Passive and model-agnostic sampling for training data development in machine learning regression

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Paolo Climaco

aus

Rom, Italien

Bonn, Januar 2025

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter/Betreuer: Prof. Dr. Jochen Garcke
Gutachter: Prof. Dr. Martin Rumpf

Tag der Promotion: 24.03.2025
Erscheinungsjahr: 2025

Abstract

Machine learning (ML) regression has a tremendous impact on advancing scientific progress. ML regression models predict continuous label values based on input features by leveraging large labeled datasets to learn the underlying ruling mechanisms. However, using large datasets may not always be feasible due to computational limitations and high data labeling costs. Such issues often arise in scientific applications, where we can typically only label a limited number of points for training due to expensive numerical simulations or laboratory experiments. The prediction quality of regression models is highly dependent on the training data. Consequently, selecting appropriate training sets is essential to ensure accurate predictions.

This work shows we can improve the prediction performance of ML regression models by selecting suitable training sets. We focus on passive and model-agnostic sampling, that is, selection approaches that solely rely on the data feature representations, do not consider any active learning procedure, and do not assume any specific structure for the regression model. This approach promotes the reusability of the labeled samples, ensuring labeling efforts are not wasted on subsets useful only for specific models or tasks.

First, we aim to improve the robustness of the models by minimizing their maximum prediction error. We study Farthest Point Sampling (FPS), an existing selection approach that aims to minimize the fill distance of the selected set. We derive an upper bound for the maximum expected prediction error of Lipschitz continuous regression models that linearly depends on the training set fill distance. Empirically, we demonstrate that minimizing the training set fill distance by sampling with FPS, thereby minimizing our derived bound, significantly reduces maximum prediction error and improves prediction stability in Gaussian kernel regression, outperforming alternative sampling methods.

Next, we focus on improving average prediction performances. We derive an upper bound for the expected prediction error of Lipschitz continuous models that depends linearly on a weighted fill distance of the training set. We propose Density-Aware FPS (DA-FPS), a novel data selection approach. We prove that DA-FPS provides suboptimal minimizers for a data-driven estimation of the weighted fill distance, thereby attempting to minimize our derived bound. We empirically show that using DA-FPS decreases the average absolute prediction error compared to other sampling strategies.

Our experiments focus on molecular property prediction, a crucial application for drug discovery and material design, which originally motivated our research effort. Traditional methods for computing molecular properties are slow. Using ML regression allows for quick predictions, accelerating the exploration of chemical space and the discovery of new drugs and materials. We empirically validate our findings with four distinct regression models and various datasets.

Acknowledgements

I want to thank Prof. Dr. Jochen Garcke for supervising the work that led to this thesis. I worked under Jochen's supervision for more than five years. I am deeply grateful for that. He has significantly contributed to my professional development. He enabled me to explore diverse scientific applications. Together, we tackled mathematical problems from various fields, including engineering, quantum chemistry, and nonlinear dynamical systems. That was a fun and formative experience. He helped me interact with multiple scientific communities from different research fields. This allowed me to broaden my perspective and understand the importance of looking at a given problem from different angles. I feel lucky I had Jochen as my supervisor, and I am thankful for our continuous interaction over the past years.

I thank my family - my sister Cecilia and my parents, Patrizia and Mauro - for their constant support, their sacrifices, that allowed me to move to Bonn in 2018, and their unconditional love. I thank Zia Rosi, for reaching out to me with creativity and love every single day.

I thank my colleagues Hannes, Inga, Mikhail and Tim for creating such a friendly environment. I will always remember with joy our conversations and the time we spent together, inside and outside the institute.

I want to express my deepest gratitude to my dear friend Jona. We met in Bonn at the beginning of our Masters, and since then, we have built up an unforgettable friendship, experience after experience. For many years in Bonn, while I was far away from my family, I was fortunate to have my dear friend nearby.

Contents

Notation and Acronyms

Notation

Data

$\mathcal{X} \subset \mathbb{R}^d$	Subset defining the data feature space
$\mathcal{Y} \subset \mathbb{R}$	Subset defining the data label space
$y_i \subset \mathcal{Y}$	Label value associated with the i-th data point
$\boldsymbol{x}_i \subset \mathcal{X}$	Data feature associated with the i-th data point
$\mathcal{D} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$	Labelled data set of available points
$\mathcal{D}_\mathcal{X} := \{\boldsymbol{x}_i\}_{i=1}^n$	Set of data features associated with set $\mathcal{D}$
$\mathcal{D}_\mathcal{Y} := \{y_i\}_{i=1}^n$	Set of data labels associated with set $\mathcal{D}$
$\mathcal{L} \subset \mathcal{D}$	Labelled training set
$\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$	Set of data features associated with set $\mathcal{L}$
$\mathcal{L}_\mathcal{Y} \subset \mathcal{D}_\mathcal{Y}$	Set of data labels associated with set $\mathcal{L}$
$\mathcal{U} := \mathcal{D} - \mathcal{L}$	Set of points in $\mathcal{D}$ with unknown labels
$S \subset \mathcal{D}_\mathcal{X}$	Set selected from $\mathcal{D}_\mathcal{X}$
$S_i \subset \mathcal{D}_\mathcal{X}$	Set selected after i iterations of a greedy algorithm
$2^{\mathcal{D}_\mathcal{X}}$	Collection of all possible subsets of $\mathcal{D}_\mathcal{X}$
$\mathcal{L}_\mathcal{X}^{FPS} \subset \mathcal{D}_\mathcal{X}$	Set selected with FPS
$\boldsymbol{m}_i \in \mathcal{D}_\mathcal{X}$	i-th medoid selected with k-medoids
$\boldsymbol{u}_j \in \mathbb{R}^d$	j-th centroid selected with k-means
$\mathcal{C}_j \subset \mathcal{D}_\mathcal{X}$	Data points cluster associated with the j-th centroid (medoid) selected with k-means (k-medoids)

Data distribution

$\boldsymbol{X}$	Data features' random variable taking value in $\mathcal{X}$
Y	Data labels' random variable taking value $\mathcal{Y}$
$\mathcal{P}$	Space of probability distributions on $\mathcal{X} \times \mathcal{Y}$
$p_\mathcal{D} \in \mathcal{P}$	Joint distribution of random variable $\boldsymbol{X}$ and Y, also called data source distribution
$p_\mathcal{L} \in \mathcal{P}$	Joint distribution of random variable $\boldsymbol{X}$ and Y, also called training data distribution
$p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) := \int_\mathcal{Y} p_\mathcal{L}(\boldsymbol{x}, y)dy$	Marginal of the training data distribution
$p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) := \int_\mathcal{Y} p_\mathcal{D}(\boldsymbol{x}, y)dy$	Marginal of the data source distribution
$\hat{p}^k_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) \in \mathbb{R}^+$	k-nearest neighbors data-driven density estimation of $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$

$\hat{p}^k_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) \in \mathbb{R}^+$	k-nearest neighbors data-driven density estimation of $p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})$

Data quantities

$\epsilon \in \mathbb{R}^+$	Labels' uncertainty
$\lambda_p \in \mathbb{R}^+$	Lipschitz constant of data regularity
$\epsilon_\mathcal{L} \in \mathbb{R}^+$	Maximum prediction error on training set $\mathcal{L}$
$\lambda_{l_\mathcal{X}} \in \mathbb{R}^+$	Lipschitz constant of error function w.r.t the first argument
$\lambda_{l_\mathcal{Y}} \in \mathbb{R}^+$	Lipschitz constant of the error function w.r.t the second argument
$s_{\mathcal{L}_\mathcal{X}} \in \mathbb{R}^+$	Separation distance of the points in $\mathcal{L}_\mathcal{X}$
$h_{\mathcal{L}_\mathcal{X},\mathcal{D}_\mathcal{X}} \in \mathbb{R}^+$	Fill distance of $\mathcal{L}_\mathcal{X}$ in $\mathcal{D}_\mathcal{X}$
$W_{\mathcal{L}_\mathcal{X},\mathcal{X}}\left(p_{\mathcal{X}_\mathcal{L}} \| p_{\mathcal{X}_\mathcal{D}}\right) \in \mathbb{R}^+$	Weighted fill distance of $\mathcal{L}_\mathcal{X}$ in $\mathcal{X}$ related to $p_\mathcal{L}$ and $p_\mathcal{D}$
$W^k_{\mathcal{L}_\mathcal{X},\mathcal{D}_\mathcal{X}} \in \mathbb{R}^+$	Estimated weighted fill distance of $\mathcal{L}_\mathcal{X}$ in $\mathcal{D}_\mathcal{X}$
$\rho_k(\boldsymbol{x}) \in \mathbb{R}^+$	Distance between $\boldsymbol{x} \in \mathcal{X}$ and its k-th nearest neighbor in $\mathcal{D}_\mathcal{X}$
$r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x}) \in \mathbb{R}^+$	Minimum between distance from $\boldsymbol{x} \in \mathcal{X}$ and its closest element in $\mathcal{L}_\mathcal{X}$ and $\rho_k(\boldsymbol{x})$
$\mathcal{N}_k(\boldsymbol{x}) \subset \mathcal{D}_\mathcal{X}$	Set of data points in the k-neighborhood of $\boldsymbol{x} \in \mathcal{X}$
$\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x}) \in \mathbb{N}$	Weight value associated with $\boldsymbol{x} \in \mathcal{X}$ used to define $W^k_{\mathcal{L}_\mathcal{X},\mathcal{D}_\mathcal{X}}$
$f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}^+$	Utility set function
$u \in \mathbb{N}$	Hyperparameter defining number of point to initially select with FPS in DA-FPS algorithm

Learning

$\mathcal{M} := \{g : \mathcal{X} \to \mathcal{Y}\}$	Function space defining solution of the regression problems
$l : \mathcal{X} \times \mathcal{Y} \times \mathcal{M} \to \mathbb{R}^+$	Error function
$m_\mathcal{L} : \mathcal{X} \to \mathcal{Y}$	Function learned training a regression model on set $\mathcal{L}$
$\boldsymbol{K}_\mathcal{L} \in \mathbb{R}^{b,b}$	Kernel matrix built on training set $\mathcal{L}$
$\lambda \in \mathbb{R}^+$	Kernel regularization parameter in KRR optimization problem
$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_b]^T \in \mathbb{R}^b$	Learning weights of the KRR
$\boldsymbol{\alpha}_\mathcal{L} \in \mathbb{R}^b$	Analytic solution to the KRR weights optimization problem on the labeled training set $\mathcal{L}$
$m_{\mathcal{L},\boldsymbol{\alpha}}(\boldsymbol{x}) \in \mathbb{R}$	Label associated with $\boldsymbol{x} \in \mathcal{X}$ predicted with kernel $\boldsymbol{K}_\mathcal{L}$ and weights $\boldsymbol{\alpha}$
$\boldsymbol{I}_b \in \mathbb{R}^{b,b}$	Identity matrix of size b.

$\gamma \in \mathbb{R}^+$	Width of the Gaussian kernel
$y(\boldsymbol{x}) \in \mathbb{R}$	Predicted label associated with $\boldsymbol{x} \in \mathcal{X}$ using an unspecified regression model

Chemistry

C	Carbon atom
F	Fluorine atom
H	Hydrogen atom
O	Oxygen atom
N	Nitrogen atom
S	Sulfur atom
P	Phosphorus atom
$\mathring{A}$	Anstrong, which is 10^{-10} metre
a.u.	Atomic units
mol	Mole, the international unit for amount of substance
kcal/mol	Kilocalories per mole, unit of energy
$n_a \in \mathbb{N}$	Number of atoms in a molecule
$\boldsymbol{r}_i = [r_{i,1}, r_{i,2}, r_{i,3}] \in \mathbb{R}^3$	Location of the i-th atom of a given molecule in the 3-dimensional space
$\boldsymbol{F}_i \in \mathbb{R}^{3n_a}$	Vector of per-atom forces acting on the i-th molecule with n_a atoms

Mathematical Symbols

$\epsilon_{\mathcal{X}}$	Positive scalar value, arbitrary small
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	n-dimensional real vector space
$\mathbb{R}^+$	Set of positive real numbers, $[0, \infty)$
$\mathbb{N}$	Set of natural numbers (positive integers and value 0)
$\mathbb{N}^+$	Set of natural numbers excluding the value 0
L_p	Space of functions for which the p-th power of the absolute value is integrable
$\lVert \cdot \rVert_p$	p-norm
$\lVert \cdot \rVert_\infty$	Infinity norm (maximum norm)

Acronyms

AI	Artificial intelligence
CM	Coulomb Matrix
CPU	Central Processing Unit
DA-FPS	Density-Aware Farthest Point Sampling

DCAI	Data-Centric AI
DFT	Density Functional Theory
FNN	Feed-Forward Neural Networks
FPS	Farthest Point Sampling
GDML	Gradient-Domain Machine Learning
KRR	Kernel Ridge Regression (with Gaussian kernel unless otherwise specified)
kNN	k-nearest neighbors
MAE	Mean Absolute Error
MAE_F	Mean Absolute Error (with vector-valued labels)
MAXAE	Maximum Absolute Error
MAXAE_F	Maximum Absolute Error (with vector-valued labels)
MAXMAE_F	Maximum Mean Absolute Error (with vector-valued labels)
ML	Machine Learning
PAM	Partitioning Around Medoids
RDM	Uniform Random Sampling
RMSE	Root Mean Square Error
SMILES	Simplified Molecular Input Line System

1. Introduction

Machine learning (ML) is a field of artificial intelligence (AI) that studies mathematical theories and numerical algorithms that enable computers to learn from data. ML systems implement algorithms that analyze data to identify underlying patterns and relationships, make predictions, or automate decision-making processes.

ML is becoming increasingly important because it can handle complex tasks that directly impact our daily lives, such as: translating languages [LCW+22, CBC+23], predicting weather [PSGA+24], and diagnosing diseases [RLJ20]. Moreover, ML is significantly enhancing our ability to gain knowledge in various scientific fields, speeding up the process of scientific breakthroughs [MFP+24].

A common application of machine learning is regression, which focuses on predicting continuous numerical values based on input features. For instance, ML regression models can predict the electrochemical properties of molecules from their geometric structure and chemical composition, helping to accelerate drug development [VCC+19]. Regression models learn the underlying prediction mechanism from the training data, for which both the input features and their corresponding output values (the labels) are already known.

The conventional research approach to ML regression tends to focus on model development, that is, algorithmic improvements. We call such an approach model-centric. Thanks to algorithmic breakthroughs and increases in computational power, the time required to build useful and effective regression models on a given benchmark task and dataset has shrunk dramatically. In the past, it would have taken decades of research. Now, it is often a matter of just a few years [KBN+21].

Unfortunately, model-centric ML research neglects the foundational role of data. Without enough data to learn from, ML is ineffective. If data is biased or corrupted with false information, ML regression models may provide biased or wrong predictions. In other words, data is crucial for developing and deploying effective regression models. Thus, data development procedures such as acquisition, labeling, and preprocessing are essential aspects to consider in ML regression. They play a crucial role in ensuring the quality and reliability of the data and, therefore, of the regression models that learn from it.

Recent advancements in algorithms have made it possible to develop effective models more quickly, emphasizing the importance of data-related issues. As a result, a portion of the scientific ML community is shifting its research focus from being model-centric to data-centric. Because of this, data-centric artificial intelligence (DCAI) is becoming an increasingly important area of research in ML and AI [ZBL+25].

This work addresses challenges related to the limited availability of labeled training data, often resulting from computational constraints and high labeling costs. These challenges frequently occur in scientific applications, where labeling typically relies on

expensive numerical simulations or laboratory experiments. We focus on molecular property prediction, a critical task in drug discovery and material design. Identifying molecules with desirable properties is essential for developing new drugs and materials. However, the space of possible molecules is large, with more than 10^{60} molecules according to some estimates [KE04], and classical numerical methods for predicting molecular properties are computationally intensive and time-consuming, limiting the pace of research. Consequently, in applications involving molecular property prediction, it is common to have access to large pools of unlabeled data, that is, molecules for which chemical and physical properties (the labels) are unknown and costly to obtain. Machine learning (ML) regression models offer an efficient and cost-effective solution for this challenge. By labeling a small subset of the data using traditional methods, we can train a regression model to predict the properties of the remaining unlabeled molecules quickly. This approach enables the fast exploration of the chemical compound space, accelerating the discovery of promising molecules [vLMT20].

The performance of ML regression models strongly depends on the quality of the training data used for learning the parameters. With a limited labeling budget, it is essential to select training sets that maximize model performance. The unlabeled data pool may contain hundreds of thousands or even millions of molecules, and the budget might allow labeling a few thousand of them for training. Manually evaluating such a large dataset to identify a suitable training set may be very time-consuming or not even feasible. It is more effective to develop algorithms that automate the data selection process by implementing a data selection strategy based on predefined principles. The labeling budget plays a critical role in shaping these selection strategies, as it determines how many data points can be labeled for training and directly impacts the effectiveness of the implemented strategy.

This work studies a fundamental question: how can we efficiently choose training data to improve the performance of regression models, especially when labeling resources are limited? In particular, we are interested in label-agnostic, passive, and model-agnostic strategies to select training data that can make ML regression models more robust, stable, and accurate according to some evaluation metrics of interest.

We call label-agnostic those sampling approaches that select points based solely on their feature representations and do not have access to the data labels at the time of selection. We categorize label-agnostic data selection strategies as active and passive.

Active learning [Set12, RXC+21] iteratively selects training points to maximize the predictive performance of a given model or model class. It involves training one or several learning models, predicting uncertainties or estimating labels for unlabeled data, and using these to determine the relevance of data points for training purposes. The most relevant data points are selected for labeling, and the cycle starts anew until (qualitative) stopping criteria are fulfilled. Consequently, active learning benefits a specific model or model class and optimizes performance for a particular learning task, as it relies on the knowledge of the labels to update the parameters of the learning models employed during the selection process. In contrast, passive sampling relies solely on feature space locations, potentially benefiting multiple learning tasks, as it does not depend on label

values. Passive sampling was first introduced in [YK10], where the authors highlight its importance in scenarios where labeled data can be difficult, time-consuming, or expensive to obtain.

We think that passive sampling strategies can be classified as model-dependent and model-agnostic. Model-dependent approaches optimize the data selection process for specific models or classes. Assuming the knowledge of the learning model may lead to developing strategies that reflect some principle of optimality for selecting the training set, as for selection approaches in experimental design [JD75, YBT06]. However, such optimal selections aim to optimize the performances of specific models, similar to active learning. Contrarily, model-agnostic strategies have the potential to benefit multiple classes of regression models, thereby enhancing reusability of costly data labeling.

Our aim is to select training data from an unlabeled pool of available points to improve the prediction performance of regression models. We train the regression models on the selected data and use them to predict labels for the points in the pool not selected for training. We select the training sets relying solely on passive and model-agnostic sampling approaches. This leads us to two key questions: How can we quantify the suitability of a selected set for training an unknown regression model? And how can we systematically and efficiently select suitable training sets from a pool of unlabeled data?

