Functional regression of densities with application to the simulation of molecular dynamics

vorgelegt von
M.Sc.
Felix Brockherde
geb. in Stadtlohn

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Klaus Obermayer
Gutachter:	Prof. Dr. Kieron Burke
Gutachter:	Prof. Dr. Marius Kloft
Gutachter:	Prof. Dr. Benjamin Blankertz

Tag der wissenschaftlichen Aussprache: 9. August 2018

Berlin 2018

Abstract

Applications of machine learning have shown promising results modeling the non-interacting kinetic energy functional in 1-D. This holds the promise of enabling orbital-free density functional theory calculations, by-passing the computationally expensive Kohn-Sham equations. This would yield substantial savings in computer-time so that larger systems or longer time scales can be simulated. These approaches, however, where limited by the need to model an accurate derivate. This thesis investigates possibilities to overcome this difficulty and extend the principle to 3-D molecules. Presented is a machine learning method for training a direct potential to density model, a machine learning Hohenberg-Kohn map. The method is set on a theoretical foundation via functional data analysis and operator-valued kernel models. The new approach is rigorously evaluated on 1-D data and applied to small molecules in 3-D. Practical challenges like data generation, sampling, and normalization problems are solved. Finally, an application to molecular dynamics simulation evaluates the method's robustness and demonstrates its practical value. The result of this thesis is a successful extension of the orbital-free principle to 3-D. The Hohenberg-Kohn map outperforms the previous approach in 1-D and archives competitive results for 3-D molecules. It even proves to be robust enough to make molecular dynamics simulations with a machine learned density functional possible and allows investigation of rare events that were not included in the training set. The Hohenberg-Kohn map thus allows construction of accurate density functionals for realistic molecular systems.

Zusammenfassung

Die Anwendung von maschinellem Lernen zeigte vielversprechende Ergebnisse beim Modellieren des nicht-interagierenden kinetischen Energiefunktionals in 1-D. Dies bringt die Aussicht Orbital-freie Dichtefunktionaltheorie Berechnungen zu ermöglichen, somit die rechenintensiven Kohn-Sham Gleichungen zu umgehen, und erhebliche Reichenzeit einzusparen, sodass größere Systeme oder längere Zeitskalen simuliert werden können. Diese Ansätze waren aber beschränkt bei der Notwendigkeit eine genaue Ableitung zu modellieren. Diese Dissertation untersucht Möglichkeiten diese Schwierigkeiten zu überwinden und das Prinzip auf 3-D Moleküle zu erweitern. Präsentiert wird eine Methode des maschinellen Lernens um ein direktes Potential-zu-Dichtefunktion Modell zu trainieren: eine Hohenberg-Kohn Abbildung via maschinelles Lernen. Die Methode wird auf eine theoretische Grundlage gesetzt mit Hilfe von funktionaler Datenanalyse und operatorwertigen Kernmethoden. Der neue Ansatz wird auf 1-D Daten und angewendet auf kleine Moleküle in 3-D. Praktische Herausforderungen wie Datengenerierung, Abtastung, und Normalisierung werden gelöst. Schließlich evaluiert eine Anwendung auf Moleküldynamiksimulationen die Robustheit der Methode und demonstriert ihren praktischen Wert. Das Ergebnis der Dissertation ist eine erfolgreiche Erweiterung des Orbital-freien Prinzips auf 3-D. Die Hohenberg-Kohn Abbildung übertrifft den vorherigen Ansatz und erreicht konkurrenzfähige Ergebnisse auf 3-D Molekülen. Sie weist sich sogar als robust genug heraus um Moleküldynamiksimulationen mit einem maschinell gelernten Dichtefunktional zu ermöglichen und erlaubt die Untersuchung von seltene Ereignissen, die nicht im Trainingsdatensatz enthalten waren. Die Hohenberg-Kohn Abbildung erlaubt somit die Konstruktion von genauen Dichtfunktionalen für realistische Molekülsysteme.

Acknowledgements

I am deeply grateful for having worked with so many brilliant and inspiring minds over the last years. I would like to thank Prof. Dr. Klaus-Robert Müller from TU Berlin for supervising this PhD thesis. He has been the visionary of this work and has provided an invaluable stream of support throughout its creation. For accompanying this thesis from a natural science point, I would like to thank Prof. Dr. Kieron Burke from the University of California at Irvine who has repeatedly explained complex chemical and physical concepts to my layman understanding and provided the theoretical foundation for this work's application.

For fruitful collaboration, I owe gratitude to Dr. Li Li from the University of California at Irvine, Dr. Leslie Vogt and Prof. Mark E. Tuckerman from the New York University, Dr. John Snyder from TU Berlin, and Dr. Henning Glawe and Prof. Dr. Hardy E.K.U Gross from the Max-Planck-Institute of Microstructure Physics in Halle.

I thank the Machine Learning Group at TU Berlin and its members, especially Kristof Schütt and Stefan Chmiela, for inspiring discussions and all the help received during this research time.

The Institute of Pure and Applied Mathematics at the University of California, Los Angeles, receives my gratitude for repeated hospitality and workshop organization without which this work could not have progressed as effectively.

I would like to thank the TU Berlin and the Max-Planck-Institute of Microstructure Physics in Halle for financing this research.

Finally, I owe my deepest gratitude to my wife Silvina Colombato for supporting me in finishing this thesis.

Table of Contents

Acronyms

CCSD(T) coupled cluster single double estimated triple
DFT density functional theory
FCI full configuration interaction
FDA functional data analysis
FFT fast Fourier transform
GTO Gaussian-type orbital
HK Hohenberg-Kohn
iid independently and identically distributed
KE kinetic energy
Kernel PCA Kernel principle component analysis
KRR kernel ridge regression
KS Kohn-Sham
KS-DFT Kohn-Sham density functional theory
LDA local density approximation exchange correlation
MAE mean average error
MD molecular dynamics
ML machine learning
ML-HK machine-learning-Hohenberg-Kohn
ML-KS machine-learning-Kohn-Sham
MO molecular orbital
MSE mean squared error
OF orbital-free
PBE Perdew-Burke-Ernzerhof exchange correlation
PCA principle component analysis
PES potential energy surface
QM quantum mechanics
RKHS reproducing kernel Hilbert space
RRM regularized risk minimization
SCF self-consistent field

1

Motivation and Background

1.1 Introduction

In recent years, there has been a surge in application of machine learning to problems in the natural sciences. The fundamental task for finding new compounds or materials is the prediction of chemical and physical quantities. The complex quantum mechanical effects involved make costly electronic structure calculations necessary. Calculating properties for just one configuration when screening databases for promising materials or simulating molecular dynamics that require high levels of accuracy from the quantum-chemical methods can take several days on high-performance clusters. Machine learning now offers to learn from previous recorded calculations and to predict properties via machine learning models that give instantaneous results. Successful application of machine learning can thus speed-up compound design in pharmaceutical industries or material design and discovery.

The majority of these machine learning applications involve predicting properties of molecules or materials from large databases of Kohn-Sham density functional theory (KS-DFT) calculations. A few applications involve finding potential energy surfaces within molecular dynamics simulations.

These approaches treat KS-DFT as a black-box and neglect the theoretical achievements in this area. However, promising research has recently shown

that a tighter integration of machine learning and density functional theory seems possible. This would allow us to take advantage of well-known parts of density functional theory and its matured methodology. If such attempts could be made practical for real world systems, we can expect an enormous speed-up in repeated density functional theory (DFT) calculations of similar species, e.g. in ab initio molecular dynamics (MD) simulations.

Since the previous research focused on 1-D toy systems, this thesis covers the adaption of these ideas to 3-D systems of real molecules.

The research dives into two rather distinct disciplines, quantum chemistry and machine learning, and is thus fundamentally interdisciplinary. Challenging was not only the gap between quantum chemistry and machine learning, but also the combination of DFT and MD. Since machine learning is a new set of tools for physicists and chemists, there is a high standard of proof that the approaches (a) actually work and (b) will become applicable and useful for practitioners.

At the beginning of the research that is documented here, both machine learning and the application of machine learning in the area of electronic structure calculations where in a different place. The machine learning research was dominated by kernel methods, whereas the recent years fueled a renewed interest in neural networks, especially deep learning. The prediction of functional data, however, is still a well suited problem for kernel methods. Although there was earlier work that applies machine learning (ML) methods to speed up DFT tasks, it was rather exotic. Nowadays ML is an established part of projects trying to go beyond what is possible with standard electronic structure codes.

When the thesis commenced it was thought to go into a different direction. The original idea was to search for superconductors and evaluate crystal properties. A fruitful collaboration with the Max-Planck-Institute of Microstructure Physics in Halle laid a foundation [Sch+14]. However, once it was realized how ground-breaking and impactful ML in physics would become, the research shifted to more fundamental and general problems. From a physical point of view, it offered the challenge of having to dive much deeper into density functional theory and electronic structure codes. From the machine learning perspective, it offers the challenge of working with high-dimensional structured objects such as 3-D density functions.

Due to the interdisciplinary nature of this thesis, it has to strive for a balance between explaining all relevant parts for reaching the conclusions on the one hand and the limited space available on the other hand. This is why the following sections give only a short introduction into the problem setting from a natural science point of view and how machine learning can be applied.

The explicit goal of this thesis is to document the research and ideas that are tremendously interesting, but were *not* published, either because they did not work well, did not yield significant improvements, or were not important enough to include in the publication. Since it is not subject to peer review, the PhD thesis is the perfect medium for this.

The thesis starts with simple 1-D toy problems but continues all the way to 3-D MD simulations. Its results have thus drawn interest far beyond the ML, DFT, or MD community.

Being able to do research in this area and being a part of the developments in this field have been an honor and will continue to shape the author's engineering and scientific mind. I wish all curious readers an interesting time with many new and inspiring ideas.

Felix Brockherde
Berlin, 2018

1.2 From quantum mechanics to machine learning

This section gives a brief introduction to the physical and chemical background of the thesis. It introduces the basics of quantum mechanics (QM), DFT and MD. Finally, the possibility to apply machine learning in these areas is discussed. It also sets the notation for the following chapters.

The state of a quantum mechanical system is described mathematically by a wavefunction $\Psi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N)$ which is a function of, for example, the positions of all particles in the system.

The stationary states of a quantum system are the solutions of the time-

independent Schrödinger equation:

$$\widehat{H}\Psi = \left[\widehat{T} + \widehat{V}_{\text{ee}} + \widehat{V}_{\text{ext}}\right]\Psi = E\Psi. \tag{1.1}$$

$\widehat{H}$ is the Hamiltonian operator. It consists of the kinetic energy operator $\widehat{T}$, the external potential energy operator $\widehat{V}_{\text{ext}}$ and the electron-electron interaction energy operator $\widehat{V}_{\text{ee}}$. The external potential energy stems from the external field generated by the nuclei in the system and thus makes the equation unique for every system. The electron-electron interaction energy operator prohibits separation into simpler single-particle equations. The constant E is the energy of the state. Solving the Schrödinger equation is a fundamental problem in quantum mechanics. Exact solutions are known only for the smallest systems. In order to solve the Schrödinger equation for interesting systems, it is necessary to make approximations, thus trading speed for accuracy. The amount of accuracy we are willing to trade depends on the application.

The first approximation we make is the so-called Born-Oppenheimer approximation. It is the assumption that the motion of atomic nuclei and electrons can be separated, and allow us to treat the motion of atomic nuclei and electrons independently. For now we assume that the atomic nuclei are fixed in position.

1.2.1 Density functional theory (DFT)

A good theory that operates within the Born-Oppenheimer approximation and that provides us with an excellent trade-off between accuracy and speed is KS-DFT[KS65]. We first introduce DFT and then describe the Kohn-Sham (KS) system.

The fundamental variable in DFT is the electron density. It is a probability density and can be interpreted as a relative likelihood of finding an electron at a specific point in space. It is always positive (≥ 0) and integrates to the number of electrons N in the system. For a given normalized ground-state Ψ, it is

defined as[1]

$$n(\boldsymbol{r}) = N \int \mathrm{d}^3 r_2 \cdots \int \mathrm{d}^3 r_N \, \Psi^*(\boldsymbol{r}, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N) \Psi(\boldsymbol{r}, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N). \tag{1.2}$$

The foundation of modern DFT is the Hohenberg-Kohn (HK) theorem[HK64] It states that, first, the ground-state density n uniquely determines the external potential v of a many-body system (up to a trivial additive constant). Thus, the HK theorem establishes a *one-to-one* relationship between ground-state density and external potential. Assuming only non-degenerate[2] states we can write the wave-function as a functional of the ground-state density: $\Psi[n]$. The HK theorem also states that the ground-state density minimizes the energy functional

$$E_v[n] = \int \mathrm{d}^3 r \, v(\boldsymbol{r}) \, n(\boldsymbol{r}) + F[n] \tag{1.3}$$

if we minimize over the set of positive normalized densities. Here, the kinetic energy and coulomb repulsion,

$$\widehat{T} = -\frac{1}{2} \sum_{i=1}^{N} \nabla_i^2 \quad \text{and} \tag{1.4}$$

$$\widehat{V}_{\mathrm{ee}} = \frac{1}{2} \sum_{i \neq j} \frac{1}{\left| \boldsymbol{r}_i - \boldsymbol{r}_j \right|}, \tag{1.5}$$

are grouped in the universal many-body functional

$$F[n] = \left\langle \Psi[n] \middle| \widehat{T} + \widehat{V}_{\mathrm{ee}} \middle| \Psi[n] \right\rangle. \tag{1.6}$$

The function $v(\boldsymbol{r})$ is the external potential. For example, given a number of atoms with nuclear charges $Z_1, \ldots, Z_{N^{\mathrm{atoms}}}$ and positions $R_1, \ldots, R_{N^{\mathrm{atoms}}}$, the Coulomb potential is given by

$$v(\boldsymbol{r}) = \sum_{i=1}^{N^{\mathrm{atoms}}} -\frac{Z_i}{\left\| R_i - \boldsymbol{r} \right\|}. \tag{1.7}$$

For plane-wave (Fourier) basis sets, pseudo-potentials are preferred in practical

[1] Note that we adopt the physics notation of writing the differential right after the integral sign. We also use the shorthand $\mathrm{d}^3 r$ for $\mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z$.

[2] Otherwise, we define $\Psi[n]$ to be the wave-function that yields n and minimizes $\widehat{T} + \widehat{V}_{\mathrm{ee}}$.

applications. They represent the core electrons and are frozen so that only the valence electrons are dealt with explicitly.

The functional F is universal in the sense that it is valid for any external potential. Minimizing $E[n]$ under the constraint that the density is normalized, i.e. $\int \mathrm{d}^3 r\, n(\boldsymbol{r}) = N$, yields

$$\frac{\delta}{\delta n}\left[F[n] + \int \mathrm{d}^3 r\, v(\boldsymbol{r})\, n(\boldsymbol{r}) - \mu\left(\int \mathrm{d}^3 r\, n(\boldsymbol{r}) - N\right)\right] = 0 \tag{1.8}$$

where μ is the Lagrange multiplier associated with the normalization constraint. The minimizing density must fulfill the corresponding Euler-Lagrange equation

$$\frac{\delta F}{\delta n(\boldsymbol{r})} + v(\boldsymbol{r}) = \mu. \tag{1.9}$$

Let us note that DFT itself did not introduce any approximation, it is just formulated in terms of densities as fundamental quantities, not wave-functions.

Since we can easily derive the density from a wave-function but not the other way round, we might worry that we lose information about our system by solely looking at the density. But since the HK theorem provides us with a one-to-one relationship between ground-state density and external potential, we know that every observable of our system can be expressed as a functional of the ground-state density. This allows us to work with three dimensional densities instead of $3N$ dimensional wave-functions.

The second part of the HK theorem (Eq. 1.3) leads us to a strategy for finding the ground-state density via iterative optimization. However, the universal many-body functional F is unknown. The idea of applying ML to learn the non-interacting kinetic energy functional is explored in the work of Snyder et al. [Sny+12], Snyder et al. [Sny+13b], and Snyder et al. [Sny+15]. Li et al. [Li+16a] explores the idea of learning a universal functional to solve chains of 1-D hydrogen atoms. We will come back to this later. First, we introduce the KS approximation.

Instead of approximating F directly, KS-DFT imagines a fictitious system of *non-interacting* electrons that gives rise to the same density as the original *interacting* system. The potential that this non-interacting system is exposed to is called the

KS potential v_s. The orbitals of the non-interacting system are given by the non-interacting Schrödinger equation

$$\left\{-\frac{1}{2}\nabla^2 + v_\mathrm{s}(\boldsymbol{r})\right\}\varphi_i(\boldsymbol{r}) = \epsilon_i\varphi_i(\boldsymbol{r}) \tag{1.10}$$

and its density by

$$n(\boldsymbol{r}) = \sum_{i=1}^{N}|\varphi_i(\boldsymbol{r})|^2. \tag{1.11}$$

The corresponding Euler-Lagrange equation is given by

$$\frac{\delta T_s}{\delta n(\boldsymbol{r})} + v_\mathrm{s}(\boldsymbol{r}) = \mu \tag{1.12}$$

where T_s is the kinetic energy of the non-interacting electrons. We can now write F of the interacting system in terms of the non-interacting kinetic energy T_s. Since the Hartree energy

$$U[n] = \frac{1}{2}\int \mathrm{d}^3r \int \mathrm{d}^3r' \, \frac{n(\boldsymbol{r})\, n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} \tag{1.13}$$

is known and contributes a large part to the difference between T_s and T, we write it out explicitly and have

$$F[n] = T_s[n] + U[n] + E_\mathrm{XC}[n]. \tag{1.14}$$

The remainder is contained in E_XC which is thus implicitly defined by Eq. 1.14. Inserting Eq. 1.14 into Eq. 1.9 and comparing to 1.12 yields

$$v_\mathrm{s}(\boldsymbol{r}) = v(\boldsymbol{r}) + v_\mathrm{H}(\boldsymbol{r}) + v_\mathrm{XC}[n](\boldsymbol{r}), \quad v_\mathrm{XC}[n](\boldsymbol{r}) = \frac{\delta E_\mathrm{XC}}{\delta n(\boldsymbol{r})}. \tag{1.15}$$

where $v_\mathrm{H}(\boldsymbol{r}) = {}^{\delta U}\!/_{\delta n(\boldsymbol{r})} = \int \mathrm{d}^3r' \, {}^{n(\boldsymbol{r}')}\!/_{|\boldsymbol{r}-\boldsymbol{r}'|}$ is the Hartree potential.

Note that the system has to be solved *self-consistently* since the KS potential depends on the density. KS-DFT makes no approximations in itself and is thus an exact theory. However, the exchange-correlation functional E_XC is unknown and we have to resort to approximations of E_XC to use KS-DFT. Several good

approximations exist, providing varying levels of accuracy.

A simple approximation is local density approximation (LDA) which solely depends on the value of the density n at each point in space, e.g.

$$E_{\mathrm{XC}}^{\mathrm{LDA}} = \int \mathrm{d}^3 r n(\boldsymbol{r}) \epsilon_{\mathrm{XC}}(n) \tag{1.16}$$

where ϵ_{XC} is the exchange-correlation energy per particle of a homogeneous electron gas of charge density n. A better approximation that will be used in Chapter 4 is the Perdew-Burke-Ernzerhof (PBE) exchange-correlation potential.

For more resources on KS-DFT, refer to Burke and Wagner [BW12].

1.2.2 Molecular dynamics

So far the atomic nuclei where always assumed fixed in their position and only the electrons were allowed to move. Each set of fixed atomic positions for a molecule is called a *configuration*. The Born-Oppenheimer approximation allowed us to separate the motion of atomic nuclei and electrons. The atomic nuclei have so far been fixed in space. We now want to look at their motion. This approach is called molecular dynamics (MD) and we use it to simulate the physical movement of atoms. The atoms are allowed to interact for a certain period of time allowing us to view the dynamical evolution of the system. This allows us to observe properties that we can not observe by investigating one configuration alone. We will later apply MD to the malonaldehyde molecule and observe proton transfer events. This allows us to estimate the energy barrier for the intramolecular proton transfer.

The forces that act on the atomic nuclei can be described by Newton's second law of motion. We can write the forces in a pair of first order differential equations

$$F(X) = M\frac{\partial V(t)}{\partial t} \tag{1.17}$$

$$V(t) = \frac{\partial X(t)}{\partial t} \tag{1.18}$$

where we write $V(t)$ for the velocities and $X(t)$ for the positions $\boldsymbol{r}$ at time t.

The equations of motion (Eqs. 1.17 and 1.18) form a system of ordinary differential equations. For N^{atoms} atoms, there are $3N^{\text{atoms}}$ coordinates and $3N^{\text{atoms}}$ velocities. Since an analytical solution is impossible, we turn to numerical solutions.

Fortunately, the numerical solution is straight forward. We iteratively evaluate velocities and forces of each atom and take small step ϵ forward, i.e. for a moment i in time

$$X_{i+1} = X_i + \epsilon V_i \tag{1.19}$$

$$V_{i+1} = V_i + \epsilon \frac{F(X_i)}{M}. \tag{1.20}$$

Later, we use the Velocity Verlet algorithm which averages two timesteps to increase robustness. Additionally, a so-called thermostat can be used to control the temperature of the simulation. To start the simulations, we initialize the velocities V_0 by drawing randomly from a Maxwell-Boltzman distribution so that the initial kinetic energy yields the desired temperature of the simulation.

To find the forces F, we differentiate between classical MD and ab initio MD.

- Classical MD. In classical MD, we define a force field which uses an interatomic potential or energy function. It can be written as a parametrized sum of terms that depend on the position of the atoms. We then can calculate the force acting on an atom as derivatives of the total energy with respect to the atom position

  $$F(X) = -\nabla_X E. \tag{1.21}$$

  The forces are typically fast to evaluate but often not sufficiently accurate because they can only take quantum effects into account by fitting the parameters to e.g. experimental data or quantum chemistry calculations. As quantum effects can be greatly affected by the molecular environment, no single parameterization can account for all environments.

- Ab initio MD. Forces can be obtained from ab initio calculations, for example DFT calculations as discussed above. However, these calculations are computationally much more expensive. To run an ab initio MD simu-

lation, we have run one DFT calculation per time step. However, it is not necessary to start from scratch in every step, it is usually possible to, for example, use the previous MD-step's electron density or wave function in the start of the Kohn-Sham loop.

At each step of the simulation, we thus have to first compute the forces acting on each atom and, secondly, move the atoms a little bit, i.e. update the positions and velocities of each atom using Newton's laws of motion.

1.2.3 Machine learning

KS-DFT is the milestone that made solving many electronic structure problems possible. For some calculations, however, especially for bigger systems, the required computational resources are still too high. At the same time, these calculations need the accuracy of KS-DFT so that introducing approximations that run faster is not helpful. This case is where machine learning comes into play. With ML we have the ability to learn models from data that can represent physical relationships that require too many computational resources to evaluate.

Three scenarios that could benefit from ML stand out.

- Searching databases for materials or molecules that have specified desirable properties: In some cases we are looking for a molecule or material with specific properties that we expect to be part of a given database, i.e. a database that indexes all published materials, or that systematically indexes all possible molecules up to a certain size. In this case we can calculate the properties of interest either with DFT or even more accurate methods of a subset of the database and use ML to create a model with which we can predict the properties of the remaining set.

  An example application is the search for new superconducting materials. Newly discovered superconductors often turn out to have been used in industry in other contexts, i.e. the structures are well known. It is thus possible to predict properties that influence superconductivity with machine learning [Sch+14].

- MD SIMULATIONS: In order to simulate molecular dynamics it is required to repeatedly run electronic structure calculations with slightly varying molecular geometries to calculate forces and energies. Additionally, an MD trajectory will repeatedly cross its previous path. MD simulations are therefore perfect application environments for ML.
- STRUCTURE SEARCH: Finding new stable structures, i.e. structures that are not part of known databases, with desirable properties is a much harder problem than finding properties for a given molecule or material. With DFT we have a computationally challenging, but conceptually straight-forward method to go from atomic positions $(Z_1, R_1), \ldots, (Z_{N^{\text{atoms}}}, R_{N^{\text{atoms}}})$ to a certain property. The inverse problem can only be tackled with more or less trial and error approaches. ML can help conceptually by providing smarter search strategies and by providing models for specific properties that allow fast evaluation and thus make exploring bigger structure spaces possible.

1.2.4 Densities and density functionals

Most prominent approaches use ML to model direct maps from atom positions to properties, e.g. potential energy surface (PES) models learn a map

$$(Z_1, R_1), \ldots, (Z_{N^{\text{atoms}}}, R_{N^{\text{atoms}}}) \mapsto E. \tag{1.22}$$

However, DFT provides us with a physically sound theory were most parts are well-known and easy to compute or good approximations exist. It is therefore desirable to use ML for the unknown or hard parts (i.e. computational bottlenecks). Approaches that follow this idea will be more complex not only conceptually but also practically for two reasons.

First, there is no training data available. Most datasets used to train ML models for quantum mechanics and quantum chemistry provide the atomic positions $R_1, \ldots, R_{N^\alpha}$ and charges $Z_1, \ldots, Z_{N^\alpha}$ plus properties of interest, e.g. atomization energy, band gap, forces, etc. If we want access to the density n or the kinetic energy T, we have to design and compute new datasets.

Secondly, if we use ML to model e.g. density functionals, we still have to find the density, for example via iterative optimization methods. We might also have to search for self-consistent solutions. This requires the ML model to be robust in the sense that it has to perform reasonably well even when slightly outside the training set region.

This area of research has been pioneered in the work of Snyder et al. [Sny+12], Snyder et al. [Sny+13b], and Snyder et al. [Sny+15]. It follows the approach of learning a model of the kinetic energy (KE) functional. An ML model for the KE functional would allow us to bypass the KS equations that use iterative optimization of the total energy functional $E[n]$. It is also a universal functional which makes it theoretically possible to transfer knowledge the model has about some system to predict on other systems, see for example [Li+16a]. However, technical difficulties (described further in Sec. 3.1) prevented progress beyond 1-D toy systems.

This thesis now deals with the extension of this approach to 3-D systems. It develops new methodology that even outperforms present approaches in 1-D. We also design datasets and run electronic structure calculations to make testing the new methodology in 3-D possible.

1.3 Organization and own contributions

The major results of this thesis have been published in

> Felix Brockherde, Leslie Vogt, Li Li, Mark E. Tuckerman, Kieron Burke, and Klaus-Robert Müller. "Bypassing the Kohn-Sham equations with machine learning". In: *Nat. Commun.* 8.1 (2017), p. 872.

Since Nature Communications is a natural science journal, the publications concentrates on the DFT and MD parts. Contributions to the publication K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K.-R. Müller, and E. K. U. Gross. "How to represent crystal structures for machine learning: Towards fast prediction of electronic properties". In: *Phys. Rev. B* 89.20 (May 2014), p. 205118 are not part of this thesis.

The thesis is roughly divided in three parts. Chapter 2 sets the machine learning foundation for the machine-learning-Hohenberg-Kohn (ML-HK) map. Chapter 3 reviews previous approaches and introduces the ML-HK map together with evaluation measures. Chapter 4 applies the new method to increasingly difficult systems: from 1-D toy systems to 3-D molecules, to molecular dynamics simulations.

The organization and each chapter's contribution is structured as follows:

Chapter 2 This chapter lays the technical foundation for the thesis. The first part covers basis representations for density functions. The second part introduces the framework of reproducing kernel Hilbert spaces (RKHSes) for functional prediction and discusses least-squares regression for functions in basis representation. The thesis contributes an elegant approach to regularize the smoothness of the machine learning model output, the density function, by using integral operator-valued kernels with a Fourier basis representation. It provides an efficient algorithm to cross-validate the hyper-parameters for functional prediction models. Finally, an experiment on toy data shows how the new method compares to baseline approaches in different settings.

