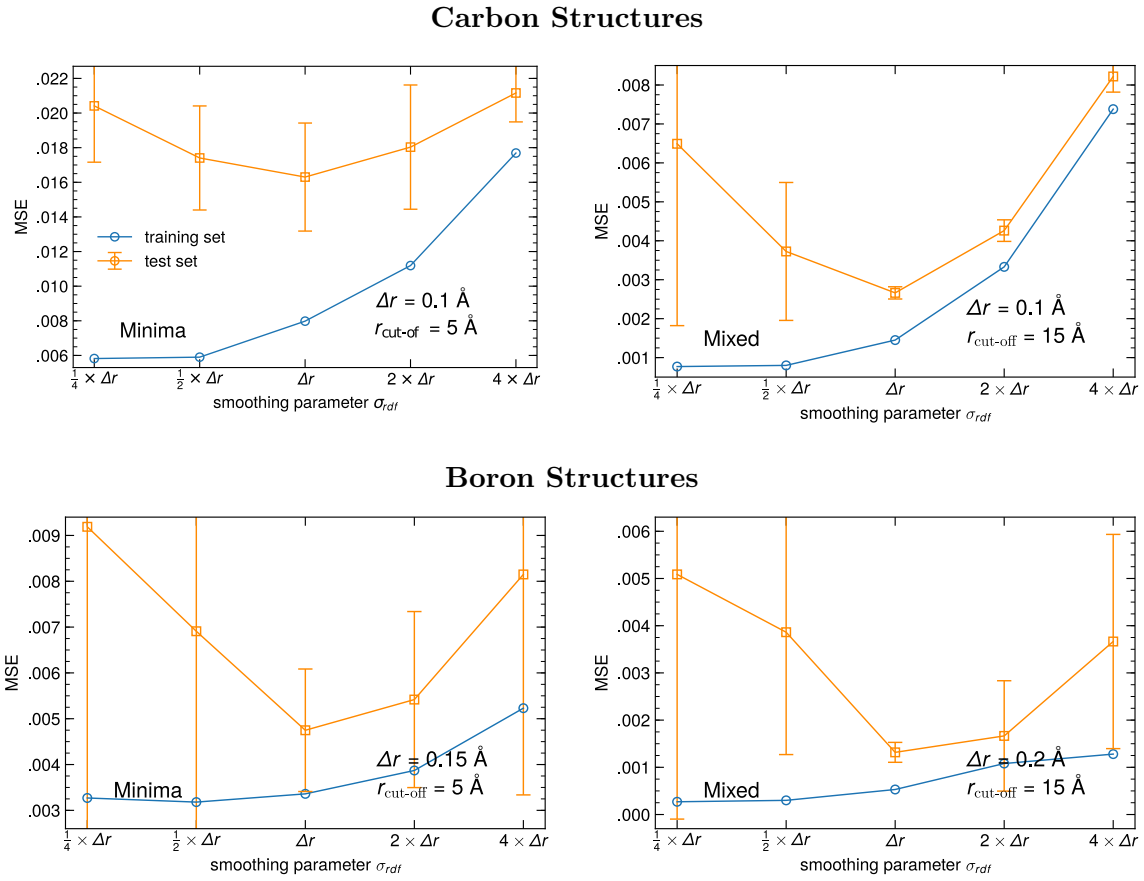


Figure 6.2: Adjusting the Gaussian smoothing of the RDF. The smoothing is clearly optimal when the parameter is set equal to the bin size used in the RDF.

6.1 Optimization of the Radial Distribution Function

The application of a radial distribution function, in data representations for machine learning models, is well known in literature [14,15]. Using the radial distribution function inefficiently, though, can have serious consequences: a highly-resolved radial distribution function describes structures too differently, which causes a failure of the Gaussian kernel measuring similarities between structures. This issue corresponds to overfitting, causing large errors in generalized predictions.

In this section we optimize the radial distribution function (70) on the datasets, by adjusting the bin size Δr, the cut-off distance $r_{\text{cut-off}}$ and the smoothing parameter σ_{rdf}. The bin size and the cut-off distance determine the characteristics of the radial distribution function, this is why both quantities have to be optimized simultaneously. The smoothing parameter can be adjusted independently. Thus, we initially set the smoothing parameter equal to the bin size. We use kernel ridge regression with `scikit-learn`'s default settings of hyperparameters to perform the optimization at fixed model parameters. We additionally use features from the previous chapter (table 5.1) in the data representation, to take long range order into consideration; the angular distribution function is not used in this section.

Figure 6.1 shows the mean squared error as a function of the bin size, measured at cut-off distances $r_{\text{cut-off}} = 5\,\text{Å}$, $10\,\text{Å}$, $15\,\text{Å}$. Horizontal lines indicate the lowest errors achieved on the test set. The results of the minima structures show that the lowest prediction errors

Database	Δr [Å]	$r_{\text{cut-off}}$ [Å]	σ_{rdf} [Å]	$\Delta\vartheta$ [°]	σ_{adf} [°]
Carbon − Minima	0.1	5	0.1	15	15
Carbon − Mixed	0.1	15	0.1	15	15
Boron − Minima	0.15	5	0.15	10	10
Boron − Mixed	0.2	15	0.2	10	10

Table 6.1: Optimal parameters for the discretization of the RDF (Δr, $r_{\text{cut-off}}$, σ_{rdf}) and the ADF ($\Delta\vartheta$, σ_{adf}).

are achieved at the cut-off distance $r_{\text{cut-off}} = 5$ Å. Beyond that distance, the model starts to overfit the data, especially in combination with small bin sizes. In the case of mixed structures, though, the radial distribution function with the long range cut-off distance $r_{\text{cut-off}} = 15$ Å yields the best performances. The optimal bin size for the carbon structures is $\Delta r = 0.1$ Å. The bin size for boron structures is optimal for the minima structures at $\Delta r = 0.15$ Å and $\Delta r = 0.2$ Å for the mixed structures.

We take these settings to determine the optimal smoothing parameter by measuring the performance using the parameter at fractions or multiples of the bin size. The results are given by figure 6.2. The plots show a common trend: the smoothing is optimal when the free parameter is set equal to the bin size. Table 6.1 contains the parameters obtained in this section.

6.2 Optimization of the Angular Distribution Function

Features used in the previous sections do not contain a detailed description of angles between nearest neighbors; an essential, yet unexploited, information. In this section we use the angular distribution function, discussed in section 4.2.4, to further develop the data representation with respect to bond angles. Again, the angular distribution function has to be described through binning, entailing the threat to overfit the data. An optimized angular distribution function potentially improves the data representation and reduces the bias error. The binning of the angular distribution function needs to be adjusted in terms of the bin size $\Delta\vartheta$ and the smoothing parameter σ_{adf}. Due to its periodicity the angular distribution function does not have any cut-off parameter.

We measure the model performance subject to the bin size and the smoothing parameter of the angular distribution function, including the optimized data representation from the previous section. The bin size is chosen with $\Delta\vartheta = 5°, 10°, 15°$ and the smoothing parameter, again, as fraction and multiple of the bin size.

Figure 6.3 shows the measured mean squared error as a function of the smoothing parameter using different bin sizes. The results show that the angular distribution with bin size $\Delta\vartheta = 15°$ gives the best performance on both carbon datasets. In contrast, the model achieves the lowest prediction errors on the boron structures using the bin size $\Delta\vartheta = 10°$.

As for the radial distribution function, the smoothing is optimal when the parameter is set equal to the bin size. We summarize the determined parameters for the discretization of the radial distribution function and the angular distribution function in table 6.1.

