JACOBS
UNIVERSITY

Kernel-based molecular machine learning for vector-valued properties

by

Antonia Savu

Bachelor Thesis in Computer Science

Submission: May 17, 2023

Supervisor: Prof. Dr. Peter Zaspel

Constructor University | School of Computer Science and Engineering

Statutory Declaration

Family Name, Given/First Name	Savu, Antonia
Matriculation number	30004198
Kind of thesis submitted	Bachelor Thesis

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

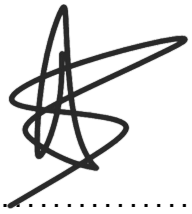
This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

17.05.2023



.....

Date, Signature

Abstract

We investigate the performance of kernel-based machine learning models on molecular data, specifically focusing on vector-valued predictions. The objective is to determine whether models applied to the entire vector-valued output are better at predicting transition dipole moments of molecular trajectories than models that predict the vector components independently. To this end, we evaluate three algorithms: Kernel Ridge Regression and Gaussian Process Regression as single task models, and Multitask Gaussian Process Regression as the multitask model. These algorithms are applied to two molecular datasets: a trajectory of benzene molecules and a trajectory of porphyrin molecules, with their respective transition dipole moments. The performance of the models is assessed using learning curves, which show how the errors of the models behave with increasing training sizes, and scatter plots, which help visualise the difference between predicted values and true output values. Our findings reveal that multitask models are not better at predicting transition dipole moments for the molecular trajectories considered in this study. On the contrary, Multitask Gaussian Process Regression yielded worse results than the single task models.

Contents

1 Introduction

The study of quantum mechanics has a profound impact in many fields of our society. In particular, material and drug development specialists heavily rely on the understanding of quantum chemical properties for innovation [7]. Such properties include ground and excited state energies or the transition dipole moment of molecules. It is often of high interest to compute quantum properties for trajectories of molecules. Chemical or physical processes shift the atoms in a trajectory of the same molecule.

However, the calculation of quantum chemical properties has proven to be computationally expensive, slowing down progress in the industry and academia [17]. New computational approaches promise to aid in this regard, such as machine learning (ML). ML can be faster since it uses statistical relationships in data instead of conducting explicit computations [19]. Regression of the quantum properties required by researchers is one possible application of ML.

One ML category of high importance for material and drug development is molecular machine learning [15]. Molecular machine learning (MML) revolves around solving problems using ML in the area of chemistry with a focus on molecular structure. Nevertheless, the field of MML is still growing and a definitive model for quantum chemical properties regression has not yet been determined.

This thesis focuses on the prediction of vector-valued quantum chemical properties. The datasets used in this thesis consist of a trajectory of molecules and their respective transition dipole moment. In particular, the efficiency of kernel-based machine learning models on such outputs is investigated. We compare two distinct approaches to predict the transition dipole moment. The first one is a classic regression model, which has as output a scalar. Such a machine learning model is independently applied on each component of the vector. These components are then concatenated to predict the quantum property. The second model is a vector valued machine learning model, which is applied on the whole vector. One approach for this is known as multitask learning [4]. Since the components are not independently computed, it is expected that the results vary from the first approach.

The objective of the thesis is to discover whether vector valued multitask machine learning models are better at predicting non-scalar quantum chemical properties than models which are independently applied on each component of a vector.

The model used for the independent task computation is Kernel Ridge Regression (KRR) [23]. This ML model combines the classic ridge (L_2 regularization) regression with a kernel function. Another alternative model for this approach is the Gaussian Process Regression [6]. For the vector valued model, the thesis uses Multi Task Gaussian Process Regression (MT-GPR) [2]. This is an adaptation of the well known Gaussian Process Regression applied in a multi-output setting such as the computation of transition dipole moment vectors.

A benchmark experiment is drafted on the given trajectories and their respective transition dipole moment values using the aforementioned approaches. The performance of the models are thoroughly evaluated in a dedicated analysis section, where the results of the three models are compared and a conclusion is drafted. To evaluate the performance of the models, learning curves are used to visualise the Mean Absolute Error (MAE) based on a validation set. The MAE is computed at different training sample sizes.

The structure of this thesis is as follows: Section 2 provides a brief introduction to the datasets and outlines the mathematical notions required for the thesis. Additionally, this section includes a review of existing literature and the current state of the art. Section 3 discusses the concrete implementation of the ML models and the design of the benchmark experiment. In Section 4, the models introduced in the previous section are analyzed, and their results are compared. The final section of the thesis presents the conclusion and an overview of the work done.

2 Background

2.1 Dataset

We use two distinct datasets for the experiment. The input data of each dataset consists of a trajectory of benzene molecules and a trajectory of porphyrin molecules, respectively. These trajectories come in the form of a xyz type file. Such a file format is often used to store atomic coordinates.

In order for a machine learning model to be able to use this data as input, it must first be parsed into an appropriate encoding [16]. There are many different descriptors for molecular data. The one used in this thesis are Coulomb matrices.

The Coulomb Matrix is an encoding of the interactions between atoms of a molecule. The entries of the matrix can be computed using the formula

$$c_{ij} = \begin{cases} 0.5Z_i^{2.4} & i = j \\ \frac{Z_i Z_j}{||R_i - R_j||} & i \neq j \end{cases},$$

where Z_i is the atomic number (nuclear charge) of atom i, and R_i is its position in atomic units. It is also worth noting that this matrix is symmetric by structure, and has as many rows and columns as there are atoms in the molecule [20].

For the benzene molecules, the Coulomb Matrix has 12 rows and 12 columns. This is due to the fact that a benzene molecule consists of 12 atoms. However, given the symmetry property, it suffices to consider only a subset of the matrix. In addition to this, the matrix gets vectorized for efficiency reasons. Therefore an array with only 78 elements is an appropriate descriptor for the benzene molecular data. The same approach is taken for the porphyrin molecules, which consist of 82 atoms. In this case, the Coulomb Matrix has of 82 rows and 82 columns. However, as with the benzene molecules, the symmetry properties of the Coulomb Matrix allow for a reduced subset of it to be considered. Consequently, an array with only 3403 elements serves as an appropriate descriptor for the porphyrin molecular data.

Given these preliminary observations , the inputs can be modeled in a stochastic sense as random vectors

$$X_{benzene} : \Omega_{benzene} \longrightarrow \mathbb{R}^{78}, X_{porphyrin} : \Omega_{porphyrin} \longrightarrow \mathbb{R}^{3403},$$

where $\Omega_{benzene}$ and $\Omega_{porphyrin}$ are underlying sample spaces. The task for the machine learning model is to predict the transition dipole moment. This is a quantum chemical property usually represented as a 3-dimensional array. The output is then given as a random vector

$$Y_{benzene} : \Omega_{benzene} \longrightarrow \mathbb{R}^3, Y_{porphyrin} : \Omega_{porphyrin} \longrightarrow \mathbb{R}^3.$$

The experiments presented in the thesis are restricted to a sample size of 768 observations and 151 observations respectively for the two datasets. Now that the features and output have been defined, the data sets can be formally presented as

$$\mathcal{T}_{benzene} = \{(x_i, y_i)\}_{i=1}^{768}, \mathcal{T}_{porphyrin} = \{(x_i, y_i)\}_{i=1}^{151}.$$

2.2 ML Background

2.2.1 Regression

Often in traditional supervised learning, regression predicts a scalar output. Such a model is concerned with one task at a time. Given a D-dimensional input X and a 1-dimensional output Y, the task is to find the function $f : \mathbb{R}^D \longrightarrow \mathbb{R}$ for the training set $\mathcal{T} = (x_i, y_i)_{i=1}^N$ such that the expected prediction error function

$$EPE(f) := E(L_2(Y, f(X)))$$

is minimal, where L_2 denotes the loss function $L_2(Y, f(X)) = (Y - f(x))^2$. That means, we want to find a function $f(X) = \hat{Y}$ such that $\hat{Y} \approx Y$.

In addition, it is known that the optimal function for this is $E(Y|X = x)$, usually referred to as the regressor. However, it is typically impossible to compute the expectation, since we do not know the probability distribution function $\rho(x, y)$. With this in mind, several models have been proposed throughout the years to find a function that approximates the regressor. One of the most popular models that is also a foundation for the machine learning models used in this thesis is linear regression.

2.2.2 Linear Regression

Linear regression proposes that the function mentioned earlier f is linear. With X and Y given as above, the linear model is defined as

$$f_\beta(X) = \beta_0 + \sum_{j=1}^{D} X_j \beta_j,$$

where β is a $D + 1$ vector of unknown parameters. These parameters are used to fine tune the model.

A parameter estimator is a method for determining the parameters β. The least squares estimator is a regularly used one

$$\hat{\beta} = \arg\min_\beta \sum_{i=1}^{N} L_2(y_i, f_\beta(x_i)).$$

To uniquely solve such an estimator, it is assumed that the matrix $\mathcal{X} = \left(\begin{array}{c|c} \mathbf{1} & \begin{array}{c} \boldsymbol{x}_1^\top \\ \boldsymbol{x}_2^\top \\ \vdots \\ \boldsymbol{x}_N^\top \end{array} \end{array}\right)$, where $x_1, x_2 \ldots x_N$ are the input measurements defined above is a full column matrix. Such a matrix will not have full column rank if there are more parameters than training data, leading to an undetermined system of equations.

This problem where the matrix does not have full column rank can be overcome by applying regularization to the model. Regularization adds another condition to the minimization problem. An often used model is the L_2 regularization linear regression, also known as ridge regression.

2.2.3 Ridge Regression

Ridge Regression enriches the classic linear model by adding a regularization parameter λ to the estimator. This implicitly adds bias to the model, but significantly reduces variance. Bias can be defined as the deviation of the mean of the predictor from the exact model. As with any machine learning model, there is a bias-variance trade-off [3].

The least squares estimator for linear ridge regression is

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} L_2(y_i, f_\beta(x_i)) + \lambda ||\beta||_2^2,$$

where

$$||\beta||_2^2 = \sum_{i=1}^{N} \beta_i^2.$$

2.2.4 Multitask Learning

In the case of transition dipole moment computation and many other real life examples, many ML models are too simple by construction and cannot handle vector valued data.

To overcome this issue, we present two possibilities:

The first one is to sequentially apply a model on each vector component. Given a vector output $Y : \Omega \longrightarrow \mathbb{R}^N$, of the form $y_i = \left(y_i^{(1)}, y_i^{(2)} \quad \ldots, y_i^{(N)}\right)^T$ we build N models $f_{ML^{(k)}}(X)$ where $k = 1, \ldots, N$.

This method offers simplicity and keeps the initial regression model. However, all the values are computed independently of each other. Because of this, the approach fails to capture the possibility of a relation between vector components.

This motivates the second approach proposed in this thesis, multitask learning. As the name suggests, multitask learning computes multiple tasks concurrently. Since these tasks are not independently learned, the model is more likely to perform well on related tasks. The assumption is that the tasks can share information with each other, leading to a better prediction [4].

2.3 Models

One of the challenges many machine learning models face is that the complexity of the model increases drastically for high dimensional inputs. This concern reduces the effectiveness of classic linear models such as the ones presented above.