Chapter 3 Earlier work on learning the non-interacting kinetic energy functional and its difficulties are presented. Then this thesis contributes the machine-learning-Hohenberg-Kohn map, a direct potential to density map, a machine learning energy functional, and the machine-learning-Kohn-Sham map, a benchmarking method. The thesis further contributes extensive error measures to allow detailed analysis of different prediction approaches.

Chapter 4 This chapter evaluates the machine learning methods and covers application specific details. The thesis contributes molecular datasets that include densities for several small molecules. It contributes normalization approaches and an efficient possibility to sample data for training from classical MD trajectories. The machine-learning-Hohenberg-Kohn (ML-HK) map derived in Chapter 3 is applied to 3-D molecules up until the simulation of molecular dynamics. The thesis contributes a method to simulate ab initio molecular

dynamics with machine learning without ever having to run ab initio MD for training data generation.

Chapter 5 This chapter concludes and provides a brief outlook.

Appendix The appendix provides a proof of concept for more advanced ideas that are touched in discussions throughout the thesis.

2

Machine Learning for Functionals and Functional Response Data

This chapter lays the foundation for and introduces the ML-HK map that is used in Chapter 4 for predicting electron densities and other observables. Sec. 2.2 reviews the theory behind reproducing kernel Hilbert spaces (RKHSes) and gives a theoretical motivation for the methodology behind the ML-HK map. Sec. 3.1 reviews the kinetic energy approach and discusses why an application to 3-D is impractical. Finally, Sec. 3.2.2 introduces the ML-HK map.

2.1 Functional data representations

We are interested in working with functional data, either as output of our models, i.e. predicting densities, or as input, i.e. learning density functionals. Functions are often described in analytical form, i.e. a coulomb potential $v : \mathbb{R}^d \to \mathbb{R}$ is given by

$$v(\boldsymbol{r}) = \sum_{\alpha=1}^{N^{\text{atoms}}} \frac{Z_\alpha}{\|\boldsymbol{r}_\alpha - \boldsymbol{r}\|}. \tag{2.1}$$

To work with functions numerically, we have to represent the analytical form in vectorial form. The simplest representation is grid based: We define a grid with grid-points $g_1, \ldots, g_G \in \mathbb{R}^d$ and define the vectorial representation of the

function v as $\widetilde{v}_i = v(g_i)$, $i = 1, \ldots, G$. A grid representation is natural and data is often measured in grid form. Problematic is the high dimensionality and over-representation.

Especially in higher dimensions, e.g. in 3-D, the number of grid points quickly exceed the limits given by storage constraints or time complexity of analysis algorithms. Functional data in grid representation is over-represented because functions are usually smooth and we can assume an inherent correlation between evaluations of neighboring grid points. Additionally when predicting functions in grid representation with machine learning even small inaccuracies in the output can lead to significant inaccuracies in derived quantities of the output functions, e.g. derivatives.

We therefore choose to represent functional data with basis functions. Basis functions are a set of functions that span a function space. We are interested in finite basis function sets so we can work with the basis function coefficients. For a function $y \in \mathcal{Y}$ and basis functions $\varphi_i, \ldots, \varphi_L$, we inherently define the basis coefficients $\widetilde{y}_1, \ldots, \widetilde{y}_L$ as

$$y(\boldsymbol{r}) = \sum_{l=1}^{L} \widetilde{y}_l \varphi_l(\boldsymbol{r}) \tag{2.2}$$

This does not always lead to well-defined basis coefficients. Depending on the set of basis functions, there might be multiple representations. For orthogonal basis systems, however, the basis coefficients are unique and thus well-defined. We say that a function y is basis representable, if $y \in \operatorname{span}\{\varphi_1, \ldots, \varphi_L\}$.

Practically, if functional data is given in grid representation and we want to transform it into basis representation, we can find basis coefficients by solving a least-squares minimization problem [RS05]

$$\min_{\widetilde{y}} \sum_{i=1}^{G} \left\| y(g_i) - \sum_{l=1}^{L} \widetilde{y}_l \varphi_l(g_i) \right\|^2 . \tag{2.3}$$

The minimization problem has a unique solution if the vectors $\Phi_l = (\varphi_l(g_1), \ldots, \varphi_l(g_G))^\top$ are orthogonal.

Since functional data is omnipresent in the sciences, many practically applica-

ble basis system are in use. Each basis system has unique properties that predetermine it for specific applications. We will now define basis function systems used throughout this thesis describe how to practically compute the basis coefficients for data given in discretized form on a grid. We define a formal grid basis, the Fourier basis, and a Kernel PCA basis. Finally, we discuss the accuracy loss that results from the use of basis functions.

2.1.1 Grid basis

We formally define a grid basis in order to make notation in later derivations applicable to functional data in grid representation. For equally spaced grid points $g_1, \ldots, g_G$ with grid spacing Δ, i.e. $\forall i = 1, \ldots, G-1$, we have $g_{i+1} - g_i = \Delta$, we can define the grid basis as

$$\varphi_l(\boldsymbol{r}) = \frac{1}{\sqrt{\Delta}} \mathbb{1}_{g_l - \frac{\Delta}{2} \leq \boldsymbol{r} < g_l + \frac{\Delta}{2}}. \tag{2.4}$$

This allows us to treat functions in grid representation and other basis representations the same throughout the theory. Since

$$\int \mathrm{d}\boldsymbol{r}\, \varphi_l(\boldsymbol{r})\varphi_l(\boldsymbol{r}) = \frac{1}{\Delta} \int \mathrm{d}\boldsymbol{r}\, \mathbb{1}_{g_l - \frac{\Delta}{2} <= \boldsymbol{r} < g_l + \frac{\Delta}{2}} = \frac{1}{\Delta}\Delta = 1 \tag{2.5}$$

and

$$\int \mathrm{d}\boldsymbol{r}\, \varphi_l(\boldsymbol{r})\varphi_m(\boldsymbol{r}) = 0 \tag{2.6}$$

the grid basis is orthonormal. The grid basis extends naturally to 3-D.

2.1.2 Fourier basis

The Fourier basis is a very popular basis system in electronic structure codes. The Fourier series decomposes any periodic function into a sum of simple oscillating functions, i.e. sines and cosines or complex exponentials. It has several key advantages.

- Periodicity. The Fourier basis is periodic and thus allows a natural treat-

ment of crystal structures. Boundary conditions are fulfilled by definition. This makes implementations easier and calculations more efficient.

- Fast Fourier transform. Transformation between basis and grid representation is easy and efficient. The fast Fourier transform (FFT) implementations are highly optimized and matured.

- Frequency order. The Fourier basis can be sorted from high-frequency to low-frequency basis functions. This makes it easy to select a truncated basis function set by setting or converging the cut-off parameter.

- Theoretical properties. The Fourier basis has several convenient theoretical properties, e.g. Parseval's theorem that directly relates the integral of a squared function to its squared Fourier transform and the convolution theorem that relates convolutions to multiplications between frequency domain and time domain.

For our application not all functions that we want to represent with a Fourier basis are periodic. However, they are defined on a bounded domain or have no support outside a bounded set, for example electronic density functions. This allows us to treat these functions as periodic where one period spans the complete space we are interested in.

The Fourier transform has real and imaginary parts. We interleave them, so we end up with a list of *real* basis coefficients. They are sorted from low to high frequency. We define the basis on a grid $0 = g_1, \ldots, g_G = 1$ as

$$\varphi_l(\boldsymbol{r}) = \begin{cases} \sqrt{2}\cos\left\{2\pi \boldsymbol{r}(l-1)/2\right\}, & l \text{ odd} \\ \sqrt{2}\sin\left\{2\pi \boldsymbol{r} l/2\right\}, & l \text{ even} \end{cases} \qquad l = 1, \ldots, L. \tag{2.7}$$

For other grids, we can stretch and translate the basis functions accordingly. The basis is orthonormal since for $m, l \in \{1, 2, \ldots\}$

$$\int_0^1 \mathrm{d}\boldsymbol{r}\, \sqrt{2}\sin(2\pi \boldsymbol{r} m)\sqrt{2}\cos(2\pi \boldsymbol{r} l) = 0 \tag{2.8}$$

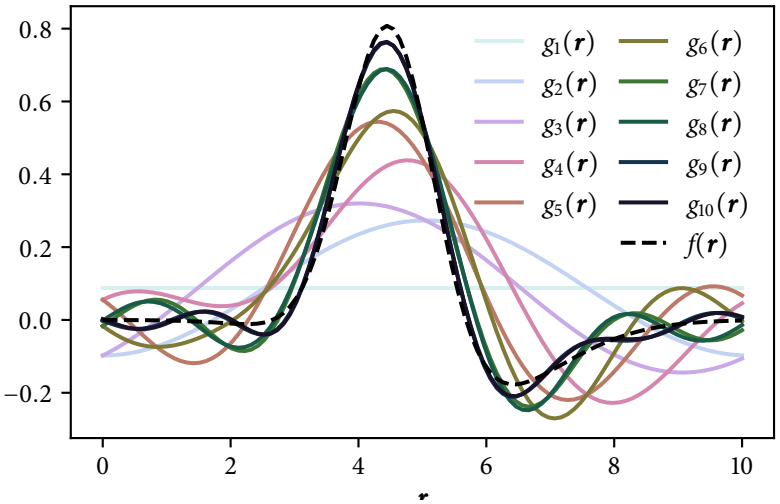


Figure 2.1: The function $f(\boldsymbol{r}) = \exp\left(-|\boldsymbol{r} - 4.5|^2\right) - 1/4 \exp\left(-1/4|\boldsymbol{r} - 5.5|^2\right)$ (dashed) should be represented by 10 Fourier basis functions. We plot the function f when represented by $m = 1, \ldots, 10$ basis functions, i.e. the truncated Fourier expansion $g_m = \tilde{f}_1\varphi_1 + \cdots + \tilde{f}_m\varphi_m$ (g_7, g_8 and g_9, g_{10} overlap).

and (with $\delta_{ml} = 1$ if $m = l$, 0 otherwise)

$$\int_0^1 \mathrm{d}\boldsymbol{r}\,\sqrt{2}\sin(2\pi\boldsymbol{r}m)\sqrt{2}\sin(2\pi\boldsymbol{r}l) = \int_0^1 \mathrm{d}\boldsymbol{r}\,\sqrt{2}\cos(2\pi\boldsymbol{r}m)\sqrt{2}\cos(2\pi\boldsymbol{r}l) = \delta_{ml}. \tag{2.9}$$

An example for how the Fourier basis fits a given function is provided in Fig. 2.1. The basis is able to fit arbitrary functions and thus comes close to the peaks. However, we can also observe the characteristical oscillations at the domain edges where the original function is nearly flat.

We can transform a function in grid representation into the Fourier basis by applying a discrete Fourier transform. The truncated discrete Fourier transform gives us accurate basis coefficients that are equal to the least-squares solution, i.e. the first N Fourier coefficients from a discrete Fourier transform are the same as the least squares fit with N basis functions [SS03].

The basis naturally extends to 3-D. We can enumerate the 3-D Fourier basis functions as

$$\varphi_{lmn}(\boldsymbol{r}) = \sqrt{2}\exp\left(2\pi\mathrm{i}(l\boldsymbol{r}_1 + m\boldsymbol{r}_2 + n\boldsymbol{r}_3)\right) \tag{2.10}$$

We could reindex the triple index imaginary coefficients to single index real coefficients and we will assume a single index for this thesis. There is, however, no need to write it out explicitly, because we do not need to use the basis functions in any implementation. If we want to implement the 3-D Fourier basis, we use the FFT, just as for the 1-D Fourier basis. For a function in grid representation, we transform along one axis first, then transform the resulting coefficients along the second axis, and so on. This gives us the 3-D discrete Fourier transform for a function $y \in \mathcal{Y}$ on a 3-D grid $g_{1,1,1}, \ldots, g_{1,1,G}, \ldots, g_{G,G,G}$:

$$\tilde{y}_{m,n,l} = \frac{1}{\sqrt{2}} \sum_{i,j,k=0}^{G-1} y(g_{ijk}) \exp\left(-2\pi\mathrm{i}(mi + nj + lk)/L\right), \quad i,j,k = 0,\ldots,G. \tag{2.11}$$

For real functions ($y(g_{ijk}) \in \mathbb{R}$), the basis coefficients $\tilde{y}_{mnl}$ show some symmetries [SS03], specifically

$$\tilde{y}_{mnl} = \tilde{y}^*_{-m,-j,-k} \quad \text{for } i,j,k \geq 1 \tag{2.12}$$

where we adopt the Fourier analysis notation of writing $G - i$ as $-i$ and the star denotes the complex conjugate. We thus truncate the basis coefficients for a fixed L to

$$\bigcup_{l,m,n \leq L} \{\tilde{y}_{l,m,n}, \tilde{y}_{-l,m,n}, \tilde{y}_{l,-m,n}, \tilde{y}_{l,m,-n}\}. \tag{2.13}$$

The 3-D Fourier basis system as defined here is also orthonormal.

2.1.3 Kernel PCA basis

Kernel principle component analysis (Kernel PCA) [SSM98a] defines a basis in kernel feature space. These are not basis functions per se, but can give a very

compact representation of functional data. The representation is given in terms of basis coefficients in the feature space.

For a dataset $x_1, \ldots, x_M$, let $\Phi : \mathcal{X} \to \mathcal{W}$ be a scalar kernel feature map and $\mathbf{K}$ be the corresponding positive definite kernel matrix $\mathbf{K}_{ij} = K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$. We assume the data points x_i are centered in the feature space $\mathcal{W}$, i.e. $\sum_{i=1}^{M} \Phi(x_i) = 0$. We find parameters p_{jl} by eigen-decomposition of the Kernel matrix $\mathbf{K}$. Let $(p_{k1}, \ldots, p_{kM})^\top$ be the k-th eigenvector corresponding to the $k - th$ largest eigenvalue λ_k. We normalize the eigenvectors p_k so that $\|p_k\|^2 = 1/\lambda_k$. The Kernel PCA basis coefficients for a data point x are given by

$$\tilde{x}_l = \langle \Phi(x), \varphi_l \rangle = \sum_{j=1}^{M} p_{lj} K(x, x_j) \tag{2.14}$$

and the basis in feature space is given by

$$\varphi_l = \sum_{j=1}^{M} p_{lj} \Phi(x_j). \tag{2.15}$$

We can show that the basis is orthonormal, since

$$\langle \varphi_l, \varphi_m \rangle = \sum_{i,j} p_{li} p_{mj} \mathbf{K}_{ij} = \langle p_l, \mathbf{K} p_m \rangle = \langle p_l, \lambda_m p_m \rangle = \lambda_m \langle p_l, p_m \rangle = \delta_{lm}. \tag{2.16}$$

As is common for kernel algorithms, the transformation $\Phi(x)$ does not have to be carried out explicitly. Problematic is then the back-projection: Given the basis coefficients $\tilde{x}$ in feature space, we know that

$$\Phi(x) = \sum_{l=1}^{L} \tilde{x}_l \varphi_l, \tag{2.17}$$

But how can we find x? The back-projection problem for Kernel PCA is not trivial but several solutions exist. Mika et al. [Mik+99] suggest a gradient descent approach. Bakır, Weston, and Schölkopf [BWS04] suggest to learn the back-projection map. This approach turns out to be very robust in practice, but has a tremendous computational overhead, because it is necessary to learn a map from $\tilde{x}$ to the high-dimensional functional objects x.

2.1.4 Atom-centered basis functions

Ab initio electronic structure calculations are often carried out using atom-centered basis functions for the KS orbitals. The orbitals of the non-interacting Schrödinger equation were given in Eq. 1.10 and the KS potential in Eq. 1.15. The non-interacting system is thus described by the Kohn-Sham equations

$$\left[T_s + v_{\mathrm{H}}(\mathbf{r}) + v_{\mathrm{XC}}(\mathbf{r}) + v(\mathbf{r})\right] \varphi_\alpha(\mathbf{r}) = \epsilon_\alpha \varphi_\alpha(\mathbf{r}). \tag{2.18}$$

These orbitals $\varphi_\alpha(\mathbf{r})$ are expanded over an atom-centered Gaussian-type orbital (GTO) basis set χ, i.e.:

$$\varphi_\alpha(\mathbf{r}) = \sum_{p=1}^{Q} C_{\alpha p} \chi_p(\mathbf{r}). \tag{2.19}$$

The basis functions χ_p depend not only on the type of atoms but also on the chemical effects one wishes to simulate. They have the format of

$$\chi(x, y, z) = x^l y^m z^n \exp\{-\zeta d^2\}, \tag{2.20}$$

where $\boldsymbol{r} = (x, y, z)^\top$ are the cartesian coordinates, d is the distance to the atomic center, and l, m, n and ζ are parameters. Quantum-chemistry codes usually implement these basis functions and make them practically usable by parametrizing them based on atom type and level of chemical accuracy.

These orbital basis functions also allow us to parametrize the density. This leads to a conceptually different approach but makes it possible to integrate the machine learning model into an electronic structure code. A proof of concept is described and discussed in Appendix A.

2.1.5 Accuracy loss in basis function representation

Although all basis function systems explored in this thesis are complete, i.e. we can represent all functions in the function space with the full basis function system and $\mathrm{span}(\varphi_1, \varphi_2, \dots) = \mathcal{Y}$, we always limit the number of basis functions L and thus introduce an error (quantified by the least-squares error in Eq. 2.3)

just by using the basis function representation.

How can we choose a reasonable number of basis functions L? L is a hyperparameter and ideally chosen like the parameters in an electronic structure calculation, often referred to as *parameter converging*. We first fix an error that we can comfortably tolerate in the output of our experiment. If we predict density functions and then predict energies with densities as features, we can fix an energy error of for example 1×10^{-2} eV. Then we increase the number of basis functions until adding new basis functions yields no improvement above the fixed energy error. This method guarantees a reasonable basis function parameters. However, we have to keep in mind that other applications that we want to use the training data and models for might have stricter requirements. For example, predicting forces as gradients of the energy might require a higher energy accuracy and thus more basis functions.

Most electronic structure codes use basis functions themselves. Here, the number of basis functions is also fixed by converging the parameter with respect to an output value like the total energy. This parameter though is often magnitudes higher than what is required for the machine learning application. This is because the electronic structure codes work numerically with the density in more complex ways whereas the machine learning application often just calculates distances between density functions. Smaller inaccuracies are treated by interpolation and regularization as noise and have little effect on the results.

A too large number of basis functions can even have negative consequences in the machine learning application. Errors in predicting coefficients of high-frequency basis functions can lead to larger inaccuracies than just setting these basis function coefficients to zero. Techniques to avoid this issue are discussed in Section 2.2.

An example of inaccuracies introduced by a basis function representation for the benzene molecule is given in Fig. 2.2. It shows that the inaccuracies are on the order of the difference resulting from PBE [PBE96] and LDA [PZ81] exchange-correlation treatment.

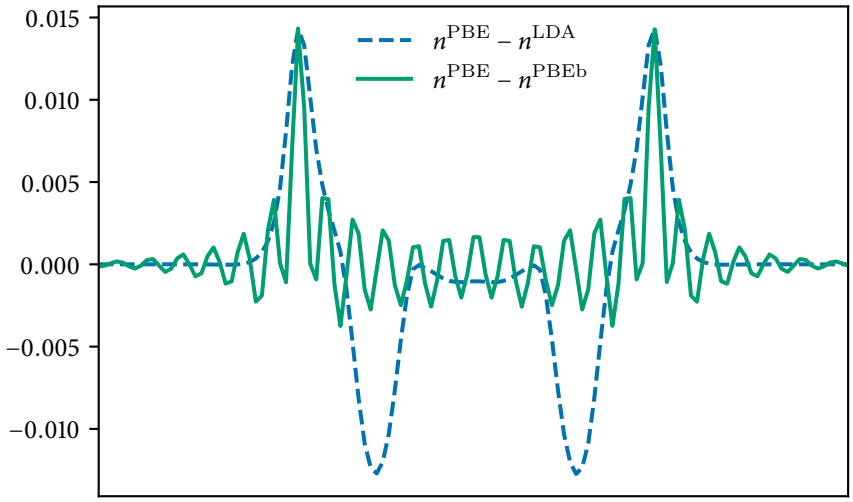


Figure 2.2: The effect of a Fourier basis representation with $L = 125000$ basis functions compared to the effect of different exchange-correlation functionals for the density of the relaxed benzene molecule. The plot shows a 1-D cut through the center of the difference between the original PBE density n^{PBE} and the information the basis representation holds n^{PBEb} (i.e. the effect of transforming into basis representation and back) in comparison to the difference between the original PBE density and the original LDA density n^{LDA}.

2.2 Machine learning for dependent data

The section introduces the RKHSes, specifically functional RKHS, and the functional least-squares regression problem for predicting functional data. We derive a numerical solution for the least-squares problem for the Gaussian kernel and contribute a solution for the integral operator-valued kernel for Fourier basis functions. After discussing efficient numerical strategies for hyper-parameter cross-validation, the section closes with toy experiments that showcase the advantages of the approach.

Sec. 2.2.1 and the beginning of Sec. 2.2.2 are based on Micchelli and Pontil [MP05]. The discussion of the functional least-squares regression problem is a generalization of Kadri et al. [Kad+16].

2.2.1 Regularized risk minimization

We start by reviewing the classical supervised learning setup and its extension to functional data. Here, we are given training samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{M}$, where the x_i are from some input space $\mathcal{X}$ and the y_i are from some output space $\mathcal{Y}$, where we also call the x_i the features of the i-th data point. Usually the output space is a set of scalar real numbers, $\mathcal{Y} \subset \mathbb{R}$. In this thesis, we focus on functions as output. In this case the $y_i \in \mathcal{L}^2(\mathbb{R}^d)$ and we call them functional responses.

For example, the features might describe a molecular geometry and the functional responses might be corresponding electron densities.

In supervised learning we try to find a hypothesis $f \in \mathcal{F}$ that provides a map between input and output space and a prediction $f(x)$ that is close to the output values y for previously unseen data that is similar to the training data. We also say, in statistical terms, that the unseen data has to be independent and identically distributed to the training data. The distance between the prediction $f(x)$ to the label y is measured by a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. For functional data, we prefer to measure the difference between $f(x)$ and y by the natural norm on $\mathcal{L}^2$,

$$l : \mathcal{L}^2(\mathbb{R}^d) \times \mathcal{L}^2(\mathbb{R}^d) \to \mathbb{R}; \quad l(f(x), y) = \int \mathrm{d}^d \boldsymbol{r} \, |(f(x))(\boldsymbol{r}) - y(\boldsymbol{r})|^2. \tag{2.21}$$

A popular approach to supervised learning is the regularized risk minimization [Vap98] framework. We minimize

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{M} l(f(x_i), y_i) + \lambda \Omega(f). \tag{2.22}$$

The first part is the empirical risk of the hypothesis f. In empirical risk minimization it is usually divided by M, but we leave this out for convenience in the derivations later on.

The second part is a regularizer $\Omega : \mathcal{L}^2(\mathbb{R}^d) \to \mathbb{R}$, and a $\lambda \in \mathbb{R}$ is the scalar regularization parameter. The regularizer, that is usually chosen to be $\Omega(f) = \|f\|_{\mathcal{F}}^2$, has two functions. First, the solution of the minimization problem might be ill-defined without it, and secondly, it penalizes the complexity of the hypothesis

f. Thus the regularization parameter allows us to control a trade-off between empirical loss on the training set and model complexity.

For functional data we are interested in the regularized least-squares regression problem

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{M} \|f(x_i) - y_i\|_{\mathcal{Y}}^2 + \lambda \|f\|_{\mathcal{F}}^2. \tag{2.23}$$

The solution depends on the Hilbert space $\mathcal{F}$ and its inner product. In the following we restrict the solution to a special kind of Hilbert spaces, the RKHSes. Its kernel will determine the distance measure between the features and thus allow solutions that are non-linear in $\mathcal{X}$. A representer theorem guarantees that the functional regression problem admits a convenient solution whose parameters can be found analytically.

We assume that the functional responses y_i are given in a basis representation, i.e.

$$y_i(\boldsymbol{r}) = \sum_{j=1}^{G} \gamma_{ij} \varphi_j(\boldsymbol{r}) \tag{2.24}$$

with basis functions $\varphi_j : \mathbb{R}^d \to \mathbb{R}$ and basis coefficients $\gamma_{ij} \in \mathbb{R}$. The output space is then all Fourier representable functions with G basis functions, i.e. the linear span of the basis functions,

$$\mathcal{Y} = \operatorname{span}\left(\{\varphi_j\}_{j=1}^{G}\right). \tag{2.25}$$

2.2.2 Reproducing kernel Hilbert space

We start out by defining a RKHS and its unique reproducing kernel. A RKHS is usually defined in its weakest form, only requiring that the evaluation functional is a bounded operator. It is then possible to show the existence of a reproducing kernel via the Fréchet-Riesz representation theorem and that the kernel uniquely determines the RKHS (up to isometry). For an overview of this approach, see for example Smola, Schölkopf, and Müller [SSM98b] or Micchelli and Pontil [MP05].

Here, we give more intuitive definitions. We first define the kernel, then define the RKHS based on this kernel. It is then possible to show that the RKHS determines the kernel. For more details on this approach, see Kadri et al. [Kad+16].

Let $\mathcal{Y}$ be a (separable) Hilbert space of functions, for example $L^2(\mathbb{R}^d)$. We can now define an (operator-valued) kernel, i.e. a function on $\mathcal{X}^2$ that maps to $\mathcal{L}(\mathcal{Y})$, the space of operators on $\mathcal{Y}$. Just like a scalar-valued kernel, it should have equivalent symmetric and positive definite properties.

Definition. An operator-valued kernel is a function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$ that is

(i) hermitian, i.e. $K(w, z) = K(z, w)^*$ for $w, z \in \mathcal{X}$ and

(ii) positive-definite, i.e. for $m \in \mathbb{N}$, $\{x_i\}_{i=1}^m \subseteq \mathcal{X}$, $\{y_i\}_{i=1}^m \subseteq \mathcal{Y}$,

$$\sum_{i,j=1}^{m} \langle K(x_i, x_j) y_i, y_j \rangle_{\mathcal{Y}} \geq 0. \tag{2.26}$$

Note that this definition generalizes the definition of a scalar-valued kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

Definition. K is a reproducing operator-valued kernel of a Hilbert space $\mathcal{F}$ if every function in $\mathcal{F}$ can be represented by taking an inner product with a kernel function, i.e. for all $f \in \mathcal{F}$, $x \in \mathcal{X}$, $f(x) = \langle K(x, \cdot), f(\cdot) \rangle$.

We can now define a (functional) RKHS. It is a Hilbert space with an associated reproducing kernel.

Definition. Let $\mathcal{F}$ be a Hilbert space of functions from $\mathcal{X}$ to $\mathcal{Y}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$. Then we say $\mathcal{F}$ is a reproducing kernel Hilbert space if there is reproducing kernel $K_{\mathcal{F}} : \mathcal{X} \times \mathcal{X} \to \mathcal{Y}$ so that

(i) for all $z, w \in \mathcal{X}$, $g \in \mathcal{Y}$, the function $z \mapsto K_{\mathcal{F}}(z, w)g$ is in $\mathcal{F}$ and

(ii) the kernel has the reproducing property, i.e. for every $f \in \mathcal{F}$, $w \in \mathcal{X}$, $g \in \mathcal{Y}$,

$$\langle f(w), g \rangle_{\mathcal{Y}} = \langle f, K_{\mathcal{F}}(w, \cdot)g \rangle_{\mathcal{F}}. \tag{2.27}$$

2.2.3 Kernels

We now define some operator valued kernels. First, we take a look at scalar-valued kernels and later describe how to adapt them to operator-valued kernels.