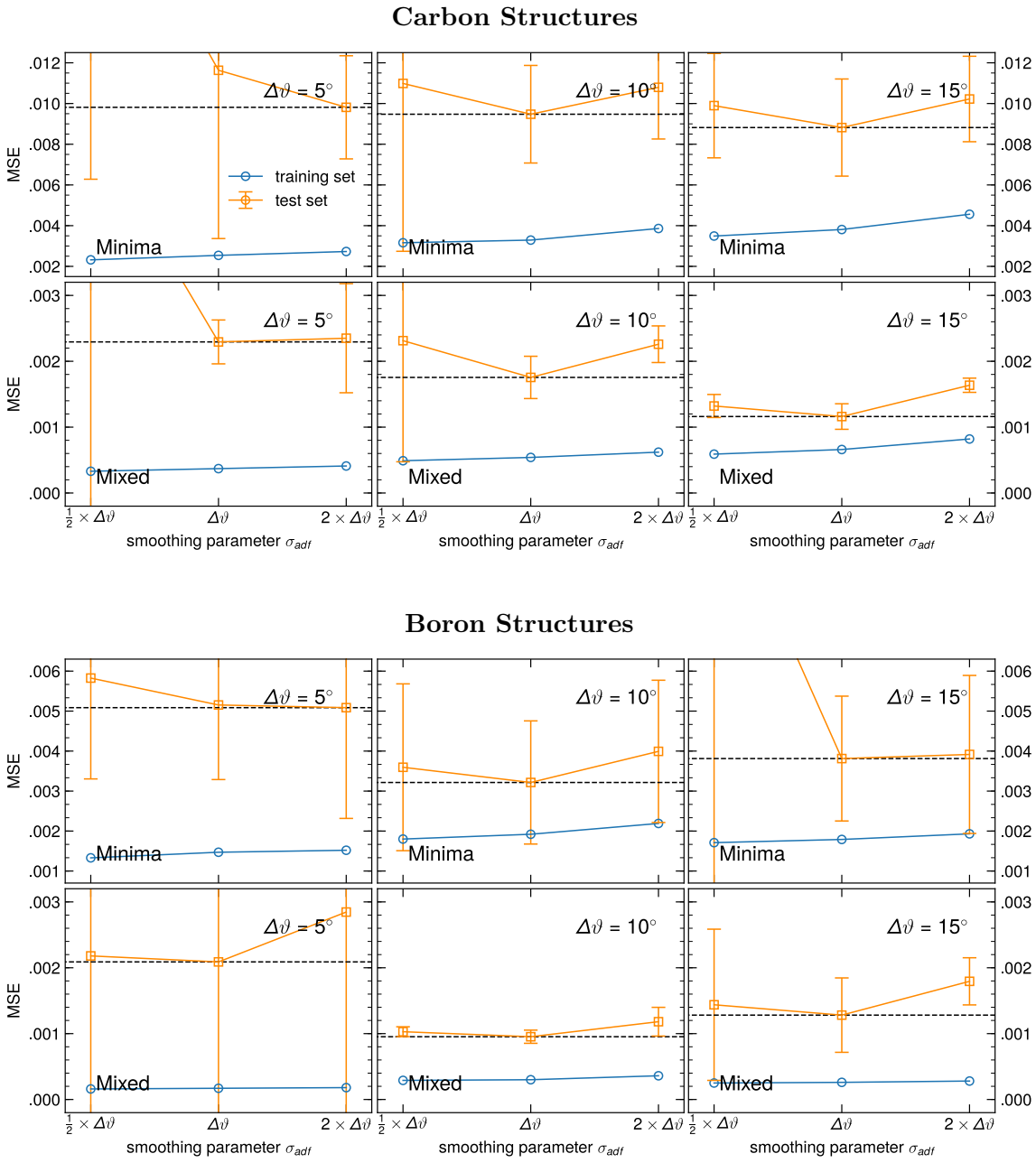


Figure 6.3: Optimizing the ADF by adjusting the bin size $\Delta\vartheta$ and the smoothing parameter σ_{adf}. The lowest prediction error determined on the test set is highlighted by a horizontal line.

Database	Regression method	γ	λ	ϵ	C
Carbon − Minima	KRR	1/5000	1e−04	-	-
	SVR	1/5000	-	0.05	1e+03
Carbon − Mixed	KRR	1/1000	1e−05	-	-
	SVR	1/1000	-	0.05	1e+04
Boron − Minima	KRR	1/10000	1e−04	-	-
	SVR	1/10000	-	0.05	1e+03
Boron − Mixed	KRR	1/10000	1e−05	-	-
	SVR	1/10000	-	0.05	1e+03

Table 6.2: Optimal kernel and model parameters of kernel ridge regression and support vector regression. The same kernel coefficient γ can be used for kernel ridge regression and support vector regression.

6.3 Fine-Tuning of the Kernel and Model Parameters

In this section the machine learning model is adjusted in terms of hyperparameters, such as the regularization parameter λ and the kernel coefficient γ. We additionally use support vector regression to compare the model performance using the ϵ-insensitive cost function. The application of support vector regression demands the additional adjustment of the penalty tolerance ϵ and the penalty parameter C; latter corresponds to the concept of regularization.

First, we optimize the kernel coefficient γ, which appears in the Gaussian kernel definition (46) and determines the sensitivity to differences in feature vectors; here, the feature vectors are given by the data representation. Hence, the kernel coefficient regulates the sensitivity to differences of structures in the data representation. Its optimal value depends entirely on the data; strictly speaking on the distances $\mathbf{x}-\mathbf{x}'$ we have adjusted with the optimization of the data representation. Therefore, the same kernel coefficient can be used for kernel ridge regression and support vector regression as well, since we are using the same data representation for both methods.

First, we use kernel ridge regression with the optimized data representation, listed in table 6.1. We choose three different values for the kernel coefficient and measure the model performance as a function of the regularization parameter λ. The results are given in figure 6.4. The kernel coefficient for the carbon structures is optimal at $\gamma = 1/5000$ for the minima and $\gamma = 1/1000$ for the mixed dataset. The prediction on the two boron structure sets is optimized at the kernel coefficient $\gamma = 1/10000$. The optimal values for the regularization parameter λ are listed in table 6.2.

We use the obtained kernel coefficients to adjust the penalty tolerance ϵ and penalty parameter C using support vector regression. The results are given in figure 6.5. We notice that the penalty tolerance is optimal at $\epsilon = 0.05$ for all structure sets. We list the optimal penalty parameter and furthermore all parameters obtained in this section in table 6.2.

