


2019

The Structural Information Filtered Features Potential for Machine Learning calculations of energies and forces of atomic systems.

Jorge Arturo Hernandez Zeledon
West Virginia University, jahernandezzeledon@mix.wvu.edu

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

 Part of the [Biological and Chemical Physics Commons](#), [Condensed Matter Physics Commons](#), and the [Other Computer Sciences Commons](#)

Recommended Citation

Hernandez Zeledon, Jorge Arturo, "The Structural Information Filtered Features Potential for Machine Learning calculations of energies and forces of atomic systems." (2019). *Graduate Theses, Dissertations, and Problem Reports*. 3790.

<https://researchrepository.wvu.edu/etd/3790>

This Dissertation is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Dissertation in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Dissertation has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

**THE STRUCTURAL INFORMATION FILTERED FEATURES
POTENTIAL FOR MACHINE LEARNING CALCULATIONS
OF ENERGIES AND FORCES OF ATOMIC SYSTEMS.**

Jorge Arturo Hernandez Zeledon

Dissertation submitted
to the Eberly College of Arts and Sciences
at West Virginia University

in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in
Physics

James P. Lewis, Ph.D., Chair
Cheng Cen, Ph.D.
Edward Flagg, Ph.D.
Tudor Stanescu, Ph.D.
Xiao-Dong Wen, Ph.D.

Department of Physics and Astronomy

Morgantown, West Virginia, 2019

Keywords: Machine Learning, Empirical Potentials, Feature Engineering

Copyright 2109 © Jorge Arturo Hernandez Zeledon

Abstract

The Structural Information Filtered Features Potential for Machine Learning calculations of energies and forces of atomic systems.

Jorge Arturo Hernandez Zeledon

In the last ten years, machine learning potentials have been successfully applied to the study of crystals, and molecules. However, more complex materials like clusters, macro-molecules, and glasses are out reach of current methods.

The input of any machine learning system is a tensor of features (the most universal type are rank 1 tensors or vectors of features), the quality of any machine learning system is directly related to how well the feature space describes the original physical system. So far, the feature engineering process for machine learning potentials can not describe complex material. The current methods are highly inefficient transforming the information of the physical structure into the feature vector, the losses of information constraint the accuracy of machine learning potentials.

This work introduces the Structural Information Filtered Features (SIFF), the SIFF is a feature engineering method, based on maximizing the transfer of information from the physical structure to the feature space. The SIFF are thought as a universal feature, universal in two senses. First is able to describe complex systems, as well as molecules, and crystals. Second it can

be easily used as input for any machine learning algorithm.

When applied to crystals the SIFF does as well as the best feature engineering methods for this materials (SOAP, CGNN). When applied to molecules the SIFF performs better than the Bag of Bonds method, especially when the number of structures is reduced to less than 10000, in this conditions the SIFF shows a superior performance, due to its superior information transference. With respect to complex system, the SIFF is compared to the Behler and Parrinello approach, here the SIFF method reach an error of $0.083eV/structure$ in 18110 second, in contrast the Behler and Parrinello method achieved an error of $0.109eV/structure$ in 61969 seconds.

The main disadvantage of the SIFF method is that the dimensionality of the feature space grows exponentially with the number of chemical species in the system.

**THE STRUCTURAL INFORMATION FILTERED FEATURES
POTENTIAL FOR MACHINE LEARNING CALCULATIONS
OF ENERGIES AND FORCES OF ATOMIC SYSTEMS.**

Approved by:

Dr. James P. Lewis, Advisor
Department of Physics and
Astronomy
West Virginia University

Dr. Cheng Cen
Department of Physics and
Astronomy
West Virginia University

Dr. Edward Flagg
Department of Physics and
Astronomy
West Virginia University

Dr. Tudor Stanescu
Department of Physics and
Astronomy
West Virginia University

Dr. Xiao-Dong Wen
Institute of Coal Chemistry
Chinese Academy of Sciences

Date Approved: April 8th,
2019

To my mother Mariana for her love, support, and encouragement to follow my dreams regardless of how unlikely they seemed. To my wife Connie for all her love that kept me working until the end.

Acknowledgments

I would like to thank my advisor Dr James P. Lewis, for his guidance and support. I want to thank my wife, mother and sisters for her love and faith in me. Special thanks to Dr Aldo Romero, first for a great quantum mechanics class, and second for all the scientific and life advice given during this time. Thanks to Dr Guillermo Aveño for introducing me to many of the tools I ended up using for this work, and also for all the technical support that he gave me through the HPC center at WVU. I also want to thank my research mates and friends Gihan Panapitiya, Pedram Tavazohi, and Uthpala Herath for all the fruitful discussions. Thanks to all my friends at the Physics and Astronomy Department for making the time in graduate school bearable, and my stayed in Morgantown unforgettable.

The completion of this dissertation wouldn't been possible without the help given by the administrative staff at the Physics and Astronomy Department, specially to Viola Bryant, and finally I want to thank Darlene and Bryon Mitchell for their support during the last stages of this work.

To all of you from the depths of my heart thanks.

J. Arturo Hernandez Zeledon

West Virginia University

April 2019

Table of Contents

Approval Page	iv
List of Tables	x
List of Figures	xi
1 Introduction	1
1.1 Introduction	1
2 Machine Learning	6
2.1 An introduction to Machine Learning	6
2.2 Feature Engineering	10
2.2.1 Examples Of Features	11
2.2.2 Feature Selection: General picture	14
2.3 Neural Networks	20
2.3.1 Fully Connected Neural Network (FCNN)	21
2.3.2 Activation Functions	26
2.3.3 The Learning Process	28
2.3.4 Convolution Of Neural Networks	35
2.4 Regression Trees And Gradient Boosting Regression	36
2.4.1 Regression Trees	38
2.4.2 Gradient Boosting regression	42
2.5 Mixtures of gaussians	44
3 Density Functional Theory	46
3.1 The original problem, many body quantum mechanics	46
3.2 Density functional theory	50
3.2.1 Approximate exchange correlation functional	55

3.2.2	DFT implementation	56
3.2.3	Pseudo-potentials	59
4	Force Fields	61
4.1	Introduction	61
4.2	Force Fields: Functional Form	63
4.2.1	Bonding Energy	63
4.2.2	Angle Bending Energy	64
4.2.3	Torsion Energy	66
4.2.4	Electrostatic Energy	67
4.2.5	Van der Waals Energy	67
4.3	The Force Field As A Whole	68
4.4	Parametrization Of A Force Field	70
4.4.1	Parametrization For The Bonding Energy And The Angle Bending Energy	71
4.4.2	Parametrization Of The Lenard Jones Potential (Van der Waals Energy)	72
4.4.3	Parametrization Of The Torsion Energy	73
4.5	Force Fields Conclusions	74
5	Machine Learning Potentials	77
5.1	Introduction To Machine Learning Potentials	77
5.2	Feature Creation For Machine Learning Potentials	79
5.3	Neural network potential with symmetry functions	83
5.3.1	Symmetry functions	84
5.4	Gaussian approximation potentials	87
5.4.1	The smooth overlap of atomic positions	88
5.5	Crystal Graph Convolution Of Neural Networks	91
5.5.1	The CGCNN algorithm	92
5.6	Molecular gaussian potentials	93
5.6.1	The coulomb matrix and the bag of bonds	94
5.7	The Partial Radial Distribution Function	96
6	The Structural Information Filtered Features (SIFF)	98
6.1	The Structural Information Filtered Features (SIFF)	98
6.2	A formal introduction to the Structural Information Filtered Features	101
6.2.1	SIFF force features	107

6.3	Results of SIFF calculations on clusters	109
6.3.1	SIFF comparison with BP on random clusters	109
6.4	Results of SIFF calculations on Molecules	117
6.4.1	Molecular data sets	117
6.4.2	Model selection	118
6.4.3	Results and analysis on the OCH data set	120
6.4.4	Results and analysis on the C7O2H12 data set	125
6.5	Results of SIFF calculations on Crystals	128
6.5.1	Crystal data set	128
6.5.2	Model selection	130
6.5.3	Results and analysis on the AlGaInO data set	132
7	Conclusions	139
7.1	Conclusions	139
A	Feature parameters	142
A.1	SIFF parameters for the C10 data set	142
A.2	BP parameters for the C10 data set	143
A.3	SIFF parameters for the CO1214 data set	144
A.4	SIFF parameters for the OCH data set	144
A.5	SIFF parameters for the C7O2H12 data set	145
A.6	BoB parameters for the OCH and C7O2H12 data sets	146
A.7	SIFF parameters for the AlGaInO data set	146
A.8	SOAP parameters for the AlGaInO data set	147
A.9	CGCNN parameters for the AlGaInO data set	147
	References	160

List of Tables

6.1	BP vs SIFF on clusters	112
6.2	Best models molecular data set	120
6.3	Best models crystal data set	132
6.4	Kaggle competition results	137

List of Figures

2.1	X matrix of input data	9
2.2	Pearson correlation coefficient plots	16
2.3	Diagram Fully Connected Neural Network	21
2.4	Layers of a Fully Connected Neural Network	23
2.5	Sigmoid activation function	27
2.6	Tanh activation function	28
2.7	Rectified Linear activation function	29
2.8	Forward and backward propagation	33
2.9	Convolution of Neural Networks	36
2.10	Regression tree	38
3.1	Diagram Hohenberg Kohn theorem	52
3.2	Self consistency Kohn Sham equation	57
3.3	Pseudo-potential vs all electron potential	60
4.1	Bond, angle, dihedral interactions	65
4.2	Force Field parametrization	70
5.1	Atomistic view vs Structure view	80
5.2	BP Neural Network calculation	83
5.3	Feature parameters	86
5.4	Bag of Bonds representation of CO_2	95
6.1	convolution of Neural Networks	103
6.2	Two-body filters	104
6.3	SIFF feature engineering process	106
6.4	Cumulative error C10 data set	113
6.5	Cumulative error CO1214 data set	114
6.6	Correlation measurements BP vs SIFF C10 data set	116
6.7	Correlation measurement CO1214 data set	117
6.8	Model selection scores molecular data sets	119
6.9	Gap manifold OCH data set	121
6.10	Cumulative error gap energy OCH data set	122
6.11	HOMO manifold OCH data set	123
6.12	Cumulative error HOMO OCH data set	124
6.13	LUMU manifold OCH data set	125
6.14	Cumulative error LUMU OCH data set	126
6.15	Internal energy manifold (C7O2H12 data set)	127

6.16	Cumulative error internal energy C7O2H12 data set)	128
6.17	Free energy manifold C7O2H12 data set)	129
6.18	Cumulative error free energy C7O2H12 data set)	129
6.19	Model selection scores crystal data sets	131
6.20	Cumulative error formation energy AlGaInO data set	132
6.21	Formation energy manifold AlGaInO data set	134
6.22	Band gap manifold AlGaInO data set	135
6.23	Cumulative error band gap AlGaInO data set	136

Chapter 1

Introduction

1.1 Introduction

The invention of materials with the potential to solve daily problems, makes material science a key discipline for the development of our society. However, the process of creating materials faces many challenges, especially in predicting the properties of a given configuration of atoms.

It is true that the wave function contains all the information known about a given system. Yet the wave function is the result of solving Schrödinger's equation (SE), which takes considerably computational resources, even for simple molecules or crystals. Moreover, prediction of properties requires extensive knowledge of the potential energy surface (PES) which is only accessible by molecular dynamics (MD) simulations. Molecular dynamics simulations rely on knowing the energy, and the forces of the system at every

step.

The limitation imposed by the complexity of solving the SE offered opportunities for alternative methods to calculate energies, and forces of systems of atoms. By far the most successful of these methods is the Density Functional Theory (DFT) developed in the 60s by Hohenberg, Kohn, and Sham. DFT is an exact theory, that instead of focusing on a many-body wave function, focuses on the many-body density which is a single value quantity. The main premise of DFT is that the electronic density contains all the information of the ground state of the many-body system. However, despite the advances introduced by DFT, ab initio simulations are still far from been feasible for systems like macro-molecules, or glasses.

Simulations for systems like macro-molecules and glasses are performed with empirical potentials, or force fields (FF). The FF are a simplification of the energy function of a system of atoms. This simplification, splits the total energy in different interactions, such as: bonds, dihedral angles, bendings, torsions, etc. Yet the functions used to represent the energy parts are unable to capture many of the quantum mechanical phenomena, in consequence molecular dynamics performed by FF is constrained to spaces of the configuration space where quantum mechanics does not play a fundamental role.

Then there is dichotomy in the world of theoretical simulations of materials, in one hand there are methods with quantum mechanical precision, but bounded by the size, and time of the simulation, on the other hand there are

methods light enough to manage thousands of particles for even nanoseconds, but without quantum mechanical accuracy. To solve this discrepancy new tools have been employed. One of the most significant tools is machine learning. The field of Machine learning lies in the intersection between: computer science, mathematics, and statistics. Machine learning are all the algorithms and methods concern with fitting the parameters of a complex parametric mathematical mapping to reproduce a given set of data. Machine learning algorithms are the engine of Machine learning potentials which are the evolution of force fields, the main idea is to use a machine learning algorithm to learn the potential energy surface of a system of atoms. Perhaps the biggest advantage of machine learning potentials is that they rely on mathematical mappings complex enough to reproduce quantum calculations. In the last 10 years the applications of machine learning methods to materials science have grown exponentially thanks to the apparition of massive data bases with quantum mechanical calculations. The increasing amount of machine learning potentials is a prove of its popularity.

While machine learning potentials have proved successful to speed up the study of materials like crystals and molecules with errors compared to DFT calculations, the reality is that there are still many challenges. Some problems are related with the limits of machine learning algorithms themselves, for example the fact that they need huge amounts of data to be trained with. Yet there are problems that come specifically from machine learning potentials, fundamentally there are four of them. First, the lack of a universal

representation to describe different kinds of materials to be the input to the machine learning algorithm. Second, the lack of a representation able to be the input of different machine learning algorithms. Third, the lack of big data set for materials due to the cost of using DFT, which makes important to have a representation able to compare systems with differences in the number of atoms, but similar in their compositions. And last but not least, there is no machine learning potential capable of describing disorder states with the same accuracy compared with which machine learning potentials are applied to crystals and molecules. This last problem is of special importance, since a structure during a long molecular dynamics simulation is likely to experience significant modifications, while passing through disordered states in its configuration space.

In this work the Structural Information Filter Features (SIFF) are introduced. The SIFF is a feature engineering algorithm to transform information from a physical structure to a feature vector. The goal of the SIFF is to maximize the information transfer, so that any physical system can be represented by a feature vector regardless of its complexity. In addition the dimension of the SIFF feature space is independent on the number of particles in a structure, and the SIFF can be used as input of any machine learning system.

This dissertation is divided in 7 chapters. Chapter 2, deals with the machine learning methods used to build the machine learning potentials, also explains the feature engineering process, and how vital good features are for the success of any machine learning application, it also explains concepts

important for the derivation of the SIFF like convolution of neural networks. Chapter 3 is devoted to DFT, the Hohenberg Kohn theorems are explained, and the Kohn Sham equation derived, in addition some remarks on practical applications are mention. Chapter 4, introduces force fields, explains details about their construction, and applications. To finally analyze their limits. Chapter 5 summarizes the field of machine learning potentials, reviewing some of the most influential methods of the last 10 years, as well as important methods that influence the SIFF. Chapter 6 is the most substantial, there the SIFF are formally introduced, also the results of different tests are presented to prove their reliability, and find their weak spots. Chapter 7 summarizes the conclusions, and expose where further developments should be done in order to improve the next generation of machine learning potentials.

Chapter 2

Machine Learning

2.1 An introduction to Machine Learning

Machine Learning is a set of methods, algorithms and procedures, to extract knowledge out of data, to make predictions. Formally speaking, Machine Learning is: all the methods, related to the learning of parametric and non-parametric mappings $f : x \rightarrow y$, to predict the value, of a different outcome variable "y", from the information stored in an input "x".

In general, machine learning methods have three stages. First the gathering and preprocessing of the data to teach (train) the machine learning algorithms (MLA). Second the training of the MLA with the data already gathered. Third and final, the use of the MLA to make predictions in a data set different from the one used for training.

In the specific, Machine Learning methods are divided in two groups by

how they are trained: supervised learning algorithms and unsupervised learning algorithms. In supervised learning, the goal is to learn the relationship between the input x and the target y variables, from a set of examples where both pieces are known. Instances of methods in the supervised learning class are: neural networks [1, 2], decision trees [3], support vector machines[4], Gaussian methods[5]. On the other hand un-supervised learning, the only known information at the time of training are the input variable x , and the goal is to find interesting patterns in the data. One well known method in the un-supervised learning class is the clustering method [6], that tries to group elements that share properties.

Before writing about specific MLA or preprocessing techniques, it is important to introduce the notations and common words of the machine learning field with a simple example. Consider the problem of predicting, whether a patient hospitalized due to a heart attack will have a second one considering as input data: demographics, diet, and concentration of glucose and fat in its blood. In this supervised learning classification problem, the goal is to teach a MLA how to predict the value of the output variable "y", also known as target or response variable from training data in which we already know the value of "y". Here "y" is a categorical variable, that can only take certain values in a definite set $y_i \in \{0, 1, 2, ..C\}$, where C is the total number of categories, y_i can belong to. In the case of the example, the output variable can have two values: $y = 0$ if the patient does not have a second heart attack, and $y = 1$ if the patient has a second heart attack. On the other

hand problems where $y \in \mathfrak{R}$ are known as regression problems.

Now with regard of the input variable "x", "x" is usually a column vector $\vec{x}^T = (x_1, \dots, x_p, \dots, x_d)$. \vec{x} is formally known as the feature vector, the feature space has a dimensionality "d", where the dimensions are features, or attributes of the system we want to described, such that everyone of the x_p in the feature vector is a piece of information needed to describe the system for which we want to predict "y". In terms of the example; the system are the hospitalized patients that had a heart attack and the components of the feature vector are: x_1 representing the demographic, x_2 the diet, x_3 the concentration of glucose and x_4 the concentration of fat.

To train a MLA using supervised learning we need several examples of the form (\vec{x}_i, y_i) , where y_i is the target values that the MLA will learn to predict, and \vec{x}_i represents the input information needed to make the prediction. Usually the set of known examples is divided in two groups; one for training the algorithm with usually 80 % of the data while the remaining 20 % of the data is used to validate the trained algorithm. It is important to note that sometimes the input information is denoted as X , which represents a matrix containing all the training examples, with dimensions (N, d) , where every row accommodates a feature vector \vec{x}^T , in the same spirit Y is a column vector of dimension N , with all the training examples concatenated (see figure 2.1).

The predictions done by the MLA are denoted by $f(\vec{x}_i)$ or y'_i . The learning process is carried out by a successive minimization of the cost function, which

$$\begin{array}{c}
 \begin{matrix} \text{"d" total features} \\ \hline \end{matrix} \\
 X = \begin{pmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_n^T \\ \vdots \\ \vec{x}_N^T \end{pmatrix} = \begin{pmatrix} x_{1,1} & \cdots & x_{1,p} & \cdots & x_{1,d} \\ \vdots & & \vdots & & \vdots \\ x_{n,1} & \cdots & x_{n,p} & \cdots & x_{n,d} \\ \vdots & & \vdots & & \vdots \\ x_{N,1} & \cdots & x_{N,p} & \cdots & x_{N,d} \end{pmatrix} \quad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{pmatrix} \left| \begin{matrix} \text{"N" total examples} \end{matrix} \right.
 \end{array}$$

Figure 2.1: Example of the X matrix to represent the input data, where the number of columns, is the number of features, in the feature space, and the number of rows is the "N" number of examples. The Y vector of the target data, contains the "N" examples.

is a metric to evaluate how well the mapping $f(\vec{x}_i)$ predicts the real output y_i . A well known example of cost function J is the one defined with the square error:

$$J = \frac{1}{2N} \sum_{i=1}^N (y_i - f(\vec{x}_i))^2 \quad (2.1)$$

In general the cost function is defined as the summation of the losses of every independent training example: $J = \frac{1}{2N} \sum_{i=1}^N L(y_i, f(\vec{x}_i))$ in the case of equation 1 the loss function is $L(y_i, f(\vec{x}_i)) = (y_i - f(\vec{x}_i))^2$.

As it was said before, the learning process is done by a systematic minimization of the J function. For the cases where the MLA is a parametric mapping, like neural networks or Gaussian process, the systematic minimization is carry out by a gradient descent algorithm [7, 4, 5], the gradient descent algorithm updates the parameters of the mapping. However if the mapping is non-parametric like decision trees, the minimization is carry out by different algorithms like C4.5 [8], and CART [3]. In both cases the learning process is

carried out by minimizing J on the training set, and the learning process is finished once J reach a minimal in the validation set.

To summarize this gentle introduction to supervised machine learning. The process of machine learning starts, by gathering data to teach the MLA, once the data is organized in $(Y, X)_{training}$ and $(Y, X)_{validation}$, the teaching algorithm can start minimizing the J function, the learning process stops once the J function reach a minimal for the validation set.

The subsequent sections of this chapter, are going to deal with either: process, and methods needed to understand the machine learning behind the machine learning potentials described in this dissertation. The section 2 of this chapter introduces the process of feature engineering giving examples of how features are selected and how to evaluated their quality. Section 3 deals with Neural Networks, how they make predictions and how they are trained. Finally section 4 is devoted to Gradient Boosting Regression and Regression Trees.

2.2 Feature Engineering

Feature engineering is all the methods and techniques, in the pre processing stage for which the input variables X are selected constructed or transformed to describe different aspects of the target variable Y .

The present section has to two parts, the first part shows examples of how some iconic features are selected or constructed in the fields of text

representation, and time series. The second part is more abstract yet of high practical importance, since it explores the information measurements needed to select a set of features capable of predicting the target variable with accuracy.

2.2.1 Examples Of Features

It does not matter what the task of the machine learning method is: a regression, a classification, or even a clustering. The success of the method is highly dependent on finding meaningful features [9].

The feature representation can be seen as the result of translating the information stored in the data, into a language in which a MLA can make sense of the original information. The features must faithfully represent the data in a meaningful way. For example, imaging a machine leaning system trying to differentiate lemons from grapefruits. It would be useless for the task to use the shape as a feature since lemons and grapefruits are semi-spherical, contrarily the color and the diameter would result in a better set of features to tell lemons and grapefruits apart.

One of the most popular applications of machine learning is the detection of spam email. The first step of this detection consist of transforming the text in the email into a numeric feature vector. There are many algorithms capable of this transformation, some of the most advanced are the word embedding algorithms like: word2vec [10] and GloVe [11]. Yet the Bag of Words [12, 13, 14] is a simple but effective method to transform texts into feature

vectors. It consist of seeing the text as a collection of independent units (what we know as words), then counting how many times certain words appears in the document. The final representation concatenates the frequencies of appearance of important words in a given document.

A formal definition of the bag of words method is (following the definitions in Ref [14]); supposed there is a natural language vocabulary (set of words) $V = v_1, v_2, v_N$ where v_i is the "i" word in the vocabulary V . Then the Bag of Words representation for the "j" text document would be:

$$\vec{x}_j^T = (c(v_1, j), c(v_2, j), \dots, c(v_N, j))$$

Where $c(v, j)$ is the number of times the word v appears in document "j". Since every word is seen as an independent unit of the text one disadvantage of the Bag of Words is that it is unable to understand syntactic or semantic relations between words.

The last example of features are the ones proposed by Mörchen to represent time series [15]. A time series is a set of repeated measurements on a system over time. The times series "Z" with $Z = (z_1, z_2, \dots, z_N)$, where the measurements " z_i " are performed at usually uniform time steps, $t = (0, t_1, \dots, t_{N-1})$. Some examples of "Z" are temperature, pressure, prices in the stock market, wav music files. Time series easily concatenate tens of thousands of elements this make the comparison between times series hard, since in spaces of high dimensionality the notion of proximity is difficult to define [16]. This is known as the curse of dimensionality and makes the ML unable to clearly distinguish between different times series resulting in a lost

of accuracy. The goal of the feature engineering algorithm is to generate a smaller and faithful representation of the time series in such a way that if a group of time series are similar they should be close together in the feature space, also if a group of time series are different, then they should be far apart in the feature space.

The Mörchen method can be summarized on the next steps:

1) Do a discrete Fourier transform (DFT) of the times series to find the c_k coefficients of the time series.

$$c_k = \frac{1}{N} \sum_{n=0}^{N-1} z(t_n) * \exp\left(\frac{-2\pi i k n}{N}\right)$$

2) Select a number K such that every time series is represented by all the c_k for $k \leq K$, the value of K must be big enough to keep the most representative values of c_k .

3) Construct a feature vector $Z_{features}^j$ for every Z_j time series

$$Z_j \rightarrow \vec{Z}_{features}^j = (c_0^j, c_1^j, \dots, c_{K-1}^j)$$

As it could be seen, from the last two examples, the process of transforming raw data (text, music, stock prices) into feature vectors is dependent on insight knowledge of the particular field the raw data is coming from. Nevertheless, the quality of the representation can be measured, this measurements allowed researchers to systematically improve the feature repre-

sentations. The next section deals with the methods needed to study the predicting power is a set of features.

2.2.2 Feature Selection: General picture

Feature selection is one of the most important pre-process in machine learning, the goal of the task is to select a set of features with the smallest dimension possible while maximizing the amount of information of the original system stored in the feature representation.

The process of finding a good set of features to faithfully represent data is not yet standardized and is still heavily depended on domain knowledge, in addition it is important to have in mind that the construction of an optimal set of features is an intractable problem (at least so far). There are two kinds of methods to analyze the quality of a given set: the search based methods, and correlation based methods [17, 14].

The searched based methods are based on searching for different subsets of combinations among the set of possible features to describe a data set. A search based method selection has three stages:

1. The selection of a subset of features.
2. The evaluation of the selected subset.
3. An stopping criterion.

The first stage can be done by randomly selecting the subset or by sequentially adding and removing features from the subset. The second part

depends on a metric for the evaluation of the subset, in the case of a wrapper model [18, 19] the metric involves a machine learning algorithm, for example a support vector machine or a decision tree. The idea is to test several combinations of features and find out which combination has better results. On the other hand, in filter methods [18, 19] no machine learning system is used to evaluate the performance of the subset of features, instead information based calculations makes the evaluation.

In the correlation based method [17, 20, 9] the features are evaluated individually using a scoring function $s(p)$. The scoring function measures the importance of a feature " x_p " by quantifying its power to predict the target variable (" y "). Some examples of scoring functions are the Pearson correlation coefficient $R(p, y)$, and the mutual information measurement $I(p, y)$ [21].

The Pearson correlation coefficient measures the linear correlation between the feature " x_p " and the target value " y ", and it is calculated through all the data examples in hand. The value of the coefficient can be in between -1 and 1, with -1 meaning total linear negative correlation, 1 meaning total linear correlation (see figure 2.2).

$$R(p, y) = \sum_{n=1}^N \frac{(x_{n,p} - \bar{x}_p)(y_n - \bar{y})}{\sqrt{\sum_{n'=1}^N (x_{n',p} - \bar{x}_p)^2 \sum_{n'=1}^N (y_{n'} - \bar{y})^2}} \quad (2.2)$$

Here \bar{y} represents the average over the N data examples of the target value, and \bar{x}_p is the average of the feature with sub-index " p " over the N data

examples.

Another of the scoring functions is the mutual information $I(p, y)$, the measurement is based on the joint probability distribution ($P(x_p, y)$) between the feature with sub-index "p" and the target value "y". The higher the value of $I(p, y)$ the stronger the relationship between the feature and the target value. In the case in which the feature and the target value are independent $P(x_p, y) = P(x_p)P(y)$, the value of $I(p, y)$ is zero.

$$I(p, y) = \sum_{n=1}^N \sum_{n'=1}^N P(x_{n,p}, y_{n'}) \log \left(\frac{P(x_{n,p}, y_{n'})}{P(x_{n,p})P(y_{n'})} \right) \quad (2.3)$$

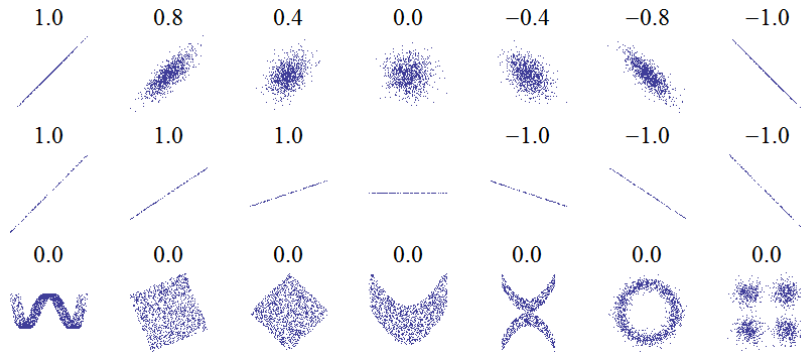


Figure 2.2: Examples of values of the Pearson correlation coefficient and its data realization. In the upper part of the image, different values of the Pearson correlation for different grades of linear dispersion. Note that in the lower part of the image non-linear correlations are shown and for all of them the Pearson correlation coefficient is zero, meaning that that the Pearson correlation coefficient can only measure linear correlations (Image created by Denis Boigelot)

The correlation selection method pursues to rank the features in four

classes[17, 14] strongly relevant, weakly relevant but non-redundant features, redundant features, and irrelevant features. To better explain the concepts behind those four subset, first remember the classification problem from before, in which a MLA must differentiate between lemons and grapefruits, the set of features (F) for this task are: the shape of the fruit (f_1), the diameter (f_2), and the color (f_3), formally the set of features can be expressed as: $F = \{f_1, \dots, f_d\}$, a feature subset (subset without feature f_j) is defined by $S_j = F - f_j$. The target value has two classes since the fruit can either be a lemon (class 0, $y=0$), or a grapefruit (class 1, $y=1$) then set of classes is $C = \{C_0, C_1\}$. Finally $P(C|f_j)$ is the probability distribution of the data into the classes by taking into account the knowledge given by the feature f_j . The formal definitions of the ranking classes for features are:

- 1) Strongly relevant features, iff:

$$P(C|f_j, S_j) \neq P(C|S_j) \tag{2.4}$$

A feature is strongly relevant, if by its own presence is able to change the probability distribution, for example in the case of the fruit classification the distribution of fruits would be confusing if only the color and the shape of the fruits are taken into account, a lemon and a grapefruit have the same shape (spherical), and similar colors (yellow-green), in contrast if the diameter is taken into account the distribution would change dramatically since the differences in diameter for lemons and grapefruits are easy to tell.

2) Weakly relevant but non-redundant features, iff:

$$P(C|f_j, S_j) = P(C|S_j), \text{ and} \quad (2.5)$$

$$\exists S'_j \subset S_j, \text{ such that } P(C|f_j, S'_j) \neq P(C|S'_j)$$

A feature is weakly relevant if it can alter the probability distribution but only with certain sub-sets of features, for example if the subset only contains the shape feature then by taking into account the color the probability distribution would change. On the other hand if a subset contains the diameter then taking into account the color makes no more difference.

3) Redundant features, iff it is also weakly relevant and has a Markov blanket ¹ M_j within F , such that :

$$P(F - M_j - \{f_j\}, C|f_j, M_j) = P(F - M_j - \{f_j\}, C|M_j) \quad (2.6)$$

Imagine (f_j) is the color, which is already a weakly relevant feature, in addition, the color of a fruit is related with the degree of ripening, which is related to the diameter, meaning that the diameter is with in the Markov blanket of the color feature, making the color a redundant feature.