A proposed solution are kernel based models [11], which utilise a kernel function. A so-called kernel function allows for mapping of training data to a higher dimensional feature space, while maintaining low computation complexity. This is done by applying what is known as the Kernel Trick [10].

2.3.1 Kernel Ridge Regression

Combining ridge regression with a kernel function results in a model usually referred to as Kernel Ridge Regression (KRR). This model could also be considered as a nonlinear

version of ridge regression [20]. The Kernel Ridge Regression model has the form

$$f_\beta(X) = \sum_{j=1}^{N} \beta_j k(X, x_j),$$

where k is the kernel function.

There are many different kernels that can be used for KRR, such as a linear kernel, Laplacian kernel, or a Gaussian kernel [11]. In this thesis, the Gaussian Kernel is used for the implementation of KRR. The Gaussian kernel is a function $k : \mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R}$ defined by

$$k(x, x') := e^{-\frac{||x-x'||_2^2}{2\sigma^2}},$$

where σ^2 is a scalar parameter that represents the width of the kernel.

With the model defined above and the kernel feature space $\mathcal{H}$, the minimization problem of finding the optimal parameters β becomes

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} L_2(f_\beta(X_i) - y_i) + \lambda ||\beta||_{\mathcal{H}}^2.$$

2.3.2 Gaussian Process Regression

Another option for single task regression is the Gaussian Process Regression (GPR) model. Unlike KRR, GPR is a non-parametric model. This means, GPR focuses on the similarity between data points rather than on imposing a strict structure upon the data. As such, for similar observations x_i, x_j the model predicts similar outputs y_i, y_j. The concept of similarity is defined by the kernel used for GPR.

With previous regression models, it was assumed that there is one unique function f which fits the data. In reality, there may be infinite such functions. GPR models describe a probability distribution over such functions [24]. GPR assumes that the initial infinite functions are samples from a Gaussian distribution. Since the input data contains more than one feature variable, we can formally say that the prior distribution of the initial functions is multivariate Normal (MVN) [24].

These functions have not yet observed any data. When an observation is used to train the model, only the functions which fit the observation are kept. This new subset of functions can be referred to as the posterior distribution. This process is repeated whenever there is a new observation, and the posterior becomes again the prior, used for calculating a new posterior. Once the there are no more observations for training, the mean of the posterior distribution is calculated and this mean is used for regression.

Such a regression function can be defined as

$$P(f|X_{observed}) = \mathcal{N}(f|\mu, K),$$

where $X_{observed} = [x_1, \ldots, x_N]$ are the N observed data points, μ is a N dimensional vector of mean functions, and K is a covariance function defined as $K_{ij} = k(x_i, x_j)$. The most common used covariance for GPR is the radial basis function kernel $cov(x_i, x_j) = \exp\left(-\frac{(x_i - x_j)^2}{2}\right)$. For consistency reasons, this thesis uses the Gaussian kernel.

The covariance in GPR is used to determine the shape of the distribution and to define the similarity of two points. In the case of the radial basis function, similarity is associated with distance.

Using the observed data $X_{observed}$ and new, unseen data points X_{new} we can define the joint distribution of the observed output data points $y_{observed} = [f(x_1), \ldots, f(x_N)]$ and the output for y_{new} for the points X_{new} , formally $P(y_{observed}, y_{new}|X_{observed}, X_{new})$ as

$$\begin{bmatrix} y_{observed} \\ y_{new} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + \sigma_n^2 I & K_* \\ K_*^T & K_{**} \end{bmatrix}\right)$$

where $K = K(X_{observed}, X_{observed})$ is the matrix of the covariances for all pairs of observed points, $K_* = K(X_{observed}, X_{new})$ for all pairs of observed and new points, and $K_{**} = K(X_{new}, X_{new})$ for all pairs of new points [18].

From this formula, the conditional probability $P(y_{new}|y_{observed}, X_{observed}, X_{new})$ needed for the regression task can be computed as

$$y_{new}|y_{observed}, X_{observed}, X_{new} \sim \mathcal{N}(K_*^T K y_{observed}, K_{**} - K_*^T K^{-1} K_*).$$

One of the great advantages of GPR is that unlike other models, it has inherent uncertainty measures over predictions [18]. This is helpful when making important decisions based on ML predictions.

2.3.3 Multitask Gaussian Process Regression

Multitask Gaussian Process Regression (MT-GPR) [2] is an extension of the GPR model presented above. While GPR models a single scalar value, MT-GPR concurrently solves multiple tasks.

Unlike in traditional GP, the tasks are not completely independent. The relationship between tasks is captured by a $M \times M$ matrix called the inter-task covariance matrix, where M represents the number of tasks.

The values of the inter-task covariance matrix indicate how much other tasks influence the prediction of a given task. When the value $c_{i,j}$ is close to one, it indicates that the two tasks i and j are highly similar and therefore heavily influence each other's predictions. Consequently, a value close to zero indicates that the tasks have significantly different input-output relationships and only marginally impact the predictions.

The formula for the s^{th} task for an input x_* is given by

$$f^{(s)}(x_*) = \sum_{t=1}^{M} c_{s,t} \sum_{i=1}^{N} k(x_*, x_i),$$

where the matrix c is the inter-task covariance, which captures how much weight a task has on another, and k is the kernel matrix. As with the previous models, there are various options for picking the kernel function. This thesis focuses on the Gaussian kernel.

2.4 Related Work

Quantum Machine Learning has gained significant research interest, and there are excellent resources that can serve as starting points. One such literature often cited in the field

is [20], which provides a foundational understanding of quantum data representations, ML models and best practices.

The prediction of transition dipole moments is of high importance, as this quantum chemical property poses high importance in the context of material and drug development. [14] predicts the values of molecular dipole moments using random forest regression.

Kernel-based models are popular in the field of quantum chemical computing, as shown by numerous publications like [25]. Such papers frequently model the chemical characteristics of molecules using Kernel Ridge Regression or comparable kernel-based models.

Multitask learning in has been heavily investigated in an effort to further improve the state-of-the-art for modeling vector valued data. Rich Caruana's paper on multitask learning [4] is considered by many one of the most important scientific papers on the topic. More specifically, [2] investigates multitask learning in the context of Gaussian process regression. Multitask Gaussian process regression is a popular approach in various domains of scientific research, particularly in the areas of chemical and biomedical development. For instance, [8] explores the application of MT-GPR in biomedical research. Meanwhile, [21] presents a more general overview of the potential uses of multitask models for chemical data analysis.

This thesis hopes to enrich the literature by providing a comparison between multitask models and regular kernel models specifically for quantum chemical data. Such a comparison could further initiate research discussions on the topic of quantum machine learning optimization.

3 Description of the Investigation

3.1 Implementation Details

The primary objective of this thesis is to evaluate the performance of kernel-based machine learning models for vector-valued molecular data. Furthermore, it is of interest to explore the potential of multitask algorithms in improving quantum machine learning. To accomplish this goal, we have implemented four distinct kernel-based machine learning algorithms on quantum chemical data. By doing so, we aim to gain a comprehensive understanding of the strengths and limitations of these models and identify areas where they can be improved.

Two sets of dummy data are generated to ensure the correctness and robustness of the algorithms. The first dataset was created using three output functions $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, and $f_3(\mathbf{x})$, which are combined to create a three-dimensional output vector $\mathbf{Y}$. These output functions are linear combinations of sine and cosine functions, as well an exponential function. This functions take as input a five-dimensional vector $\mathbf{X}$. As such, the output components are defined as

$$
\begin{aligned}
f_1(\mathbf{X}) &= \sin(X_1) + \cos(X_2) + e^{X_3} + \sin(X_4) + \cos(X_5) \\
f_2(\mathbf{X}) &= \cos(X_1) + e^{X_2} + \sin(X_3) + \cos(X_4) + e^{X_5} \\
f_3(\mathbf{X}) &= \sin(X_1) + e^{X_2} + \cos(X_3) + \sin(X_4) + e^{X_5},
\end{aligned}
$$

where $\mathbf{X}$ represents the input vector of five dimensions, and X_i denotes the i-th feature of $\mathbf{X}$. All the features of the input vector are randomly generated with values ranging between 0 and 1, and the dataset consists of 3000 observations, each with a corresponding three-dimensional output vector $\mathbf{Y}$. A second dataset is produced by applying a 45-degree rotation about all axes to the output of the first dummy dataset. This is done to closely resemble the circumstances of molecular data, as such a rotational element is often present in the transition dipole moment.

The experiment component of this thesis can be broadly divided into three main areas: pre-processing of molecular data, development of machine learning algorithms, and performance analysis. This section details how the data is represented and any additional preparatory steps, how the algorithms are structured as well as the method used to evaluate said algorithms.

3.2 Data Representation and Pre-Processing

This thesis primarily focuses on two datasets: the DFTB trajectory of benzene molecules and a porphyrin trajectory, along with their respective dipole moments. Density Functional Tight Binding(DFTB) [12] is a quantum mechanical method used to describe the structure of a molecule. The molecules inside a trajectory have different geometries, leading to different dipole moments.

The two trajectories are stored in an xyz file format. However, in order for this molecular data to be usable, it needs to be parsed into an appropriate format. The format used in the thesis are Coulomb Matrices. To apply the Coulomb matrices, the Quantum Machine Learning (QML) toolkit is utilized. QML [5] is a Python library that provides a range of molecular representations and other functions that are essential for working with quantum data.

The library computes only one molecule representation at a time. To work around this limitation, the trajectory files containing the data are split into 768 benzene xyz files and 151 porphyrin xyz files, respectively. For each of these xyz files generated from the trajectory data, the corresponding Coulomb matrix is calculated through an iterative process. The resulting list of Coulomb matrices is then saved in a numpy binary file format, which allows for efficient storage of the data. One such file is created for each dataset. This approach ensures optimal utilization of computational resources and easy accessibility of the data for analysis without any redundancy. The binary files are used as the input for the ML algorithms.

3.3 ML Algorithms

To obtain the best results, hyperparameter tuning is applied. Tuning the hyperparameters involves selecting the optimal values for these parameters, which can significantly improve the accuracy and efficiency of the algorithm [1]. The hyperparamater we optimized for is the width of the kernel. In order to decrease the complexity of the hyperparameter tuning algorithm, the regularization parameter of 10^{-9} and Gaussian kernel are fixed. For the benzene dataset, we use a training size of 200 and a testing size of 50 to identify the optimal kernel width for each component of the transition dipole moment. The hyperparameter tuning algorithm aims at minimizing the mean absolute error. To this end, the algorithm tries all the kernel widths within the range of 1 to 5000, in increments of 100, to find the one which yields the lowest mean absolute error.

The algorithm determined the optimal kernel width values for the benzene dataset to be 100 for the X component, 200 for the Y component, and 4900 for the Z component. As the size of the porphyrin dataset is limited, no significant results were obtained from hyperparameter tuning. Therefore, a kernel width of 100 is chosen for all three transition dipole moment components of the dataset. For both of the dummy datasets, the algorithm yielded a kernel width of 100 for all three components.