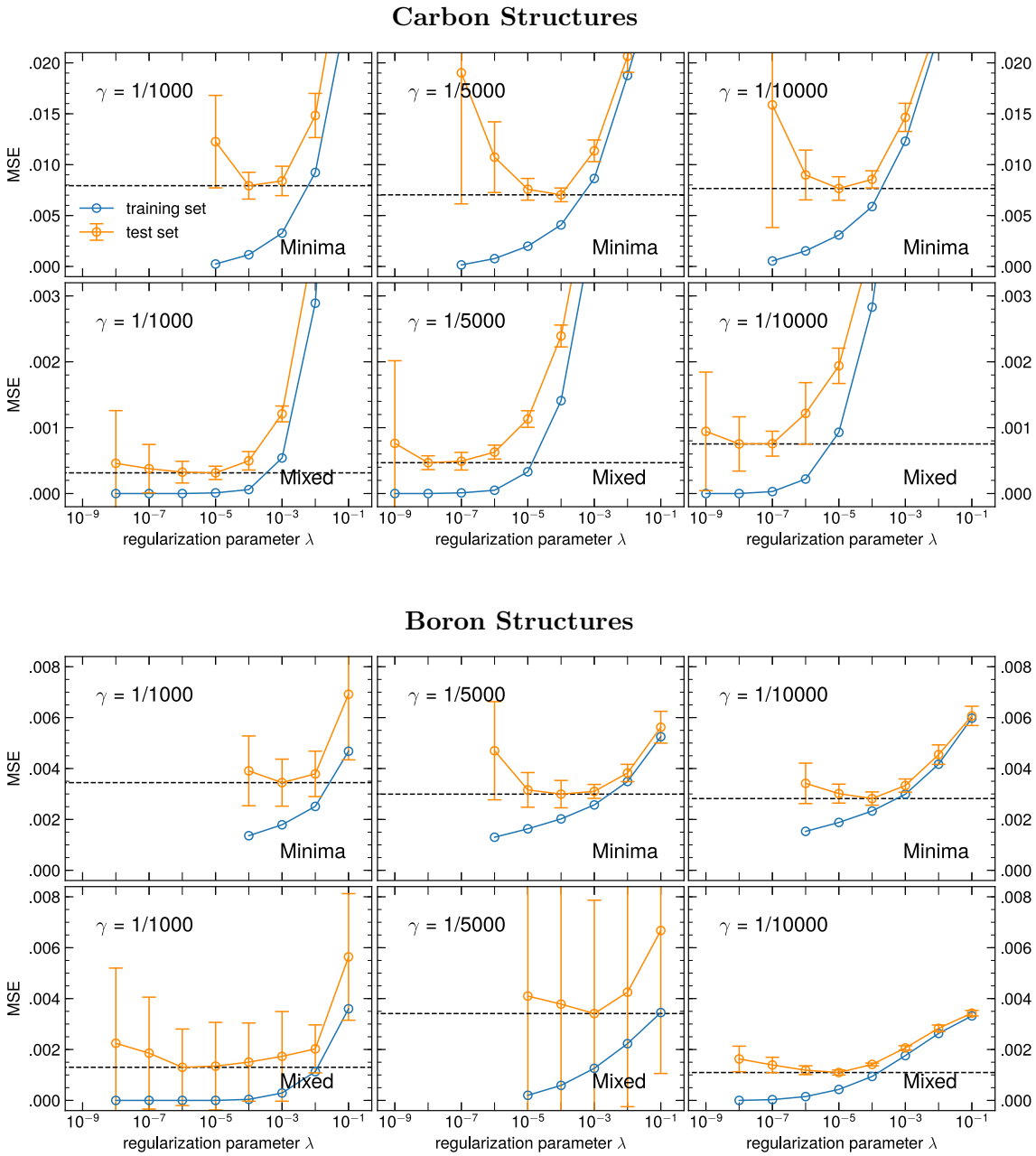


Figure 6.4: Determining the optimal kernel coefficient γ and the regularization parameter λ of kernel ridge regression. The lowest prediction error determined on the test set is highlighted by a horizontal line.

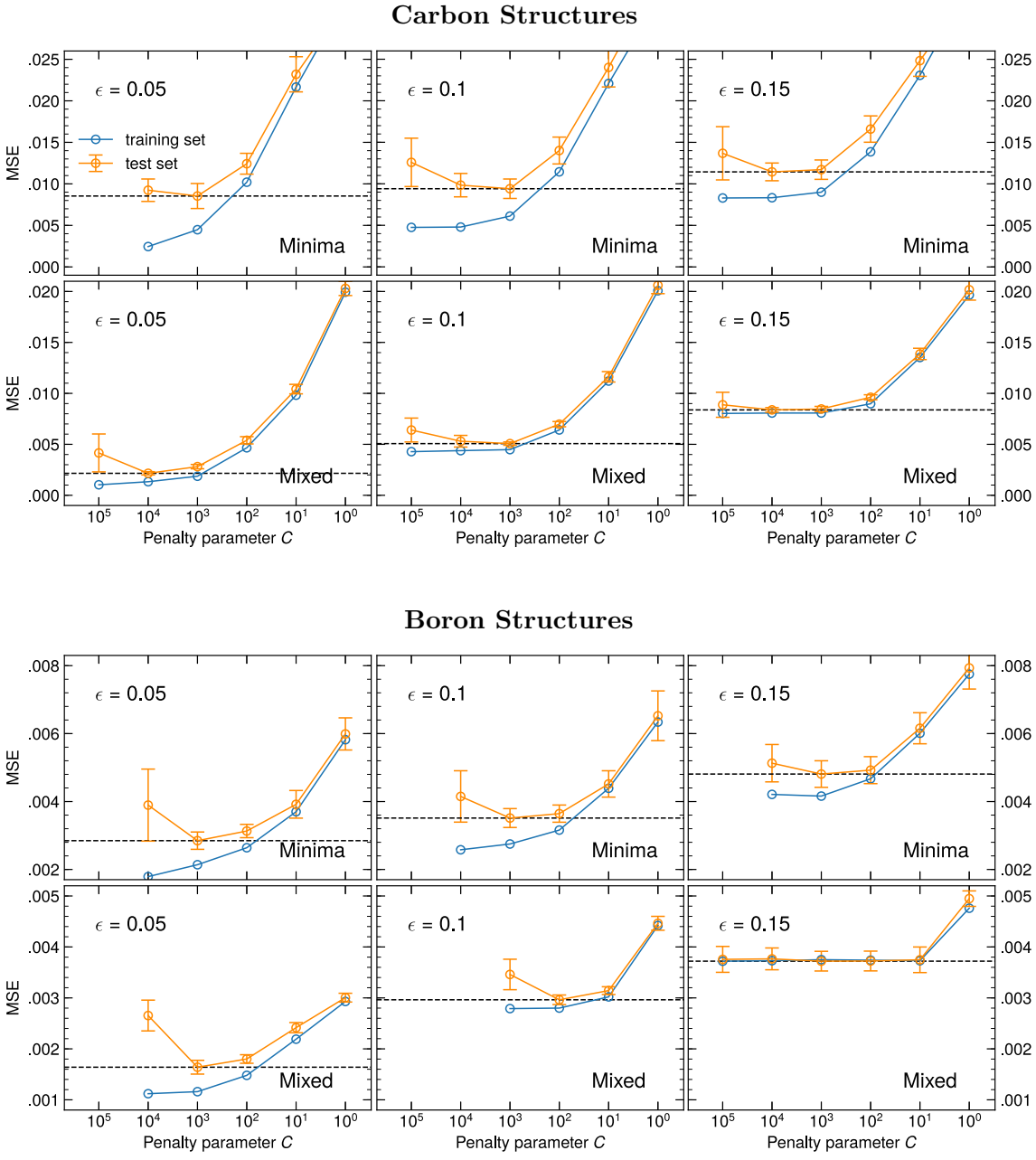


Figure 6.5: Determining the optimal penalty tolerance ϵ and penalty parameter C, used by the support vector regression. The lowest prediction error determined on the test set is highlighted by a horizontal line.

6.4 Learning Curves

The learning curves at the optimal settings are shown in figure 6.6. With respect to the learning process, we do not observe any significant difference between kernel ridge regression and support vector regression. Both regression methods show an equal learning progress when the number of training examples is incrementally increased. The learning curves do not show any serious over- or underfitting issues. However, one can already notice that the performance of support vector regression on the mixed dataset is remarkable worse than the performance of kernel ridge regression. This will be validated quantitatively in the next and final section of this chapter.

We further notice that the prediction error on the training set, using kernel ridge regression on the carbon mixed structures, is completely reduced, i.e. the model achieves perfect predictions on the training set. This in turn implies, that we have completely reduced the bias error in the machine learning application, while we have also optimized the performance on the test set. Therefore, we assume that the gap between the training and the test set in the case of the mixed carbon structures already corresponds to the generalization gap; this also means that the model performance cannot be further improved significantly.