4) Irrelevant features, iff:

$$\forall S'_j \subset S_j, P(C|f_j, S'_j) = P(C|S'_j) \quad (2.7)$$

¹In the context of feature space, a group of features belongs to the same Markov blanket if the features are related, tow features (p and p') are in the same Markov blanket if $P(c|x_p, x'_p) \neq P(c|x_p)P(c|x'_p)$

A feature is irrelevant if this feature does not make any change in the probability distribution, for example since both, lemons and grapefruits have spherical shape, the shape feature is irrelevant.

So far the usefulness of the features is based on scoring functions measuring the interaction between the features and the target variable. Every feature is evaluated individually ignoring that in the learning process the features interact to create a picture of the system they describe. To fix the discrepancy between evaluating the features individually and using them as an interacting system, M. Hall [22] proposed a correlation method for feature selection to take into account the interaction among features. In his method:

”The acceptance of a feature will depend on the extent to which it predicts classes (values of the target variable) in areas of the instance (feature) space not already predicted by other features.”

This method not only accounts for correlations among individual features and the target variable, but also accounts for correlations between individual features, so that in addition to calculating $s(i)$ it is also important to measure $s(i, j)$, which is the scoring function between feature ”i” and feature ”j”. The Pearson correlation coefficient is reproduced here with the feature to feature approach.

$$R(i, j) = \sum_{n=1}^N \frac{(x_{n,i} - \bar{x}_i)(x_{n,j} - \bar{x}_j)}{\sqrt{\sum_{n'=1}^N (x_{n',i} - \bar{x}_i)^2 \sum_{n'=1}^N (x_{n',j} - \bar{x}_j)^2}} \quad (2.8)$$

If the value of $s(i, j)$ is high then either ”i” or ”j” is redundant and/or irrel-

evant. Then one of them could be eliminated from the set, here is important to note that some small redundancy among features is required to decrease the noise[9].

In conclusion a good set of features, must be strongly correlated with the target variable while the correlations between features are low. In other words, every feature must contribute to the collective information of the target with an independent piece of information about the target.

2.3 Neural Networks

This subsection introduces some of the key concepts behind a Neural Network (NN). This concepts are needed to understand the Neural Networks Atomic Potentials (NNAP) [23]. A NN is a parametric mapping $f : x \rightarrow y$, where the number of parameters can easily reach the order of thousands. However is thanks to the elevate number of parameters and non-linear activation functions that NN can approximate functions regardless of its complexity, with the condition that enough data for training is provided.

The fundamental unit of a NN is a neuron, the neurons are organized in layers, and the layers are connected to form the processing system. The architecture of the NN is defined by the number of neurons, layers, and how the component are connected. The architecture studied in this subsection is the Fully Connected Neural Network (FCNN), which is the key of the NNAP introduced by Behler and Parrinello, in addition, this subsection explains the

learning process of a NN, and finally it introduces the concept of convolution of neural networks.

2.3.1 Fully Connected Neural Network (FCNN)

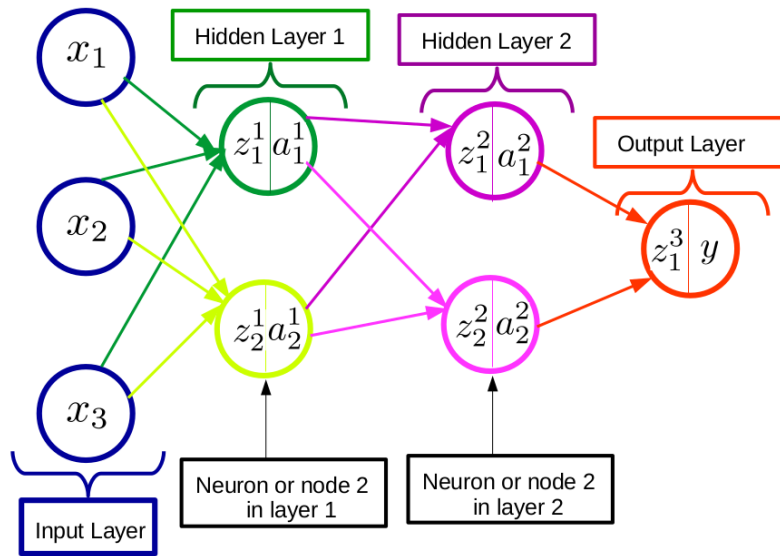


Figure 2.3: Diagram of a Fully Connected Neural Network FCNN. Here the NN has two hidden layers, plus the input layer and the output layer. Every node (neuron) performs two operations: first the acquisition and linear processing of the signals (z), and the calculation of the activation (a), in the output layer the result of the NN processing is communicated through the output variable "y"

A NN is composed by nodes or neurons organized in layers, the layers are interconnected to form a processing network [4, 5, 2] as it can be seen in figure 2.3.1. The process of transforming the information from the input layer to the output layer is called forward propagation it starts with the input layer, that has as many nodes as the input feature vector has components,

every input node is a channel to feed the information, stored in the feature vector. The layers in between the input layer, and the output layer are the hidden layers, in fully connected architectures, every node in a current layer "j", is connected to all the nodes of the immediately before layer "j-1". The architecture of a FCNN is specified by the number of nodes in every layer separated by dashes "-", for example the architecture of the FCNN of figure 3 is (3-2-2-1), it is normal to refer to the number of nodes in the "j" layer as n^j , for example: $n^0 = 3$, $n^1 = 2$, $n^2 = 2$, and $n^3 = 1$. A NN can have as many hidden layers as needed to learn any function. Increasing the number of layers makes the NN more versatile, but it also increases the amount of data needed for training. The concept of deep learning comes from the idea of stacking several hidden layers to process information. As figure 2.3.1 shows every node is divided in two parts (the node performs two operations), first the gathering and linear processing of the input information "z", and then the calculation activation "a". Figure 3 also shows the indexing notation, where z_q^j , represents the linear function in layer "j" and node "q", the same convention goes for the activation a_q^j .

The communication between layers is done by linear functions, transforming a_p^{j-1} into z_q^j , the linear functions are dependent of the parameters $\theta \rightarrow (\omega, b)$.

$$z_q^j = \sum_p^{n^{j-1}} a_p^{j-1} \omega_{p,q}^j + b_q^j \quad (2.9)$$

The parameters $\omega_{p,q}^j$ are the weights communicating the output of node "p"

in layer "j-1", with the node "q" in layer "j", the parameters b_q^j are the biases. The value of a_q^j is the result of applying the activation function to z_q^j .

$$a_q^j = \sigma^j(z_q^j) \quad (2.10)$$

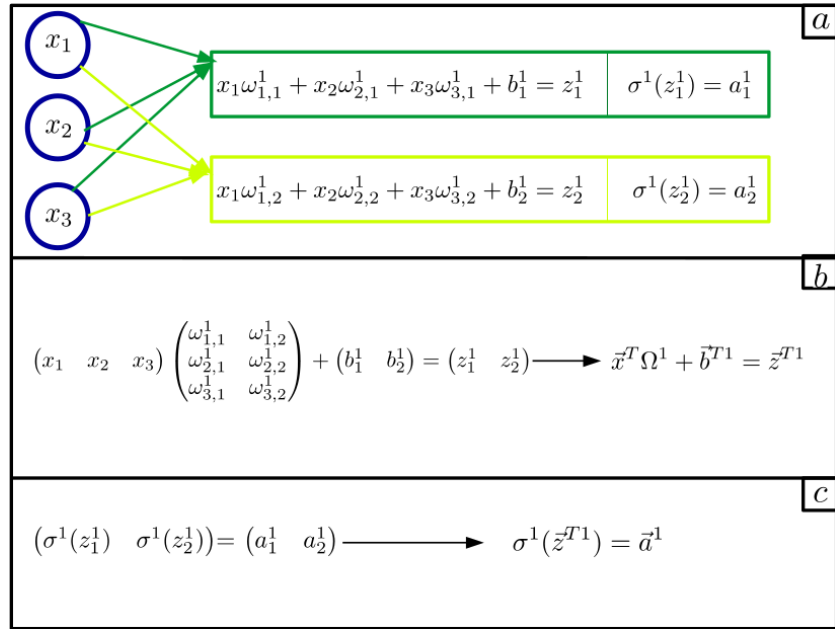


Figure 2.4: Diagram of a FCNN. In this NN with two hidden layers, plus the input layer and the output layer. Every node (neuron) performs two operations: first the acquisition and linear processing of the signals (z), and the calculation of the activation (a)

The σ^j is the activation function acting in the layer "j", the activation functions are responsible for the non-linear properties of the NN, latter in this subsection different types of activation functions are shown.

Continuing with some notation in figure 2.3.1 part a) there is a scheme of the forward propagation between the input layer and the first hidden

layer the scheme shows how the output of a layer (in this case the input layer) mixes with the weights and biases to connect with the nodes of the forward layer, then the activations are calculated and the information is communicated to the next forward layer. Parts b) and c) of figure 2.3.1 shows the "trick" of writing the forward operations like matrix operations. Taking the NN of figure 2.3.1 and writing its operations in matrix form, the forward propagation looks like:

0) Taking as input a feature vector \vec{x} with "d" features in this case 3, the vector \vec{x} acts as \vec{a}^0 , since \vec{x} is the output of the input layer.

1.1) The propagation of \vec{x} into the first layer is done by:

$$\vec{z}^{t1} = \vec{x}^T \Omega^1 + \vec{b}^{t1}$$

Here Ω^1 is a matrix with dimensions (n^0, n^1) , $n^0 = 3$ number of nodes of the before layer (input layer), and $n^1 = 2$ number of nodes in the current layer (hidden layer 1), \vec{z}^1 and \vec{b}^1 have $n^1 = 2$ dimensions

1.2) Calculation of the activation vector of the hidden layer 1 \vec{a}^1 by applying the activation function $\sigma^1()$ to the linear transformation \vec{z}^{t1} :

$$\vec{a}^{t1} = \sigma^1(\vec{z}^{t1})$$

2.1) The propagation to the second hidden layer:

$$\vec{z}^{t2} = \vec{a}^{t1} \Omega^2 + \vec{b}^{t2}$$

Here Ω^2 is a (2,2) matrix because the before layer (hidden layer 1) has 2 nodes and the current layer (hidden layer 2) has to 2 nodes. In addition \vec{z}^2 and \vec{b}^2 have 2 dimensions

2.2) The calculation of the activation of the second layer:

$$\vec{a}^{T2} = \sigma^2(\vec{z}^{t2})$$

3.1) The propagation to the output layer:

$$\vec{z}^{t3} = \vec{a}^{T2}\Omega^3 + \vec{b}^{t3}$$

Here Ω^3 is a matrix with dimensions (n^2, n^3) , $n^2 = 2$ number of nodes of the before layer (hidden layer 2), and $n^3 = 1$ number of nodes in the current layer (output layer), \vec{z}^3 and \vec{b}^3 are scalars since the output layer has only one node.

3.2) Finally the output y' , is calculated applying the activation function to the \vec{z}^3 :

$$y' = \sigma^3(\vec{z}^{t3})$$

The process of forward propagation is a succession of linear transformations and the application of a non-linear function, where the output of the before layer is the input of the current layer repeating this process until the information makes its way out through the output node or nodes.

In real life applications of NN, the forward propagation is not done one feature vector at the time. In general the input is the "X" matrix presented in figure 2.1, where the columns are every one of the features in a set of "d" features, and the rows are every example in a group of "N" examples. Thanks to highly efficient matrix operations must Deep Learning packages

manage the forward propagation as follows:

$$Z^j = A^{j-1}\Omega^j + B^j \quad (2.11)$$

Here Z^j is a matrix with dimensions (N, n^j) , A^j is also a matrix with dimensions (N, n^j) , and $A^0 = X$, B^j is the result of a broadcasting operation where the b^j with dimensions $(1, n^j)$, is stacked "N" times into the rows of B^j to create a matrix with dimensions (N, n^j) , in addition the application of the activation function is an element wise operation, where every scalar component of Z^j is transformed by $\sigma^j()$ as it is shown in figure 4c.

$$A^j = \sigma^j(Z^j) \quad (2.12)$$

Now the output of the NN Y' has dimensions (N, n^J) , where n^J is the number of nodes in the output layer.

2.3.2 Activation Functions

As we saw in the last subsection the propagation of information through the NN has two fundamental steps, first one linear transformation, and then the application of a non-linear function. Without the second step, NN would be linear regressors. It is thanks, to the flexibility introduced by the activation functions, that NN are able to approximate with success the behavior of complex functions.

There are many functions that are used as activation functions, nevertheless the NNAP relies on the hyperbolic tangent, in addition to this activation function, the sigmoid and the Rectified Linear are also introduced.

The logistic or sigmoid function:

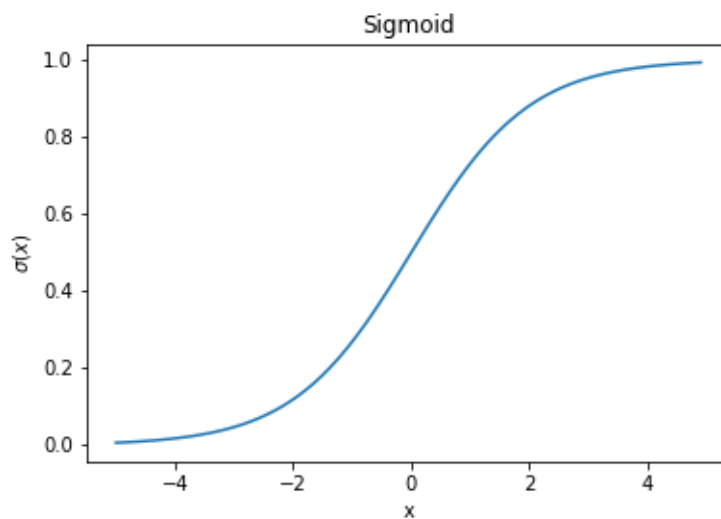


Figure 2.5: Sigmoid activation function

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.13)$$

The hyperbolic tangent tanh

$$\sigma(z) = \frac{e^{-z} - e^z}{e^{-z} + e^z} \quad (2.14)$$

The Rectified Linear:

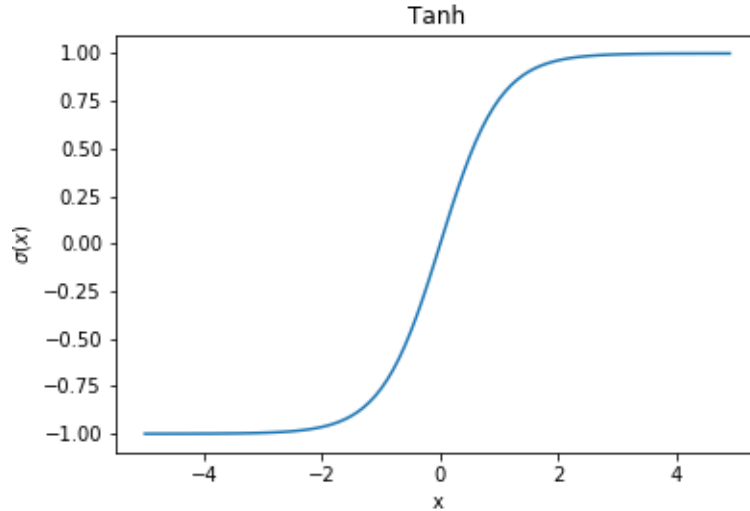


Figure 2.6: Tanh activation function

$$\sigma(z) = \begin{cases} 0 & \text{for } z < 0; \\ z & \text{for } z \geq 0 \end{cases} \quad (2.15)$$

2.3.3 The Learning Process

The result of an analysis made by a Neural Network is dependent on the values of the parameters, weights ω and biases b . The goal of the learning (training) process is to successively improve the value of the parameters, such that, the output of the NN resembles the values of the training targets Y .

The performance of the NN is measure by the cost function(defined in equation 1, reproduced here again):

$$J = \frac{1}{2N} \sum_{i=1}^N (y_i - y'_i)^2$$

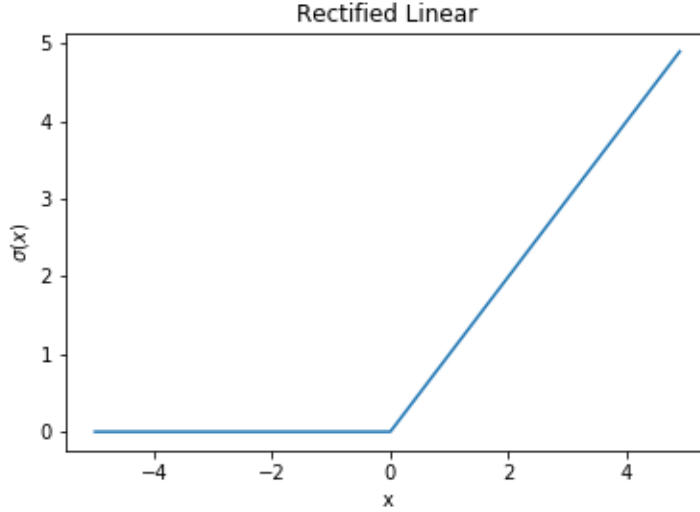


Figure 2.7: Rectified Linear activation function

There are many algorithms to improve the values of the parameters, one of the simplest, but yet powerful methods is the steepest descent, where the rule to update the parameters is:

$$\omega_{p,q}^j := \omega_{p,q}^j - \alpha \frac{\partial J}{\partial \omega_{p,q}^j} \quad (2.16)$$

$$b_q^j := b_q^j - \alpha \frac{\partial J}{\partial b_q^j} \quad (2.17)$$

Where the α is known as the learning rate. The calculation of the partial derivatives is carry out by back propagation [7], which is an algorithm to compute the derivatives using the chain rule. The name, back propagation comes from the fact, that the first partials derivatives to be evaluated, are the ones in the output layer, from there, the calculation of the partial derivatives

propagates to deeper layers, until it finally reach the input layer.

Before deriving the back propagation method, let us review the hypothetical last two layers of a FCNN with "j", layers:

$$\text{Last two layers, "j" and "j-1"} \left\{ \begin{array}{l} z_{i,p}^{j-1} = \sum_r^{n^{j-2}} a_{i,r}^{j-2} \omega_{r,p}^{j-1} + b_r^{j-1} \\ a_{i,p}^{j-1} = \sigma^j(z_{i,p}^{j-1}) \\ z_{i,q}^j = \sum_p^{n^{j-1}} a_{i,p}^{j-1} \omega_{p,q}^j + b_q^j \\ a_{i,q}^j = \sigma^j(z_{i,q}^j) \end{array} \right.$$

Where "i" is one of the "N" examples in the input data. The first partial derivative calculated is: $\frac{\partial J}{\partial \omega_{p,q}^j}$:

$$\frac{\partial J}{\partial \omega_{p,q}^j} = \frac{\partial J}{\partial a_{i,q}^j} \frac{\partial a_{i,q}^j}{\partial z_{i,q}^j} \frac{\partial z_{i,q}^j}{\partial \omega_{p,q}^j} \quad (2.18)$$

It is usual to call the term $\frac{\partial J}{\partial a_{i,q}^j} \frac{\partial a_{i,q}^j}{\partial z_{i,q}^j}$ as $dz_{i,q}^j$, now calculating the value of every term:

$$\begin{aligned} \frac{\partial J}{\partial a_{i,q}^j} &= \frac{-1}{N} (y_i - a_{i,q}^j) \\ \frac{\partial a_{i,q}^j}{\partial z_{i,q}^j} &= \sigma'^j(z_{i,q}^j) \\ \frac{\partial z_{i,q}^j}{\partial \omega_{p,q}^j} &= a_{i,p}^{j-1} \end{aligned}$$

Now the term $\frac{\partial J}{\partial \omega_{p,q}^j}$ in equation 2.18 can be written as:

$$\frac{\partial J}{\partial \omega_{p,q}^j} = \sum_{i=1}^N dz_{i,q}^j a_{i,p}^{j-1} \quad (2.19)$$

And for the b_q^j term:

$$\frac{\partial J}{\partial b_q^j} = \frac{\partial J}{\partial a_{i,q}^j} \frac{\partial a_{i,q}^j}{\partial z_{i,q}^j} \frac{\partial z_{i,q}^j}{\partial b_q^j} \quad (2.20)$$

But $\frac{\partial z_{i,q}^j}{\partial b_q^j} = 1$ then

$$\frac{\partial J}{\partial b_q^j} = \sum_{i=1}^N dz_{i,q}^j \quad (2.21)$$

Now for the parameters in the "j-1" layer, the partial derivatives goes like:

$$\frac{\partial J}{\partial \omega_{r,p}^{j-1}} = \frac{\partial J}{\partial a_{i,q}^j} \frac{\partial a_{i,q}^j}{\partial z_{i,q}^j} \frac{\partial z_{i,q}^j}{\partial a_{i,p}^{j-1}} \frac{\partial a_{i,p}^{j-1}}{\partial z_{i,p}^{j-1}} \frac{\partial z_{i,p}^{j-1}}{\partial \omega_{r,p}^{j-1}} \quad (2.22)$$

Which can be rewritten as:

$$\frac{\partial J}{\partial \omega_{r,p}^{j-1}} = dz_{i,q}^j \frac{\partial z_{i,q}^j}{\partial a_{i,p}^{j-1}} \frac{\partial a_{i,p}^{j-1}}{\partial z_{i,p}^{j-1}} \frac{\partial z_{i,p}^{j-1}}{\partial \omega_{r,p}^{j-1}} \quad (2.23)$$

Then writing down the therms:

$$\frac{\partial z_{i,q}^j}{\partial a_{i,p}^{j-1}} = \omega_{p,q}^j \quad (2.24)$$

$$\frac{\partial a_{i,p}^{j-1}}{\partial z_{i,p}^{j-1}} = \sigma^{j-1}(z_{i,p}^{j-1}) \quad (2.25)$$

Now it is possible to define:

$$dz_{i,p}^{j-1} = dz_{i,q}^j \frac{\partial z_{i,q}^j}{\partial a_{i,p}^{j-1}} \frac{\partial a_{i,p}^{j-1}}{\partial z_{i,p}^{j-1}}$$

And using equations 2.23 and 2.24

$$dz_{i,p}^{j-1} = dz_{i,q}^j \omega_{p,q}^j \sigma'^{j-1}(z_{i,p}^{j-1}) \quad (2.26)$$

The equations 2.19, 2.21, and 2.26, summarized the process of back propagation, these equations can be written in matrix notation:

$$dZ^{j-1} = (dZ^j \cdot \Omega^{Tj}) * \sigma'^{j-1}(Z^{j-1}) \quad (2.27)$$

$$\frac{\partial J}{\partial \Omega^j} = A^{Tj-1} \cdot dZ^j \quad (2.28)$$

$$\frac{\partial J}{\partial B^j} = \text{Sum}(dZ^j)_i \quad (2.29)$$

where the the symbol \cdot represent the usual matrix product and the $*$ represent an element wise multiplication. Also the partial derivatives $\frac{\partial J}{\partial \Omega^j}$ and $\frac{\partial J}{\partial B^j}$ represent the gradients respect to Ω^j and B^j . The $\text{Sum}(dZ^j)_i$ represents the addition of all the elements of the matrix dZ^j with dimensions (N, n^j) in the first axis, the result of this operation is a vector with the same dimensions of B^j . In matrix notation the updates rules for the steepest decent are:

$$\Omega^j := \Omega^j - \alpha \frac{\partial J}{\partial \Omega^j} \quad (2.30)$$

$$B^j := B^j - \alpha \frac{\partial J}{\partial B^j} \quad (2.31)$$

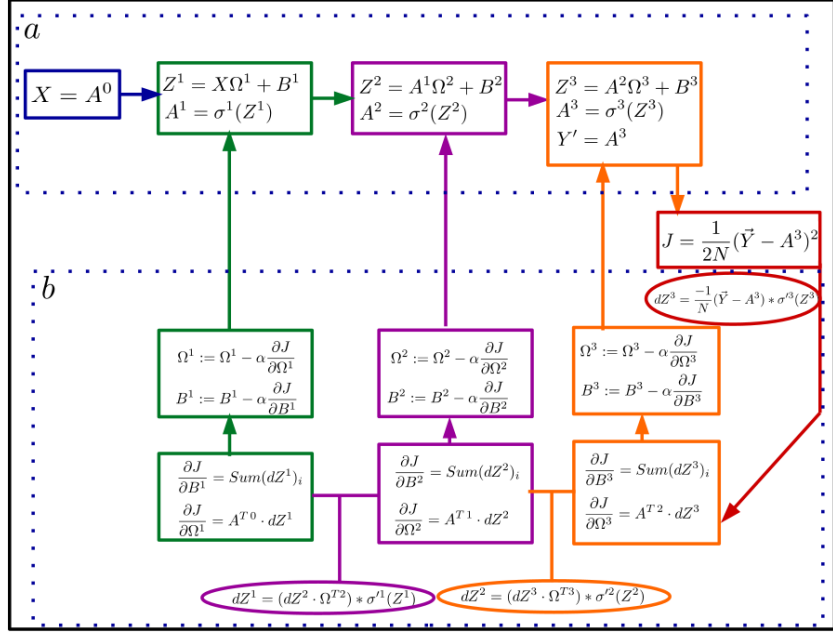


Figure 2.8: Diagram with the forward and backward propagation for the FCNN from figure 2.3.1. The part a) has the forward propagation, every box represent the process inside every one of the layers of the NN. Part b) has the back propagation, the way the NN learns, the calculation of the dZ^j happens in between layers, then the gradients ($\frac{\partial J}{\partial \Omega^j}$, $\frac{\partial J}{\partial B^j}$) are calculated, and then the parameters (Ω^j, B^j) are updated with the information from the learning examples.

The process of training a NN is summarized in figure 2.8. Part a) shows the forward propagation, while part b) shows the backward propagation. The cycle of learning, starts with the input of the feature representation, of the learning examples stored in X , the input propagates through the NN, until the output A^3 is produced, then the cost function is evaluated $J = \frac{1}{2N}(\vec{Y} - A^3)^2$, where \vec{Y} are the target values of the learning examples. After the evaluation of the cost function the update of the parameters

$\theta = \{\Omega^j, B^j\}$ by back propagation starts. First the calculation of dZ^3 is communicated to the output layer, to calculate the gradients of the cost function, respect to the parameters of the output layer $(\frac{\partial J}{\partial \Omega^3}, \frac{\partial J}{\partial B^3})$, latter the parameters are updated, and the algorithm moves to the next backward layer, and repeats the steps, calculate dZ^j , calculate gradients $(\frac{\partial J}{\partial \Omega^j}, \frac{\partial J}{\partial B^j})$, and update parameters. This process is repeated until the parameters in the input layer are updated.

One cycle of forward and back propagation makes one training step, some times the number of learning examples overflows the memory of the system, in those cases the number of examples are divided in batches, and every learning step is carried out in every batch at the time. A cycle over all the batches is an epoch. It is normal to have thousands of training steps. Sometimes after many training steps the value of the cost function evaluated in the validation set start to increase instead of decrease, when this situation happens, it means, that the NN is over fitting the training data, and it starts to lost the generality, needed to make predictions out of the training set. On the other hand when the cost function evaluated in the training set does not decrease or decrease a little to then reach a valley, it means that the NN is lacking the complexity, needed to reproduce the function of the learning data, in this cases the number of the parameters and/or the number of layers should be increased. In order the make the learning process faster, it is usual to use the Rectified Linear activation function, instead of the Logistic or the Tanh, looking at figures 5, 6, and 7 is clear that the derivatives of the Logistic

and Tanh have bigger values, only for $-1 < x < 1$, instead the derivative of the Rectified Linear is 1 for $x > 0$, then the learning process is faster using the Rectified Linear activation function.

2.3.4 Convolution Of Neural Networks

The convolution of neural networks is a widely used neural network architecture in the field of machine vision,[4] where a normal neural network is fed with the output of several layers of convolutions, the convolutional layers are a filtering process in between the input images and the neural network, the goal of the convolution is to extract some important property out of the raw images.

As an example figure 2.9 shows the process of a convolution layer. In the first stage there is a filter and a raw image, both represented by matrices, it is important to note that the filter is smaller than the image, the goal of the filter is to block some pixels and let through other ones. In the second stage the filter is applied to the image, note that the same filter is applied to different sections over the same image, in this case the filter extract the non-diagonal elements of the subimages matching the application, the extraction is carried out by an element wise multiplication between the elements of the image and the elements of the filter that share position, then the output of every individual application of the filter is added together, and concatenated into a feature vector. Keeping the output of every filter separate, helps the feature vector to keep some of the original geometrical information of the raw

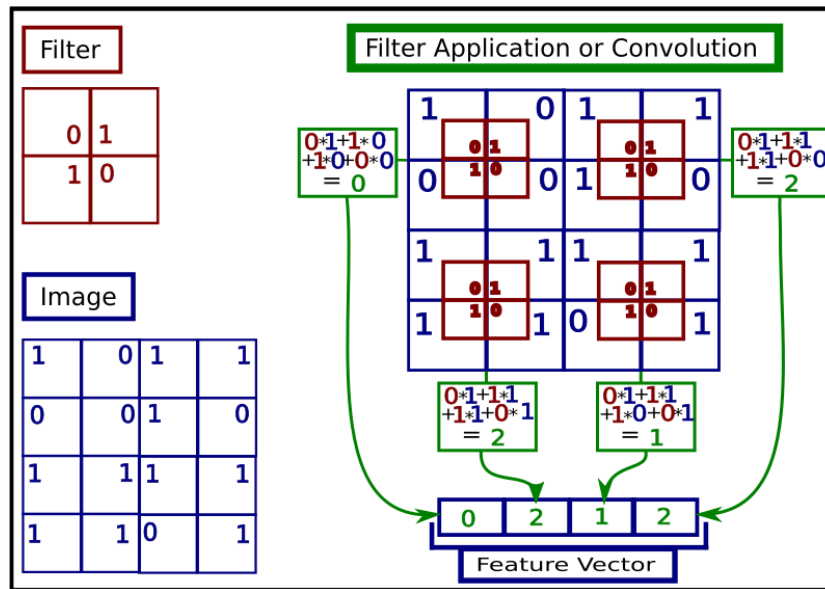


Figure 2.9: Diagram of the convolution of NN, in the left part, there are a filter and input image (matrix of pixels). In the center part the filter is applied to the image. In the right part the products of the filtering are order in a feature vector.

image. The convolution of neural networks is an example of an application feature selection in which the raw data is filtered in a way that the features conserve inside information of the original system in this case geometrical information.

2.4 Regression Trees And Gradient Boosting Regression

In most machine learning methods, the goal is to exhaustively train a single mapping $f : \vec{x} \rightarrow y$, to minimize the error between the target values "y" and

the predictions $f(\vec{x})$. In Boosting Methods the goal is still the same, reduce the error between "y" and $f(\vec{x})$, but with one difference, instead of investing all the resources in training a single mapping, the Boosting Method takes "M", under trained, and simple mappings, and combined them together to create a new map. In Boosting Methods, the final mapping is $F(\vec{x})$, and the individual simple mappings, or estimators are $f_m(\vec{x})$, the general idea of Boosting is expressed in the equation:

$$F(\vec{x}) = f_0(\vec{x}) + \sum_{m=1}^M \alpha_m f_m(\vec{x}) \quad (2.32)$$

In the machine learning literature the $F(\vec{x})$, mapping with error close to 0, is known as a strong estimator, or in the case of classification a strong classifier, the under-trained mappings $f_m(\vec{x})$ with high errors, are known as weak estimators or in the case of classification weak classifiers. Topically the weak estimators are a type of simple mapping known as regression tree, similar to a decision or classification tree, but used to predict values of a variable in \mathfrak{R} , instead of predict classes.