To quantify the suitability of a selected set for training purposes, we use scalar-valued functions. These functions take a selected set as input and evaluate its quality based on the distribution of its points within the input space. By leveraging these scalar-valued functions, we derive theoretical bounds for the expected prediction error of a regression model trained on the selected set. The concept is straightforward: the smaller the scalar value associated with a training set, the tighter the bound on the model's prediction error.

Once we establish a method to measure the suitability of a set for training purposes, the next critical step is to devise a systematic and efficient approach for selecting suitable sets. To this end, we introduce and develop algorithmic procedures that identify subsets from a pool of unlabeled data according to specific principles. Furthermore, we provide theoretical insights demonstrating that these selection methods align with our derived theoretical results. Specifically, they select training data aiming at minimizing our derived bounds on the prediction error of ML regression models.

We have two primary objectives. The first is to determine an algorithmic procedure, supported by theoretical results, for selecting training sets that improve the robustness of regression models by minimizing their maximum prediction error. The second objective is to develop an algorithmic procedure, also supported by a solid theoretical motivation, for selecting training sets that improve the average performance of regression models by reducing their mean absolute prediction error.

Note that our work originates from addressing challenges related to molecular property prediction tasks. Consequently, the experimental sections, where we provide empirical validation of our theoretical results, mainly focus on such application scenario. Nonetheless, the theoretical results and algorithms we analyze and develop are independent of the specific application context.

1.1. Structure

In **Chapter 2** we provide an overview of data-centric AI (DCAI). We mention the core principles and methodologies that drive this paradigm shift from the model-centric perspective in AI research. We report various definitions of DCAI, providing a solid foundation for readers new to this area or seeking a clearer understanding of this research field. Moreover, we introduce and describe the key objectives and tasks in DCAI. In particular, we focus on training data development procedures, specifically data selection approaches. We delve into why data selection is critical for the training data development process, clarifying its role in building robust and effective machine learning models. We also provide a classification of data selection procedures.

In **Chapter 3** we focus on coresets, a specific class of data selection methods of interest in this work. Coresets are practical tools for performing label-agnostic, passive and model-agnostic sampling. We introduce and describe various coreset selection methods that serve as baselines in our numerical experiments. These methods include uniform sampling, cluster-based approaches, and greedy techniques.

In **Chapter 4** we provide a general overview of ML for molecular property prediction, which is the application focus of this work. Additionally, we provide details of all the tools we use in the experimental sections, including datasets, data preprocessing procedures, regression models, evaluation metrics and additional info on open-source coding libraries.

In **Chapter 5**, we introduce the first optimization problem we aim to solve. The problem involves selecting a training set from a pool of unlabeled points to minimize the maximum expected prediction error of an unknown Lipschitz continuous regression model trained on the selected set. To address the problem we use the concept of fill distance (Definition 5.1). The fill distance is a scalar quantity we can associate with subsets of points in a given unlabeled data pool. We use this scalar to quantify the suitability of a set for training purposes. We derive an upper bound for the maximum expected prediction error that linearly depends on the training set fill distance (Theorem 5.1). Additionally, we analyze the Farthest Point Sampling (FPS) an existing greedy algorithm that we can use to select training sets by aiming to minimize their fill distance, thereby minimizing our derived bound. Furthermore, recapturing results from numerical mathematics, we theoretically show that selecting training sets with the FPS can also increase model stability for the specific case of Gaussian kernel regression approaches.

In **Chapter 6** we introduce the second optimization problem we aim to solve. The problem involves selecting a training set from a pool of unlabeled points to minimize the expected prediction error of a Lipschitz continuous regression model trained on the selected set. To address the problem, we introduce the concept of weighted fill distance (Definition 6.1). We define the weighted fill distance as a scalar quantity we can associate with finite sets of points in a given bounded space in $\mathbb{R}^d$, $d \in \mathbb{N}^+$. We introduce this scalar

as an alternative approach to the fill distance to quantify the suitability of a set for training purposes. We derive an upper bound for the expected prediction error of Lipschitz continuous regression models that is linearly dependent on the weighted fill distance of the training set (Theorem 6.1). Furthermore, we propose Density-Aware Farthest Point Sampling (DA-FPS), a novel data sampling approach. DA-FPS (Algorithm 6), aims to select training sets by minimizing the weighted fill distance. We show that DA-FPS provides suboptimal minimizers for a data-driven estimation of the weighted fill distance, thereby aiming to minimize our derived bound (Theorem 6.2 and Theorem 6.3).

In **Chapter 7** we provide experimental validation to the theoretical results from Chapter 5 and Chapter 6. In Section 7.1 we empirically show that selecting a training set by aiming to minimize the fill distance using FPS, significantly reduces the maximum prediction error of various regression models, outperforming alternative sampling approaches by a large margin (Figs. 7.1, 7.2 and 7.10). Furthermore, we show that selecting training sets with FPS can also increase model stability for the specific case of Gaussian kernel regression approaches (Fig. 7.3). In Section 7.2 we empirically show that selecting training sets with DA-FPS, thus attempting to minimize the weighted fill distance of the selected set, significantly decreases the average prediction error of Lipschitz continuous regression models (Figs. 7.11, 7.13 and 7.19). Moreover, we empirically investigate the benefits of combining FPS with other well-established passive and model-agnostic strategies. We show that augmenting such approaches with FPS during the initial steps of the sampling process consistently improves the average performance of the predictions (Fig. 7.12). We conduct the experiments with the regression models and datasets introduced and described in Chapter 4 and Appendix B.

In **Chapter 8** we provide a summary highlighting the key findings and contributions. Additionally, we reflect on the limitations of this work and suggest possible future research directions to overcome them.

We note that part of the work presented in this thesis, primarily in Chapter 5 and Section 7.1, has been published in [CG24].

2. Data-Centric AI (DCAI)

In this chapter we provide an overview of DCAI. We start by introducing key definitions, objectives and tasks. Next, we focus on training data development, which is one of the main branches of DCAI and particularly of interest in this work. We analyze the main aspects of data selection, explaining why data selection plays a key role in the development of training data. Furthermore, we provide a classification of data selection procedures.

2.1. The path to DCAI

Artificial intelligence (AI), and more specifically machine learning (ML), has a tremendous impact on a large variety of domains such as finance [HSK19], healthcare [MWW+17], chemistry [HvL21], agriculture [LBM+18] and many others. The rise of ML is due to three main factors: the increasing availability of large amounts of data, advancements in computing power (the hardware), and improvements in learning algorithms and models development (the software). The increase in computing power allows for complex calculations to be performed quickly and efficiently, while the advancements in algorithms enable ML systems to process and interpret data more accurately. The exponential increase in data production provides the fuel needed to train, refine and effectively deploy ML methodologies. Data play one of the three key roles in the development and deployment of ML. However, in the initial phase of ML development the dominant paradigm was to build better software technologies by developing more effective algorithms and models to increase performances on one or several tasks, while keeping the data mainly fixed. Unfortunately, such a model-centric approach, consistently relying on a fixed dataset for model development, may overlook several issues related to deploying the developed ML model into the real world. For instance, the cost and reliability of data development procedures, such as acquisition, labeling and preprocessing, may strongly affect the fidelity and relevance of the analyzed data to the underlying problem, possibly leading to a variety of issues that could interfere with the effectiveness of the model, such as limited amount of data, data biases and mislabeled data. Consequently, the ML research community is increasingly focusing on data-related issues. The need for better data practices can also be highlighted by the success of new initiatives within the industry landscape, such as Cleanlab, a start-up founded in 2021 that developed an automated data curation platform now used by several Fortune 500 Companies and that raised $25 million in funding in 2023 [Kon23]. Another example is Snorkle AI [RBE+19], which among other services develops systems for automatic data annotations without hand labeling. It started in 2015 as a research project and was valued at $1 billion in

2021 [Van21].

The rising interest and recognition of the value of data to improve ML applications has led to the creation of a new concept in the machine learning research community: Data-centric AI, coined by Prof. Andrew Ng in 2021. Since then, the ML research community is increasingly more enthusiastic and involved in data-centric AI. This is evidenced by various factors, including the NeurIPS 2023 Datasets and Benchmark track receiving approximately 1000 submissions − almost twice as many as the previous year. Additionally, there is a growing community effort to develop and promote data-centric AI competitions, such as the six Dataperf Benchmark challenges [MBY+23]. Furthermore, The Journal of Data-centric Machine Learning Research (DMLR) has been established to provide a reputable publication venue for high-quality articles focused on the data aspect of machine learning research.

2.2. Definitions of DCAI

Data-centric AI is a developing area of research that aims to enhance various data aspects of machine learning procedures. There have been several attempts to define it, with some authors defining it as a research field that "encompasses methods and tools to systematically characterize, audit, generate, and monitor the underlying data used to train and evaluate ML models" [SIS24]. Others have defined it as "a framework to develop, iterate and maintain data for AI systems" [ZBL+25]. While these definitions, and many others [PZ21, JMG23, ZLB+23], clarify that DCAI focuses on improving data development and maintenance procedures to enhance ML performances, they do not provide a clear explanation of the practices and procedures involved. To address this issue, [ZBL+25] proposed a first research framework for data-centric AI that defines a list of goals and associated tasks. While this framework is likely to be updated as the field evolves, it provides a clear outline for researchers and practitioners to navigate and understand data-centric AI. In the next section, we briefly summarize the framework they propose.

2.3. Goals and tasks of data-centric AI

Following along [ZBL+25] we frame data centric AI in three main areas or goals: training data development, inference data development, and data maintenance. Training data development aims to collect and produce high-quality data for machine learning models. It includes data collection [BFPK20], labeling [KMLE20], preparation [AFK+14, SJ19], reduction [LCW+17, MBL20], and augmentation [FAKA+18]. Inference data development aims to create new evaluation sets and benchmarks to test the models capabilities. This involves generating samples that adhere to or shift from the training data distribution, in order to assess or unlock various capabilities of the model [MDFF16, KSM+21]. Data maintenance aims to ensure the quality and reliability of data in a dynamic environment. It includes data understanding [BW13], quality assurance [SDD+18], and efficient storage

and retrieval [BJL19]. It plays a crucial role in ensuring that the data used in AI training and inference is accurate, up to date and easily retrievable.

In this work, we are interested in tasks related to training data development. Note that, training data development is crucial in both supervised and unsupervised machine learning. In supervised machine learning, training data is labeled, and the quality of the labels is essential for effective model training. Ensuring the data is representative, correctly labeled, and well-balanced across different categories is critical to achieve high model performance. In unsupervised machine learning, training data development involves careful preparation to ensure that meaningful patterns can be extracted. This includes selecting appropriate features, removing noise, normalizing data, and sometimes even engineering features to highlight important relationships. Although there are no explicit labels, the quality of the data still significantly affects the ability to discover useful clusters, associations, or embeddings.

In this work, we mainly focus on tasks related to training data development for supervised machine learning. Key tools of training data development procedures are data selection techniques. Unfortunately in [ZBL+25], data selection techniques are merely seen as a tool for data reduction. We think this is just one on the application of data selection approaches. Next, we show that selecting points or subsets from a previously collected data pool plays a key role in training data development and is used not only for data reduction, but it is also required for other tasks including data labeling and cleaning.

2.4. Importance of data selection in training data development for supervised ML

The first step in training data development is data collection. Data collection is the process of gathering and acquiring data with the goal of creating a pool of data points related to the underlying task. Data collection practices include discovery, integration, and data synthesis. Dataset discovery consists of finding related and useful datasets, data integration combines datasets from different sources into one and data synthesis consists of generating datasets, for instance, through numerical simulations. In some cases, it is better to create a dataset with desired patterns than to collect them from the real world.

At the end of the data collection process, we are provided with a pool of data points. In a supervised ML application context, either all the gathered points are used for training purposes, or only a portion of those is selected as the training set. Unfortunately, using all the gathered data for training may be unfeasible or negatively affect the learning process. If the collected points are already labeled, there could be mislabeled points polluting the dataset with false information that may affect the final performances of the trained model. Thus, there is a need to preprocess the data pool, selecting and excluding the mislabeled points. When the collected data is unlabeled, labeling every point may be too costly or time-consuming. Furthermore, training on the entire dataset might be prohibitive due to resource constraints. In such cases, a representative subset needs to be selected for data reduction. Note that different training subsets may lead to different

performances of the same learning algorithm. Therefore, selecting a suitable subset from the collected data pool is crucial to ensure good model performance.

Consequently, it is important to develop data selection strategies that aim to choose training sets that can enhance the performance of ML models while taking into account the available knowledge at the time of selection, and considering the costs of the selection process as well as the available budget.

Another important task to consider is data splitting. For instance, benchmark datasets used for model evaluation purposes are typically split into training, validation, and test sets. Different splits provide different representations of the same underlying data population. Therefore, selecting appropriate training, validation, and test sets is essential to evaluate models properly.

2.4.1. A simple classification of data selection procedures for supervised ML

We categorize data selection approaches as label-aware and label-agnostic. Label-aware approaches have access to the data points labels and use them to guide their selection strategy. Label-agnostic approaches select points based on their feature representation.

Label-aware selection approaches can be based on data valuation techniques or on geometrical consideration of the data points in the feature and label spaces. Data valuation approaches aim to associate each data point with a specific value, rating its quality according to some specific criteria. Data point valuation can be model-dependent or model-agnostic. Model-dependent valuation approaches analyze how data points affect the performances of a model or set of models, giving higher value to those points that are more likely to lead to a performance increase. For instance, in [GZ19], the authors propose using data Shapley values as a metric to measure the value of individual training data points in increasing the predicting performance of a specific supervised prediction approach. Model-agnostic data valuation approaches rate the quality of data points independently of an underlying learning algorithm. For example, [JKW$^+$23] develops a novel method to value individual data based on the sensitivity analysis of the Wasserstein distance, taking into account data labels and features. Label-aware selection approaches based on geometrical consideration of the data points aim to select subsets providing good representations of the feature and label space by making geometrical considerations related to the data positions. For instance, in [CHE$^+$21], the authors propose an approach aiming at maximizing Euclidean distances between data features and data labels to select subsets providing a better representation of the data space. Label-agnostic data selection procedures select points only considering the data feature representation. We distinguish between active and passive label-agnostic data selection. Active learning [Set12, RXC$^+$21] involves training one or several regression models, predicting uncertainties or estimating labels for unlabeled data, and use these to determine the relevance of data points for training purposes. The most relevant are selected for labeling and the cycle starts anew, until (qualitative) stopping criteria are fulfilled. It typically only benefits a specific model or model class and optimizes the performance of the models for a specific learning task, as it exploits the knowledge of the computed labels to iteratively update the parameters of the models during the selection process. Passive sampling [YK10] is based only

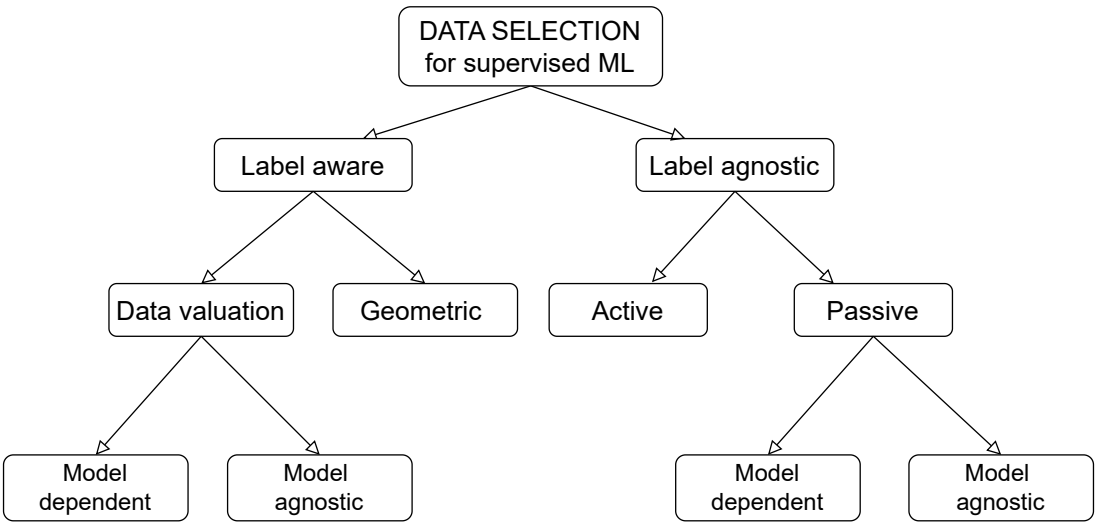


Figure 2.1.: Flowchart illustrating a simple classification of data selection approaches for supervised ML

on the feature representation of the data points, that is, their location in the feature space. Consequently, it has the potential to be beneficial for multiple learning tasks that pertain to the same data, as it is entirely independent of the label values associated with the analyzed data points. We think passive sampling can be further divided into two subclasses: model-dependent and model-agnostic. Model dependent passive sampling strategies are developed to benefit specific learning models or model classes, such as linear and kernel regression [YBT06], similar to active learning. Assuming the knowledge of the learning model may even lead to the development of strategies that reflect some principle of optimality for the selection of the training set, as in the case of approaches developed in the field of experimental design [JD75]. Contrarily, model-agnostic strategies have the potential to benefit multiple classes of regression models rather than just one. We note that passive sampling strategies are categorized as label-agnostic in our classification. Therefore, when we refer to passive sampling, we implicitly refer to approaches that do not consider data labels. In this work, we focus on passive and model-agnostic sampling. In the next chapter we provide an overview on "coresets" a class of approaches that can be effectively used for passive model-agnostic data selection and that we consider as baselines methods in the experimental sections of this work.

2.4.2. Data labeling budgets

We highlight that an important aspect to consider for the choice of the training data selection strategy is the data labeling budget, that is, the amount of resources we can invest in the labeling process, which determines the number of points we can label

for training purposes. Depending on the number of points that can be selected and labeled for training, the training data selection process can either focus on improving the robustness of the regression models, the average accuracy of their predictions or, ideally, both. Later, in Section 4.4 we see that robustness refers to the ability of a regression model to maintain low maximum prediction error over entire data space, while average accuracy measures how close the predictions of the model are to the true label values on average across the entire dataset.

In this work, we qualitatively refer to three main budget scenarios. Low, medium and large budget. The quantitative notions of low, medium and large data budget are context-dependent. They relate to various factors, including the costs of the labeling procedure, the size of the available pool and the distribution of the data locations. Nonetheless, we can consider qualitative, context-independent notions that provide an intuitive understanding of such data regimes.

We say we are in a low data budget regime when we can only label a small amount of points for training. In particular, in such a low data budget regime, it is not possible to adequately represent in the training set the variability and complexity of the distribution of the data points in the available pool, independently of how we choose the training points. One potential downside of having a low data budget is that a regression model trained on few labeled data points may perform very poorly in those regions of the data space that are not represented in the training set. This implies that there are portions of the data space where the predictions of the trained model are not reliable. One approach that can be implemented to increase the reliability of a regression model is to improve its robustness by considering a training data selection strategy that prioritizes performance improvement on the worst-case scenarios, that is, minimizing the maximum error. By aiming to improve robustness, we ensure that the performance of the model is consistently reliable across the considered data space, even if the average prediction quality might not be optimal. In Chapter 5 we study a data selection approach aiming at improving the robustness of the regression models.