6.5 Final Prediction and Discussion

To determine the predictive power of our machine learning model we use the optimal parameters listed in table 6.1 and 6.2. Cross-validation is used with 100 repetitions to obtain the mean and the standard deviation of the scoring parameters.

The measured values are given in table 6.3; the predictions are visualized in figure 6.7. Compared to the results of ridge and lasso regression, one can observe a great improvement of the model performance using kernelized regression methods. In the case of the mixed datasets, kernel ridge regression achieves a mean absolute error of 9.5 meV/atom on carbon structures and 16.0 meV/atom on boron structures. These prediction errors are far smaller than errors made on the minima structures using kernel ridge regression, as well as using support vector regression on any dataset. We assume that these observations share a common explanation: each dataset is a sample of the potential energy surface, which in the case of the dataset containing solely relaxed structures is a sparse representation compared to dataset including also unrelaxed structures. Hence, the denser representation of the potential energy surface in the case of mixed structures allows more accurate predictions because the Gaussian kernel learns the potential energy surface more precisely. When only relaxed structures are used in the dataset, the information from unstable structures is missing; this is why the model predicts energies less well. Since support vector regression generally obtains a sparse representation, the computational advantage – which is not taken into account in this thesis – is obtained at the expense of the prediction accuracy. The difference in prediction errors between kernel ridge regression and support vector regression is less significant used on the minima datasets.

With the model performance achieved using kernel ridge regression on the mixed structure sets, we assume to have optimized the machine learning application towards prediction errors comparable to the DFT calculations. In particular, we have seen from the learning curves that in case of the carbon mixed structures we could reduce the bias error completely; therefore, we expect that the error of 9.5 meV/atom cannot be further reduced significantly, but this error already reflects the approximate statistical error of the underlying system. In the case of the boron structures, we assume to be able to achieve an

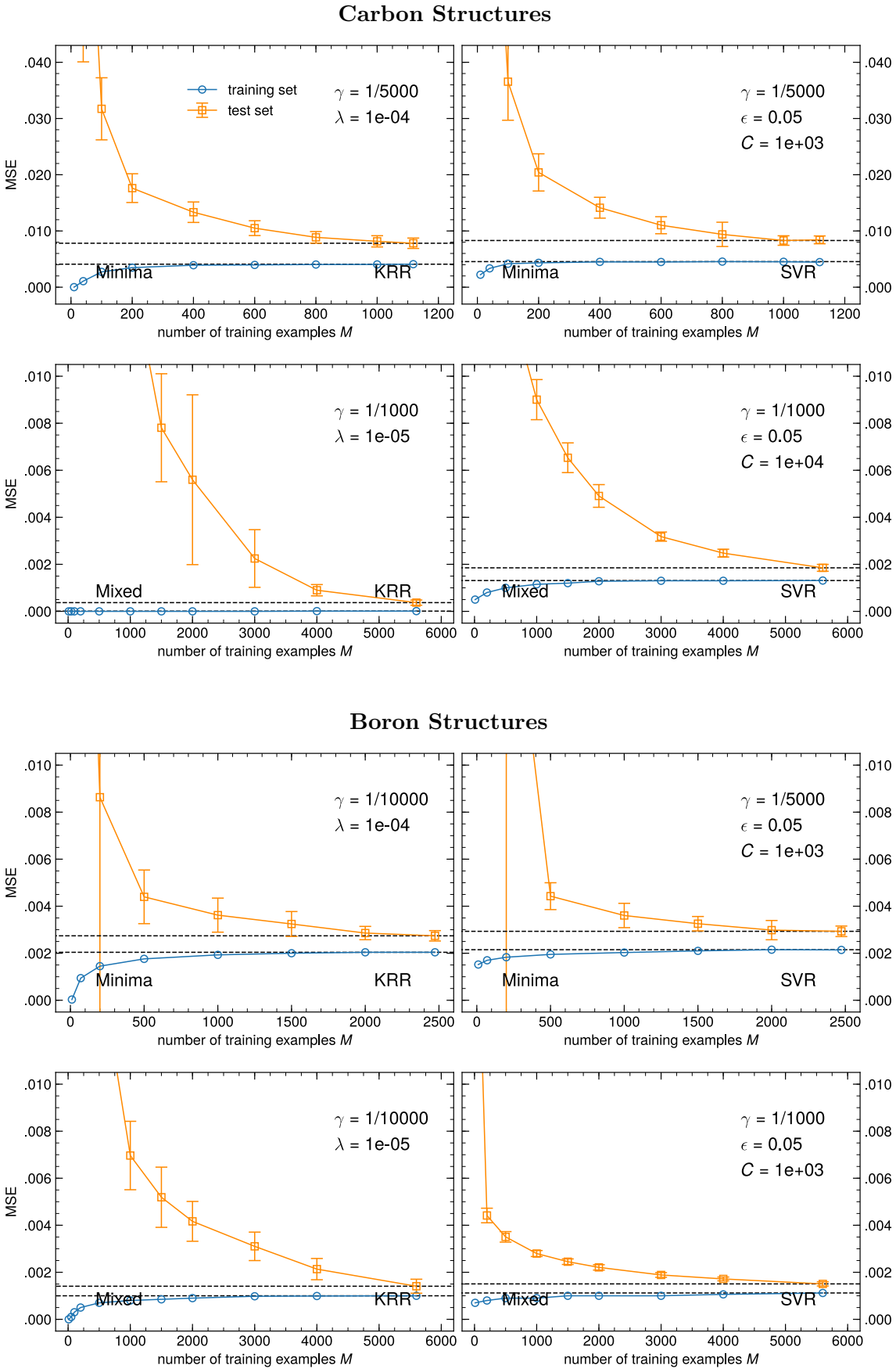


Figure 6.6: Learning Curves at optimal model parameters. The horizontal lines highlight the final prediction errors on the test and training set.

Database	Regression method	R^2 [%]	RMSE [meV/atom]	MAE [meV/atom]
Carbon – Minima	KRR	91.2 ± 1.4	89.5 ± 4.5	67.5 ± 3.0
	SVR	90.6 ± 1.7	91.0 ± 5.5	68.5 ± 3.5
Carbon – Mixed	KRR	99.7 ± 0.1	19.0 ± 3.5	9.5 ± 0.5
	SVR	98.3 ± 0.2	45.5 ± 2.0	37.5 ± 1.0
Boron – Minima	KRR	85.9 ± 1.8	53.5 ± 3.0	39.0 ± 1.5
	SVR	85.7 ± 1.4	52.5 ± 2.5	38.5 ± 1.0
Boron – Mixed	KRR	98.1 ± 0.3	28.5 ± 2.0	16.0 ± 0.5
	SVR	96.4 ± 0.2	38.5 ± 1.5	32.5 ± 1.0

Table 6.3: Model performance measured by the coefficient of determination R^2, the root mean squared error (RMSE) and the mean absolute error (MAE).

equal error, if more crystal structures were used for the optimization and in the training process.