This subsection is divided in two parts, first an introduction to regression trees, and then an explanation of the Boosting method applied to regression trees and optimized with gradient methods.

2.4.1 Regression Trees

A regression tree (RT) [3] is a method to learn the value of a function, but instead of optimizing a preconceive parametric mapping, the RT learns by systematically dividing the feature space in rectangles, and assigning a constant value to everyone of the rectangular regions, this process, of recursively dividing the feature space, it is done by growing the RT using binary splits of the data.

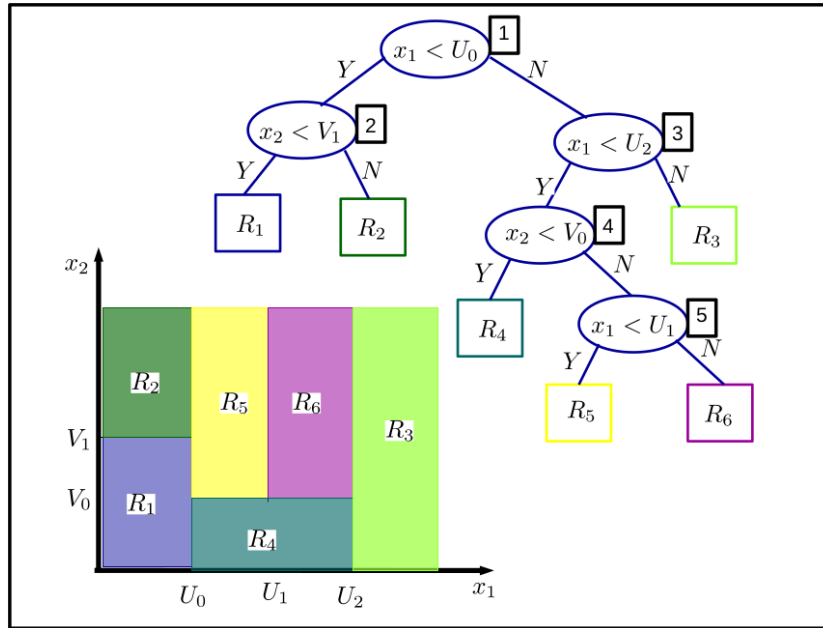


Figure 2.10: Scheme of a regression tree RT, and how the feature space (x_1, x_2) is divided in 6 regions. Every node shows the binary splitting process, and how the tree grows, by successive binary splitting every node, until the stopping criterion is reach, in which case the node became a leaf, and no subsequent splitting is done.

To introduce the process of growing the RT [2], consider learning a RT

from "N" examples of the form $\{\vec{x}_n, y_n\}$, where every \vec{x} is represented in a feature space of the form $\vec{x} = (x_1, x_2)$. The RT are grown from top to down, in series of binary splittings (as it can be seen in figure 2.10). At first all the "N" training examples are in the first node (the root node, node 1) where the growing algorithm makes the first binary split, the split is done in one of the feature coordinates, with respect to a threshold value, how the feature and the threshold value are chosen will be addressed later, for now, in the first binary splitting the feature selected is x_1 and the threshold value is U_0 , the examples where $x_1 < U_0$ advance through the left side branch, or yes branch, while the examples where the statement is not true advance through the right side branch, or no branch. In either case the data goes to their respective next node. For the data in the left hand side of the tree, the next node (node 2) makes the binary splitting on the x_2 feature, with V_1 as threshold value, after this partition the data goes to their respective terminal nodes (leaves). Every leaf R_m represents a region in the feature space. The RT makes the regression by assigning the same output value " γ_m " to all the examples in the same R_m region. The set of all the R_m regions, and all the s splittings, needed to create the tree, are known as the θ parameters of the RT $\theta = \{R_m, s\}$, in addition, the mapping done by a RT with parameters θ is usually referenced as $T(\vec{x}, \theta)$. Then for a RT with "M" total regions the prediction values are:

$$y'_n = T(\vec{x}_n, \theta) = \sum_m^M \gamma_m I(\vec{x}_n \in R_m) \quad (2.33)$$

Where $I(\vec{x}_n \in R_m) = 1$ if the hypothesis inside is true and it is 0 other wise.

Now in order to grow a tree in a systematic manner, there are tree points the algorithm must address [3]

1. A rule to split the data at every node, this is how to choose the feature (coordinate) to split and the value of the threshold to do the division.
2. A rule to determine when a node is terminal, when a node became a leaf.
3. A rule to assign the value of every γ_m .

The algorithm assumes that there are "N" training examples of the form (\vec{x}_n, y_n) where \vec{x}_n is a feature vector with "d" components $\vec{x}_n = (x_{n1}, \dots, x_{nd})$. The first point to be addressed is the third one, Here the algorithm makes two assumptions: first, at the end the data is grouped in M different regions, and second the cost function for minimization has the form $J = \sum_{n=1}^N \frac{1}{2} (y_n - \sum_m^M \gamma_m I(\vec{x}_n \in R_m))^2$, under this conditions the value of γ_m that minimizes J is:

$$\gamma_m = \text{aver}(y_n | \vec{x}_n \in R_m) \quad (2.34)$$

Now with respect to the rule to split the data in a node, it was shown by Hyafil and Rivest [24] that an optimal solution to a decision tree is an NP problem, as consequence building an optimal decision tree is unrealistic, then the splitting problem is solved, with a greedy approach, where at every node, the algorithm looks for the best split at that point, regardless of whether,

that particular split, is going to lead to a good split latter down in the tree. Then at every node the splitting problem is reduced to finding the x_p feature (coordinate) and the s value for which the splitting reduces the J function at that particular node. The splitting cuts the x_p coordinate in two planes: region 1 $R_1(p, s) = \{\vec{x}_n | x_{np} \leq s\}$ and region 2 $R_2(p, s) = \{\vec{x}_n | x_{np} > s\}$, then the best splitting is:

$$\operatorname{argmin}_{p,s} \left[\operatorname{argmin}_{c_1} \sum_{\vec{x}_n \in R_1(p,s)} (y_n - c_1)^2 + \operatorname{argmin}_{c_2} \sum_{\vec{x}_n \in R_2(p,s)} (y_n - c_2)^2 \right] \quad (2.35)$$

Where $c_m = \operatorname{aver}(y_n | \vec{x}_n \in R_m(p, s))$, $m \in \{1, 2\}$. The greedy algorithm looks for the best split (p,s), by scanning all the possible "s" values, for either all the "p" coordinates, if the feature space is not too big, or a randomly generated subset of features, if the feature space is too big. After knowing which particular splitting reaches the biggest minimization of J , that splitting is carried out, and the same process is done in the next nodes.

Finally, there are two mainstream stopping criterion, one is by fixing the number of total nodes in the tree, at the beginning of the training process, the other one is by defining the minimal number of training examples at every terminal node, such that once a node reaches, that amount of examples, it automatically became a leaf. It is important to note that, regression trees are easy to over fit the data, this means that if a tree grows to big, with too many nodes or too few examples per leaf, then it would have a low error in the training set, but a high error in the validation set. To avoid the over fitting

problem, many trees are grown to a tall structure but then they are pruned to accommodate a smaller number of nodes. Another common procedure is to grow several different trees, with different numbers of nodes, and selecting the one with the small error and fewer nodes, as the final regression tree.

2.4.2 Gradient Boosting regression

Recalling the beginning of this section, where the Boosting method [6, 2] was introduced in equation 32, as a precise collective regression method, built by integrating several under trained MLA. In this subsection the goal is to show how to construct a Boosting method out of regression trees, and training the collective regression method using a gradient approach.

In terms of regression trees $T_m(\vec{x}_i, \theta_m)$ the boosting regression mapping looks like:

$$F(\vec{x}_i) = T_0(\vec{x}_n, \theta_0) + \sum_{m=1}^M \alpha_m T_m(\vec{x}_i, \theta_m) \quad (2.36)$$

Where $T_m(\vec{x}_i, \theta_m)$ is a regression tree with regions and splitting parameters $\theta_m = \{R_{i,m}, s_m\}$, α_m is the shrinkage parameter which has the role of a learning rate, and "M" is the total number of RT ($T_m(\vec{x}_i, \theta_m)$) to be used as basic regressors in the Boosting method.

The Boosting mapping of equation 2.36 is trained in an iterative manner with a gradient approach. For the case of regression it is common to define the cost function $J = \sum_{i=1}^N L(y_i, F(\vec{x}_i))$ as a sum of square losses $L(y_i, F(\vec{x}_i)) = (y_i - F(\vec{x}_i))^2$.

The iterative training fits the mapping $F(\vec{x}_i)$, by fitting one tree at the time, every training step seeks to find the "m" RT such that:

$$\operatorname{argmin}_{\theta_m} \sum_{i=1}^N L(y_i, F_{m-1}(\vec{x}_i) + T_m(\vec{x}_i, \theta_m)) \quad (2.37)$$

To minimize the cost function, its gradient is taken with respect to the $F(\vec{x}_i)$ mapping and evaluated with respect to the $F_{m-1}(\vec{x}_i)$:

$$g_{i,m} = \left. \frac{\partial L(y_i, F(\vec{x}_i))}{\partial F(\vec{x}_i)} \right|_{F(\vec{x}_i)=F_{m-1}(\vec{x}_i)}$$

Using $L(y_i, F(\vec{x}_i)) = (y_i - F(\vec{x}_i))^2$ the gradient became:

$$g_{i,m} = -(y_i - F_{m-1}(\vec{x}_i)) \quad (2.38)$$

Then the "m" tree is trained with $(\vec{x}_i, g_{i,m})$ instead of (\vec{x}_i, y_i) , once the "m" tree is grown the $F_{m-1}(\vec{x}_i)$ is updated with the rule:

$$F_m(\vec{x}_i) = F_{m-1}(\vec{x}_i) + \alpha_m T_m(\vec{x}_i, \theta_m) \quad (2.39)$$

Then in every new iteration, the learning algorithm is trying to predict the residues (gradients) of the step before, in this way the learning regardless of being slower is more robust.

With regard to over fitting, is important to recall that there are three important hyper-parameters for Gradient Boosting Regression: the total number of RT "M", the size of every RT, and the α . The α value can be adjusted by hand, taking into account that a smaller value is better than a bigger

value, since slower learners are more robust. To avoid over fitting the size of every RT must be small between 4 and 8 nodes [2] remembering that, the key behind the GBR is not a single strong regressor, but several weak ones, so it does not matter if every single RT has high error by it self. Then the important hyper parameter is "M" that is usually as high as a 1000, this depending on the diversity, of the data set it has to learn, but it should be taking care of not being to big that the GBR will over fit.

2.5 Mixtures of gaussians

The mixture of gaussians is a regression model. As any other machine learning model, the main idea is to learn a function $f(\vec{x}_i, \{\omega\}) = y_i$ from a set of "N" examples $\{\vec{x}, y_{target}\}$, with a set of $\{\omega\}$ parameters. As its name points out, the parametric model is an addition of gaussians:

$$f(\vec{x}_i) = \sum_{j=1}^N \alpha_j K(\vec{x}_i, \vec{x}_j) \quad (2.40)$$

Where the sum is over α_j are known as mixing parameters, and $K(\vec{x}_i, \vec{x}_j)$ represents the gaussian kernel measuring the similarity between the data points "i" and "j", $K(\vec{x}_i, \vec{x}_j) = e^{\frac{-|\vec{x}_i - \vec{x}_j|^2}{2\sigma^2}}$.

The minimization of the cost function (equation 2.1) with a regularization

term of the form $\lambda \sum_i \alpha_i^2$, leads to the minimization problem:

$$\min_{\alpha} \sum_i (y_{target,i} - f(\vec{x}_i))^2 + \lambda \sum_i \alpha_i^2 \quad (2.41)$$

The solution for the α values using a vector notation, $\vec{\alpha} = (\alpha_1, \dots, \alpha_N)$, $\vec{\lambda} = (\lambda_1, \dots, \lambda_N)$, $\vec{y}_{target} = (y_{target,1}, \dots, y_{target,N})$, and the kernel matrix $\mathbf{K} \rightarrow K_{i,j} = K(\vec{x}_i, \vec{x}_j)$:

$$\vec{\alpha} = (\mathbf{K} + \vec{\lambda}\mathbf{I})^{-1}\vec{y}_{target} \quad (2.42)$$

Compare with the other methods exposed in this chapter, the mixture of gaussians is less powerful, however it had been useful for some machine learning potentials, specially on molecular applications, as it is going to be shown in the next chapters.

Chapter 3

Density Functional Theory

3.1 The original problem, many body quantum mechanics

To know properties like energy, phonon spectra, stability, bond order, while studying a material with standard quantum mechanics. It is necessary to know the wave function of the system, the wave function is the solution to the Schrödinger equation taking into account the most important interactions. For a system with N nuclei, and n electrons, the total Hamiltonian looks like:

$$\begin{aligned}
H_{total} = & \sum_I^N \frac{-\nabla_I^2}{2M_I} + \sum_{I,J \neq I} \frac{1}{2} \frac{Z_I Z_J}{|\vec{R}_I - \vec{R}_J|} \\
& + \sum_i^n \frac{-\nabla_i^2}{2} + \frac{1}{2} \sum_{i,j \neq j} \frac{1}{|\vec{r}_i - \vec{r}_j|} \\
& + \sum_{I,i}^{N,n} \frac{Z_I}{|\vec{r}_i - \vec{R}_I|} \quad (3.1)
\end{aligned}$$

The total Hamiltonian is in atomic units, the terms in the first line refer to the kinetic energy ($\sum_I^N \frac{-\nabla_I^2}{2M_I}$) of the nuclei, and the Coulomb interaction between the nuclei ($\sum_{I,J \neq I} \frac{1}{2} \frac{Z_I Z_J}{|\vec{R}_I - \vec{R}_J|}$) at positions \vec{R}_I , and \vec{R}_J , with atomic numbers Z_I , and Z_J . The second line has the electronic terms, first the kinetic energy of every electron in the system ($\sum_i^n \frac{-\nabla_i^2}{2}$), and then the Coulomb interaction between electrons ($\frac{1}{2} \sum_{i,j \neq j} \frac{1}{|\vec{r}_i - \vec{r}_j|}$). The last line of the Hamiltonian is the Coulomb interaction between the nuclei and the electrons ($\sum_{I,i}^{N,n} \frac{Z_I}{|\vec{r}_i - \vec{R}_I|}$). The presence of terms like $\frac{Z_I Z_J}{|\vec{R}_I - \vec{R}_J|}$, $\frac{1}{|\vec{r}_i - \vec{r}_j|}$, $\frac{Z_I}{|\vec{r}_i - \vec{R}_I|}$ makes impossible to have single particle solutions, hence the wave function will be dependent on the positions of all the nuclei and electrons in the system ($\Psi(\{\vec{R}_I\}, \{\vec{r}_i\})$).

Since it is not possible to find a analytic solution for equation 3.1 [25, 26], several simplifications have to be done. Born and Oppenheimer (BP) introduced an approximation, that uncouples the electronics, and nuclei degrees of freedom [27], the approximation is based on the fact that, from the electronic perspective the nuclei are fixed, with this, the kinetic energy of the nuclei goes to zero, and the nuclei-nuclei Coulomb potential is a constant, and the

nuclei-electron interaction can be seen as an external potential. However, even with this simplification, the Hamiltonian (H_{BO}) is complicated, and for a system with many electrons it still lacks an analytic solution.

$$H_{BO} = \sum_i^n \frac{-\nabla_i^2}{2} + \frac{1}{2} \sum_{i,j \neq i} \frac{1}{|\vec{r}_i - \vec{r}_j|} + \sum_{I,i}^{N,n} \frac{Z_I}{|\vec{r}_i - \vec{R}_I|} + E_{Nuclei} \quad (3.2)$$

The next approximations to try to solve H_{BO} were introduced by Hartree and Fock. Hartree used an ansatz, assuming that the wave function of the system ($\Psi(\vec{x}_1, \dots, \vec{x}_n)_H$) can be modeled, using the product of one electron wave functions, or single particle orbitals ($\Psi(\vec{x}_1, \dots, \vec{x}_n) = \phi_1(\vec{x}_1) \dots \phi_i(\vec{x}_i) \dots \phi_n(\vec{x}_n)$), with this approximation the Hartree energy can be written as:

$$\begin{aligned} E_H &= \langle \Psi_H | H_{BO} | \Psi_H \rangle \\ &= \sum_i^n \langle \phi_i | \frac{-\nabla_i^2}{2} + V_{ext}(r_i) | \phi_i \rangle + \frac{1}{2} \sum_{i,j \neq i} \langle \phi_i \phi_j | \frac{1}{|\vec{r}_i - \vec{r}_j|} | \phi_i \phi_j \rangle \end{aligned} \quad (3.3)$$

The Hartree Hamiltonian is the result of the minimization principal the E_H , with respect to single particle orbitals ($|\phi_i\rangle$), assuming that the right single particle orbitals are those that make E_H minimal:

$$H_H = \frac{-\nabla_i^2}{2} + V_{ext}(r_i) + \sum_{j \neq i} \langle \phi_j | \frac{1}{|\vec{r}_i - \vec{r}_j|} | \phi_j \rangle \quad (3.4)$$

Here $V_{ext}(r_i)$ is the potential due to the nuclei.

As a result of the Hartree wave function ansatz, H_H , is a single particle Hamiltonian that can be solved self consistently, since the Hamiltonian acting on $|\phi_i\rangle$ depends on $|\phi_j\rangle$. The electron-electron interaction is mimicked by the term $\langle\phi_j|\frac{1}{|\vec{r}_i-\vec{r}_j|}|\phi_j\rangle$, this term can be seen as a mean field approach [28], where the interaction between electrons is substituted by the interaction between the "i" electron, and the effective field produced by the other electrons.

While the simplification introduced by Hartree was successful to transform the many-body problem to a single particle problem, it does not take into account the fermionic character of the electrons. As it is known, electrons are indistinguishable, as consequence, the wave function of a system of electrons must change sign every time the positions of two electrons are exchanged (Pauli exclusion principle). To take into account this constraint, Fock used as ansatz a wave function following the Slater determinant, with the one electron wave functions as basis set.

$$\Psi_{HF} = \frac{1}{\sqrt{n!}} \begin{vmatrix} \phi_1(\vec{x}_1) & \phi_1(\vec{x}_2) & \dots & \phi_1(\vec{x}_n) \\ \phi_2(\vec{x}_1) & \phi_2(\vec{x}_2) & \dots & \phi_2(\vec{x}_n) \\ \vdots & \vdots & & \vdots \\ \phi_n(\vec{x}_1) & \phi_n(\vec{x}_2) & \dots & \phi_n(\vec{x}_n) \end{vmatrix}$$

The Hartree-Fock energy is:

$$\begin{aligned}
E_{HF} &= \langle \Psi_{HF} | H_{BO} | \Psi_{HF} \rangle \\
&= \sum_i^n \langle \phi_i | \frac{-\nabla_i^2}{2} + V_{ext}(r_i) | \phi_i \rangle + \frac{1}{2} \sum_{i,j \neq i} \langle \phi_i \phi_j | \frac{1}{|\vec{r}_i - \vec{r}_j|} | \phi_i \phi_j \rangle - \frac{1}{2} \sum_{i,j \neq i} \langle \phi_i \phi_j | \frac{1}{|\vec{r}_i - \vec{r}_j|} | \phi_i \phi_j \rangle
\end{aligned} \tag{3.5}$$

Following the same method, as for the E_H , minimizing E_{HF} with respect to the single particle orbitals, the Hartree-Fock Hamiltonian is:

$$H_{HF} = \frac{-\nabla_i^2}{2} + V_{ext}(r_i) + \sum_{j \neq i} \langle \phi_j | \frac{1}{|\vec{r}_i - \vec{r}_j|} | \phi_j \rangle - \sum_{j \neq i} \langle \phi_j | \frac{1}{|\vec{r}_i - \vec{r}_j|} | \phi_i \rangle \tag{3.6}$$

Where the new term $\langle \phi_j | \frac{1}{|\vec{r}_i - \vec{r}_j|} | \phi_i \rangle$ is a consequence of the exchange symmetry obeyed by the electrons, this term has the particularity that is dependent on the single particle orbital, the Hamiltonian is solved for ($|\phi_i\rangle$).

While Hartree-Fock methods reach an acceptable performance for describing physical systems [29, 30], their treatment of the exchange and correlation is rather simplistic, then further improvements needs a more careful treatment of this term.

3.2 Density functional theory

The density functional theory (DFT) was developed in the 1960s by Kohn, Sham, and Hohenberg to solve the many body quantum problem exposed

before [31, 32], DFT resembles the Hartree-Fock method, however its derivation comes from a different idea. In the Hartree-Fock method, the ansatz of the many body wave function is the key to the simplification process, that transform the many-body Hamiltonian in an effective single particle one. On the other hand, DFT is derived from the idea that, the electronic density is the quantity that determines all the ground state properties of the electronic system, then DFT is a theory about a quantity dependent on a single variable, the electronic density ($\rho(\vec{r})$), instead of being a theory about the individual electronic states. The idea of calculating molecular properties using electronic densities come from the 1920s, with calculations made independently by Fermi, Thomas, and Dirac, however the Thomas-Fermi-Dirac model did not produce good results when applied to molecules [33].

The main steps into the utilization of the density to describe the many-body quantum problem come from the theorems presented by Hohenberg and Khon, the theorems are:

- For any system of interacting particles in an external potential $v_{ext}(\vec{r})$, the potential is determined uniquely, except for a constant, by the ground state particle density $\rho_o(\vec{r})$.
- A universal functional of the energy $E[\rho]$ in terms of the density $\rho(\vec{r})$ can be defined, valid for any external potential. For any particular $v_{ext}(\vec{r})$ the exact ground state energy of the system is the global minimum value of this functional, and the density $\rho(\vec{r})$ that minimizes the functional

is the exact ground state density $\rho_o(\vec{r})$.

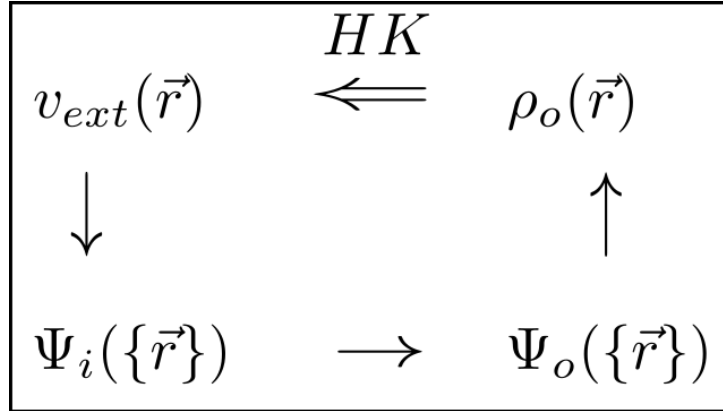


Figure 3.1: Diagram, showing the usefulness of the Hohenberg Kohn theorems. The single arrows show the solution cycle of the Kohn Sham equation, the external potential ($v_{ext}(\vec{r})$) defines the system, and all its states $\Psi_i(\{\vec{r}\})$, even the ground state ($\Psi_o(\{\vec{r}\})$). The double arrows show how the Hohenberg Kohn theorems link the ground state energy to the $v_{ext}(\vec{r})$ defining the system, this figure was taken from Ref [34].

However, the theorems only prove the existence of two things. First a universal energy functional, and second an electronic density that minimizes the energy functional, which is the truth ground state density. These theorems would have been a theoretical curiosity without the reformulation of the problem made by Kohn and Sham. The reformulation results in an auxiliary system, that is soluble, and shares some properties of the interacting many-body system.

The energy functional of the auxiliary system proposed by Kohn and Sham is:

$$E[\rho] = T[\rho] + E_{xc}[\rho] + \frac{1}{2} \int \int \frac{\rho(\vec{r})\rho(\vec{r}')d^3rd^3r'}{|\vec{r}-\vec{r}'|} + \int v_{ext}(\vec{r})\rho(\vec{r})d^3r \quad (3.7)$$

Where $T[\rho] = \sum_i^n \langle \phi_i | \frac{-\nabla_i^2}{2} | \phi_i \rangle$ is the kinetic energy of system of not interacting particles, $\frac{1}{2} \int \int \frac{\rho(\vec{r})\rho(\vec{r}')d^3rd^3r'}{|\vec{r}-\vec{r}'|}$ is an approximation of the electron electron energy known as the Hartree energy V_H (do not confuse with the Hartree energy from the last section), $\int v_{ext}(\vec{r})\rho(\vec{r})d^3r$ is the external potential due to the nuclei, and $E_{xc}[\rho]$ is the exchange correlation energy. Conceptually the exchange correlation can be expressed as:

$$E_{xc}[\rho] = T_{Int}[\rho] - T[\rho] + V_{ee} - V_H \quad (3.8)$$

The exchange correlation functional is the term accounting for all the approximations done in equation 3.7, it corrects for using the kinetic energy of a non interacting system, instead of the kinetic energy of an interacting system ($T_{Int}[\rho]$), it also corrects for using V_H instead of the exact potential interaction between electrons V_{ee} . On the paper the DFT functional expressed in equation 3.7 is an exact theory of the many-body problem since the exact E_{xc} introduces all the needed corrections, however, the exact form of the $E_{xc}[\rho]$ functional is unknown, then for real life applications it has to be approximated.

The electronic density is defined in terms of the f_i occupancy, and the

single orbital states $\psi_i(\vec{r})$:

$$\rho(\vec{r}) = \sum_i f_i |\psi_i(\vec{r})|^2 \quad (3.9)$$

Then the auxiliary problem proposed by Kohn and Sham materialize by minimizing the energy functional defined in equation 3.7 with respect the electronic density, which ends up with the Kohn-Sham equation:

$$\left(\frac{-\nabla_i^2}{2} + v_{ext}(\vec{r}) + \int \frac{\rho(\vec{r}') d^3 r'}{|\vec{r} - \vec{r}'|} + V_{xc} \right) \psi_i(\vec{r}) = \epsilon_i \psi_i(\vec{r}) \quad (3.10)$$

Which has the form:

$$H_{KS} |\psi_i(\vec{r})\rangle = \epsilon_i |\psi_i(\vec{r})\rangle \quad (3.11)$$

Where $V_{xc} = \frac{\delta E_{xc}[\rho]}{\delta \rho(\vec{r})}$. Equation 3.10 describe a system of individual particles, with the same (up to the exchange correlation functional) ground state density than the original interacting system, then by solving equation 3.10 and finding $\rho_o(\vec{r})$ the many body system is solved.

Finally to calculate the energy of the system with the solutions of equation 3.9, and the density defined in equation 3.9 the following formula is used:

$$E = \sum_i \epsilon_i - \frac{1}{2} \int \int \frac{\rho(\vec{r}) \rho(\vec{r}') d^3 r' d^3 r}{|\vec{r} - \vec{r}'|} + E_{xc}[\rho] - \int V_{xc}[\rho] \rho(\vec{r}) d^3 r \quad (3.12)$$

The first term adds the eigen-energies of all the occupied molecular orbitals, then the second term corrects for over counting the electron-electron interac-

tion, finally the exchange correlation is added and the effects of the exchange correlation potential in ϵ_i are taken away by subtracting V_{xc} .

3.2.1 Approximate exchange correlation functional

How the exchange correlation functional is approximated defines the success of a given application of DFT. A good functional is supposed to capture all the many-body effects, and handle the errors coming from not using the right kinetic energy, and electron-electron interaction.

The two dominant approaches to approximate the E_{xc} are: the local density approximation (LDA), and the general gradient approximation (GGA). The LDA was the path followed by Kohn and Sham originally, it assumes that the E_{xc} does not change abruptly with $\rho(\vec{r})$, then the functional can be written like:

$$E_{xc}[\rho] = \int \rho(\vec{r}) \epsilon_{xc}(\rho(\vec{r})) d^3r \quad (3.13)$$

Where $\epsilon_{xc}(\rho(\vec{r})) = -\frac{3}{4} \left(\frac{3}{\pi} \rho(\vec{r}) \right)^{(1/3)}$ account for the exchange and correlation per electron in an uniform electron gas [35, 31, 26].

The LDA approximation is an expansion where the only terms taken into account are the ones depending on $\rho(\vec{r})$, to increase the perturbation accuracy, the generalized gradient approximation (GGA) [36, 37, 38] include terms dependent on $\vec{\nabla}[\rho(\vec{r})]$, the GGA functional represent an improvement, specially in calculations on finite systems like molecules.

For a calculation, it is important to note that, there is not a single univer-

sal functional [39], then, the selection of E_{xc} depends upon the properties to calculate, as well as the atomic system (crystal, molecules, surface) on which the calculations are going to be performed.

The limitations on the current E_{xc} functionals, are the result of treating an interaction as a local one, while it is non-local in nature, both the LDA and GGA describe the E_{xc} as a function of $\rho(\vec{r})$ or $\vec{\nabla}[\rho(\vec{r})]$, this means that the point \vec{r} is only affected by the value of the density and its gradient at that given point, ignoring that a more faithful representation should be a function of \vec{r} and \vec{r}' , to capture the correlation with other parts of the system. However the development of a non-local representation is extremely complicated, and it is unlikely that a multipurpose potential will be developed in the near future [26].

3.2.2 DFT implementation

The following discussion is going to outline the main properties of the DFT implementation proposed by Sankey and Niklewski [40], since this method is the root of the FIREBALL software [41], with the LDA approximation [42, 43, 44] used for most of the calculations in this dissertation.

FIREBALL is a software, and a method to solve equation 3.10, find an approximation to the ground state density, and evaluate the energy functional of equation 3.12.

The basis set used by FIREBALL is composed by pseudo-atomic-orbitals (PAO) [45], this orbitals simulate the valence electron states of an atom in

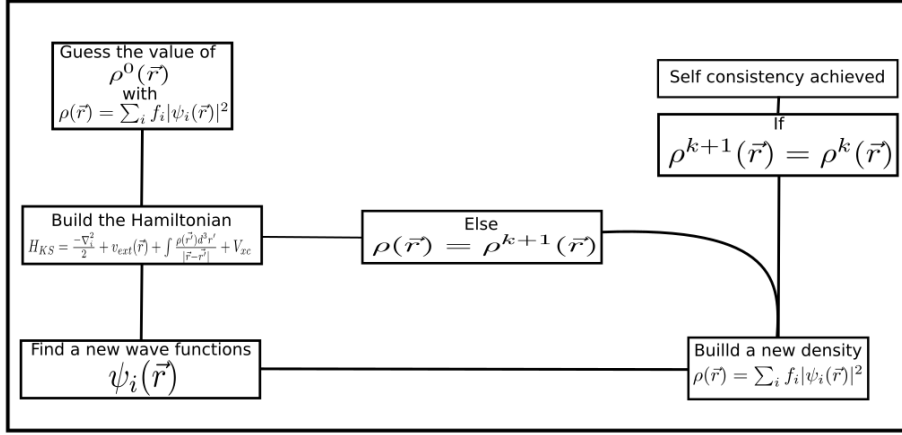


Figure 3.2: Diagram of the a self consistent cycle to solve the Kohn-Sham equation. First a density is guessed ($\rho^o(\vec{r})$), the H_{KS} is build with $\rho(\vec{r}) = \rho^o(\vec{r})$, the Hamiltonian is diagonalized to find a set of molecular orbitals $\Psi_i(\vec{r})$, this states in addition with the occupation number of every molecular orbital (f_i) a new density is calculated ($\rho^{k+1}(\vec{r})$), if the new density is equal to the old density (up to a threshold) self consistency is achieved and the calculation ended, if the densities are no equal, then the new density feeds te Kohn-Sham Hamiltonian and the process starts again until self consistency is achieved, the density from the last step 'K' is close to the ground state energy $\rho^K(\vec{r}) \approx \rho_o(\vec{r})$.

its neutral ground state, they are calculated using the Herman-Skillman [40] approach using pseudo-potentials, and a local density approximation for the exchange correlation. The boundary condition imposed over the PAO make them vanish after a certain cutoff radius, the effect of this confined that the orbitals are slightly exited.