Popular scalar-valued kernels are the linear kernel

$$G(x, x') = \langle x, x' \rangle_{\mathcal{X}} \tag{2.28}$$

or the polynomial kernel of degree d

$$G(x, x') = \left(\langle x, x' \rangle_{\mathcal{X}} + c \right)^d. \tag{2.29}$$

The hyper-parameter $c \geq 0$ controls the trade-off between higher-order and lower-order terms in the polynomial. The most popular kernel, however, is the Gaussian kernel.

Definition. The Gaussian kernel is defined as

$$G : \mathcal{X} \times \mathcal{X} \to \mathbb{R}; \quad G(x, x') = \exp\left(-\sigma \left\| x - x' \right\|_{\mathcal{X}}^2 \right) \tag{2.30}$$

for a hyper-parameter $\sigma > 0$.

Its versatility makes it ubiquitous in kernel learning approaches. The hyper-parameter σ has significant influence on the resulting model and needs careful cross-validation. Another kernel that has proven to yield competitive results in quantum-chemical applications is the Laplacian kernel

$$G(x, x') = \exp\left(-\sigma \left\| x - x' \right\|_{\mathcal{X}} \right). \tag{2.31}$$

The hyper-parameter σ follows the same function as for the Gaussian kernel. Its norm inside the exponential is not squared, thus leading to a stronger penalty on high-frequency model functions which gives good results in low-noise settings.

We now define a way to adapt scalar-valued kernels into operator-valued kernels by multiplying with a positive-definite operator. We can show that kernels that are constructed in this way are positive-definite and then give two examples.

Definition. The adapted operator-valued kernel is a multiplicative adaption of a scalar-valued kernel to an operator valued-kernel. It is defined as

$$K_{\mathcal{F}} : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y}); \tag{2.32}$$

$$(K_{\mathcal{F}}(x_i, x_j)y)(\boldsymbol{r}) = G(x_i, x_j)(T(y))(\boldsymbol{r}) \tag{2.33}$$

where $G : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a scalar-valued kernel and $T : \mathcal{Y} \to \mathcal{Y}$ is a positive-definite operator.

Since G is symmetric as a scalar-valued kernel and T is self-adjoined as a positive-definite operator, the kernel $K_{\mathcal{F}}$ must be hermitian. When we fix $\mathcal{Y} \subseteq L^2(\mathbb{R}^d)$, we know the inner product and can write

$$\sum_{i,j} \langle K_{\mathcal{F}}(x_i, x_j)y_i, y_j \rangle_{\mathcal{Y}} \tag{2.34}$$

$$= \sum_{i,j} \langle G(x_i, x_j)T(y_i), y_j \rangle_{\mathcal{Y}} \tag{2.35}$$

$$= \int \mathrm{d}^d r \; G(x_i, x_j) \, (T(y_i))(\boldsymbol{r}) \, y_j(\boldsymbol{r}) \tag{2.36}$$

$$= \int \mathrm{d}^d r \sum_{i,j} G(x_i, x_j) \, (T(y_i))(\boldsymbol{r}) \, y_j(\boldsymbol{r}) \tag{2.37}$$

The sum is non-negative for each $\boldsymbol{r}$ since G is positive definite as a scalar-valued kernel and thus $K_{\mathcal{F}}$ is positive definite and an operator-valued kernel.

Definition. The trivial operator-valued kernel is an adapted operator-valued kernel based on the identity operator, i.e. $(T(y))(\boldsymbol{r}) = y(\boldsymbol{r})$. We can write it as

$$K_{\mathcal{F}} : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y}); \tag{2.38}$$

$$(K_{\mathcal{F}}(x_i, x_j)y)(\boldsymbol{r}) = G(x_i, x_j)y(\boldsymbol{r}) \tag{2.39}$$

where $G : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a scalar-valued kernel.

The trivial operator-valued kernel allows us to recover naive approaches to learning kernel models with functional responses later on. We now define an operator valued kernel that was popularized by the functional data analysis (FDA) community.

Definition. The integral operator-valued kernel is an adapted operator-valued kernel based on the positive-definite integral operator, i.e. $(T(y))(\boldsymbol{r}) = \int \mathrm{d}^d r'\, m(\boldsymbol{r}-\boldsymbol{r}')y(\boldsymbol{r})$ where $m(\boldsymbol{r}-\boldsymbol{r}') = \exp\{-|\boldsymbol{r}-\boldsymbol{r}'|\}$ links the functional responses in the $\mathbb{R}^d$ space. The kernel is defined as

$$K_{\mathcal{F}} : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y}); \tag{2.40}$$

$$(K_{\mathcal{F}}(x_i, x_j)y)(\boldsymbol{r}) = G(x_i, x_j) \int \mathrm{d}^d r' \exp\{-|\boldsymbol{r}-\boldsymbol{r}'|\}y(\boldsymbol{r}') \tag{2.41}$$

where, again, $G : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a scalar-valued kernel.

Numerical solutions to functional least-squares regression with these two kernels and intuitive interpretations are derived in Secs. 2.2.4 and 2.2.5. Sec. 2.3 compares the two kernels experimentally.

2.2.4 Functional least-squares regression

We now want to solve the regularized empirical risk minimization problem, that was introduced in Sec. 2.2.1 for the two kernels defined above. The problem is given by

$$\min_{f\in\mathcal{F}} \sum_{i=1}^{M} \|y_i - f(x_i)\|_{\mathcal{Y}}^2 + \lambda \|f\|_{\mathcal{F}}^2 \tag{2.42}$$

where $\mathcal{F}$ is the RKHS associated with the operator-valued kernel $K_{\mathcal{F}}$.

We first state an analog of the representer theorem for functional data in basis representation that will allow us to solve the regularized least-squares problem for functional data. Then we derive the numerical solution for the trivial operator-valued kernel and arbitrary basis functions.

Theorem. *Let $\mathcal{F}$ be a functional RKHS with kernel $K_{\mathcal{F}}$. Consider the regularized least-squares minimization problem*

$$\min_{f\in\mathcal{F}} \sum_{i=1}^{M} \|y_i - f(x_i)\|_{\mathcal{Y}}^2 + \lambda \|f\|_{\mathcal{F}}^2. \tag{2.43}$$

Then the solution $f^* \in \mathcal{F}$ has the representation

$$f^*(x) = \sum_{i=1}^{M} K_{\mathcal{F}}(x_i, x)\beta_i \tag{2.44}$$

with $\beta_i \in \mathcal{Y}$.

A detailed proof, even for general function spaces, is given in Kadri et al. [Kad+16]. The idea is setting the analogue of the directional derivative (the Gâteaux derivative) of the objective in Eq. 2.43 to zero.

Using the representer theorem, we can replace the minimization over the RKHS by a minimization over weights $\beta \in \mathcal{Y}$:

$$\min_{\beta} \sum_{i=1}^{M} \left\| y_i - \sum_{j=1}^{M} K_{\mathcal{F}}(x_i, x_j)\beta_j \right\|_{\mathcal{Y}}^2 + \lambda \left\| \sum_{i=1}^{M} K_{\mathcal{F}}(\cdot, x_i)\beta_i \right\|_{\mathcal{F}}^2 \tag{2.45}$$

If we expand the norm, we can use the reproducing property and

$$\left\| \sum_{i=1}^{M} K_{\mathcal{F}}(\cdot, x_i)\beta_i \right\|_{\mathcal{F}}^2 \tag{2.46}$$

$$= \left\langle \sum_{i=1}^{M} K_{\mathcal{F}}(\cdot, x_i)\beta_i, \sum_{j=1}^{M} K_{\mathcal{F}}(\cdot, x_j)\beta_j \right\rangle_{\mathcal{Y}} \tag{2.47}$$

$$= \sum_{i,j=1}^{M} \langle K_{\mathcal{F}}(\cdot, x_i)\beta_i, K_{\mathcal{F}}(\cdot, x_j)\beta_j \rangle_{\mathcal{Y}} \tag{2.48}$$

$$= \sum_{i,j=1}^{M} \langle K_{\mathcal{F}}(x_i, x_j)\beta_i, \beta_j \rangle_{\mathcal{Y}}. \tag{2.49}$$

This yields

$$\min_{\beta} \sum_{i=1}^{M} \left\| y_i - \sum_{j=1}^{M} K_{\mathcal{F}}(x_i, x_j)\beta_j \right\|_{\mathcal{Y}}^2 + \lambda \sum_{i,j=1}^{M} \langle K_{\mathcal{F}}(x_i, x_j)\beta_i, \beta_j \rangle_{\mathcal{Y}}. \tag{2.50}$$

First, we solve this optimization problem for the trivial operator-valued kernel, i.e.

$$(K_{\mathcal{F}}(x_i, x_j)y)(\boldsymbol{r}) = G(x_i, x_j)y(\boldsymbol{r}). \tag{2.51}$$

Let $\mathbf{G}$ be the scalar-valued kernel matrix, i.e. $\mathbf{G}_{ij} = G(x_i, x_j)$.

Since we assume that the output space consists of Fourier representable functions, we can write y and β in basis representation. Let γ be the basis coefficients for y, and b be the basis coefficients for β, i.e.

$$y_i = \sum_{j=1}^{L} \gamma_{ij}\varphi_j \qquad \text{and} \qquad \beta_i = \sum_{j=1}^{L} b_{ij}\varphi_j. \tag{2.52}$$

Then, the optimization problem is given by

$$\min_b \sum_{i=1}^{M} \left\| \sum_{l=1}^{L} \gamma_{il}\varphi_l - \sum_{j=1}^{M} \mathbf{G}_{ij} \sum_{l=1}^{L} b_{jl}\varphi_l \right\|_{\mathcal{Y}}^2 \tag{2.53}$$

$$+ \lambda \sum_{i,j=1}^{M} \langle \mathbf{G}_{ij} \sum_{l=1}^{L} b_{il}\varphi_l, \sum_{l=1}^{L} b_{jl}\varphi_l \rangle_{\mathcal{Y}}. \tag{2.54}$$

Expanding the norm, using the linearity of the inner product, and reordering of sums yields

$$\min_b \sum_{l,m}^{L} \Bigg[\sum_{i=1}^{M} \gamma_{il}\gamma_{im} - 2 \sum_{i,j=1}^{M} \mathbf{G}_{ij}\gamma_{il}b_{jm} + \sum_{i,j,k=1}^{M} \mathbf{G}_{ij}\mathbf{G}_{ik}b_{jl}b_{km} \tag{2.55}$$

$$+ \lambda \sum_{i,j=1}^{M} \mathbf{G}_{ij}b_{il}b_{jm} \Bigg] \langle \varphi_l, \varphi_m \rangle_{\mathcal{Y}}. \tag{2.56}$$

This can be simplified substantially if we assume that the basis functions are orthonormal. This is the case for the Fourier basis and the grid basis which allows us to treat functions given on a grid. The optimization problem reduces to

$$\min_b \sum_{l=1}^{L} \left[\sum_{i=1}^{M} \gamma_{il}^2 - 2 \sum_{i,j=1}^{M} \mathbf{G}_{ij}\gamma_{il}b_{jl} + \sum_{i,j,k=1}^{M} \mathbf{G}_{ij}\mathbf{G}_{ik}b_{jl}b_{kl} + \lambda \sum_{i,j=1}^{M} \mathbf{G}_{ij}b_{il}b_{jl} \right], \tag{2.57}$$

a multivariate regression problem of b on $\mathbf{G}$, i.e. we can solve for each basis function φ_l independently. Moreover the problem is independent of the actual basis function, relevant are only the basis coefficients. To derive the solution of each individual φ_l, we drop the l indices and use $b = (b_1, \ldots, b_M)^\top$ and $\gamma =$

$(\gamma_1, \ldots, \gamma_M)^\top$ as vectors. In matrix form, we have to solve

$$\min_b \|\gamma - \mathbf{G}b\|^2 + \lambda b^\top \mathbf{G} b. \tag{2.58}$$

This is just regular kernel ridge regression (KRR). The solution is well known: We set the derivative to zero,

$$-2\mathbf{G}^\top \gamma + 2\mathbf{G}^\top \mathbf{G} b + 2\lambda \mathbf{G} b = 0, \tag{2.59}$$

and arrive at the well-known analytical solution

$$b = (\mathbf{G} + \lambda \mathbf{I})^{-1} \gamma. \tag{2.60}$$

We thus are able to reduce the problem that arises from a function-valued treatment of our data to a multivariate vector-valued problem. For grid-based data and orthonormal basis functions, we can use the trivial operator-valued kernel based on the identity operator to recover the naive solutions of learning each grid point or each basis function coefficient independently. Indeed, since the evaluation is given by

$$f(x) = \sum_{i=1}^{M} K_{\mathcal{F}}(x_i, x)\beta_i = \sum_{i=1}^{M} G(x_i, x) \sum_{l=1}^{L} b_{il}\varphi_l, \tag{2.61}$$

the solution for the l-th basis function coefficient of $f(x) \in \mathcal{Y}$ is given by

$$\sum_{i=1}^{M} G(x_i, x) b_{il}. \tag{2.62}$$

2.2.5 Integral operator-valued kernel

We now derive the numerical solution of the functional least-squares regression problem for the integral operator-valued kernel. The derivation assumes a Fourier basis function system. Without assuming a specific basis function system it is not possible to calculate the integrals and arrive at a numerical solution.

The integral operator-valued kernel

$$K_{\mathcal{F}} : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y}); \tag{2.63}$$

$$(K_{\mathcal{F}}(x_i, x_j)y)(\boldsymbol{r}) = G(x_i, x_j) \int \mathrm{d}^d \boldsymbol{r}'\, m(\boldsymbol{r} - \boldsymbol{r}')y(\boldsymbol{r}') \tag{2.64}$$

creates a dependence between the output evaluations $y(\boldsymbol{r})$ in $\mathbb{R}^d$ space. We usually set

$$m(\boldsymbol{r} - \boldsymbol{r}') = \exp\{-\nu|\boldsymbol{r} - \boldsymbol{r}'|\} \tag{2.65}$$

and thus create a distance measure in feature space that isolates functions that are not smooth. The effect can be controlled with the weight parameter ν inside the exponential, just as for the Gaussian kernel. We can remove all independence by setting

$$m(\boldsymbol{r} - \boldsymbol{r}') = \delta_{\boldsymbol{r}-\boldsymbol{r}'}. \tag{2.66}$$

This recovers the trivial operator-valued kernel and thus the naive approach to functional response learning as discussed above.

We can now find an elegant solution for the regularized least-squares regression problem Eq. 2.50 for the integral operator-valued kernel and Fourier basis functions. Let us introduce a shorthand for the integral operator applied to a basis function

$$\mu_l(\boldsymbol{r}) = \int \mathrm{d}^n r'\, m(\boldsymbol{r} - \boldsymbol{r}')\varphi_l(\boldsymbol{r}'). \tag{2.67}$$

Since we would like to eliminate the orthonormal basis functions from our optimization problem later on, it is useful to write μ_l in its basis representation as well. We can implicitly define the basis coefficients $\tilde{\mu}$ and $\tilde{m}$ as

$$\mu_l(\boldsymbol{r}) = \sum_{p=1}^{L} \tilde{\mu}_{lp}\varphi_p(\boldsymbol{r}) \quad \text{and} \quad m(\boldsymbol{r}) = \sum_{p=1}^{L} \tilde{m}_p\varphi_p(\boldsymbol{r}). \tag{2.68}$$

We can observe that μ_l is a convolution with a basis function that admits a very simple Fourier basis representation. A convolution in the real domain is a mul-

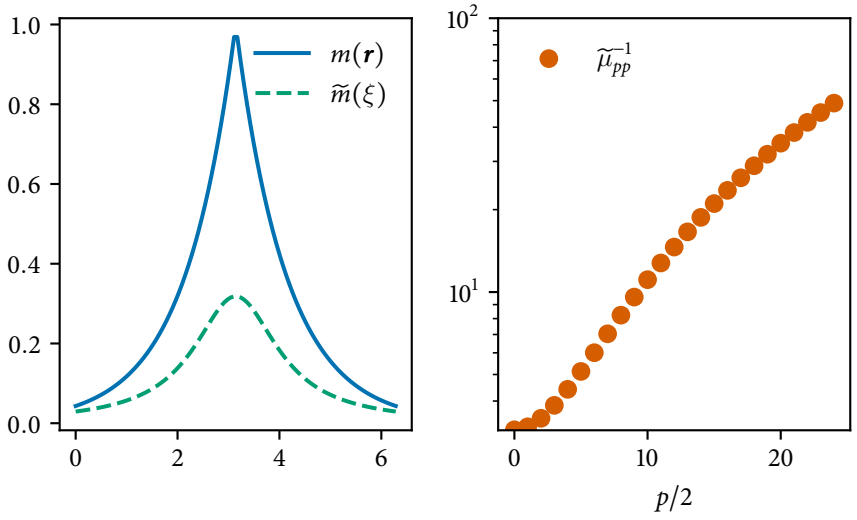


Figure 2.3: (Left) The exponential function $m(\boldsymbol{r})$ for $\nu = 1$ and its Fourier transform $\widetilde{m}(\xi)$. (Right) The regularization multiplier $\widetilde{\mu}^{-1}$ for an example Fourier basis.

tiplication in the Fourier domain.

The exponential function m has Fourier transform

$$\widetilde{m}(\xi) = \frac{1}{\pi} \frac{\nu}{\xi^2 + \nu^2}, \tag{2.69}$$

a Lorentzian function. Its constant scaling factor depends on the normalization of the Fourier basis. Note that m might not be Fourier representable with G basis functions, i.e. we can not write m as written in Eq. 2.68. We ignore this for now and give a justification later. Its basis coefficients are then (according to the definition of the Fourier basis in Section 2.1.2) given by

$$\widetilde{m}_p = \widetilde{m}\left(\left\lfloor \frac{p}{2} \right\rfloor\right) = \frac{1}{\pi\sqrt{2}} \frac{\nu}{\left\lfloor \frac{p}{2} \right\rfloor^2 + \nu^2}. \tag{2.70}$$

The function $m(\boldsymbol{r})$ and its Fourier transform $\widetilde{m}(\xi)$ are visualized in Fig. 2.3.

The second part, φ_l, is already a Fourier basis function, so its l-th basis coeffi-

cient is 1 and the rest are zero. This gives us the basis coefficients of μ as

$$\tilde{\mu}_{lp} = \begin{cases} \frac{1}{\pi\sqrt{2}} \frac{\nu}{\left[\frac{p}{2}\right]^2 + \nu^2}, & \text{if} \quad l = p \\ 0, & \text{else.} \end{cases} \tag{2.71}$$

After this preparation, we can start solving the optimization problem. Using the new terminology and the basis coefficients γ and b for y and β we can write it, after reordering the basis coefficient sum, as

$$\min_b \sum_{i=1}^{M} \left\| \sum_{l=1}^{L} \gamma_{il}\varphi_l - \sum_{j=1}^{M} G_{ij} \sum_{l=1}^{M} b_{jl}\mu_l \right\|_{\mathcal{Y}}^2 + \lambda \sum_{ij} \left\langle G_{ij} \sum_{l=1}^{L} b_{il}\mu_l, \sum_{l=1}^{L} b_{jl}\varphi_l \right\rangle_{\mathcal{Y}}. \tag{2.72}$$

After replacing μ with its basis representation and separating the sums over the basis functions, we get

$$\min_b \sum_{p,q=1}^{L} \Bigg[\sum_{i=1}^{L} \Bigg(\gamma_{ip}\gamma_{iq} - 2 \sum_{j=1}^{L} \sum_{l=1}^{M} \gamma_{iq} G_{ij} b_{jl} \tilde{\mu}_{lp} \tag{2.73}$$

$$+ \sum_{j,k=1}^{M} \sum_{l,m=1}^{G} G_{ij} G_{ik} b_{jl} b_{km} \tilde{\mu}_{lp} \tilde{\mu}_{mq} \Bigg) \tag{2.74}$$

$$+ \lambda \sum_{i,j=1}^{M} \sum_{l=1}^{L} G_{ij} b_{il} b_{jq} \tilde{\mu}_{lp} \Bigg] \langle \varphi_p, \varphi_q \rangle_{\mathcal{Y}}. \tag{2.75}$$

Since the inner product is zero for $p \neq q$ and $\tilde{\mu}_{lp}$ is zero for $l \neq p$, the objective gets significantly shorter. Note that this will be the case for all linking functions $m(\boldsymbol{r} - \boldsymbol{r}')$ that are even. This is due to the convolution theorem of the Fourier basis. We have

$$\min_b \sum_{p=1}^{L} \sum_{i=1}^{M} \Bigg[\gamma_{ip}^2 - 2 \sum_{j=1}^{M} \gamma_{ip} G_{ij} b_{jp} \tilde{\mu}_{pp} + \tag{2.76}$$

$$\sum_{j,k=1}^{M} G_{ij} G_{ik} b_{jp} b_{kp} \tilde{\mu}_{pp}^2 \Bigg] + \lambda \sum_{i,j=1}^{M} G_{ij} b_{ip} b_{jp} \tilde{\mu}_{pp}. \tag{2.77}$$

This objective is independent for each p again. This also means that additional basis functions, in case m is not Fourier representable with G basis functions,

become irrelevant for the solution and we can safely ignore them. Dropping the index, we can write the p independent objectives as

$$\min_b \|\gamma - (\tilde{\mu}G)b\|^2 + \lambda b^\top (\tilde{\mu}G)b. \tag{2.78}$$

The solution is already given in Eq. 2.60 and only has a modified kernel here as in

$$b = (\tilde{\mu}G - \lambda\mathbf{I})^{-1}\gamma \tag{2.79}$$

For further analysis, we divide by $\tilde{\mu}$,

$$b = \frac{1}{\tilde{\mu}} \underbrace{\left(G - \frac{\lambda}{\tilde{\mu}}\mathbf{I}\right)^{-1}\gamma}_{=b'}, \tag{2.80}$$

and define a b' as the solution to a trivial operator-valued kernel problem that has a modified regularization parameter $\lambda/\tilde{\mu}$.

Let us now analyze the evaluation of the model function f. With $x \in \mathcal{X}$, the evaluation $f(x) \in \mathcal{Y}$ is given by

$$f(x)(\boldsymbol{r}) = \sum_{i=1}^{M} K_{\mathcal{F}}(x_i, x)\beta_i(\boldsymbol{r}) \tag{2.81}$$

$$= \sum_{i=1}^{M} G(x_i, x) \int \mathrm{d}^d\boldsymbol{r}'\, m(\boldsymbol{r} - \boldsymbol{r}')\beta_i(\boldsymbol{r}') \tag{2.82}$$

$$= \sum_{i=1}^{M} G(x_i, x) \sum_{l=1}^{L} b_{il} \int \mathrm{d}^d\boldsymbol{r}'\, m(\boldsymbol{r} - \boldsymbol{r}')\varphi_l(\boldsymbol{r}') \tag{2.83}$$

and its Fourier basis coefficients by

$$\widetilde{f(x)}_p = \sum_{i=1}^{M} G(x_i, x) \sum_{l=1}^{L} b_{il} \tilde{\mu}_{lp} \tag{2.84}$$

$$= \sum_{i=1}^{M} G(x_i, x) b_{ip} \tilde{\mu}_{pp} \qquad (\tilde{\mu}_{lp} = 0 \text{ for } l \neq p) \tag{2.85}$$

$$= \sum_{i=1}^{M} G(x_i, x) b'_{ip} \tag{2.86}$$

Thus we recover also the evaluation of the trivial operator-valued kernel model for the problem with modified regularization parameter

$$\frac{\lambda}{\tilde{\mu}_{pp}} = \frac{\lambda \pi \sqrt{2}}{\nu} \left(\left\lfloor \frac{p}{2} \right\rfloor^2 + \nu^2 \right). \tag{2.87}$$

A visualization of the modification $\widetilde{\mu}_{pp}^{-1}$ obtained in an example setting is given in Fig. 2.3.

2.2.6 Numerical solution and hyper-parameter cross-validation

We now discuss ways to solve the problem (Eqs. 2.60 and 2.80) numerically. To ease notation, we concentrate on solving

$$b = (\mathbf{G} + \lambda \mathbf{I})^{-1} \gamma. \tag{2.88}$$

and comment on the effects of a modified regularization parameter $\lambda / \tilde{\mu}$ later.

As a kernel matrix, $\mathbf{G}$ is positive semidefinite. And since $\lambda \mathbf{I}$ is positive definite for $\lambda > 0$, $\mathbf{G} + \lambda \mathbf{I}$ is positive definite as well. Although this is only a theoretical guarantee, we do not need to worry much about numerical inaccuracies. We can always enforce a positive definite matrix by regularizing a bit more.

We can avoid an expensive and unstable inversion of $\mathbf{G} + \lambda \mathbf{I}$ by viewing the problem as a system of linear equations. These can be solved by LU decomposition [Sch95] and subsequent forward and backward substitution. Here though, since $\mathbf{G} + \lambda \mathbf{I}$ is guaranteed to be positive definite, we can almost halve the computation time required by using a Cholesky decomposition instead [Pre+92].

For functional data, we have to solve many problems of Eq. 2.88 (one for each basis function coefficient) with γ being the only changing variable. Thus, we can do the Cholesky decomposition once and just do many forward and backward substitutions.

The numerical inversion offers an interesting alternative. We have to invert once and then only do many vector multiplications (compared to many forward and backward substitutions). Leaving stability concerns aside, there is a point when the inversion becomes computationally cheaper overall than the Cholesky decomposition approach (when the number of basis functions $L \gg M$, the number of data points). However since the numerical instability of the matrix inversion can lead to significant performance losses for the regression models, we avoid this approach and only use the Cholesky decomposition.

At the beginning of this chapter we assumed that the functional responses are given in basis representation. We saw that we can fit the model based on basis coefficients alone. For the cross-validation of hyper-parameters we are now again interested in optimizing the loss in $\|\cdot\|_{\mathcal{Y}}$, but would like to keep working with basis coefficients. Fortunately, we can use the same approach as above and take advantage of orthonormal basis functions. For $a, b \in \mathcal{Y}$ with basis representations $\widetilde{a}, \widetilde{b}$, we have

$$\|a - b\|_{\mathcal{Y}}^2 = \left(\sum_{l,m=1}^{L} \widetilde{a}_l \widetilde{a}_m - 2\widetilde{a}_l \widetilde{b}_m + \widetilde{b}_l \widetilde{b}_m \right) \langle \varphi_l, \varphi_m \rangle_{\mathcal{Y}} \tag{2.89}$$

$$= \left\| \widetilde{a} - \widetilde{b} \right\|_2^2 \tag{2.90}$$

and we can cross-validate the mean squared error (MSE) of the basis coefficients.

The complete approach, for one cross-validation fold and the Gaussian kernel function (Eq. 2.30), is summarized in Algorithm 1.

Note that we have to run only one Cholesky decomposition per hyper-parameter combination. For the integral-operator valued kernel, we have different regularization parameters for each basis coefficient and thus need to run Cholesky decompositions not only for each hyper-parameter combination, but also for each basis coefficient. This is computationally much more complex,

however it scales nicely to high-performance-clusters because the loops are parallelizable.

Kernel-ridge-regression allows efficient leave-one-out cross-validation by computing an eigen-decomposition of the kernel matrix. Changing the regularization then only requires inverting a diagonal matrix. This efficiency translates to the integral-operator valued kernel approach with Fourier basis functions.