We are in the medium data budget regime if we can label a moderate amount of data points for training, such that, depending on the selection process, the training set can potentially represent the entire data space and capture the distribution of the points in the available pool. Thus, depending on the selection strategy, each point in the data pool may have a close selected point and, simultaneously, the distribution of the data points in the selected training set may resemble that of the points in the data pool. Consequently, in the medium data budget scenario, a data selection approach can focus on optimizing robustness and improving average prediction quality. In Chapter 6 we propose one approach to do that, where we first include in the training set data points across the entire data space of interest. After that, we focus the sampling on portions of the data space with higher density to better represent in the training set those regions that are more populated and where it is necessary to optimize the performances of a model in order to improve the average quality of its predictions. The approach we propose initially aims at including in the training set data points across the entire data space to prevent very large prediction errors in the underrepresented regions. Note that, depending on

their magnitude, large prediction errors may have a significant impact on the average quality of the predictions. Thus, robustness of the predictions plays an important role in ensuring the improvement of average performances.

In the large data budget regime it is possible to select training sets capturing the full distribution of the analyzed dataset. That is, it is possible to properly represent all the regions in the data space of interest. In this work, we are not interested in such a labeling budget scenario, as we restrict our study to applications where labeling resources are limited, and training data selection has a higher impact on the performances of the resulting trained regression model.

3. Data Selection Via Coresets

Coresets [Fel19] is a class of data selection approaches that can be effectively employed for passive and model-agnostic sampling. Coresets can be used to identify (informative) subsets from a pool of unlabeled points, according to some specific principle and solely relying on the data points' locations. Following along [Fel19], we organize coresets into four main categories: uniform sampling, importance sampling, cluster-based methods, and greedy methods. Uniform sampling involves randomly selecting data points with equal probability. Importance sampling selects data points that are more relevant or important to the problem at hand. Cluster-based methods group similar points together and select representatives from each cluster. Greedy methods iteratively add the most influential point at each step, according to a given principle. In the following sections, we describe each of these approaches in more detail.

3.1. Uniform and importance sampling

Uniform sampling involves selecting samples uniformly at random across the domain of interest. This means that each point in the available data pool has an equal probability of being selected, independently of its location in the feature space or label value. While this sampling approach is the simplest coreset, it is also considered the natural benchmark for all the other sampling strategies [Fel19]. Formally, given a dataset $\mathcal{D}_{\mathcal{X}} := \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, $d \in \mathbb{N}$, consisting of data points locations, we have that the probability of selecting $\boldsymbol{x}_i \in \mathcal{D}_{\mathcal{X}}$ is $p(\boldsymbol{x}_i) := \frac{1}{n} \; \forall \; i = 1, \ldots, n$. The uniform sampling approach can result in biased results if the dataset is imbalanced or if some data points are more important than others. Importance sampling strategies aim to overcome this limitation, by assigning a weight $a_i \in \mathbb{R}$ to each data point $\boldsymbol{x}_i$ based on its relevance to the problem being addressed. Typically, the higher is the weight the more relevant the point. By assigning weights to the data points, we can deterministically select them based on a ranking system that prioritizes points with higher weights. Alternatively, we can select points in a randomized fashion by constructing a probability distribution $p_a(\boldsymbol{x}_i)$ depending on the weights, that is, $\sum_{i=1}^n p_a(\boldsymbol{x}_i) = 1$, $p_a(\boldsymbol{x}_i) \geq 0 \, \forall \, i = 1, \ldots, n$, and for each $i \neq j$ we have $p_a(\boldsymbol{x}_i) \geq p_a(\boldsymbol{x}_j) \iff a_i \geq a_j$. This leads to a non-uniform randomized selection that prioritizes more important data points. The importance weights $\{a_i\}_{i=1}^n$ can be defined through domain knowledge and statistical or geometrical arguments. Importance weights may define the importance of the data points with respect to the learning process of a specific model o with respect to a general model independent principle. For instance, in [XSML23] the authors propose an important sampling approach for selecting suitable pre-training datasets for large language models (LLM) based on weights

that are independent of the specific language model considered. Alternatively, [TMO23] provides an importance sampling selection method to improve data distillation processes and [KF18, JG18] propose importance sampling for accelerating the training of deep neural networks. Another example is the CUR approach [MD09], which selects data points based on an importance score determined to ensure that the most important points are those providing a best low-rank approximation of the data matrix. The CUR approach can be effectively used for data points and data features selection [CHE+21].

3.2. Cluster based methods

Clustering algorithms are essential tools in machine learning and data analysis for grouping similar data points into distinct clusters. A cluster can be defined as a group of data points that are more similar to each other than to those outside the group. The similarity criteria used to define a cluster can vary depending on the application, but typically in ML, it is based on a notion of distance between the data points locations. Thus, we can think of a cluster as a group of data points whose inter-point distances are small compared with the distances to points outside the cluster. In this work we consider the Euclidean distance. Cluster-based coresets segment the feature space in clusters and select representative points from each cluster to build a representative subset (a coreset). Well known and established methods within this class are as k-medoids and k-means that we now introduce.

3.2.1. k-means

The k-means [Mac67] partitions a given dataset $\mathcal{D}_{\mathcal{X}} := \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ into some number k of clusters and provides a coreset consisting of k data points, called centroids, which are representative of each cluster. For now, we shall suppose that the value of k is given. The goal of k-means is to determine clusters $\{\mathcal{C}_j\}_{j=1}^k$ of points in $\mathcal{D}_{\mathcal{X}}$ and centroids

Algorithm 1 k-means algorithm

b

Input: Data points $\mathcal{D}_{\mathcal{X}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and number of clusters k

Output: Cluster centroids $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k\}$

1: Initialize cluster centroids randomly: $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k$.
2: **repeat**
3: **for** $i = 1$ to n **do**
4: Assign each point $\boldsymbol{x}_i$ to the nearest centroid $\boldsymbol{u}_i^* = \arg\min_{1 \le j \le k} \{\|\boldsymbol{x}_i - \boldsymbol{u}_j\|_2^2\}$
5: **for** $j = 1$ to k **do**
6: Update cluster centroid $\boldsymbol{u}_j$ as the mean of all data points assigned to it:
7: $\boldsymbol{u}_j = \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{x} \in \mathcal{C}_j} \boldsymbol{x}$ where $\mathcal{C}_j = \{\boldsymbol{x}_i \in \mathcal{D}_{\mathcal{X}} | \boldsymbol{u}_i^* = \boldsymbol{u}_j\}$
8: **until** Convergence criterion is met.

$\{\boldsymbol{u}_j\}_{j=1}^k \subset \mathbb{R}^d$, solving the following optimization problem

$$\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_k \in \mathbb{R}^d} \sum_{j=1}^{k} \sum_{\boldsymbol{x} \in \mathcal{C}_j} \|\boldsymbol{x} - \boldsymbol{u}_j\|_2^2, \tag{3.1}$$

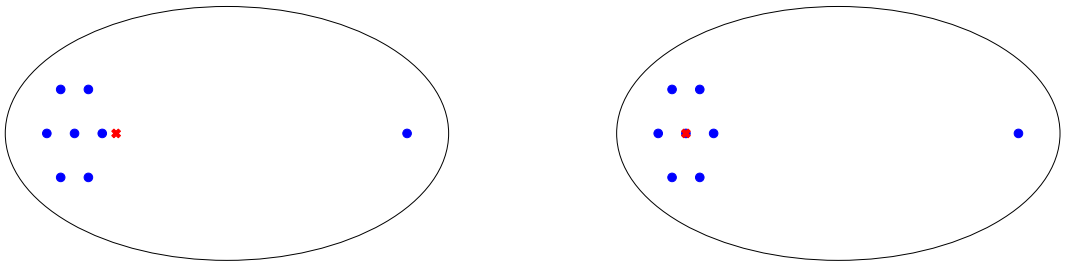
where $\|\cdot\|_2$ is the Euclidean norm and each point in $\mathcal{D}_\mathcal{X}$ is assigned to the cluster represented by its closest centroid. In other words, the goal k-means is to find a set of centroids $\{\boldsymbol{u}_j\}_{j=1}^k$ minimizing the sum of squared distances between each data point and the representative of its cluster, that is, the nearest centroid.

The k-means procedure, described in Algorithm 1, starts by randomly selecting k data points $\{\boldsymbol{u}_j\}_{j=1}^k \subset \mathcal{D}_\mathcal{X}$ to serve as the initial centroids, which act as the center points for the clusters. Next, the algorithm assigns each data point $\boldsymbol{x}_i \in \mathcal{D}_\mathcal{X}$ to the nearest centroid $\boldsymbol{u}_i^* := \arg\min_{1 \le j \le k} \|\boldsymbol{x}_i - \boldsymbol{u}_j\|_2^2$, creating k clusters $\{\mathcal{C}_j\}_{j=1}^k$ based on proximity, where $\mathcal{C}_j = \{\boldsymbol{x}_i \in \mathcal{D}_\mathcal{X} | \boldsymbol{u}_i^* = \boldsymbol{u}_j\}$. After assigning each data point to a cluster, the k-means algorithm recalculates the centroids by taking the mean of all data points assigned to each cluster, that is, $\boldsymbol{u}_j = \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{x} \in \mathcal{C}_j} \boldsymbol{x}$ for all $j = 1, \ldots, k$, where $|\mathcal{C}_j|$ is the cardinality of the j-th cluster. Note that, the new centroids are locations in $\mathbb{R}^d$ that might not be in $\mathcal{D}_\mathcal{X}$. These steps of assigning and updating centroids are repeated until the centroids no longer change significantly or a maximum number of iterations is reached. At this point, we say the algorithm converges. By following this iterative process, the k-means algorithm can effectively cluster data points into distinct groups and provide a set of k centroids as representative of each cluster. Overall, the time complexity of the k-means algorithm is $\mathcal{O}(nki)$, where n is the number of points in the dataset, k is the number of clusters or centroids we consider and i is the number of iterations required for convergence. The computational cost of the k-means algorithm is dominated by the assignment step, consisting of assigning the points to the respective centroid, which has a computational cost of $\mathcal{O}(nk)$ and must be iteratively performed.

The k-means algorithm has two main issues: Firstly, outliers can significantly influence the computation of the mean required to calculate new centroids. Secondly, k-means provides coresets consisting of centroids that may not be included in the initial dataset $\mathcal{D}_\mathcal{X}$, which, depending on the application, may not be of interest. The k-medoids algorithm, which we introduce next, was developed to address these issues.

3.2.2. k-medoids

k-medoids [PJ09] is a variation of k-means that uses data points in $\mathcal{D}_\mathcal{X}$ as cluster representatives. The medoid is the most centrally located object of the cluster with the minimum sum of squared distances to other points within the cluster. k-medoids is more resilient to noise and outliers than k-means by providing coresets of representative points that are subsets of the given data pool. Fig. 3.1 illustrates the difference between the k-means and k-medoids algorithms in a 2-dimensional example, where $k = 1$. In the figure, a group of points arranged in a hexagon shape represents a cluster, while the



(a) Mean (b) Medoid

Figure 3.1.: The figure displays the results of applying k-means and k-medoids algorithms, with k=1, to a dataset consisting of eight 2-dimensional points (in blue). The points are arranged to form an hexagon-shaped cluster with an outlier on the right. In (a), the red point is the centroid selected by the k-means algorithm, which is not one of the points of the dataset. The k-means algorithm is affected by the outlier, resulting in a centroid that does not represent the cluster center accurately. In (b), the point in red is the medoid selected with the k-medoids algorithm, which is one of the points in the dataset. The k-medoids algorithm is robust to the outlier, selecting a medoid that correctly represents the cluster center.

rightmost point is an outlier. On the one hand, Fig. 3.1a shows that when computing the centroid with the k-means algorithm, the outlier affects the result and the centroid cannot correctly represent the cluster center. On the other hand, Fig. 3.1b illustrates that the k-medoids algorithm is robust to the outlier and provides a medoid that correctly represents the cluster center.

The k-medoids attempts to determine clusters $\{\mathcal{C}_j\}_{j=1}^k$ of points in $\mathcal{D}_\mathcal{X}$ and medoids $\{\boldsymbol{m}_j\}_{j=1}^k \subset \mathcal{D}_\mathcal{X}$, solving the following optimization problem

$$\min_{\boldsymbol{m}_1,\dots,\boldsymbol{m}_k \in \mathcal{D}_\mathcal{X}} \sum_{j=1}^{k} \sum_{\boldsymbol{x}\in\mathcal{C}_j} \|\boldsymbol{x} - \boldsymbol{m}_j\|_2^2, \tag{3.2}$$

where, as for the k-means, each point in $\mathcal{D}_\mathcal{X}$ is assigned to the cluster represented by its closest medoid. Thus, the k-medoids procedure aims to minimize the same optimization objective as k-means in (3.1) with the difference that the solution space of the optimization problem is restricted to points in $\mathcal{D}_\mathcal{X}$.

The k-medoids algorithm is conceptually similar to the k-means, iteratively assigning points in the dataset to medoids that are themselves iteratively updated. The major difference with k-means is in the updating process of the medoids, which is restricted to points in $\mathcal{D}_\mathcal{X}$.

Finding an optimal solution to the k-medoids optimization problem in (3.2) is NP-hard. Various heuristic methods have been developed to compute approximate solutions. The partitioning around medoids approach (PAM) [LK90], described in Algorithm 2, is known

Algorithm 2 k-medoids (PAM) algorithm

Input: Data points $\mathcal{D}_{\mathcal{X}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, number of clusters k.
Output: Cluster medoids $\{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k\}$.

1: Initialize cluster medoids randomly: $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k$.
2: **for** $i = 1$ to n **do**
3: Assign each point $\boldsymbol{x}_i$ to the nearest medoid $\boldsymbol{m}_i^* = \underset{1 \leq j \leq k}{\arg\min} \{\|\boldsymbol{x}_i - \boldsymbol{m}_j\|_2^2\}$.
4: $TD \leftarrow \sum_{j=1}^{k} \sum_{\boldsymbol{x} \in \mathcal{C}_j} \|\boldsymbol{x} - \boldsymbol{m}_j\|_2^2$, where $\mathcal{C}_j = \{\boldsymbol{x}_l \in \mathcal{D}_{\mathcal{X}} | \boldsymbol{m}_l^* = \boldsymbol{m}_j\}$.
5: **repeat**
6: **for** $\boldsymbol{m}_l \in \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k\}$ **do**
7: **for** $\boldsymbol{x}_q \in \mathcal{D}_{\mathcal{X}}$ with $\boldsymbol{x}_q \notin \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k\}$ **do**
8: $TD_{\text{swap}} \leftarrow$ compute TD as in step 4 with $\boldsymbol{x}_q$ as a medoid instead of $\boldsymbol{m}_l$
9: **if** $TD_{\text{swap}} < TD$ **then**
10: $TD \leftarrow TD_{\text{swap}}$ and $\boldsymbol{m}_l \leftarrow \boldsymbol{x}_q$.
11: **until** Convergence criterion is met

Algorithm 3 k-medoids (alternating) algorithm

Input: Data points $\mathcal{D}_{\mathcal{X}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, number of clusters k.
Output: Cluster medoids $\{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k\}$.

1: Initialize cluster medoids randomly: $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k$.
2: **repeat**
3: **for** $i = 1$ to n **do**
4: Assign each point $\boldsymbol{x}_i$ to the nearest medoid $\boldsymbol{m}_i^* = \underset{1 \leq j \leq k}{\arg\min} \{\|\boldsymbol{x}_i - \boldsymbol{m}_j\|_2^2\}$
5: **for** $j = 1$ to k **do**
6: $\boldsymbol{m}_j = \underset{\hat{\boldsymbol{x}} \in \mathcal{C}_j}{\arg\min} \sum_{\boldsymbol{x} \in \mathcal{C}_j} \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2$ where $\mathcal{C}_j = \{\boldsymbol{x}_i \in \mathcal{D}_{\mathcal{X}} | \boldsymbol{m}_i^* = \boldsymbol{m}_j\}$
7: **until** Convergence criterion is met

to be one of the most effective algorithm for the k-medoids problem. PAM approach starts by randomly selecting k data points $\{\boldsymbol{m}_j\}_{j=1}^{k} \subset \mathcal{D}_{\mathcal{X}}$ to serve as the initial medoids and assigning each point in $\mathcal{D}_{\mathcal{X}}$ to its closest medoid. Next, considers for each of the initially randomly selected medoids, swapping it with each non-medoid point available and for each possible swap calculates the total cost, which is the optimization objective in (3.2). After considering all the possible swap combinations, PAM chooses the medoid-non-medoid pairs with the lowest total cost and performs the swap. The computational cost of PAM is $\mathcal{O}(k(n-k)^2 i)$, where k is the number of clusters considered, n the number of data points in the given data pool $\mathcal{D}_{\mathcal{X}}$ and i the number of iterations of the medoids update procedure [JH17]. The computational complexity of PAM scales quadratically with the number of data points, making it not suitable to analyze large datasets. To address the computational cost issue, alternative approaches building on the PAM procedure have been proposed for the medoids update, such as CLARA [KR86] applying the PAM algorithm on smaller subsets of $\mathcal{D}_{\mathcal{X}}$, CLARANS [NH02], an extension of CLARA based on randomized search, Fast and Faster PAM [SR21] implementing faster approaches for the swap procedure, and others [VdLPB03, ZC05]. A simpler and computationally efficient method, not inspired by PAM, was developed in [PJ09] where the authors propose to update the medoids similarly to k-means, by selecting as the new medoid of each cluster the data point minimizing the total distance to other objects in its cluster. This approach, described in Algorithm 3, is known as "alternating". While it can lead to poorer performances than PAM-based approaches, its authors claim it has a runtime of $\mathcal{O}(nk)$ per iteration, as for the k-means. In practice, the alternate approach is widely used because of its simplicity and computational efficiency, making it suitable for analyzing very large datasets. In Table 3.1 we recapture information from [SL22] providing an overview of the various libraries available in different programming languages to implement the k-medoids including sklearn_extra [sci21], kmedoids [SL22], ELKI [SZ19], PyClustering [Nov19] and biopython [CAC+09]. This table serves as resource for researchers and practitioners who are looking for efficient ways to implement k-medoids in their projects.

Library	**Algorithm**	**Language**
sklearn_extra	Alternating	Python
kmedoids [SL22]	Alternating	Python, Rust
ELKI	Alternating	Java
biopython	Alternating	Python, C
kmedoids	FasterPAM	Python, Rust
ELKI	FasterPAM	Java
kmedoids	PAM	Python, Rust
ELKI	PAM	Java
sklearn_extra	PAM	Python
PyClustering	PAM	Python, C++

Table 3.1.: Libraries for k-medoids

3.2.3. The importance of initializing: k-medoids++

We mentioned various methods for updating the initially randomly selected medoids. However, the choice of the initial medoids significantly impacts the quality of the clustering solution and, therefore, of the chosen coreset consisting of the selected medoids. Several heuristics have been proposed for initializing the k-medoids algorithm. Such methods have different computational costs and performances. We mention the most relevant. The BUILD [LK90] initialization approach was developed and proposed together with the PAM updating procedure as an alternative to the naive random choice commonly used with k-means. BUILD is a greedy approach initializing the medoids by iteratively choosing the point minimizing the optimization objective in (3.2), starting with the point for which the sum of the distances with all other points is as small as possible. Initializing the k-medoids algorithm with the BUILD approach takes $\mathcal{O}(n^2k)$ operations.