The molecular orbitals in FIREBALL are expanded in terms of the PAO functions:

$$\psi_i(\vec{r}) = \sum_{l,\mu} a_i(l, \mu) \phi_\mu^{PAO}(\vec{r} - \vec{R}_l) \quad (3.14)$$

Where the "i" index counts the molecular orbitals, "l" counts the center of the atomic like orbital (usually the position of a nuclei), "μ" is the type of atomic orbital (s, p_x, p_y, p_z, etc).

To solve equation 3.10 a initial density is needed to initialize the Hamiltonian, in FIREBALL the initial density is the result of adding the neutral and spherical atomic densities of the the atomic like potentials ($\rho^0(\vec{r})$ is the initially guessed density), with this initialization, and substituting equation 3.14 into equation 3.10, then multiplying by $\phi_\nu^{PAO}(\vec{r} - \vec{R}_l)$, to have a system of algebraic equations to find the $a_i(l, \mu)$ coefficients.

$$\sum_{l', \nu} h_{\mu, \nu}^{l, l'} a_i(l', \nu) = \epsilon_i \sum_{l', \nu} S_{\mu, \nu}^{l, l'} a_i(l', \nu) \quad (3.15)$$

Where the elements of the Hamiltonian and the Overlap matrix are calculated like:

$$h_{\mu, \nu}^{l, l'} = \langle \phi_\mu^{PAO}(\vec{r} - \vec{R}_l) | H_{KS} | \phi_\nu^{PAO}(\vec{r} - \vec{R}_{l'}) \rangle \quad (3.16)$$

$$S_{\mu, \nu}^{l, l'} = \langle \phi_\mu^{PAO}(\vec{r} - \vec{R}_l) | \phi_\nu^{PAO}(\vec{r} - \vec{R}_{l'}) \rangle \quad (3.17)$$

The eigen-vectors and eigen-values are find with the secular equation:

$$\det |h - \epsilon S| = 0 \quad (3.18)$$

The Kohn-Sham equation is solved in a self consistent manner, where the solutions of equation 3.15 are use to construct a new density $\rho^1(\vec{r})$, this process is carry out "k" times to minimize the energy and find and approximation to

the truth ground density $\rho_o(\vec{r})$, in FIREBALL comparing $\rho^k(\vec{r})$ with $\rho^{k+1}(\vec{r})$ constitute the stopping criteria, for a given energy threshold.

3.2.3 Pseudo-potentials

Many of the physical properties of systems of atoms, like crystals, and molecules are due, primarily to the dynamics of valence electrons. Valence electrons are screened by the core electrons in the inner layers of atoms, then, they are less attached to their original core nuclei, having the freedom to interact with other cores. As a consequence the behavior close to the nuclei do not need to be over realistically represented. With this thought pseudo potentials are introduced to facilitate the description of the physics in materials.

The wave function close to the nuclei has higher frequencies than the wave function far from the nuclei where the behavior is more of a decay, image 3.2.3 compares an ionic potential V_1 and its pseudo-potential V_2 . The ionic potential produces the Ψ_1 wave function, also known as all electrons wave function. The divergence of the ionic potential results in an all electron wave function with rapid oscillations inside the core region of the atom. An accurate description of these oscillations has a limited impact on molecular calculations, but they require several basis functions for its description, making the calculations harder to carry out. The pseudo-potential on the other hand, makes a faithful representation of the physics in the valence region, so that, the pseudo-wave function Ψ_2 is indistinguishable from the all electron

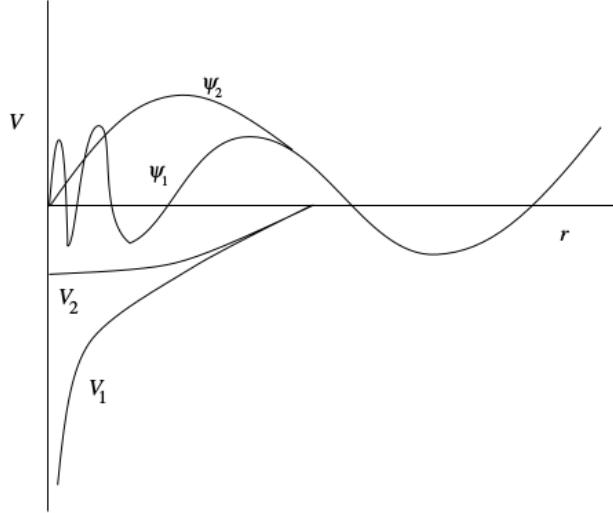


Figure 3.3: V_1 represents the all electrons ionic potential. Ψ_1 is the wave function resulting from solving the Schrödinger equation with V_1 as potential. V_2 represents the pseudo-potential. Ψ_2 is the wave function resulting solving the Schrödinger equation with V_2 as potential.

wave function after certain cut off radii r_c , where the valence properties are more important than the core properties. The pseudo wave function has no nodes inside the core region, and it is easy to describe with fewer basis functions. One constraint over Ψ_2 is it the total charge in the core region must be the same whether it is described with Ψ_2 or Ψ_1 . The pseudo-potential is calculated for every element, by taking into account an isolated atom. Then the resulting pseudo-potential is used to represent the ionic potential of that given element in a DFT calculation.

The pseudo-potentials used in FIREBALL are the separable and non-local, more details of its implementation in Ref [46, 42].

Chapter 4

Force Fields

4.1 Introduction

Many studies of material properties for large and complex materials like glasses and biological macromolecules are carried out with computational simulations like molecular dynamics (DM). The key component of the computational simulations (aka MD) are the force fields (FFs)[47, 48, 49], FFs are the intellectual parents of machine learning potentials, in the sense that several properties and parts of the machine learning potentials are heavily influenced by force fields. A FF is a parametric function, no more complex than a polynomial, from which the energy and forces of a system of particles can be easily evaluated, only knowing the set of positions $\{\vec{R}_n\}$, and species $\{Z_n\}$ of all the atoms in the system, unlike any ab initio method, where the calculation of energies and forces require the solution of complex partial

differential equations. The main idea behind the FF is to fit the parameters of the parametric functions to reproduce benchmark data obtained from experiments, or ab initio calculations. After the fitting process, the FF is able to estimate the energies and forces of systems similar to the ones used for fitting the parameters. However if the systems processes by the FF are far from the configurations used for fitting the parameters, or if the complexity of the system's potential energy surface (PES)¹ is higher than the complexity the parametric function can handle, then the FF is not going to estimate the energies and forces accurately. To summarize the accuracy of the FF is dependent on the number parameters of the parametric functions used to represent the PES, and the data used to fit those parameters, and in general, the FFs have small areas of prediction in the configuration space.

This chapter is devoted to the force fields, section 4.2 introduces the FF in a more formal manner, in addition, every part of the FF is explained in the subsections of section 4.2. Section 4.3 is devoted to the process of fitting the parameters of the force field, this process is also known as the parametrization of a force field, this section is a review of the specific techniques used to fit every family of parameters. Finally as a form of conclusion section 4.4, talks about the limitations of the FFs and puts in perspective the Machine Learning Potentials as a solution to the problems of FFs.

¹The Potential Energy Surface of a system of "N" atoms is a manifold in a space of 3N coordinates, describing the energy of the system as function of the positions of the "N" constituent atoms $E(\{\vec{R}_1, \dots, \vec{R}_N\})$

4.2 Force Fields: Functional Form

A force field is a parametric function, that approximates the PES for certain regions of the configuration space of a structure². The main idea supporting the functional form of the FFs is that the total energy of a structure can be divided into quasi-independent terms, every term representing a type of interaction added to the total energy. The most common energy decomposition for FFs is [49]:

$$E_{structure} = E_{bonds} + E_{angle\ bending} + E_{torsion} + E_{electrostatic} + E_{Van\ der\ Waals} \quad (4.1)$$

Equation 4.2 is the prototypical expression for popular force fields like: AMBER [50], OPLS [48, 51], CHARMM [52], and GROMOS [47]. Every one of the terms represented has an specific functional form (polynomial) based on physical insights about the interaction it is meant to reproduce. In the next subsections all of this energy terms will be described in deepness.

4.2.1 Bonding Energy

The first term in equation 4.2, is the energy stored in the stretching of the bonding between two atoms, based on physical experience, the bonding interactions resembles (in a first approximation) a harmonic potential, the variable defining the value of the energy in the harmonic potential is the in-

²Structure: In this dissertation, a structure will refer to crystals, molecules, clusters, and in general any system ordered or disordered constituted with atoms

teratomic distance between the atoms making up the interaction, the actual form for the parametric function describing the E_{bond} is:

$$E_{bond} = \sum_{bonds} k_r (r - r_0)^2 \quad (4.2)$$

Every component of the bonding energy is defined by its force constant k_r and its equilibrium bonding distance r_0 . Figure 4.2.2 part A, shows different configurations of bonding, that need different values of the force constant to be properly described, the value of the force constant change regarding the species involved in the bonding, as well as the quantum properties of the bonding. The final value of the E_{bond} is the result of adding all the bonding interactions, up to a cutoff radius, or bonding criteria, for example in some cases only bonding among near neighbors is considered.

4.2.2 Angle Bending Energy

The second term in the energy takes into account the energy stored in the bending of the angle defined by three atoms (1,2,3), where atom 1 is in the vertex, then the mathematical expression for the angle is: $\theta = \cos^{-1}\left(\frac{\vec{R}_{12} \cdot \vec{R}_{13}}{|\vec{R}_{12}| |\vec{R}_{13}|}\right)$. The angle bending energy functional form is also a harmonic potential of the angle θ .

$$E_{angle \ bending} = \sum_{angles} k_\theta (\theta - \theta_0)^2 \quad (4.3)$$

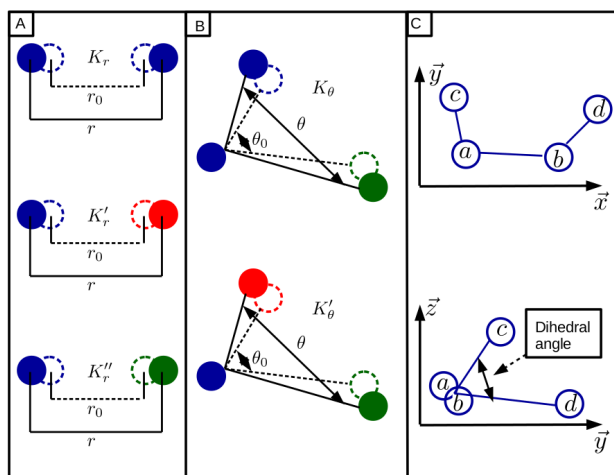


Figure 4.1: A) In rectangle "A" three different classes of bonding, showing how every kind of bonding stretching interaction has its own k_r parameter, for instance, the upper image K_r models the bonding interactions between "blue" atoms, K'_r models the interaction between "blue" and "red" atoms. In addition r_0 is the equilibrium position, and r the distance between the atoms in the bonding. B) rectangle "B" shows the angle interaction, where trios with different kind of constituents needs different K_θ for a proper description. Here θ_0 is the equilibrium angle, and θ is just the angle defined by the three atoms. C) Description of the dihedral angle, in the upper part a molecule with three atoms a, b, c, d, over the (\vec{x}, \vec{y}) plane is seen from \vec{z} , the atom "a" and "b" are bonded, as well as "a" and "c", and "b" with "d", the dihedral angle is the angle between the (a,c) and (b,d) bonding, projected over the plane intersecting the (a,b) bonding, in the case of the lower image that plane is (\vec{y}, \vec{z})

The parameters are the force constant k_θ and equilibrium bonding angle θ_0 , similarly to the E_{bond} not all the possible angles are taken into account only those angles formed by first or second bonding neighbors. Figure 4.2.2 part B shows different realizations of the angle bending interaction, in which different values of the k_θ force constant are necessary to properly describe the interactions.

Finally the θ_0 is the angular equilibrium position, due to the symmetry of the angular bending, one single interaction described by one k_θ , may need several values of θ_0 to account for all the different equilibrium positions the system can have, it is clear that the harmonic potential does not describe the interaction in a faithful manner, however the complexity of a realistic description carries a heavier cost than the numerical error introduced by this simplification.

4.2.3 Torsion Energy

The third term in the total energy expansion equation 4.2 is the contribution due to the torsion (rotations) of the dihedral angle. The dihedral angle is the angle defined by subsets of 4 atoms (a,b,c,d), the bonds of the set of atoms are $\{(c, a), (a, b), (b, d)\}$, the dihedral angle is the angle formed by the intersection of two planes, plane 1 containing (c,a,b) and plane 2 containing (a,b,d) as the figure 4.2.2 part C shows.

The energy as result of perturbing the dihedral angle is a periodic function [53], therefore the representation in the force field is a Fourier series of the dihedral angle ϕ , with parameters: k_ϕ amplitude, n multiplicity of the dihedral angle, and δ phase angle. The $E_{Torsion}$ is the last of the internal or intermolecular terms (bond, angle bending, torsion)

$$E_{torsion} = \sum_{dihedrals} k_\phi(1 + \cos(n\phi + \delta)) \quad (4.4)$$

4.2.4 Electrostatic Energy

The electrostatic energy is the next component in the FF expansion. The usual Coulomb potential is employed to model the interaction, where the atoms in the systems are supposed to be point-like charges, the total contribution of the electrostatic energy is:

$$E_{electrostatic} = \sum_{i < j} \frac{q_i q_j}{r_{ij}} \quad (4.5)$$

As usual " q_i ", " q_j " represent the charges of atoms "i" and "j", r_{ij} is the distance between the same atoms, the summation takes into account bonds between first and second neighbors, in some cases other degrees of neighboring are used but in those cases the charges are re-scaled to represent the screening effect of electrons and other atoms. [54].

4.2.5 Van der Waals Energy

The Van der Waals (VdW) is the last term in the FF energy expansion, the VdW interaction is usually model by a Lenard Jones (LJ) potential:

$$E_{Van\ der\ Waals} = 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right] \quad (4.6)$$

The VdW interaction is a simplified representation that describes what happens when two atoms with electronic clouds interact with each other. At first if the two atoms are far apart, but start getting closer, their electronic

clouds move to form two dipoles, the dipole-dipole interaction is attractive at first, and described by the $\left(\frac{\sigma_{ij}}{r_{ij}}\right)^6$ term, if the atoms keep getting closer, then the dipole-dipole attraction lose preponderance compared to the repulsion between the nuclei, this is when the $\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12}$ term is predominant. The parameters of the LJ potential are the ϵ which is the depth of the potential well, and σ_{ij} is the distance between atoms "i" and "j" at which the attractive and repulsive forces balance each other.

4.3 The Force Field As A Whole

The PES is a complex manifold in a space of $3N$ (N number of particles in the structure) dimensions. An actual representation of the PES is (so far) only possible for small systems [55]. For realistic systems, the FFs are oversimplifications of the actual PES. However it does not mean that FFs are simple to calculate, a FF for a system with many particles and/or different species requires thousands of parameters to even describe the system around the equilibrium configurations, for example, the FF OPTLS 2005 [51] has around 6000 parameters, 1054 for bonding, 3997 for angle bending, and 1576 for torsions.

It is hard to think where do all the parameters come from by only looking at equation 4.2, but the reality is that every one of the energy terms in equation 4.2 is encapsulating many different interactions of a specific kind. For exam-

ple, figure 4.2.2 part A, shows 3 different contributions to the E_{bond} , every one of this bond interactions would need a different K_r to have a proper description $K_r \rightarrow (K_r, K'_r, K''_r)$. Besides figure 4.2.2 part A only shows cases where the differences in bonding are the result of different species in the interaction, however, even interactions with the same species, may need different parameters due to changes in the properties of the interactions, for instance, a C-C bonding is different, regarding whether the atoms have a π or a σ bond, and also which hybridization is involve. In general, the diversity of parameters is not only a consequence of the interactions between different species but also due to differences in the quantum properties of the interactions.

The FF is then a recipe to build the energy of a structure, every one of the terms is an ingredient representing and specific interaction. The particularities of the interactions define a chemical environment, the goal of the FF is to learn as many different chemical environments as possible, however, if the chemical environments became too specific then the FF would lose transferability to reproduce other configurations in the phase space of the PES.

A FF able to describe many chemical environments need a large number of parameters, the value of the parameters is selected to reproduce benchmark data from previous experimental and theoretical results. The procedure to fit the FF parameters to the benchmark data is known as the parametrization of the FF, the next section gives a glance of how a standard parametrization looks like.

4.4 Parametrization Of A Force Field

Similar to the learning process of a machine learning algorithm (described in chapter 2), the FF parameters are selected to reproduce training data. However, unlike the machine learning training, the fitting of the FF parameters is done by training the parameters separately depending on the interaction the parameters are supposed to reproduce, this means that the fitting of the bonding parameters is carried out independently and with different data than the training of the torsion parameters, which increases the difficulty of fitting FF. In addition the ordering of the fitting process is important, due to correlations between the different components in the energy expansion. The subsequent sections are going to explain the different stages of the Parametrization of a FF, following the usual order of parametrization.

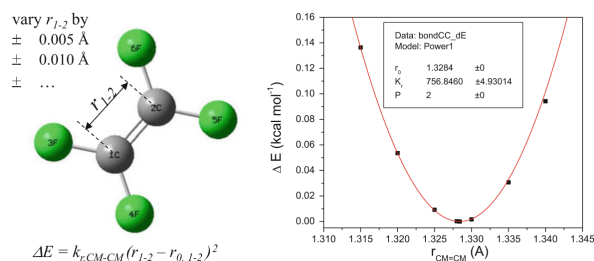


Figure 4.2: Image taken from [54] page 151. The energy of a small molecules is calculated as function of the distance between to atoms of carbon, the goal is to find the value of the K_r for a carbon carbon interaction, in this case the cause the changes in energy is isolated to the change in distance between the carbon dimer

4.4.1 Parametrization For The Bonding Energy And The Angle Bending Energy

The fitting process starts with the selection of the equilibrium positions (r_0, θ_0) [56]. In the case where the goal of the FF is to describe molecules, it is usual to obtain the value of the equilibrium positions from isolated ab initio calculations of dimers representing the chemical environment of interest. In the case where the FF is meant to describe bulk materials, then the equilibrium positions come from experimental data like X-ray diffraction patterns. For the force constants, (k_r, K_θ) the fitting process follows a similar trend. In the case where the FF is going to be used to calculate properties of complex molecules, the optimization of the force constants is done by directly calculating the energy of a smaller molecule containing the bond or angle of interest, in this approach the energy is calculated as a function of different values of bond distance or bending angle, then the k_r or K_θ are chosen to reproduce the data, as shown in image 4.4. On the other hand when the force constants are meant to reproduce bulk materials, then ab initio methods are used to calculate the vibrational spectra of specific normal modes, assuming that the excited normal modes are directly related to the value of the force constants k_r and K_θ , in some cases, is not the vibrational modes but the elements of the Hessian matrix the ones used to fit force constants. In general, the main idea behind the selection of the value of a parameter is to reproduce a physical quantity (vibrational spectra), that is link exclusively

to the interaction represented by the fitted parameter.

Parametrization Of The Electrostatic Energy

The next parameters in the fitting process are the atomic charges of the electrostatic interactions. The usual method to find the value of the atomic charges is known as CHELP [57]. In the CHELP an ab initio method calculates the electrostatic potential of a model structure, that resembles the chemical environment of the system of interest (where the FF will be applied), after the ab initio method gives the value of the electrostatic potential for a set of interest point around the model structure, the goal is to reproduce the potential, with a set of point-like charges distributed over the atomic positions. The calculation of the charges values uses a least square method to replicate the ab initio electrostatic potential.

4.4.2 Parametrization Of The Lenard Jones Potential (Van der Waals Energy)

The parametrization of the Lennard Jones parameters is different from the last terms (E_{bonds} , $E_{anglebending}$, $E_{electrostatic}$), the fitting strategies rely on the calculation of physical quantities (electrostatic potential, vibration spectra) related to parameters in smaller model system, however due to the many-

body character and the correlation with other terms, the parametrization of the Lennard Jones parameters follows an iterative approach, where the FF with the $(E_{bonds}, E_{anglebending}, E_{electrostatic})$ starts working.

The parametrization process starts with the initialization of the Lennard Jones parameters, the initial values come from former Lennard Jones parameters used for a similar system. Then the FF field start doing molecular dynamics with the $(E_{bonds}, E_{anglebending}, E_{electrostatic})$ parameters, plus the Lennard Jones, that was just initialized. The molecular dynamics from the FF is compared with an ab initio molecular dynamics, the tuning of the Lennard Jones parameters account for the differences between the results from the FF and the ab initio method. This approach of optimizing the Lennard Jones parameters by initializing them, with parameters from similar systems was introduced for the development of the OPLS [48] force field.

4.4.3 Parametrization Of The Torsion Energy

The torsion parameters are the more difficult in the optimization process Raabe2017, the difficulties arise from the fact that the torsion term is a Fourier series, in addition, the torsion energy is correlated with most of the other terms. The optimization starts by calculating the benchmark data for the fitting process, the data comes from an energy calculation with an ab initio method performed on a model system with a similar chemical environment, the chemical environment of the model system must resemble the

chemical environment of the system in which the FF will be used. In specific the ab initio method calculates the energy of a smaller model system as function of the dihedral angle of interest, however, due to the correlations between terms, the torsion energy parameters are not directly fitted to the energy calculated by the ab initio method, instead, the energy calculation as a function of the dihedral angle is carried out again with the FF without the torsion energy terms. Finally, the fitting process fits the torsion energy terms to the difference of the ab initio energies and the FF energies.

4.5 Force Fields Conclusions

Force fields are simplified mappings of the PES allowing researchers to predict physical and chemical properties of complex materials like proteins and RNA under realistic conditions [47, 58, 59]. Most of these calculations are still impossible to carry out with ab initio methods since the complexity of the simulations would require larger amounts of computational resources than the ones in hand of current researchers. Among their advantages, FFs have a relatively high accuracy around equilibrium configurations, making the FFs able to describe the dynamics of systems with similar chemical environments around the equilibrium point. How similar must the chemical environments be? It is a matter of the transferability of the FF, but as long as the chemical species and concentrations resemble the ones from the training data, then the FF should be accurate enough.

However, regardless of their widespread use and popularity, force fields have limitations. The biggest limitation is related to their functional form, for example, the case of the bonding energy the only interactions taken into account are the ones where $r < r_{cutoff}$, in which the harmonic approximation holds. The angle bending interaction is also model like a harmonic potential, despite the fact that the interaction has explicit periodicity[54]. In addition, the charges in the electrostatic interactions are assumed to be point-like charges, when the reality is that the charge distribution in a structure is hardly spherically symmetric. In consequence, the simplified functional form can only describe small regions of the PES, close to equilibrium configurations, leaving disordered structures out of the reach of FF [60, 61].

The limited reach of FFs is a well-known problem, and the improvement of FFs is an active field of research, attempts to improve and redesign the FFs are proposed in [58, 62], however, the improvement attempts face another intrinsic disadvantage of FFs. To increase the reach and accuracy the FF must increase the number of parameters and the complexity of the model, making the training process even more difficult.

The process to fit the parameters of the FF resembles more an art than a science, there is a particular order for the parameters to be fitted, and the data to fit the parameters is a mix of experimental, and ab initio calculations carefully crafted depending on the interaction needed to describe, so that, an increment on the number of parameters or an increase in the complexity of the function would make the FF as hard to parametrize as it is to calculate

ab initio properties for complex systems.

Under this circumstances the physical community has moved on to develop new FFs, but instead of assuming a physically motivated functional form, the new FFs are powered by machine learning algorithms, this approach solves the problem of increasing the difficulties of training when increasing the complexity of the mapping in the model, plus the training methods for machine learning algorithms are standard. However, the solution to this problem brings new problems to the field. In the next chapter, the Machine Learning Potentials are introduced.

Chapter 5

Machine Learning Potentials

5.1 Introduction To Machine Learning Potentials

Machine Learning Potentials (MLPs) are the next step in the evolution of force fields [FF]. MLPs as the FF try to approximate the PES using less demanding methods than the ab initio approach. However, and in spite of sharing the same goal as FF, MLPs are not based on physically motivated approximations of the total energy, instead the MLPs rely on machine learning method to directly estimate the energy and forces of an input structure.

The last chapter shown how trying to increase the accuracy, or the reach of FF predictions, would lead to an increment of parameters, as well as an increment in the complexity of its functional form, therefore the already difficult process to fit the parameters of the model, would be even more dif-

difficult. Using machine learning algorithms, instead of physically motivated but simplified potentials, directly skips the limitations of FF. Machine learning mappings are complex enough to mimic any function (*restrictions apply). Moreover, MLPs learn directly from calculations of the total energy and forces, with no need to split the benchmark calculations on independent terms regarding different degrees of freedom.

Nevertheless, while it is true that MLPs do not have the same problems of FF, MLPs have their own flaws. For example; it was just said that they "can mimic any function", the reality is that to reproduce complex functions machine learning algorithms need massive amounts of data to learn from. Moreover, in spite of being able to calculate the PES for small molecules [63, 64, 65] and crystals [66, 67], the description of disordered materials is still a challenge for MLPs [68, 69], in addition, to the best of our knowledge, MLPs have not been employed yet to carry out simulations of complex systems like proteins.

The process of building a MLPs can be divided in two steps. First the preprocessing of the data. This stage is concerned with transforming the information from the physical structure to a format suitable to be the input of a machine learning algorithm. The input of a machine learning algorithm is a vector composed by features, also known as feature vector. In the context of machine learning potentials, features are also known as: descriptors, descriptors of chemical environments, atomic descriptors, and fingerprints. The second step in the MLP construction is the actual learning or fitting of

the machine learning method. The machine learning method learns the association between the input features and the target value (energy or forces).

The preprocessing has two approaches depending on how the physical structure is represented in the feature space, one approach has an atomistic view, whereas the other has a structure view. In the atomistic view, the feature construction process, associates a feature vector to every particle in the structure, then the machine learning algorithm estimates atomic energies, so that, the energy of the structure is the addition of the atomic energies, one disadvantage of this approach from a quantum mechanical perspective is that, there is no information about atomic energies in a system of atoms, making the partition arbitrary [70]. On the other hand for structure view, the feature construction process, generates a feature vector per every structure, then the machine learning algorithm calculates the energy of the structure directly, figure 5.1 compares the two approaches. This chapter introduces the feature creation process and some of the most popular machine learning potentials, and their applications.

5.2 Feature Creation For Machine Learning Potentials

The success of any machine learning project is highly dependent on how the raw data is transformed into the feature representation. In many cases

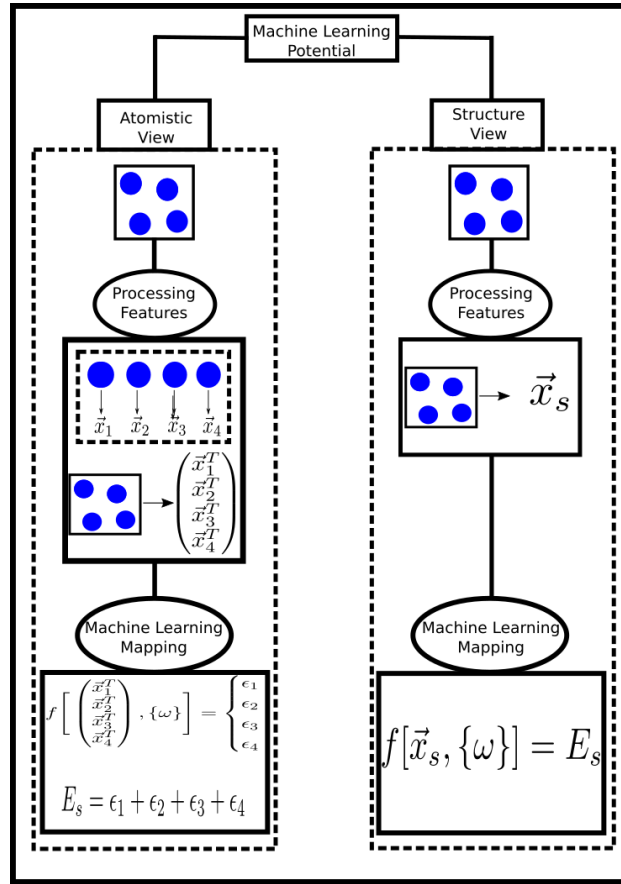


Figure 5.1: Comparison between the atomistic view and the structure view. In the left the atomistic view, the feature construction process calculates a feature vector for every atom ($\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4$ in the input structure (four blue atoms)), then the representation of the structure is the matrix of the feature vectors, in the next step the machine learning potential calculates an atomic like energy (ϵ_i) for every atom in the input structure, finally the energy of the structure is the addition of all the atomic energies $E_s = \sum_i \epsilon_i$. In the right side the structure view, where a feature vector (\vec{x}_s) represents the input structure, and the machine learning algorithm calculates the energy of the structure E_s directly.

the transformation is trivial, for example; in the case of the time series of chapter 2 it is natural to try to represent them as the coefficients of a Fourier

series. Other example of an easy to understand representation is the bag of words, also described in chapter 2, the goal of any representations is to encode the original information into a set of numerical features to which a machine learning method can assign meaning, and extract knowledge out of several training examples.

However, for physical structure the feature representation is less trivial, for instance in Ref [71] Lorenz Gross and Scheffler tried to learn the PES for a hydrogen molecule (H_2) using neural networks as their empirical potential, in this work, the structure was represented by concatenating the Cartesian coordinates of the atoms in space, into a vector of 6 features $\vec{x}^T = (R_{1x}, R_{1y}, R_{1z}, R_{2x}, R_{2y}, R_{2z})$. Despite of the simplicity of the representation, the approach was successful and the PES was well represented by the neural network, however, the neural network trained with this features had some important limitations. If the order of the atoms in \vec{x}^T is exchanged, then the structure would be the same, but the neural network would assign a different energy since it was trained with an specific order. Moreover, any rotation or translation of the H_2 molecule would change the value of the \vec{x}^T without altering the energy of the molecule, but since the values of \vec{x}^T are different, then the neural network would assign a different value for the energy.