Data: Features $X = (x_1, \ldots, x_M)$, Labels $\Gamma = (\gamma_1, \ldots, \gamma_M)$, Kernel parameters $\Sigma \subset \mathbb{R}$, Regularization parameters $\Lambda \subset \mathbb{R}$
Result: Loss for each combination of kernel parameter, regularization parameter, and basis coefficient $\mathrm{err}_{\sigma,\lambda,l}$

$D \leftarrow$ pairwise distance matrix of X
foreach $\sigma \in \Sigma$ **do**
  $G_\sigma = \exp\left(-\sigma D^2\right)$
  foreach $\lambda \in \Lambda$ **do**
    $C \leftarrow$ Lower triangular, s.t. $CC^\top = G_\sigma + \lambda I$ (Cholesky decomposition)
    for $l \leftarrow 1$ **to** L **do**
      $S_l \leftarrow C^{-1}(\gamma_{1l}, \ldots, \gamma_{Ml})^\top$ (via forward substitution)
      $b_{\sigma,\lambda,l} \leftarrow C^{\top -1} S_l$ (via backward substitution)
    end
    $\mathrm{err}_{\sigma,\lambda,l} = \left\| G_\sigma b_{\sigma,\lambda,l} - (\gamma_{1l}, \ldots, \gamma_{Ml})^\top \right\|_2^2$
  end
end

Algorithm 1: Efficient cross-validation fold for functional response learning. Using the integral-operator valued kernel requires moving the Cholesky decomposition into the l-loop.

2.3 Toy experiment

To demonstrate the integral operator-valued kernel approach with Fourier basis functions, we create a toy dataset.

2.3.1 Dataset

We create $M = 100$ functional responses as sums of sine functions that are given by

$$y_i'(\boldsymbol{r}) = a_i \sin(\boldsymbol{r} - b_i) + c_i \sin(2\boldsymbol{r}). \tag{2.91}$$

We assume that the y_i' are given on an equi-distant grid $g_1, \ldots, g_G$ with endpoints $g_1 = 0$ and $g_G = 2\pi$ and $G = 100$ grid points.

We add random independently and identically distributed (iid) noise to the functional responses. We will try to learn from the noisy examples but evaluate on the functional responses without noise, so we set

$$y_{ij} = y_i'(g_j) + \epsilon_{ij} \tag{2.92}$$

and then use a FFT to find $G = 50$ basis functions $\widetilde{y}_p$, $1 \leq p \leq G$. The noise ϵ_i and the parameters a_i, b_i, c_i are drawn normally distributed, i.e.

$$\epsilon_{ij} \sim \mathcal{N}(0, \tau) \quad \text{and} \tag{2.93}$$

$$a_i \sim \mathcal{N}(1, 0.2), \quad b_i \sim \mathcal{N}(0, 0.2), \quad c_i \sim \mathcal{N}(1, 0.1). \tag{2.94}$$

We run experiments with different noise levels $\tau \in \{0.01, 0.05, 0.1, 0.2, 0.3\}$. Some example responses y_i are given in Fig. 2.4. For the features, we simply choose the model parameters, i.e.

$$x_i = (a_i, b_i, c_i)^\top. \tag{2.95}$$

After predicting, we transform the responses back into grid space and evaluate using the MSE, that is given by

$$\mathrm{MSE}(f, x, y) = \frac{1}{MG} \sum_{i=1}^{M} \sum_{j=1}^{G} \left| f(x_i)(g_j) - y_i(g_j) \right|^2. \tag{2.96}$$

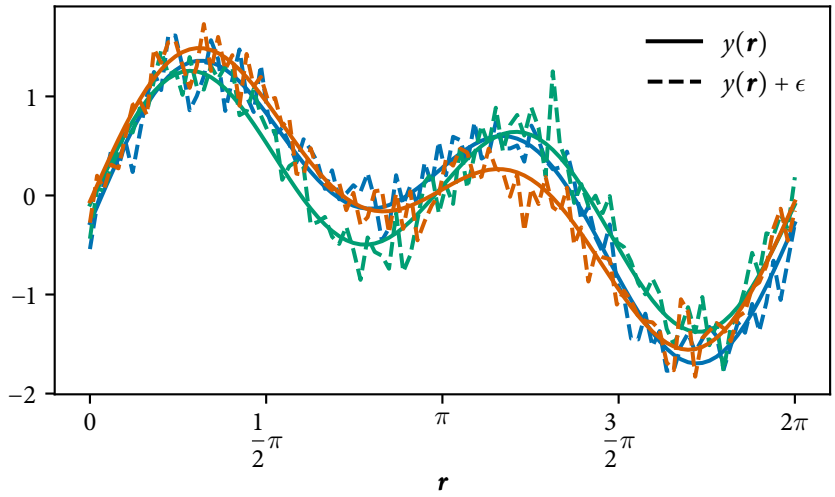


Figure 2.4: Three randomly chosen functional responses $y(\boldsymbol{r})$ (solid curves) and noisy versions $y(\boldsymbol{r}) + \epsilon$ (dashed curves) for the toy dataset with noise level $\tau = 0.2$.

2.3.2 Methods

We compare our approach to three baselines.

- Naive approach: The first baseline is simply the naive approach to functional response learning, i.e. learning each basis function coefficient independently. This is equivalent to the trivial operator-valued kernel learning approach or the integral operator-valued kernel learning approach using the delta function as a linkage function.
- Independent CV approach: The second approach is a slight variation of the first. We can learn each basis function coefficient independently but also cross-validate its hyper parameters independently. This gives the model more flexibility and might lead to better performance. On the other hand, it might overfit on higher frequency basis function coefficient, should the coefficients accidentally correlate with the noise. This case is more likely for larger basis sets.
- Moving average approach: In this approach, we take the model generated

by the naive approach and apply a moving average (MA) filter with rolling window of size $2l+1$ ($l \in \mathbb{N}$) to its results, i.e.

$$\mathrm{MA}(f(x))(g_i) = \frac{1}{\min(l, G) - \max(0, -l) + 1} \sum_{j=\max(0,-l)}^{\min(l,G)} f(x)(g_j). \tag{2.97}$$

This approach might work well on this toy dataset to counter the random iid noise we added. On other datasets, where the noise is not iid, it might decrease prediction performance.

For the **Integral operator** approach with Fourier basis functions, we cross-validate the integral operator linking function parameter ν over $\{10^{-3}, \ldots, 10^{1}\}$.

2.3.3 Cross-validation and hyper-parameter selection

We cross-validate the Gaussian kernel parameter σ and the regularization parameter λ over $\{2^{-60}, \ldots, 2^{20}\}$. We use 5-fold cross-validation for the hyper-parameters (same folds for every method). To evaluate the performance we randomly split the dataset into 90 training samples and 10 test samples. The hyper-parameters are cross-validated on the training samples, so the test samples remain unseen until evaluation.

We repeat the procedure including the drawing of the random variables 3 times and for different noise levels τ.

2.3.4 Results

The results are given in Table 2.1 and are visualized in Fig. 2.5.

The integral operator-valued approach outperforms the other approaches consistently. The effect is largest for scenarios with more noise. It gives consistently good results even in low-noise settings unlike the moving average correction. The independent CV approach overfits in the high noise setting. The naive approach, the independent CV approach and the integral operator-valued kernel approach yield similar performance in low-noise settings.

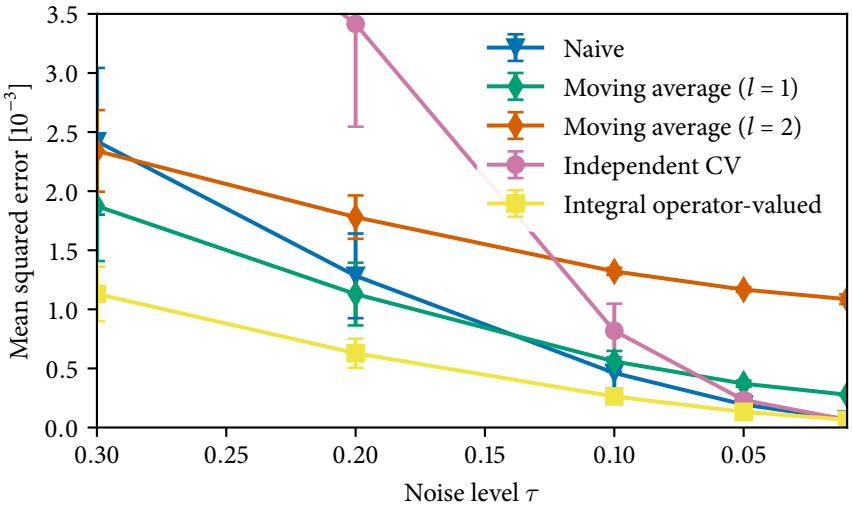


Figure 2.5: Comparison of different model approaches for the toy dataset.

	Noise level τ				
Method	0.3	0.2	0.1	0.05	0.01
Naive	2.4±0.62	1.3 ±0.36	0.46±0.13	0.19±0.050	0.069±0.0031
Moving avg. (l = 1)	1.9±0.46	1.1 ±0.27	0.56±0.088	0.37±0.026	0.28 ±0.0067
Moving avg. (l = 2)	2.3±0.35	1.8 ±0.18	1.3 ±0.031	1.2 ±0.021	1.1 ±0.041
Independent CV	5.1±1.0	3.4 ±0.87	0.82±0.23	0.23±0.048	0.065±0.0016
Integral operator	1.1±0.23	0.63±0.12	0.26±0.050	0.13±0.021	0.065±0.00096

Table 2.1: Mean squared errors in 10^{-3}. We report the mean and standard deviation over the repetitions.

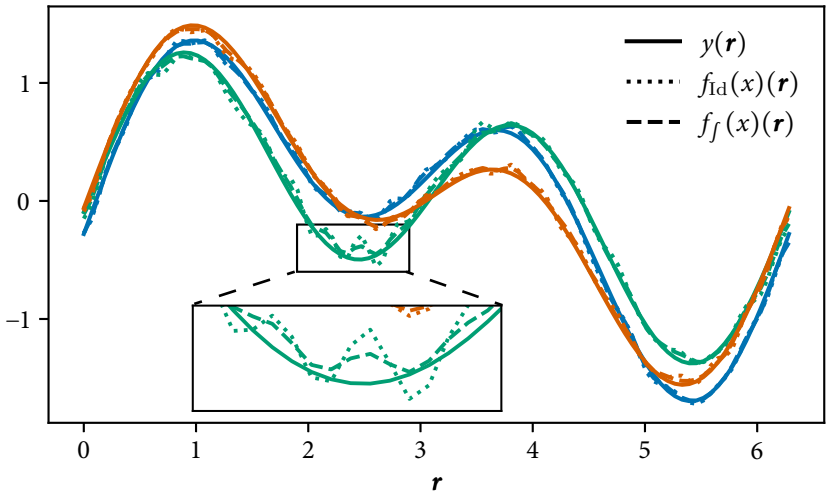


Figure 2.6: Three randomly chosen example prediction for the toy dataset with noise level $\tau = 0.2$. For each example, the functional response without noise (f, solid line), the prediction of the identity operator-valued kernel model (f^{Id}, dotted line), and the prediction of the integral operator-valued kernel model ($f^{\int}$, dashed line), are shown.

2.4 Discussion

We can note two important results. First, solving the regularized least-squares regression problem for the integral operator-valued kernel and Fourier basis representations is computationally more complex than a naive functional response learning approach but scales nicely. We can even reuse standard KRR implementations.

Second, we can now interpret what the integral operator-valued kernel does from an intuitive point of view: Since it modifies the regularization to $\lambda/\tilde{\mu}$, we can say that it increases regularization of high-frequency Fourier basis functions. The effect of this regularization could be controlled by adding a hyperparameter to the exponential of m, as mentioned above.

One might argue that smoothing achieves a similar effect. And although smoothing is a well established technique in functional data analysis [RS05], the integral operator-valued kernel yields a more sophisticated approach. It

does not apply blind smoothing but instead shifts the bias-variance tradeoff for high-frequency basis coefficients.

We have thus a method that regularizes twofold: the integral-operator valued kernel regularizes the model function $f(x)$ over x via λ and the functional responses $(f(x))\,(\boldsymbol{r})$ over $\boldsymbol{r}$ via ν.

However, regularizing the functional responses provides little improvement in accuracy for low-noise settings such as for electron densities. The additional computational complexity on the other hand makes this approach less practical for large numbers of basis functions. The ML-HK map in Sec. 3.2.2 is therefore introduced using the trivial operator-valued kernel.

3

Hohenberg-Kohn map

The previous chapters introduced the basics of DFT, basis representations, and functional prediction. Now these concepts will be applied to predicting electron densities. First, the discussion of the kinetic energy approach in Sec. 3.1 will serve as a motivation to model the electron density as a functional prediction problem. This concept is introduced in Sec. 3.2.2 as the machine-learning-Hohenberg-Kohn (ML-HK) map. Additionally, an ML energy functional E^{ML} that allows practical application of the ML-HK map is described in Sec. 3.2.3. Finally, the machine-learning-Kohn-Sham (ML-KS) map is introduced in Sec. 3.2.4 to serve as a benchmark and allow relative performance evaluation. The goal is to introduce a robust possibility to learn and apply an ML model for electron densities in 3-D. Finally, a set of physically interpretable error measures is introduced in Sec. 3.3.

3.1 Kinetic energy approach

This section reviews the previous approaches to learn electron densities and density functions in 1-D. Additionally it gives some analysis of why this approach does not translate to 3-D.

3.1.1 Introduction

Density functional theory tells us that the energy is a function of the electron density, $E = E[n]$. A natural approach is to guess a density and try a gradient descent approach.

In its most simple form, we can assume that the electrons are not interacting, thus assuming that $U = E_{\mathrm{XC}} = 0$ and $F[n] = T_s[n]$. This leaves us for 1-D with

$$E[n] = T_s[n] + \int \mathrm{d}\boldsymbol{r}\, v(\boldsymbol{r})\, n(\boldsymbol{r}). \tag{3.1}$$

Since we have

$$\frac{\delta E}{\delta n(\boldsymbol{r})} = \frac{\delta T_s[n]}{\delta n(\boldsymbol{r})} + v(\boldsymbol{r}), \tag{3.2}$$

we only have to find the derivative of T_s for the gradient descent approach. The derivative of T_s at a given density is equal to the associated external potential plus the chemical potential μ, as given by the Euler-Lagrange equation

$$\frac{\delta T_s}{\delta n(\boldsymbol{r})} = -v(\boldsymbol{r}) + \mu. \tag{3.3}$$

That would require us to learn a map from n to v. Although the Hohenberg-Kohn theorem gives us a one to one relationship between v and n, learning a map from n to v is much harder than learning a map from v to n. The different complexity of these maps is apparent from the perspective of electronic structure calculations. Whereas there it is relatively straight-forward to find the density given a potential with modern electronic structure codes, the most promising strategy to find a potential given a specific density is to guess an initial potential and iterate, comparable to an inverse problem. Comparing the L_2 distances between potentials and densities of the 1-D dataset introduced in Snyder et al. [Sny+12] supports this hypothesis. For each data point (v, n, E), we gather the three nearest neighbors with respect to the potential v and the three nearest neighbors with respect to the density n. For each neighbor pair, we visualize the distances between potentials and densities in Fig. 3.1. We can observe that similar potentials lead to similar densities, however, similar densi-

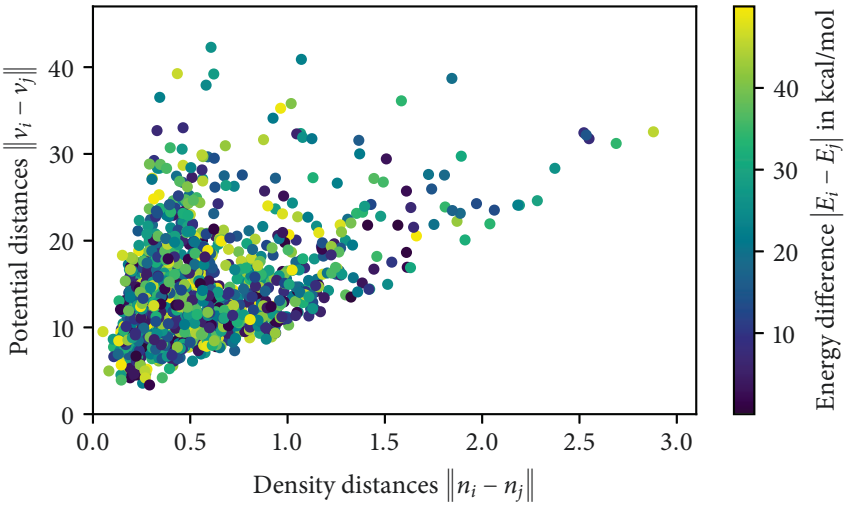


Figure 3.1: L_2 distances between densities Δn against distances between potentials Δv for three nearest neighbors of potentials and densities, colored by the difference in energy ΔE, for the 1-D dataset introduced in Snyder et al. [Sny+12].

ties can result in very different potentials. This further explains why learning a density → potential map can not be successful.

Instead, it would be much easier to learn a machine learning model for the kinetic energy T_s. Since T_s is a universal functional it technically does not require a new training set for each application. However, it is unlikely that different applications (e.g. simulations of different molecules) lead to similar densities. It would be necessary to *scale* the approach by predicting kinetic energy contributions for single atoms or smaller parts of the molecule.

3.1.2 Gradient descent

The approach of learning a model for the kinetic energy functional and finding ground-state densities and energies through gradient descent minimization has been explored in Snyder et al. [Sny+12]. Several follow-up papers [Sny+13a; Sny+13b; Sny+15; Li+16b] explore improved solutions to the main challenge of this approach — the gradient noise issue, which is discussed in Sec. 3.1.3.

If we assume a training set with densities $n_1, \ldots, n_M$, kernel model with kernel k of the form[1]

$$T^{\mathrm{ML}}[n] = \sum_{j=1}^{M} \alpha_j k[n_j, n] \tag{3.4}$$

with model parameters α, we can calculate the functional derivative analytically. We assume that k is a Gaussian kernel, whose functional derivative is given by

$$\frac{\delta k[n, n']}{\delta n(\boldsymbol{r})} = \frac{1}{\sigma^2}(n'(\boldsymbol{r}) - n(\boldsymbol{r}))k[n, n']. \tag{3.5}$$

The functional derivative of the kernel model T^{ML} is then given by

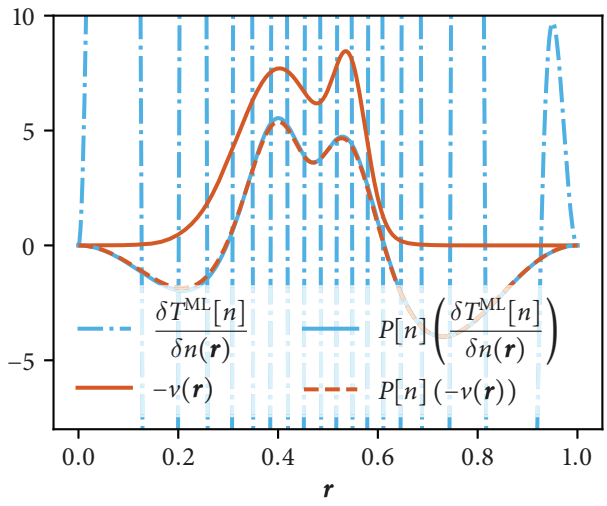
$$\frac{\delta T^{\mathrm{ML}}[n]}{\delta n(\boldsymbol{r})} = \sum_{j=1}^{M} \frac{1}{\sigma^2}\alpha_j(n(\boldsymbol{r}) - n_j(\boldsymbol{r}))k[n_j, n]. \tag{3.6}$$

For orthogonal basis functions, we can just calculate with the vectorial basis coefficients instead of the density functions. The Euler-Lagrange equation (Eq. 3.3) gives us the correct derivative and thus a possibility to evaluate the derivative of the kinetic energy model directly. Unfortunately, the accuracy turned out to be disastrous. The training densities lie on a manifold and are thus very sparse inside the space of functions the kinetic energy model can map. With the functional derivative, we want to know how the kinetic energy would change when modifying a density in every direction independently. This information is not given by the data and the kinetic energy model has no knowledge about it. The functional derivative calculated in Eq. 3.6 is a regularized extrapolation, but is far from the truth. An example derivative is given in Fig. 3.2a.

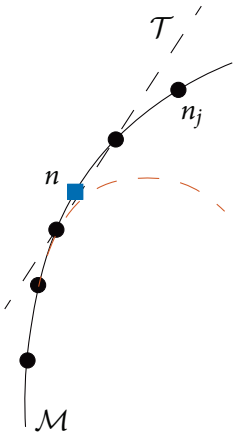
3.1.3 Gradient de-noising

The oscillating derivatives are not encouraging, but a strategy is presented: We assume that the density that minimizes $E[n]$ lies on the training densities man-

[1] Note that T^{ML} can be an operator from the functional space of densities or a multivariate function from a discretized function or basis coefficient space. Adaption of the presented equations is straight-forward.



(a) Gradient, potential, and projections



(b) Projection idea

Figure 3.2: (a) Examples of a functional derivative of the kinetic energy model. The solid blue line is the true derivative $-v(\boldsymbol{r})$ (the negative corresponding external potential), the dot-dashed blue line is the derivative of the kinetic energy model $\frac{\delta T[n]}{\delta n(\boldsymbol{r})}$ and oscillates wildly. The dashed orange and solid blue lines are the true derivative and the derivative of the kinetic energy model projected on the density manifold, $P[n](-v(\boldsymbol{r}))$ and $P[n]\left(\frac{\delta T[n]}{\delta n(\boldsymbol{r})}\right)$. (b) The idea of using a localized principle component analysis (PCA) projection in each gradient descent step. The densities n_j lie in the densities manifold $\mathcal{M}$. Optimizing the total energy might traverse the dashed orange line out of the regions with data points. Applying a localized PCA based on a fixed number of neighbors (2 here), provides a manifold $\mathcal{T}$ where a small step can be taking without leaving the region with data points.

ifold and can thus restrict the gradient descent trajectory to the densities manifold where the kinetic energy model has more information. The idea is visualized in Fig. 3.2b.

To restrict the gradient descent trajectory, we have to describe the densities manifold. The earliest idea, first presented in Snyder et al. [Sny+12] and described in more detail in Li et al. [Li+16b], utilizes a *localized* PCA. Here, localized means that for each gradient descent step, the local neighborhood of the current density is considered. The principle components of the p nearest neighbors are computed and the gradient is projected onto the local PCA manifold and back via a localized projected operator $P[n]$. The complete projection gradient descent update step is

$$n_{(t)}(\boldsymbol{r}) = n_{(t)}(\boldsymbol{r}) - \epsilon P[n_{(t)}] \left(\left. \frac{\delta T^{\mathrm{ML}}[n]}{\delta n(\boldsymbol{r})} \right|_{n=n_{(t)}} + v(\boldsymbol{r}) \right). \tag{3.7}$$

This method accurately reproduces the projected correct gradient. Fig. 3.2a shows a comparison between the projected gradient prediction and the *projected* potential.

While the presented approach works for the 1-D particle in a box dataset, it has not been applied successfully to more complex systems. Suggestions have been made to improve the robustness of the gradient procedure, mainly by improving the description of the densities manifold.

The densities manifold can be described globally by Kernel PCA components, see Sec. 2.1.3. Global in the sense that Kernel PCA components are computed once for the complete training set and stay the same for every gradient descent step. The projection in features space is given by

$$Q = \sum_{l=1}^{L} \varphi_l \varphi_l^\top \tag{3.8}$$

where φ_l are the Kernel PCA components in feature space.

Optimization along a manifold described by a small number of Kernel PCA components can be more stable because the localized PCA components do not combine to a smooth global manifold. Since a global Kernel PCA avoids the

necessity to compute localized principle components in every gradient step, it is much faster as well.

Restricting the optimization trajectory to a Kernel PCA manifold, however, still allows the gradient steps to traverse into regions outside the training data. These regions still lie on the densities manifold but the kinetic energy model might be very inaccurate in this area. Additionally, the number of Kernel PCA components has to chosen in advance. A specific choice might be too little for some areas of the training data or too many for others.

Ideally, the number of components that are used to describe the densities manifold should be adapted locally. Although this is slower, it can give more accurate results and be more robust. Crucial is the question on how to adapt the dimensionality of the local densities manifold.

One method is presented as *non-linear gradient denoising* in Snyder et al. [Sny+13a] and Snyder et al. [Sny+15]. An application to 1-D bond-breaking is presented in Snyder et al. [Sny+13b]. Here, the general idea is to observe the curvature of the function that describes the Kernel PCA reconstruction error

$$p[n] = \|\Phi[n] - Q\Phi[n]\|^2 \tag{3.9}$$

and analyze its Hessian. The eigenvalues of the Hessian are a measure of the amount of curvature along the direction of the corresponding eigenfunctions. High curvature corresponds to areas outside the training set region. It is thus possible to set a cut-off parameter for the curvature and construct a projection with all eigenfunctions corresponding to eigenvalues below the cut-off parameter.

Unfortunately even with these improvements, the approach of learning a kinetic energy functional and following the energy gradient remains challenging because of the trade-off between describing the densities manifold accurately and removing the noise. Snyder et al. [Sny+15] compares the non-linear gradient de-noising approach with the localized PCA approach for the 1-D particle in a box dataset. In Sec. 4.1 we show a comparison between the gradient approaches and density prediction approaches.

3.2 Electron density prediction

This section introduces the machine-learning-Hohenberg-Kohn (ML-HK) map for predicting the electron density from potentials. It also introduces the energy map E^{ML} in Sec. 3.2.3 to allow practical application of the ML-HK map. Finally, the machine-learning-Kohn-Sham (ML-KS) map is introduced in Sec. 3.2.4 to allow comparison and performance evaluation in 3-D.

3.2.1 Introduction

Since the application of the kinetic energy approach (Sec. 3.1) to 3-D molecules remains unsuccessful, a different approach seems necessary. A small motivation has been given while discussing the one-to-one relationship between potential and density via the Hohenberg-Kohn theorem

$$v \longleftrightarrow n. \tag{3.10}$$

While the $n \to v$ map is hard to learn via machine learning, the $v \to n$ map seems more promising. We refer to this map as the *Hohenberg-Kohn map*.

A theoretical motivation for learning the Hohenberg-Kohn map can be derived from Ribeiro et al. [Rib+15]. Here, the authors show that Hohenberg-Kohn map can be approximated extremely accurately using semiclassical expressions.

It is thus suggested to circumvent the kinetic-energy approach and directly learn a multivariate machine learning model for the Hohenberg-Kohn map, a machine-learning-Hohenberg-Kohn (ML-HK) map.

Besides the ML-HK map, we differentiate two other machine learning approaches in the following. One approach is the kinetic energy approach discussed in Sec. 3.1, also referred to as the orbital-free (OF) approach. The second approach was discussed in Sec. 1.2.3 and is the idea of learning potential energy surface (PES), or a direct map from potential to energy ($v \to E$). We refer to this map as the Kohn-Sham (KS) map. An overview of the different approaches is given in Fig. 3.3.

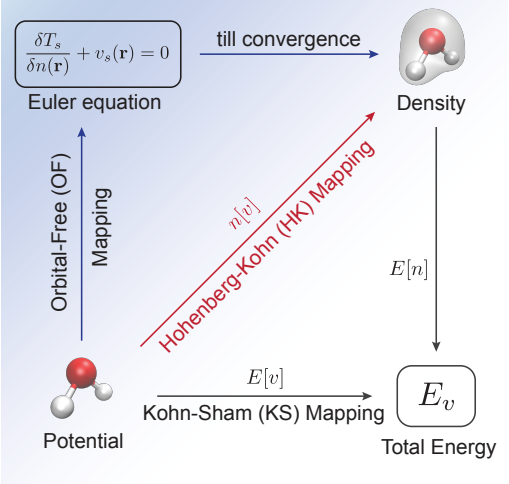


Figure 3.3: The bottom arrow represents $E[v]$, the result of a conventional electronic structure calculation, i.e. KS-DFT, and is thus called the KS map. $E[n]$ is the total energy density functional. The red arrow is the HK map $n[v]$ from external potential to its ground state density. The upper left side describes the approach of learning the kinetic energy functional which leads to a solution without KS orbitals and is thus called the orbital-free (OF) approach.