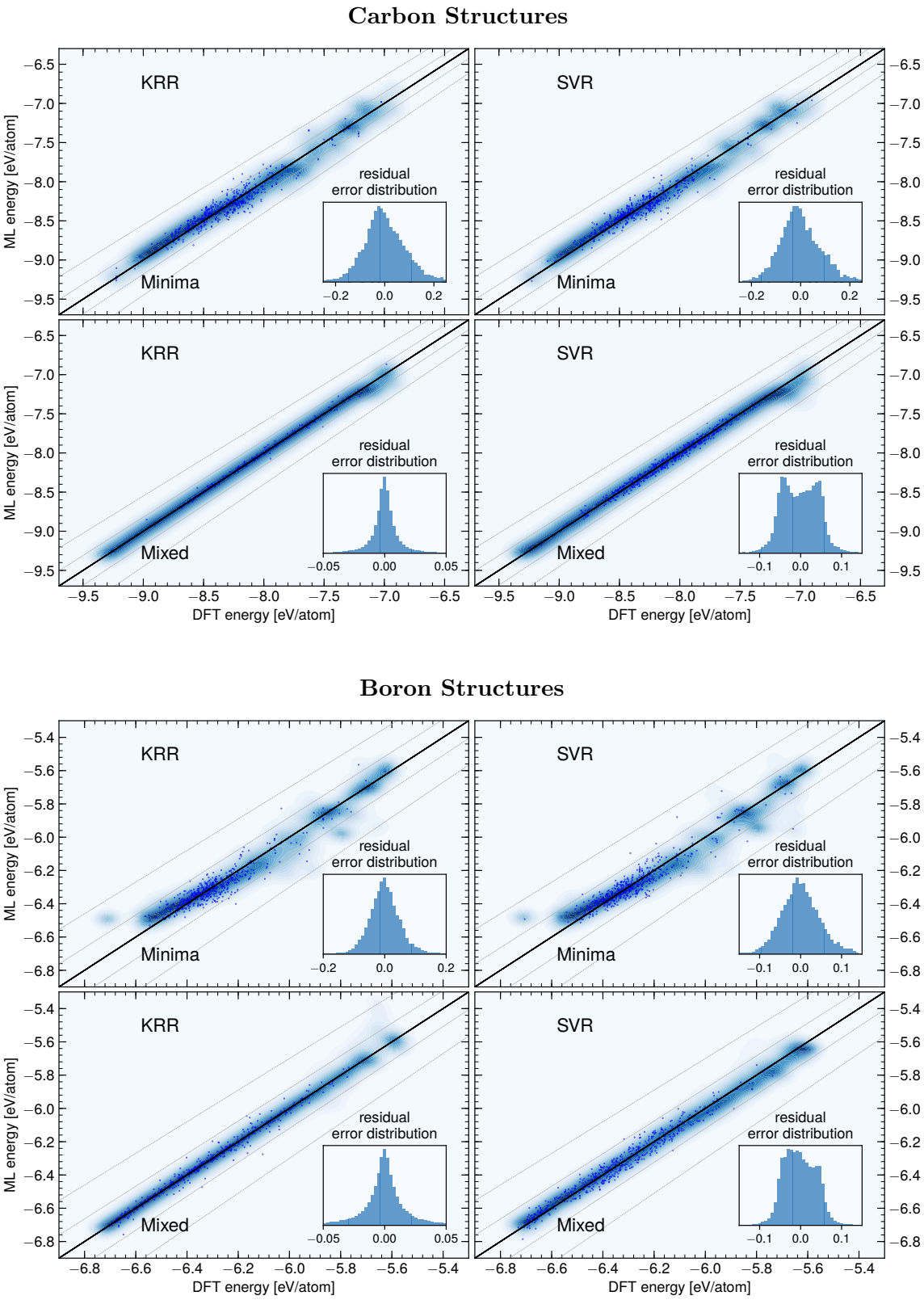


Figure 6.7: Model predictions: a comparison between predicted ML energies and DFT energies. The background contours represent the prediction tendency of the model; dots are sample predictions. The inset axis shows the distribution of the residual errors. The thin lines correspond to a 2%- and 5%-deviation from DFT energies.

Chapter 7

Conclusion

Machine learning models are capable of predicting material properties with accuracies comparable to first-principles methods, but significantly reduced computational cost. In order to achieve low prediction errors, the machine learning application has to be optimized on the system under study; in particular, in terms of data, data representation and model parameters.

In this thesis we presented the application of a machine learning model to predict DFT energies for mono-elemental crystal structures of carbon and boron. The discussion on DFT and used crystal structure prediction techniques was given in chapter 2. We presented the regression methods considered in this work in chapter 3. The discussion on the available crystal structures and the used data representation was covered in chapter 4. Chapter 5 presented the optimization and prediction accuracy of ridge and lasso regression. The results using kernel ridge regression and support vector regression were provided in chapter 6.

Our results showed that kernel ridge regression performs best among the various regression methods. With the use of the Gaussian kernel and an optimized data representation we could achieve a steady prediction accuracy of 99.7% on the carbon and 98.1% on the boron structures in terms of R^2-values. The corresponding prediction errors, measured by the mean absolute error, were 9.5 meV/atom for the carbon and 16.0 meV/atom for the boron structures.

These prediction accuracies were obtained on datasets which included relaxed and unrelaxed structures; applying machine learning on relaxed structures only could not achieve the same prediction accuracies. This behavior was explained by the particular choice of the Gaussian kernel, which makes predictions on new structures according to the similarity to learned structures. Including unrelaxed structures in the training process gave a more detailed presentation of the potential energy surface, and hence a better starting point for the algorithm to measure similarities. Support vector regression could not obtain equal prediction accuracies, since the algorithm uses for the prediction a sparse selection of training examples. Ridge and lasso regression do not apply kernel functions, why the limited extent of the data representation could be seen as reason for less accurate energy predictions.

With our optimized data representation and machine learning model, capable of making predictions with accuracies of DFT calculations, we provide the basis for further development and improvement of machine learning applications on mono-elemental crystal structures. As open questions a few interesting points can be mentioned. The data representation was tested on crystal systems containing up to 24 atoms. The prediction accuracy and model performance using the same data representation on larger systems

has not been validated, and represents an unanswered question. Furthermore, the machine learning estimation of further material quantities could be tested; for instance the prediction of the density of states, the band gap or the hardness may be of interest for practical implementations in materials science. Finally, extending the data representation for the machine learning application on crystal structures of binary compounds would improve the field of application greatly.

Appendices

Appendix A

Conventions

Throughout this thesis, we use Hartree atomic units:

$$\hbar = e = m_{\mathrm{e}} = \frac{1}{4\pi\epsilon_0} = 1 \ , \tag{77}$$

where $\hbar$ is the reduced Planck's constant, e the elementary charge, m_{e} the electron mass and ϵ_0 the vacuum permittivity. Furthermore, we use a short hand notations for electronic and atomic coordinates:

$$\mathbf{r}_{\mathrm{e}} \equiv (\mathbf{r}_1, .., \mathbf{r}_N) \quad , \quad \mathbf{R}_{\mathrm{n}} \equiv (\mathbf{R}_1, .., \mathbf{R}_M) \ . \tag{78}$$

General math notation

Symbol	Meaning
$\equiv$	Defined as
$\rightarrow$	Tends towards
∞	Infinity
$\propto$	Proportional to
$\in$	Is an element of
$\mathbb{R}$	Real numbers
$\otimes$	Tensor product
!	Factorial
d	Derivative
∂	Partial derivative
δ	Functional derivative
∇	Vector of first derivatives
Δ	Difference
$\langle\ \Psi\ \vert$	Bra vector
$\vert\ \Psi\ \rangle$	Ket vector
$\{x_i\}$	Set
$\vert x \vert$	Absolute value
$\bar{x}$	Mean value
$\boldsymbol{x}$	Vector
$\boldsymbol{x}^T$	Transpose of a vector
$\Vert\boldsymbol{x}\Vert$	Euclidean or ℓ^2-Norm, $\Vert\boldsymbol{x}\Vert = \sqrt{\sum_i x_i^2}$

$\boldsymbol{X}$	Matrix
$\boldsymbol{X}^{-1}$	Inverse of a matrix
$\boldsymbol{X}^{T}$	Transpose of a matrix
$\mathcal{O}$	Order of magnitude
θ	Heaviside-function