The work done in [EC91] integrates the "alternating" approach into the greedy BUILD heuristic, updating the medoid of the existing clusters when the next medoid is added. Such an algorithm called "GreedyG" can provide initial medoids for which the optimization objective has even lower values than with BUILD and, therefore, can be considered better distributed. Nonetheless, GreedyG still requires at least $\mathcal{O}(n^2k)$ operations. A faster but still effective alternative for the initialization of the medoids was developed initially for k-means and takes the name of k-means++[AV07]. The initialization of the medoids based on k-means++, described in Algorithm 4, consists of selecting the first medoid uniformly at random from the available pool and then choosing the next medoids at random with probability proportional to their distance to the nearest selected medoids. If we assume there is a cluster of points and no medoid nearby, the probability mass of this cluster is high, and the next medoid will likely be selected within that cluster. After that, the presence of the new nearby medoid decreases the probability mass of the cluster. While there are works arguing that BUILD still provides better initialization [SR19], the procedure based on k-means++ provides points that are $\mathcal{O}(\log k)$ competitive to the optimal solution, that is, there are theoretical guarantees that it generates good starting conditions. Moreover, it requires only $\mathcal{O}(nk)$ operations because it computes the distances to all other points only for the k iteratively selected medoids [LPP14], thus making it more suitable for analyzing large datasets. We refer to the k-medoids algorithm initialized with this procedure as k-medoids++.

Another initialization strategy with a computational cost of $\mathcal{O}(nk)$ is the linear

Algorithm 4 k-medoids++ initialization procedure

Input: Data points $\mathcal{D}_{\mathcal{X}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, number of clusters k.
Output: Cluster medoids $\{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k\}$.

1: Randomly choose the first medoid $\boldsymbol{m}_1 \in \mathcal{D}_{\mathcal{X}}$ with uniform probability
2: **for** $i = 2$ to k **do**
3: Compute the distance $D(\boldsymbol{x})$ of each data point $\boldsymbol{x}$ to the nearest medoid
4: Choose the $\boldsymbol{x}_i \in \mathcal{D}_{\mathcal{X}}$ as the next medoid with probability $\frac{D(\boldsymbol{x}_i)^2}{\sum_{\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}} D(\boldsymbol{x})^2}$

approximation of the original PAM BUILD (LAB). LAB consists of applying the BUILD algorithm on subsets of the data set. While [SR19] provides examples where this initialization outperforms the one based on k-means++, no theoretical guarantee is provided to motivate its improved effectiveness, which is supported mainly by their experiments.

3.3. Greedy approaches

Greedy approaches are a class of selection methods that start with an empty set and iteratively select the next points according to some deterministic principle, aiming to optimize certain objectives. To be more specific, let us start considering a dataset $\mathcal{D}_{\mathcal{X}} := \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ and a utility set function $f : 2^{\mathcal{D}_{\mathcal{X}}} \to \mathbb{R}^+$, mapping subsets of $\mathcal{D}_{\mathcal{X}}$ into some positive real value determining the utility of the subset, where $2^{\mathcal{D}_{\mathcal{X}}}$ represent the collection of all possible subsets of $\mathcal{D}_{\mathcal{X}}$. Greedy approaches are particularly useful for solving optimization tasks involving combinatorial or discrete choices to maximize or minimize the utility function f, that is, greedy algorithms can efficiently find solutions to optimization problems such as

$$\max_{S \subset \mathcal{D}_{\mathcal{X}}} f(S) \text{ and } \min_{S \subset \mathcal{D}_{\mathcal{X}}} f(S), \tag{3.3}$$

subject to some constraint on S, e.g., a cardinality constraint. Although greedy algorithms do not guarantee the optimal solution to the optimization problems in (3.3), they can provide solutions that are fast to compute and sufficiently close to the optimal solution for a wide range of problems, including NP and NP-hard problems. In some cases the fact that greedy algorithms provide good solutions to complicated optimization problems can even be proven mathematically. For instance, if the set function f has certain properties, such as submodularity, it is possible to show that greedy procedures provide solutions close to the optimal one, as we see later. We note that the greedy procedures aiming at solving optimization problems as in (3.3), may consider data labels or label predictions of a fixed model, as in the case of active learning [WIB15]. Nevertheless, our focus is on optimization problems and greedy procedures that solely rely on locations of the data points in the feature space and do not consider data labels or data label predictions.

Greedy algorithms Given a dataset $\mathcal{D}_{\mathcal{X}} := \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, greedy algorithms start with an initial set $S_0 \subset \mathcal{D}_{\mathcal{X}}$, possibly empty, and iteratively augment it. At the i-th iteration, a point $\boldsymbol{x}_i \in \mathcal{D}_{\mathcal{X}}$ is selected according to some specific principle obtaining a set

$$S_i = S_{i-1} \cup \boldsymbol{x}_i. \tag{3.4}$$

In general, the principle for selecting the new point $\boldsymbol{x}_i$ depends on the application and optimization problem at hand. We mention two examples of greedy procedures for selecting the new point, where together with $\mathcal{D}_{\mathcal{X}}$, we are provided with a utility set function $f : 2^{\mathcal{D}_{\mathcal{X}}} \to \mathbb{R}$. The first greedy procedure selects, at the $i-th$ iteration, the point $\boldsymbol{x}_i \in \mathcal{D}_{\mathcal{X}}$ satisfying the following principle.

$$\boldsymbol{x}_i := \arg\max_{\boldsymbol{x}\in\mathcal{D}_\mathcal{X}} |f(S_{i-1}\cup\boldsymbol{x}) - f(S_{i-1})|, \tag{3.5}$$

Thus, this greedy procedure aims to iteratively select a new data point that, if added to the already selected set, maximizes change in the output of the utility function. The second greedy procedure selects the $i-th$ point according to the following principle

$$\boldsymbol{x}_i := \arg\max_{\boldsymbol{x}\in\mathcal{D}_\mathcal{X}\setminus S_{i-1}} f(\boldsymbol{x}). \tag{3.6}$$

In other words, it aims to select a new point among those not yet selected that maximizes the output of the utility function f. Note that, we could perform different choices for the iterative selection steps than those in (3.5) and (3.6). For instance, we could consider the minimum instead of the maximum. Nevertheless, the greedy selection principles in (3.5) and (3.6) can be employed to find solution to a large variety of interesting problems, as we see shortly. Note also that they are independent of the specific application as they only depend on the choice of the set function f.

Submodular function optimization problems is a class of NP-hard optimization problems that often arise in machine learning, and for which the greedy procedure in (3.5) is guaranteed to find solutions close to the optimal [Bil22]. In the following sections, we define submodular functions, provide examples of submodular functions that are commonly used to find coresets in various applications and describe the related optimization problems. We also explain why the greedy algorithm is particularly effective in solving these problems.

3.3.1. Optimization problems with submodular functions

In this subsection we follow along [KG14] and introduce submodularity, which is a property of set functions $f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}$, that is, functions that map subsets of a given dataset, $\mathcal{D}_\mathcal{X} := \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, into a scalar value. There are two equivalent definitions of submodularity that we now provide

Definition 3.1 *A set function $f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}$ is submodular if for every $A \subset B \subset \mathcal{D}_\mathcal{X}$ and $\boldsymbol{x} \in \mathcal{D}_\mathcal{X}\setminus B$ it holds that*

$$f(A\cup\boldsymbol{x}) - f(A) \geq f(B\cup\boldsymbol{x}) - f(B). \tag{3.7}$$

Equivalently, a set function $f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}$ is submodular if for every $A, B \subset \mathcal{D}_\mathcal{X}$,

$$f(A\cap B) + f(A\cup B) \leq f(A) + f(B). \tag{3.8}$$

If in (3.7) and (3.8) the equality holds, then we say the set function f is modular.

In other words, the more intuitive formula (3.7) says that a function is submodular if adding an element to a smaller set gives a larger increase in the function value compared to adding the same element to a larger set. Thus, submodularity captures the idea that

the more points have been already selected, the less additional value each new point adds. This is known as the diminishing return property.

Submodular functions have various useful properties that make them a very powerful and adaptable tool for coresets selection. For instance, submodularity is preserved under non-negative linear combinations of submodular functions, that is, if $f_1, f_2, \ldots, f_m$ are submodular set functions and $\alpha_1, \alpha_2, \ldots, \alpha_m$ are non-negative scalar values, then $g := \sum_{i=1}^{m} \alpha_i f_i$ is also submodular. This property allows designing more complex submodular functions by linear combination of simpler ones. Additionally, residuals of submodular functions are also submodular, that is, given $f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}$ submodular, disjoint sets $A, B \subset \mathcal{D}_\mathcal{X}$, then $g : 2^A \to \mathbb{R}$, with $g(S) := f(S \cup B) - f(B)$, $S \subset A$ is submodular. See [KG14] to read about other interesting properties of submodular functions.

A subclass of submodular functions that are of particular interest to coreset selection are those that are non-decreasing.

Definition 3.2 *A set function $f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}$ is non-decreasing if for each $A \subset B \subset \mathcal{D}_\mathcal{X}$ $f(A) \leq f(B)$.*

Thus, non-decreasing set functions do not decrease as the argument set gets larger. Next we provide two examples of classes of submodular non-decreasing set functions: the weighted coverage and the facility location functions.

Weighted coverage Weighted coverage functions are among the simplest examples of submodular functions. Let us start introducing the simpler modular weighted coverage functions, for which the equality holds in (3.7), that is, for every $A \subset B \subset \mathcal{D}_\mathcal{X}$ and $\boldsymbol{x} \in \mathcal{D}_\mathcal{X} \backslash B$ it holds that $f(A \cup \boldsymbol{x}) - f(A) = f(B \cup \boldsymbol{x}) - f(B)$. Given a dataset $\mathcal{D}_\mathcal{X}$, a weight function $w : \mathcal{D}_\mathcal{X} \to \mathbb{R}$ and a set $S \subset \mathcal{D}_\mathcal{X}$, we define a modular weighted coverage set function $f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}$ as

$$f(S) := \sum_{\boldsymbol{x} \in S} w(\boldsymbol{x}). \tag{3.9}$$

The weighted coverage set function in (3.9) associates to each subset of the dataset a scalar value given by the sum of all the weights associated with the points in the considered subset. The weight function $w : \mathcal{D}_\mathcal{X} \to \mathbb{R}$, mapping data points into their respective scalar weight, must be defined a priori and depends on the specific application. Notice that, if the weight function is non-negative, then the modular weighted coverage function in (3.9) is non-decreasing. The simplest example of weight function is the constant function, e.g., $w(\boldsymbol{x}) = 1$ for all $\boldsymbol{x} \in \mathcal{D}_\mathcal{X}$. In such a case we have that $f(S) := |S|$, which is easy to verify is modular non-decreasing. More elaborate examples of weighted coverage functions are those providing the weighted coverage of a collection of subsets of the dataset $\mathcal{D}_\mathcal{X}$. Consider a non-negative modular function $f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}^+$ as in (3.9), a collection of subsets $C \subset 2^{\mathcal{D}_\mathcal{X}}$ and subcollection $D \subset C$ then the function

$$g(D) := f(\bigcup_{S \in D} S) = \sum_{\boldsymbol{x} \in \bigcup\limits_{S \in D} S} w(\boldsymbol{x}) \tag{3.10}$$

is submodular non-decreasing. It is possible to show that g is submodular for any submodular function f. Moreover, g is non-decreasing if and only if f is non-decreasing [KG14].

It is interesting to note that greedy approaches considering a weighted coverage set function can be used to implement importance sampling techniques, discussed in Subsection 3.1, in a deterministic fashion. Recall that, importance sampling approaches assign weights to data points based on their relevance to the problem at hand. If the importance weights do not change over the selection process and can be defined by a single weight function $w : \mathcal{D}_{\mathcal{X}} \to \mathbb{R}$, then a greedy algorithm aiming at maximizing the weighted coverage function defined by the computed importance weights can be used to select data points deterministically according to their importance.

Facility location Another important class of submodular functions are facility location functions, originally designed to determine the optimal location of facilities or service centers in order to minimize costs or maximize coverage. Given a dataset $\mathcal{D}_{\mathcal{X}}$, a similarity function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ and a subset $S \subset \mathcal{D}_{\mathcal{X}}$, a facility location function $f : 2^{\mathcal{D}_{\mathcal{X}}} \to \mathbb{R}$ quantifies how well the selected points in S cover the entire set according to the following principle

$$f(S) := \sum_{\tilde{\boldsymbol{x}} \in \mathcal{D}_{\mathcal{X}}} \max_{\bar{\boldsymbol{x}} \in S} \phi(\tilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}), \tag{3.11}$$

with $f(\emptyset) := 0$. Given that the similarity function ϕ is non-negative, the set function defined in (3.11) is submodular non-decreasing. It is interesting to note that if in (3.11) we set the similarity function

$$\phi(\tilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}) := m_0 - \|\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}}\|_2^2, \tag{3.12}$$

with $\|\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}}\|_2^2 \leq m_0$ for each $\tilde{\boldsymbol{x}}, \bar{\boldsymbol{x}} \in \mathcal{D}_{\mathcal{X}}$, then the problem of maximizing the facility location function is equivalent to the k-medoids minimization problem in (3.2) [MKSK16]. Note that other choices are possible for the similarity function ϕ. For instance, the similarity based on the Gaussian function [BCD+24], i.e.,

$$\phi(\tilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}) := e^{-\gamma \|\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}}\|_2^2}, \tag{3.13}$$

with γ hyperparameter to be fine-tuned. Facility location functions have been very successfully employed to improve various aspects of machine learning models, such as data-efficient training of machine learning models [MBL20, PDM22], robust training of neural networks against noisy labels [MCL20], accelerating training process [PYM+23], maximize accuracy performance of classifiers trained on sets of limited size [WLKB14], and, more recently, for selecting effective datasets for fine-tuning of large language models [BCD+24].

While we mentioned weighted coverage and facility location functions as examples of submodular functions, there are many other submodular functions used in a wide range of machine learning applications. For instance, entropy and mutual information, which can be used in sensor placement problems [KSG08, SDK15], cut functions for feature selection in text classification [KNTB09], and saturated coverage for document

summarization [LB11]. These applications all involve using a greedy algorithm to find a subset that maximizes a submodular function. This means that submodular function maximization problems arise in many ML areas. The underlying common denominator of these problems is that their solutions typically yield subsets that offer greater diversity, information, spread, or coverage [Bil22].

It is worth mentioning that also submodular minimization problems are of interest in various applications. In particular, solving submodular minimization problems typically leads to sets with increased homogeneity, conformity, or coherence. For instance, image segmentation problems can be addressed via submodular function minimization [ENV17]. Consider a scenario where we analyze a dataset consisting of pixels in an image. In such a case, we may want to select a group of pixels that represent a specific object and have similar properties, such as color and luminance. Thus, in this case, we may be interested in selecting a subset of the dataset that, even if large, minimizes the diversity of the selected elements, quantified by the submodular function value. Nevertheless, in this work, we focus on coreset strategies for training data development in ML regression, which typically requires solving tasks such as data summarization and data compression. Thus, we are mainly interested in selecting subsets maximizing data representation, diversity and information. Consequently, we focus on submodular function maximization.

Note that if a function f is submodular then $-f$ may not be. That is, submodular maximization of a function f can not be rephrased as submodular minimization of the function $-f$. The author of [Bil22] mentions that "Unconstrained submodular maximization is NP-hard (albeit approximable), and this is not surprising given that there are an exponential number of sets needing to be considered. Remarkably, submodular minimization does not require exponential computation, is not NP-hard, and in fact, there are polynomial time algorithms for doing so...". This implies that, solving submodular minimization problems is generally computationally less demanding than solving submodular maximization problems. We refer the reader to [Fuj05, Iwa07] for further details on submodular function minimization. Next, we describe submodular function maximization problems formally and present a result that proves the effectiveness of greedy algorithms in providing solutions that are close to optimal.

Greedy submodular functions maximization

Typically, given dataset $\mathcal{D}_{\mathcal{X}}$ and set function $f : 2^{\mathcal{D}_{\mathcal{X}}} \to \mathbb{R}$, a submodular function maximization problem takes the form

$$\max_{S \subset \mathcal{D}_{\mathcal{X}}} f(S) \text{ subject to some constraint on S} \tag{3.14}$$

Here we consider the simpler cardinality constraint on the selected set, that is, we require that $|S| \leq b$ for some budget value $b \in \mathbb{N}$. More complicated constraints could be applied. For instance, if the points in the dataset are associated with some categories or classes, we could require that the selected set consists of points that are fairly distributed among the categories. Unfortunately, even considering the simpler cardinality constraint, the submodular maximization problem is NP-Hard [KG14].

Various branch and bound algorithms have been developed to address the submodular function maximization problem [NW81, KNTB09]. However, because of the hardness of the optimization problem in (3.14) they are not scalable to large datasets. In [NWF78], the authors prove that the greedy procedure in (3.5) provides good approximations of the optimal solution to the problem in (3.14) of maximizing a non-decreasing submodular function, which is NP-Hard. In particular, they propose the following theorem

Theorem 3.1 ([NWF78]) *Fix a non-negative non-degreasing submodular function* $f : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}^+$*, and let* $\{S_i\}_{i\geq 0}$ *be the greedy sets defined in (3.4) selected according to the principle in (3.5), with* $S_0 = \emptyset$*. Then for all positive integers* b *and* l*,*

$$f(S_l) \geq \left(1 - e^{-\frac{l}{b}}\right) \max_{S:|S|\leq b} f(S). \tag{3.15}$$

In particular, for $l = b$*,* $f(S_b) \geq \left(1 - \frac{1}{e}\right) \max_{S:|S|\leq b} f(S)$*.*

Theorem 3.1 demonstrates that greedy algorithms can effectively compute good approximate solution to submodular maximization problems. The greedy algorithm considered in Theorem 3.1 requires $\mathcal{O}(nb)$ evaluations of the submodular function, where n is the size of the dataset and b is the number of points that needs to be selected. This is because at each iteration of the algorithm, in order to perform the greedy step defined in (3.5) and select the $(i+1)$-th point, one has to compute the function value $f(S_i \cup \{\boldsymbol{x}\})$ for all $\boldsymbol{x} \in \mathcal{D}_\mathcal{X}$ that are not in S_i. The cost of evaluating the submodular function may change depending on the function considered. For instance, evaluating the quantity $f(S_i \cup \{\boldsymbol{x}\})$ for the weighted coverage function in (3.9), assuming that the weights are given, has a cost of $\mathcal{O}(1)$ as it simply requires to add the weight associated with the data point $\boldsymbol{x}$ to the sum of the weights associated with the points in S_i already computed in the previous step. Alternatively, for the case of facility location functions as in (3.11), assuming that the pairwise similarities are already provided, computing the quantity $f(S_i \cup \{\boldsymbol{x}\})$ for some $S_i \subset \mathcal{D}_\mathcal{X}$ has a cost of $\mathcal{O}(n)$. This is because in order to evaluate the function value one has to check for each of the n points in the dataset if the new considered point $\{\boldsymbol{x}\}$ added to S_i is the one that maximizes the pairwise similarity or not. Moreover, if the similarities between the data points need to be computed before initializing the greedy algorithm, then the overall computational effort could be higher. For example, if the similarity between two points is based on the computation of their Euclidean distance, then computing the similarities would cost at least $\mathcal{O}(n^2)$.