3.2.2 The machine-learning-Hohenberg-Kohn map n^{ML}

Learning a machine learning model for the HK map describes a functional regression problem as discussed in Chapter 2. Here, not only the output as the density n is functional, but also the input as the potential v. Although this setting received special attention in earlier research [RS05], the application of kernel methods with isotropic (or homogeneous) kernels (like the Gaussian or Laplace kernel) solely require a distance measure. Here, we use the natural L^2 norm to measure distances between potentials.

To keep notation consistent, we write $f[v]$ for the physical relationship and $f^{\mathrm{ML}}[v]$ for the machine learning equivalent, e.g. $n[v]$ and $n^{\mathrm{ML}}[v]$.

Since the training data for the ML-HK map will stem from converged electronic structure calculations, we expect little benefit from introducing a regularization for the functional responses $n^{\mathrm{ML}}[v](\boldsymbol{r})$ over $\boldsymbol{r}$ as additionally validated by the experiments in Sec. 2.3. We thus use a trivial operator-valued kernel and the problem reduces to learning several independent regression models, one for each basis function coefficient.

For the scalar-valued part, we use the Gaussian kernel

$$G(v, v') = \exp\left\{-\sigma \left\| v - v' \right\|^2\right\}. \tag{3.11}$$

It is not only robust and yields excellent results in a wide variety of settings but also agrees with the physical intuition that the HK map is smooth.

We define an artificial Gaussian potential for use in the ML-HK map. It is designed to give an appropriate distance measure for the potential but will not be physically accurate or relevant since it is never used to calculate physical quantities directly. We define it for a set of charges Z_i and atom positions R_i as

$$v(\boldsymbol{r}) = \sum_{i=1}^{N^{\text{atoms}}} Z_i \exp\left\{\frac{\|R_i - \boldsymbol{r}\|^2}{2\gamma^2}\right\}. \tag{3.12}$$

The γ is a hyper-parameter and can be cross-validated. A value of $\gamma = 0.2$Å is reasonable and yields good results in all applications.

For given training data with potentials $v_1, \ldots, v_M$ and densities $n_1, \ldots, n_M$ with basis representation

$$n_i(\boldsymbol{r}) = \sum_{l=1}^{L} \tilde{n}_{il} \varphi_l(\boldsymbol{r}), \qquad i = 1, \ldots, M, \tag{3.13}$$

we thus write the ML-HK map as

$$n^{\text{ML}}[v](\boldsymbol{r}) = \sum_{l=1}^{L} \beta_l[v] \varphi_l(\boldsymbol{r}) = \sum_{i=1}^{M} G(v, v_i) \sum_{l=1}^{L} \tilde{\beta}_{il} \varphi_l(\boldsymbol{r}) \tag{3.14}$$

with independent maps for each basis function coefficient

$$\beta_l[v] = \sum_{i=1}^{M} \tilde{\beta}_{il} G(v, v_i) \tag{3.15}$$

for which we have to solve

$$\tilde{\beta}_l = (\mathbf{G} + \lambda \mathbf{I})^{-1} \begin{pmatrix} \tilde{n}_{1l} \\ \vdots \\ \tilde{n}_{Ml} \end{pmatrix}. \tag{3.16}$$

3.2.3 The energy map E^{ML}

Once we have a density, we can predict further properties like the total energy. The total energy density functional is given by

$$E_v[n] = T_s[n] + U[n] + E_{\mathrm{XC}}[n] + \int \mathrm{d}^3 r \, v(\boldsymbol{r}) \, n(\boldsymbol{r}). \tag{3.17}$$

Since the Hartree functional U is known and good approximations exist for E_{XC}, it would be sufficient to learn a machine learning model for T_s. This approach worked well for non-interacting particles in 1-D [Sny+12]. For real world applications, however, there are practical challenges. We usually do not work with full electron densities, but with pseudo-densities stemming from the use of pseudo-potentials, see Sec. 1.2.1. Evaluations of the functionals would then require implementations for the specific pseudo-potentials used. The complexity of these implementations make them impractical for our experiments. Instead, we work with pseudo-densities and learn a machine learning model for the total energy $E[n]$.

Learning a model for the total energy $E[n]$ is a simple regression problem and can be trained via KRR. The kernel model is given by

$$E^{\mathrm{ML}}[n] = \sum_{j=1}^{M} \alpha_j k_{\mathrm{E}}(n, n_j). \tag{3.18}$$

Many practical applications (e.g. the simulation of molecular dynamics) will require a chained evaluation of the HK map and the energy map, i.e. $E^{\mathrm{ML}}[n^{\mathrm{ML}}[v]]$. How fast this evaluation can be computed will depend on the dimensionality of the n^{ML} output, i.e. the number of basis functions L used for n.

It is desirable to make the chained evaluation independent of the number of basis functions. A theoretical possibility exists but turns out to be numerically impractical: Evaluation of the chained models $(E^{\mathrm{ML}} \circ n^{\mathrm{ML}})(v)$ requires (as part of the kernel evaluation k_{E} and assuming orthogonal basis functions) computation of the distances D with

$$D_j = \|n^{\mathrm{ML}}(v) - n_j\| = \left\| \sum_{i=1}^{M} \tilde{\beta}_i G(\|v - v_i\|) - \tilde{n}_j \right\|. \tag{3.19}$$

Introduce the shortcut $k = (G(\|v - v_1\|), \ldots, G(\|v - v_M\|))^T$. Then the distances require evaluation of the dot product $\tilde{\beta}k$ with $\tilde{\beta} \in \mathbb{R}^{L \times M}$ and L very large. This costly operation has to be repeated several times in every MD simulation step.

Fortunately, we can rewrite the distance evaluation as:

$$D_j^2 = \left\| \sum_{i=1}^{M} \tilde{\beta}_i G(\|v - v_i\|) - \tilde{n}_j \right\|^2 \tag{3.20}$$

$$= \|\tilde{\beta}k - \tilde{n}_j\|^2 \tag{3.21}$$

$$= k^T \tilde{\beta}^T \tilde{\beta} k - 2k^T \tilde{\beta}^T \tilde{n}_j + \tilde{n}_j^T \tilde{n}_j \tag{3.22}$$

This allows us to precompute $\tilde{\beta}^T \tilde{\beta} \in \mathbb{R}^{M \times M}$, $\tilde{\beta}^T \tilde{n}_j \in \mathbb{R}^M$, and $\tilde{n}_j^T \tilde{n}_j \in \mathbb{R}$ which makes the distance evaluation, that is necessary in every MD step, independent of the number of basis functions L.

Unfortunately this approach turns out to be rather impractical. It is well known that the absolutes of the individual parts of the summation in Eq. 3.22 are often much larger than D_j^2 which results in numerical errors. Here, these errors are particularly large. This can be alleviated partially by increasing the regularization parameter of the n^{ML} model to yield smaller $\tilde{\beta}$ values but nevertheless results in significant performance drops of the total energy prediction.

3.2.4 The ML-KS map $E^{\mathrm{ML\text{-}KS}}$

Since the orbital-free approach will turn out too inaccurate for application in 3-D, this section introduces a machine learning model for the Kohn-Sham (KS)

map $E[v]$ that maps the external potential v to the total energy E (see Fig. 3.3). Just as the energy map E^{ML}, it solves a simple regression problem and can be trained via KRR. The kernel model of the ML-KS map is given by

$$E^{\text{ML-KS}}[v] = \sum_{j=1}^{M} \alpha_j k(v, v_j). \tag{3.23}$$

This map can be used to benchmark the performance of the Hohenberg-Kohn map n^{ML} in Sec. 4.3. It is not meant to compete with other potential energy surface (PES) approaches (Sec. 1.2.3). Predicting the total energy given a potential will be less accurate than approaches that start with the atomic positions, due to the restricted normalization options as will be explained in Sec. 4.2.1.

3.3 Model evaluation and error measure

To train the ML-HK map, we optimize the L_2 norm. To evaluate the accuracy of the density predictions this is not intuitive though. The requirement from the application perspective is that the density prediction must be accurate enough so that the resulting total energy reaches chemical accuracy compared to the reference (e.g. PBE) energy of the system.

Recently it has been shown how to separate out the error in the density and the error in the functional on the resulting total energy error of any approximate, self-consistent DFT calculation [KSB13]. Let $\tilde{F}$ be an approximate many-body functional and $\tilde{n}$ an approximate ground-state density that results from using $\tilde{F}$ in the Euler equation (Eq. 1.9). By defining $\tilde{E}[n] = \tilde{F}[n] + \int \mathrm{d}^3 r\, n(\boldsymbol{r})\, v(\boldsymbol{r})$, it is possible to separate the error made by both the density approximation and the functional approximation as

$$\Delta E = \tilde{E}[\tilde{n}] - E[n] = \Delta E_{\mathrm{F}} + \Delta E_{\mathrm{D}} \tag{3.24}$$

with

$$\Delta E_{\mathrm{F}} = \tilde{F}[n] - F[n] \tag{3.25}$$

being the functional-driven error and

$$\Delta E_{\mathrm{D}} = \tilde{E}[\tilde{n}] - \tilde{E}[n] \tag{3.26}$$

being the density-driven error. For most DFT calculations, ΔE is dominated by ΔE_{F}. The standard DFT approximations can, in some specific cases, produce abnormally large density errors that dominate the total error. In such situations, using a more accurate density can greatly improve the result [KSB13; KSB14; Kim+15].

Inspired by this separation of errors, we can define accuracy measures for the machine learning application.

Since we will not train a separate machine learning model for the many-body functional $F[n]$ in 3-D (only a total energy model E^{ML}) we define

$$E^{\mathrm{ML}}[n] = T^{\mathrm{ML}}[n] + \int \mathrm{d}\boldsymbol{r}\, n(\boldsymbol{r})\, v(\boldsymbol{r}) \tag{3.27}$$

for 1-D experiments. This allows us to keep the notation between 1-D and 3-D consistent and leads us to define the following error measures. We differentiate the energy error, a functional-driven error, and an ML-density-driven error.

The *energy error*, given a potential v, is measured as

$$\Delta E = E^{\mathrm{ML}}\left[n^{\mathrm{ML}}[v]\right] - E_v. \tag{3.28}$$

The *functional-driven* error, given a density n, is measured as

$$\Delta E_{\mathrm{F}} = E^{\mathrm{ML}}[n] - E[n]. \tag{3.29}$$

Note that this leads to $\Delta E_{\mathrm{F}} = T^{\mathrm{ML}}[n] - T_s[n]$ in 1-D. Lastly, the (ML-)*density-driven error*, given a potential v, is measured as

$$\Delta E_{\mathrm{D}}^{\mathrm{ML}} = E^{\mathrm{ML}}\left[n^{\mathrm{ML}}[v]\right] - E^{\mathrm{ML}}[n_v]. \tag{3.30}$$

In 1-D, where we limit the experiments to 1-electron simulations, we can use the

von-Weizsäcker kinetic energy functional [DG90],

$$T^{\mathrm{vW}}[n] = \frac{1}{8}\int \mathrm{d}\boldsymbol{r}\,\frac{\nabla n(\boldsymbol{r})\cdot\nabla n(\boldsymbol{r})}{n(\boldsymbol{r})}, \tag{3.31}$$

that is exact for 1-electron systems. Using this functional, we can define a more exact *density-driven error* for 1-D experiments only:

$$\Delta E_{\mathrm{D}} = T^{\mathrm{vW}}\left[n^{\mathrm{ML}}[v]\right] + \int \mathrm{d}\boldsymbol{r}\, n^{\mathrm{ML}}[v]\, v - \left(T^{\mathrm{vW}}[n_v] + \int \mathrm{d}\boldsymbol{r}\, n_v\, v\right). \tag{3.32}$$

Since the densities vanish at the edges, a naive implementation of the von-Weizsäcker functional leads to numerical inaccuracies. We therefore first define $\varphi(\boldsymbol{r}) = \sqrt{n(\boldsymbol{r})}$ and calculate its numerical derivative via central differences. Then we can calculate the kinetic energy by integrating, e.g. via Simpson's rule,

$$T_s[n] = \frac{1}{2}\int \mathrm{d}\boldsymbol{r}\left|\varphi'(\boldsymbol{r})\right|^2. \tag{3.33}$$

Since the general kinetic energy functional is unknown, we can not use the same definition for E_{D} for the experiments in 3-D.

To ensure that the energy error ΔE is comparable to errors made by a KS map, we always use the same training point pairs (v, n, E, T) (or (v, n, E) in 3-D) for training both the n^{ML} and E^{ML} maps.

3.4 Discussion

The ML-HK map is conceptually different to the gradient descent approach. On the one hand, this is motivated by the challenge to describe and limit the gradient descent to the density manifold. On the other hand, it is motivated by the reality of electronic structure codes. Apart from a kinetic energy model, it would also be necessary to implement the other potential functionals, e.g. the exchange-correlation functional, and work with pseudo-potentials. The ML-HK map is immediately applicable to 3-D molecules and provides a more direct path to interesting applications. Electronic structure codes often do not implement a direct kinetic energy functional, but rewrite the problem in terms

of orbital basis coefficients and an eigenvalue problem. More details and how this leads to density matrix predictions is described in Appendix A.

The ML-HK map reduces to a set of independent KRR models and is thus straight-forward without many complexities. This is due to the application driven design approach. The goal is to start with a simple model that works well and robust for the intended application. Adding complexity to the model, e.g. by using an integral-operator valued kernel, comes with additional computational cost whereas the improvement with respect to the application might not be significant. The strategy is therefore to increase the application complexity and only increase the model's complexity once it fails.

In the next chapter we will see that the tangential areas, like functional representation, normalization, choosing which simulations to run for training data generation, and physical properties like the choice of pseudo-potentials, have a larger influence on the machine learning approach's performance than the model itself. This is less interesting from an ML research perspective but demanded by the drive to succeed on the application side.

The errors are measured unconventionally. For example, the ML-HK map can be evaluated by measuring the L^2 error, however, this error does not lead to an easily interpretable quantity and thus does not allow us to see how significant methodological changes are. The following chapter will also look at the density prediction error by comparing the influence of ML, basis representation, and physical theory level.

4

Prediction of quantum mechanical observables

After setting the theoretical foundation and the methodological prerequisites it is now possible to turn to the application. The first Sec. 4.1 will evaluate the ML-HK map on 1-D densities. Sec. 4.2 a deals with representation, normalization, and sampling of molecular structures for the specific case of density prediction. Sec. 4.3 applies the HK map to several 3-D datasets of increasing complexity. Additionally it showcases the ability to simulate MD with the ML-HK map. The final Sec. A covers the prediction of density matrices: an approach that promises a solution for the scalability problem of predicting densities.

4.1 Application to 1-D particle-in-a-box data

We will first evaluate the ML-HK map approach on the 1-D particle-in-a-box dataset published in Snyder et al. [Sny+12]. The dataset includes randomly generated potentials of the form

$$v_i(\boldsymbol{r}) = -\sum_{i=1}^{3} a_i \exp\left\{-|\boldsymbol{r}-b_i|^2/2c_i^2\right\} \tag{4.1}$$

and the resulting exact densities n_i, kinetic energies T_i, and total energies E_i when assuming that particles are subject to the potential v_i in a $[0,1]$ box with infinite walls. The densities were found on an equi-distant grid

$$g_1 = 0, \ldots, g_G = 1 \text{ with } G = 500 \tag{4.2}$$

using Numerov's method [HNW93]. An example of a potential is given in Fig. 3.2a.

To fit the ML-HK map on the 1-D dataset, we use a Gaussian kernel. To measure the distances between potentials we discretize the potentials on the same grid as the densities and take the euclidean distance

$$\left\| v_i - v_j \right\| = \left\| \left(v_i(g_1), \ldots, v_i(g_G)\right)^\top - \left(v_j(g_1), \ldots, v_j(g_G)\right)^\top \right\|_2. \tag{4.3}$$

To get a total energy model $E^{\mathrm{ML}}[n]$, we follow Snyder et al. [Sny+12] and train a KRR model for the kinetic energy $T^{\mathrm{ML}}[n]$ and calculate the potential energy

$$V[n] = \int_0^1 \mathrm{d}\boldsymbol{r}\, n(\boldsymbol{r})\, v(\boldsymbol{r}) \tag{4.4}$$

by integrating over the grid via Simpson's rule.

We employ standard cross-validation procedures to evaluate the performance of our models [Han+13]. We follow the approach of splitting of a hold-out test dataset of 1000 data points to evaluate the ML models on. The remaining points are available for training. Of these points we take subsets of varying sizes M which become the training set for both the T^{ML} and n^{ML} model. The subset is chosen as described in Sec. 4.2.2.

We use 5-fold cross-validation on the training set to choose the hyper-parameters of the model. Both models T^{ML} and n^{ML} have a regularization parameter and a Gaussian kernel width parameter. These parameters for the T^{ML} model and the n^{ML} model are chosen independently from each other. We optimize the hyper-parameters for best mean-squared-error.

The results for the different error measures are given in Table 4.1. A direct comparison of the density-driven error is given in Figure 4.1.

	ML-OF						ML-HK (grid)						ML-HK (other)			
	ΔE		ΔE_F		ΔE_D		ΔE		ΔE_D		ΔE_D^{ML}		ΔE_D (Fourier)		ΔE_D (KPCA)	
M	MAE	max	MAE	max	MAE	max	MAE	max	MAE	max	MAE	max	MAE	max	MAE	max
20	7.7	47	7.7	60	8.7	58	3.5	27	0.76	8.9	9.7	70	0.58	8	0.15	2.9
35	2.3	22	2.4	16	1.8	35	5.2	40	0.18	2.9	4.9	28	0.17	2.9	0.018	0.41
50	1.4	12	1.3	7.3	0.92	10	1.2	7.1	0.079	0.92	0.27	2.4	0.078	0.91	0.011	0.17
75	0.84	10	0.28	3.1	0.61	7.3	0.3	3.5	0.042	0.63	0.17	1.9	0.044	0.64	0.0026	0.061
100	0.74	21	0.2	2.6	0.69	15	0.19	2.1	0.027	0.43	0.18	2.4	0.031	0.42	0.0012	0.028
150	0.23	8.7	0.078	1.4	0.15	11	0.11	1.2	0.01	0.22	0.1	1.2	0.019	0.24	0.0007	0.024
200	0.16	2.4	0.039	0.6	0.1	6.2	0.042	0.59	0.0065	0.15	0.02	0.46	0.017	0.14	0.00055	0.015

Table 4.1: Energy errors for the 1-D particle in a box dataset. The errors are given in kcal/mol for various training set sizes M. Displayed are the energy error ΔE, the functional-driven error ΔE_{F}, the density-driven error ΔE_{D} and its approximation $\Delta E_{\mathrm{D}}^{\mathrm{ML}}$.

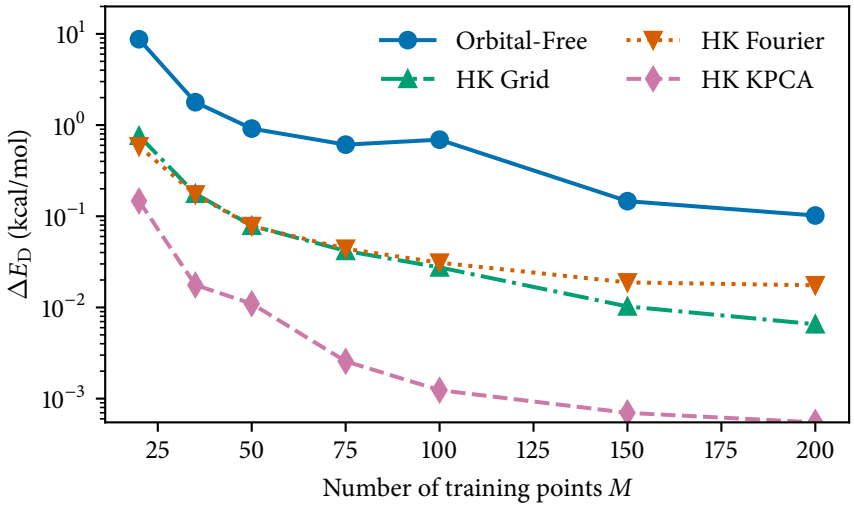


Figure 4.1: The development of the density-driven error with increasing training set sizes for the orbital-free approach and the ML-HK approach with different basis function representations.

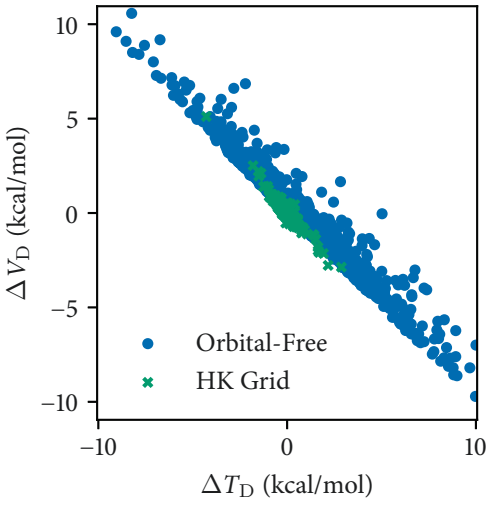
Originally the quality of the densities were measured by a density-driven error based on an ML kinetic energy functional, not a total energy functional, trained on 200 densities. A comparison of these errors in both approaches is given in Table 4.2a. Note that the errors here are much higher than the density-driven error based on the total energy functional (Eq. 3.30). This phenomenon is usual for DFT calculations and is due to a cancellation of errors in kinetic energy and potential energy. A plot of error in kinetic energy vs. error in potential energy for both approaches is given in Fig. 4.2b.

The results show us that the density-driven error ΔE_D for the OF approach is always comparable to, or greater than, the functional-driven error ΔE_F. This is due to the poor functional derivative generated by the OF approach. As mentioned in Sec. 3.3, this is an abnormal calculation and the results can be greatly improved by using more accurate densities.

As the number of training points M grows, the error becomes completely dominated by the density-driven error. This also indicates that the largest source of error lies in solving the Euler equation with the ML approximation of T_s to find

	ML-OF		ML-HK	
M	MAE	max	MAE	ma
20	21	382	4.1	33
35	12	182	0.95	10
50	4.5	95	0.69	6.3
75	2.5	81	0.29	2.9
100	3.3	91	0.22	4.3
150	1.9	64	0.11	3.2
200	2.6	31	0.081	2.3

(a) ΔT_{D} errors for the orbital-free approach and the ML-HK map for a grid representation.



(b) The density-driven error in the kinetic energy functional vs. the potential energy functional.

Figure 4.2

the density.

The ML-HK approach is always more accurate than the OF approach, and its relative performance improves as M increases. The density-driven error ΔE_{D} is even an order of magnitude better for ML-HK than for OF. The approximation $\Delta E_{\mathrm{D}}^{\mathrm{ML}}$ which uses T^{ML} is a considerable overestimate (a factor of 3 too large). This justifies measuring the density-driven error in 3-D using $\Delta E_{\mathrm{D}}^{\mathrm{ML}}$ alone.

The high dimensionality of the grid representation is prohibitive in 3-D. Therefore a Fourier basis representation is tested using $l = 200$ basis functions. The performance is almost identical to the grid on average, although maximum errors are much less. For $M = 100$, the error that originates from the basis representation starts to dominate. This is a motivation for exploring a Kernel PCA basis representation. Good results are achieved with only 25 basis functions. The Kernel PCA approach gives better results because it can take the non-linear structure in the density space into account. However, it introduces the pre-image problem: It is not trivial to project the densities from Kernel PCA space

back to their original (grid) space (see Sec. 2.1.3). It is thus not immediately applicable to 3-D and we therefore use the Fourier basis for the 3-D molecules in Sec. 4.3.

4.2 Geometry normalization and sampling in 3-D

4.2.1 Representation and normalization

To use kernel based machine learning approaches, it is necessary to transform the features into a format that the kernel works on. For most kernels this is a vectorial form. In the previous section the input data was given in the form of a potential, however, the input data in 3-D is usually given in the form of molecular geometries, i.e. atom positions and charges. When representing a molecular geometry in a vectorial form we should pay attention to the invariances of a molecule. For example, a rotation of a molecular system changes atom positions but does change the energy of the system.

Machine learning models benefit tremendously from features that keep all the invariances of the underlying data. In the case of rotation invariance a Gaussian kernel based model would need training data for each rotation angle if the features are not rotation invariant. This is because it would see rotated geometries as different geometries which properties it would learn separately although it is the same molecule.

In the case of density prediction, there is another complication. Rotating a molecular system rotates its density as well, it is rotation *transparent* ($f(\mathbf{R}x) = \mathbf{R}f(x)$). It is thus not possible to use a rotation invariant feature representation, because the rotation information is necessary for the density prediction.

Molecules are rotation and translation invariant, but also invariant to atom indexing: it does not matter in which order the atoms are listed, the system stays the same. For energy prediction, it is preferable to use a representation that is invariant to all these properties. For density prediction, the feature representation and model has to be rotation and translation transparent, but should be invariant to atom indexing.

We note that even in the 1-D case training data could be almost halved when the rotation (or flip) invariance would be taken into consideration. The presented results, however, do not take any invariances into account to stay comparable with other published results.

Different concepts for invariances

We can differentiate three concepts to treat invariances in molecular geometries.

The first concept is to find a representation that is *intrinsically* invariant. An example is the Coulomb matrix presented in Rupp et al. [Rup+12]. The coulomb matrix is intrinsically invariant to rotation and translation because it is based on distances between atoms. An extension to make the representation invariant to atom indexing is presented in Montavon et al. [Mon+13]. A representation that changes continuously with atom position changes is given in Bartók, Kondor, and Csányi [BKC13]. An examples for crystal structures is given in Schütt et al. [Sch+14]. An example for building an intrinsically invariant model via neural networks is given in Schütt et al. [Sch+17]. The community's predominant interest in predicting energies explains the popularity of this concept. For predicting densities, we can not use translation and rotation invariant representations, because the model has to be transparent towards these invariances.

This leads us the the second concept: *normalization*. The idea is to translate and rotate all molecular geometries so that they are most similar to each other, then find a representation that is invariant to atom indexing, but not invariant to translations or rotations. This should yield the benefit of considering invariances but still allow the predictions of densities. We discuss different approaches that use this concept below.

The third concept is data augmentation. Instead of making the representation invariant, we can, for example, rotate a geometry and introduce it as a new data point. However, this concept is not suitable for kernel learning approaches and almost only used with iterative learning approaches because the number of training points multiplies. An example application of this concept is to flip images when training image classifiers.

Normalization approaches

For smaller molecules with few parameters, we can find a configuration specific normalization approach. For example, for H_2 the atoms can be placed centered on the x axis. For H_2O the Oxygen atom can be placed in the coordinate system center, the bonds can have equal angle from the y axis with the longer bond always being on a fixed side.

The configuration specific approaches become to complicated to set up for larger molecules with many parameters. We therefore introduce two methods. The first method is based on principle directions in the molecule and the second methods is based on aligning with a base geometry.