Machine learning/statistics notation

Symbol	Meaning
$\boldsymbol{\beta}$	Parameter vector
$\hat{\boldsymbol{\beta}}$	Estimate of $\boldsymbol{\beta}$
$\boldsymbol{x}$	Feature vector
$x_j^{(i)}$	j-th feature of the i-th training example
$\boldsymbol{\phi}(\boldsymbol{x})$	Basis function expansion of feature vector $\boldsymbol{x}$
$h_{\hat{\boldsymbol{\beta}}}$	Hypothesis function with estimated parameters $\hat{\boldsymbol{\beta}}$
$J(\hat{\boldsymbol{\beta}})$	Cost function
$\boldsymbol{I}_N$	Identity matrix of dimension N
$\boldsymbol{X}$	Design matrix
$\boldsymbol{K}$	Kernel matrix
κ	Kernel function
$\hat{\alpha}$	Estimate of dual variable
$p(x)$	Probability density
$p(x\|y)$	Conditional probability density of x given y
$\mathcal{N}$	Gaussian distribution

Appendix B

Maximum a Posteriori Estimation

In the following section, we discuss the parameter estimation in linear regression from a probabilistic point of view, using *Bayesian probability theory*. The discussion is based on the book by Kevin P. Murphy [50]. The fundamental idea of Bayesian probability theory is to express probability in terms of the uncertainty or knowledge of an event, rather than long term frequencies. Central object is Bayes' theorem which is given by:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{normalization}} \, . \tag{79}$$

The *posterior* represents the probability density $p(\hat{\boldsymbol{\beta}}|\mathcal{D})$ for the weight parameters $\hat{\boldsymbol{\beta}}$ of the underlying linear relationship, given the sample pairs $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$. This probability density is given by the product of the *likelihood* and the *prior*. The likelihood $p(\mathcal{D}|\hat{\boldsymbol{\beta}})$ describes the probability for the data $\mathcal{D}$, given the linear relation at the parameters $\hat{\boldsymbol{\beta}}$. The prior $p(\hat{\boldsymbol{\beta}})$ represents the probability density for the weights $\hat{\boldsymbol{\beta}}$ based on suitable assumption or constraints. We express Bayes' theorem (79) as follows:

$$p(\hat{\boldsymbol{\beta}}|\mathcal{D}) = \frac{p(\mathcal{D}|\hat{\boldsymbol{\beta}})p(\hat{\boldsymbol{\beta}})}{p(\mathcal{D})} \, , \tag{80}$$

where $p(\mathcal{D})$ represents the normalization factor. A possible point estimate of optimal weights $\hat{\boldsymbol{\beta}}$ is given by the *mode* of the posterior distribution, which equals its maximum and is referred to as the *maximum a posteriori* (MAP) estimate.

B.1 Ordinary Least Squares

For the standard *ordinary least squares* (OLS) approach, we assume an uniform prior $p(\hat{\boldsymbol{\beta}}) = \text{const}$. In this case, the MAP equals the maximum of the likelihood, referred to as the *maximum likelihood estimate* (MLE), which can be formally expressed by

$$\hat{\hat{\boldsymbol{\beta}}}^{\text{MLE}} = \text{argmax}_{\hat{\boldsymbol{\beta}}} \, p(\mathcal{D}|\hat{\boldsymbol{\beta}}) \, . \tag{81}$$

We assume that the statistical error in the data, hence the observed target value y, is Gaussian distributed:

$$p(y|\boldsymbol{x}, \hat{\boldsymbol{\beta}}) = \mathcal{N}(y|\hat{\boldsymbol{\beta}}^T\boldsymbol{x}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - \hat{\boldsymbol{\beta}}^T\boldsymbol{x})^2\right) \, , \tag{82}$$

where $\hat{\boldsymbol{\beta}}^T \boldsymbol{x}$ represents the theoretical mean and σ^2 the variance. Furthermore, we assume the M training examples to be independent and identically distributed *idd*. Thus, the log-likelihood is given by

$$\log p(\mathcal{D}|\hat{\boldsymbol{\beta}}) = \log \prod_{i=1}^{M} p(y^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\beta}}) = \sum_{i=1}^{M} \log p(y^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\beta}}) \; . \tag{83}$$

For practical reasons, we express the MLE in terms of the negative log-likelihood (NLL) to determine the optimal weights which minimize the NLL:

$$\begin{aligned} \mathrm{NLL}(\hat{\boldsymbol{\beta}}) &= -\sum_{i=1}^{M} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (y^{(i)} - \hat{\boldsymbol{\beta}}^T \boldsymbol{x}^{(i)})^2 \right) \right] \\ &= \frac{M}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \mathrm{RSS}(\hat{\boldsymbol{\beta}}) \; . \end{aligned} \tag{84}$$

Here we have used the definition of the *residual sum of squares* (RSS),

$$\mathrm{RSS}(\hat{\boldsymbol{\beta}}) = \sum_{i=1}^{M} (y^{(i)} - \hat{\boldsymbol{\beta}}^T \boldsymbol{x}^{(i)})^2 \; . \tag{85}$$

The RSS is convex in the weights $\hat{\boldsymbol{\beta}}$, hence numerical optimization methods, e.g. gradient descent, can be used to determine its minimum. Typically, one divides the RSS by the number of given data points, which results in the *mean squared error* (MSE):

$$\mathrm{MSE}(\hat{\boldsymbol{\beta}}) = \frac{1}{M} \sum_{i=1}^{M} (y^{(i)} - \hat{\boldsymbol{\beta}}^T \boldsymbol{x}^{(i)})^2 \; . \tag{86}$$

The optimization of the weight parameters in machine learning is referred to as minimizing the *cost function* $J(\hat{\boldsymbol{\beta}})$; here given by the MSE.

To determine the minimum analytically, we use the vectorized form of the given labels and features from (33) and (34), to rewrite the RSS according to:

$$\begin{aligned} \mathrm{RSS}(\hat{\boldsymbol{\beta}}) &= (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^T (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) \\ &= \boldsymbol{y}^T \boldsymbol{y} - \boldsymbol{y}^T \boldsymbol{X}\hat{\boldsymbol{\beta}} - (\boldsymbol{X}\hat{\boldsymbol{\beta}})^T \boldsymbol{y} + (\boldsymbol{X}\hat{\boldsymbol{\beta}})^T (\boldsymbol{X}\hat{\boldsymbol{\beta}}) \\ &= \boldsymbol{y}^T \boldsymbol{y} - 2 \cdot (\boldsymbol{X}^T \boldsymbol{y})^T \hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\beta}}^T (\boldsymbol{X}^T \boldsymbol{X}) \hat{\boldsymbol{\beta}} \; . \end{aligned} \tag{87}$$

We use

$$\frac{\partial \left(\boldsymbol{b}^T \boldsymbol{a} \right)}{\partial \boldsymbol{a}} = \boldsymbol{b} \qquad , \qquad \frac{\partial (\boldsymbol{a}^T \boldsymbol{A} \boldsymbol{a})}{\partial \boldsymbol{a}} = (\boldsymbol{A} + \boldsymbol{A}^T) \boldsymbol{a} \tag{88}$$

to obtain the derivative of the RSS with respect to the weights $\hat{\boldsymbol{\beta}}$:

$$\frac{\partial \, \mathrm{RSS}(\hat{\boldsymbol{\beta}})}{\partial \hat{\boldsymbol{\beta}}} = 2 \cdot (\boldsymbol{X}^T \boldsymbol{X} \hat{\boldsymbol{\beta}} - \boldsymbol{X}^T \boldsymbol{y}) \; . \tag{89}$$

The minimum of (89), hence the optimal weights $\hat{\boldsymbol{\beta}}$, is given by the closed form solution, referred to as the *normal equations*:

$$\hat{\boldsymbol{\beta}}^{\mathrm{MLE}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \; . \tag{90}$$

The MLE yields the optimal weights for modeling the data, exclusively used in the optimization and training process. As a consequence, this method is prone to overfit the data.