The former exemplifies the importance of the feature representation. A feature representation is too specific to details like the reference frame in which the calculation were made, would lead to a MLP with a lack of trans-

ferability, then as it was shown in the example of the H_2 molecule, the feature vector must do more than uniquely represent the structure. So far the MLPs community has come up with some consensus about which requirements, a set of features should meet in order to be a good set of features, those requirements are [72, 73]:

1. The features should be invariant under the symmetries of the structure, so that if the structure is invariant under rotation, reflection and translation, the feature vector does not change after the application of any of this transformations over the structure under study.
2. The features have to be independent of the ordering of the atoms, and invariant under permutations of atoms from the same element.
3. The features should be continuous and differentiable.
4. A good representation should uniquely represents the system it is meant to describe.

Following this principles many feature representations have been proposed, in the following lines some of the features methods are introduced along with their machine learning potentials.

5.3 Neural network potential with symmetry functions

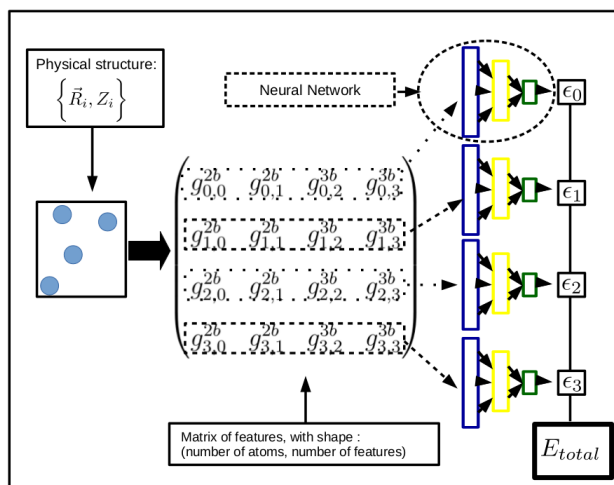


Figure 5.2: Diagram of the process of calculating E_{total} with the BP neural network. First the structure information about positions and species is transformed into feature space. In the feature space every atom in the structure has a feature vector. Then every feature vector is input into a neural network to predict the atomic like energy ϵ_i , $E_{total} = \sum_{i=1}^N \epsilon_i$. In general the representation of the structure is a matrix with rows equal to the number of atoms, and columns equal to the number of features in the feature space.

The neural network potential proposed by Behler and Parrinello [23, 74] is an example of an atomistic MLP. In this approach every atom in a given structure is described by a feature vector, the feature vector is then input into a fully connected neural network, the output of the neural network is an atomic like energy (ϵ_i), the energy of the structure is then the addition of all the atomic like energies ($E_{structure} = \sum_{i=1}^N \epsilon_i$), figure 5.3 depicts this process. Making the total energy dependent on atomic like energies solves

the problem of the specific ordering of the atoms in the structure, since the final result ($E_{structure}$) is independent on the ordering of the elements in the sum. However, it introduces other problems; first, the atomic energies have no quantum mechanical meaning and are an arbitrary value [70], second every structure is represented by a matrix which number of rows is equal to the number of atoms in the structure, therefore, the size of the feature representation is dependent on the number of atoms in the structure, this makes the potential harder to implement with machine learning frameworks like TensorFlow. And third, the back propagation algorithm must run once per atom, instead of once per structure, slowing down the learning process.

The neural network potential has been successfully applied to calculate reduced sections of the potential energy surface of Silicon [23], Copper [66], phonons in crystals [75]. Nevertheless, the method has problems describing clusters or disorder states [68, 67, 76].

In the Behler and Parrinello neural network, every feature of the input vector is the result of a symmetry function, the symmetry functions are the topic of the next subsection.

5.3.1 Symmetry functions

In Behler and Parrinello atomistic approach, there is a vector of features per atom in the structure, the feature calculation seeks to transform the information of the chemical environment surrounding the atom into a feature

vector. The transformation process rely on two types of symmetry functions:

$$g_{ip}^{2b} = \sum_{j \neq i} \exp[-\eta_p (R_{ij} - R_s)^2] f_c(R_{ij}) \quad (5.1)$$

$$g_{ip}^{3b} = 2^{1-\xi_p} \sum_{j \neq i} \sum_{k \neq j, i} (1 + \lambda_p \cos(\theta_{ijk}))^{\xi_p} \exp[-\eta_p (R_{ij} + R_{ik} + R_{jk})^2] * f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}) \quad (5.2)$$

$$f_c(R_{ij}) = \begin{cases} 0.5 \left[\cos \left(\frac{\pi R_{ij}}{R_c} \right) + 1 \right] & \text{if } R_{ij} \leq R_c \\ 0 & \text{if } R_{ij} > R_c \end{cases} \quad (5.3)$$

The g_{ip}^{2b} is the symmetry function of the two body interaction center at the "i" atom, with parameter $\eta_p^{two-body}$, where R_{ij} is the distance between atom i and atom j $R_{ij} = |\vec{R}_j - \vec{R}_i|$, in addition g_{ip}^{3b} is the symmetry function of the three body interaction with parameters $\eta_p^{three-body}$, λ_p , ξ_p , where $\cos(\theta_{ijk}) = \frac{\vec{R}_{ij} \cdot \vec{R}_{ik}}{|\vec{R}_{ij}| |\vec{R}_{ik}|}$, with $\vec{R}_{ij} = \vec{R}_j - \vec{R}_i$. Under this feature set, every structure is represented by a matrix with dimensions (number of atoms, number of features), every "i" row represents the chemical environment of the "i" atom, the number of features is defined by different combinations of the p parameters ($\eta_p^{two-body}$, $\eta_p^{three-body}$, ξ_p , and λ_p), then every row is a feature vector $\vec{g}_i^T = (g_{ip}^{2b}, \dots, g_{ip'}^{2b}, g_{ip}^{3b}, \dots, g_{ip'}^{3b})$ for more details see figure 5.3.1, in applications of the BP features the dimension of the feature space is in between 50 and a 100 symmetry functions.

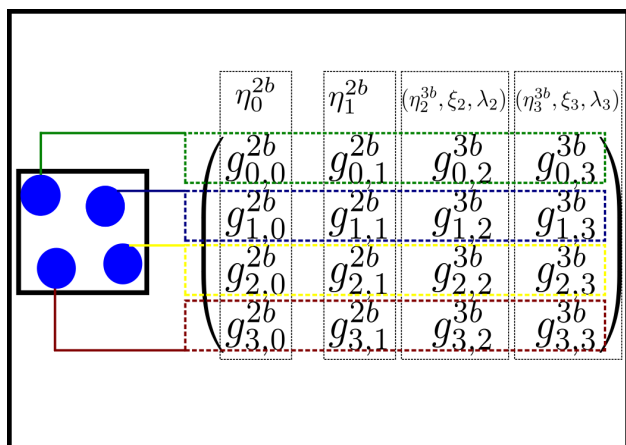


Figure 5.3: Diagrammatic representation of the feature matrix representation of a given structure. Every row represent the feature vector of every one of the 4 atoms. The feature vector is the concatenation of the two-body, and three-body symmetry function, in addition every feature (column) is calculated with a different set of parameters.

The BP features follow the 4 requirements to be a good feature, the values of the symmetry functions are invariant under rotations, translations, and reflections over the structure, they are also continuous and differentiable. Now with respect to uniquely describing a structure, there is only one direct evaluation to the best of our knowledge, of how well the BP descriptor uniquely identifies a chemical environment, the evaluation procedure proposed in Ref [77] determines that in the case of the BP descriptor, the feature vector loses the ability to uniquely describe a particular chemical environment, when the number of neighbors taken into account is higher than 10. However due to the good results of molecular dynamics simulations carried out with the BP features [66, 67], it is safe to think, that they make a uniquely enough representation, at least for the small regions of the phase space in which they

have been used. Moreover, with regard to the ordering of the atoms, it is clear that after choosing the "i" atom (where the function is center) any permutation of atoms will result in the same value for the symmetry function, nevertheless, the representation of the structure as a whole is dependent on the ordering of the atoms as it was seen before. Finally, the BP features lost some of the geometrical information during the summation over the neighboring atoms, as a result the BP features are highly correlated as it is going to be demonstrated in chapter 6.

5.4 Gaussian approximation potentials

The Gaussian approximation potential (GAP) the method was introduced by Bartók and Csányi in Ref [78]. The GAP method calculates the energy of a physical system with an atomistic approach, in which, every "i" atom of the structure is represented by a feature vector \vec{b}_i , the features for the GAP are known as: the smooth overlap of atomic positions (SOAP). Then a Gaussian process is used to calculate the energy of every atom (ϵ_i), and finally the energy of the structure is the addition of the atomic like energies ($E_{structure} = \sum_{i=1}^N \epsilon_i$).

The functional for of the atomic like energies is:

$$\epsilon(\vec{b}_i) = \sum_n \alpha_n e^{-\sum_l \frac{(b_{i,l} - b_{n,l})^2}{2\theta_l}} = \sum_n \alpha_n G(\vec{b}_i, \vec{b}_n) \quad (5.4)$$

Where θ_l is a set of hyper parameters, \vec{b}_n are all the SOAP vectors of the

atomic environments in the training set, the α_n represents the energy contribution of a given "n" environment, and $G(\vec{b}_i, \vec{b}_n)$ measures the similarity between the the environment represented by \vec{b}_i and \vec{b}_n . However, the values of the ϵ_i energies are arbitrary since the only information known is the total energy $E_{structure}$, furthermore, this approach results in a complicated procedure for fitting the model (more details about Gaussian process regression in chapter 2).

5.4.1 The smooth overlap of atomic positions

The SOAP features are also an atomistic based descriptor of the chemical environment surrounding an atom, the method as described in the original formulation starts by calculating the density of particles centered at some atom "i" $\rho_i(\vec{r})$:

$$\rho_i(\vec{r}) = \sum_j \delta(\vec{r} - \vec{r}_{ij}) f_{cut}(r_{ij}) \quad (5.5)$$

However newer applications of the method try to smooth the atomic densities changing the delta distribution for a gaussian function, resulting in smooth overlap of atomic positions (SOAP):

$$\rho_i(\vec{r}) = \sum_j e^{-\frac{(\vec{r}-\vec{r}_{ij})^2}{2\sigma}} f_{cut}(r_{ij}) \quad (5.6)$$

$$f_{cut}(r_{ij}) = \frac{1}{2} \left[1 + \cos \left(\frac{r_{ij}\pi}{r_{cut}} \right) \right] \quad (5.7)$$

where $\vec{r}_{ij} = \vec{r}_j - \vec{r}_i$ and r_{ij} is the magnitude of \vec{r}_{ij} , and $f_{cut}(r_{ij})$ is the cutoff function with r_{cut} cutoff radius.

The densities in equations 5.5 and 5.6 are invariant under exchange of particles, and translations, yet they are not invariant under rotations, this because the densities are dependent of the particular orientation of the coordinate system in which they were calculated. To make the density invariant under rotations, the older versions of SOAP expands the density $\rho_i(\vec{r})$ on a series of spherical harmonics, with the particularity that those spherical harmonics belong to a sphere in 4 dimensions, this spherical harmonics are known as the Wigner matrices $U_{m,m'}^j$, the coefficients of the series are $c_{i,m,m'}^j = \langle U_{m,m'}^j | \rho_i(\vec{r}) \rangle$. The advantage to employ the $U_{m,m'}^j$ matrices to encode the information is that, the expansion does not need a special treatment of the radial degree of freedom, since the surface of a 4 dimensional sphere is a 3 dimensional manifold, then $\rho_i(\vec{r})$ belongs to a 3 dimensional space, the process makes the $c_{i,m,m'}^j$ coefficients a faithful representation.

However, in latter developments, the SOAP method makes the projection of the density on 3 dimensional spherical harmonics $Y_{l,m}(\theta, \phi)$, and radial functions $g_n(r)$ [69], such that the series representation of the density looks like:

$$\rho_i(\vec{r}) = \sum_{n,l,m} c_{n,l,m}^i g_n(r) Y_{l,m}(\theta, \phi) \quad (5.8)$$

Then the expansion coefficients ($c_{n,l,m}^i$) are used to create a power spectrum:

$$p_{n,n',l}^i = \sqrt{\frac{8\pi^2}{2l+1}} \sum_{m,m'} (c_{n,l,m}^i) * c_{n',l,m'}^i \quad (5.9)$$

Then one SOAP feature vector is build for every atom "i" by concatenating several $p_{n,n',l}^i$ up to a threshold value of "n", and "l" depending on the resolution of the computational implementation of the density and the basis set functions.

The SOAP features are invariant under translation, rotation and reflections of the space, also every feature vector is invariant under the exchange of particles, in addition the SOAP density is continuous and differentiable. Now, with regard to uniquely describing a chemical environment the study carried out in Ref [77], concludes that the SOAP is able to differentiate between similar atomic environments for higher number of neighbors than 10. However, even after matching the 4 conditions to be a good descriptor the SOAP method had problems describing amorphous or disordered materials [69]. This is extremely interesting, specially because the solution, with which Deringer and Csányi came up to describe the amorphous phase of carbon was to increase the dimension of the feature vector, but not by increasing the number of $c_{n,l,m}^i$ taken into account, they introduced two new kind features, one regarding two-body interactions and other one regarding three-body interactions. The two-body interaction feature is the distance between 2 atoms $r_{ij} = |\vec{r}_j - \vec{r}_i|$ and the three-body interaction centered at atom "i" interacting with atoms "j" and "k" is a vector of the form $[r_{ij} + r_{ik}, (r_{ij} - r_{ik})^2, r_{jk}]^T$.

The introduction of this features increases the accuracy of the GAP method, the fact that, to increase the accuracy of the method new features were needed and not just more combinations of $c_{n,l,m}^i$, shows that, what the SOAP method lacked was information about different interactions, to say it in other way, the information about specific two-body and three-body interactions was invisible in the many body interaction given by the density of particles.

5.5 Crystal Graph Convolution Of Neural Networks

The Crystal Graph Convolution Of Neural Networks (CGCNN) introduced [79] is a machine learning potential. The method applies an atomistic view to extract features out of a crystal, and a structure view to predict the energy of the structure. The first stage is based on a convolution of neural networks. The convolution input depends on a graph built with the information of the crystal, every node in the graph is related to an atom in the crystal (the relationship may not be 1:1) and every edge in the graph represents a bond in the crystal (the relationship may not be 1:1). The convolution network produces an embedding representation of the structure to feed a fully connected network that makes the energy calculation.

5.5.1 The CGCNN algorithm

As it was written above, the CGCNN method has two stages, one extract information from the structure (seen as a graph), and creates an embedding, the second part takes the embedding representation and makes a prediction. The first part combines atomic features \vec{v}_i , and bonding features $\vec{u}_{(i,j)k}$ in a convolution of neural networks to create a structure representation. Seeing it as a graph every atom is a node represented by \vec{v}_i , and every bond is an edge represented by $\vec{u}_{(i,j)k}$, where "i", and "j" are the atoms present in the bond, and k accounts for the type of bond. The goal of the "R" convolution layers is to build a feature vector for the structure \vec{v}_s with the graph representation $\{\vec{v}_i, \vec{u}_{(i,j)k}\}$ as input, mathematically speaking the convolution process is:

$$\vec{v}_i^{(t+1)} = \vec{v}_i^{(t)} + \sum_{j,k} \sigma \left(\vec{z}_{(i,j)k}^{(t)} W_f^{(t)} + b_f^{(t)} \right) \odot g \left(\vec{z}_{(i,j)k}^{(t)} W_c^{(t)} + b_c^{(t)} \right) \quad (5.10)$$

$$\vec{z}_{(i,j)k}^{(t)} = \vec{v}_i^{(t)} \oplus \vec{v}_j^{(t)} \oplus \vec{u}_{(i,j)k} \quad (5.11)$$

Where \odot is an element wise multiplication and \oplus is the concatenation of vectors.

The application of the pool layer follows the application of "R" convolution layers, to create the feature vector representation of the crystal structure (\vec{v}_s)

$$\vec{v}_s = Pool(\vec{v}_0^{(0)}, \dots, \vec{v}_N^{(0)}, \dots, \vec{v}_N^{(R)}) \quad (5.12)$$

The *Pool* function is an average of the atomic feature vectors of the last convolution layer.

It is important to note that, the CGCNN method encodes the information about inter-atomic distances and orientations in separate ways. The inter-atomic distances are encoded in the $\vec{u}_{(i,j)k}$ features, while the orientations are encoded in the \vec{v}_i feature vector, the orientations are encoded using features like: group number, and period number. As a result, the CGCNN method would lost accuracy as the system under study lacks symmetries, for this reason the applications of the CGCNN method so far are only crystal systems [79].

5.6 Molecular gaussian potentials

The molecular gaussian potentials (MGP) calculate the energy of a structure (molecule) with a structural approach. For the MGP every structure is represented by a feature vector, and the energy of the structure is calculated directly, without estimating the energy of every atom the in the structure.

The MGP uses a gaussian process to calculate the energy of given molecule, the functional form of the model is:

$$E_{structure}(\vec{M}) = \sum_{l=1}^N \alpha_l e^{-\frac{d(\vec{M}, \vec{M}_l)}{\sigma}} \quad (5.13)$$

Where \vec{M} , is the feature vector representing a molecule, the addition runs over all the N molecules in the training set, α_l is a regression coefficient, σ

is an hyper parameters, and $d(\vec{M}, \vec{M}_l)$ is a kernel measuring the similarity between the M molecule, and the M_l molecule in the training data set. The $d(\vec{M}, \vec{M}_l)$ can be different functions, it can be the Cartesian norm $d(\vec{M}, \vec{M}_l) = |\vec{M} - \vec{M}_l|^2$, or a cosine norm $d(\vec{M}, \vec{M}_l) = \frac{\vec{M} \cdot \vec{M}_l}{|\vec{M}| \cdot |\vec{M}_l|}$.

The MGP methods for feature selection are the coulomb matrix and the bag of bonds, they have proved successful in predicting the energy of molecular data sets [80, 81, 82].

5.6.1 The coulomb matrix and the bag of bonds

The next sets of descriptors, belong to structure view in which the feature vector directly represents the structure, and the MLP calculates the energy of the system, without estimating the energy of every atom in the structure separately. The Coulomb matrix (CM) [65] and the Bag of Bonds (BoB) [63] are similar, actually it is fair to say, that, the BoB method is the direct decedent of the CM. The main application of the CM, and BoB methods is to represent molecular structures for energy predictions.

The process to represent a structure with the CM methods starts by building a matrix of the form:

$$CM_{i,j} = \begin{cases} 0.5Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i Z_j}{|\vec{r}_i - \vec{r}_j|} & \text{for } i \neq j \end{cases} \quad (5.14)$$

Then the eigenvalues of the CM matrix are calculated: $CM \rightarrow \{\epsilon_l\}$, in the

next step, the feature vector is assembled by concatenating the $\{\epsilon_l\}$ in decreasing absolute value. In cases where the systems have different atoms the feature vector would have different dimensions, to skip this problem the smaller vector add values of 0 to create the extra dimensions needed to make the size of the feature vectors even. The CM method is invariant under the needed symmetries of rotation, translation, and reflection. The matrix is dependent on the ordering of the atoms, however the representation is based on the eigenvalues which are invariant under permutations of columns and rows, in addition the representation is continuous and it is possible to calculate forces with this approach [83], though, the uniqueness of the representation is challenging since, the QM is only dependent on distances between pairs of atoms with no information about the relative orientation of the atoms.

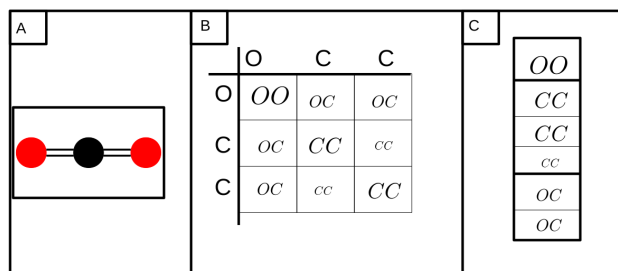


Figure 5.4: A) Representation of a CO_2 molecule. B) The Coulomb matrix representation of the molecule, where $OO = 0.5Z_O^{2.4}$, $OC = \frac{Z_O Z_C}{|\vec{r}_O - \vec{r}_C|}$, and so fort and so on. C) The bag of bonds representation where every element of the CM are order in different bags, depending the type of bond, in this example there are the OO , the CC , and the OC bags, to form the final representation the bags are concatenated into a vector.

The Bag of Bonds [63] (BoB) is the evolution of the CM, the first step to build the representation is still the calculation of the elements CM_{ij} from

equation 5.14, but instead of forming a matrix with them, the CM_{ij} elements are concatenated in a vector, the ordering of the elements follows a bond ordering, where every type of bond is concatenated in a bag to finally form the feature vector concatenating the different bags, as it is shown in figure 5.6.1, for molecules with different number of bonds the BoB will have different number of dimensions, to fix this, the smaller bags of bonds are filled with zeros to ensure that all the vectors in the data set have the same size. The BoB representation is invariant under rotation, translations, and reflections, the ordering of the atoms also does not matter since the final representation is sorted in specific order. Finally the BoB is not able to distinguish between molecules with the same set species and pair wise distances, but with different relative orientations.

5.7 The Partial Radial Distribution Function

This representation, Partial Radial Distribution Function (PRDF) [84] is a crystal exclusive representation, and it is not used to calculate energy or forces, but density of states, however there are some important insights from the approach employed by KT Schütt. The PRDF features are:

$$g_{\alpha\beta}(r_n) = \frac{1}{N_\alpha N_\beta V} \sum_i^{N_\alpha} \sum_j^{N_\beta} \theta(d_{\alpha_i\beta_j} - r_n) \theta(r_n + dr - d_{\alpha_i\beta_j}) \quad (5.15)$$

In here α, β are species of atoms, α_i, β_j are the "i" and "j" atoms of species α , and β . $d_{\alpha_i\beta_j}$ is the distance between the atoms "i" and "j" of species α , and β . $g_{\alpha\beta}(r_n)$ is accounting for how many α, β interactions are in a ring center at r_n with width dr , the interesting point of these features is that, they not only account for the bond information, but they also communicate information about the geometry of the system. Another important point about the PRDF is that the size of the representation is independent of the size of the system it is describing, since the features add over the number of atoms in the structure, so for the PRDF there is not need to add zeroes to create dimensions to compare structure with different numbers of atoms in them. Since the PRDF are not employed to energy and force calculations its presentation is not going to be deeper.

Chapter 6

The Structural Information Filtered Features (SIFF)

6.1 The Structural Information Filtered Features (SIFF)

Despite the success of the methods mentioned in chapter 5, the field of machine learning potentials lacks a universal set of features to accurately describe any physical system [69, 76] (crystal, molecule, cluster). So far, there are different methods focus on specific kinds of materials. For example in the case of crystalline systems there are the symmetry functions method introduced by Beheler and Parrinello [85] (BP), the SOAP method [78], and the CGCNN method [79, 86]. Molecular systems on the other hand, can be described by the CM method [65], or the BoB method [63].

However, for systems with medium range symmetries or disordered states like: clusters or glasses, there are no feature method, yet successful to describe them [87, 69, 88, 89, 90, 91]. The reason why methods like: BP, SOAP, CGCNN could not describe clusters and glasses is because these methods (BP, SOAP, CGCNN) rely on long range order symmetries, to encode physical information into feature representations [77, 92]. Furthermore, methods like CM, and BoB, fail to describe clusters and glasses, because they exclusively rely on the distance matrix to encode short range two-body interactions [77, 70, 83].

Then the need for a universal feature representation became even more clear, when thinking that, one of the possible applications of machine learning potentials is to perform long runs of molecular dynamics simulations, in a long molecular dynamic simulation, the system is likely to visit disordered configurations while transitioning between stable, and metastable phases [93]. Then a feature method able to describe disordered states as well as ordered states is desired to carry out long molecular dynamics simulations with machine learning potentials.

In this chapter the Structural Information Filtered Features (SIFF) are introduced as an answer to the lack of an universal feature method able to describe disordered configurations as well as crystals and molecules. The SIFF method follows the original 4 requirements that outline a good set of features [72, 73]:

- The features should be invariant under the symmetries of the structure,

so that if the structure is invariant under rotation, reflection and translation, the feature representation does not change after the application of any of these transformations over the structure under study.

- The features have to be independent of the ordering of the atoms, and invariant under permutations of atoms from the same element.
- The features should be continuous and differentiable.
- A good representation should uniquely represent the system it is meant to describe.

In addition to develop the SIFF method, the following 3 requirements are added:

- The calculation of the features should be as simple as possible, without losing information that would make the machine learning system misidentify the structures.
- As much geometrical information of the system should be included, but methods of information redundancy can be utilized to only incorporate features that contribute significant information about the system.
- The size of the feature representation should be independent on the number of atoms in the structure.

In the next sections of this chapter, the SIFF method is formally introduced as a consequence of the 7 requirements listed above, additionally, the SIFF

method is tested to ensure its capabilities to describe molecules, crystals, and disordered clusters, while keeping the dimension of the feature vector constant despite the number of particles of the structures described. The testing process also compared the performance of the SIFF method with the SOAP, and CGCNN methods in the case of crystals, the BoB method in the case of molecules, and since there is no specific feature method for disordered clusters, a comparison with the BP method was carry out to show the performance of the SIFF method on disordered structures.

6.2 A formal introduction to the Structural Information Filtered Features

The feature engineering process to build the feature representation of a physical system, can be seen as a process in which information is transferred, the goal of any good feature method is to maximize the information transference, from the physical structure to the feature representation used as input in the machine learning algorithm.

To answer the question of: Which information from the physical structure is needed for a proper description? The Hohenberg-Kohn theorems [31] from the Density Functional Theory [32] are of good use. Knowing that, the external potential is responsible for uniquely determining the energy functional of a system of interacting particles. Then the information needed to define the external potential is the information needed to uniquely identify

the system. In the case of atomistic systems (crystals, molecules, clusters), the external potential is due to the positions and species of the atoms making up the system. Then the only information needed to use as input, for the feature method are the coordinates and species of the atoms in the structure.

Any proposition of a new feature method must follow the original 4 requirements previously discussed, for convenience the Structural Information Features are based on the same building blocks as the BP features, where the building blocks are: the distance matrix ($R_{ij} = |\vec{R}_j - \vec{R}_i|$) between pair of atoms (i and j) in the structure, and the cosine tensor ($\cos(\theta_{ijk}) = \frac{\vec{R}_{ij} \cdot \vec{R}_{ik}}{|\vec{R}_{ij}| |\vec{R}_{ik}|}$, $\vec{R}_{ij} = \vec{R}_j - \vec{R}_i$) taking into account the relative orientations between trios of atoms in the structure. Using R_{ij} and $\cos(\theta_{ijk})$ have the advantage that, the features automatically meet the original 4 requirements, and the additional requirement of being easy to calculate. Furthermore, the only information needed to calculate R_{ij} and $\cos(\theta_{ijk})$ are the positions of the atoms in the structure.

The last two requirements to take into account for the development of the SIFF method are: the maximization of geometrical information, and keeping the dimension of the feature representation constant. To maximize the information transfer, the SIFF method filters the geometrical information originally stored in R_{ij} , and $\cos(\theta_{ijk})$ following the example of convolutions of neural networks [94]. The idea behind the filtering process is to organize the information originally stored in R_{ij} , and $\cos(\theta_{ijk})$ in the features, in such a way, that every feature has a piece of the total information. Figure 6.2

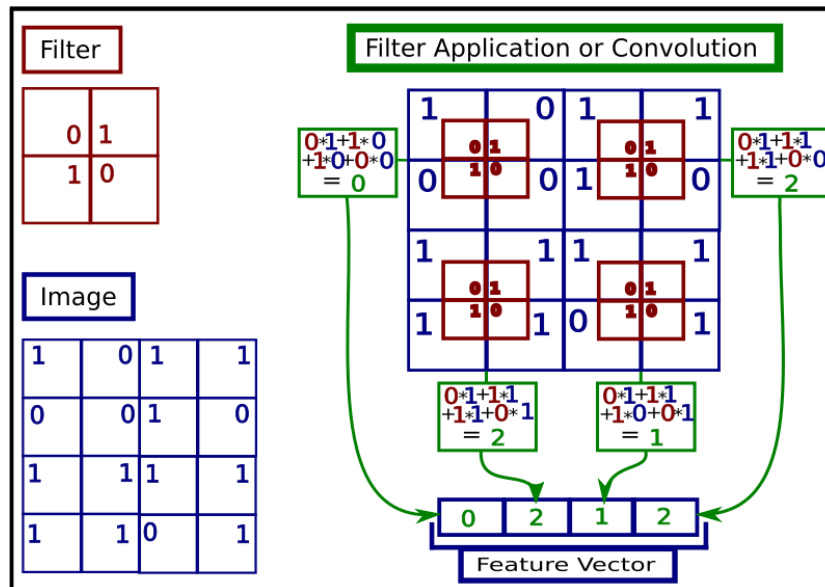


Figure 6.1: Application of a single convolution layer. In the first stage there is an image and a filter, then the application of the filter over the image makes a convolution, selecting certain pixels and combining them into features in a feature vector

shows a diagram of a convolution process, in the first stage, the information originally stored in the image (matrix) is filtered. In the figure 6.2 the filter extracts the information of the anti-diagonal pixels and condensed into features, then the features are organized in a vector conserving the ordering of the raw picture. To keep the dimension of the feature space constant, the SIFF organizes the output of the filtering process following the approach introduced by The Partial Radial Distribution Function [84] described in chapter 5.

There are two kinds of SIFF features, the first kind is dependent on the

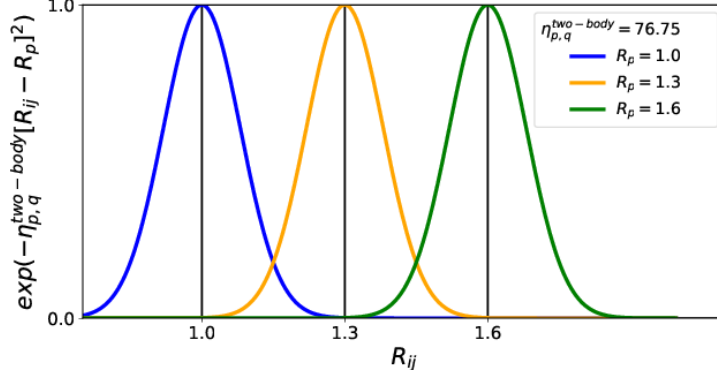


Figure 6.2: Three different filters. Everyone of them extract the information of bonding distances that are similar to R_p while η_r controls the selectivity.

R_{ij} and is known as $SIFF^{two-body}$:

$$SIFF_{pq}^{two-body} = \sum_i \sum_{j \neq i} e^{-\eta_{p,q}^{two-body} (R_{ij} - R_p)^2} * \delta_{ij,q}^{two-body} \quad (6.1)$$

Here the R_p parameter group the information about two-body interactions in such a way that, distances around R_p have a larger weight to those distances significantly less than R_p . The gaussian $e^{-\eta_{p,q}^{two-body} (R_{ij} - R_p)^2}$ acts as a filter centered at R_p ; the parameter $\eta_{p,q}^{two-body}$ defines the reach of the gaussian. A very small $\eta_{p,q}^{two-body}$ does not filter significantly, and a very large $\eta_{p,q}^{two-body}$ filters too much, so that, critical information about the original structure will be missed. Figure 6.2 shows three different filtering gaussians. In general, we can also use different combinations of R_p and $\eta_{p,q}^{two-body}$ for improving the sampling of the geometrical space of the structures we want to represent. The function $\delta_{ij,q}^{two-body}$ insures that the filtering parameter $\eta_{p,q}^{two-body}$ is only applied

to specific element pairwise interactions; for example, the filtering parameter between one pair of elemental interactions, e.g. AB, will be different than the filtering parameter between elemental interactions AA or BB.