Various approaches have been developed to make the greedy procedure for submodular function maximization faster than its naive version considered in Theorem 3.1. Among the most relevant of these implementations there is lazy (or accelerated) greedy [Min78], which uses the property of diminishing returns of submodular functions to avoid re-evaluating examples that provide little gain. Another important implementation is Stochastic greedy [MBK+15], a randomized version of the greedy algorithm that finds near optimum solution in time linear in the size of the dataset and independent of the cardinality constraint. Additionally, there is GreeDi [MKSK16] a two-stage algorithm for distributed submodular function maximization designed to work on datasets that are

too large to fit into memory. There are various libraries that can perform submodular function optimization considering different submodular functions and greedy procedures. Apricot[SBN20] is a peer-reviewed and widely used Python library for submodular function optimization. Other available libraries for submodular function optimization are SFO [Kra10] written in MATLAB and Submodlib[KRI22] in Python.

The usefulness of greedy algorithms is not restricted to submodular maximization problems. Another group of problems that can be solved effectively and efficiently using greedy procedures are the min-max-min problems, which we introduce next.

3.3.2. Greedy algorithms for Min-Max-Min optimization problems

Given dataset $\mathcal{D}_\mathcal{X}$, a collection of set functions $f_i : 2^{\mathcal{D}_\mathcal{X}} \to \mathbb{R}$, $i = 1, \ldots, m$, for some integer $m \in \mathbb{N}$ and a budget $b \in \mathbb{N}$, min-max-min optimization problems have the form

$$\min_{\substack{S \subset \mathcal{D}_\mathcal{X} \\ |S|=b}} \max_{1 \leq i \leq m} \min_{\boldsymbol{x} \in S} f_i(\boldsymbol{x}). \tag{3.16}$$

The authors of [LPW23], show that greedy procedures can be used to find good solutions for all min-max-min problems that have smooth and strongly convex objectives. One relevant min-max-min problem in ML applications is the k-center problem [HP11] defined as follows

$$\min_{\substack{S \subset \mathcal{D}_\mathcal{X} \\ |S|=k}} \max_{\boldsymbol{x}_i \in \mathcal{D}_\mathcal{X}} \min_{\boldsymbol{x}_j \in S} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2, \tag{3.17}$$

where $\|\cdot\|_2$ is the Euclidean norm. Note that formula (3.17) is a specific case of (3.16), which we can obtain by setting $f_i(\boldsymbol{x}_j) := \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$ for all $\boldsymbol{x}_i \in \mathcal{D}_\mathcal{X}$ and $\boldsymbol{x}_j \in S$.

In words, the k-center problem consists of selecting k points, or centers, from a given dataset so that the maximum distance between a point in the dataset and its closest center is minimized. Put differently, the optimization problem in (3.17) aims to select a set $S \subset \mathcal{D}_\mathcal{X}$ minimizing the fill distance of the selected set defined as follows

$$h_{S,\mathcal{D}_\mathcal{X}} := \max_{\boldsymbol{x}_i \in \mathcal{D}_\mathcal{X}} \min_{\boldsymbol{x}_j \in S} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2. \tag{3.18}$$

The quantity on the right-hand side of (3.18) is called the fill distance of S in $\mathcal{D}_\mathcal{X}$. It is a quantity we can associate to subsets of the dataset $\mathcal{D}_\mathcal{X}$, and it indicates that any point $\boldsymbol{x}_i \in \mathcal{D}_\mathcal{X}$ has a point $\boldsymbol{x}_j \in S$ not farther away than $h_{S,\mathcal{D}_\mathcal{X}}$.

The concept of fill distance is not new and has been applied in various contexts that do not involve a k-center problem. For example, it is associated with the earliest notion of dispersion, which is used to quantify approximation errors in Quasi-Monte Carlo optimization methods [Hla76, RT96]. Additionally, the authors of [JMY90] use the fill distance in the context of designing computer experiments, linking it to specific optimality criteria in experimental design under the assumption that the experiments are generated by a Gaussian process or a similar framework. It is only more recently that the fill distance has been used in connection with the k-center problem to assess

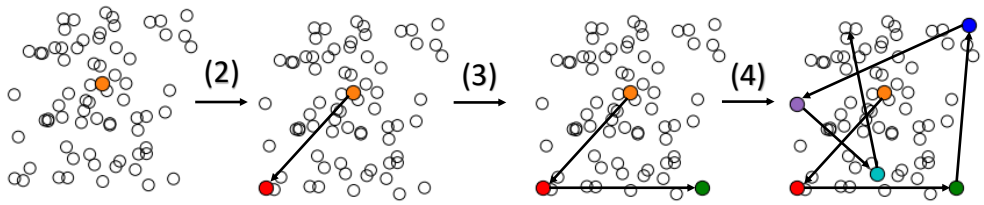


Figure 3.2.: Illustration of the Farthest Point Sampling iterative selection procedure.

Algorithm 5 Farthest Point Sampling (FPS)

Input Dataset $\mathcal{D}_{\mathcal{X}} = \{\boldsymbol{x}_i\}_{i=1}^{n} \subset \mathcal{X}$ and data budget $b \in \mathbb{N}$, $b \ll n$.
Output Subset $\mathcal{L}_{\mathcal{X}}^{FPS} \subset \mathcal{D}_{\mathcal{X}}$ with $|\mathcal{L}_{\mathcal{X}}^{FPS}| = b$.

1: Choose $\hat{\boldsymbol{x}} \in \mathcal{D}_{\mathcal{X}}$ randomly and set $\mathcal{L}_{\mathcal{X}}^{FPS} = \hat{\boldsymbol{x}}$.
2: **while** $|\mathcal{L}_{\mathcal{X}}^{FPS}| < b$ **do**
3: $\quad \bar{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}_q \in \mathcal{D}_{\mathcal{X}}} \min_{\boldsymbol{x}_j \in \mathcal{L}_{\mathcal{X}}^{FPS}} \|\boldsymbol{x}_q - \boldsymbol{x}_j\|_2$.
4: $\quad \mathcal{L}_{\mathcal{X}}^{FPS} \leftarrow \mathcal{L}_{\mathcal{X}}^{FPS} \cup \bar{\boldsymbol{x}}$.

the quality of selected sets for training predictive models in classification or regression tasks [SS18, CG24].

Unfortunately, the fill distance minimization problem is NP-Hard [Hoc84], as the other min-max-min optimization problems, but there is a greedy algorithm called Farthest Point Sampling (FPS) that provides fast and effective solutions. The FPS, described in Algorithm 5 and illustrated in Fig. 3.2, iteratively selects points from a dataset by first choosing a random initial point. Next, in each iteration it selects the point the farthest away from the currently chosen subset, ensuring a maximally spaced subset of points.

The following theorem shows that the FPS selects sets with fill distance at most a factor of 2 from the minimal fill distance.

Theorem 3.2 ([HP11]) *Assume $O \subset \mathcal{D}$ is a subset of cardinality b with minimal fill distance. Then, the fill distance of a set $\mathcal{L}^{FPS} \subset \mathcal{D}$, $|\mathcal{L}^{FPS}| = b$, obtained using FPS, is at most two times the minimal fill distance, that is,*

$$h_{\mathcal{L}_{\mathcal{X}}^{FPS}, \mathcal{D}_{\mathcal{X}}} \leq 2 h_{O_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}}. \tag{3.19}$$

FPS can be implemented using $\mathcal{O}(n)$ space and takes $\mathcal{O}(nb)$ time [HP11], where n is the amount of points in the dataset and b is the amount of points to select. It is worth to note that reducing the factor of approximation below 2 would require solving an NP-hard problem [HS85]. Thus, FPS provides a suboptimal solution, but obtaining a better approximation with theoretical guarantees would not be feasible in polynomial time. In applications the FPS has been used as a passive and model-agnostic sampling strategy to select training sets improving prediction performances of various machine

learning models [YK10, WLH19, SS18, CG24] or to ensure diversity in the selected training data [CHE+21, DBB+21]. Note that, step 3 of Algorithm 5 can be accurately modified so that the resulting algorithm can address various min-max-min problems with robust theoretical approximation guarantees. See [LPW23] for more details about the quality of greedy algorithmic solutions for this class of optimization problems.

3.4. A conceptual comparative analysis of coresets

We think the effectiveness of a data sampling procedure it is determined by the strategy it implements but also by the context in which such strategy is applied. Specifically by the relation between the underlying data distribution and the final goal of the sampling procedure. Simply put, data selection approaches are not "good" or "bad" a priori. For instance, in [SDF+24] the authors provide examples where the most straightforward uniform random selection is as effective, or even outperforms, more sophisticated active learning procedures, for training data selection for supervised ML. Therefore, understanding and comparing the underlying principles implemented by the various sampling methodologies and how these relate to the problem at hand is crucial for choosing an appropriate selection method. Next, we provide a conceptual comparative analysis of the coresets mentioned in the previous sections, such as uniform random sampling, importance sampling, cluster-based methods, and greedy methods. We highlight the main characteristics of the different sampling approaches and comment on their strengths or weaknesses in relation to training data selection. We remark that, the effectiveness of a given sampling approach is determined by the strategy it implements, but it also strongly depends on the specific context of the application. Nonetheless, in what follows we emphasize some general characteristics and principles of various coreset strategies that may be useful to consider when choosing a suitable sampling technique.

Uniform sampling is fast and easy to understand and implement. It gives each data point in a dataset an equal chance of selection, aiming for uniformity among sampled points. However, it does not guarantee a uniform distribution across the data space. This is a key aspect to consider in training data selection for supervised ML regression, where we may be interested in maximizing diversity of the selected points. Fig. 3.3 shows results of uniform random sampling on two datasets of 1000 two-dimensional points in the unit square $[0, 1]^2$. One dataset consists of uniformly distributed points, while the other has 150 uniformly distributed points and 850 normally distributed points clustered at the center and enclosed in a red circle. The figure illustrates that random sampling may be uniform over the dataset but might not accurately represent a uniform distribution across the entire data space. If points in a dataset cluster around specific locations of the data space, a random sampled subset may not include points in areas with lower data density, e.g., upper-right corner of the unit square associated with uniform random sampling in the bottom row of the figure. Alternatively, if the points are "well" distributed and the goal is to select a subset that reflects their distribution, uniform random sampling can be effective.

Importance sampling prioritizes the selection of data points that are more beneficial

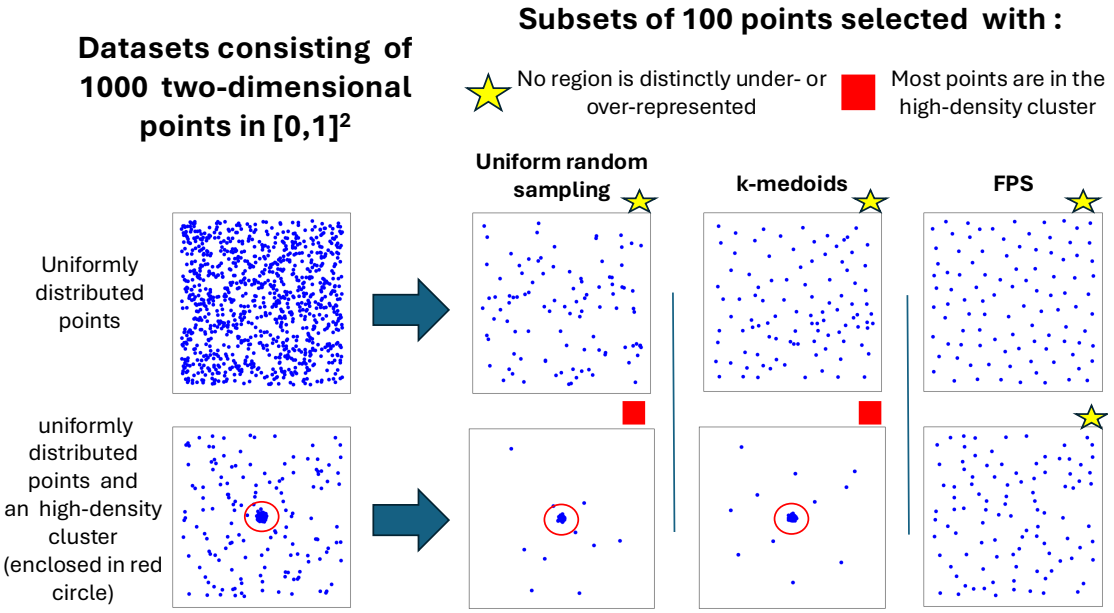
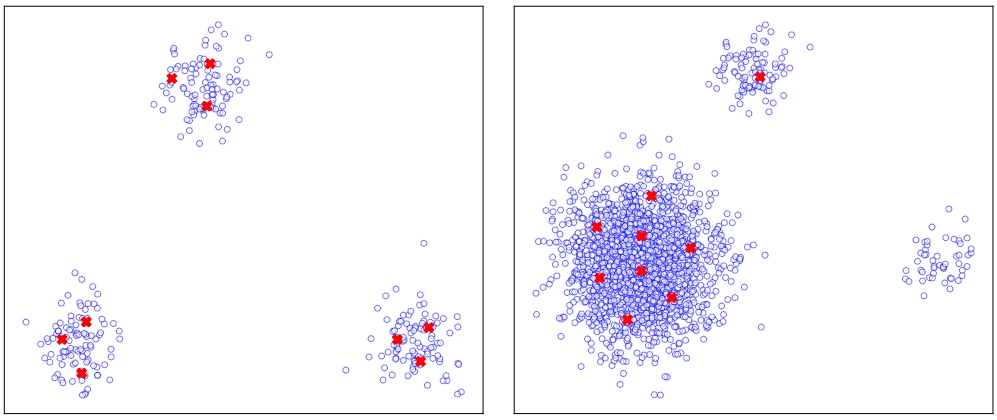


Figure 3.3.: Comparison of three sampling methods on two datasets shown on the left. Each dataset contains 1000 two-dimensional points in the unit square $[0, 1]^2$. One dataset consists of uniformly distributed points, while the other has 150 uniformly distributed points and 850 normally distributed points, forming a cluster in the center within a red circle. The scatter plots on the right-hand side show that the distribution of 100 points selected through uniform random sampling and k-medoids depends on the dataset's point distribution. Moreover, uniform random sampling may neglect areas with lower data density, such as the upper right corner of the unit square. In contrast, FPS sampling offers broader coverage of the data space, regardless of the dataset's distribution.

in a given context. This sampling strategy is based on a concept of importance that must be defined a priori. Suppose we know a suitable concept of importance that we can use to effectively rank the data points by defining importance weights. In such a scenario, this method allows for targeted data selection that aligns with the final goal of the sampling procedure. That is, this method is characterized by strong adaptability to specific contexts. However, accurately determining the importance of data points can be challenging, and incorrect estimation of the importance weights may lead to biased selections. Therefore, the effectiveness and reliability of importance sampling strongly depend on how well and reliably we can define a concept of importance and compute the relative importance weights in a given context.

Cluster-based sampling typically ensures that all segments in a given dataset are represented in the selected set. Consequently, cluster-based sampling approaches, such as k-medoids, are particularly beneficial in scenarios where the dataset from which we



(a) Balanced clusters (b) Unbalanced clusters

Figure 3.4.: Results of k-medoids sampling on two two-dimensional datasets with $k = 9$ and initialized with random sampling. The blue circles are the data points from which k-medoids selects, and the red crosses are the selected points. (a) A balanced dataset consisting of three clusters with 100 points each. k-medoids selected three samples per cluster. (b) Unbalanced dataset with three clusters of different sizes, consisting of 2000, 100 and 50 points. k-medoids selected eight points from the larger cluster and one from the medium-sized cluster.

sample is segmented into different clusters. It is important to note that cluster-based sampling may not be effective in scenarios where clusters have large size imbalances. For instance, approaches similar to k-medoids, aiming at minimizing the distance between data points in a cluster and the selected cluster center, may fail to effectively represent the smaller clusters in the selected set. Such approaches may focus the sampling mainly on larger clusters. Fig. 3.4 shows the results of k-medoids sampling on two two-dimensional datasets with $k = 9$. Fig. 3.4a illustrates the result of the sampling from a balanced dataset consisting of three clusters with 100 points each. k-medoids selects three samples per cluster. That is, the selected set equally represents all the segments in the dataset. Fig. 3.4b illustrates the result of the sampling from an unbalanced dataset with three clusters of different sizes, consisting of 2000, 100 and 50 points. k-medoids selects eight points from the larger cluster, one from the medium-sized one and neglects entirely the smaller one. This simple illustrative example highlights one of the challenges that a cluster-based approach relying on a data similarity concept may encounter. Cluster size imbalance may lead to prioritizing sampling in the larger clusters, potentially neglecting points in the smaller ones. Furthermore, Fig. 3.3 illustrates that, the distribution of the points selected with uniform random sampling and k-medoids are qualitatively very similar, independently of the underlying distribution of the points in the dataset from which the subsets are sampled. Fig. 3.3 shows that k-medoids is less prone than random

sampling to neglect entire regions of the data space during sampling. Nonetheless, in scenario where the data has simple distribution, such as uniform distribution, or only one cluster, the simpler uniform random sampling might be sufficient, eliminating the need for the added complexity of clustering approaches.

Greedy approaches are a broad category that implements intrinsically different strategies. In this work, we primarily focus on facility location and farthest point sampling (FPS). The strategy implemented by the facility location algorithm is determined by choice of the similarity function considered in the optimization problem in (3.11). In this work, we consider the similarity functions defined in (3.12) and (3.13), which are based on a sum of Euclidean distances, and on the Gaussian function, respectively. If we consider the similarity function in (3.12), it is possible to show that the facility location algorithm aims to minimize the sum of the squared distances between the points in the pool and their closest selected element, similar to k-medoids [MKSK16]. However, the fundamental difference is that facility location is a greedy technique, while k-medoids is based on a segmentation of the data points into clusters. FPS aims to maximize the coverage of the data spaces by sampling aiming to maximize inter-points distances. Fig. 3.3 shows the results of FPS sampling on two datasets of 1000 two-dimensional points in the unit square $[0, 1]^2$. The figure clearly shows that FPS does not tend to over or under-represent any region of the data space, independently of the distributions of the points in the datasets from which it samples. The comparison between FPS and the other two strategies considered in Fig. 3.3 suggests that, FPS may behave similarly to random sampling and k-medoids if the dataset from which it samples consists of uniformly distributed points. But it may lead to very different results in scenarios where the datasets include high-density clusters.

Furthermore, we note that the data dimensionality is another important factor to consider when selecting a sampling method based on the distances between data points. Working with high-dimensional data presents various challenges, a phenomenon known as the "curse of dimensionality", a term originally introduced by [Bel61]. For instance, the intuitive concept of distance breaks down in high dimensions. This may strongly affect the effectiveness and interpretability of the employed selection method. For more literature on the effects of the high dimensions in ML application, we refer the reader to [RNI10].