B.2 Gaussian & Laplacian Prior

In order to reduce overfitting, and aim for a more general choice of weights, we encourage the weights to be small by defining a zero-mean Gaussian prior:

$$p(\hat{\boldsymbol{\beta}}) = \prod_{j=1}^{N} \mathcal{N}(\hat{\beta}_j|0,\tau^2) \; . \tag{91}$$

Here, the free parameter τ^2 controls the strength of the prior, and we do not put any constraint on the bias weight $\hat{\beta}_0$. Consequently, the MAP estimate is given by the likelihood and the non-uniform prior:

$$\hat{\boldsymbol{\beta}}^{\text{MAP}} = \text{argmax}_{\hat{\boldsymbol{\beta}}} \left[\sum_{i=1}^{M} \log \mathcal{N}(y^{(i)}|\hat{\boldsymbol{\beta}}^T \boldsymbol{x}^{(i)}, \sigma^2) + \sum_{j=1}^{N} \log \mathcal{N}(\hat{\beta}_j|0,\tau^2) \right] \; . \tag{92}$$

Again, we have used a more convenient expression by taking the logarithm. Similar to the previous approach, we obtain an expression for the new cost function:

$$J(\hat{\boldsymbol{\beta}}) = \frac{1}{M} \sum_{i=1}^{M} (y^{(i)} - \hat{\boldsymbol{\beta}}^T \boldsymbol{x}^{(i)})^2 + \lambda \sum_{j=1}^{N} \hat{\beta}_j^2 \; , \tag{93}$$

where we have defined a *regularizaton parameter* with $\lambda \equiv 1/2\tau^2$. The first term in (93) corresponds to the OLS approach, while the second term comes into play due to the defined prior in (91). This regression approach is commonly referred to as *Tikhonov regularization* [51] or *ridge regression*. The analytical solution of the weights, which minimize equation (93), is given by

$$\hat{\boldsymbol{\beta}}^{\text{ridge}} = (\lambda \boldsymbol{I}_{N+1} + \boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \; , \tag{94}$$

where $\boldsymbol{I}_{N+1}$ denotes the identity matrix with dimension $N+1$. Here, the matrix dimension is $N+1$ due to the bias term being considered explicitly.

For the *least absolute shrinkage and selection operator* (lasso) method, we consider a prior of the form

$$p(\hat{\boldsymbol{\beta}}) = \prod_{j=1}^{N} \text{Lap}(\hat{\beta}_j|0,1/\lambda) \propto \prod_{j=1}^{N} e^{-\lambda|\hat{\beta}_j|} \; . \tag{95}$$

Similar to previous approaches, we obtain the corresponding cost function

$$J(\hat{\boldsymbol{\beta}}) = \frac{1}{M} \sum_{i=1}^{M} (y^{(i)} - \hat{\boldsymbol{\beta}}^T \boldsymbol{x}^{(i)})^2 + \lambda \sum_{j=1}^{N} |\hat{\beta}_j| \; , \tag{96}$$

which in contrast to (93) has no closed form solution due to the absolute value function being not differentiable at $\hat{\beta}_j = 0$.

Appendix C

The Kernel Trick

In order to demonstrate the application of the kernel trick, we recall the shape and dimension of the design matrix and its transpose, which we explicitly express in terms of a basis function expansion $\boldsymbol{\phi}(\boldsymbol{x})$:

$$\boldsymbol{X} = \begin{bmatrix} -\boldsymbol{\phi}(\boldsymbol{x}^{(1)})^T - \\ -\boldsymbol{\phi}(\boldsymbol{x}^{(2)})^T - \\ \vdots \\ -\boldsymbol{\phi}(\boldsymbol{x}^{(M)})^T - \end{bmatrix} \in \mathbb{R}^{M \times N'} \quad , \quad \boldsymbol{X}^T = \begin{bmatrix} | & & | \\ \boldsymbol{\phi}(\boldsymbol{x}^{(1)}) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}^{(M)}) \\ | & & | \end{bmatrix} \in \mathbb{R}^{N' \times M} \, , \tag{97}$$

where M denotes the number of training examples, and N' is the dimension of the feature space after the basis transformation. The MAP approach, including the zero-mean Gaussian prior, is discussed in appendix B.2, and yields the analytical solution:

$$\hat{\boldsymbol{\beta}} = (\lambda \mathbf{I}_{N'} + \boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \, . \tag{98}$$

Here, the cost of the matrix inversion is $\mathcal{O}(N'^3)$. We use the matrix inversion lemma or *Sherman-Morrison-Woodbury formula* [52]

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{D})^{-1}\boldsymbol{B}\boldsymbol{C}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{C} + \boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \tag{99}$$

to rewrite equation (98) as follows:

$$\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_M)^{-1}\boldsymbol{y} \, . \tag{100}$$

The matrix $\boldsymbol{X}\boldsymbol{X}^T$ is called the *Gram matrix* of $\boldsymbol{X}$, and its shape is given by

$$\boldsymbol{X}\boldsymbol{X}^T = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}^{(1)})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}^{(1)}) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}^{(1)})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}^{(M)}) \\ & \vdots & \\ \boldsymbol{\phi}(\boldsymbol{x}^{(M)})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}^{(1)}) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}^{(M)})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}^{(M)}) \end{bmatrix} \in \mathbb{R}^{M \times M} \, . \tag{101}$$

According to Mercer's theorem and the proof of $\kappa(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \boldsymbol{\phi}(\boldsymbol{x}^{(i)})^T \cdot \boldsymbol{\phi}(\boldsymbol{x}^{(j)})$, we replace the matrix $\boldsymbol{X}\boldsymbol{X}^T$ with the Gram matrix

$$\boldsymbol{K} = \begin{bmatrix} \kappa(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & \cdots & \kappa(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(M)}) \\ & \vdots & \\ \kappa(\boldsymbol{x}^{(M)}, \boldsymbol{x}^{(1)}) & \cdots & \kappa(\boldsymbol{x}^{(M)}, \boldsymbol{x}^{(M)}) \end{bmatrix} \, , \tag{102}$$

and obtain the partially kernelized optimization problem

$$\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T(\boldsymbol{K} + \lambda \boldsymbol{I}_M)^{-1}\boldsymbol{y} \; . \tag{103}$$

We perform a Legendre transformation by defining the so-called *dual variables* as follows:

$$\hat{\boldsymbol{\alpha}} \equiv (\boldsymbol{K} + \lambda \boldsymbol{I}_M)^{-1}\boldsymbol{y} \; . \tag{104}$$

Here, the cost of the matrix inversion is $\mathcal{O}(M^3)$. The primal weights in (103) thus are given as a linear sum of the feature vectors of the M training examples:

$$\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T\hat{\boldsymbol{\alpha}} = \sum_{i=1}^{M} \hat{\alpha}^{(i)}\boldsymbol{\phi}(\boldsymbol{x}^{(i)}) \; . \tag{105}$$

We use this relation to finally kernelize the hypothesis function (39) as well:

$$h(\boldsymbol{x}) = \hat{\boldsymbol{\beta}}^T\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{i=1}^{M} \hat{\alpha}^{(i)}\boldsymbol{\phi}(\boldsymbol{x}^{(i)})^T\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{i=1}^{M} \hat{\alpha}^{(i)}\kappa(\boldsymbol{x}, \boldsymbol{x}^{(i)}) \; . \tag{106}$$

By this means, the optimization problem can be completely performed in the kernelized dual problem, and the original computational effort $\mathcal{O}(N'^3)$ in the primal problem is changed to $\mathcal{O}(M^3)$ in the dual problem.