The second kind is dependent on the $\cos(\theta_{ijk})$ and is known as $SIFF^{three-body}$:

$$SIFF_{pq}^{three-body} = \sum_i \sum_{j \neq i} \sum_{k \neq j, i} e^{-\eta_{p,q}^{three-body} (\cos\theta_{ijk} - \cos\theta_p)^2} * \delta_{ijk,q}^{three-body} \quad (6.2)$$

Here the $\cos\theta_p$ filters the information about relative orientations. The filter group trios of atoms by the angle between them. The $SIFF^{three-body}$ does not have a cutoff function depending on the distances between atoms, for two reasons, first the $SIFF^{three-body}$ only takes into account angular information, and second because such function would increase the correlation between the $SIFF^{three-body}$ and the $SIFF^{two-body}$, the increment in correlation would decrease the amount of information stored in the features. Similar to its two-body analog, $\eta_{p,q}^{three-body}$ controls how far from the center of the gaussian the interactions are taken into account. The function $\delta_{ijk,q}^{three-body}$ is similar to its two-body analog, that is used to define different feature filters for each elemental trio of interactions, i.e. the filter for the elemental interaction AAA is different than the filter for the elemental interaction ABA, so on and so forth.

Figure 6.3 gives a diagrammatic representation of how the SIFF features are calculated for a given structure, it also explains what the final dimension of the feature representation is. A) Given the S structure, it has three atoms,

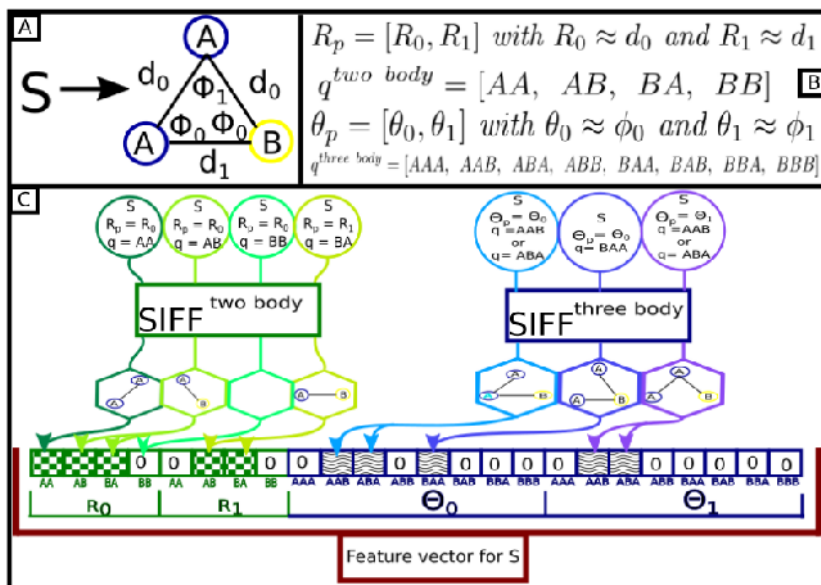


Figure 6.3: Diagram of the feature engineering algorithm for obtaining a feature vector with the Structural Information Filtered Features potentials. The checkboard and wavy pattern in the feature vector mark the features that are different from 0

two different species A and B, the distances between the atoms are either d_0 or d_1 , the angles are ϕ_0 or ϕ_1 . B) The parameters for the SIFF calculations, R_p is the center of the Gaussian part of the SIFF two body, it selects (filters) which interatomic distances are accounted for a given feature, here it can have two values R_0 or R_1 , the $q^{two-body}$ is the set of all the two body combinations of the species in the system, Θ_p is the center of the Gaussian part of the $SIFF^{three-body}$, it selects (filters) which trios of particles are taken into account in a given feature, the $q^{three-body}$ is the set of all the three body combinations of the species in the system. C) Scheme of the feature vector calculation, the circles represent the input needed to calculate an individual

feature for the feature vector, in the case of the $SIFF^{two-body}$, the inputs are: the structure, the value of R_p and the q interaction, in the case of the first feature (from left to right) $R_p = R_0 \approx d_0$ and the q interaction is AA, with these parameters the output (represented in the hexagon) is the result of extracting (filtering) the geometrical parts of the structure that meet the parameters requirements, in the third example for $SIFF^{two-body}$ the input information is $R_p = R_0 \approx d_0$ and the q interaction is BB, in this case, there are no geometrical part of the structure that meets the parameters, thus the output is 0 (left empty in the figure empty). The three-body feature calculation follows a similar process but instead of filtering the structure by interatomic distance, and two body combination, the selection process relies on the angle and the three-body combination. Finally, the feature vector for the S structure is the concatenation of all the individual features. The dimension of the feature vector is the number of R_p values, times the number of $q^{two-body}$ interactions, plus the number of Θ_p values, times the number of $q^{two-body}$. Then the dimension of the feature space is independent of the number of atoms in the system, and it is only dependent on the geometrical filters and types of interactions considered.

6.2.1 SIFF force features

The force calculation with the SIFF method is similar to the energy calculation with respect to the parameters, and the dimension of the feature space, however, there are some important differences. In first place the learning

process is not; structure \rightarrow feature representation \rightarrow energy (target value), but, atom \rightarrow feature representation \rightarrow force (target value). Furthermore, the feature representation for a given atom, is not the SIFF, but, its gradient with respect to the atom for which the force wants to be calculated.

The expression for the calculation of the gradients of the SIFF functions with respect to the “l” atom are:

$$\vec{\nabla}_l(SIFF_{pq}^{two-body}) = \sum_i \sum_{j \neq i} -2\eta_{p,q}^{two-body} e^{-\eta_{p,q}^{two-body}(R_{ij}-R_p)^2} * \delta_{ij,q}^{two-body} \vec{R}_{ij} [\delta_{jl} - \delta_{il}] \quad (6.3)$$

$$\vec{\nabla}_l(SIFF_{pq}^{three-body}) = \sum_i \sum_{j \neq i} \sum_{k \neq j,i} -2\eta_{p,q}^{three-body} e^{-\eta_{p,q}^{three-body}(\cos\theta_{ijk}-\cos\theta_p)^2} * \delta_{ijk,q}^{three-body} \vec{\nabla}_l(\cos\theta_{ijk}) \quad (6.4)$$

Where:

$$\begin{aligned} \vec{\nabla}_l(\cos\theta_{ijk}) = & \left(\frac{\vec{R}_{ij}}{|\vec{R}_{ij}||\vec{R}_{ik}|} - \frac{\cos\theta_{ijk}\vec{R}_{ik}}{|\vec{R}_{ik}|^2} \right) \delta_{il} + \left(\frac{\vec{R}_{ik}}{|\vec{R}_{ij}||\vec{R}_{ik}|} - \frac{\cos\theta_{ijk}\vec{R}_{ij}}{|\vec{R}_{ij}|^2} \right) \delta_{il} \\ & - \left(\frac{\vec{R}_{ij}}{|\vec{R}_{ij}||\vec{R}_{ik}|} - \frac{\cos\theta_{ijk}\vec{R}_{ik}}{|\vec{R}_{ik}|^2} \right) \delta_{kl} - \left(\frac{\vec{R}_{ik}}{|\vec{R}_{ij}||\vec{R}_{ik}|} - \frac{\cos\theta_{ijk}\vec{R}_{ij}}{|\vec{R}_{ij}|^2} \right) \delta_{jl} \end{aligned} \quad (6.5)$$

Here $(\delta_{il}, \delta_{jl}, \delta_{kl})$, are the Kronecker deltas, and $\vec{R}_{ij} = \vec{R}_j - \vec{R}_i$.

Then a machine learning map $f(x, \omega) = y_{target}$ can be trained using as input the gradient of the SIFF vector respect to the position of atom

"l" ($\vec{\nabla}_l(SIFF)$), and as target, the value of the force on atom "l" (\vec{F}_l),
 $f(\vec{\nabla}_l(SIFF), \omega) = \vec{F}_l$.

6.3 Results of SIFF calculations on clusters

6.3.1 SIFF comparison with BP on random clusters

For clusters, the SIFF is tested on two data sets, with different complexities. The two data sets are: C10 composed by 20000 clusters of Carbon with 10 atoms each, and CO1214 with 25000 clusters of Carbon and Oxygen with 12 and 14 atoms in different proportions.

For the C10 data set, SIFF and BP features were calculated, and both feature representations were used to train neural networks with the same number of parameters, and similar architectures. The goal of these experiments is to determine the performance of the SIFF features, and compared them with the reference value given by the BP features. Additionally, a second experiment is carried out, with the SIFF method only, and using the more complex CO1214 data set. The experiment determines the effects on transferability of the SIFF method, for a complex data set. Finally, the SIFF method is used as input for a Gradient Boosting Regression (GBR), to show the advantages of a feature method, that can be used as input of different machine learning algorithms [95].

Clusters data sets creation

The process to build C10 started with the generation of 1000 different clusters (structures) with the firefly algorithm as implemented in the Pychemia software [96], the 1000 structures were generated randomly under certain constraints (e.g. the atoms shouldn't be too close) to ensure that the structures are uncorrelated among them but still physically realistic, then every one of these 1000 structures were input into the FIREBALL software [97] to perform DFT energy calculations, and 20 steps of free dynamics molecular dynamics, with temperature increasing from 500k to 1000k, the exchange correlation functional was the LDA functional implemented in FIREBALL [98, 99], in all the DFT MD steps the energy was converged until 1meV. With a time steps in the MD of 0.1fs.

The second data set for the cluster test C01214 was generated in a similar fashion to C10. First 250 random structures were generated for 10 different concentrations of C and O ($C_{10}O_2$, C_8O_4 , C_6O_6 , C_2O_{10} , $C_{12}O_2$, $C_{10}O_4$, C_8O_6 , C_6O_8 , C_4O_{10} , C_2O_{12}) for a total of 2500 random structures created with Pychemia, then the structures followed the same process described for the C10 data set, with one difference, instead of using 20 steps of free dynamics MD for C01214, 10 steps of free dynamics MD were used.

Results on the C10 data set

The C10 data set, were split in 17000 structures for training, and 3000 for validation, the feature space for both methods SIFF, and BP have 48 di-

mensions (details about feature calculations in the Appendix). Both feature representations are scaled with the MaxAbsScaler function implemented in Scikit-learn [100].

The architecture for the neural network with the best performance for the BP features has; an input layer with 48 nodes (dimension of the feature space), a first hidden layer with 20 nodes, a second hidden layer with 20 nodes, and an output layer with a single node representing the atomic like energy. The activation function, between the input layer, and the first hidden layer is a sigmoid function, as well as in the transition between the first and second hidden layers. Yet, the activation function between the second hidden layer, and the output layer is a linear function. This architecture has 1421 parameters to train, the neural network was implemented using Google’s deep learning framework TensorFlow [101].

The training process for the BP features stopped when over fitting appeared at 200 000 steps, it took the neural net a total of 61 969 seconds (about 17 hours) to do the process. At the end, the root-mean-square error (RMSE) in the training and validation set was: $RMSE_{training} = 0.107eV/structure$ and $RMSE_{validation} = 0.109eV/structure$, both agree with similar measures of error found in Ref [102, 68, 76, 67, 66]

For the SIFF method, the neural network used has the same architecture employed for the BP features, with 1421 parameters. However, in the case of the SIFF method, the neural network predicts directly the energy of the structure, instead of an atomic like energy, as a result of this simplification,

the time needed to carry out the 200 000 training steps was of 18 110 seconds (about 5 hours). At the end, the root-mean-square error (RMSE) in the training and validation set was: $RMSE_{validation} = 0.083eV/structure$, and $RMSE_{training} = 0.072eV/structure$. Table 6.3.1 compares the results for both methods, using the neural net with 1421 parameters.

Feature	Training time 200 000 steps	$RMSE_{training}$	$RMSE_{validation}$
BP	61 969 seconds	0.107 eV/structure	0.109 eV/structure
SIFF	18 110 seconds	0.072 eV/structure	0.083 eV/structure

Table 6.1: Comparison between the BP and SIFF methods, for a neural network with 1421 parameters

Furthermore, the SIFF features were used to train a Gradient Boosting Regression (GBR) model. The GBR model employed was implemented using Scikit-learn, the parameters were: 800 estimators, a maximal depth of 8, minimal samples split of 2, minimal samples leaf of 2, learning rate of 0.109, and the loss function is the least squares regression. At the end, the root-mean-square error (RMSE) in the training and validation set was: $RMSE_{validation} = 0.036eV/structure$, and $RMSE_{training} = 0.0006eV/structure$.

Figure 6.4 shows a plot of the cumulative error (meV/atom) for the three models, BP with neural network (NN), SIFF with NN, and SIFF with GBR; for the BP features only 20% of the structures has an error smaller than 10meV/atom in the validation set, for the SIFF in the same neural network the percentage of structures with error less than 10meV/atom in the validation set increases to 35%. Showing that the SIFF method increase

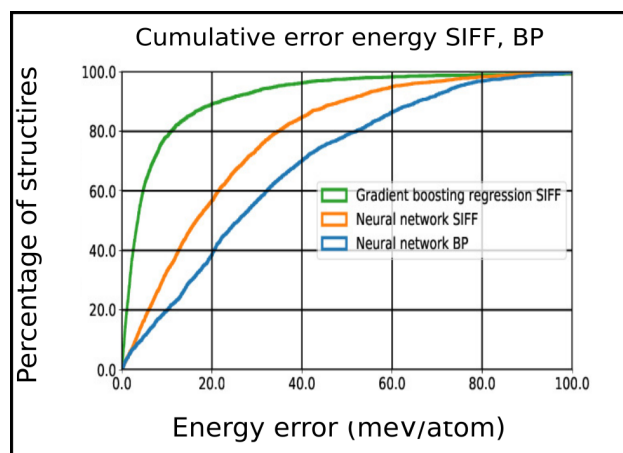


Figure 6.4: (C10 data set) Plot of the cumulative error in the validation set; in blue, a neural network with 1421 parameters after 200 000 epochs with the BP features; in orange, a neural network with 1421 parameters after 200 000 epochs with the SIFF features; and in green the Gradient Boosting regression with the SIFF features

the accuracy of machine learning potentials. However, figure 6.4 also show that for GBR+SIFF nearly 80% of the validation structures have errors less than 10meV/atom, thereby demonstrating the advantage of using a feature method (SIFF) able to be input to different machine learning algorithms.

Results on the CO1214 data set

For the CO1214 data set, only the SIFF features were calculated in a feature space of 152 features (details about feature calculations in the Appendix), the features were scaled with the MaxAbsScaler function implemented in Scikit-learn. The 25000 total structures were split in a validation data set with 3000 structures, and 22000 structures for training. The parameters for the GBR energy model were: 650 estimators, a maximal depth of 7, minimal

samples split of 2, minimal samples leaf of 3, learning rate of 0.1732, and the loss function is the least squares regression. Also, the parameters for the GBR force model were: 900 estimators, a maximal depth of 8, minimal samples split of 2, minimal samples leaf of 3, learning rate of 0.1732, and the loss function is the least squares regression.

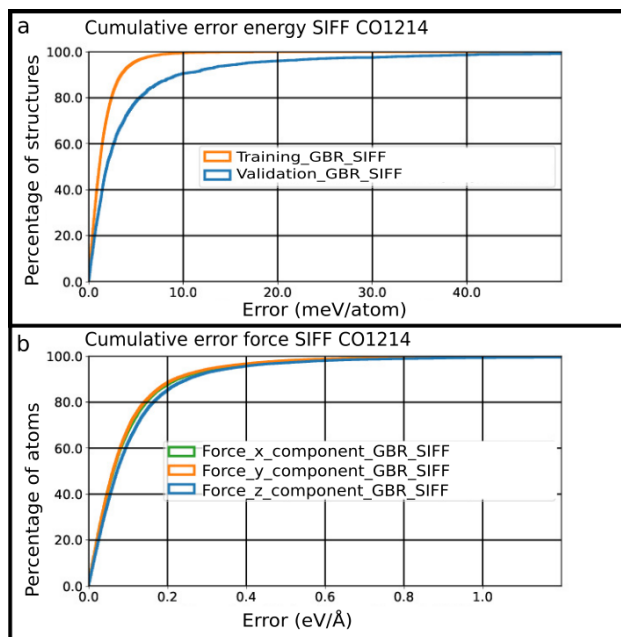


Figure 6.5: (CO1214 data set)a) Cumulative error for energy using a GBR model with the SIFF method, training set in orange, and validation set in blue. b) Cumulative error for force component using a GBR model with the graient SIFF method, the results shown are part of the validation set, in green the x component of the force, in orange the y component, and in blue the z component.

Figure 6.5 a) shows how for energy, the SIFF method is able to keep 90% of the structures in the validation set with an error of less than $10\text{meV}/\text{atom}$. In addition, figure 6.5 b) shows that, 90% of the structures in validation set

have a force error of less than $0.25eV/\text{\AA}$, moreover, the average error for the force calculations in the validation set were of $0.0666eV/\text{\AA}$, which is less than the $0.1eV/\text{\AA}$ error reported by successful calculations of phonons [103, 75].

These results demonstrate the efficiency of the SIFF method to transform the information from the physical structure to the feature representation, regardless of the complexity of the data set. With the SIFF method, a machine learning algorithm can simulate the potential energy function of a data set composed by structures with different species, concentrations, number of atoms, and without any specific symmetry.

Analysis of results from the C10 and CO1214 data sets

Figure 6.4 shows that the use of the SIFF method improved the performance of a NN with the same architecture, number of parameters, and training steps, than the BP method. The reason why the SIFF method does better than the BP method is because, the feature vectors calculated with the SIFF method have more information about the original structure than the feature vectors calculated with BP. To prove this hypothesis, the correlation matrix for both feature representations were calculated (see figure 6.6 c and 6.6 d).

The elements of the correlation matrix are the Pearson correlation coefficients between features in the feature vector. In the case of the SIFF and BP features vectors for the C10 data set, the size of the correlation matrices is 48x48. The results are shown in figure 6.6, where is clear that, for the features calculated with BP, more than 50% of the correlation coefficients

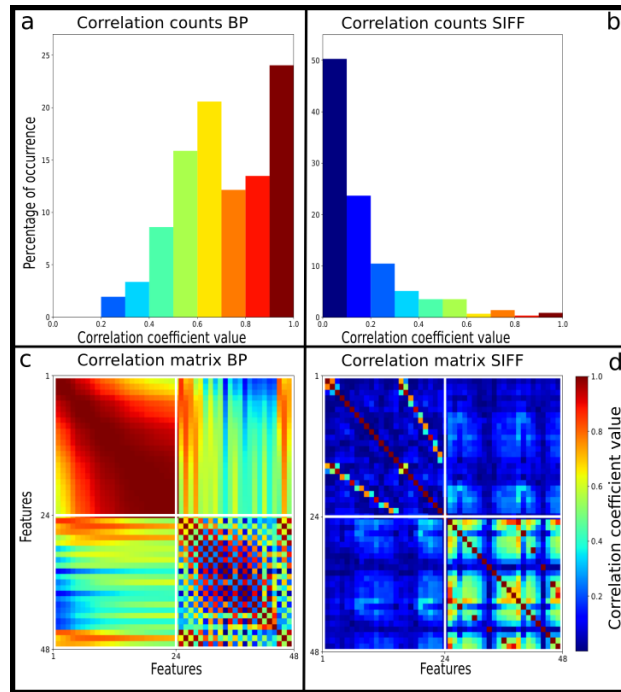


Figure 6.6: (CO 10 data set) a) Histogram of the correlation coefficients for the BP method, the diagonal elements of the correlation matrix were excluded of the count. b) Histogram of the correlation coefficients for the SIFF method. c) Correlation matrix, between the 48 BP features, red means 1.0 or perfect correlation, and blue means 0.0 or non correlation. d) Correlation matrix, between the 48 SIFF features

rank in between 0.6 and 1.0 (figure 6.6 a). On the contrary, for the SIFF features, more than 50% of the correlation coefficients rank in between 0.0 and 0.2 (figure 6.6 b).

Finally, in the case of the CO1214 data set, the reason for the accuracy of the machine learning model, rely on two sources; first the good representation of the potential energy function done by the GBR algorithm, and second, the low amount of correlation present in the SIFF feature vectors (figure 6.7).

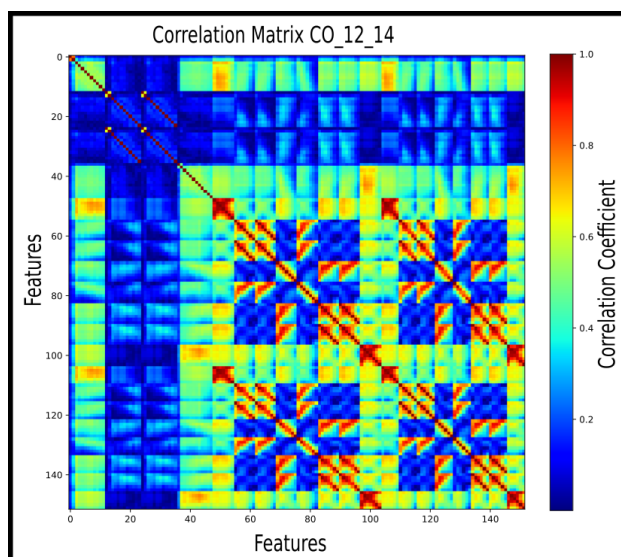


Figure 6.7: Correlation matrix, between the 152 SIFF features for the CO1214 data set, red means 1.0 or perfect correlation, and blue means 0.0 or non correlation.

Hence, as it was discussed in chapter 2: the lower the correlation among features the higher the information passed to the machine learning algorithm making it more accurate.

6.4 Results of SIFF calculations on Molecules

6.4.1 Molecular data sets

The performance of the SIFF method was studied on molecules from the GDB9-14B database, the actual DFT calculations and structures were obtained from the site <http://quantum-machine.org> [104]. The machine learning calculations were performed on two different subsets of data: OCH, and

C7O2H12. The properties of OCH were calculated with a DFT/B3LYP/6-31G(2df,p) level of theory, while the properties of the C7O2H12 data set were calculated with a G4MP2 level theory.

The first data set, OCH has 50592 molecules with different concentrations of Oxygen, Carbon, and Hydrogen, and with between 3 and 29 atoms. The SIFF method was used to predict the lowest unoccupied molecular orbital (LUMO), highest occupied molecular orbital (HOMO), and gap (the gap between the LUMO and HOMO levels), of the OCH molecules, with a machine learning potential. In addition, the performance of the SIFF method was compared with the Bag of Bonds (BoB) method.

The second data set C7O2H12 contains 6095 molecules of $C_7O_2H_{12}$. The SIFF method was used to predict internal energy and free energy, of the C7O2H12 molecules. In addition, the performance of the SIFF method was compared with the Bag of Bonds (BoB) method.

6.4.2 Model selection

A uniform grid search, was the method used to find the best combination of parameters for the machine learning potentials. The Mean Absolute Error (MAE), was the metric used to evaluate every model. Due to its good performance and lower training time, the Gradient Boosting Regression (GBR) was the only kind of model considered.

For the OCH data set, the testing grid has 27 points. The number of estimators could take values of [500, 600, 700]. The maximal depth could

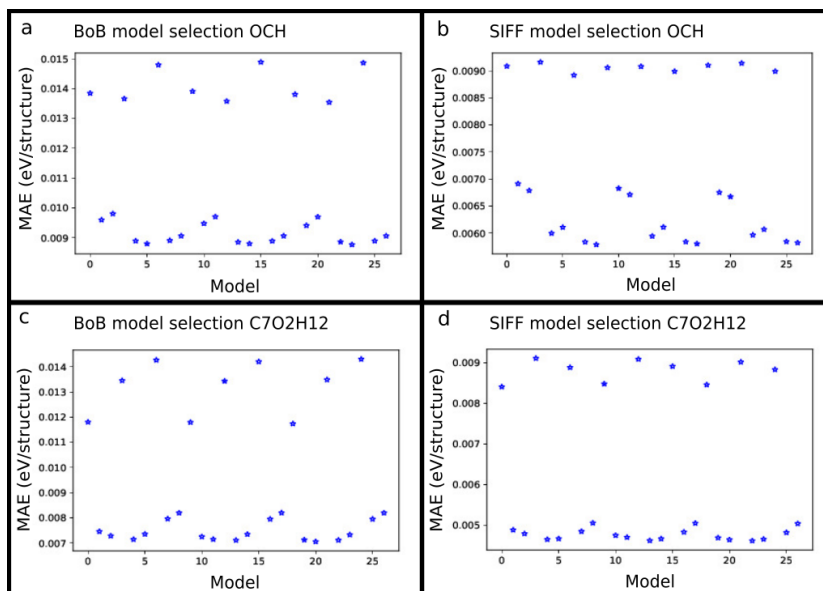


Figure 6.8: a) Models studied for predictions with the BoB method on the OCH data set. Mean Absolute Error (MAE) was the metric used to evaluate the models b) Models evaluated for predictions with the SIFF method, on the OCH data set.c) Models studied for predictions with the BoB method on the C7O2H12 data set. d) Models studied for predictions with the SIFF method on the C7O2H12 data set.

take values of [3, 5, 7], and the learning rate could take values of [0.8, 0.11, 0.15]. Some parameters stayed constant during the searching process, minimal sample split = 3, minimal sample leaf = 3, and the loss function was the least square regression. The models were rated by their ability to predict the gap energy, figure 6.8 a and b shows the results of the 27 models for the OCH data set, for both feature methods (BoB, SIFF) .

In the case of the C7O2H12 data set, the testing grid also had 27 points. The number of estimators could take values of [400, 500, 600]. The maximal depth could take values of [3, 5, 7], and the learning rate could take values

of [0.8, 0.11, 0.15]. Some parameters stayed constant during the searching process, minimal sample split = 3, minimal sample leaf = 3, and the loss function was the least square regression. The models were rated by their ability to predict the free energy, figure 6.8 c and d shows the results of the 27 models for the C7O2H12 data set, for both feature methods (BoB, SIFF).

The parameters for the best models are summarized in table 6.4.2

Feature	Data set	MAE (eV/structure)	LR	MD	NE
BoB	OCH	0.0087	0.15	5	700
SIFF	OCH	0.0057	0.15	7	500
BoB	C7O2H12	0.0067	0.15	3	600
SIFF	C7O2H12	0.0045	0.11	5	600

Table 6.2: Models with the best performance for every method and molecular data set. Learning rage LR, maximal depth MD, number of estimators NE.

6.4.3 Results and analysis on the OCH data set

There are three physical properties used as target in the OCH data set: gap, HUMO, and LUMO. For every property, and every feature type (SIFF, BoB) a GBR model were trained. All trained models follow the parameters described in table 6.4.2. The split of the 50529 structures was: 37944 structures for training and 12648 for validation, the MAE reported below belongs to the validation set. The SIFF method produced a feature space with 297 dimensions (more details about parameters in the Appendix), and the BoB method a feature space with 484 dimensions.

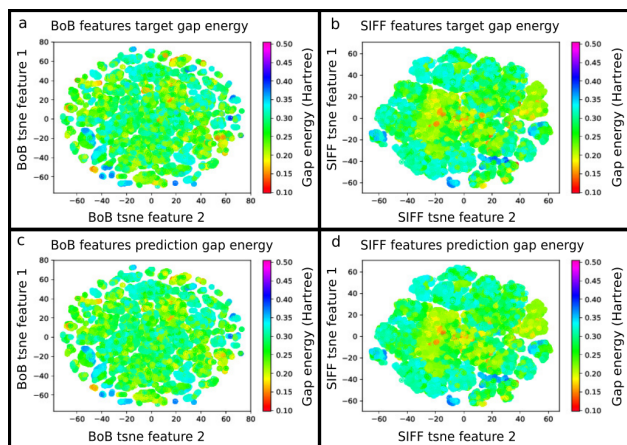


Figure 6.9: (OCH data set) a) Visualization of the learned manifold from gap energy target, using BoB features. b) Visualization of the learned manifold from gap energy target, using SIFF features. c) Visualization of the predicted manifold for gap energy validation using BoB features. d) Visualization of the predicted manifold for gap energy validation using SIFF features.

The MAE for the gap prediction using the SIFF method was 0.11 kcal/mol, for the BoB method was 0.15 kcal/mol, independent results for BoB and other features on similar (molecular) data sets can be found in Ref [80, 81, 82, 105]. Figure 6.9 is a visualization of the reduced feature manifold, the reduction was achieved using the TSNE method implemented in Scikit-learn. Figure 6.9 a and b, shown the learned manifolds. The SIFF method groups the higher energies in an outer rim, and the energy increases, as the radius of the rim decreases. In contrast, the BoB method has ghettos of lower gap values scattered over the feature space. Figures 6.9 c and d, shown the predicted manifold using the validation data. Both pictures exhibit the consistency of the features, predicting similar properties for structures with similarities in the feature space.

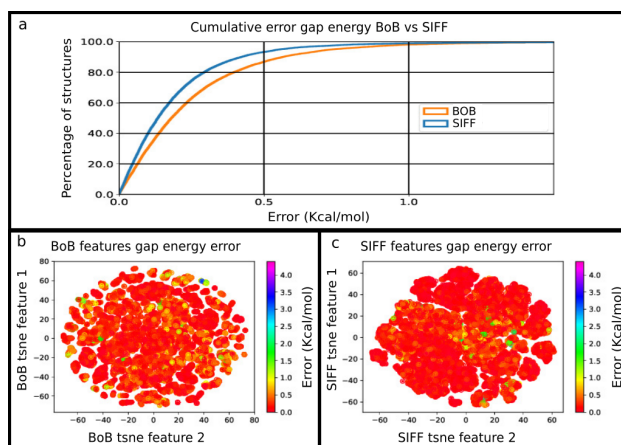


Figure 6.10: (OCH data set) a) Cumulative error comparison for the validation set using SIFF and BoB. b) Error manifold for the validation set using BoB. c) Error manifold for the validation set using SIFF.

A deeper analysis of the error comes from figure 6.10. Part a demonstrate that, the SIFF does a better prediction than the BoB method. Parts b and c details the regions on the manifolds where the errors come from. For the BoB method, points with higher errors (more than 1.0 kcal/mol, yellow) scatter all over the feature space, while for the SIFF these points with higher errors, seem to be group toward the center of the feature space.