For the first method, we calculate the *principle directions* of the atom positions weighted by their atomic mass, similar to PCA. Let R_i be the atom positions and m_i the masses. We compute the centered atom positions $\bar{R}_i = R_i - \bar{R}$, where

$$\bar{R} = \frac{1}{\sum_{i=1}^{N^{\text{atoms}}} m_i} \sum_{i=1}^{N^{\text{atoms}}} m_i R_i \tag{4.5}$$

is the weighted molecule center. Then we compute the scatter matrix

$$S = \sum_{i=1}^{N^{\text{atoms}}} m_i^2 \bar{R}_i \bar{R}_i^\top \tag{4.6}$$

and the principle directions of the molecule are given by its eigenvectors V_j sorted by its corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$. To avoid the sign ambiguity of the eigenvectors, we project the weighted atom positions onto the eigenvectors and correct each principle direction by the sign

$$V_j' = V_j \operatorname{sign}\left(\sum_{i=1}^{N^{\text{atoms}}} V_j^\top \bar{R}_i m_i \right), \quad j = 1, 2, 3. \tag{4.7}$$

The principle directions determine the new coordinate system. The centered

points are rotated via

$$R_i' = \begin{pmatrix} | & | & | \\ V_1' & V_2' & V_3' \\ | & | & | \end{pmatrix} \bar{R}_i, \quad i = 1, \ldots, N^{\text{atoms}}. \tag{4.8}$$

For the second method, we select a base geometry, e.g. the equilibrium geometry R_i^0. Then we rotate and translate the atoms so that the squared distance of the atom positions to the corresponding atoms in the base geometry is minimized, i.e. we find a *least squares* fit to a base geometry and solve

$$\min_{\mathbf{R},t} \sum_{i=1}^{N^{\text{atoms}}} \left\| \mathbf{R}R_i + t - R_i^0 \right\|^2. \tag{4.9}$$

For this optimization problem, there exists an analytical solution via calculating the singular value decomposition of a covariance matrix between the base geometry positions R_i^0 and the positions of the geometry to be fitted R_i [AHB87]. The normalized atom positions are then given by

$$R_i' = \mathbf{R}R_i + t, \quad i = 1, \ldots, N^{\text{atoms}}. \tag{4.10}$$

To make the approach more robust to the fluctuating hydrogen atoms, we only fit the positions of the heavy atoms ($Z_i > 1$). Note that this normalization method always matches according to atom indexing although switching the atom assignment might be more advantageous for the ML model, especially for symmetrical molecules.

In practical applications, the least squares approach works much better. The primary reason is that some molecules, like the highly symmetrical benzene, can have very different principle directions, as is shown in Fig. 4.3. It compares both the PCA and least squares approach. For the application to 3-D molecules we therefore choose the least squares approach in all examples.

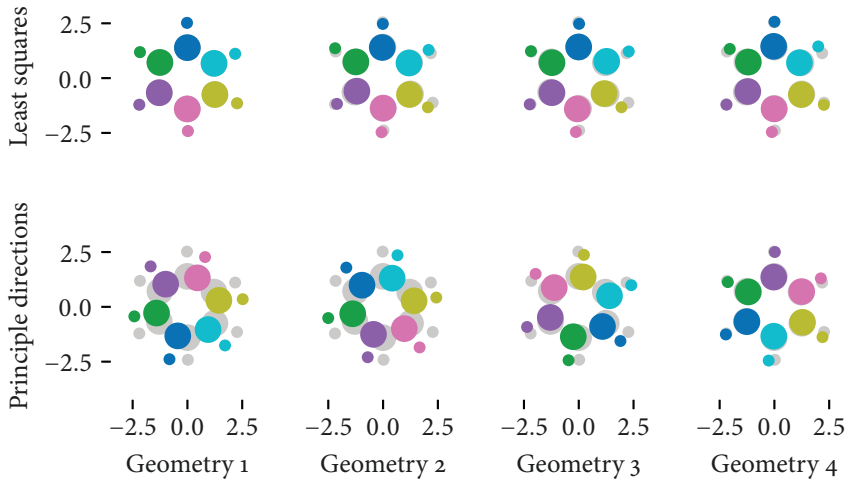


Figure 4.3: A comparison of normalization results on four example geometries of benzene. The top row shows the results of the least squares normalization approach, the bottom row shows the results of the principle directions normalization. For the least squares normalization the left-most first geometry is chosen as the base geometry. This geometry is shown in shaded grey for easy comparison in all plots.

4.2.2 Sampling

In the setting of simulating molecular dynamics we need to train a machine learning model on different geometries of the same molecule. How can we choose the training geometries? The training geometries should cover the complete space a molecule might reach during an MD simulation. However, MD is precisely the method to estimate this space. One strategy is to run an ab initio MD simulations, record the trajectory, train an ML model on the data, and continue the simulation by replacing the DFT part with the ML model. We can refer to this approach by *sampling via ab-inito MD*. The approach has the disadvantage that many expensive DFT calculations are necessary to cover the complete space.

A different suggestion is to use cheap classical MD. Classical MD is not guaranteed to cover the geometry space ab initio MD can cover and training on

classical MD trajectories can make the MD trajectory run into regions the ML model has not seen before. In Section 4.3.3 we discuss how to circumvent this via chemical knowledge and intuition. A different approach to circumventing this is active learning or running DFT whenever the ML model trained on classical MD trajectories is below a certain confidence threshold. We call the data points gathered from classical MD trajectories training point *proposals*. We can take a strategic selection of these proposal geometries and only calculate DFT energies and densities for this selection. This leads to far fewer DFT calculations than necessary for the ab initio MD sampling approach.

It should be pointed out that DFT calculations in the ab-inito MD approach run much faster than the DFT calculations in the classical MD sampling approach. This is due to the fact that ab-inito DFT codes take into account that the potential changes vary little from trajectory step to trajectory step and thus kick-start the DFT iteration with the data from the previous trajectory point. Nevertheless, the time spend on DFT calculations required for classical MD sampling approach remains significantly shorter.

It remains to be discussed how the training points from the training point proposal set can be selected. The motivation is to create a machine learning model without weak spots throughout the region a trajectory might cover. Without having trained any models and without labels, the idea is to select training points from the proposals so that every point is relatively close to at least one training point. We therefore strive to minimize the distance between proposals points and training points.

Let R^j be the normalized atomic positions of the proposed geometry j. We want to find training points $\tilde{R}^1, \ldots, \tilde{R}^M$ so that

$$\min_{\tilde{R}^1,\ldots,\tilde{R}^M} \sum_{i=1}^{M} \sum_{j \in \#_i} ||\tilde{R}^i - R^j||^2, \tag{4.11}$$

where $\#_i = \{j \mid i = \min_k \left\| R^j - \tilde{R}^k \right\|\}$ is the index set of all proposal positions that are closest to the training point $\tilde{R}_i$. The norm averages the euclidean distances over each atom position of the molecule. The optimization problem describes the K-means clustering algorithm [Ste56] for which fast and robust implemen-

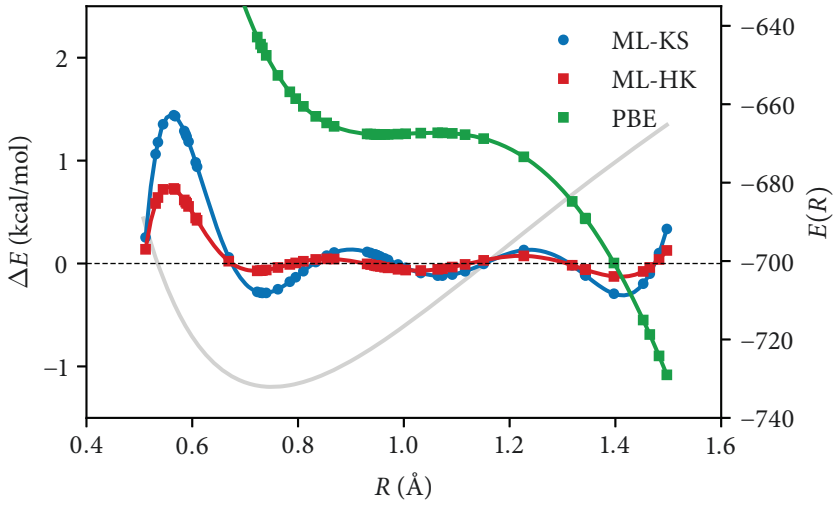


Figure 4.4: Comparison of ΔE for the ML-KS map (blue curve) and ML-HK map (red curve) on the H_2 dataset trained on $M = 7$ training points. The accuracy of PBE DFT compared to full configuration interaction (FCI) calculations is shown via the difference FCI – PBE in the green curve. The curves are interpolated through predictions on the test points. The grey curve shows the energy curve $E(R)$ for H_2. Its values correspond to the axis on the right-hand side.

tations exist [AV07]. To avoid the possibility of selecting chemically unreasonable geometries we do not select the $\tilde{R}$ directly as training points but choose for each $\tilde{R}^i$ the point from the proposed geometries R^j that is closest.

4.3 Application to 3-D molecules

Finally, we can show results of applying the ML-HK map to 3-D molecules. The applications are presented in increasing order of complexity where the final application covers simulating molecular dynamics.

4.3.1 H_2 molecule

The first example in 3-D is the Hydrogen gas molecule. It is the most simple molecule, consisting only of two Hydrogen atoms. Its only degree of freedom is the distance between the Hydrogen atoms. A data set of 150 geometries is created by varying the distance between the atoms R between 0.5 and 1.5 Å (sampled uniformly). A randomly chosen subset of 50 geometries is designated as the test set and is unseen by the ML model. These geometries are used to measure the out-of-sample error.

The energy curve $E(R)$ is shadowed in Fig. 4.4. It shows the behavior of the H_2 molecule around the PBE equilibrium bond length ($R_0 = 0.74\mathring{A}$). The total energy for the H_2 molecule grows quickly as the atoms get closer. As the Hydrogen atoms drift apart the energy curve grows slower and converges to double the energy of an isolated Hydrogen atom.

The remaining 100 geometries are the training point proposals. The geometries are normalized by centering the molecule and fixing the Hydrogen atoms to the x-axis. Subsets of varying sizes M are chosen as training points Because the required training subsets are so small, careful selection of a subset that covers the complete range of R is necessary. This is accomplished by selecting the M training points out of the training point proposals so that the distances R are nearly equally spaced as described in Sec. 4.2.2.

The performance of the ML-HK map is compared by evaluating the ML-KS map that maps from the Gaussian potential to the total energy and the combination of $n^{\mathrm{ML}}[v]$ that maps from Gaussian potential to the ground-state density in a 3D Fourier basis representation ($l = 25$) and $E^{\mathrm{ML}}[n]$ that maps from density to total energy.

The prediction errors are listed in Table 4.2. Both the mean average error (MAE) and the maximum error of the energy evaluated using the ML-HK map are significantly smaller than those of the ML-KS map. This indicates that even for a 3D system, learning the potential-density relationship via the HK map is much easier than directly learning the potential-energy relationship via the KS map.

Fig. 4.4 shows the errors made by the ML-KS map and the ML-HK map. The

		ML-KS		ML-HK			
		ΔE		ΔE		ΔE_D^{ML}	
	M	MAE	max	MAE	max	MAE	max
	5	1.3	4.3	0.18	0.54	0.70	2.9
H_2	7	0.37	1.4	0.054	0.16	0.17	0.73
	10	0.080	0.41	0.017	0.086	0.019	0.11
	10	0.27	0.94	0.099	0.60	0.12	0.39
H_2O	15	0.11	0.43	0.032	0.13	0.044	0.21
	20	0.015	0.064	0.011	0.058	0.0091	0.060

Table 4.2: Errors are shown for increasing numbers of training points M for the ML-KS and ML-HK approaches for both H_2 and H_2O. In addition, the estimated density-driven contribution to the error for the ML-HK approach (Eq. 3.30) is given. Energies are given in kcal/mol.

error of the ML-HK map is smoother than the ML-KS map error and is much smaller, even for the most problematic region when R is smaller than the equilibrium bond distance.

The mean average error (MAE) that is introduced by the PBE approximation on the H_2 dataset is 2.3 kcal/mol. This number is obtained by comparison to FCI calculations. FCI calculations are computationally significantly more expensive but give much more accurate energies. Here, the PBE errors are well above the errors of the ML model and verifies that the error introduced by the ML-HK map is negligible for a DFT calculation.

The H_2 molecule is of special interest to the DFT community because most exchange-correlation functionals cannot predict the disassociation behavior of the H atoms correctly [CMY08]. Ironically the most simplest molecule is challenging whereas the exchange-correlation functionals are mostly robust for far more complex molecules. Since ML fits the energy curve it does not suffer from errors of this nature.

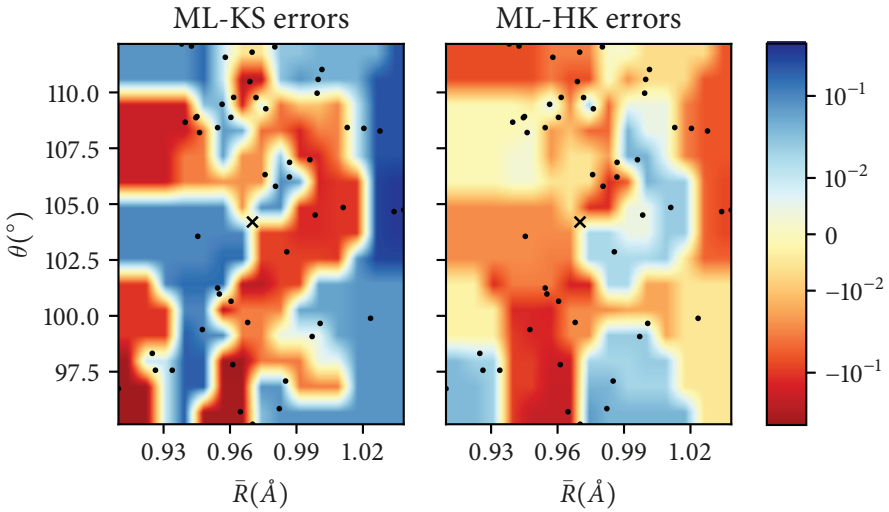


Figure 4.5: Distribution of energy errors on the H_2O data set for ML-KS and ML-HK. The errors are plotted on a symmetric log scale with a linear threshold of 0.01, using nearest neighbor interpolation from a grid scan for coloring. Black dots mark the equilibrium geometry. The bond lengths are averaged.

4.3.2 H_2O molecule

The next example is the water molecule H_2O, pictured in Fig. 4.6b. It is parametrized by its two bond lengths R and R' and its angle θ and thus has three degrees of freedom. To create a set of geometries, the PBE equilibrium geometry with $R_0 = R'_0 = 0.97\text{Å}$ and $\theta_0 = 104.2$ is varied by changing bond length and angles. 350 geometries are created by uniformly sampling the bond lengths by ±0.075Å and the angle by ±8.59 (±0.15rad) around the equilibrium geometry.

To normalize the geometries, the Oxygen atom is placed in the origin. The Hydrogen atoms are placed in the upper half of the x-y-plane so that the bond angle is mirrored on the y-axis and the longer bond is on the left side.

Again, 50 points are taking as the out-of-sample test set. The remaining 300 points are the training set proposal points. Just as for H_2, the training points are chosen as described in Sec. 4.2.2.

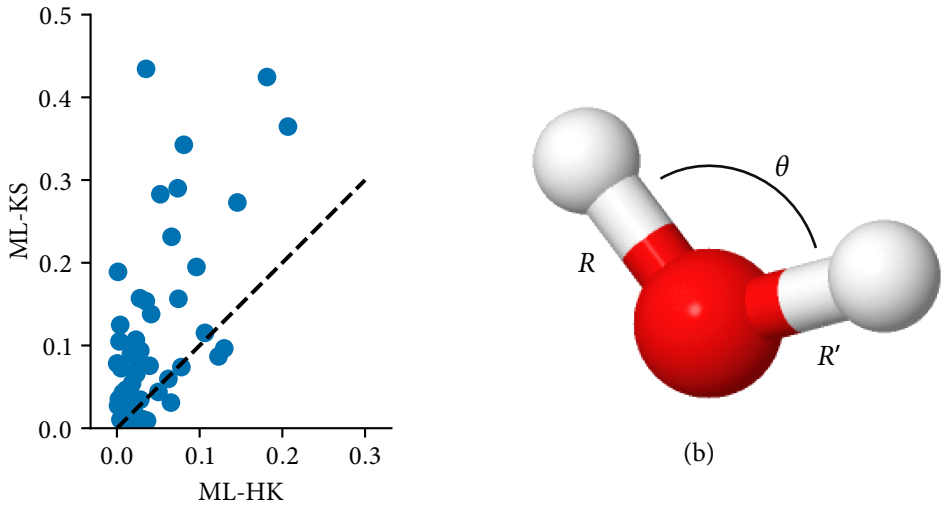


Figure 4.6: (a) Comparison of the absolute errors the ML-HK and ML-KS approaches make on the test set when training on 15 training points. (b) The parametrization of H_2O.

The results of the experiments are given in Table 4.2. As expected, the increase in degrees of freedom for H_2O compared to H_2 requires a larger training set size M. However, even for the more complicated molecule, the ML-HK map is consistently more precise than the ML-KS map (Fig. 4.6a) and provides an improved potential energy surface, as shown in Fig. 4.5. With an MAE of 1.2 kcal/mol for PBE energies relative to CCSD(T) calculations (similar to FCI a more expensive but more accurate method than PBE KS-DFT) for this data set, we again show that the ML model does not introduce a new significant source of error.

4.3.3 Ethane, benzene, and malonaldehyde

As a next step, we turn to molecules that have significantly more degrees of freedom: ethane with chemical formula C_2H_6 (Fig. 4.7), benzene with chemical formula C_6H_6 (Fig. 4.8), and malonaldehyde with chemical formula $C_3H_4O_2$ (Fig. 4.9).

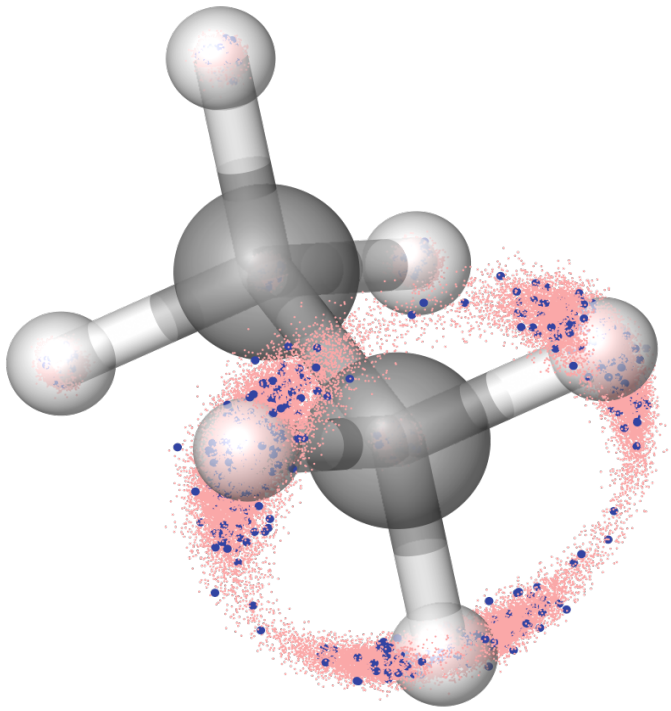


Figure 4.7: A visualization of the 300 K + 350 K ethane dataset. Red points mark the atom positions of the training set proposal points as selected by the k-means approach. Blue points mark the atom positions of the points in the test set, taken from an independent MD trajectory. To better visualize the dataset, the molecule was aligned to the carbon atoms and the Hydrogen atoms in the background.

Ethane has in total eight atoms and thus serves as a first more complex molecule to test the proposed methodology on. It has a small energy barrier for the relative rotation of the methyl groups. Benzene has a highly symmetrical ring structure and thus serves as a challenge for the normalization procedure. Malonaldehyde has a Hydrogen atom that jumps between the Oxygen atoms (referred to as an intramolecular proton transfer). This proton transfer does not occur when simulating via classical MD and is therefore not part of the training set. The malonaldehyde molecule thus serves as an extrapolation challenge for the ML models.

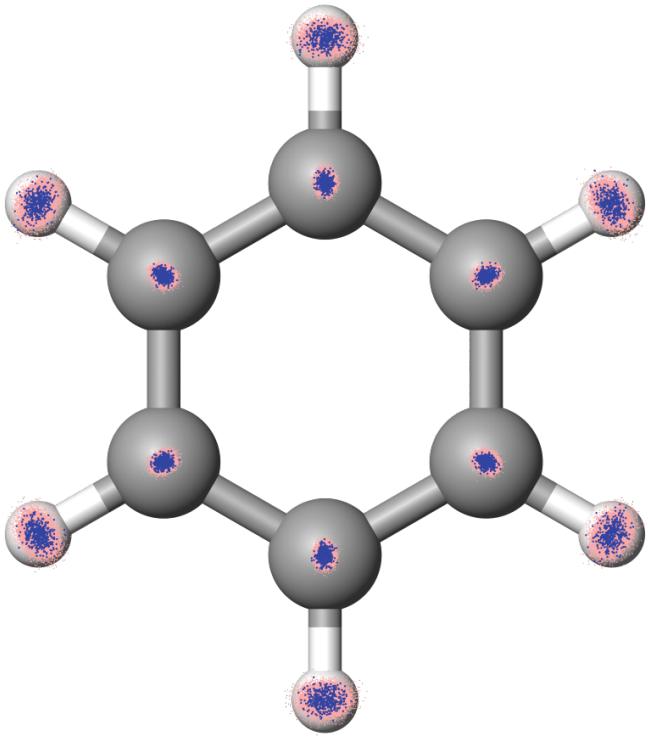


Figure 4.8: A visualization of the 300 K + 350 K benzene dataset. Red points mark the atom positions of the training set proposal points as selected by the k-means approach. Blue points mark the atom positions of the points in the test set, taken from an independent MD trajectory.

The increased degrees of freedom make it impractical to parameterize and to sample the parameters to generate a dataset of realistic molecular geometries. It is therefore necessary to establish a new sampling strategy. To sample these datasets we take the approach of running cheap classical MD simulations as described in Sec. 4.2.2. We run simulations at different temperatures. Although we want to predict data points from an MD trajectory run at 300 K (room temperature), we generate proposals by running classical isothermal MD not only at 300 K, but also at 350 K and 400 K temperature for ethane and benzene. The MD simulations are run using the General Amber Force Field (GAFF) [Wan+04] in the PINY_MD package [Tuc+00]. For malonaldehyde we run trajectories at

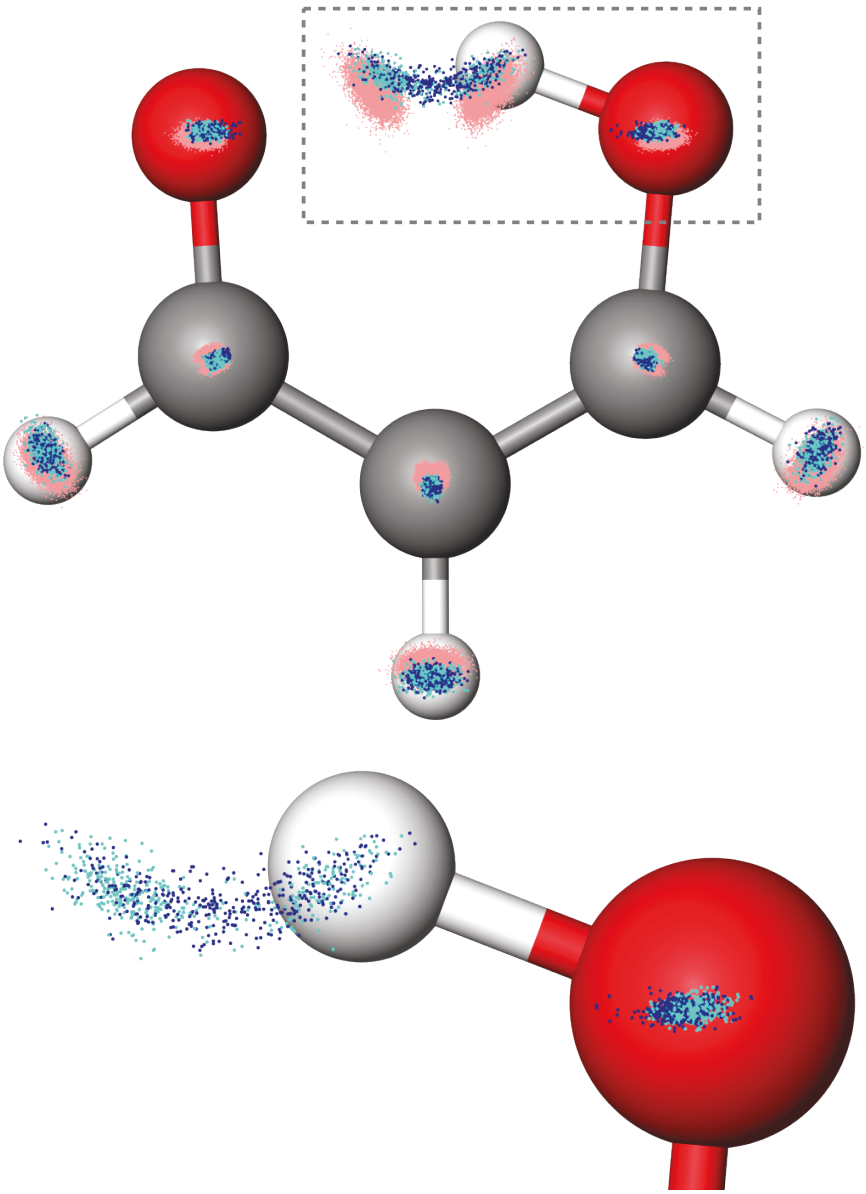


Figure 4.9: **Top.** The distribution of malonaldehyde atom positions. Red points mark the atom positions of the training set proposal points as selected by the k-means approach. Turquoise points mark the atom positions of the points in the test set, taken from an independent ab initio MD trajectory. Blue points mark the atom positions sampled by the ML-HK model.
Bottom. A closer view of the region outlined with a dashed box for the ab initio (turquoise) and ML-HK (blue) trajectories.

300 K and 350 K temperature.

This has two effects. First, an ab initio trajectory for 300 K might cover areas that an classical trajectory for 300 K never reaches. Sampling with higher temperatures (here: 350 K and 400 K) makes the molecule move in a wider range and is thus more likely to cover the complete area of an ab initio trajectory. Secondly, MD trajectories stay mostly in low energy areas and rarely reach areas that require higher energies. Increasing the temperature of the simulation make the trajectory run through higher energy areas more often and thus sample these regions more completely. We later analyze in how far data points from higher temperature simulations help the machine learning model.

We combine geometries of one 300 K trajectory and one higher temperature trajectory (each 1 ns) in order to generate more sets of training set proposal points. From these proposals we select 2000 training points via the k-means approach as described in Sec. 4.2.2. We thus have, for each ethane and benzene, three different datasets for training: 300 K, 300 K + 350 K, and 300 K + 400 K. For malonaldehyde we combine two sets of 300 K and 350 K simulations, one for each tautomer (the proton being attached to either of the two oxygens).

The number of 2000 training points is chosen because it leads to a data set that is still comfortably manageable with the kernel models but should also cover the sampling area well. This will allow us to evaluate whether the sampling, normalization, and training approach works on molecules with more degrees of freedom. Evaluating different numbers of training points here would require a different selection of the proposal points and thus more expensive DFT calculations.

The test sets should be independently distributed from the training set. It is therefore necessary that the test set geometries stem from an independent MD trajectory. For ethane and benzene, we simulate independent trajectories via classical MD and select 200 snapshots following a random initial timestamp as test points. The overlay of training and test data is visualized for ethane and benzene in its Figs. 4.7 and 4.8 correspondingly. For malonaldehyde, the most complex molecule, we choose to simulate a more realistic, but also computationally more expensive, 0.25 ns ab initio MD trajectory at 300 K for the test set. Fig. 4.9 shows how the sampling from the classical trajectory differs from the

			ΔE		$\Delta E_{\mathrm{D}}^{\mathrm{ML}}$	
Molecule	Training traj.	Test trajectory	MAE	max	MAE	max
Benzene	300 K	classical 300 K	0.42	1.7	0.32	1.5
	300 K + 350 K		0.37	1.8	0.28	1.5
	300 K + 400 K		0.47	2.3	0.30	1.8
Ethane	300 K	classical 300 K	0.20	1.5	0.17	1.3
	300 K + 350 K		0.23	1.4	0.19	1.1
	300 K + 400 K		0.14	1.7	0.098	0.62
Malonaldehyde	300 K + 350 K	ab initio 300 K	0.27	1.2	0.21	0.74

Table 4.3: The errors (in kcal/mol) of the ML-HK models for the MD datasets on their test set trajectories.

points of the ab initio trajectory.