References

[1] Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical review*, 136(3B):B864, 1964.

[2] Walter Kohn and Lu Jeu Sham. Self-consistent equations including exchange and correlation effects. *Physical review*, 140(4A):A1133, 1965.

[3] Colin W Glass, Artem R Oganov, and Nikolaus Hansen. Uspex: Evolutionary crystal structure prediction. *Computer physics communications*, 175(11-12):713–720, 2006.

[4] Stefan Goedecker. Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems. *The Journal of chemical physics*, 120(21):9911–9917, 2004.

[5] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[6] Katja Hansen, Grégoire Montavon, Franziska Biegler, Siamac Fazli, Matthias Rupp, Matthias Scheffler, O Anatole Von Lilienfeld, Alexandre Tkatchenko, and Klaus-Robert Müller. Assessment and validation of machine learning methods for predicting molecular atomization energies. *Journal of Chemical Theory and Computation*, 9(8):3404–3419, 2013.

[7] Katja Hansen, Franziska Biegler, Raghunathan Ramakrishnan, Wiktor Pronobis, O Anatole Von Lilienfeld, Klaus-Robert Müller, and Alexandre Tkatchenko. Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. *The journal of physical chemistry letters*, 6(12):2326–2331, 2015.

[8] Ghanshyam Pilania, Chenchen Wang, Xun Jiang, Sanguthevar Rajasekaran, and Ramamurthy Ramprasad. Accelerating materials property predictions using machine learning. *Scientific reports*, 3:2810, 2013.

[9] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.

[10] Jörg Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *The Journal of chemical physics*, 134(7):074106, 2011.

[11] Albert P Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013.

[12] Runhai Ouyang, Stefano Curtarolo, Emre Ahmetcik, Matthias Scheffler, and Luca M Ghiringhelli. Sisso: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates. *Physical Review Materials*, 2(8):083802, 2018.

[13] SR Xie, GR Stewart, JJ Hamlin, PJ Hirschfeld, and RG Hennig. Functional form of the superconducting critical temperature from machine learning. *arXiv preprint arXiv:1905.06780*, 2019.

[14] KT Schütt, H Glawe, F Brockherde, A Sanna, KR Müller, and EKU Gross. How to represent crystal structures for machine learning: Towards fast prediction of electronic properties. *Physical Review B*, 89(20):205118, 2014.

[15] Shreyas Honrao, Bryan E Anthonio, Rohit Ramanathan, Joshua J Gabriel, and Richard G Hennig. Machine learning of ab-initio energy landscapes for crystal structure predictions. *Computational Materials Science*, 158:414–419, 2019.

[16] David Wesley. Development of an angular distribution function for the study of atomic lattice structures used in atomistic simulation. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING, 1991.

[17] David Sholl and Janice A Steckel. *Density functional theory: a practical introduction*. John Wiley & Sons, 2011.

[18] John P Perdew and Stefan Kurth. Density functionals for non-relativistic coulomb systems in the new century. In *A primer in density functional theory*, pages 1–55. Springer, 2003.

[19] M. Born and R. Oppenheimer. Zur Quantentheorie der Molekeln. *Annalen der Physik*, 389:457–484, 1927.

[20] EKU Gross and Walter Kohn. Local density-functional theory of frequency-dependent linear response. *Physical review letters*, 55(26):2850, 1985.

[21] John P Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical review letters*, 77(18):3865, 1996.

[22] John P Perdew and Karla Schmidt. Jacob's ladder of density functional approximations for the exchange-correlation energy. In *AIP Conference Proceedings*, volume 577, pages 1–20. AIP, 2001.

[23] Charles Kittel, Paul McEuen, and Paul McEuen. *Introduction to solid state physics*, volume 8. Wiley New York, 1976.

[24] Warren J Hehre, Robert F Stewart, and John A Pople. self-consistent molecular-orbital methods. i. use of gaussian expansions of slater-type atomic orbitals. *The Journal of Chemical Physics*, 51(6):2657–2664, 1969.

[25] John C Slater. Wave functions in a periodic potential. *Physical Review*, 51(10):846, 1937.

[26] Norman Troullier and José Luís Martins. Efficient pseudopotentials for plane-wave calculations. *Physical review B*, 43(3):1993, 1991.

[27] G Kresse and J Furthmüller. Software vasp, vienna (1999). *Phys. Rev. B*, 54(11):169, 1996.

[28] AR Oganov and VL Solozhenko. Boron: a hunt for superhard polymorphs. *Journal of Superhard Materials*, 31(5):285, 2009.

[29] AP Drozdov, MI Eremets, IA Troyan, Vadim Ksenofontov, and SI Shylin. Conventional superconductivity at 203 kelvin at high pressures in the sulfur hydride system. *Nature*, 525(7567):73, 2015.

[30] J Pannetier, J Bassas-Alsina, J Rodriguez-Carvajal, and V Caignaert. Prediction of crystal structures from crystal chemistry rules by simulated annealing. *Nature*, 346(6282):343, 1990.

[31] R Martoňák, Alessandro Laio, and Michele Parrinello. Predicting crystal structures: The parrinello-rahman method revisited. *Physical review letters*, 90(7):075503, 2003.

[32] David J Wales and Jonathan PK Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

[33] Li Zhu, Maximilian Amsler, Tobias Fuhrer, Bastian Schaefer, Somayeh Faraji, Samare Rostami, S Alireza Ghasemi, Ali Sadeghi, Migle Grauzinyte, Chris Wolverton, et al. A fingerprint based metric for measuring similarities of crystalline structures. *The Journal of chemical physics*, 144(3):034203, 2016.

[34] Mario Valle and Artem R Oganov. Crystal fingerprint space–a novel paradigm for studying crystal-structure sets. *Acta Crystallographica Section A: Foundations of Crystallography*, 66(5):507–517, 2010.

[35] Artem R Oganov and Mario Valle. How to quantify energy landscapes of solids. *The Journal of chemical physics*, 130(10):104504, 2009.

[36] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2017.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[39] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[40] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[41] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[42] James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.

[43] Andrei Nikolajevits Tihonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math.*, 4:1035–1038, 1963.

[44] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[45] NZ Shor. New development trends in nondifferentiable optimization. *Cybernetics and Systems Analysis*, 13(6):881–886, 1977.

[46] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[47] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[48] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[49] Dil K. Limbu, Michael U. Madueke, Raymond Atta-Fynn, David A. Drabold, and Parthapratim Biswas. Ab initio density-functional studies of 13-atom cu and ag clusters. *arXiv: 1809.00300*, 2018.

[50] Kevin P Murphy. Machine learning: A probabilistic perspective, ; adaptive computation and machine learning series, 2018.

[51] CW Groetsch. The theory of tikhonov regularization for fredholm equations. *104p, Boston Pitman Publication*, 1984.

[52] William W Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989.

Acknowledgements

First of all, I want to thank my supervisor Wolfgang von der Linden for the opportunity to work on this interesting topic. The more I have dealt with the subject, the more I became aware of its elegance, packed with challenges and pleasures.
I want to thank my co-supervisor Bernhard C. Geiger from the Know-Center in Graz for sharing his valuable Know-How on the practical use of machine learning.
Furthermore, I would like to express my deep gratitude to Lilia Boeri for her time, advice and great amount of assistance.
I would also like to show my greatest appreciation to Santanu, Simone and my elder brother Christian for various discussions, their constant support and contribution to this work.
The author would like to acknowledge the use of HPC resources provided by the ZID of the Graz University of Technology.
Finally I want to thank the ones who contribute the most to this work by making this education possible in the first instance. This are my parents Christine and Franz, my grandparents Gertraud and Franz and my aunt Veronika.
Thank you so much for your support!