Algorithm 1 Hyperparameter Tuning

function HYPERTUNING($X_{train1}, X_{test1}, \sigma$)
    $min\_mae \leftarrow \infty$
    $optimal\_sigma \leftarrow$ **None**
    for σ_val **in** σ **do**
        $mae\_templ \leftarrow$ KRR($X_{train1}[0:200]$, $X_{test1}[0:50]$, $y_{train1}[0:200]$, $y_{test1}[0:50]$, σ_val)
        if $mae\_templ < min\_mae$ **then**
            $min\_mae \leftarrow mae\_templ$
            $optimal\_sigma \leftarrow \sigma\_val$
    return $optimal\_sigma$

The first regression algorithm implemented is Kernel Ridge Regression utilizing the QML library. This library offers a built in function for KRR. This function takes as input the training data, test data, and the optimal kernel width parameter found previously, and returns the mean absolute error.

Algorithm 2 Kernel Ridge Regression

Input:
x_{train}: Input data for training, corresponding to an array of Coulomb matrices
x_{test}: Input data for testing, corresponding to an array of Coulomb matrices
y_{train}: Output data for training, corresponding to an array of scalar values
y_{test}: Output data for testing, corresponding to an array of scalar values
σ: Kernel width
Output
MAE: Mean absolute error over the test samples of the model

function KRR($x_{train}, x_{test}, y_{train}, y_{test}, \sigma$)
    $K \leftarrow$ gaussianKernel($x_{train}, x_{train}, \sigma$)
    $K[\text{diag}(K)] \leftarrow K[\text{diag}(K)] + 10^{-9}$
    $\alpha \leftarrow$ choleskySolve(K, y_{train})
    $K_s \leftarrow$ gaussianKernel($x_{train}, x_{test}, \sigma$)
    $y_{predict} \leftarrow \alpha^T K_s$
    $MAE \leftarrow$ meanAbsError($y_{predict}, y_{test}$)
    return MAE

Similarly, Kernel Ridge Regression is implemented using scikit-learn [13]. Scikit-learn is a popular Python library that is used for machine learning, providing all the necessary tools for data analysis and modeling tasks. This library also has a built in function for KRR which is used. For the sake of consistency, the same hyperparameters are used.

As previously mentioned, Gaussian Process Regression (GPR) is another approach similar to KRR. For implementing GPR, we use GPyTorch [9], a PyTorch-based library for GP modeling. With GPyTorch, a GPR algorithm is defined to predict the molecular data. The model is initialised with the same hyperparameters used in KRR to yield comparable results.

Algorithm 3 Gaussian Process Regression

Input:
$X_{train} \in \mathbb{R}^{n \times d}$: Input data for training
$y_{train} \in \mathbb{R}^{n}$: Output data for training
$X_{test} \in \mathbb{R}^{m \times d}$: Input data for testing
σ_n: Observation noise
k: Kernel function
Output
$\mu_* \in \mathbb{R}^{m}$: Mean prediction for the test data

function GPR($X_{train}, y_{train}, X_{test}, \sigma_n, k$)
    $K \leftarrow k(X_{train}, X_{train})$
    $K[\text{diag}(K)] \leftarrow K[\text{diag}(K)] + \sigma_n^2$
    $L \leftarrow$ cholesky(K)
    $\alpha \leftarrow L^T \setminus (L \setminus y_{train})$
    $K_* \leftarrow k(X_{test}, X_{train})$
    $\mu_* \leftarrow K_* \alpha$
    return μ_*

Since the transition dipole moment is a three-dimensional vector, the algorithms men-

tioned above are applied independently to each of the three dimensions. However, Multitask Gaussian Process Regression (MT-GPR) is able to compute the transition dipole moment in a single iteration, as it calculates the vectors of the components in a dependent manner. The Multitask Gaussian Process Regression (MT-GPR) algorithm uses a inter-task covariance matrix rank of 1, which implies that the components have a lesser impact on each other than in the case of a full matrix of rank 3. This value was chosen after considering the three possible options and determining that it yielded the best performance. This algorithm is also implemented using GPyTorch.

3.4 Evaluation of Models

To evaluate the models presented above, all datasets are split into a training set and a validation set. We assume a 80-20 split, which means that 80% of the data is assigned to the training set and the rest 20% to the validation set. The models are first trained using the training set and then evaluated on their prediction on the validation set. To measure their accuracy, we use the Mean Absolute Error (MAE) on the validation set. MAE measures the average absolute difference between the predicted values and the real values. MAE can be defined as

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y_i}|\,,$$

where y_i are the real outputs of the validation set, $\hat{y_i}$ are the predicted outputs, and N is the size of the validation set.

The models' performances are evaluated using a learning curve, which shows how the mean absolute error (MAE) changes with increasing training set size. Training set sizes in powers of two ($2^1, 2^2 \ldots 2^{int(\log_2 N)}$) are randomly selected from the initial training set, and the MAE is calculated on a fixed validation set. To compensate for the randomness of the selection process, this is repeated 10 times for each training set size. The same 10 shuffles are used for all algorithms, to ensure consistent results for a given dataset.

Another technique for evaluating a model is to utilize scatter plots. These charts show the relationship between the predicted values of the model and the actual output values of the validation set. A scatter plot with points close to the diagonal indicate a low error, while points far from the diagonal indicate a high error. For this method, the training size was equivalent to the length of the training set.

To ensure the results of the thesis can be reproduced, the following versions of software and packages are used: QML version 0.2.1, GPyTorch version 1.9.1, Scikit-learn version 1.2.2, and Python version 3.10.9. By using these specific versions, any reader can recreate the results presented in this thesis using the same code and input data. All the experiments were carried on a Computer node Titlis with the following specifications: sysGen/SUPERMICRO SuperServer SYS-1029GQ-TRT, 2x Intel® Xeon® Scalable Processor "Cascade Lake", Silver 4210, 2.20 GHz, 10-Core, 192GB (12x 16GB) DDR4 PC2933, 960GB Intel SSD D3-S4510 960GB, 2.5", SATA, NVIDIA® Quadro RTX 4000 GPU.

4 Evaluation of the Investigation

4.1 Dummy Data

4.1.1 Kernel Ridge Regression Results

As previously noted, one common approach to analyzing models is through the use of learning curves. Figures 1, 2, and 3 illustrate the learning curves obtained by applying Kernel Ridge Regression on the two dummy datasets. As expected, the mean absolute error decreases with the increase in training sizes. The plots do not show any notable spikes, nor do they show any unusual differences between the output components. This normal behaviour suggests that Kernel Ridge Regression is a good fitting algorithm for the data, especially as the overall mean absolute error is low. Furthermore, the learning curve for the rotated dummy dataset, as depicted in Figure 3, does not show any significant differences compared to the regular dummy set, highlighting the inherent similarity between the two sets. Figures 1 and 2 show the learning curves for the two different implementations of Kernel Ridge Regression, QML and scikit-learn, applied to the regular dummy dataset. The learning curves produced by both implementations are almost identical, which can be attributed to the similarity of the underlying mathematical approaches of the two packages. As a result, only the QML implementation is presented in following evaluations.

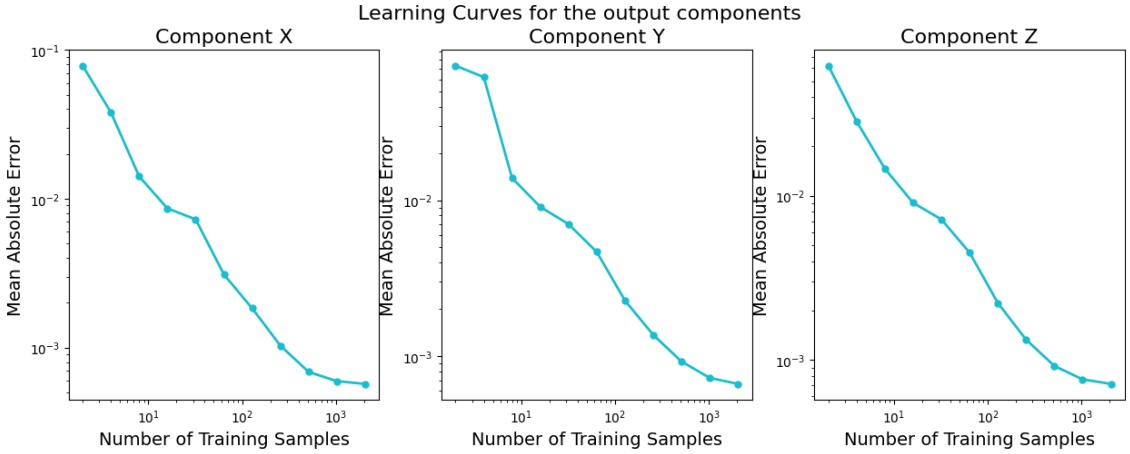


Figure 1: Learning Curves for the three components of the dummy output data using Kernel Ridge Regression, implemented via QML package, log scaled

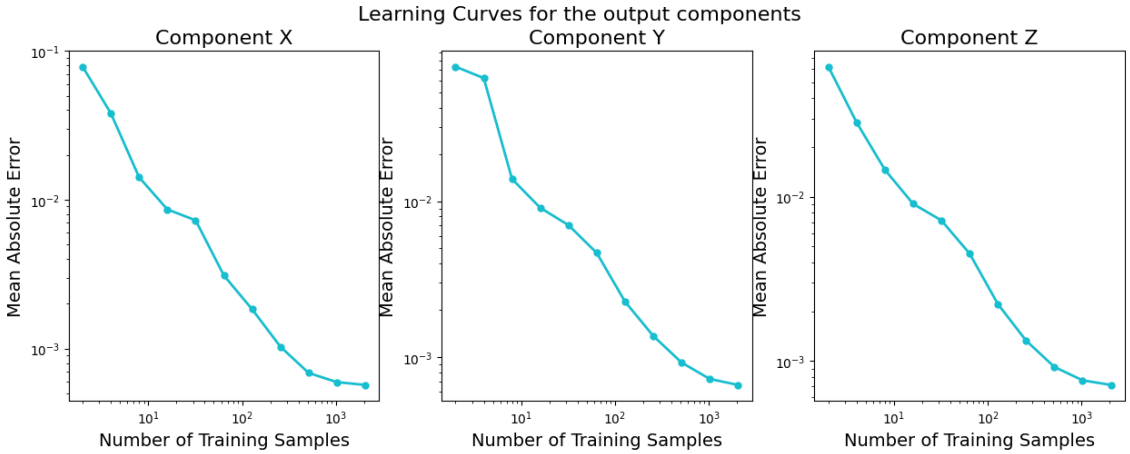


Figure 2: Learning Curves for the three components of the dummy output data using Kernel Ridge Regression, implemented via scikit-learn package, log scaled