In the prediction of the HOMO the SIFF method scores a MAE of 0.0707 kcal/mol, while the BoB method scores 0.0850 kcal/mol. Figure 6.11 follows the trend set by the gap manifold, where the BoB method has dispersed pockets of regions with similar energies, whereas the SIFF method has bigger and concentrated pockets in specific areas of the feature space. Moreover, figure 6.12 a, shows the SIFF method doing slightly better, however, figure 6.12 b and c, follow the trend of a BoB method with high errors straggling,

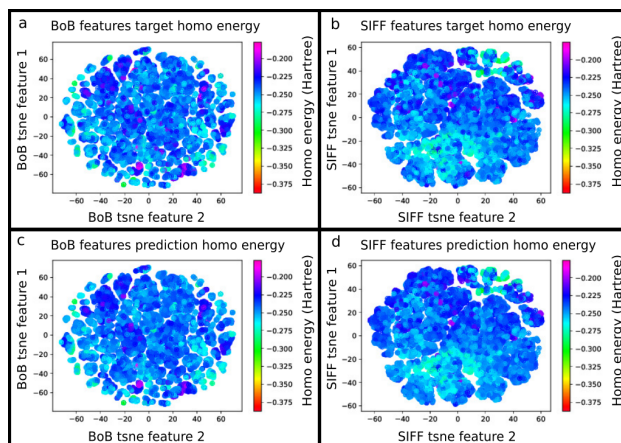


Figure 6.11: (OCH data set) a) Visualization of the learned manifold from HOMO target, using BoB features. b) Visualization of the learned manifold from HOMO target, using SIFF features. c) Visualization of the predicted manifold for HOMO validation using BoB features. d) Visualization of the predicted manifold for HOMO validation using SIFF features.

while the SIFF errors stay in concentrated areas.

The last property predicted with the OCH data set was the LUMO, where the SIFF had MAE of 0.0844 kcal/mol, while the BoB method had a mae of 0.1229kcal/mol. Figure 6.13 depict the LUMO manifold in the reduced feature space. The SIFF method is consistent on grouping structures with similar energies in concentrated areas of the reduced feature space. Whereas, the BoB method group structures with similar energies in scatter pockets around the feature space. Figure 6.14 b and c illustrate the discrepancy between the grouping capabilities of the two methods. The SIFF method has small concentrated pockets of high error structures, in contrast the high error points are situated on extensive areas of the BoB feature space. Furthermore, figure 6.14 a) shows how the SIFF method makes a more accurate prediction

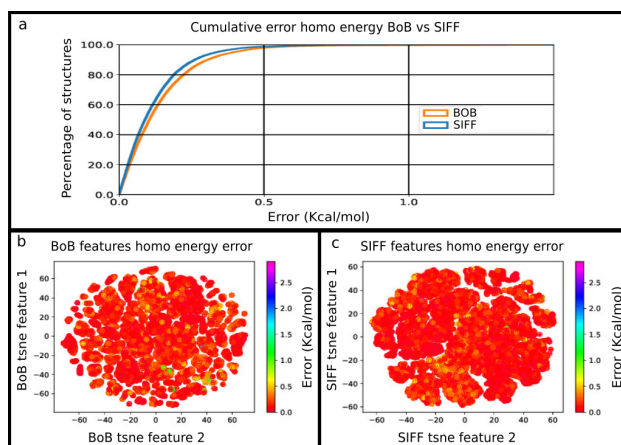


Figure 6.12: (OCH data set) a) Cumulative error comparison for the validation set using SIFF and BoB. b) Error manifold for the validation set using BoB. c) Error manifold for the validation set using SIFF.

of the LUMO values.

The tests carried out on the OCH data set, consistently show that, the SIFF method is more accurate to predict the value of the physical property, whether it is gap, HOMO or LUMO. In addition, for the SIFF method, there are well-defined regions with structures with similar values of gap, HOMO, or LUMO, also in the SIFF method the errors are consistently reduced to specific regions of the feature space. The BoB method on the other hand, struggles to make general pockets, so that, small groups of structures with similar gap, HOMO or LUMO are scattered on the feature space, similarly, the error points are dispersed on the feature space.

The SIFF method does a better grouping of the OCH structures, than the BoB method. A potential explanation come from the fact that, the BoB method rely on organizing the distance matrix elements by type of bonding,

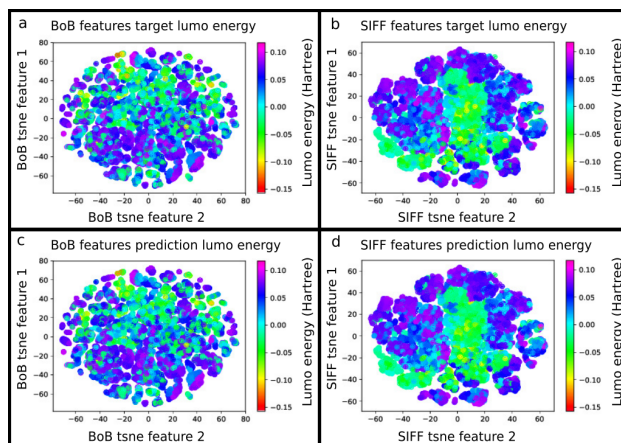


Figure 6.13: (OCH data set) a) Visualization of the learned manifold from LUMO target, using BoB features. b) Visualization of the learned manifold from LUMO target, using SIFF features. c) Visualization of the predicted manifold for LUMO validation using BoB features. d) Visualization of the predicted manifold for LUMO validation using SIFF features.

however, this approach accounts only for the strength of the bonding ignoring the relative orientations. On the other hand, the SIFF method takes into account the strength of the bonding and its relative orientation. The extra information results in a more accurate representation in feature space, and as consequence, an improvement in performance.

6.4.4 Results and analysis on the C7O2H12 data set

There are two physical properties used as target in the C7O2H12 data set: internal energy, and free energy. Following the same procedure as for the OCH data set, a GBR model was trained for every property, and feature type. The trained models have the parameters described in 6.4.2. From the

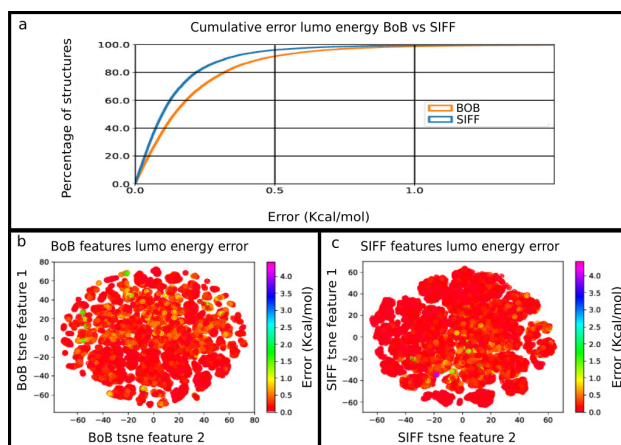


Figure 6.14: (OCH data set) a) Cumulative error comparison for the validation set using SIFF and BoB. b) Error manifold for the validation set using BoB. c) Error manifold for the validation set using SIFF.

6095 total structures, 4571 were used as training set, and 1524 as validation set. The dimension of the feature space with the SIFF method was 702 (more details about parameters in the Appendix), while for the BoB method, the dimension of the feature space was 702.

The MAE for the internal energy prediction on the validation set was 0.0440 kcal/mol for the SIFF method, while for the BoB method was 0.1150 kcal/mol. Figure 6.16 a, shows how 80% of the structures described by the SIFF features have less than 0.25 kcal/mol error, in comparison only 60% of the structures described with BoB have errors of less than 0.25 kcal/mol. Besides, figure 6.16 b and c, shows that, for the BoB method the structures with higher errors are distributed over the whole manifold, however, for the SIFF features the structures with higher errors are group in a specific region. The figure 6.15 depicts the learned manifolds, despite to the small number

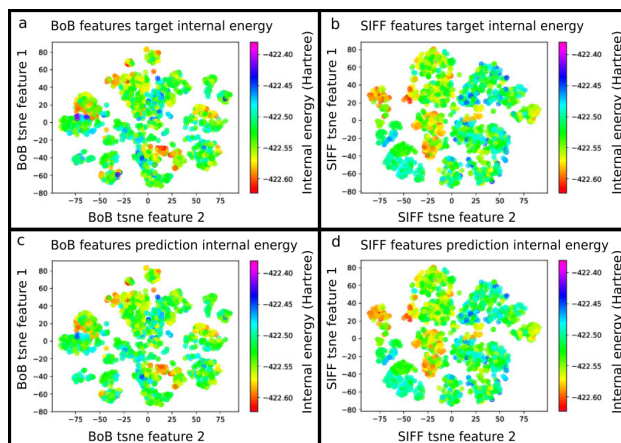


Figure 6.15: (C7O2H12 data set) a) Visualization of the learned manifold from internal energy target, using BoB features. b) Visualization of the learned manifold from internal energy target, using SIFF features. c) Visualization of the predicted manifold for internal energy validation using BoB features. d) Visualization of the predicted manifold for internal energy validation using SIFF features.

of structures of this data set it is possible to see specific clusters of particles with similar internal energies emerging.

The MAE for the formation energy prediction on the validation set was of 0.043 kcal/mol for the SIFF features, and 0.1140 kcal/mol for the BoB feature. Similarly, to the internal energy, figure 6.19 a, shows that, 80% of the structures represented using the SIFF method have errors of less than 0.25 kcal/mol, while for the BoB features is about 50% of the structures the ones with errors less than 0.25 kcal/mol. In addition, figure 6.19 b and c, reinforce the idea that the SIFF features concentrate the error into specific zones unlike the BoB method. The results of the predictions on the C7O2H12 show that, the combination of SIFF + GBR is able to accurately learn the

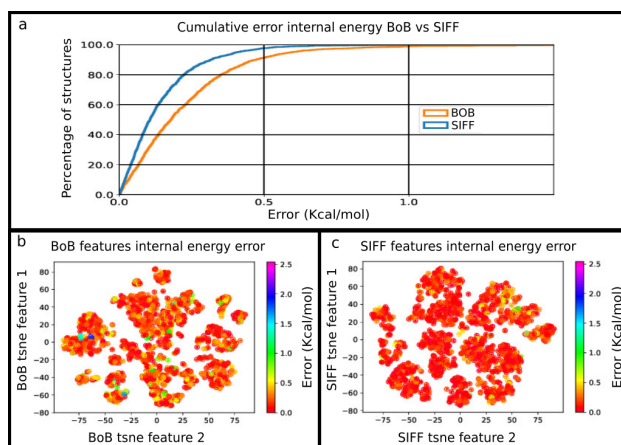


Figure 6.16: (C7O2H12 data set))a) Cumulative error comparison for the validation set using SIFF and BoB. b) Error manifold for the validation set using BoB. c) Error manifold for the validation set using SIFF.

manifold of a physical property (PES in the case of energy being the physical property), regardless of a limited number of training examples. These results are of paramount importance since, realistic application of machine learning potentials must deal with the fact that, the number of molecules in the data sets may decrease, as the number of particles in the molecules increases.

6.5 Results of SIFF calculations on Crystals

6.5.1 Crystal data set

The crystal data set is composed by 2400 structures of $(Al_xGa_yIn_z)_2O_3$ where $x + y + z = 1$. The data set is called AlGaInO, and come from the machine learning competitions site <https://www.kaggle.com/c/nomad2018-predict->

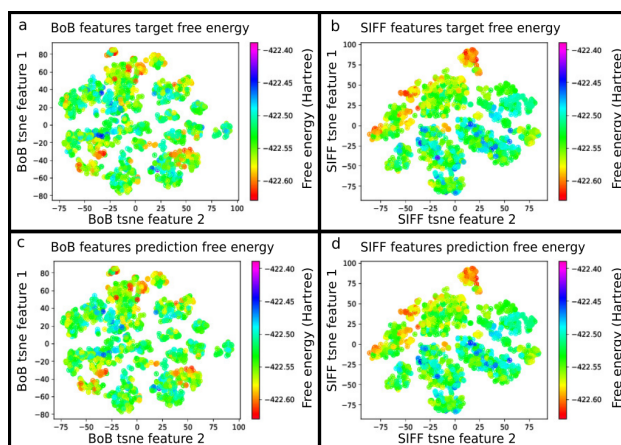


Figure 6.17: (C7O2H12 data set) a) Visualization of the learned manifold from free energy target, using BoB features. b) Visualization of the learned manifold from free energy target, using SIFF features. c) Visualization of the predicted manifold for free energy validation using BoB features. d) Visualization of the predicted manifold for free energy validation using SIFF features.

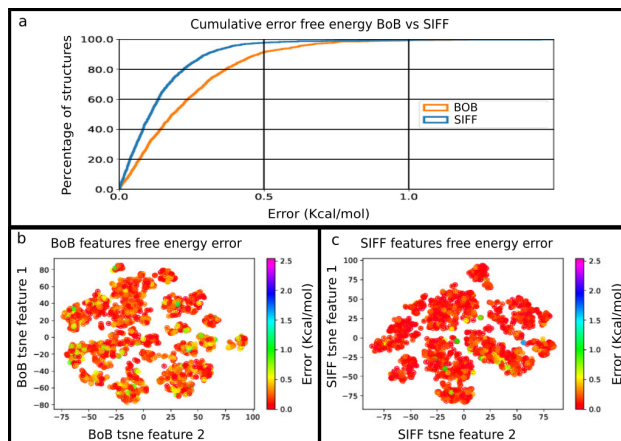


Figure 6.18: (C7O2H12 data set) a) Cumulative error comparison for the validation set using SIFF and BoB. b) Error manifold for the validation set using BoB. c) Error manifold for the validation set using SIFF.

transparent-conductors/data’, the data set was part of a competition to predict formation energy (eV/atom), and band gap (eV).

The best models in the competition [106] used CGCNN [79, 107] and SOAP [78, 108, 109] methods to calculate features (more details about the competition in Ref [106]). In the competition, the CGCNN method achieved a MAE of 114meV for the band gap, and a MAE of 15 meV/atom for the formation energy, while the SOAP features achieved a MAE of 93 meV for the band gap, and a MAE of 13 meV/atom for the formation energy. In this section the SIFF features are compared to the CGCNN method and the SOAP method.

6.5.2 Model selection

For the AlGaInO data set, the model selection method follows the same steps as the model selection for molecular data sets. The GBR models are evaluated with respect to the MAE while predicting the formation energy property.

The testing grid for the SOAP features has 54 points. The number of estimators could take values of [550, 600, 700]. The maximal depth could take values of [5, 6, 7], the learning rate could take values of [0.12, 0.15, 0.18]. The minimal samples split and the minimal samples leaf both could take values of [5, 6], the loss function was the least square regression.

The testing grid for the SIFF features has 36 points. The number of estimators could take values of [400, 500]. The maximal depth could take

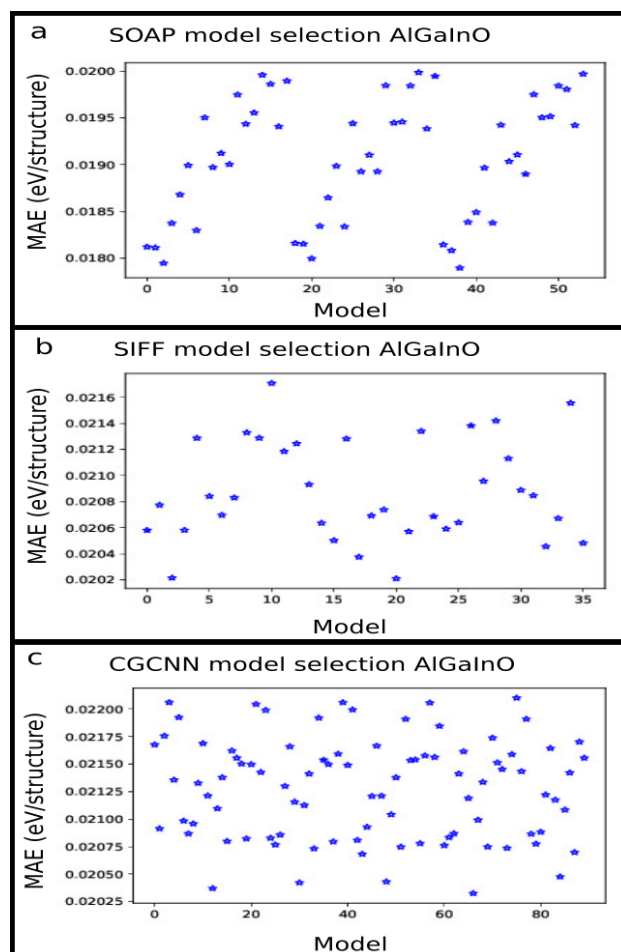


Figure 6.19: Models performance for the AlGaInO data set.

values of [5, 6, 7], the learning rate could take values of [0.13, 0.15, 0.17]. The minimal samples split and the minimal samples leaf both could take values of [4, 6], the loss function was the least square regression.

The testing grid for the CGCNN features has 90 points. The number of estimators could take values of [300, 400, 500, 600, 700]. The maximal depth could take values of [5, 6, 7], the learning rate could take values of [0.12,

0.15, 0.18]. The minimal samples split and the minimal samples leaf both could take values of [5, 6], the loss function was the least square regression, the models with the best performances are summarized in table 6.5.2.

Feature	MAE (eV/atom)	LR	MD	NE	MS
CGCNN	0.0203	0.12	7	600	5
SOAP	0.01789	0.15	5	700	5
SIFF	0.0202	0.15	5	500	4

Table 6.3: Models with the best performance for every method in the AlGaInO data set. Learning rage LR, maximal depth MD, number of estimators NE, minimal samples (split, leaf) MS.

6.5.3 Results and analysis on the AlGaInO data set

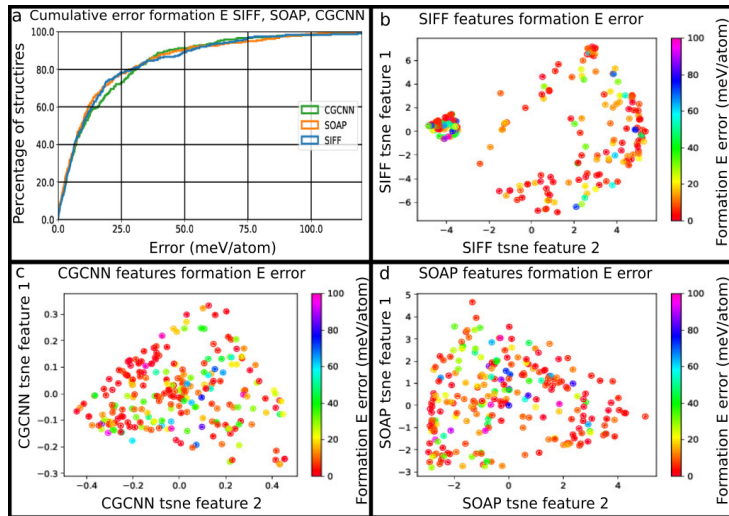


Figure 6.20: (AlGaInO data set) a) Cumulative error comparison for the validation set using CGCNN, SOAP, and SIFF. b) Error manifold for the validation set using SIFF. c) Error manifold for the validation set using CGCNN. d) Error manifold for the validation set using SOAP

Similarly to the original Kaggle competition, the properties predicted were: the formation energy and the band gap. For every property and feature type a GBR model was trained. All the trained models follow the parameters for its given feature as shown in table 6.5.2. The split of the 2400 structures was: 1920 structures for training, and 480 structures for validation. All MAE reported below belongs to the validation set. The SIFF method produced a feature space with 1744 dimensions (more details about parameters in the Appendix), the SOAP method produced a feature space with 500 dimensions, and the CGCNN method produced a feature space with 92 dimensions.

The MAE for the formation energy prediction using the SIFF method was 18.45 meV/atom, using the SOAP method the MAE was 18.02 meV/atom, and using the CGCNN method was 18.71 meV/atom. As figure 6.21 shows, the three sets of features are good at creating pockets of structures with similar energies. Moreover, figure 6.20 a shows that, the three sets of features have a similar performance on describing the formation energy for the Al-GaInO data set. However, figure 6.20 also shows that, non of the features used were able to isolate the structures with higher errors into a concentrated zone.

The MAE for the band gap prediction using the SIFF method was 117.39 meV, using the SOAP method the MAE was 115.27 meV, and using the CGCNN method was 138.24 meV. Figure 6.22 illustrate the band gap manifold, the SIFF, and SOAP methods created clusters of structures with similar band gaps, also the clusters seem ordered on a smooth transition of band gap

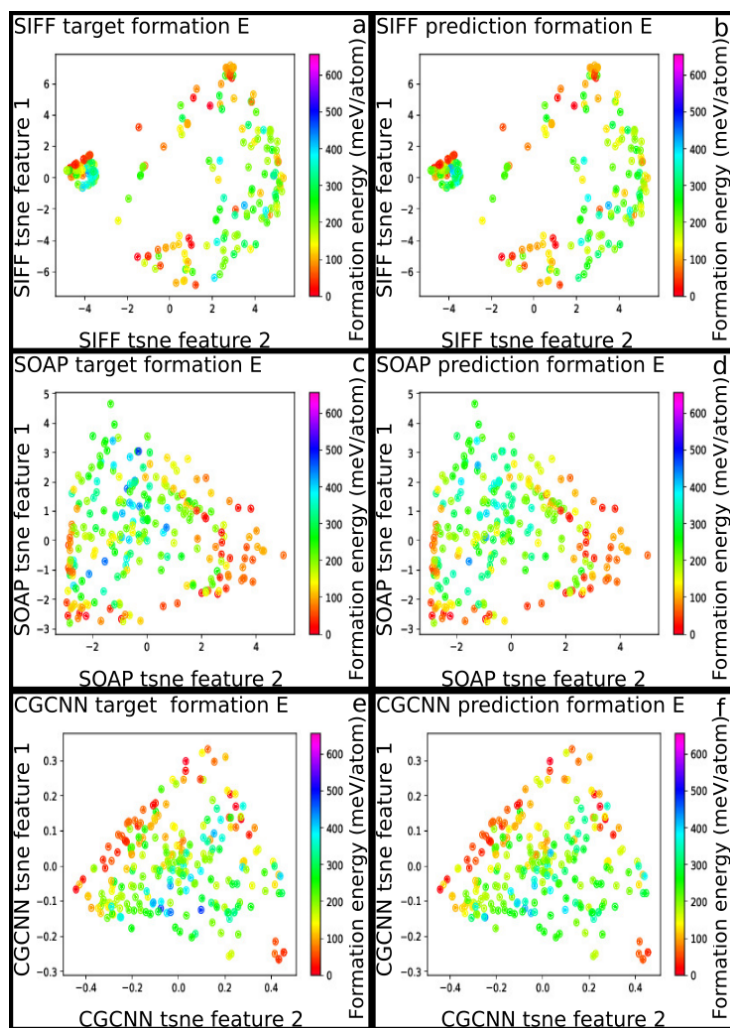


Figure 6.21: (AlGaInO data set) a) Visualization of the learned manifold from formation energy target, using SIFF features. b) Visualization of the predicted manifold for formation energy validation using SIFF features. c) Visualization of the learned manifold from formation energy target, using SOAP features. d) Visualization of the predicted manifold for formation energy validation using SOAP features. e) Visualization of the learned manifold from formation energy target, using CGCNN features. f) Visualization of the predicted manifold for formation energy validation using CGCNN features.

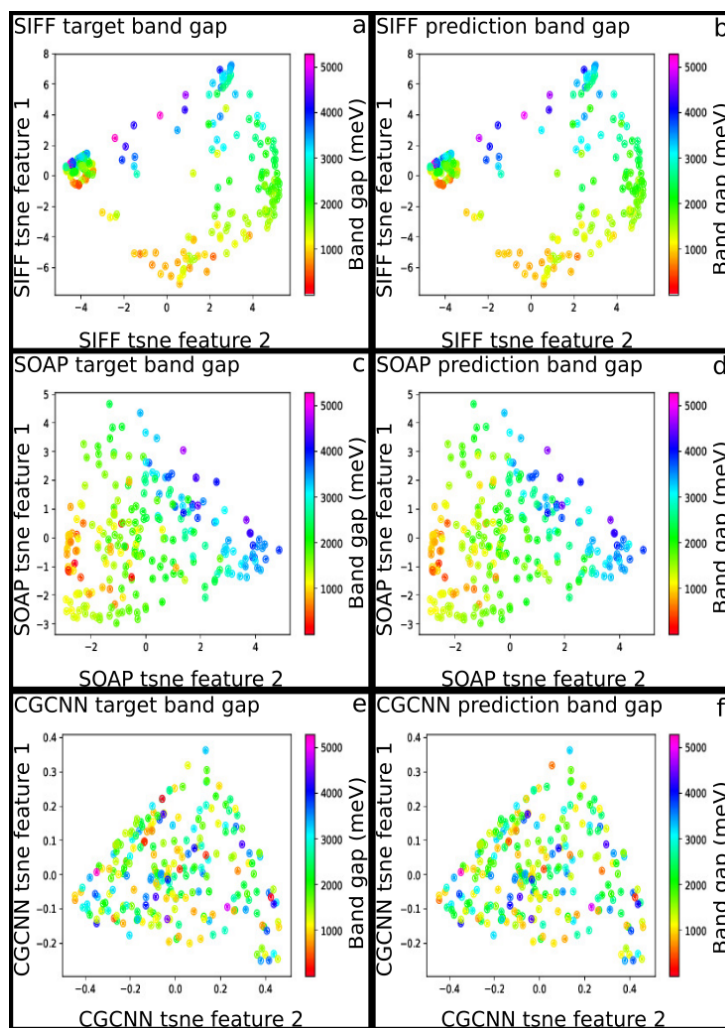


Figure 6.22: (AlGaInO data set) a) Visualization of the learned manifold from band gap target, using SIFF features. b) Visualization of the predicted manifold for band gap validation using SIFF features. c) Visualization of the learned manifold from band gap target, using SOAP features. d) Visualization of the predicted manifold for band gap validation using SOAP features. e) Visualization of the learned manifold from band gap target, using CGCNN features. f) Visualization of the predicted manifold for band gap validation using CGCNN features.

values. Figure 6.23 b, c, and d demonstrate that, none of the feature methods can isolate the error in a specific section of the feature space. Moreover, 6.22 a) shows that the performances of the three method in this data set are fairly similar.

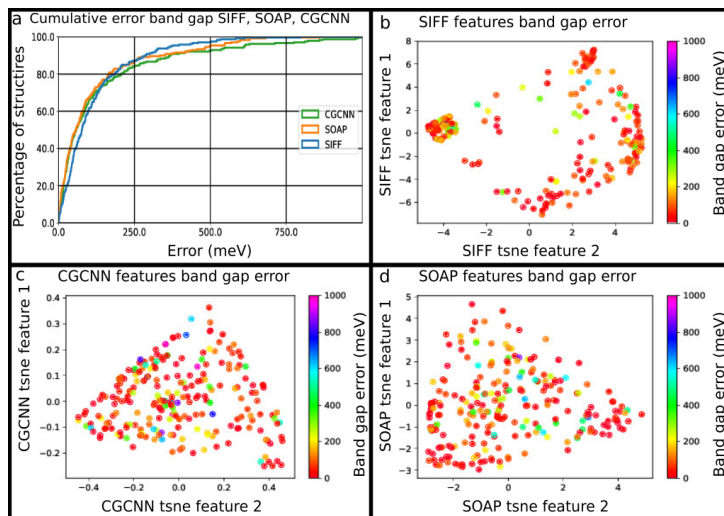


Figure 6.23: (AlGaInO data set) a) Cumulative error comparison for the validation set using CGCNN, SOAP, and SIFF. b) Error manifold for the validation set using SIFF. c) Error manifold for the validation set using CGCNN. d) Error manifold for the validation set using SOAP

It is important to note that, neither the SOAP method, nor the CGCNN methods used in the Kaggle competition and in this section are the exact original methods. In the case of the SOAP method for this section and following the example of the Kaggle competition, a structure base SOAP descriptor was used instead of the atomistic base SOAP descriptor. The structure SOAP is the result of averaging the atomic contributions for a given structure. The CGCNN method used in this section employed the

original neural network to create a structure embedding. This embedding is the same used by the CGCNN as input for its neural network to make predictions, however, a GBR method was used to make predictions instead of the neural network in this section. Also the values of MAE from the Kaggle competition reported in table 6.5.3 were performed on a validation set with 600 structures, and it is different to the validation data set for this section.

Feature	MAE BG* (meV)	MAE BG	MAE FE* (meV/atom)	MAE FE
CGCNN	114	138	15	18.7
SOAP	93	115	13	18.0
SIFF	-	117	-	18.5

Table 6.4: Reference of the results from the Kaggle competition for SOAP and CGCNN. BG* stands for the band gap value in the competition, BG stands for band gap results in this section. FE* stands for the formation energy value in the competition, FE stands for formation energy results in this section

However, even when the values of MAE achieved in the Kaggle competition, and the MAE values obtained in this section are not directly comparable, it is important to note that, they do not differ substantially, having values in similar ranges. The calculations from the official competition were performed on a training data set of 2400 structures, and a validation set of 600 structures. In comparison, the calculations outline in this section were performed with 2160 structures in the training set, and 240 in the validation set, and both sets were part of the original 2400 training set from the Kaggle competition.

Nevertheless, from the calculations performed in this section is possible

to conclude that, the SIFF method does as well as the best methods designed to describe crystal structures. The SIFF method does not improve the performance compared to other methods, likely because crystal structures are not highly complex in comparison with molecules and random clusters, then methods like CGCNN and SOAP, that, rely on large symmetries are good enough to describe the crystals. On the other hand a disadvantage of the SIFF method is the rapid increase in dimension of the feature space, while dealing with many different species of atoms. For the AlGaInO data set nearly 1800 features were need it to describe the structures.

Chapter 7

Conclusions

7.1 Conclusions

The prediction of material properties requires long runs of molecular dynamics simulations, and calculations on systems with many particles. Ab initio methods can perform long runs of MD simulations, and calculations on systems with many particles, nevertheless, the amount of time, and computational resources needed is such, that, the calculations are possible but infeasible. Machine learning potentials can make this kind of simulations feasible, however, a standard method for transforming physical structures into feature vectors is still needed.

This work introduces the Structural Information Filtered Features to feed machine learning potentials. The SIFF method is an answer to the lack of a universal and standard method to feed machine learning potentials.

Unlike the old feature methods introduced in Chapter 5, the SIFF method is derived not only taking into account physical insights like: symmetries and bonding strength, but also feature engineering premises like: maximizing the information storage, and keeping the dimension of the feature space constant, regardless of the number of atoms in the structure.

The SIFF method is a universal feature construction framework in two senses. First it is able to properly transform the information stored in: crystals, disordered clusters, and molecules, into vectors of features for machine learning potentials. Second it can feed different machine learning algorithms, from neural networks, to regression trees.

The SIFF method increases the amount of information in the features. The increment is the result of storing parts of information of the structure, in separate features (the filtering process). In this way every feature communicates a valuable and independent part of the total information. Thanks to the increment in information and the low correlations between features, the SIFF method can accurately describe structures in a disordered state, the accurate description is not limited to calculation of energies, the SIFF method can also calculate atomic forces, such that, long runs of MD simulations where the system visits configurations far from its equilibrium are possible.

Moreover, the SIFF method represent every structure by a feature vector, among the advantages of this approach, the machine learning potential can predict the energy of the system directly, saving time without compromising

accuracy. It also allows the feature representation to be the input of any machine learning algorithm. Another important property of the SIFF method is that. It produces a feature space that is independent on the number of atoms in the structure. This make possible to compare systems with similar compositions, but different number of atoms, it also makes easier the implementation of machine learning potentials with machine learning frameworks like: TensorFlow or Scikit-Learn.

This work also opens the door to future improvements. One of them regards the management of systems with many species, for the AlGaInO data set the SIFF method needed about 1800 features to properly describe the structures. Furthermore, a better selection of the validation set is needed, so far the selection is random, but this end up with areas of the configuration space that are over represented, a better algorithm will select structures trying to uniformly represent the configuration space.