The results for evaluations on the test set trajectories are given in Table 4.3 and are visualized for ethane and benzene in Fig. 4.10. The ML error is reduced by creating the training set from trajectories at both the target temperature and a higher temperature to increase the representation of more distorted geometries. The MAE of the ML-HK map for the benzene data set using training geometries from 300 K and 350 K trajectories is only 0.37 kcal/mol for an energy range that spans more than 10 kcal/mol (Table 4.3). For ethane the ML-HK model reproduces the energy of geometries with a MAE of 0.23 kcal/mol for an independent MD trajectory at 300 K (Fig. 4.10). This test set includes conformers from the sparsely-sampled eclipsed configuration (Fig. 4.7). Using points from a 400 K trajectory improves the ML-HK map due to the increased probability of higher energy rotamers in the training set (Table 4.3). Generating appropriate geometries for training via computationally cheap classical MD thus significantly decreases the cost of the ML-HK approach.

The ab-initio MD test trajectory of malonaldehyde includes a proton transfer event, i.e. points that would not be sampled in the classical MD training trajec-

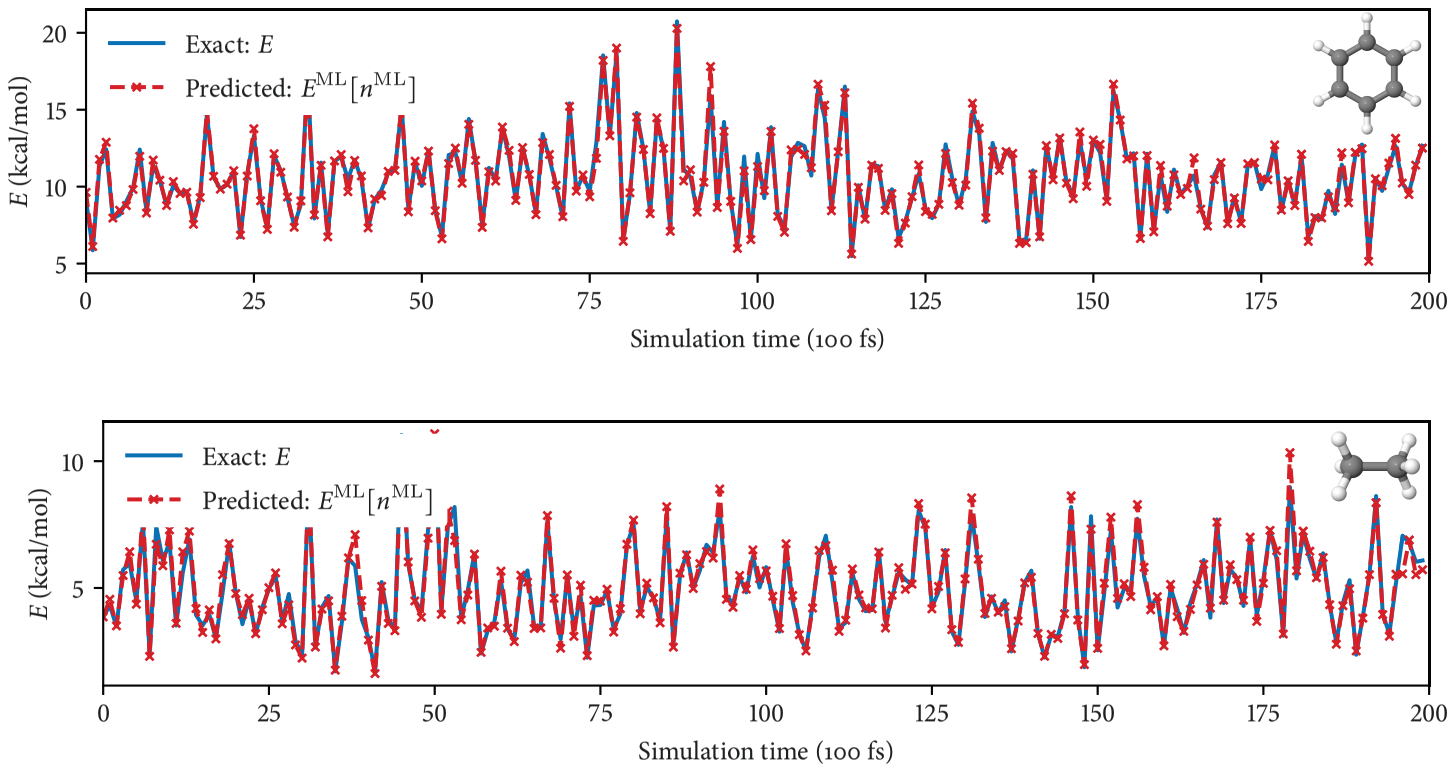


Figure 4.10: Energy errors of ML-HK along classical MD trajectories. PBE values in blue, ML-HK values in red. **Top.** A 20 ps classical trajectory of benzene. **Bottom.** A 20 ps classical trajectory of ethane.

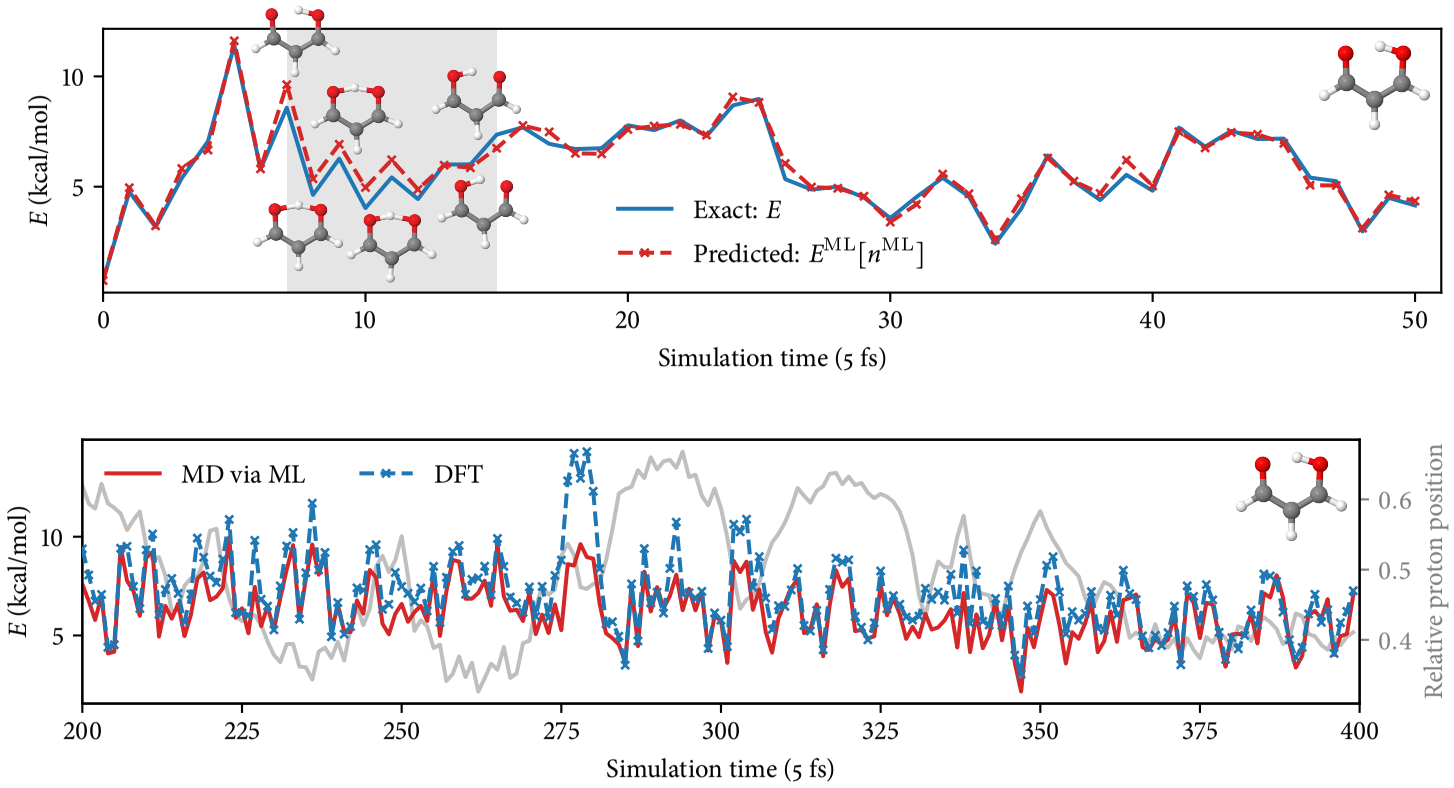


Figure 4.11: **Top.** Energy errors of ML-HK along a 0.25 ps ab initio MD trajectory of malonaldehyde. PBE values in blue, ML-HK values in red. The ML model correctly predicts energies during proton transfer in frames 7–-15 without explicit inclusion of these geometries in the training set.
Bottom. Energy errors of ML-HK along a 1 ps MD trajectory of malonaldehyde generated by the ML-HK model. ML-HK values in red, PBE values of trajectory snapshots in blue. The shaded grey curve shows the relative proton position.

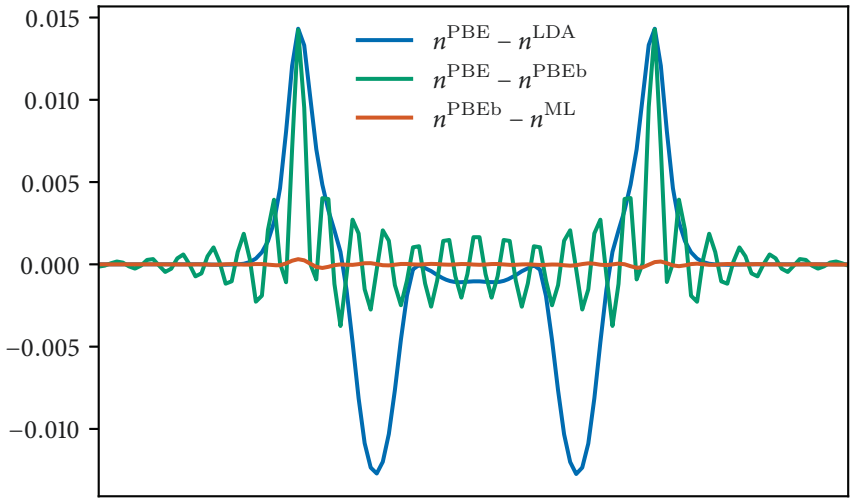


Figure 4.12: The effect of basis representations and theory levels along a 1-D cut through the valence density of benzene. The blue curve shows the difference of the valence density for DFT calculations with PBE and LDA exchange correlation functionals. The green line shows the effect of the Fourier basis representation with $l = 25$. The orange curve shows the error the n^{ML} model makes.

tories. Fig. 4.11 shows that the ML-HK map is able to predict DFT energies during a proton transfer event despite being trained on classical geometries that did not include these intermediate points. For the whole trajectory the ML models reach an MAE of 0.27 kcal/mol (Table 4.3).

4.3.4 Fourier basis function accuracy

The number of Fourier basis functions used for the density representation has been set to $l = 25$ in each direction. Evaluating the influence of the number of basis functions on the ML error would require the generation of additional datasets. We therefore evaluate the accuracy of the basis set representation by comparing it to (1) the difference in the density function resulting from different levels of physical theory (PBE vs. LDA) and (2) by comparing the accuracy of the basis set representation to the accuracy of the ML model n^{ML}.

An example of a benzene density predicted by the n^{ML} model is given in Fig. 4.12. It is clear that the errors introduced by the Fourier basis representation are larger than the errors introduced by the ML-HK map by two orders of magnitude. Furthermore, the ML-HK errors in density (as evaluated on a grid in the molecular plane of benzene) are also considerably smaller than the difference in density between density functionals (PBE vs. LDA). This result verifies that the ML-HK map is specific to the density used to train the model and should be able to differentiate between densities generated with other electronic structure approaches.

The basis representation introduces errors of the same magnitude as the difference in theory levels. The final energy predictions are unaffected by this because the E^{ML} map learns directly from the the densities affected by the basis representation. However, depending on how the densities should be used, a larger number of basis functions or a basis set that is more specifically tailored to the molecule might be necessary.

4.4 Molecular dynamics with machine learning models

As a final step in the evaluation of the ML-HK approach, we run an MD simulation with the ML model, i.e. an MD simulations without running any DFT calculations.

Sec. 1.2.2 explains how MD simulations work in general. To run ab initio quality MD simulations without running DFT calculations, we have to obtain forces from the ML model. So far, we can obtain densities (via n^{ML}) and energies (via E^{ML}) from the ML-HK approach. To obtain the forces, we need a way to calculate the gradients of the total energy $E^{\mathrm{ML}}[n^{\mathrm{ML}}]$ with respect to the atom positions.

There are two possibilities. First, the forces could be predicted by a specifically trained forces model. This approach promises a fast and reliable approach to force prediction and is popular when learning PES models [Puk+09; Chm+17]. The forces, however, are tied to each atom of the molecule which makes it necessary to keep track of atom indexing. By predicting the density $n(\boldsymbol{r})$ in the n^{ML}

map, the prediction result becomes invariant to atom indexing and thus make this idea impractical for the ML-HK approach.

The second possibility is to derive the forces from energy predictions by calculating the gradients numerically, i.e. via finite differences. This approach, however, tends to be slower due to the high number of necessary $E^{\mathrm{ML}}[n^{\mathrm{ML}}]$ evaluations. In Sec. A we will discuss a possibility to obtain forces fast by learning densities in specific basis function representations that make analytical force calculations possible.

The force acting upon atom i with position R_i is given by the negative gradient

$$F_i = -\nabla_{R_i} E^{\mathrm{ML}}[n^{\mathrm{ML}}[v_R]] \in \mathbb{R}^3 \tag{4.12}$$

where the Gaussians potential v is given as in Eq. 3.12. We compute the gradient via central finite differences, i.e.

$$\delta_h(R) = E^{\mathrm{ML}}[n^{\mathrm{ML}}[v_{R+\frac{1}{2}h}]] - E^{\mathrm{ML}}[n^{\mathrm{ML}}[v_{R-\frac{1}{2}h}]]. \tag{4.13}$$

The forces are then computed via

$$F_i = -\frac{1}{\epsilon}\Big(\delta_{(\epsilon,0,0)^\top}(R_i), \delta_{(0,\epsilon,0)^\top}(R_i), \delta_{(0,0,\epsilon)^\top}(R_i)\Big)^\top. \tag{4.14}$$

The evaluation of the forces thus requires six energy evaluations per atom. The finite difference distance parameter ϵ has to be chosen carefully. A smaller value yields more accurate gradients in general, however, a larger ϵ smooths out the ML model's slightly inaccurate energy manifold and yield more accurate gradients in this case. We can identify a good value for ϵ by either comparing computed forces to exact ones from the reference calculations or by starting with a large ϵ and making it smaller until the force values become unstable. Here, we find a value of 0.001 Å to yield accurate force predictions.

It is important to remember that the atom positions have been normalized for training and ML model evaluation. To get accurate results, this normalization procedure has to be built in to the force calculation procedure as well. Both the MD simulation and the finite difference displacement change the atom positions and require a normalization step.

Data: Atomic positions $R_1, \ldots, R_{N^{\text{atoms}}} \in \mathbb{R}^3$, Models n^{ML} and E^{ML}, Gaussians potential function V_R, Normalization function $\text{normalize} : R \mapsto R$, Finite difference distance parameter ϵ
Result: Forces for each atom: $F_1, \ldots, F_{N^{\text{atoms}}}$

for $i = 1, \ldots, N^{\text{atoms}}$ **do**
  for $d = 1, 2, 3$ **do**
    $R_i^{\text{l}} \leftarrow R_i - \frac{\epsilon}{2} e_i$ (e_i is i-th unit vector)
    $R_i^{\text{r}} \leftarrow R_i + \frac{\epsilon}{2} e_i$
    $R_i^{\text{l}} \leftarrow \text{normalize}(R_i^{\text{l}})$
    $R_i^{\text{r}} \leftarrow \text{normalize}(R_i^{\text{r}})$
    $v^{\text{l}} \leftarrow V_{(R_1, \ldots, R_i^{\text{l}}, \ldots R_{N^{\text{atoms}}})}$
    $v^{\text{r}} \leftarrow V_{(R_1, \ldots, R_i^{\text{r}}, \ldots R_{N^{\text{atoms}}})}$
    $f_d \leftarrow -\frac{1}{\epsilon} \left(E^{\text{ML}}[n^{\text{ML}}[v^l]] + E^{\text{ML}}[n^{\text{ML}}[v^r]] \right)$
  end
  $F_i \leftarrow (f_1, f_2, f_3)^\top$
end

Algorithm 2: Force calculation for MD with ML via central finite differences.

The complete algorithm for calculating forces from the n^{ML} and E^{ML} models is summarized in Algorithm 2.

Since the simulation should run at a fixed temperature of 300 K, it is necessary to use a thermostat. Here, we use the Langevin thermostat as implemented in the Atomic Simulation Environment (ASE) [Hjo+17]. To maintain temperature it modifies Newton's second law of motion by adding friction and a random force. The amount of friction is controlled by a parameter γ. We want to set the parameter so that the trajectory conserves energy well (see below), in this case the value is set to 0.413 fs^{-1} (0.01 atomic units). We initialize the velocities by drawing randomly from a Maxwell-Boltzman distribution.

The energy curve of the MD simulation that is generated by forces calculated from the ML model is shown in Fig. 4.11 (bottom). No DFT calculation was run to obtain this trajectory.

We now evaluate the quality of the simulation in three different ways. First, we can take the geometry snapshots of the MD trajectory generated by ML and compute the total energies via DFT. Fig. 4.11 shows the comparison between the

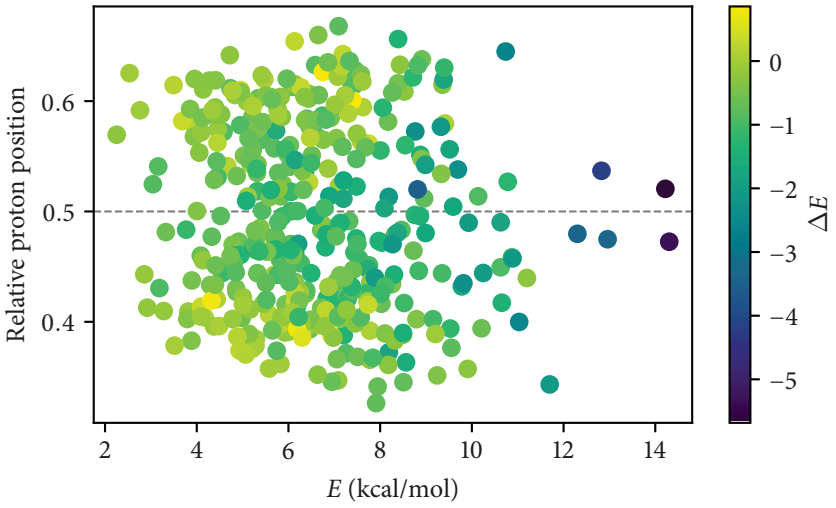


Figure 4.13: Total energy error distribution from ML-HK generated trajectory snapshots of malonaldehyde. Plotted is the total energy vs. the relative proton position. The geometries are colored by the energy error ΔE. High energy geometries where the proton is centered are underestimated.

energies from the ML model and the energies from snapshot DFT calculations. It also shows the relative proton position between the two oxygen atoms, e.g. it is zero when the proton overlaps with the first oxygen atom and one when it overlaps with the second oxygen atom. The ML model underestimates the total energy when the proton moves into the transfer region where the model has no training examples. This explains the high frequency of proton transfer events in the ML generated MD trajectory. A more detailed view is given in Fig. 4.13: The ML model underestimates the energy inside the transfer region, especially for high energy geometries. However, the calculated forces are sufficiently large to bring the atoms back toward their equilibrium positions, resulting in a stable molecular trajectory. The Fig. 4.9 shows the distribution of the atom positions from the trajectory generated by ML compared to the ab initio MD trajectory and the classical MD trajectories used as training set proposals. A numerical comparison is given in Table 4.4.

Fig.4.14 shows the conservation of energy for the ML generated MD trajectory

	Atom type			
MD trajectory	C	O	H(–CH)	H(–OH)
DFT	0.052	0.076	0.166	0.289
ML-HK	0.051	0.095	0.171	0.242

Table 4.4: Difference between DFT and ML-HK sampling of malonaldehyde configurations during a 2 ps MD simulation. The values are root mean squared deviations (Å) from the optimized geometries (the average coordinates of the two optimized enol tautomers).

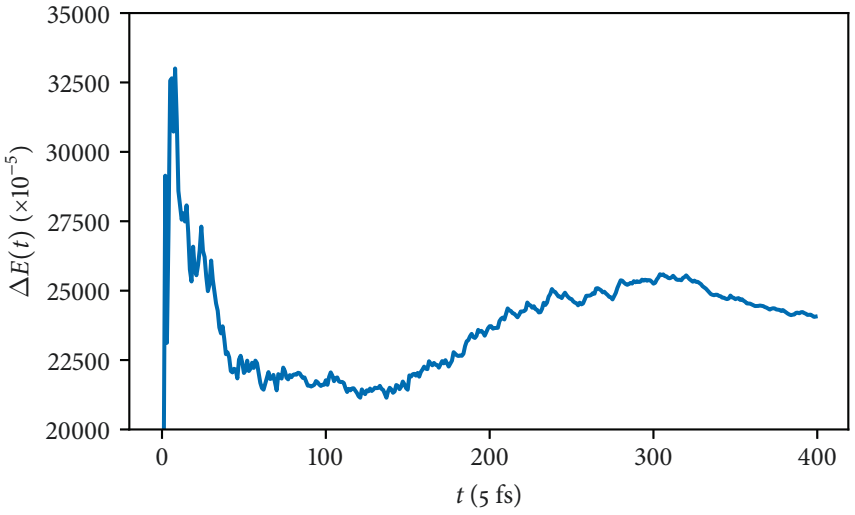


Figure 4.14: The energy conservation measure $\Delta E(t)$ as defined in Eq. 4.15 for the malonaldehyde trajectory generated by ML-HK.

[Tuc10]. Here, $E(t)$ is the energy at time step t and $\Delta E(t)$ is defined as

$$\Delta E(t) = \frac{1}{t} \sum_{t'=0}^{t} \left| \frac{E(t') - E(0)}{E(0)} \right|. \tag{4.15}$$

The trajectory conserves energy reasonably well over the 2 ps trajectory.

4.5 Discussion

Application of the ML-HK map showed that the approach works in 1-D and 3-D. For 1-D it outperforms previous gradient descent based approaches. It is robust enough for complex applications. The approach can be applied without much adaptation to 1-D toy problems, smaller and larger molecules, and even MD simulations.

The chained model is robust enough to yield gradients that are accurate enough for MD simulations. Here, calculating the energy gradients (forces) via finite differences requires a trade-off when choosing the distance parameter ϵ. More interesting is the outlook that the prediction of density matrices gives us in Appendix A. There, the approach of predicting densities generalizes to the prediction of density matrices. Once density matrices are predicted, it is possible to calculate the energy gradients analytically and thus avoid the finite differences.

The robustness of the chained model in regions where no training data is available, i.e. in the region of the proton transfer, is interesting. The models based on the Gaussian kernel flatten the predicted energy manifold and thus lead the molecular dynamics trajectory into this area. However, the energies' trend is predicted well enough to also lead the trajectory out and back into known territory. This behavior can be particularly useful when the ML aided simulation of molecular dynamics is applied in combination with active learning.

The MD simulation then leads to exploratory predictions, which in machine learning are a classical application for active learning. It describes how the ML-HK approach with the classical MD sampling scheme can work in practice. In this setting the model would query a DFT code whenever it is not confident in its own prediction and retrain itself with this new data. This allows relatively

cheap exploration of rare events. The time-consuming part of predicting geometries from the often visited regions is then expedited by machine learning.

A limit to the approach will be the size of the molecule. Increasing the number of atoms in the molecule will make it necessary to increase the number of Fourier basis functions for the density. At some point this number reaches the computational complexity limit, for example, when the basis coefficients can not fit into memory. A possibility to circumvent this problem is given by utilizing atom contributions. Atom contributions to the density are imaginable when atom-centered basis functions are used. The first step in this directions are density matrices and the proof of concept described in Appendix A.

5

Conclusion

The application of machine learning to density functional theory had just started to gain interest and first successful attempts had been made on 1-D systems when this research began. The purpose of the thesis was to adapt these approaches to 3-D systems. This has been successful with the machine-learning-Hohenberg-Kohn map. A new framework has been introduced that outperforms the previous work in 1-D and also analyzes how and where the errors are made that contribute to this result.

In the course of applying the methods to 3-D many tangential problems have been solved. A basis set had to be selected that works with electronic structure codes and allows efficient application of the machine learning training and inference routines. The molecule geometries had to be normalized in a robust way. To work with smooth densities, the right pseudo-potentials had to be selected. To generate training data for the MD parts, a sampling routine based on classical dynamics had to be designed.

This thesis also sets a solid theoretical foundation for the machine-learning-Hohenberg-Kohn map by embedding it into the framework of functional predictions with operator-valued kernels and the use of basis function representations. An efficient multivariate KRR training routine has been implemented.

The new approach not only works in 3-D but is also robust enough to allow application to the simulation of molecular dynamics. It was even possible to

simulate chemical phenomena that were not included in the training set. The scope of these results go far beyond what was initially thought possible. Where the original draft of the publication Brockherde et al. [Bro+17] only included application to 1-D systems, a 2-year long series of internal discussions, peer reviews, and addition of more authors, made it grow to not only contain application to 3-D system but also an application to MD simulations. The American Chemical Society's *Chemical & Engineering News* now lists it among the most notable chemistry "research of the year" 2017 [Jac17].

Future research that builds upon the results presented here is already underway. Part of this is the extension to density matrices where preliminary results are presented in the Appendix A. Other work involves predicting CCSD(T) energies from DFT densities.

A

Integrated prediction of density matrices

A.1 Motivation

The thesis results show that learning electron densities promise good excellent results for prediction of DFT energies with the approach even being able to simulate molecular dynamics. However, the method of predicting electron densities via the HK map requires a total energy map E^{ML} and does not give direct access to forces. Moreover it does not preserve the physics of DFT and does not make use of easily computable parts that are either known or for which good approximations exist, for example the Hartree and exchange-correlation potentials. Making use of these parts of DFT would make the predictions more robust and would allow us to easily calculate other quantum chemical quantities directly without having to learn separate machine learning models like E^{ML}.

To demonstrate the idea of integrating density predictions as a proof of concept, this part of the thesis follows the approach of combining self-consistent field (SCF) calculations and density predictions by avoiding the computationally dominant Roothaan-Hall diagonalization in KS-DFT and introducing a machine learning model that predicts *density matrices* from non-interacting Hamiltonians in basis set representation (i.e. Fock matrices).

Density matrices (which are defined in the following section) contain significantly more information than electron densities. The benefit of density matrices is that they make use of specifically tailored *atom-centered* basis functions

that are physically motivated. This leads to the disadvantage that the number of basis functions grows with the number of atoms and orbitals and the approach is thus not orbital-free. A possibility to avoid the scaling problem that this introduces is discussed later on.

A.2 Density matrices

The basics of DFT where introduced in Sec. 1.2.1. It was discussed that Kohn-Sham DFT reduces the intractable many-body problem of interaction electrons in a static potential to an auxiliary problem of non-interacting electrons in an effective potential.