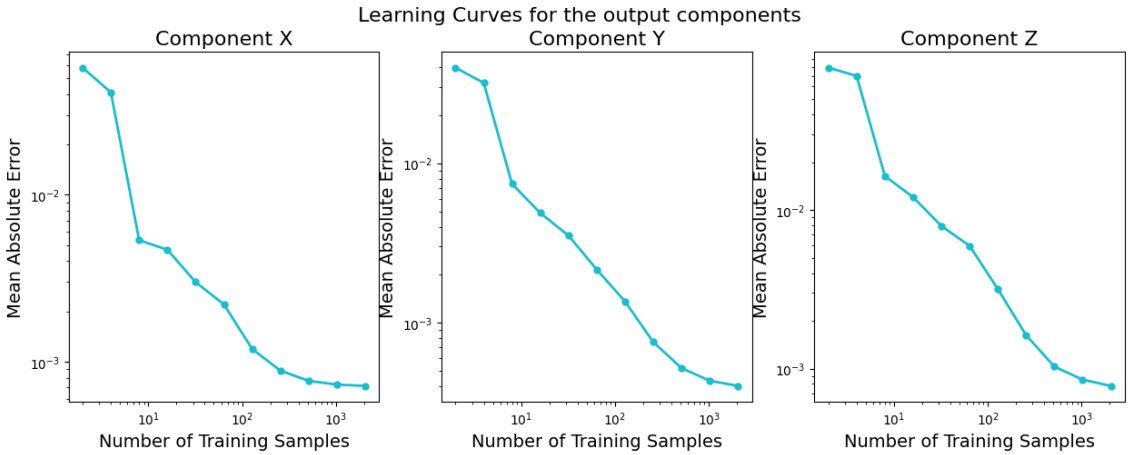


Figure 3: Learning Curves for the three components of the rotated dummy output data using Kernel Ridge Regression, implemented via QML package, log scaled

Figures 4 and 5 illustrate well the effectiveness of the Kernel Ridge Regression algorithm in predicting the output values for the two dummy datasets. The scatter plots provide a visual representation of the predicted values versus the actual output data for the validation set points when the algorithm is trained with the entire training set. The plots show that the predicted values are very close to the diagonal, indicating that the error is small. Moreover, the scatter plots of the regular output and the rotated output show no notable differences. This outcome confirms that the selected hyperparameters and algorithm are appropriate for both dummy datasets.

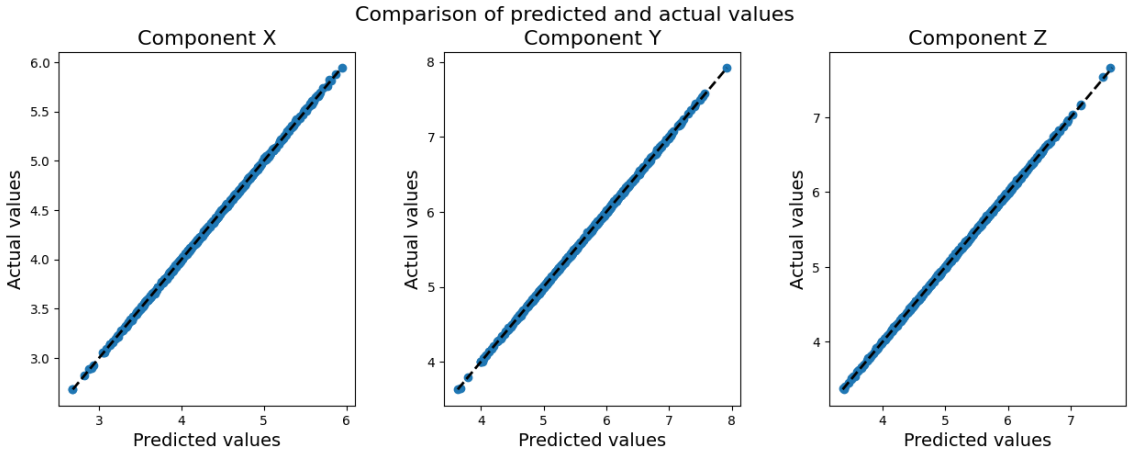


Figure 4: Scatter plot comparing the values of the dummy output and the predicted values using Kernel Ridge Regression

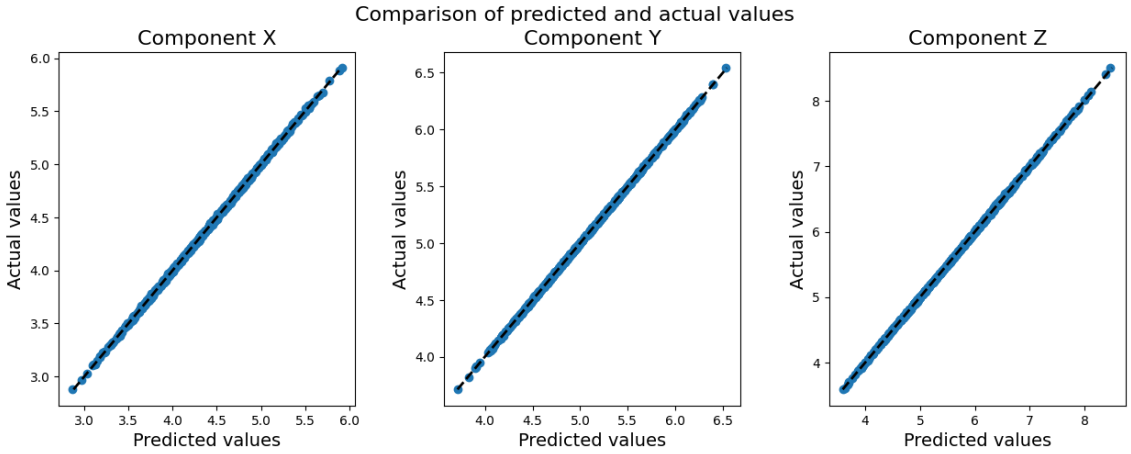


Figure 5: Scatter plot comparing the values of the rotated dummy output and the predicted values using Kernel Ridge Regression

4.1.2 Gaussian Process Regression Results

The performance of Gaussian Process Regression on the dummy datasets is evaluated using the same measures as for Kernel Ridge Regression. The learning curves for the dummy and rotated dummy datasets are depicted in Figures 6 and 7, respectively. The plots for all components closely resemble those obtained from the Kernel Ridge Regression algorithm, up to a higher training size. However, both datasets show a spike in mean absolute error for one of the training sizes. This finding suggests that Gaussian Process Regression may not be as suitable for the functions in the dummy output data as Kernel Ridge Regression.

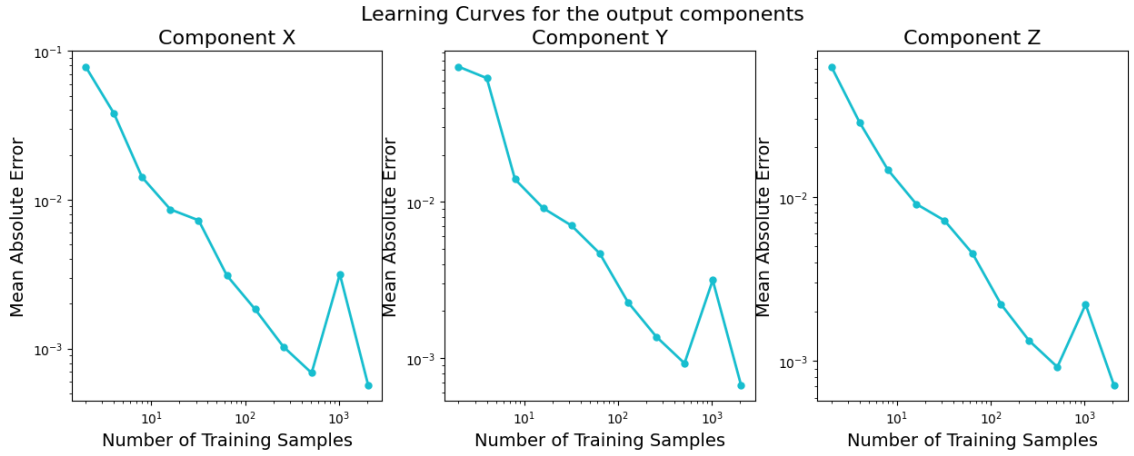


Figure 6: Learning Curves for the three components of the dummy output data using Gaussian Process Regression, log scaled

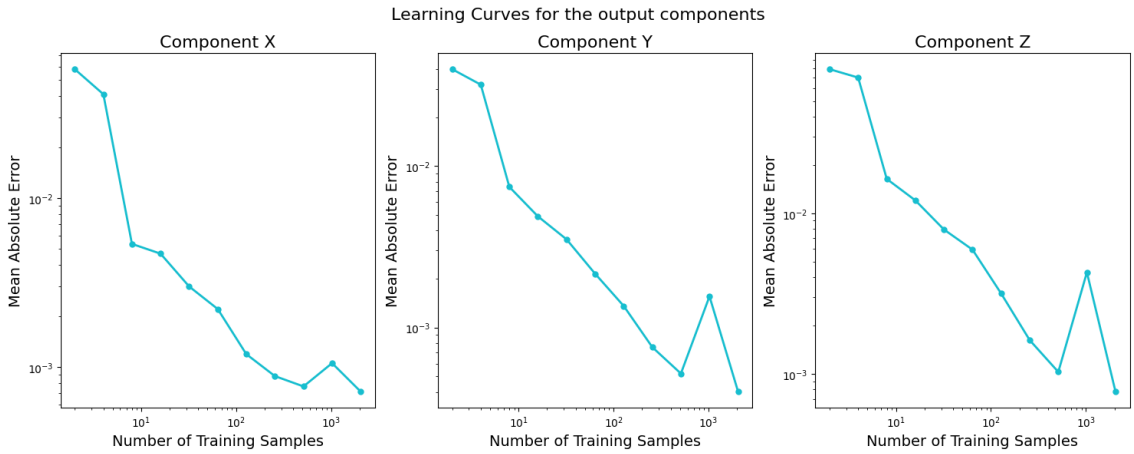


Figure 7: Learning Curves for the three components of the rotated dummy output data using Gaussian Process Regression, log scaled

Figures 8 and 9 present a comparison between the predicted values by the Gaussian Process Regression algorithm and the actual output data of the two datasets. The scatter plots reveal that the predicted values are close to the diagonal, which indicates that the algorithm has low error. Notably, the spike in mean absolute error observed in the learning curves is not present in these scatter plots, as they depict the predictions made when the algorithm was trained using the whole training set. The results from the two evaluation methods show that the Gaussian Process Regression algorithm is still an appropriate choice for modeling the two dummy datasets.