3.5. More on coresets

In this section we discuss two additional sampling approaches that are related to coresets: Data twinning and single-shot batch active learning.

Data twinning. Data twinning [VJ22], focuses on partitioning datasets into statistically similar twin sets. The primary goal of this approach is to ensure that the dataset splits maintain statistical similarity. The original motivation of data twinning to ensure unbiased model evaluations when one of the twin sets is used for training and the other (typically the smaller one) for testing.

The underlying principle that drives the implementation of the twinning approach is to minimize an energy associated to one of the two twin sets. Given a set of data points $\mathcal{D}_{\mathcal{X}} := \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ and a data budget $b \in \mathbb{N}$, the goal of the Twinning approach is to select a set $\mathcal{L}_{\mathcal{X}} \subset \mathcal{D}_{\mathcal{X}}$ of cardinality b solving the following problem

$$\min_{\substack{\tilde{\mathcal{L}}_{\mathcal{X}} \subset \mathcal{D}_{\mathcal{X}} \\ |\tilde{\mathcal{L}}_{\mathcal{X}}| = b}} \left(\frac{2}{nb} \sum_{\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}} \sum_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{L}}_{\mathcal{X}}} \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2 - \frac{1}{b^2} \sum_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{L}}_{\mathcal{X}}} \sum_{\hat{\boldsymbol{x}} \in \tilde{\mathcal{L}}_{\mathcal{X}}} \|\hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}}\|_2 \right) \tag{3.20}$$

By minimizing the quantity between the brackets, the Twinning algorithm aims to select a set $\mathcal{L}_{\mathcal{X}}$ with a similar distribution as the set $\mathcal{U}_{\mathcal{X}} := \{\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}} \mid \boldsymbol{x} \notin \mathcal{L}_{\mathcal{X}}\}$ and representative of the original dataset $\mathcal{D}_{\mathcal{X}}$. On the one hand, the first argument of the optimization objective in (3.20) ensures that the elements in $\mathcal{D}_{\mathcal{X}}$ are close to the selected elements in $\mathcal{L}_{\mathcal{X}}$, that is, the selected subset covers the dataset well in terms of proximity. On the other hand, the second component of the optimization objective promotes diversity within the selected subset by ensuring that the selected points are not too close to each other.

The input to the Twinning algorithm is a dataset $\mathcal{D}_{\mathcal{X}}$ that needs to be partitioned into twin sets, i.e., $\mathcal{D}_{\mathcal{X}} = \mathcal{L}_{\mathcal{X}} \sqcup \mathcal{U}_{\mathcal{X}}$. In addition, the algorithm also takes in input and integer value $r \in \mathbb{N}$ determining the ratio of the smaller twin set, e.g., if $r = 5$ than the smaller set consists of $\frac{100}{r}\% = 20\%$ of the data points in $\mathcal{D}_{\mathcal{X}}$. The output of the algorithm are two statistically similar twin sets, $\mathcal{L}_{\mathcal{X}}$ and $\mathcal{U}_{\mathcal{X}}$, that segment the dataset and such that the smaller of the two sets minimizes the quantity in (3.20). For more details on the Twinning algorithm see [VJ22].

The main characteristic of the Twinning approach for training data selection is its ability to sample a set of data points whose distribution closely reflects that of the entire dataset, similar to random sampling, while promoting diversity within the selected subset. Data twinning is a passive and model-agnostic sampling strategy and therefore of interest in this work. We use the Twinning python implementation from [VJ22] for our numerical experiments in Section 7.2.6.

Single-shot batch active learning Single-shot (or zero-shot) batch active learners is another class of data selection approaches that, despite being called "active learners", we think is part of the family of coresets strategies. More specifically, we classify them as passive data selection strategies. The authors of [VKL19], mention that single-shot batch active learners, similarly to coresets, aim to "select representative samples" and "there is no label information at the time the batch of queries is determined". Thus, they are label-agnostic approaches. Moreover, "The advantage of zero-shot active learning is that all queries can be computed ahead of time, and collected labels do not have to be fed into the active learners". That is, there is no active procedure involved in the selection process. Thus, we classify them as passive strategies.

Similarly to our work, the theoretical results motivating the effectiveness of single-shot active learners consist of upper bounds on the error performances of the prediction

methods. Accordingly, single-batch active learning approaches typically attempt to select a training set in order to minimize such bounds.

However, differently from what we consider here, the relevant work related to single-shot batch active learning selection is limited to zero-one loss functions [CWF+12], that is, to classification tasks, or it assumes a specific class of regression models, such as L_2 regularized kernel models [VKL19]. We are not aware of single-batch active learning approaches that are model-agnostic, that is, that do not assume a specific regression model design.

For more on state-of-the-art single-shot batch active learning, we refer the reader to [VKL19]. Notably, in the mentioned paper, the authors theoretically and empirically analyze three greedy single-batch active learning approaches for L_2 regularized kernel models: Minimum-maximum discrepancy, discrepancy and nuclear discrepancy. Note that the sampling strategies from [VKL19], are based on a greedy approach requiring the computation of the eigenvalues of a kernel matrix to be computed for each new candidate point at each iteration of the greedy selection procedure. This is computationally very demanding, which makes such approaches not suitable for large datasets.

4. Machine Learning for molecular property prediction

In this chapter we introduce ML for molecular property prediction, which provides the application scenario we consider for our experiments.

Predicting the properties of molecules is crucial for exploring the chemical compound spaces [vLMT20]. Accurate prediction models can significantly reduce the cost and time associated with experimental testing, enabling the rapid identification of promising compounds for various applications, including drug discovery, materials science, and chemical engineering. Traditional methods based on quantum mechanics, such as density functional theory [CF23], while reliable, are often computationally expensive and may not scale well with the increasing complexity and size of molecular datasets. Machine learning (ML) offers a powerful alternative, leveraging vast amounts of data to learn patterns and relationships that can generalize to unseen molecules.

Various datasets, molecular descriptors, and software toolkits have been developed and made publicly available to facilitate research and the application of ML technologies for molecular property prediction tasks. Datasets play a crucial role in training and benchmarking ML predictive models. Notable examples include the Quantum-Machine datasets [WRF$^+$18], such as QM7, QM8 and QM9. Together these datasets contain quantum mechanical calculations for hundreds of thousands of small organic molecules and are widely used benchmarks in the research community [HvL16, LSB18, SKSF$^+$17, SUG21, STR$^+$19]. Molecular descriptors represent molecular structures in a format that machine learning models can interpret. Molecular descriptors can be based on the molecules' 3D geometry or only on the 2D topological structure of the molecule. Software toolkits for quantum chemistry ML perform various tasks, including representing and comparing molecules and developing and training ML predictive models. Such software toolkits play a crucial role in speeding up the research development and employment of ML technologies in this application field. Key tools include: DeepChem [REW$^+$19], an open-source toolkit that integrates various machine learning models and datasets specifically designed for chemical and drug discovery applications, providing a comprehensive platform for researchers. RDKit [Lan12] is a collection of cheminformatics and machine learning tools used for manipulating chemical data, widely adopted in the research community. Mordred [MTKT18] is a Python library that can provide vector-valued molecular descriptors based on the molecules' topological information. Together, these datasets, descriptors, and toolkits form the backbone of research and development in machine learning for molecular property prediction, enabling significant advancements in various scientific and industrial domains. In what follow we introduce the Molecular descriptors, datasets, toolkits and regression models that we employ in this work.

4.1. Molecular descriptors

In this work, we focus on the task of molecular property prediction, that is, given a molecule we aim to use ML regression techniques to predict a label scalar value associated with it, e.g., atomization energies and HOMO-LUMO gaps. To achieve this, we need to represent molecules in a computer-readable way. More specifically, we need to represent molecules as (multidimensional) numerical objects that we can give into input to ML algorithms. Approaches for molecular representation are either based on the 3D geometry of the molecules or their 2D configuration, which is characterized by their topological structure. We describe the general principles of these molecular representations and introduce in detail those that we use in this work.

4.1.1. Topological descriptors

The topological structure of a molecule consists of the number of atoms, the atom types and how they are bonded together. One approach to encode the topological structure in a computer-readable chemical notations is to represent molecules as linear strings of symbols, similar to natural language. Notable examples of such notations are SMILES [Wei88] and SELFIES [KHN+20]. Such approaches represent molecules as strings providing information on the atom types and how they are bonded together, but do not generally provide information or their 3-Dimensional representation, e.g., atom coordinates or relative positions of the atoms. The topological information encoded in the string representations can then be processed to obtain various numerical values associated with each represented molecule. Such numerical values can be collected together in vector-valued descriptors that can can be used for ML tasks. There are various software toolkits and libraries that can be used to obtain vector-valued molecular descriptions from string representations of the molecules. Notable examples are Mordred [MTKT18] and PaDel-Descriptor [Yap10]. In this work we use SMILES strings and Mordred library. SMILES descriptors are strings encoding the topological structure of the molecules and Mordred is a library we use to obtain vector-valued molecular descriptors from SMILES strings. On the one hand, we choose SMILES as they are an established and widely employed tool to representing molecules. On the other hand, we choose the Mordred library as it provides a large variety of numerical values quantifying molecular characteristics related to the topological structure of the molecules. Moreover, Mordred computes such numerical values fast, it is easy to use and to install. Additionally, all direct dependent libraries in Mordred, except RDKit [Lan12] and NumPy [vdWCV11], are coded in Python, which we use as the programming language in this work, and, at the time of writing, is also the main programming language for ML applications.

Thus, the topological descriptors we use in this work are based on SMILES strings and the Mordred library (Fig. 4.1), which we now introduce more in detail.

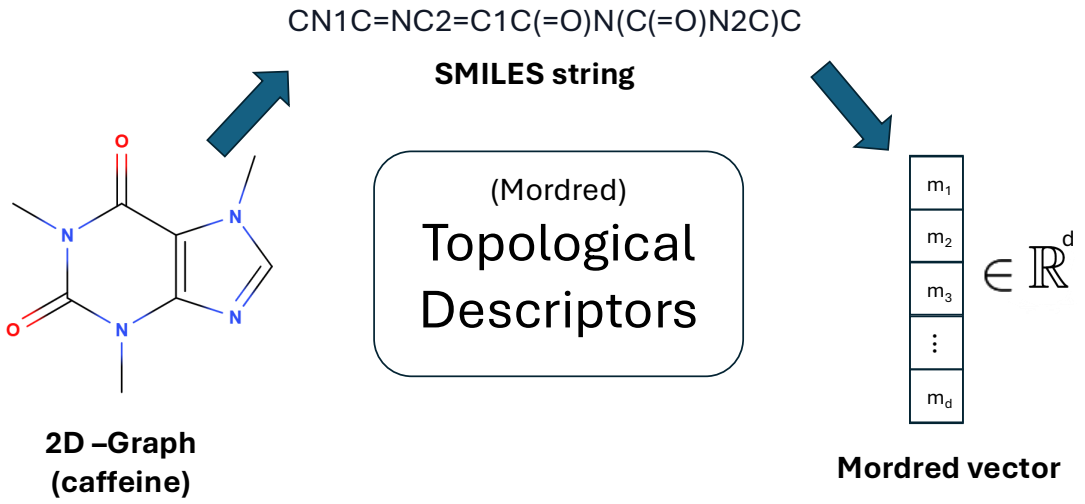


Figure 4.1.: Steps to compute Mordred vector-valued molecular descriptors from the 2D representation of the molecule. We considered caffeine for this example.

SMILES

Simplified Molecular Input Line System (SMILES) [Wei88] is a chemical notation language that encodes molecular structures including information related to atom types, bond types, branches, cyclic structures, disconnected structures and aromaticity. SMILES do not represent any particular three-dimensional arrangement of the atoms. We follow along [Wei88] and provide a description of the SMILES syntax.

Atoms are represented by their atomic symbols enclosed in square brackets. The second letter of two-character symbols must be entered in lowercase. Elements in the "organic subset", B, C, N, 0, P, S, F, C1, Br, and I, may be written without brackets if the number of attached hydrogens conforms to the lowest normal valence consistent with explicit bonds. For example, the SMILES string C represents methane (CH_4), N describes Ammonia (NH_3), and O describes water (H_20). Atoms in aromatic rings are specified by lowercase letters, e.g., normal carbon is represented by the letter C, and aromatic carbon is represented by c. Elements not in the organic subset must be represented with brackets. For instance, the symbol [Au] is for elemental gold. If there are attached hydrogens or formal charges, these are always specified inside brackets. The number of attached hydrogens is shown by the symbol H followed by an optional digit. Similarly, a formal charge is shown by one of the symbols + or -, followed by an optional digit. If unspecified, the number of attached hydrogens and charges is assumed to be zero for an atom inside the bracket. Examples are [H+] proton, [OH-] hydroxyl anion, [OH3+] hydronium cation, [Fe+2] iron cation, [NH4+] ammonium cation. The SMILES also

recognizes constructions of the form [Fe+++] as being synonymous with the form [Fe+3].

Single, double, triple, and aromatic bonds are represented by the symbols -, =, #, and :, respectively. Single and aromatic bonds may be, and usually are, omitted. Examples are CC for ethane (CH_3CH_3), C=C for ethylene (CH_2=CH_2), O=C=O for carbon dioxide (CO_2), and [H][H] molecular hydrogen (H_2).

A branch is specified by placing the symbol(s) for the branch between parentheses. The string in parentheses is placed directly after the symbol for the atom to which it is connected. If it is connected by a double or triple bond, the bond symbol immediately follows the left parenthesis. Examples are CC(O)C for 2-Propanol and CC(=O)C for 2-Propanone.

SMILES uses numbers to identify opening and closing atoms in ring structures. Different numbers are used for multiple rings. For example, in C1CCCCC1, the first carbon has a number "1" which connects by a single bond with the last carbon which also has a number "1". The resulting structure is cyclohexane. Bond symbols are used before the ring closure number for double, single or aromatic bonds.

Disconnected compounds are written as individual structures separated by a period. For instance, [Na+].[Cl-] is the SMILES string for sodium chloride.

Aromatic structures can be described by writing the atoms in the aromatic ring in lower case letters, for example, c1ccccc1C(=O)O for benzoic acid.

Mordred

Mordred [MTKT18] is a software application that can calculate 1613 numerical values describing various characteristics related to the 2D molecular structure and 213 related to the 3D configuration for a total of 1826 for each molecule. We use the Mordred library to compute descriptors from the 2D representation of the molecules encoded in the SMILES strings. Thus, we do not make use of the numerical values related to the 3D configuration, which, if calculated from the molecular 2D representation encoded in the SMILES, would output non-numerical values indicating an error has occurred during the computation. Moreover, to work with a more compact representation, we do not consider all the numerical values that have zero variance across the dataset.

Mordred is freely available and can be easily installed and used in the Linux command-line interface, as a web application, or as a Python package on all major platforms, such as Windows, Linux, and macOS.

It is important to mention that Mordred strongly relies on the RDKit library, which provides the necessary tools for handling molecular structures, including reading SMILES strings, parsing them into molecular objects, and preparing them for the calculation of the numerical values. The RDKit library plays a key role in the computer-based analysis of molecular objects and is a key tool for the development and deployment of ML technologies in the field of chemistry.

4.1.2. Geometrical descriptors

Geometrical descriptors represent molecules using information about atom types and their position in the Cartesian space, possibly in addition to the information provided by the 2D representation of the molecules, such as bonds and bond types between atoms. The development of molecular descriptors based on the geometry of the molecules is a highly studied area of research. Providing a comprehensive review of the geometrical descriptors in the literature is outside the scope of this work. Nonetheless, we provide an overview of such descriptors and point the readers to [HvL21] for more a more accurate description.

Following along [HvL21], we classify geometrical descriptors into two main categories: discrete and continuous. Discrete descriptors represent each molecule as a collection of points (the atoms) in the Cartesian space. Notable examples include the Coulomb Matrix (CM) [RTMvL12], which is based exclusively on atom types and their location in the Cartesian space, and the Bag of Bonds (BoB) [HBR$^+$15], which aims to improve the CM representation by taking into account information related to bonds and bond types. Various generalizations of these representations exist, such as BAML [HvL16], which takes into account not only the bonds but also their angle and higher-order interactions between atoms, and many-body dispersion (MBD) based representations involving two and three body terms [PTM18]. The main advantage of discrete molecular descriptors is that they can be computed fast. Nonetheless, they have some limitations. For instance, they often depend on some indexing of the atoms, which is an arbitrary choice that may affect the quality of the resulting descriptors and that may require artificial methods to ensure atom indexing invariance. Continuous descriptors represent molecules as continuous functions over the molecular environment in the Cartesian space and are derived from atomic properties or spectra. Continuous descriptors can also capture long range interaction between atoms, contrary to the discrete mainly providing information related to the atoms' local environment. Notable examples are the Smooth Overlap of Atomic Potentials (SOAP) [BKC13] and Atomic Spectrum of London Axilrod-Teller-Muto (aSLATM) [HvL20]. Continuous descriptors address the indexing problem related to discrete ones and typically provide more effective representations. However, they are often more expensive to compute.

Note that, while geometrical descriptors have shown to be very effective for molecular property prediction tasks, they may not always be applied in ML applications. For instance, if we consider a scenario in which we aim to predict the DFT-calculated atomization energy of the molecules in their equilibrium geometry, to obtain the geometrical descriptors, we would need the knowledge of the equilibrium geometry obtained via DFT calculations that already yield the target quantity [DBB$^+$21]. That is, to obtain the descriptors required to set up the regression problem, we should perform calculations that would already provide us the energy values in which we are interested, thus removing the need of solving a regression problem. Nonetheless, the predictive power of geometrical descriptors is evident and suggests that they may be able to achieve other useful goals. Therefore, we also consider them in this work. In particular, we consider the Coulomb Matrix (CM) as it has been considered as an essential baseline for the

interpretation, analysis, and further development of various machine learning tasks in the context of quantum chemistry [HvL21]. Next, we provide a detailed description of the CM descriptor.

Coulomb Matrix

The Coulomb Matrix (CM) descriptor [RTMvL12] is a representation of molecular structures used in computational chemistry and machine learning. It encodes the geometrical and atomic information of a molecule into a numerical matrix form. Given a molecule consisting of a collection of $n_a \in \mathbb{N}$ atoms, each associated with a unique index in $\{1, \dots, n_a\}$, the CM is and object $\boldsymbol{C} \in \mathbb{R}^{n_a \times n_a}$ defined as

$$\boldsymbol{C}_{i,j} = \begin{cases} \frac{1}{2} z_i^{2.4} & \text{if } i = j \\ \frac{z_i z_j}{\|\boldsymbol{r}_i - \boldsymbol{r}_j\|_2} & \text{if } i \neq j \end{cases} \tag{4.1}$$

where z_i is the nuclear charge of the i-th atom, $\boldsymbol{r}_i \in \mathbb{R}^3$ is the coordinates vector describing its location and $i, j \in \{1, \dots, n_a\}$. The off-diagonal entries of the matrix represent the Coulomb repulsion between the i-th and j-th atoms, whereas the elements on the diagonal contain a polynomial approximation of atomic energies based on the nuclear charge [Eng14]. The Coulomb matrix is invariant to translations and rotations of the molecule, which is essential for ensuring that the descriptor accurately reflects the molecule's intrinsic properties and is independent of its orientation in space. Note that such invariance to translations and rotations is enforced by considering pairwise inter-atomic distances $\|\boldsymbol{r}_i - \boldsymbol{r}_j\|_2$. The CM descriptors are not permutation invariance: if we consider a different order for the atoms we would obtain a different descriptor. Various approaches have been developed to preserve permutation invariance. For instance, it is possible to preserve permutation invariance by considering the eigenvalues of the CM, although this approach sacrifices uniqueness [Mou12, RTMvL12]. Another method attempting to address the permutation invariance issue involves using sets of randomly permuted CMs [MRG$^+$13], while a third option is to sort by norms of rows [HMB$^+$13].