**

Appendix A

Feature parameters

A.1 SIFF parameters for the C10 data set

The parameters for the equation:

$$SIF F_{pq}^{two-body} = \sum_i \sum_{j \neq i} e^{-\eta_{p,q}^{two-body} (R_{ij} - R_p)^2} * \delta_{ij,q}^{two-body} \quad (A.1)$$

For $R_p = [1.0, 1.3, 1.6, 1.9, 2.2, 2.5, 2.8, 3.1, 3.4, 3.7, 4.0, 4.3, 4.6, 4.9, 5.2]$, the value of $\eta_{p,q}^{two-body}$ was 76.75.

For $R_p = [1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5]$, the value of $\eta_{p,q}^{two-body}$ was 27.63.

The parameters for the equation:

$$SIF_{pq}^{three-body} = \sum_i \sum_{j \neq i} \sum_{k \neq j, i} e^{-\eta_{p,q}^{three-body} (\cos\theta_{ijk} - \cos\theta_p)^2} * \delta_{ijk,q}^{three-body} \quad (A.2)$$

For $\cos\theta_p = [0.97, 0.91, 0.80, 0.66, 0.50, 0.30, 0.10, -0.10, -0.30, -0.50, -0.66, -0.80, -0.91, -0.97]$, the value of $\eta_{p,q}^{three-body}$ was 950.0.

For $\cos\theta_p = [9.51e-01, 8.09e-01, 5.88e-01, 3.09e-01, 7.96e-04, -3.08e-01, -5.86e-01, -8.08e-01, -9.50e-01]$, the value of $\eta_{p,q}^{two-body}$ was 1000.0.

A.2 BP parameters for the C10 data set

The parameters for the equation:

$$g_{ip}^{2b} = \sum_{j \neq i} \exp[-\eta_p (R_{ij} - R_s)^2] f_c(R_{ij}) \quad (A.3)$$

The R_s parameter is 0. the R_c (cut off radius) is 6.1, η_p takes values $[0.05, 0.075, 0.10, 0.15, 0.20, 0.225, 0.25, 0.30, 0.350, 0.4, 0.425, 0.45, 0.50, 0.55, 0.60, 0.65, 0.675, 0.70, 0.75, 0.775, 0.80, 0.825, 0.85, 0.875, 0.90, 0.925, 0.95, 0.975, 1.0]$

The parameters for the equation:

$$g_{ip}^{3b} = 2^{1-\xi_p} \sum_{j \neq i} \sum_{k \neq j, i} (1 + \lambda_p \cos(\theta_{ijk}))^{\xi_p} \exp[-\eta_p (R_{ij} + R_{ik} + R_{jk})^2] * f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}) \quad (A.4)$$

The values of the parameters are: $\eta_p = [0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.4, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00]$,
 $\xi_p = [1.000, 2.000, 3.000, 4.00, 5.00, 6.00, 7.00, 8.00, 9.00, 10.00, 11.00, 12.00, 13.0, 14.00, 15.00, 16.00, 17.00, 18.00, 19.00, 20.00]$,
 $p = [1.000, -1.000, 0.750, -0.75, 0.90, -0.90, 0.60, -0.60, 0.85, -0.85, 0.40, -0.40, 0.5, -0.5, 0.35, -0.35, 0.25, -0.25, 0.15, -0.15, 0.40, -0.40, 0.5, -0.5]$.

A.3 SIFF parameters for the CO1214 data set

For equation A.1 the parameters are: for $R_p = [1.0, 1.4, 1.8, 2.2, 2.6, 3.0, 3.4, 3.8, 4.2, 4.6, 5.0, 5.4]$, the value of $\eta_{p,q}^{two-body}$ was 43.17.

The parameters for the equation A.2 are: for $\cos\theta_p = [9.23e-01, 7.07e-01, 3.83e-01, 7.96e-04, -3.83e-01, -7.07e-01, -9.23e-01]$, the value of $\eta_{p,q}^{three-body}$ was 70.0.

For $\cos\theta_p = [0.90, 0.62, 0.22, -0.22, -0.62, -0.90]$, the value of $\eta_{p,q}^{two-body}$ was 75.0.

A.4 SIFF parameters for the OCH data set

For equation A.1 the parameters are: for $R_p = [1.0, 1.4, 1.8, 2.2, 2.6, 3.0, 3.4, 3.8, 4.2, 4.6, 5.0, 5.4]$, the value of $\eta_{p,q}^{two-body}$ was 43.17.

The parameters for the equation A.2 are: for $\cos\theta_p = [9.23e-01, 7.07e-01, 3.83e-01, 7.96e-04, -3.83e-01, -7.07e-01, -9.23e-01]$, the value of $\eta_{p,q}^{three-body}$

was 70.0.

For $\cos\theta_p = [0.90, 0.62, 0.22, -0.22, -0.62, -0.90]$, the value of $\eta_{p,q}^{two-body}$ was 75.0.

A.5 SIFF parameters for the C7O2H12 data set

For equation A.1 the parameters are: for $R_p = [1.0, 1.4, 1.8, 2.2, 2.6, 3.0, 3.4, 3.8, 4.2, 4.6, 5.0, 5.4]$, the value of $\eta_{p,q}^{two-body}$ was 43.17.

$R_p = [1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5]$, the value of $\eta_{p,q}^{two-body}$ was 27.63.

$R_p = [1.0, 1.6, 2.2, 2.8, 3.4, 4.0, 4.6, 5.2]$, the value of $\eta_{p,q}^{two-body}$ was 19.19. The parameters for the equation A.2 are: for $\cos\theta_p = [9.23e-01, 7.07e-01, 3.83e-01, 7.96e-04, -3.83e-01, -7.07e-01, -9.23e-01]$, the value of $\eta_{p,q}^{three-body}$ was 70.0.

For $\cos\theta_p = [0.90, 0.62, 0.22, -0.22, -0.62, -0.90]$, the value of $\eta_{p,q}^{two-body}$ was 75.0.

$\cos\theta_p = [9.51e-01, 8.09e-01, 5.88e-01, 3.09e-01, 7.96e-04, -3.08e-01, -5.86e-01, -8.08e-01, -9.50e-01]$, the value of $\eta_{p,q}^{two-body}$ was 1000.0.

A.6 BoB parameters for the OCH and C7O2H12 data sets

The BoB features were calculated with the *molml* software (<https://pypi.org/project/molml/>) with the *BagOfBonds()* object.

A.7 SIFF parameters for the AlGaInO data set

For equation A.1 the parameters are: $R_p = [1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5]$, the value of $\eta_{p,q}^{two-body}$ was 27.63.

$R_p = [1.0, 1.3, 1.6, 1.9, 2.2, 2.5, 2.8, 3.1, 3.4, 3.7, 4.0, 4.3, 4.6, 4.9, 5.2]$, the value of $\eta_{p,q}^{two-body}$ was 76.75.

The parameters for the equation A.2 are: for $\cos\theta_p = [9.23e-01, 7.07e-01, 3.83e-01, 7.96e-04, -3.83e-01, -7.07e-01, -9.23e-01]$, the value of $\eta_{p,q}^{three-body}$ was 70.0.

$\cos\theta_p = [0.97, 0.91, 0.80, 0.66, 0.50, 0.30, 0.10, -0.10, -0.30, -0.50, -0.66, -0.80, -0.91, -0.97]$, the value of $\eta_{p,q}^{three-body}$ was 950.0.

A.8 SOAP parameters for the AlGaInO data set

The SOAP features were calculated with the dscribe package (there is no article yet about the dscribe package) the code can be find in '<https://github.com/SINGROUP/dscribe>'

The features were calculated with:

```
from dscribe.descriptors import SOAP
rcut= 10.0 nmax= 4 lmax= 4 periodic_soap = SOAP([49, 31, 13, 8], rcut, nmax, lmax, periodic =
True, sparse = False)
```

```
soap_ctrl = periodic_soap.create(ctrl) soap_ctrl aver = np.average(soap_ctrl, axis =
0) soap_feat.append(soap_ctrl aver)
```

Where is an instance of `Crystal` from `ase.spacegroup` import `Crystal`

A.9 CGCNN parameters for the AlGaInO data set

The CGCNN [79] features were calculated with the CGCNN package '<https://github.com/txie-93/cgcnn>'.

The embedding features where calculated with parameters: training set 2400, atom features 90, hidden features length 90, number of convolution 1, hidden layers 3, epochs 30, learning rate 0.3.

References

- [1] B. Cheng and D. M. Titterton, “Neural networks: a review from a statistical perspective,” *Stat. Sci.*, vol. 9, no. 1, pp. 2–30, 1994.
- [2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2009, +0,525258258525252787787474*/==741æø’ m,.mnbvcx12, ISBN: 978-0-387-84857-0. arXiv: 1010.3003.
- [3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification And Regression Trees*. Routledge, 2017, ISBN: 9781315139470.
- [4] C. M. Bishop, *Pattern Recognition and Machine Learning*. 2006, ISBN: 978-0-387-31073-2. arXiv: 1011.1669.
- [5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. 1991, pp. 73–78,216–244, ISBN: 9780262018029. arXiv: 0-387-31073-8.
- [6] J. Gareth, W. Daniela, H. Trevor, and R. Tibshirani, *An Introduction to Statistical Learning*. 2006, vol. 102, p. 618, ISBN: 9780387781884. arXiv: arXiv:1011.1669v3.
- [7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [8] S. L. Salzberg, “C4.5: programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993,” *Mach. Learn.*, vol. 16, no. 3, pp. 235–240, 1994.

- [9] I. Guyon, A. Elisseeff, and A. M. De, “An introduction to variable and feature selection,” *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.
- [10] T. Mikolov, G. Corrado, K. Chen, and J. Dean, “Efficient estimation of word representations in vector space,” *Proc. Int. Conf. Learn. Represent. (ICLR 2013)*, pp. 1–12, 2013. arXiv: arXiv:1301.3781v3.
- [11] J. Pennington, R. Socher, and C. Manning, “Glove: global vectors for word representation,” in *Proc. 2014 Conf. Empir. Methods Nat. Lang. Process.*, 2014, pp. 1532–1543, ISBN: 9781937284961. arXiv: 1504 . 06654.
- [12] J. Provost, “Naïve-bayes vs. rule-learning in classification of email,” *Univ. Texas Austin, Artif. Intell. Lab, CiteSeer*, no. Ingebrigsten, pp. 1–4, 1999.
- [13] E. Rudkowsky, M. Haselmayer, M. Wastian, M. Jenny, Š. Emrich, and M. Sedlmair, “More than bags of words: sentiment analysis with word embeddings,” *Commun. Methods Meas.*, vol. 12, no. 2-3, pp. 140–157, 2018.
- [14] Y. Li and T. Li, “Feature engineering for machine learning and data analytics,” in G. Dong and H. Liu, Eds., CRC Press, ch. Chapter 8, ISBN: 9781138744387.
- [15] F. Mörchen, “Time series feature extraction for data mining using dwt and dft,” *Tech. Report, No. 33, Dep. Math. Comput. Sci. Univ. Marburg, Ger.*, pp. 1–31, 2003.
- [16] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” Tech. Rep.
- [17] I. Guyon and A. Elisseeff, *An Introduction to Feature Extraction*, 2006th ed., M. Nikravesh and L. Zadeh, Eds. Springer, ISBN: 978-3-540-35487-1. arXiv: arXiv:1011.1669v3.

- [18] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” *Artif. Intell.*, vol. 97, no. 1-2, pp. 273–324, 1997.
- [19] R. Kohavi, “Wrappers for performance enhancement and obvious decision graphs,” PhD thesis, Stanford University, 1995.
- [20] H.-H. Hsu and C.-W. Hsieh, “Feature selection via correlation coefficient clustering,” *J. Softw.*, vol. 5, no. 12, pp. 1371–1377, 2010.
- [21] A. Kraskov, H. Stögbauer, and P. Grassberger, “Estimating mutual information,” *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, vol. 69, no. 6, p. 16, 2004. arXiv: 0305641 [cond-mat].
- [22] M. Hall, “Correlation-based feature selection for machine learning,” *Methodology*, vol. 21i195-i20, no. April, pp. 1–5, 1999. arXiv: 9809069v1 [arXiv:gr-qc].
- [23] J. Behler and M. Parrinello, “Generalized neural-network representation of high-dimensional potential-energy surfaces,” *Phys. Rev. Lett.*, vol. 98, no. 14, 2007.
- [24] L. Jyyafjl and R. L. Rjvest, “Constructjng o{~}jmaljrnary i;ecjgjon trees is nj’-complete* laurent jyyafjl,” *Inf. Process. Lett.*, no. 1, pp. 1–3, 1976.
- [25] J. M. J. M. Thijssen, *Computational physics*. Cambridge University Press, 2007, p. 620, ISBN: 9780521833462.
- [26] E. Kaxiras, *Atomic and Electronic Structure of Solids*. 2010, ISBN: 9780511755545. arXiv: arXiv:1011.1669v3.
- [27] J. M. Combes, P. Duclos, and R. Seiler, “The born-oppenheimer approximation,” in *Rigorous At. Mol. Phys.* Boston, MA: Springer US, 1981, pp. 185–213.
- [28] R. M. Martin, L. Reining, and D. M. Ceperley, *Interacting Electrons Theory and Computational Approaches*. 2016, p. 840, ISBN: 9780521871501.

- [29] J. R. C. And, M. J. Frisch, F. J. D. And, and P. J. Stephens, “Hartree-fock and density functional theory ab initio calculation of optical rotation using gjaos: basis set dependence,” 2000.
- [30] A. P. S. And and L. Radom*, “Harmonic vibrational frequencies: an evaluation of hartree-fock, moller-plesset, quadratic configuration interaction, density functional theory, and semiempirical scale factors,” 1996.
- [31] P. Hohenberg and W. Kohn, “Inhomogeneous electron gas,” *Phys. Rev.*, vol. 136, no. 3B, B864–B871, 1964.
- [32] W. Kohn and L. J. Sham, “Self-consistent equations including exchange and correlation effects,” *Phys. Rev.*, vol. 140, no. 4A, A1133–A1138, 1965.
- [33] T Clark, “Quo vadis semiempirical mo-theory?” *J. Mol. Struct. THEOCHEM*, vol. 530, no. 1-2, pp. 1–10, 2000.
- [34] R. Martin, *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004, ISBN: 9780511805769.
- [35] E. G. Lewars, *Computational Chemistry*. 2011, ISBN: 978-90-481-3860-9. arXiv: arXiv:1011.1669v3.
- [36] J. P. Perdew and W. Yue, “Accurate and simple density functional for the electronic exchange energy: generalized gradient approximation,” *Phys. Rev. B*, vol. 33, no. 12, pp. 8800–8802, 1986.
- [37] A. D. Becke, “Density-functional exchange-energy approximation with correct asymptotic behavior,” *Phys. Rev. A*, vol. 38, no. 6, pp. 3098–3100, 1988.
- [38] M. Head-Gordon, “Quantum chemistry and molecular processes,” Tech. Rep., 1996.

- [39] S. F. Sousa, P. A. Fernandes, and M. J. Ramos, “General performance of density functionals,” *J. Phys. Chem. A*, vol. 111, no. 42, pp. 10 439–10 452, 2007.
- [40] O. F. Sankey and D. J. Niklewski, “Ab initio multicenter tight-binding model for molecular-dynamics simulations and other applications in covalent systems,” *Phys. Rev. B*, vol. 40, no. 15, 1989.
- [41] A. A. Demkov, J. Ortega, O. F. Sankey, and M. P. Grumbach, “Electronic structure approach for complex silicas,” *Phys. Rev. B*, vol. 52, no. 3, pp. 1618–1630, 1995.
- [42] P. Jelínek, H. Wang, J. P. Lewis, O. F. Sankey, and J. Ortega, “Multicenter approach to the exchange-correlation interactions in ab initio tight-binding methods,” *Phys. Rev. B*, vol. 71, no. 23, p. 235 101, 2005.
- [43] J. Harris, “Simplified method for calculating the energy of weakly interacting fragments,” *Phys. Rev. B*, vol. 31, no. 4, pp. 1770–1779, 1985.
- [44] A. P. Horsfield, “Efficient ab initio tight binding,” *Phys. Rev. B*, vol. 56, no. 11, pp. 6594–6602, 1997.
- [45] J. M. Soler, E. Artacho, J. D. Gale, A. García, J. Junquera, P. Ordejón, and D. Sánchez-Portal, “The siesta method for ab initio order-n materials simulation,” *J. Phys. Condens. Matter*, vol. 14, no. 02, pp. 2745–2779, 2002.
- [46] D. Vanderbilt, “Rapid communications soft self-consistent pseudopotentials in a generalized eigenvalue formalism,” Tech. Rep., pp. 15–1990.
- [47] W. R. Scott, P. H. Hünenberger, I. G. Tironi, A. E. Mark, S. R. Billeter, J. Fennen, A. E. Torda, T. Huber, P. Krüger, and W. F. Van Gunsteren, “The gromos biomolecular simulation program package,” *J. Phys. Chem. A*, vol. 103, no. 19, pp. 3596–3607, 1999.

- [48] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, “Development and testing of the opls all-atom force field on conformational energetics and properties of organic liquids,” *J. Am. Chem. Soc.*, vol. 118, no. 15, pp. 11 225–11 236, 1996. arXiv: [arXiv:1905.08941v1](#).
- [49] A. D. Mackerell, “Empirical force fields for biological macromolecules: overview and issues,” *J. Comput. Chem.*, vol. 25, no. 13, pp. 1584–1604, 2004.
- [50] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman, “A second generation force field for the simulation of proteins, nucleic acids, and organic molecules,” *J. Am. Chem. Soc.*, vol. 117, no. 19, pp. 5179–5197, 1995. arXiv: [z0024](#).
- [51] E. Harder, W. Damm, J. Maple, C. Wu, M. Reboul, J. Y. Xiang, L. Wang, D. Lupyan, M. K. Dahlgren, J. L. Knight, J. W. Kaus, D. S. Cerutti, G. Krilov, W. L. Jorgensen, R. Abel, and R. A. Friesner, “Opls3: a force field providing broad coverage of drug-like small molecules and proteins,” *J. Chem. Theory Comput.*, vol. 12, no. 1, pp. 281–296, 2016.
- [52] N. Foloppe and A. D. MacKerell, “All-atom empirical force field for nucleic acids: i. parameter optimization based on small molecule and condensed phase macromolecular target data,” *J. Comput. Chem.*, vol. 21, no. 2, pp. 86–104, 2000.
- [53] A. A. Pádua, “Torsion energy profiles and force fields derived from ab initio calculations for simulations of hydrocarbon-fluorocarbon diblocks and perfluoroalkylbromides,” *J. Phys. Chem. A*, vol. 106, no. 43, pp. 10 116–10 123, 2002. arXiv: [arXiv:0705.2255v1](#).
- [54] G. Raabe, *Applications and Perspectives Molecular Simulation Studies on Thermophysical Properties With Application to Working Fluids*. Springer, 2017, ISBN: 9789811035449.
- [55] S. A. Alexander and R. L. Coldwell, “A ground state potential energy surface for using monte carlo methods the potential energy surface of

- h₂ energy levels and potential energy curves for h₂, n₂, and o₂ with an independent particle model new born-oppenheimer potential energy curve and,” *Accurate Adiabatic Treat. Gr. State Hydrog. Mol. J. Chem. Phys.*, vol. 121, p. 3663, 2004.
- [56] B. W. H. Van Beest, G. J. Kramer, K. . Shell, and R. A. Van Santen, “Vqlume 64, number 16 physical review letters 16,” Tech. Rep., 1990.
- [57] L. E. C. Francl and M M, “Atomic charges derived from electrostatic potentials – a detailed study,” *J. Comp. Chem.*, vol. 8, no. 6, pp. 894–905, 1987.
- [58] P. Robustelli, S. Piana, and D. E. Shaw, “Developing a molecular dynamics force field for both folded and disordered protein states.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 115, no. 21, E4758–E4766, 2018.
- [59] D. Tan, S. Piana, R. M. Dirks, and D. E. Shaw, “Rna force field with accuracy comparable to state-of-the-art protein force fields,” *Proc. Natl. Acad. Sci.*, p. 201713027, 2018.
- [60] J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. De Groot, H. Grubmüller, and A. D. MacKerell, “Charmm36m: an improved force field for folded and intrinsically disordered proteins,” *Nat. Methods*, vol. 14, no. 1, pp. 71–73, 2016. arXiv: 15334406.
- [61] S. Rauscher, V. Gapsys, M. J. Gajda, M. Zweckstetter, B. L. De Groot, and H. Grubmu, “Structural ensembles of intrinsically disordered proteins depend strongly on force field: a comparison to experiment,” *J. Chem. Theory Comput*, vol. 16, p. 1, 2015.
- [62] S. Piana, A. G. Donchev, P. Robustelli, D. E. Shaw, and D. E. Shaw, “Water dispersion interactions strongly influence simulated structural properties of disordered protein states,” 2015.
- [63] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, “Machine learning predictions of molecular properties: accurate many-body potentials and

- nonlocality in chemical space,” *J. Phys. Chem. Lett.*, vol. 6, no. 12, pp. 2326–2331, 2015.
- [64] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld, “Machine learning prediction errors better than dft accuracy,” 2017. arXiv: 1702.05532.
- [65] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, “Fast and accurate modeling of molecular atomization energies with machine learning,” *Phys. Rev. Lett.*, vol. 108, no. 5, p. 058 301, 2012.
- [66] N. Artrith and J. Behler, “High-dimensional neural network potentials for metal surfaces: a prototype study for copper,” *Phys. Rev. B - Condens. Matter Mater. Phys.*, vol. 85, no. 4, 2012.
- [67] G. C. Sosso, G. Miceli, S. Caravati, J. Behler, and M. Bernasconi, “Neural network interatomic potential for the phase change material geTe,” *Phys. Rev. B*, vol. 85, 2012.
- [68] R. Z. Khaliullin, H. Eshet, T. D. Kühne, J. Behler, and M. Parrinello, “Graphite-diamond phase coexistence study employing a neural-network mapping of the ab initio potential energy surface,” *Phys. Rev. B*, vol. 81, no. 10, p. 100 103, 2010.
- [69] V. L. Deringer and G. Csányi, “Machine learning based interatomic potential for amorphous carbon,” *Phys. Rev. B*, vol. 95, 2017.
- [70] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, “Machine learning of accurate energy-conserving molecular force fields,” *Sci. Adv.*, vol. 3, no. 5, e1603015, 2017.
- [71] S. Lorenz, A. Groß, and M. Scheffler, “Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks,” *Chem. Phys. Lett.*, vol. 395, no. 4-6, pp. 210–215, 2004.
- [72] O. A. Von Lilienfeld, R. Ramakrishnan, M. Rupp, and A. Knoll, “Fourier series of atomic radial distribution functions: a molecular

- fingerprint for machine learning models of quantum chemical properties,” *Int. J. Quantum Chem.*, vol. 115, no. 16, pp. 1084–1093, 2015. arXiv: 1307.2918.
- [73] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler, “Big data of materials science: critical role of the descriptor,” *Phys. Rev. Lett.*, vol. 114, no. 10, 2015. arXiv: arXiv:1411.7437v2.
- [74] J. Behler, “Erratum: “perspective: machine learning potentials for atomistic simulations” [j. chem. phys. 145, 170901 (2016)],” *J. Chem. Phys.*, vol. 145, no. 21, p. 219901, 2016.
- [75] S. Hajinazar, J. Shao, and A. N. Kolmogorov, “Stratified construction of neural network based interatomic models for multicomponent materials,” *Phys. Rev. B*, vol. 95, 2017.
- [76] M. O. J. Jäger, E. V. Morooka, F. Federici Canova, L. Himanen, and A. S. Foster, “Machine learning hydrogen adsorption on nanoclusters through structural descriptors,” *Npj Comput. Mater.*, vol. 4, no. 1, p. 37, 2018.
- [77] A. P. Bartok, R. Kondor, and d. . P. f. . h. j. . P. t. . O. u. . h. v. . . y. . . Csanyi Gabor,
- [78] A. P. Bartok, M. C. Payne, R. Kondor, and G. Csanyi, “Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons,” *Phys. Rev. Lett.*, vol. 104, no. 13, 2010. arXiv: 0910.1019.
- [79] T. Xie and J. C. Grossman, “Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties,” *Phys. Rev. Lett.*, vol. 120, no. 14, 2018. arXiv: 1710.10324.
- [80] J. S. Smith, O Isayev, and A. E. Roitberg, “Ani-1: an extensible neural network potential with dft accuracy at force field computational cost †,” 2017.

- [81] W. Pronobis, A. Tkatchenko, and K.-R. Mu, “Many-body descriptors for predicting molecular properties with machine learning: analysis of pairwise and three-body interactions in molecules,” *J. Chem. Theory Comput.*, vol. 14, p. 7, 2018.
- [82] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld, “Prediction errors of molecular machine learning models lower than hybrid dft error,” *J. Chem. Theory Comput.*, vol. 13, no. 11, pp. 5255–5264, 2017.
- [83] M. Rupp, R. Ramakrishnan, and O. Anatole Von Lilienfeld, “Machine learning for quantum mechanical properties of atoms in molecules,” *J. Phys. Chem. Lett.*, vol. 6, p. 46, 2015.
- [84] K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K. R. Müller, and E. K. U. Gross, “How to represent crystal structures for machine learning: towards fast prediction of electronic properties,” *Phys. Rev. B*, vol. 89, 2014.
- [85] J. Behler and M. Parrinello, “Generalized neural-network representation of high-dimensional potential-energy surfaces,” *Phys. Rev. Lett.*, vol. 98, no. 14, 2007.
- [86] O. Isayev, C. Oses, C. Toher, E. Gossett, S. Curtarolo, and A. Tropsha, “Universal fragment descriptors for predicting properties of inorganic crystals,” *Nat. Commun.*, vol. 8, p. 15 679, 2017.
- [87] S. V. Kalinin, B. G. Sumpter, and R. K. Archibald, “Big–deep–smart data in imaging for guiding materials design,” *Nat. Mater.*, vol. 14, no. 10, pp. 973–980, 2015.
- [88] C. Cazorla and J. Boronat, “Simulation and understanding of atomic and molecular quantum crystals,” *Rev. Mod. Phys.*, vol. 89, no. 3, 2017. arXiv: 1605.05820.

- [89] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, “Machine learning for molecular and materials science,” *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.
- [90] R. Ramprasad, R. Batra, G. Piliya, A. Mannodi-Kanakkithodi, and C. Kim, “Machine learning in materials informatics: recent applications and prospects,” *Npj Comput. Mater.*, vol. 3, no. 1, p. 54, 2017.
- [91] E. D. Cubuk, S. S. Schoenholz, J. M. Rieser, B. D. Malone, J. Rottler, D. J. Durian, E. Kaxiras, and A. J. Liu, “Identifying structural flow defects in disordered solids using machine-learning methods,” *Phys. Rev. Lett.*, vol. 114, no. 10, 2015. arXiv: 1409.6820.
- [92] D. C. Elton, Z. Boukouvalas, M. S. Butrico, M. D. Fuge, and P. W. Chung, “Applying machine learning techniques to predict the properties of energetic materials,” *Sci. Rep.*, vol. 8, no. 1, p. 9059, 2018.
- [93] S. Goedecker, “Minima hopping: an efficient search method for the global minimum of the potential energy surface of complex molecular systems,” *J. Chem. Phys.*, vol. 120, no. 21, pp. 9911–9917, 2004. arXiv: 0402136 [cond-mat].
- [94] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, “Quantum-chemical insights from deep tensor neural networks,” *Nat. Commun.*, vol. 8, p. 13890, 2017.
- [95] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, 1997.
- [96] G. Avendaño-Franco and A. H. Romero, “Firefly algorithm for structural search,” *J. Chem. Theory Comput.*, vol. 12, no. 7, pp. 3416–3428, 2016.
- [97] J. P. Lewis, K. R. Glaesemann, G. A. Voth, J. Fritsch, A. A. Demkov, J. Ortega, and O. F. Sankey, “Further developments in the local-orbital density-functional-theory tight-binding method,” *Phys. Rev. B*, vol. 64, no. 19, p. 195103, 2001.

- [98] J Harris, “Simplified method for calculating the energy of weakly interacting fragments,” *Phys. Rev. B*, vol. 31, no. 4, pp. 1770–1779, 1985.
- [99] W. M. C. Foulkes and R. Haydock, “Tight-binding models and density-functional theory,” *Phys. Rev. B*, vol. 39, no. 17, pp. 12 520–12 536, 1989.
- [100] F. Pedregosa FABIANPEDREGOSA, N. Alexandre Gramfort, V. Michel, B. Thirion BERTRANDTHIRION, O. Grisel, M. Blondel, P. Prettenhofer PETERPRETTENHOFER, R. Weiss, V. Dubourg, J. Vanderplas VANDERPLAS, A. Passos, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, P. Prettenhofer, J. Vanderplas, M. Brucher, M. Perrot an Edouard Duchesnay PEDREGOSA, A. Matthieu Brucher MATTHIEUBRUCHER, M. Perrot MATTHIEUPERROT, and C. F. Edouard Duchesnay EDOUARDDUCHESNAY, “Scikit-learn: machine learning in python gaël varoquaux,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [101] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: large-scale machine learning on heterogeneous distributed systems,” 2016. arXiv: 1603.04467.
- [102] T. J. Giese and D. M. York, “Representing potential energy surfaces by high- dimensional neural network potentials quantum mechanical force fields for condensed phase molecular simulations,” *J. Phys. Condens. Matter*, vol. 26, 2014.
- [103] Z. Li, J. R. Kermode, and A. De Vita, “Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces,” *Phys. Rev. Lett.*, vol. 114, no. 9, 2015.

- [104] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, “Quantum chemistry structures and properties of 134 kilo molecules,” *Sci. Data*, vol. 1, p. 140022, 2014.
- [105] B. Huang and O. A. Von Lilienfeld, “Communication: understanding molecular representations in machine learning: the role of uniqueness and target similarity,” *J. Chem. Phys.*, vol. 145, no. 16, p. 161102, 2016. arXiv: 1608.06194.
- [106] C. Sutton, L. M. Ghiringhelli, T. Yamamoto, Y. Lysogorskiy, L. Blumenthal, T. Hammerschmidt, J. Golebiowski, X. Liu, A. Ziletti, and M. Scheffler, “Nomad 2018 kaggle competition: solving materials science challenges through crowd sourcing,” 2018. arXiv: 1812.00085.
- [107] S. Zeng, G. Li, Y. Zhao, R. Wang, and J. Ni, “Machine learning-aided design of materials with target elastic properties,” *J. Phys. Chem. C*, vol. 123, pp. 5042–5047, 2019.
- [108] A. P. Bartok, S. De, C. Poelking, N. Bernstein, J. R. Kermode, G. Csanyi, and M. Ceriotti, “Machine learning unifies the modeling of materials and molecules,” *Sci. Adv.*, vol. 3, no. 12, e1701816, 2017.
- [109] A. P. Bartok and G. Csanyi, “Gaussian approximation potentials: a brief tutorial introduction,” *Int. J. Quantum Chem.*, vol. 115, no. 16, pp. 1051–1057, 2015. arXiv: 1502.01366.