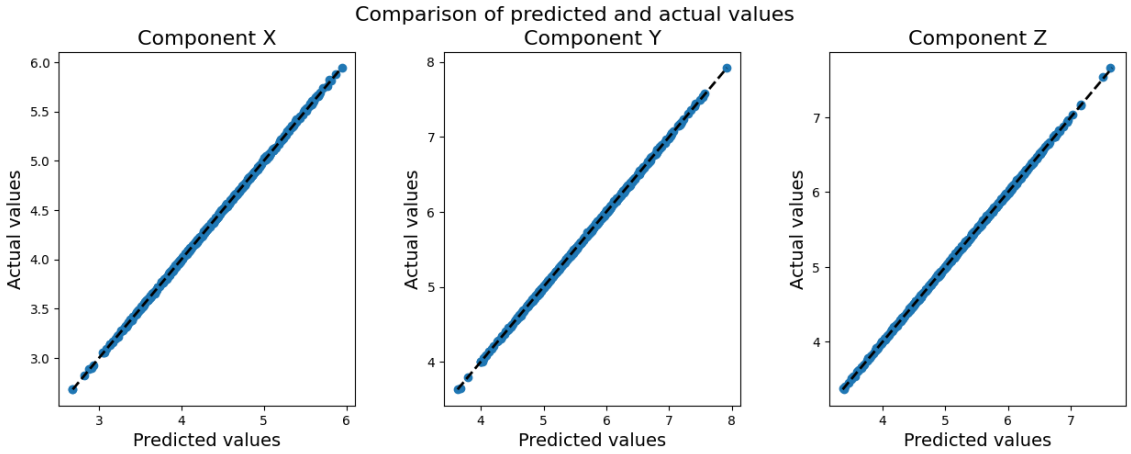


Figure 8: Scatter plot comparing the values of the dummy output and the predicted values using Gaussian Process Regression

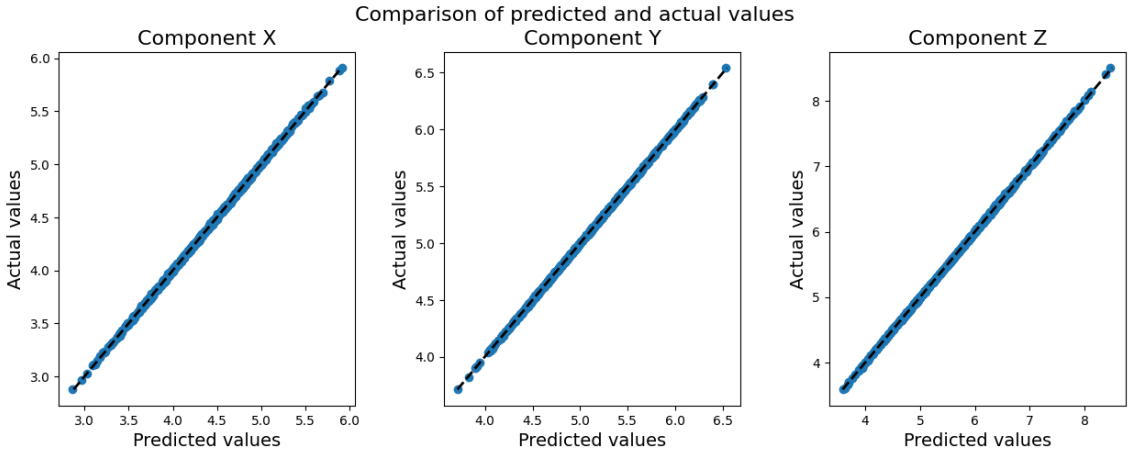


Figure 9: Scatter plot comparing the values of the rotated dummy output and the predicted values using Gaussian Process Regression

4.1.3 Multitask Gaussian Process Regression Results

Figures 10 and 11 illustrate the learning curves for the two dummy datasets using the Multitask Gaussian Process Regression Algorithm. It is important to highlight that, unlike the previous algorithms, there is only one learning curve per dataset and not per output component. This is because Multitask Gaussian Process Regression predicts all three components concurrently, rather than individually. Therefore, the mean absolute error is calculated directly as a mean of the error for each component. The two Figures show an expected behaviour of decreasing mean absolute error as training size increases, but the errors are significantly higher than in the previous algorithms. The order of magnitude difference between the previous algorithms and Multitask Gaussian Process Regression suggests that this algorithm is unsuitable for the dummy datasets. This could be due to the structure of the output functions of the dummy datasets, as they might not share a usable correlation. It is also worth noting that due to the nature of the Multitask Gaussian Process Regression algorithm, the plots obtain differ slightly with each code run. This can hinder the reproducibility of the results.

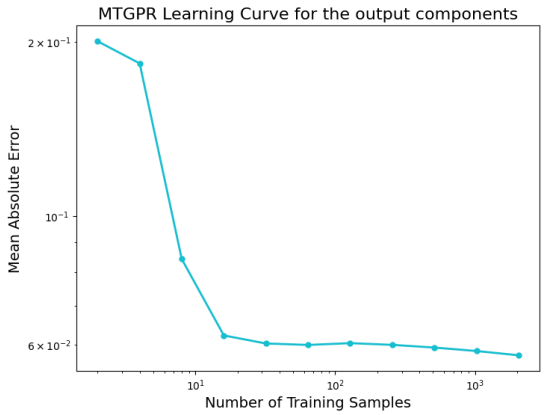


Figure 10: Learning Curve for the dummy output data using MT-GPR, log scaled

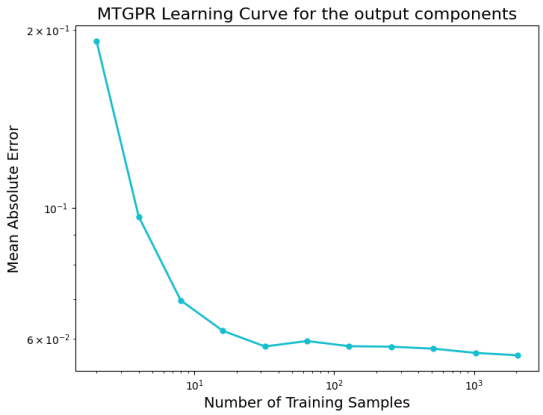


Figure 11: Learning Curve for the rotated dummy output data using MT-GPR, log scaled

To better visualise the relationship between the true output data and the values predicted by the Multitask Gaussian Process Regression, Figures 12 and 13 depict the scatter plots for the regular dummy data and the rotated dummy data, respectively. The order of magnitude difference in mean error can be easily observed in the scatter plots, as the values are far from the diagonal, indicating a high error. Multitask Gaussian Process Regression becomes evidently unsuitable especially when comparing the two scatter plots with Figures 4 and Figures 5 or Figures 8 and 9.

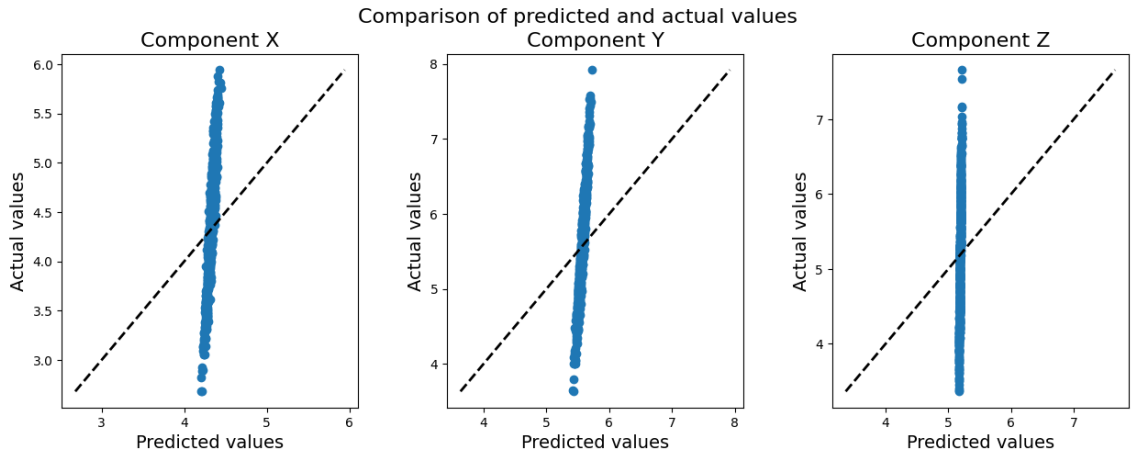


Figure 12: Scatter plot comparing the values of the dummy output and the predicted values using Multitask Gaussian Process Regression

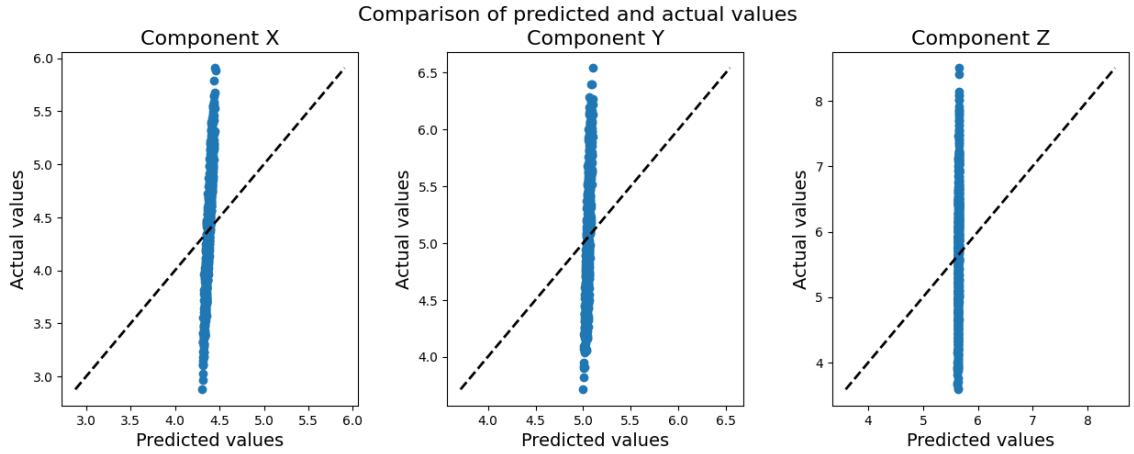


Figure 13: Scatter plot comparing the values of the rotated dummy output and the predicted values using Multitask Gaussian Process Regression

In conclusion, the two dummy datasets helped us gain valuable insight into the Regression algorithms used in this thesis. Both datasets paired well with Kernel Ridge Regression and Gaussian Process Regression, as they had low errors for the two algorithms. Multitask Gaussian Process Regression however performed poorly on the dummy datasets, showing high errors even when trained with the whole training set. These results could be attributed to various factors, such as the structure of the input data, the structure of the output functions, the shuffles selected or the size of the datasets. Additionally, we gained no significant insight by adding the rotational element in the output, as there are no notable differences between the regular and the rotated output on any of the algorithms, which could be due to the specific rotation chosen or the algorithms' ability to handle the rotation in the second dummy dataset.

4.2 Molecular Data

4.2.1 Kernel Ridge Regression Results

Figures 14, 15 and 16 display the learning curves for the two molecular datasets. For the benzene dataset, as shown in the first two Figures, an increase in the mean absolute error is observed in the early stages of the algorithm. This can be attributed to the presence of a pre-asymptotic range, in which the limited number of training samples and the inherent randomness of the training sets have a pronounced effect. However, as the number of training samples increases to an appropriate level, a steady decrease in mean absolute error is observed. Both Kernel Ridge Regression algorithms in Figures 14 and 15, using QML and using scikit-learn respectively, depict almost identical results. As mentioned in the discussion of the dummy datasets' results, this is due to the same mathematical concepts used. Given this increased similarity, the scikit-learn algorithm will be omitted from further discussions.