Geometrical descriptors not including atom types

For some applications, molecular descriptors may not need to include information about atom types, as for the CM, and can be solely based on the positions of the atoms. One example of such applications is force-field predictions for a molecule along its thermodynamic trajectory, where the objective is to predict the inter-atomic forces acting on different conformers of the same molecule. Conformers are molecules composed of the same atoms but with different relative positions between the atoms. One molecular representation we can use for such tasks is the one proposed by [CTS$^+$17], consisting, for each molecule with n_a atoms, of a matrix $\boldsymbol{M} \in \mathbb{R}^{n_a \times n_a}$ defined as follows

$$\boldsymbol{M}_{ij} = \begin{cases} \|\boldsymbol{r}_i - \boldsymbol{r}_j\|_2^{-1} & \text{if } i > j \\ 0 & \text{if } i \leq j \end{cases} \tag{4.2}$$

where $\boldsymbol{r}_i \in \mathbb{R}^3$ is the location of the i-th atom. Such representation is invariant to translations and rotations of the molecule but still not invariant to permutations of the atoms.

4.2. Quantum chemistry datasets

This subsection introduces in detail the datasets we use in this work. In particular, we introduce the QM datasets, the rMD17 and the ZINC dataset. Additionally, we also describe the data preprocessing procedures we apply.

QM7

QM7 [BR09, RTMvL12] is a benchmark dataset in quantum chemistry consisting of 7165 small organic molecules with up to 23 atoms including 7 heavy atoms: C, N, O and S. It includes information such as the Cartesian coordinates of the atoms in each molecule and the atomization energy of the molecules. We use QM7 for a regression task, where the feature vector used to describe a molecule is the Coulomb matrix [RTMvL12]. Consequently, each molecule in the QM7 is represented as an element in $\mathbb{R}^{529}$, and the label value to predict is the atomization energy, a scalar value describing amount of energy in electronvolt (eV) required to completely separate all the atoms in a molecule into individual gas-phase atoms.

QM8

QM8 [RvDBR12, RHTvL15] is a curated collection of 21,786 organic molecules with up to 8 heavy atoms (C, N, O, and F). For each of the molecules it provides the SMILES representation [Wei88] together with various molecular properties, such as the lowest two singlet transition energies and their oscillator strength. These molecular properties have been computed considering different approaches. In this study we consider those values computed with hybrid exchange correlation functional PBE0. We consider the 21716 molecules in the QM8 with unique smile representation. Next, to generate the molecular descriptors we employ Mordred [MTKT18], a publicly available library that exploits the molecules' topological information encoded in the SMILES strings to provide 1826 physical and chemical features. We set to zero all the descriptor values that could not be computed by the Mordred library and to work with a more compact representation, we remove 530 features for which the values across the dataset have zero variance. Thus, each molecule in QM8 is represented by a vector in $\mathbb{R}^{1296}$. Furthermore, we normalize the features provided by the Mordred library, to scale them independently in the interval (0, 1). The label value to predict in the regression task is the lowest singlet transition energy (E1), measured in eV, describing the energy difference between the ground state and the lowest excited state in a molecule. It is an important property in understanding the electronic behavior of molecules.

QM9

QM9 [RvDBR12, RDRvL14] is a publicly available quantum chemistry dataset containing the properties of 133,885 small organic molecules with up to nine heavy atoms (C, N, O, F). QM9 is frequently used for developing and testing machine learning models for predicting molecular properties and for exploring the chemical space [FHH$^+$17, RvL17, PSTM18]. QM9 contains the SMILES representation [Wei88] of the relaxed molecules, as well as their geometric configurations and 19 physical and chemical properties. In order to ensure the integrity of the dataset, we exclude all 3054 molecules that did not pass the consistency test proposed by [RDRvL14]. Additionally, we remove the 612 compounds that could not be interpreted by the RDKit package [Lan12]. Furthermore, in order to ensure the uniqueness of data points, we exclude 17 molecules with SMILES representations identical to those of other molecules in the dataset. Following this preprocessing procedure, we obtain a smaller version of the QM9 dataset consisting of 130202 molecules. The molecular representation we employ is based on the Mordred [MTKT18] library, as for the QM8 dataset. We set to 0 all the features that could not be computed by Mordred. Next, remove 519 features for which the values across the dataset have zero variance. Thus, each molecule in QM9 dataset is represented by a vector in $\mathbb{R}^{1307}$. When specified in the experiments, we also normalize the computed features to scale them independently in the interval (0, 1). The label value to predict is the HOMO-LUMO energy, measured in eV, describing the difference between the highest occupied (HOMO) and the lowest unoccupied (LUMO) molecular orbital energies. It is a useful quantity for examining the molecules kinetic stability.

Revised MD17

The revised MD17 [CvL20a, CvL20b] (rMD17) is an updated version of the molecular dynamics dataset (MD17) [UCS$^+$21] commonly used for developing and testing machine learning models for force-field prediction [SKSF$^+$17, GSS$^+$22, LWL$^+$22]. The rMD17 consists of temporal trajectories of various small organic molecules of varying sizes and complexity. The dataset provides information on the Cartesian coordinates, atomic charges, and per-atom forces, that is, the force-field, of each molecule at each time step of the molecules' trajectories. The per-atom forces are provided in $\frac{\text{kcal}}{\text{mol}\times\text{Å}}$, where Å is angstrom (10^{-10} meter). We use the rMD17 for a multivariate regression task. Using the atoms coordinates we aim at predicting the per atom forces of the molecules over the course of their trajectories. We consider the molecular descriptors described in (4.2). In this work, we study the trajectories of the Benzene with 9 atoms, Uracil and Malonaldehyde each consisting of 12 atoms. The trajectories of each of the considered molecules consists of 100000 time steps.

ZINC

The ZINC dataset [GBWD$^+$18] consists of approximately 250,000 molecules with up to 38 heavy atoms selected from the ZINC database [SI15], which contains over 120 million

purchasable organic molecules. To reduce the computational effort of our analysis, we follow along [DJL+23] and consider a subset of the ZINC dataset. Specifically, we use a subset of ZINC consisting of 24000 molecules that we select uniformly at random. The molecular representation we employ is based on the Mordred [MTKT18] library, as for the QM8 and QM9 datasets. We normalize the features provided by the Mordred library, to scale them independently in the interval (0, 1). We set to zero all the descriptor values that Mordred could not compute, remove the features for which the values across the dataset have zero variance and apply PCA to reduce the dimension of the feature vectors to 100. The label value to predict is the water-octanol partition coefficient (LogP), describing the molecules' solubility.

4.2.1. Underlying characteristics of the datasets

In this work, we study datasets specifically developed for molecular property and force field prediction. Given the specificity of these datasets to the applications we consider in this work, it is important to reflect on their underlying characteristics, as they may impact the results of an empirical analysis.

First, we note that while the space of molecules may be very large, with more than 10^{60} molecules [KE04], it is intrinsically discrete. Therefore, when we represent molecules in the QM and ZINC datasets with vector-valued descriptors in $\mathbb{R}^d$, $d \in \mathbb{N}$, it is important to take into account the fact that each molecule is uniquely associated with one real-valued vector, but not each real-valued vector represents a physically valid molecule. The fact that the QM and ZINC datasets have such a "discrete" structure may limit the applicability of some selection strategies. For example, in this situation, the centroids chosen with the k-means approach (described in Algorithm 1) may not be suitable as a training set. This is because centroids can potentially be at any locations in $\mathbb{R}^d$. In particular, they could be at locations that do not correspond to any molecule in the available datasets or to any valid molecule at all.

Note that, the rMD17 provides a different scenario. In this dataset each trajectory is associated with a specific molecule, and the atoms of the molecule change their relative positions at different time steps of the trajectory. While physical constraints may limit the movements of the atoms, such movements are continuous. That is, if we consider a specific molecule and, at each step of the trajectory, we represent a specific atom's location with a three-dimensional vector $\boldsymbol{r} = [r_1, r_2, r_3]$, the entries of such vector may vary continuously in some bounded subspace of $\mathbb{R}$. That is $r_i \in \mathcal{S}_i \subset \mathbb{R}$ for each $i = 1, 2, 3$.

It is important to note that all the datasets we consider arise from numerical simulations. These simulations are based on a theoretical framework from which numerical models describing the phenomena of interest are derived. Therefore, while the datasets we consider represent very complex quantum mechanics phenomena, there are underlying numerical models that produce the molecular configurations and associated properties (labels). These numerical models are explicitly known. Thus, we can rely on some consistency in the labeling process.

The numerical methods employed to compute the labels may be subject to errors in approximating the true solution. Nonetheless, we can rely on the fact that all the data

points' labels are computed using some numerical method which is explicitly known and for which an approximation error and biases are likely quantifiable. Such a reliability of the labeling process is not always guaranteed in ML applications. There may be applications in which labeling procedures may be affected by external factors on which we do not have any control. Examples of such labeling procedures are laboratory experiments. Labeling procedures based on laboratory experiments may be intrinsically less consistent than numerical simulations, as they may be affected by several hidden variables over which scientists do not have control. On the contrary, numerical experiments follow clearly defined mathematical rules where all relevant variables used to compute the solutions are known.

Simulated datasets arising from numerical experiments may consist of subcollections of points, each generated using a distinct simulation method or the same simulation approach but with different parameter settings. For instance, molecular datasets, such as the QM datasets, are computed using numerical methods based on density functional theory (DFT) [KBP96]. DFT is a quantum-mechanical modelling method for the energetic structures of many body systems, such as atoms and molecules. DFT models the total energy of the system as a functional of the electron density function. Different functionals (e.g., PBE, B3LYP) offer varying levels of accuracy and computational requirements. Thus, the same molecule can be associated with similar but different energy values depending on the accuracy level considered. Thus, it may be possible that molecules in a simulated dataset are labeled according to different levels of accuracy, that is, using different numerical approaches.

In this work, we perform experiments considering scenarios where molecules within each dataset have been labeled using the same level of accuracy. Recall that, we consider a scenario in which we want to perform label-agnostic training data selection. That is, we want to select training data exclusively based on their locations in the feature space. In such a scenario, we must rely on the assumption that the labeling procedure is reliable or at least consistent so that we can assume the existence of ground truth or, more formally, the existence of a unique underlying map connecting the features with the labels. The goal of the regression task is to recover such a map from the training data. If there are data points that are mislabeled or if label values arise from different procedures that may associate the same data point to different labels, the assumption of a unique underlying map connecting features with the labels would be violated. This may affect the effectiveness of the data selection procedures we employ. For instance, even if we are able to select training data points at the optimal locations, assuming that such optimal locations are well-defined and exist, if they are mislabeled, we would not have the guarantee that the optimal selected points are of value for the training procedure. On the contrary, the presence of mislabeled points may pollute the information in the selected training set, deteriorating the performances of the resulting trained model.

4.3. Regression models

In this work we use ML regression models that have been utilized in previous works for molecular property and force field prediction tasks. Specifically, we consider kernel ridge regression with the Gaussian kernel (KRR) [STR+19, DBB+21] and feed forward neural networks (FNNs) [PMS+20] for molecular property prediction and the gradient-domain machine learning (GDML) method [CTS+17] for force field prediction. KRR and FNN are of interest to us because of their Lipschitz continuity, which is a required property to validate the theoretical results we propose in later chapters. Next, we formally introduce KRR, FNN and GDML regression models.

4.3.1. Kernel Ridge Regression (KRR)

Kernel regression models with the Gaussian kernel, is a class of regression approaches successfully employed in various applications, such as molecular and material sciences [DBB+21], and robotics [DFR15]. Kernel ridge regression is an ML technique that combines the concepts of kernel methods and ridge regression to perform non-parametric, regularized regression [DBB+21]. In this work, we use a Gaussian kernel function. Given two data points $\boldsymbol{x}_q, \boldsymbol{x}_r \in \mathbb{R}^d$, the Gaussian kernel is defined as follows:

$$k(\boldsymbol{x}_q, \boldsymbol{x}_r) := e^{-\gamma \|\boldsymbol{x}_q - \boldsymbol{x}_r\|_2^2}, \tag{4.3}$$

where $\gamma \in \mathbb{R}^+$ is a kernel hyperparameter to be selected through an optimization process. Provided a training set $\mathcal{L} = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^b$ and set of weights $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_b]^T \in \mathbb{R}^b$, the predicted label value $m_{\mathcal{L},\boldsymbol{\alpha}}(\boldsymbol{x}) \in \mathbb{R}$ associated with a data location $\boldsymbol{x} \in \mathbb{R}^d$ of a new data point is defined as follows

$$m_{\mathcal{L},\boldsymbol{\alpha}}(\boldsymbol{x}) := \sum_{j=1}^{b} \alpha_j k(\boldsymbol{x}, \boldsymbol{x}_j). \tag{4.4}$$

The scalar $m_{\mathcal{L},\boldsymbol{\alpha}}(\boldsymbol{x})$ is the label predicted by the KRR method associated with the training data locations $\{\boldsymbol{x}_j\}_{j=1}^b$ and weights $\boldsymbol{\alpha}$. The weights of a trained KRR model are learned by solving the following minimization problem

$$\boldsymbol{\alpha}_{\mathcal{L}} = \arg\min_{\bar{\boldsymbol{\alpha}}} \sum_{j=1}^{b} (m_{\mathcal{L},\bar{\boldsymbol{\alpha}}}(\boldsymbol{x}_j) - y_j)^2 + \lambda \bar{\boldsymbol{\alpha}}^T \boldsymbol{K}_{\mathcal{L}}\, \bar{\boldsymbol{\alpha}}. \tag{4.5}$$

Here, $\boldsymbol{K}_{\mathcal{L}} \in \mathbb{R}^{b,b}$ is the kernel matrix, i.e., $\boldsymbol{K}_{\mathcal{L}}(q,r) = k(\boldsymbol{x}_q, \boldsymbol{x}_r)$, and the parameter $\lambda \in \mathbb{R}^+$ is the so-called regularization parameter that addresses eventual ill-conditioning problems of the matrix $\boldsymbol{K}_{\mathcal{L}}$. The closed-form solution to the minimization problem in (4.5) is given by

$$\boldsymbol{\alpha}_{\mathcal{L}} = (\boldsymbol{K}_{\mathcal{L}} + \lambda \boldsymbol{I}_b)^{-1} \boldsymbol{y} \tag{4.6}$$

where $\boldsymbol{y} = [y_1, y_2, \ldots, y_b]^T$ and $\boldsymbol{I}_b$ the identity matrix of size b.

In the experiments we perform in Chapter 7 we analyze how the performance of a given regression model vary as the training set used to learn the weights changes. Given a pool of unlabeled points, we consider training sets of various sizes and selected according to different sampling strategies. To ensure that change in performance are only due to changes of the training set, we keep the hyperparameters of the model fixed. The choice of the KKR hyperparameters γ and λ is the result of the following optimization process: first we perform a cross-validation grid search to find the best hyperparameter for each training set size using subsets obtained by random sampling. Next, the average of the best parameters pair for each training set size is used to build the final model. Note that, we do not use an optimal set of hyperparameters for each selection strategy and training set size. This decision is made because we aim to analyze the qualitative behavior of a fixed model, where the only variable affecting the quality of the predictions is the selected training set.

To address the question of the Lipschitz continuity of the KRR with the Gaussian kernel we have the following lemma:

Lemma 4.1 *The regression function provided by the trained kernel ridge regression algorithm with the Gaussian kernel is Lipschitz continuous with respect to the absolute difference of the predictions.*

Proof. Consider the training set $\mathcal{L} = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{b}$ and set of learned weights $\boldsymbol{\alpha}_{\mathcal{L}} := [\alpha_1, \alpha_2, \ldots, \alpha_b]^T \in \mathbb{R}^b$ obtained by training the KRR on $\mathcal{L}$. Then, given $\boldsymbol{x} \in \mathbb{R}^d$ the predicted label $y(\boldsymbol{x}) \in \mathbb{R}$ provided the KRR approximation function can be computed as follows:

$$y(\boldsymbol{x}) = \sum_{j=1}^{b} \alpha_j k(\boldsymbol{x}, \boldsymbol{x}_j) = \boldsymbol{\alpha}_{\mathcal{L}}^T \boldsymbol{k}_{\boldsymbol{x}}, \tag{4.7}$$

where $k(\boldsymbol{x}, \boldsymbol{x}_j) := e^{-\gamma \|\boldsymbol{x} - \boldsymbol{x}_j\|_2^2}$, and $\boldsymbol{k}_{\boldsymbol{x}} := [k(\boldsymbol{x}, \boldsymbol{x}_1), k(\boldsymbol{x}, \boldsymbol{x}_2), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_b)]^T \in \mathbb{R}^b$. Next, considering $\tilde{\boldsymbol{x}}, \hat{\boldsymbol{x}} \in \mathcal{X}$, we have

$$\begin{aligned}
|y(\tilde{\boldsymbol{x}}) - y(\hat{\boldsymbol{x}})| &\leq |\boldsymbol{\alpha}_{\mathcal{L}}^T \boldsymbol{k}_{\tilde{\boldsymbol{x}}} - \boldsymbol{\alpha}_{\mathcal{L}}^T \boldsymbol{k}_{\hat{\boldsymbol{x}}}| \\
&\leq \|\boldsymbol{\alpha}_{\mathcal{L}}\|_2 \|\boldsymbol{k}_{\tilde{\boldsymbol{x}}} - \boldsymbol{k}_{\hat{\boldsymbol{x}}}\|_2 \\
&= \|\boldsymbol{\alpha}_{\mathcal{L}}\|_2 \sqrt{\sum_{j=1}^{b} \left(e^{-\gamma \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_j\|_2^2} - e^{-\gamma \|\hat{\boldsymbol{x}} - \boldsymbol{x}_j\|_2^2} \right)^2} \\
&\leq \|\boldsymbol{\alpha}_{\mathcal{L}}\|_2 \sqrt{b} \lambda_k \|\tilde{\boldsymbol{x}} - \hat{\boldsymbol{x}}\|_2,
\end{aligned}$$

where λ_k is the Lipschitz constant of the function $e^{-\gamma r^2}$, $r \in \mathbb{R}^+$. □

4.3.2. Feed Forward Neural Networks (FNNs)

Feed-forward neural networks [GBC16] (FNNs) are among the simplest deep neural networks. Given $\boldsymbol{x} \in \mathcal{X}$ the predicted label, $y(\boldsymbol{x})$, provided by a FNN, with $l \in \mathbb{N}$ layers, can be expressed as the output of a composition of functions, that is,

$$y(\boldsymbol{x}) := \phi_l \circ \sigma_l \circ \phi_{l-1} \circ \sigma_{l-1} \circ \cdots \circ \phi_1(\boldsymbol{x}), \tag{4.8}$$

where the ϕ_i are affine linear functions or pooling operations and the σ_i are nonlinear activation functions. Following along [PMS+20], we set $l = 3$, consider only ReLu activation functions and define

$$\phi_i(\boldsymbol{x}) = \boldsymbol{W}_i \boldsymbol{x} + \boldsymbol{b}_i \tag{4.9}$$

where the weight matrices $\boldsymbol{W}_i$ and the biases $\boldsymbol{b}_i$ are learned by minimizing the mean absolute error or the mean squared error between the true and predicted labels of the data points in the training set, depending on the application. To learn the weight matrices $\boldsymbol{W}_i$ and the biases $\boldsymbol{b}_i$, we use the Pytorch [PGM+19] Adam optimizer with a learning rate of 0.001, betas range (0.9, 0.999) and weight decay 0.001. We use a batch size of 516, independently on the dataset, and consider 1000 or 250 epochs, depending on the computational cost required for the training procedure, which is determined by the dataset size and the dimension of the data descriptors employed. Moreover, to ensure that the changes in performance are exclusively due to the changes of the training set, the training procedure of the FNN is always initialized with the same set of random weights. The Lipschitz continuity of FNN and other more advanced neural networks has been already shown in the literature [SV18, GFPC20].