The orbitals of the non-interacting Schrödinger equation were given in Eq. 1.10 and the KS potential in Eq. 1.15. The non-interacting system is thus described by the Kohn-Sham equations

$$\left[T_s + v_{\mathrm{H}}(\boldsymbol{r}) + v_{\mathrm{XC}}(\boldsymbol{r}) + v(\boldsymbol{r})\right]\varphi_\alpha(\boldsymbol{r}) = \epsilon_\alpha \varphi_\alpha(\boldsymbol{r}). \tag{A.1}$$

We expand the orbitals $\varphi_\alpha(\boldsymbol{r})$ over an atom-centered Gaussian-type orbital (GTO) basis set χ, i.e.: The GTO basis set χ was already discussed in Sec. 2.1.4. It allows us to expand the orbitals φ_α via

$$\varphi_\alpha(\boldsymbol{r}) = \sum_{p=1}^{Q} C_{\alpha p}\chi_p(\boldsymbol{r}). \tag{A.2}$$

This basis set allows us to reformulate Eq. A.1 as a generalized eigenvalue problem by multiplying on the left by $\chi_q^*(\boldsymbol{r})$ and integrating over $\boldsymbol{r}$. We then have the Roothaan-Hall equation

$$FC = \Lambda SC \tag{A.3}$$

which we have to solve for the *molecular orbital (MO)* coefficients C. Due to KS-DFT's historic similarity to Hartree-Fock methods, the matrix

$$F_{qp} = \int \chi_q^*(\boldsymbol{r})\left[T_s + v_{\mathrm{H}}(\boldsymbol{r}) + v_{\mathrm{XC}}(\boldsymbol{r}) + v(\boldsymbol{r})\right]\chi_p(\boldsymbol{r})\mathrm{d}\boldsymbol{r}. \tag{A.4}$$

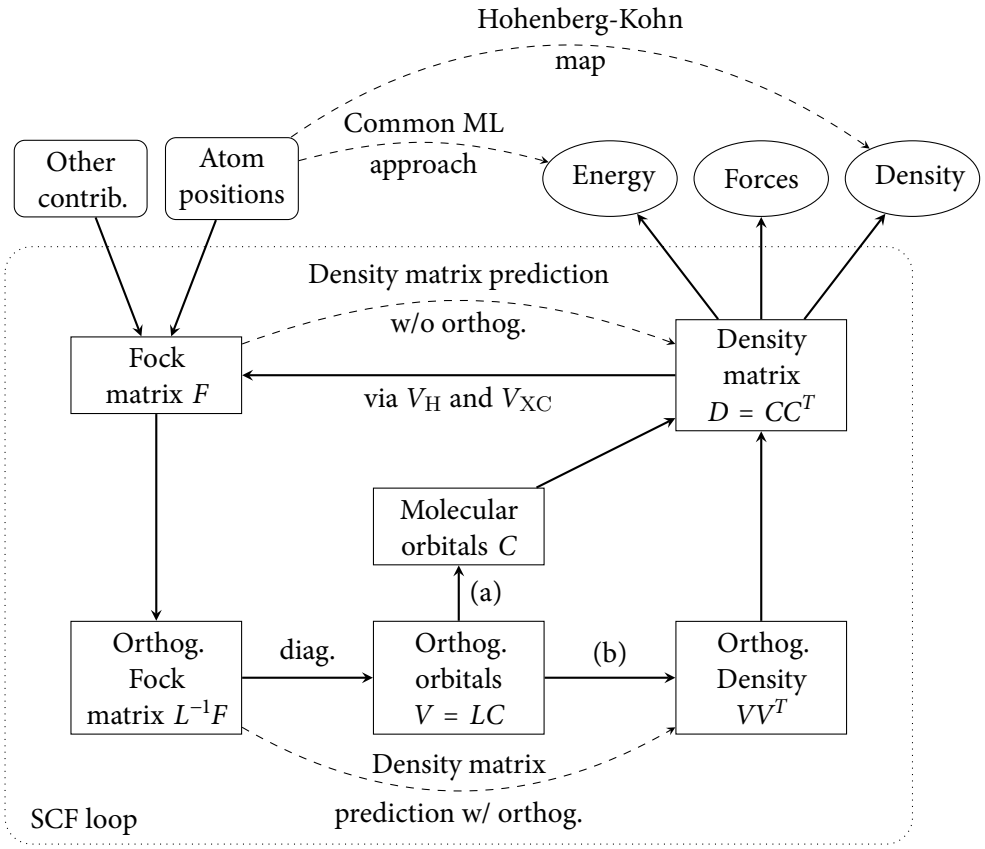


Figure A.1: Position of the approach in Kohn-Sham density functional theory and its self-consistent field loop. Dashed lines are machine learning models. DFT codes usually take route (a) to have access to the molecular orbitals. Here, we take route (b) when predicting with orthogonalization. For predicting without orthogonalization we avoid the lower orthogonalization half entirely. Note that common ML approaches that learn the potential energy map can neither factor in other external contributions nor have they access to other properties like analytic energy gradients (forces). The main part of the thesis (the approach via the Hohenberg-Kohn map) discusses predicting the density n directly.

is referred to as the *Fock matrix*. The *overlap matrix S* accounts for the non-orthogonality of the basis set and is given by

$$S_{qp} = \int \chi_q^*(\boldsymbol{r})\chi_p(\boldsymbol{r})\mathrm{d}\boldsymbol{r}. \tag{A.5}$$

Solving Eq. A.3 is not straight forward: The Hartree potential v_H and the exchange correlation functional v_XC depend on the *density matrix* which itself depends on the molecular orbital coefficients. We define the density matrix D implicitly via the density n of the system:

$$n(\boldsymbol{r}) = \sum_{\alpha=1}^{N} \varphi_\alpha^*(\boldsymbol{r})\varphi_\alpha(\boldsymbol{r}) = \sum_{pq}\sum_{\alpha=1}^{N} C_{\alpha q}^* C_{\alpha p}\chi_q^*(\boldsymbol{r})\chi_p(\boldsymbol{r}) = \sum_{pq} D_{pq}\chi_q^*(\boldsymbol{r})\chi_p(\boldsymbol{r}).$$

To solve the eigenvalue problem, the self-consistent field (SCF) method iterates over

1. building the Fock matrix and
2. diagonalizing it to find the density

until convergence. There are two parts that dominate the computational complexity: The evaluation of the two-electron integrals that are necessary for the Hartree potential v_H in part (1) and the diagonalization of the Fock matrix in part (2). The naive evaluation of the two-electron integrals has $\mathcal{O}(Q^4)$ complexity, but usually only a few two-electron integrals are significantly larger than zero. It is thus possible and common to find these non-zero integrals by bounding the integrals via Cauchy-Schwartz inequality. This only requires the computation of Q^2 integrals plus the only two-electron integrals that will be significantly larger than zero.

This leaves the diagonalization as the only remaining bottleneck of the DFT calculation. It is possible to lessen the computational burden by using power iteration to diagonalize F because only the lowest N (occupied) eigenvalue vectors are used to compute the density. However, the number of necessary eigenvectors is still substantial so that the diagonalization step remains the computationally dominant part of the iteration.

In this approach we want to avoid the diagonalization completely by replacing

step (2) with a machine learning model. Eq. A.3 is usually solved by orthogonalizing the basis functions first. This removes the overlap and leads to a regular eigenvalue problem. The basis functions are orthogonalized as $V = L^T C$ by finding a square root L of the overlap matrix $S = LL^T$. We can then rewrite Eq. A.3 as

$$L^{-1}FL^{-T}V = \Lambda V. \tag{A.6}$$

The density matrix is then given by $D = L^{-T}VV^TL^{-1}$.

There are two common choices for the square root L. One, known as *symmetric orthogonalization*, is to use the Cholesky decomposition since the overlap matrix S should be positive definite. However, this might not always be the case which is why another choice is often preferred: *canonical orthogonalization*. This approach uses an eigen-decomposition of the overlap matrix and incorporates the eigenvalues bigger than zero into the eigenvector matrices to yield a square root of S.

A.3 Machine learning model for density matrix prediction

This section explains how ML can be used to skip the diagonalization in Kohn-Sham DFT. Fig. A.1 contains an overview of Kohn-Sham DFT and clarifies the ML approaches. All density matrix prediction approaches predict a density matrix given a Fock matrix. The differences are in how the Fock matrices are orthogonalized. The different orthogonalization possibilities lead to three possible machine learning models:

No orthogonalization We ignore the overlap matrix and predict $F \longrightarrow D$. This approach has the advantage that we do not have to find a square root of the overlap matrix which is computationally expensive. However, the map $F \longrightarrow D$ might be ill-defined. Without knowledge of S there might be many possible density matrices that fit one Fock matrix. We could clumsily add the overlap matrix to the ML model (i.e. by adding it as extra features) but this turned out to be unnecessary in practice.

Symmetric orthogonalization Here, we predict $L^{-1}FL^{-T} \longrightarrow VV^T$ and then compute D. This is a clean solution because the map is well-defined. The

disadvantages are a possibly not positive definite overlap matrix and the cost of the Cholesky decomposition.

Canonical orthogonalization We predict $L^{-1}FL^{-T} \longrightarrow VV^T$ just as above but use canonical orthogonalization to find L. This seems to be the best solution. The only disadvantage is the cost of the eigen-decomposition of S but this has to be done only once for a DFT calculation since S does not change during the SCF iteration.

The application of all three models require the SCF loop. To collect training data, we have to do full-featured DFT calculations and collect the (F, D) or $(L^{-1}FL^{-T}, VV^T)$ pairs in each iteration. (In the following, we refer to all pairs as (F, D) regardless of the orthogonalization approach.)

For the two ML approaches with orthogonalization, there are two notable deviation from common KS-DFT implementations. First, we avoid the computation of the molecular orbital coefficients $C = L^{-1}V$ because we need the orthogonalized density matrix VV^T instead. We discuss the implications later. The second deviation is the orthogonalization after the first SCF iteration. Some codes use the molecular orbitals C of the previous iteration to orthogonalize F. We instead use the initial orthogonal basis L (i.e. the square root of S) for all SCF iterations.

The most successful ML total energy prediction approaches are based on predicting atom-environment contributions as in

$$E_{\text{tot}} = \sum_{i=1}^{N_{\text{atoms}}} \epsilon_i.$$

Here, ϵ_i is the ML prediction of the energy contribution of the i-th atom environment. This energy decomposition generally has no physical meaning but seems intuitive with the near-sightedness principle in mind. For the prediction of electron densities we where not able to make use of this idea. However, by using atom-centered basis functions this becomes possible in an analogous way.

We are not dealing with atom environments but with basis function coefficients: Let's say that q and p are close if the atoms corresponding to χ_q and χ_p are close to each other. Then, since the GTO basis functions are centered around one specific atom, we can — with the same argument as above — also postulate that

D_{qp} does not depend on $F_{q'p'}$ when either q' or p' centers are far away from q or p centers (and thus $D_{qp} = 0$ for distant q and p centers). For close q and p centers we can predict D_{qp} from $F_{q'p'}$ that are part of the atom environment. In fact, this idea might work even better than for the energy prediction, because we are predicting single-electron orbitals and all electron-interactions are still handled by the exchange-correlation potential.

This view might provide the possibility to scale this method to much larger molecules. For the results on small molecules presented here however, we do not make use of the orbital environment view.

We use the methodology introduced in Chapter 2 to predict basis coefficients for predicting density matrix coefficients, i.e. with independent kernel models

$$D_{qp}^{\mathrm{ML}}(F) = \sum_{i=1}^{N} \alpha_{qp}^{(i)} k(F, F^{(i)}). \tag{A.7}$$

To generate training data, we collect (F, D) pairs from multiple DFT calculations. We use stratified cross-validation, i.e. we do not validate the hyperparameters on (F, D) pairs that were collected from calculations that also contributed to the training set. We thus avoid training a model that relies on knowledge about DFT iterations from a geometry for which we want to make predictions.

We apply a similar method for evaluating the accuracy of the method. All presented results are predictions on geometries that did not contribute to the model training.

A.4 Experiments

We tested the theoretic suggestions by implementing the methodology into the computational chemistry software NWChem [Val+10]. NWChem implements gaussian-type orbital Kohn-Sham density functional theory. We use the `6-31G*` basis functions [DHP71] provided in the NWChem library and B3LYP [Bec93; Ste+94] exchange-correlation.

NWChem by default uses canonical orthogonalization for the first iteration but

	Orthogonalization	MAE	90%	max
H_2	None	0.064	0.12	0.59
	Canonical	0.0028	0.0052	0.024
	Cholesky	0.053	0.15	0.36
H_2O	None	0.083	0.15	0.29
	Canonical	1.2	0.77	16
	Cholesky	0.020	0.053	0.15

Table A.1: Total energy mean average errors for density matrix prediction with different orthogonalization approaches. For H_2, we trained the model with 5 reference calculations; for H_2O, we trained the model with 15 reference calculations. Errors in kcal/mol.

then orthogonalizes by writing the Fock matrix in a basis of the previous iteration's molecular orbitals.

We evaluate the method on the H_2 and H_2O atom positions used in Secs. 4.3.1 and 4.3.2 by comparing total energies. The results are given in Table A.1.

What happens if the density matrix prediction becomes inaccurate for some iterations? This can happen when the iteration trajectory leaves the training data manifold. We simulate this by adding noise to the density for the first five iterations. We measure the variance of the density matrix entries $v_{qp} = \mathrm{Var}(D_{qp})$ for the training data. Then we add Gaussian noise with variance σ times v_{qp}, i.e.

$$D_{qp} \leftarrow D_{qp} + \mathcal{N}(0, \sigma v_{qp}) \tag{A.8}$$

for the first five iterations. We plotted the energy trajectory for calculations of several geometry with added noise ($\sigma = 50\%$) in Fig. A.2. The trajectory does not deviate too much even with heavy noise influence and converges to the ground-state energy in every case.

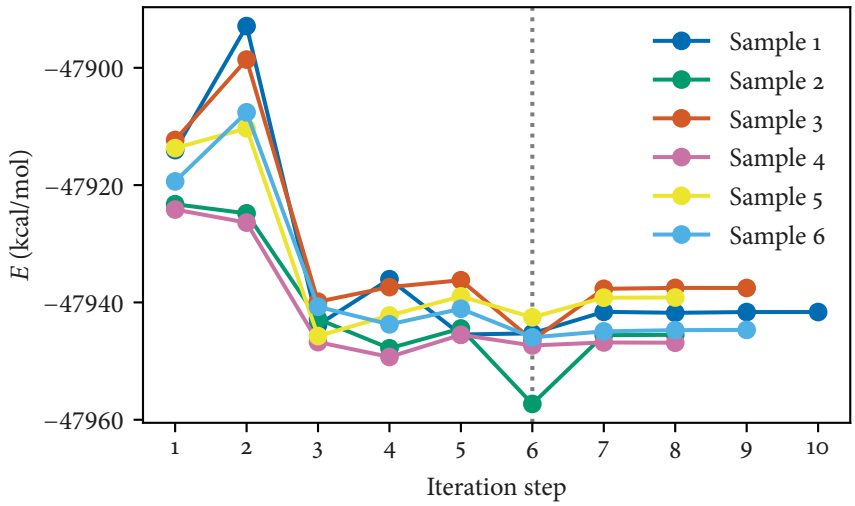


Figure A.2: The figure shows the density matrix prediction energy trajectories of six different geometries. Here, the predicted densities of the first five iterations were manually contaminated with noise. The remaining iterations receive regular density predictions and the calculations converge to the ground-state energy.

A.5 Discussion

The density matrix prediction approach outperforms the ML-HK approach presented in Sec. 4.3 for H_2 and H_2O. This is not surprising since the density matrix prediction approach does not have to predict Hartree and exchange-correlation contributions and the energy is calculated, not predicted, from the predicted density matrix. It also has access to more training data, specifically to all iterations of the Kohn-Sham loop.

The experiments show that all three orthogonalization methods are able to yield accurate energy predictions. However, *canonical orthogonalization* works less reliably for H_2O but gives best results on H_2. We attribute this to a higher sensitivity to noise in the larger eigen-directions of the orthogonalized molecular orbitals for H_2O.

Symmetric orthogonalization and *no orthogonalization* yield comparable accuracies. Surprisingly, the ML model does not need any overlap information to

predict the density matrix. This means that the overlap information must be encoded in the Fock matrix, i.e. a change in overlap changes the Fock matrix so distinctly that the data manifold does not contain similar Fock matrices associated to different overlap matrices. These would result in different density matrices and thus be unpredictable by the ML model.

It is interesting that this approach is robust, even with heavy noise influence. We presume that the Fock matrix step provides some kind of self correcting behavior.

The approach skips the diagonalization of the Fock matrix and directly predicts the corresponding density matrix. We thus have no access to either the molecular orbital coefficients nor the eigenvalues. The eigenvalues are important because they are used to determine the HUMO-LUMO gap but they can be retrieved via the Roothaan-Hall equation.

This example implementation and its result show that the principle of predicting densities is even adaptable to SCF calculations. Having access to the density matrix even allows the calculation of analytical energy gradients (forces) [Pul14]. Implementing the energy gradients is possible, yet not straight-forward. It requires substantial modifications to the NWChem code and goes beyond the scope of this thesis.

Bibliography

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. "Least-Squares Fitting of Two 3-D Point Sets". In: *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-9.5 (Sept. 1987), pp. 698–700.

[AV07] David Arthur and Sergei Vassilvitskii. "k-means++: the advantages of careful seeding". In: *Proc. eighteenth Annu. ACM-SIAM Symp. Discret. algorithms*. New Orleans, Louisiana: Society for Industrial and Applied Mathematics Philadelphia, 2007, pp. 1027–1035.

[Bec93] Axel D. Becke. "Density-functional thermochemistry. III. The role of exact exchange". In: *J. Chem. Phys.* 98.7 (1993), pp. 5648–5652.

[BKC13] Albert P Bartók, Risi Kondor, and Gábor Csányi. "On representing chemical environments". In: *Phys. Rev. B* 87.18 (May 2013), p. 184115.

[Bro+17] Felix Brockherde, Leslie Vogt, Li Li, Mark E. Tuckerman, Kieron Burke, and Klaus-Robert Müller. "Bypassing the Kohn-Sham equations with machine learning". In: *Nat. Commun.* 8.1 (2017), p. 872.

[BW12] Kieron Burke and Lucas O. Wagner. "DFT in a nutshell". In: *Int. J. Quantum Chem.* 113.2 (2012), pp. 96–101.

[BWS04] Gökhan H Bakır, Jason Weston, and Bernhard Schölkopf. "Learning to find pre-images". In: *Adv. Neural Inf. Process. Syst.* 2004, pp. 449–456.

[Chm+17] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. "Machine learning of accurate energy-conserving molecular force fields." In: *Sci. Adv.* 3.5 (May 2017), e1603015.

[CMY08] Aron J. Cohen, Paula Mori-Sanchez, and Weitao Yang. "Insights into Current Limitations of Density Functional Theory". In: *Science* 321.5890 (Aug. 2008), pp. 792–794.

[DG90] Reiner M. Dreizler and Eberhard K. U. Gross. *Density Functional Theory: An Approach to the Quantum Many-Body Problem*. Springer Berlin Heidelberg, 1990.

[DHP71] R. Ditchfield, W. J. Hehre, and J. A. Pople. "Self-Consistent Molecular-Orbital Methods. IX. An Extended Gaussian-Type Basis for Molecular-Orbital Studies of Organic Molecules". In: *J. Chem. Phys.* 54.2 (1971).

[Han+13] 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". In: *J. Chem. Theory Comput.* 9.8 (Aug. 2013), pp. 3404–3419.

[Hjo+17] Ask Hjorth Larsen et al. "The atomic simulation environment—a Python library for working with atoms". In: *J. Phys. Condens. Matter* 29.27 (July 2017), p. 273002.

[HK64] P. Hohenberg and W. Kohn. "Inhomogeneous Electron Gas". In: *Phys. Rev.* 136.3B (Nov. 1964), B864–B871.

[HNW93] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving ordinary differential equations I: Nonstiff Problems*. New York: Springer, 1993.

[Jac17] Mitch Jacoby. *Research of the year*. Ed. by Chemical & Engineering News. https://cen.acs.org/articles/95/i49/chemistry-research-of-the-year-2017.html. [Online; posted December 18, 2017]. 2017.

[Kad+16] Hachem Kadri, Emmanuel Duflos, Philippe Preux, Stéphane Canu, Alain Rakotomamonjy, and Julien Audiffren. "Operator-valued Kernels for Learning from Functional Response Data". In: *J. Mach. Learn. Res.* 17.20 (2016), pp. 1–54.

[Kim+15] Min-Cheol Kim, Hansol Park, Suyeon Son, Eunji Sim, and Kieron Burke. "Improved DFT Potential Energy Surfaces via Improved Densities". In: *J. Phys. Chem. Lett.* 6 (2015), pp. 3802–3807.

[KS65] W. Kohn and L. J. Sham. "Self-Consistent Equations Including Exchange and Correlation Effects". In: *Phys. Rev.* 140.4A (Nov. 1965), A1133–A1138.

[KSB13] Min-Cheol Kim, Eunji Sim, and Kieron Burke. "Understanding and Reducing Errors in Density Functional Calculations". In: *Phys. Rev. Lett.* 111.7 (Aug. 2013), p. 73003.

[KSB14] Min-Cheol Kim, Eunji Sim, and Kieron Burke. "Ions in solution: Density corrected density functional theory (DC-DFT)". In: *J. Chem. Phys.* 140.18 (2014), 18A528.

[Li+16a] Li Li, Thomas E. Baker, Steven R. White, and Kieron Burke. "Pure density functional for strong correlation and the thermodynamic limit from machine learning". In: *Phys. Rev. B* 94.24 (Dec. 2016), p. 245129.

[Li+16b] Li Li, John C. Snyder, Isabelle M Pelaschier, Jessica Huang, Uma-Naresh Niranjan, Paul Duncan, Matthias Rupp, Klaus-Robert Müller, and Kieron Burke. "Understanding machine-learned density functionals". In: *Int. J. Quantum Chem.* 116.11 (2016), pp. 819–833.

[Mik+99] Sebastian Mika, Bernhard Schölkopf, Alex J. Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. "Kernel PCA and denoising in feature spaces". In: *Adv. Neural Inf. Process. Syst.* 1999, pp. 536–542.

[Mon+13] Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. "Machine learning of molecular electronic properties in chemical compound space". In: *New J. Phys.* 15.9 (Sept. 2013), p. 95003.

[MP05] Charles A. Micchelli and Massimiliano Pontil. "On Learning Vector-Valued Functions". In: *Neural Comput.* 17.1 (Jan. 2005), pp. 177–204.

[PBE96] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. "Generalized Gradient Approximation Made Simple". In: *Phys. Rev. Lett.* 77.18 (Oct. 1996), pp. 3865–3868.

[Pre+92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. 2nd ed. Cambridge University England EPress, 1992, p. 994.

[Puk+09] A. Pukrittayakamee, M. Malshe, M. Hagan, L. M. Raff, R. Narulkar, S. Bukkapatnum, and R. Komanduri. "Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks". In: *J. Chem. Phys.* 130.13 (Apr. 2009), p. 134101.

[Pul14] Peter Pulay. "Analytical derivatives, forces, force constants, molecular geometries, and related response properties in electronic structure theory". In: *Wiley Interdiscip. Rev. Comput. Mol. Sci.* 4.3 (2014), pp. 169–181.

[PZ81] J. P. Perdew and Alex Zunger. "Self-interaction correction to density-functional approximations for many-electron systems". In: *Phys. Rev. B* 23.10 (May 1981), pp. 5048–5079.

[Rib+15] Raphael F. Ribeiro, Donghyung Lee, Attila Cangi, Peter Elliott, and Kieron Burke. "Corrections to Thomas-Fermi Densities at Turning Points and Beyond". In: *Phys. Rev. Lett.* 114.5 (Feb. 2015), p. 050401.

[RS05] James Ramsay and Bernard W. Silverman. *Functional Data Analysis (Springer Series in Statistics)*. Springer, 2005.

[Rup+12] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. "Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning". In: *Phys. Rev. Lett.* 108.5 (Jan. 2012), p. 58301.

[Sch+14] K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K.-R. Müller, and E. K. U. Gross. "How to represent crystal structures for machine learning: Towards fast prediction of electronic properties". In: *Phys. Rev. B* 89.20 (May 2014), p. 205118.

[Sch+17] Kristof T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus-Robert Müller, and Alexandre Tkatchenko. "Quantum-chemical insights from deep tensor neural networks". In: *Nat. Commun.* 8 (Jan. 2017), p. 13890.

[Sch95] A. Schwarzenberg-Czerny. "On matrix factorization and efficient least squares solution." In: *Astron. Astrophys. Suppl.* 110 (Apr. 1995), p. 405.

[Sny+12] John C. Snyder, Matthias Rupp, Katja Hansen, Klaus-Robert Müller, and Kieron Burke. "Finding Density Functionals with Machine Learning". In: *Phys. Rev. Lett.* 108.25 (June 2012).

[Sny+13a] John C. Snyder, Sebastian Mika, Kieron Burke, and Klaus-Robert Müller. "Kernels, Pre-images and Optimization". In: *Empirical Inference*. Springer Berlin Heidelberg, 2013, pp. 245–259.

[Sny+13b] John C. Snyder, Matthias Rupp, Katja Hansen, Leo Blooston, Klaus-Robert Müller, and Kieron Burke. "Orbital-free bond breaking via machine learning". In: *J. Chem. Phys.* 139.22 (Dec. 2013), p. 224104.

[Sny+15] John C. Snyder, Matthias Rupp, Klaus-Robert Müller, and Kieron Burke. "Nonlinear gradient denoising: Finding accurate extrema from inaccurate functional derivatives". In: *Int. J. Quantum Chem.* 115.16 (Aug. 2015), pp. 1102–1114.

[SS03] Elias M. Stein and Rami Shakarchi. *Fourier analysis: an introduction*. Princeton University Press, 2003.

[SSM98a] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". In: *Neural Comput.* 10.5 (July 1998), pp. 1299–1319.

[SSM98b] Alex J. Smola, Bernhard Schölkopf, and Klaus-Robert Müller. "The connection between regularization operators and support vector kernels." In: *Neural Netw.* 11.4 (June 1998), pp. 637–649.

[Ste+94] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, and M. J. Frisch. "Ab Initio Calculation of Vibrational Absorption and Circular Dichroism Spectra Using Density Functional Force Fields". In: *J. Phys. Chem.* 98.45 (1994), pp. 11623–11627.

[Ste56] H. Steinhaus. "Sur la division des corps matériels en parties". In: *Bull. Acad. Pol. Sci. Cl. III.* 4 (1956), 801–804 (1957).

[Tuc+00] Mark E. Tuckerman, D. A. Yarne, Shane O. Samuelson, Adam L. Hughes, and Glenn J. Martyna. "Exploiting multiple levels of parallelism in Molecular Dynamics based calculations via modern techniques and software paradigms on distributed memory computers". In: *Comput. Phys. Commun.* 128.1–2 (2000), pp. 333–376.

[Tuc10] Mark E. Tuckerman. *Statistical mechanics : theory and molecular simulation*. Oxford University Press, 2010, p. 123.

[Val+10] M. Valiev et al. "NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations". In: *Comput. Phys. Commun.* 181.9 (2010), pp. 1477–1489.

[Vap98] Vladimir Naumovich Vapnik. *Statistical learning theory*. Wiley, 1998.

[Wan+04] Junmei Wang, Romain M. Wolf, James W. Caldwell, Peter A. Kollman, and David A. Case. "Development and testing of a general amber force field". In: *J. Comput. Chem.* 25.9 (2004), pp. 1157–1174.