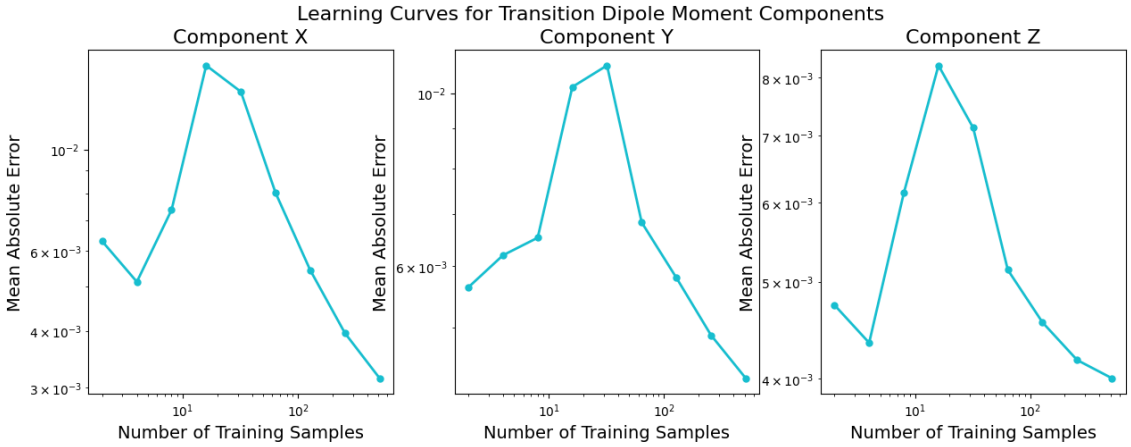


Figure 14: Learning Curves for the three components of the transition dipole moment of the benzene dataset using Kernel Ridge Regression, implemented via QML package, log scaled

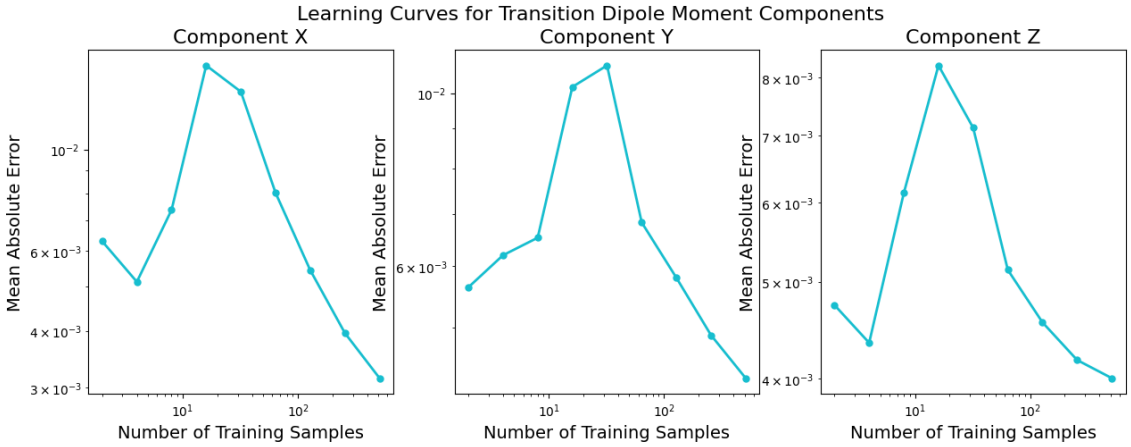


Figure 15: Learning Curves for the three components of the transition dipole moment of the benzene dataset using Kernel Ridge Regression, implemented via scikit-learn package, log scaled

For the porphyrin dataset, the observed learning curve in figure 16 follows a different trend. The errors are consistently high until the last training size, at which a strong decrease is observed. Additionally, the mean absolute errors present in the learning curves of the components are high. Both of these irregularities are a result of the very small sample size.

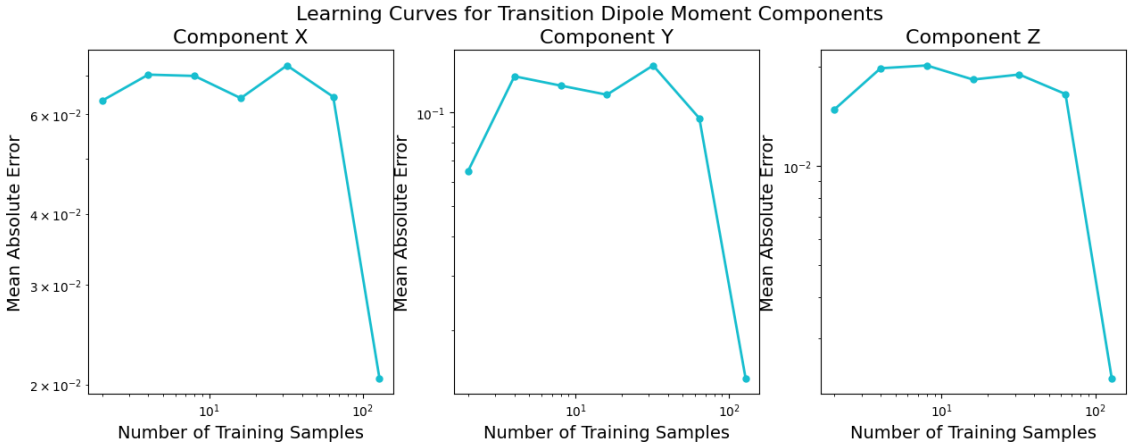


Figure 16: Learning Curves for the three components of the transition dipole moment of the porphyrin dataset using Kernel Ridge Regression, implemented via QML package, log scaled

Figures 17 and 18 represent the scatter plots for the two molecular datasets, using benzene and porphyrin molecules, respectively. The scatter plots show the difference between the predicted values and the actual values, as explained previously for the dummy datasets.

For the benzene dataset, the first two scatter plots in Figure 17 depict comparable results, implying that the data shares some common features and the selected regression model performs decently on the two components. However, the scatter plot for the third component of the transition dipole moment shows a different behavior, having the majority of the points far from the diagonal. This can be attributed to the hyperparameters used as well as the the likelihood that the Z component of the transition dipole moment does not match the chosen model. For the porphyrin dataset, Figure 18 helps highlight the poor performance of the model. As the model has a high error, few points are near the diagonal line. There is no notable difference between the three components of the transition dipole moment scatter plots. As the porphyrin dataset has only 151 samples, the model's ability to make meaningful predictions is limited.

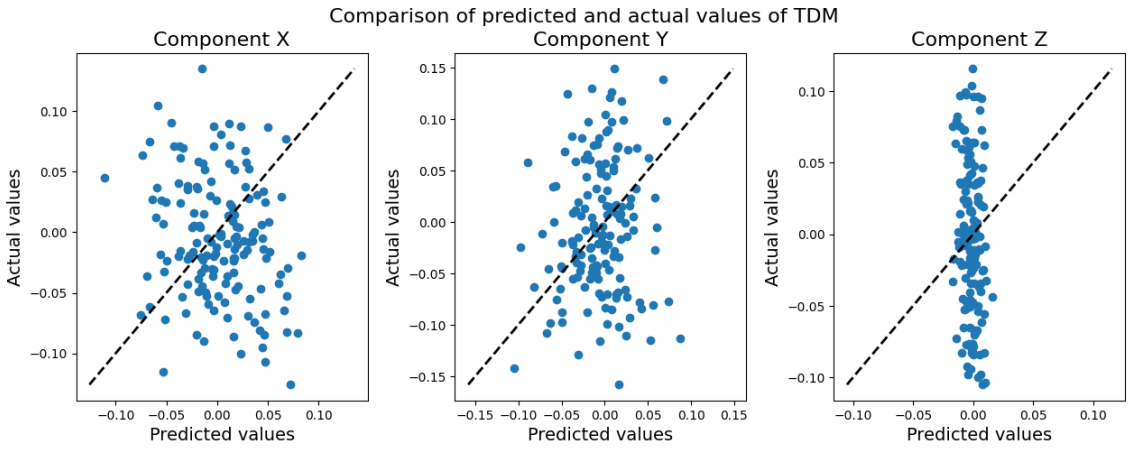


Figure 17: Scatter plot comparing the values of the benzene dataset output and the predicted values using Kernel Ridge Regression

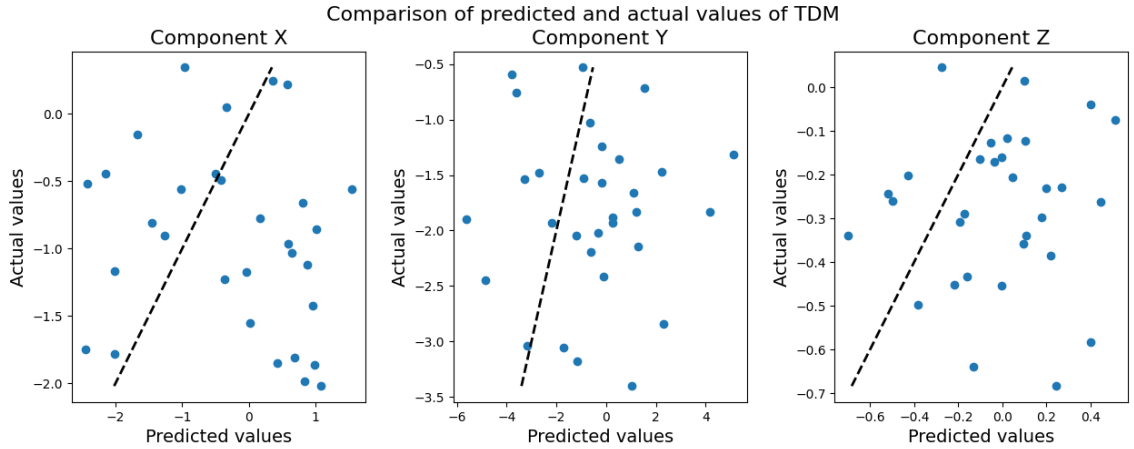


Figure 18: Scatter plot comparing the values of the porphyrin dataset output and the predicted values using Kernel Ridge Regression

4.2.2 Gaussian Process Regression Results

Figures 19 and 20 present the learning curves for Gaussian Process Regression on the benzene and porphyrin dataset. These figures demonstrate a remarkable similarity to their corresponding plots for Kernel Ridge Regression, shown in Figures 14 and 16, respectively. Similarly, the scatter plots 21 and 22 present almost identical results to their Kernel Ridge Regression counterparts, Figures 17 and 18. This can be attributed to the fact that both GPR and KRR belong to the class of kernel-based models and utilise the same hyperparameters. Therefore, it is reasonable to expect that the two models produce similar results.

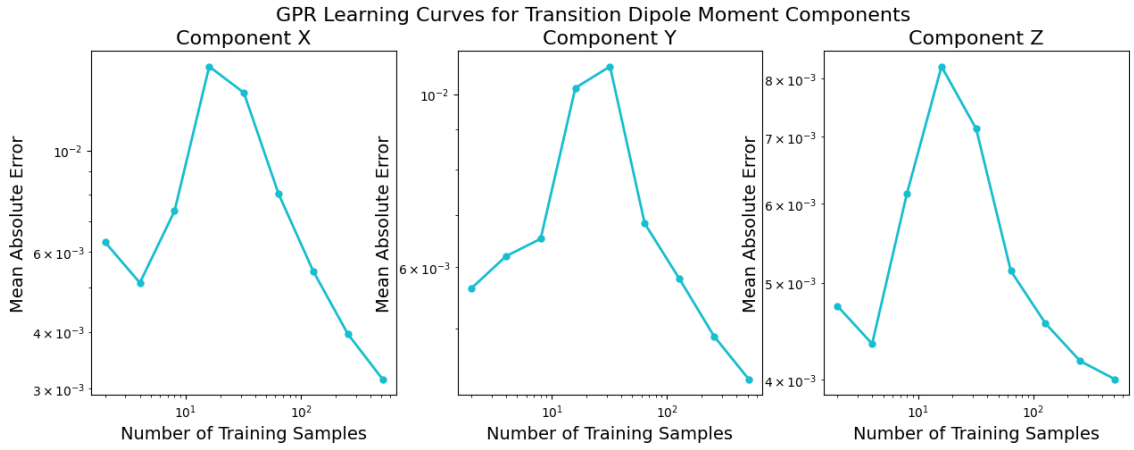


Figure 19: Learning Curves for the three components of the transition dipole moment of the benzene dataset using Gaussian Process Regression, log scaled

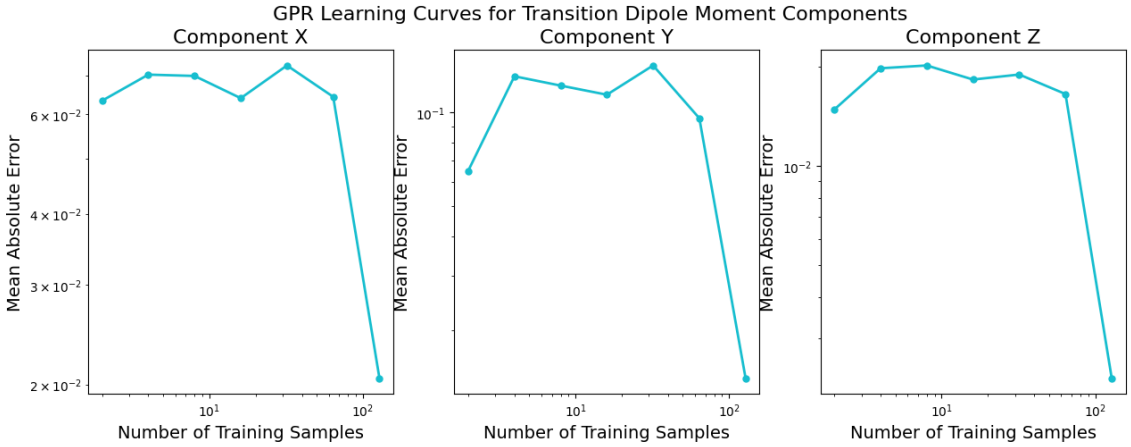


Figure 20: Learning Curves for the three components of the transition dipole moment of the porphyrin dataset using Gaussian Process Regression, log scaled

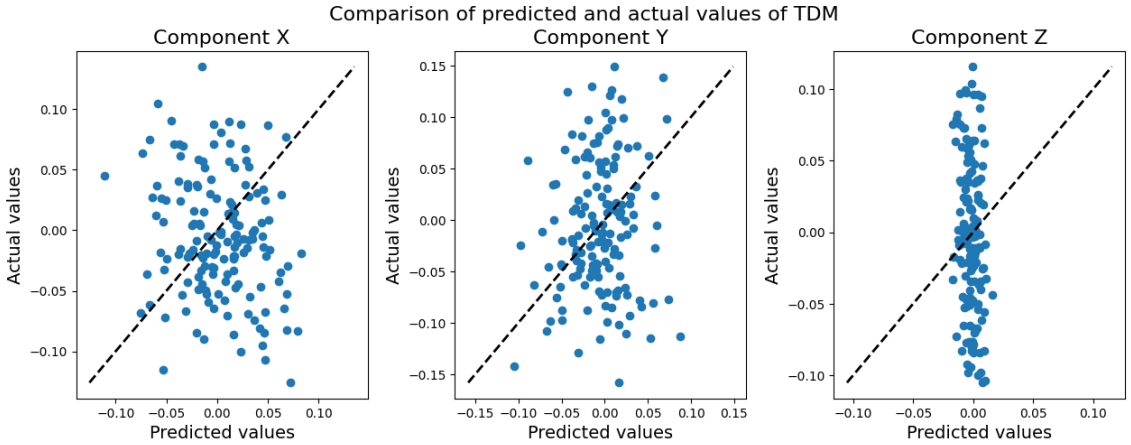


Figure 21: Scatter plot comparing the values of the benzene dataset output and the predicted values using Gaussian Process Regression

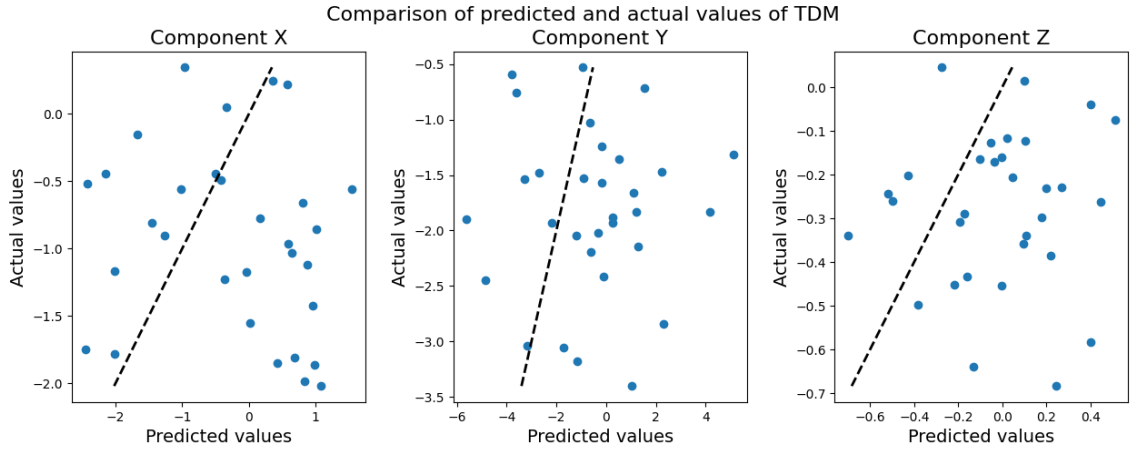


Figure 22: Scatter plot comparing the values of the porphyrin dataset output and the predicted values using Gaussian Process Regression

4.2.3 Multitask Gaussian Process Regression Results

The final section of the analysis chapter presents the performance of Multitask Gaussian Process Regression on the two molecular datasets. Figure 23 presents the learning curve for the benzene dataset. Compared to the two algorithms presented above, the learning curve for MT-GPR does not include a pre-asymptotic range, in which the error increases. The small sample size of the initial training sizes does not appear to have a significant effect when predicting all three components of the transition dipole moment at once. This claim is supported by the fact that the mean absolute error decreases only slightly even when training with the largest training size allowed. As previously mentioned, the MT-GPR outputs different predictions with each code run due to its structure. However, we found these predictions to be similar across multiple function calls.

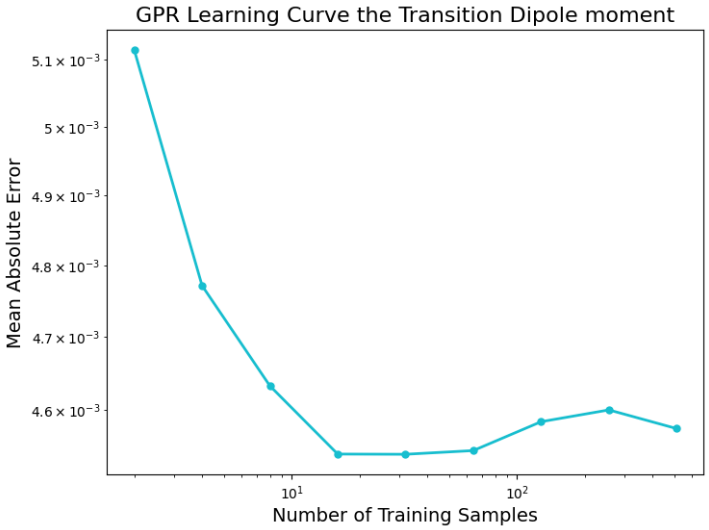


Figure 23: Learning Curves for the three components of the transition dipole moment of the benzene dataset using Multitask Gaussian Process Regression, log scaled

Figure 24 presents the learning curve of the MT-GPR algorithm applied on the porphyrin dataset. The learning curve, in contrast to the other molecular dataset, has a pre-asymptotic range. Because the initial training sizes include a limited number of samples, this behavior is to be expected. Furthermore, as a result of the small sample space, the learning curves show greater variation across different function calls than in the benzene dataset. The Figure also indicates that the algorithm performs worse than KRR and GPR for the porphyrin dataset, as the mean absolute errors observed are higher than in Figure 18 and Figure 22.

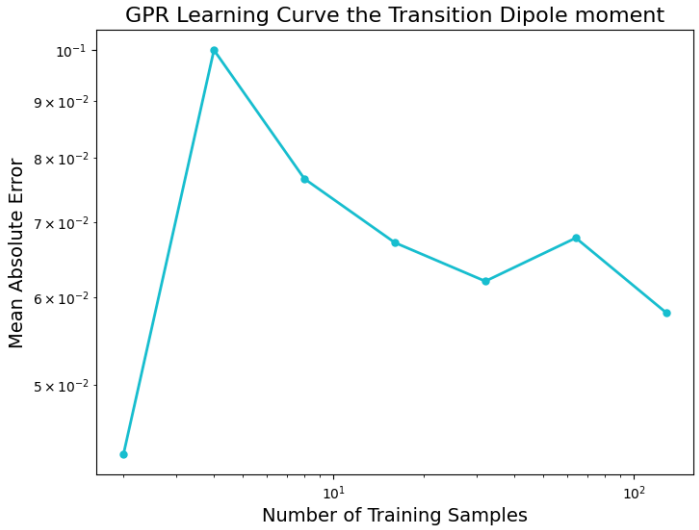


Figure 24: Learning Curves for the three components of the transition dipole moment of the porphyrin dataset using Multitask Gaussian Process Regression, log scaled

To further visualise the predictions of the MT-GPR on the molecular data, Figures 25 and 26 illustrate the scatter plots for the benzene dataset and the porphyrin dataset, respectively. In both Figures, it can be observed that a large number of samples deviate radically from the diagonal line. This indicates the algorithm performs poorly at predicting the data, having a high general error. This effect is especially pronounced in Figure 26, where very few points lay in proximity of the diagonal for all three components of the transition dipole moment.

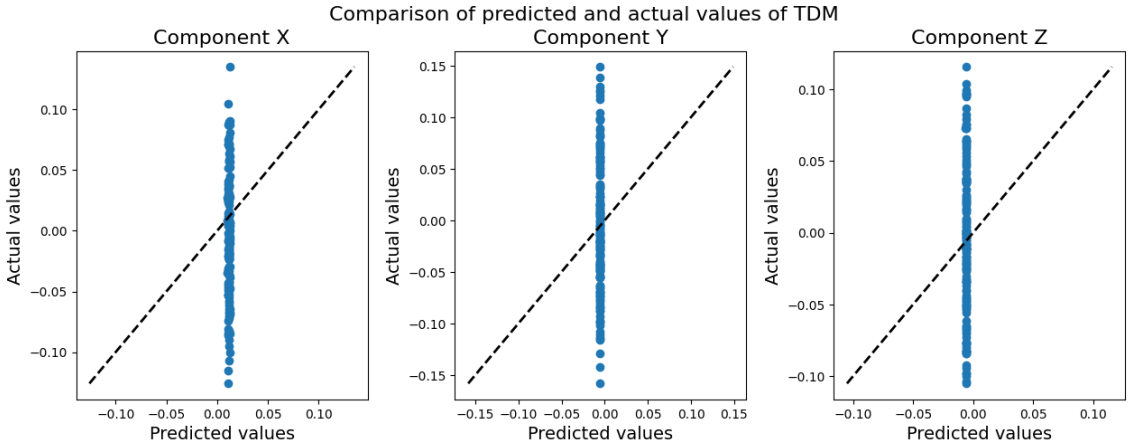


Figure 25: Scatter plot comparing the values of the benzene dataset output and the predicted values using Multitask Gaussian Process Regression

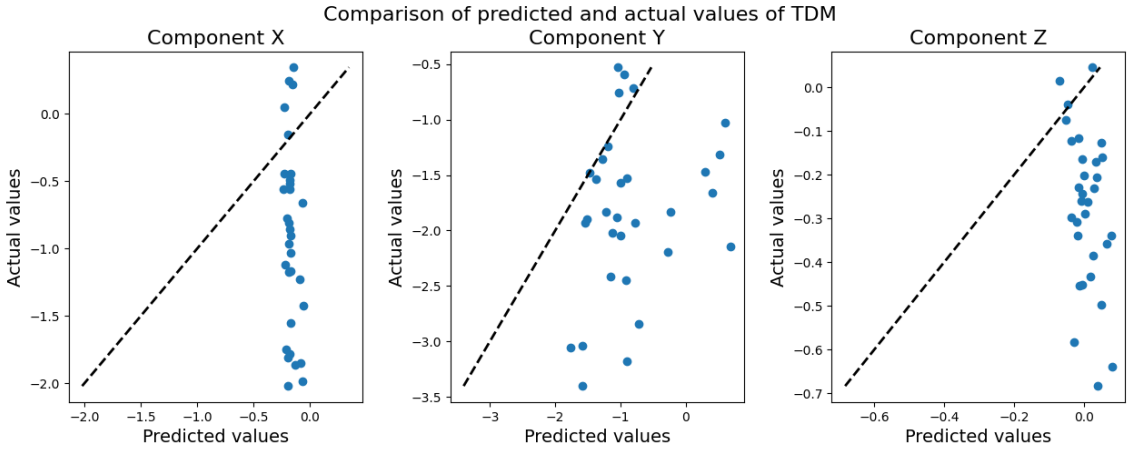


Figure 26: Scatter plot comparing the values of the porphyrin dataset output and the predicted values using Multitask Gaussian Process Regression

The Figures presented above show that Multitask Gaussian Process Regression does not have a better performance compared to the single task algorithms, namely Kernel Ridge Regression and Gaussian Process Regression. Interestingly, the opposite effect is highlighted, as MT-GPR displays inferior predictions for transition dipole moments across all the datasets used in this thesis. This outcome could potentially be attributed to various factors such as the limited sample sizes, the chosen hyperparameters, or the structure of the molecular trajectories.

5 Conclusions

The primary objective of the thesis was to observe whether multitask models outperform single task machine learning models at predicting the transition dipole moments associated with molecular trajectories. With this goal in mind, we used two methods for single task regression, namely Kernel Ridge Regression and Gaussian Process Regression. For the multitask model, we chose the Multitask Gaussian Process Regression, as it is also a kernel based model. To optimize the models, we used hyperparameter tuning, in which we found the best kernel width for the Gaussian kernel used throughout all three algorithms. To further test the models, we created two dummy datasets using random data as input and a linear combination of simple functions as output. One of the two datasets contained a rotational element in the output data to better mimic molecular data, especially transition dipole moment data.

As the main datasets, we used a trajectory of benzene molecules and a trajectory of porphyrin molecules as input and their respective transition dipole moments as outputs. We used Coulomb matrices to encode the molecular input data. The two datasets had 768 observations and 151 observations, respectively. To assess the performance of the models, we used two methods: learning curves and scatter plots. The learning curves provided an overview of the mean absolute error of models for increasing training sizes, while the scatter plots highlighted the differences between predicted values and the validation set.

Upon careful analysis of the three algorithms applied on all the datasets, we draw the conclusion that multitask machine learning models do not have a better performance for predicting transition dipole moments compared to single task algorithms. The opposite effect was seen, as Multitask Gaussian Process Regression performed worse at predicting molecular data than Kernel Ridge Regression or Gaussian Process Regression.

The results presented above deviate from the scenarios often presented in literature [22]. This observation suggests that in order to better fit the data, it may be necessary to investigate alternate kernels or entirely different machine learning models. Finding the best strategy for modeling the data can be difficult as the prediction of transition dipole moment data is a recent area of study. This thesis focused only on kernel-based models, a limited set of hyperparameters and moderately sized datasets.

References

[1] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization”. In: *Journal of Machine Learning Research* 13.Feb (2012), pp. 281–305.

[2] Edwin V Bonilla, Kian Chai, and Christopher Williams. “Multi-task Gaussian process prediction”. In: *Advances in neural information processing systems* 20 (2007).

[3] Erica Briscoe and Jacob Feldman. “Conceptual complexity and the bias/variance tradeoff”. In: *Cognition* 118.1 (2011), pp. 2–16. ISSN: 0010-0277. DOI: https://doi.org/10.1016/j.cognition.2010.10.004. URL: https://www.sciencedirect.com/science/article/pii/S0010027710002295.

[4] Rich Caruana. “Multitask Learning”. In: *Machine Learning* 28.1 (July 1997), pp. 41–75. ISSN: 1573-0565. DOI: 10.1023/A:1007379606734. URL: https://doi.org/10.1023/A:1007379606734.

[5] Anders. S. Christensen et al. *QML: A Python Toolkit for Quantum Machine Learning*. https://github.com/qmlcode/qml. 2017.

[6] Volker L. Deringer. “Gaussian Process Regression for Materials and Molecules”. In: *Chemical Reviews* 121.16 (2021). PMID: 34398616, pp. 10073–10141. DOI: 10.1021/acs.chemrev.1c00022. eprint: https://doi.org/10.1021/acs.chemrev.1c00022. URL: https://doi.org/10.1021/acs.chemrev.1c00022.

[7] Hieu A. Doan et al. “Quantum Chemistry-Informed Active Learning to Accelerate the Design and Discovery of Sustainable Energy Storage Materials”. In: *Chemistry of Materials* 32.15 (2020), pp. 6338–6346. DOI: 10.1021/acs.chemmater.0c00768. eprint: https://doi.org/10.1021/acs.chemmater.0c00768. URL: https://doi.org/10.1021/acs.chemmater.0c00768.

[8] Robert Dürichen et al. “Multi-task Gaussian process models for biomedical applications”. In: *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*. 2014, pp. 492–495. DOI: 10.1109/BHI.2014.6864410.

[9] Jacob R Gardner et al. “GPYtorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration”. In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 7575–7585.

[10] Martin Hofmann. “Support Vector Machines — Kernels and the Kernel Trick An elaboration for the Hauptseminar “ Reading Club : Support Vector Machines ””. In: 2006.

[11] Thomas Hofmann, Bernhard Sch, and Alexander Smola. “A Review of Kernel Methods in Machine Learning”. In: (Jan. 2006).

[12] Augusto F. Oliveira et al. “Density-functional based tight-binding: an approximate DFT method”. In: *Journal of the Brazilian Chemical Society* 20.J. Braz. Chem. Soc., 2009 20(7) (2009), pp. 1193–1205. ISSN: 0103-5053. DOI: 10.1590/S0103-50532009000700002. URL: https://doi.org/10.1590/S0103-50532009000700002.

[13] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[14] Florbela Pereira and João Aires-de-Sousa. “Machine learning for the prediction of molecular dipole moments obtained by density functional theory”. In: *Journal of Cheminformatics* 10 (Aug. 2018). DOI: 10.1186/s13321-018-0296-5.

[15] Philipp M. Pflüger and Frank Glorius. “Molecular Machine Learning: The Future of Synthetic Chemistry?” In: *Angewandte Chemie International Edition* 59.43 (2020), pp. 18860–18865. DOI: https://doi.org/10.1002/anie.202008366. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.202008366. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.202008366.

[16] Shampa Raghunathan and U. Deva Priyakumar. “Molecular representations for machine learning applications in chemistry”. In: *International Journal of Quantum Chemistry* 122.7 (2022), e26870. DOI: https://doi.org/10.1002/qua.26870. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qua.26870. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.26870.

[17] Raghunathan Ramakrishnan et al. “Quantum chemistry structures and properties of 134 kilo molecules”. In: *Scientific Data* 1.1 (Aug. 2014), p. 140022. ISSN: 2052-4463. DOI: 10.1038/sdata.2014.22. URL: https://doi.org/10.1038/sdata.2014.22.

[18] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006, pp. I–XVIII, 1–248. ISBN: 026218253X.

[19] Andrew S. Rosen et al. “Machine learning the quantum-chemical properties of metal frameworks for accelerated materials discovery”. In: *Matter* 4.5 (2021). ISSN: 2590-2385. DOI: https://doi.org/10.1016/j.matt.2021.02.015. URL: https://www.sciencedirect.com/science/article/pii/S2590238521000709.

[20] Matthias Rupp. “Machine learning for quantum mechanics in a nutshell”. In: *International Journal of Quantum Chemistry* 115.16 (2015), pp. 1058–1073. DOI: https://doi.org/10.1002/qua.24954. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qua.24954. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.24954.

[21] Sergey Sosnin et al. “A Survey of Multi-task Learning Methods in Chemoinformatics”. In: *Molecular Informatics* 38.4 (2019), p. 1800108. DOI: https://doi.org/10.1002/minf.201800108. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/minf.201800108. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/minf.201800108.

[22] Tom Viering and Marco Loog. “The Shape of Learning Curves: A Review”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–20. DOI: 10.1109/TPAMI.2022.3220744.

[23] Vladimir Vovk. “Kernel Ridge Regression”. In: *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*. Ed. by Bernhard Schölkopf, Zhiyuan Luo, and Vladimir Vovk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 105–116. ISBN: 978-3-642-41136-6. DOI: 10.1007/978-3-642-41136-6_11. URL: https://doi.org/10.1007/978-3-642-41136-6_11.

[24] Jie Wang. *An Intuitive Tutorial to Gaussian Processes Regression*. 2020. DOI: 10.48550/ARXIV.2009.10862.

[25] Julia Westermayr et al. “Neural networks and kernel ridge regression for excited states dynamics of CH2NH: From single-state to multi-state representations and multi-property machine learning models”. In: *Machine Learning: Science and Technology* 1.2 (May 2020), p. 025009. DOI: 10.1088/2632-2153/ab88d0.