4.3.3. Gradient-Domain Machine Learning (GDML)

For force-field prediction tasks we use the gradient-domain machine learning (GDML) method developed in [CTS+17], which we follow to introduce the main idea behind GDML, briefly. See [CTS+17] for further information on this regression technique. In force-field prediction tasks the label values are multidimensional. GDML aims to learn the functional relationship

$$\hat{\boldsymbol{f}}_F : \boldsymbol{x}_i \to \boldsymbol{F}_i$$

between the coordinates $\boldsymbol{x}_i \in \mathbb{R}^{3n_a}$ of the atoms in a given molecule and the per-atom forces $\boldsymbol{F}_i \in \mathbb{R}^{3n_a}$. The GDML method relies on a kernel ridge regression technique with Matern kernel functions to learn such function relationships from data. Given a training dataset $\mathcal{L} = \{(\boldsymbol{x}_j, \boldsymbol{F}_j)\}_{j=1}^{b}$, the estimation of the function $\hat{\boldsymbol{f}}_F$ on a data point $\boldsymbol{x}$ representing the per atom location in the Cartesian space takes the form

$$\hat{\boldsymbol{f}}_F(\boldsymbol{x}) := \sum_{j=1}^{b} \sum_{i=1}^{3n_a} (\alpha_{j,i}) \frac{\partial}{\partial x_{j,i}} \nabla k(\boldsymbol{x}, \boldsymbol{x}_j)$$

where $\boldsymbol{x}_j = [x_{j,1}, \ldots, x_{j,3n_a}]$. The parameters $\boldsymbol{\alpha} \in \mathbb{R}^{b \times 3n_a}$ are learned by solving a ridge regression type optimization problem, and the function $k : \mathbb{R}^{3n_a} \times \mathbb{R}^{3n_a} \to \mathbb{R}$ is the employed Matern kernel function. Given that GDML is based on a differentiable Matern kernels we expect its predictions to exhibit some regularity. However, analyzing the Lipschitz continuity of this regression model is beyond the scope of this work.

4.4. Metrics for evaluating model performance

In this section we introduce and define the metrics we consider to evaluate the performances of the studied regression models. We consider evaluation metrics for univariate and multivariate tasks, that is, for regression tasks with scalar and vector-valued labels, respectively.

Before providing the formulas describing the evaluation metrics we consider in this work, it is helpful to have an intuitive understanding of the concept of prediction quality. We characterize prediction quality by breaking it down into two distinct concepts: robustness and average accuracy.

Robustness refers to a model's ability to maintain low prediction error over the entire data space of interest, including regions of the data space that may not be represented in the training set. One way to measure robustness is through the maximum absolute prediction error, that is, the largest absolute difference between one label value and the related prediction provided by the regression model. A low maximum prediction error implies that the predictions of the considered model are accurate over the entire data space and, therefore, robust and reliable.

Average accuracy in the context of regression models typically refers to how close the model's predictions are to the actual values on average across the entire dataset. One way to measure average accuracy is to consider the mean of the predictions absolute errors in the data space on interests. Note that, models with low average prediction error may not be robust and still perform very poorly on some regions of the data; that is, they may still lead to a large maximum prediction error[CG24].

4.4.1. Univariate regression

We consider three metrics to evaluate the performance of the ML methods used for regression tasks with scalar label values: Maximum Absolute Error (MAXAE), Mean Absolute Error (MAE) and the root mean squared error (RMSE). The MAXAE is the maximum absolute difference between the true target values $\{y_i\}_{i=1}^n$ and the predicted values $\{\tilde{y}_i\}_{i=1}^n$, that is,

$$\text{MAXAE} := \max_{1\leq i\leq n} |y_i - \tilde{y}_i|, \tag{4.10}$$

where n is the number of unlabeled data points in the analyzed data pool. The MAE is calculated by averaging the absolute differences between the predicted values and the true target values, that is,

$$\text{MAE} := \frac{1}{n}\sum_{i=1}^{n} |y_i - \tilde{y}_i|. \tag{4.11}$$

The RMSE is calculated by taking the square root of the average absolute square differences between the predicted values and the true target values and, that is,

$$\text{RMSE} := \sqrt{\frac{1}{n}\sum_{i=1}^{n} |y_i - \tilde{y}_i|^2}. \tag{4.12}$$

The RMSE penalizes large errors more than the MAE, those providing insight on the robustness of the predictions.

4.4.2. Multivariate regression

We consider three metrics to evaluate the performance of the ML method used for the force-field regression tasks with vector-valued label values: the atom-wise maximum error over the predicted forces (MAXAE_F), the molecule-wise maximum MAE (MAXMAE_F) and the mean absolute error (MAE_F). The MAXAE_F is the maximum absolute difference between the entries of the true target values $\{\boldsymbol{F}_i\}_{i=1}^n \subset \mathbb{R}^{3n_a}$, describing the per-atom forces of the analyzed molecule with n_a atoms, and those of the predicted values $\{\tilde{\boldsymbol{F}}_i\}_{i=1}^n \subset \mathbb{R}^{3n_a}$, that is,

$$\mathrm{MAXAE}_F := \max_{1\leq i\leq n} \max_{1\leq j\leq 3n_a} |F_{i,j} - \tilde{F}_{i,j}|, \tag{4.13}$$

where $\boldsymbol{F}_i = [F_{i,1}, F_{i,2}, \ldots, F_{i,3n_a}]$. The MAXMAE_F is defined as follows:

$$\mathrm{MAXMAE}_F := \max_{1\leq i\leq n} \left(\frac{1}{3n_a} \sum_{j=1}^{3n_a} |F_{i,j} - \tilde{F}_{i,j}| \right). \tag{4.14}$$

Both, the MAXAE_F and the MAXMAE_F, are quantities we introduce to evaluate the robustness of the predictions of a given regression model for the force-field prediction task. The MAXAE_F provides an atom-wise information on the worst case prediction error while the MAXMAE_F focuses on the molecule-wise worst case error. To evaluate the average performance of a multivariate regression model we consider the MAE_F, that is, the average absolute differences between the predicted values and the true target values:

$$\mathrm{MAE}_F := \frac{1}{3nn_a} \sum_{i=1}^{n} \sum_{j=1}^{3n_a} |F_{i,j} - \tilde{F}_{i,j}|. \tag{4.15}$$

5. On minimizing the training set fill distance

In the previous chapter we saw that one of the key applications of Machine Learning (ML) regression is to label pools of unlabeled data points for which the existing labeling methods are too expensive in terms of computation, time, or money. To achieve this, a subset of the unlabeled pool is labeled and used to train a regression model. The trained model is then employed to get fast predictions for the labels of points not considered during training. However, the effectiveness of regression models is strongly dependent on the training data used for learning the regression parameters. Therefore, the selection of a suitable training set is crucial for the quality of the predictions of the model. Our focus is on selecting data points that result in a good performance for a variety of regression models. This ansatz ensures that the labeling effort is not wasted on subsets that may only be useful for specific learning models, classes of models, or prediction tasks.

In this chapter we are interested in a data selection strategy that maximizes model robustness in the low data budget regime. That is, we have few points we can choose for training and want to select them to minimize the maximum prediction error of a given regression model over the remaining unlabeled data points in the pool. Note that the maximum prediction error is a helpful evaluation metric in various applications, such as those related to material science and chemistry, where the average error provides an incomplete evaluation of the predictions of a model [SBG+20]. The authors of [VSH21], mention the maximum prediction error among those metrics that are "key to comparing the performance of different models and thus for developing guidelines and best practices for the successful application of machine learning in chemistry". In [ZHSK22] and [HvLKB23], the maximum error is considered to evaluate the prediction quality of machine learning models trained to study and explore the chemical or conformational spaces. Moreover, the authors of [GBP+20] use the maximum error to evaluate the prediction quality of Nuclear Magnetic Resonance spectroscopy parameters for 3-dimensional chemical structures.

Specifically, in this chapter we investigate theoretically the impact of minimizing training set fill distance through Farthest Point Sampling (FPS) for ML regression. We show that minimizing the fill distance of the training set can reduce the maximum expected prediction error of Lipschitz continuous regression models. Farthest point sampling (FPS) [ELPZ94] is a well-established passive and model-agnostic sampling strategy for training set selection, as introduced in Section 3.3.2. FPS was already employed in various application fields, such as image classification [SS18] or chemical and material science [DBB+21]. FPS provides suboptimal solutions to the k-center problem [HP11], which involves selecting a subset of k points from a given set by

minimizing the fill distance of the selected set, that is, the maximum over the distances between any point in the remaining set and selected point nearest to it.

For classification tasks, it was shown that minimizing the fill distance of the training set reduces the average prediction error of Lipschitz-continuous classification models with soft-max output layer and bounded error function [SS18]. Unfortunately, these results do not carry over to regression tasks, even for simpler Lipschitz-continuous approaches, such as kernel ridge regression with the Gaussian kernel (KRR) or feed-forward neural networks (FNNs). In particular, in Chapter 7 we provide examples where reducing the training set fill distance does not significantly lower the average prediction error compared to random selection. For regression, the concept of fill distance was already used to obtain, under several assumptions, error bounds for the prediction error of specific model classes, such as kernel methods based on Gaussian Process regression [WBG21]. However, we are interested in error bounds that are model-agnostic, that is, we do not assume any specific framework or methodology for the regression model learning process. The use of FPS in regression has been studied in various works [YK10, WLH19, DBB$^+$21], where it was also argued that passive sampling strategies such as FPS are more effective than active learning in terms of data efficiency and prediction accuracy. However, these works lack theoretical motivation, relying only on domain knowledge or heuristics.

We derive an upper bound for the maximum expected prediction error of Lipschitz continuous regression models that is linearly dependent on the training set fill distance. Our theoretical analysis offers results, which set it apart from previous works. Specifically, we extend the theoretical work of [SS18] from classification to regression, demonstrating that reducing training set fill distance lowers the maximum prediction error of the regression model. Moreover, contrary to [YK10] and [WLH19], who studied the advantages of using FPS for regression tasks, our findings are supported by mathematical results providing theoretical motivation for what we show empirically, in this chapter with a simple example, and in Chapter 7 with an extensive empirical analysis. We emphasize that, according to our knowledge, prior research did not detect the relationship between reducing the fill distance of the training set using FPS and decreasing the maximum prediction error of Lipschitz continuous regression models, neither theoretically nor empirically. In addition, we provide further theoretical examination to show supplementary advantages of selecting training sets with the FPS for kernel regression models using a Gaussian kernel. Specifically, our findings indicate that employing FPS for selecting training sets enhances the stability of this particular category of models.

5.1. Problem definition

We now formally define the problem. We consider a supervised regression problem defined on the feature space $\mathcal{X} \subset \mathbb{R}^d$ and the label space $\mathcal{Y} \subset \mathbb{R}$. We assume the solution of the regression problem to be in a function space $\mathcal{M} := \{g : \mathcal{X} \to \mathcal{Y}\}$, and that for any set of weights $\boldsymbol{w} \in \mathbb{R}^m$ there exists a function in $\mathcal{M}$ associated with it. $\mathcal{M}$ can be interpreted as the space of functions that we can learn by training a given regression approach through the optimization of its weights $\boldsymbol{w} \in \mathbb{R}^m$. Additionally, we consider an error function

$l : \mathcal{X} \times \mathcal{Y} \times \mathcal{M} \to \mathbb{R}^+$. The error function takes as input the features of a data point, its label, and a trained regression model and outputs a real value that measures the quality of the prediction of the model for the given data point. The smaller the error, the better the prediction.

Furthermore, we consider a dataset $\mathcal{D} := \{(\boldsymbol{x}_q, y_q)\}_{q=1}^n \subset \mathcal{X} \times \mathcal{Y}$, $n \in \mathbb{N}^+$, consisting of independent realizations of random variables $(\boldsymbol{X}, Y)$ taking values in $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ with joint probability measure $p_{\mathcal{Z}}$. We study a scenario in which we have only access to the realizations $\{\boldsymbol{x}_q\}_{q=1}^n$, while the labels $\{y_q\}_{q=1}^n$ are unknown, and the goal is to use ML techniques to predict the labels accurately and fast, recovering from data the relation between the random variables $\boldsymbol{X}$ and Y. In supervised ML, we first label a subset $\mathcal{L} := \{(\boldsymbol{x}_{q_j}, y_{q_j})\}_{j=1}^b \subset \mathcal{D}$, $b \ll n$, with $q_j \in \{1, 2, \dots, n\}$ $\forall j$. We then train a regression model $m_{\mathcal{L}} : \mathcal{X} \to \mathcal{Y}$ using a learning algorithm $A(\cdot) : 2^{\mathcal{D}} \to \mathbb{R}^m$ that maps a labeled subset $\mathcal{L} \subset \mathcal{D}$ into weights $\boldsymbol{w} \in \mathbb{R}^m$ determining the learned function $m_{\mathcal{L}} \in \mathcal{M}$ used to predict the labels of the remaining unlabeled points in $\mathcal{U} := \{\boldsymbol{x} \in \mathcal{D} \text{ such that } \boldsymbol{x} \notin \mathcal{L}\}$. The symbol $2^{\mathcal{D}}$ represents the set of all possible subsets of $\mathcal{D}$. In the following, we renumber the indices $\{q_j\}_{j=1}^b$ associated with the selected set $\mathcal{L}$, and denote them as j, that is, $\mathcal{L} := \{(\boldsymbol{x}_j, y_j)\}_{j=1}^b$. Furthermore, given a set $\mathcal{L} := \{(\boldsymbol{x}_j, y_j)\}_{j=1}^b \subset \mathcal{D}$ we define $\mathcal{L}_{\mathcal{X}} := \{\boldsymbol{x}_j\}_{j=1}^b$ and $\mathcal{L}_{\mathcal{Y}} := \{y_j\}_{j=1}^b$.

In several applications the labeling process is computationally expensive, therefore, given a budget $b \ll n$ of points to label, the goal is to select a subset $\mathcal{L} \subset \mathcal{D}$ with $|\mathcal{L}| = b$ that is most beneficial to the learning process of algorithm $A(\cdot)$. In this chapter we focus on promoting robustness of the predictions, that is, we want to minimize the maximum expected error of the predictions of the labels obtained with the learned function. Specifically, the problem we want to solve can be expressed as follows:

$$\min_{\substack{\mathcal{L} \subset \mathcal{D}, \\ |\mathcal{L}| = b}} \max_{\boldsymbol{x} \in \mathcal{U}_{\mathcal{X}}} \mathbb{E}[l(\boldsymbol{x}, Y, m_{\mathcal{L}}) | \boldsymbol{x}], \tag{5.1}$$

In other words, we aim to select and label a training set $\mathcal{L}$ of cardinality b, so that the maximum expected error associated to a trained regression model $m_{\mathcal{L}}$ evaluated on the unlabeled points is minimized. We focus on model-agnostic training set sampling strategies that have the potential to benefit various learning algorithms. In particular, we do not optimize the data selection process to benefit only an a priori chosen class of learning models.

5.2. Effects of a training set fill distance minimization approach.

Direct computation of the solution to the optimization problem in (5.1) is not possible as we do not know the labels for the points. To cope with this issue, we derive an upper bound for the minimization objective in (5.1) that depends linearly on the fill distance of the training set. Afterwards, we describe FPS, which provides a computationally feasible approach to obtain suboptimal solution for minimizing the fill distance.

First, let us introduce the fill distance, a quantity we can associate with subsets of

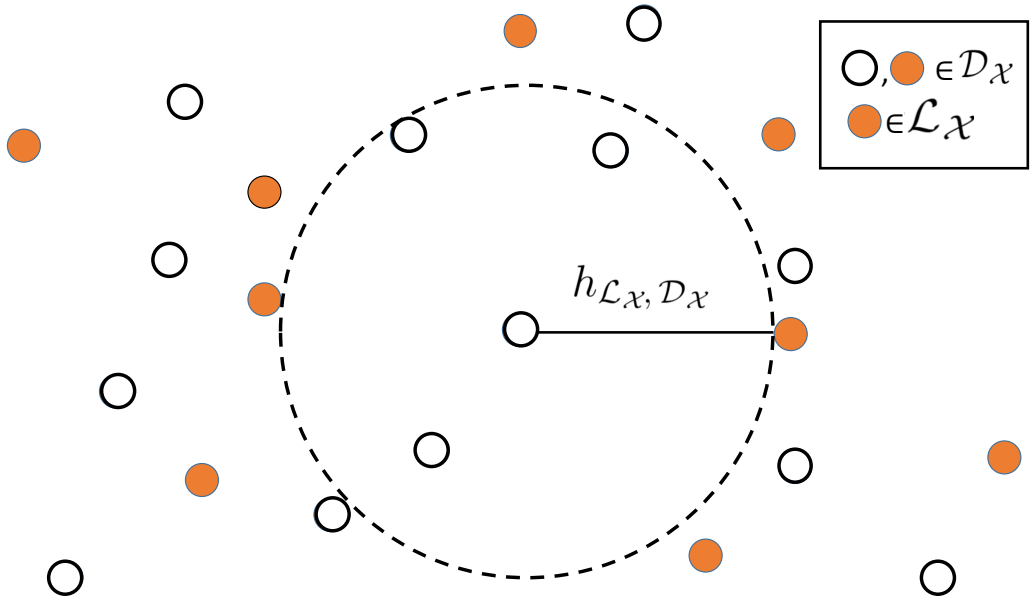


Figure 5.1.: Illustration of the concept of fill distance. The two-dimensional data points in the figure, represented as orange or white circles, constitute a finite dataset $\mathcal{D}_\mathcal{X}$. The orange circles represent a selected subset $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$. The fill distance $h_{\mathcal{L}_\mathcal{X}, \mathcal{D}_\mathcal{X}}$ of the set $\mathcal{L}_\mathcal{X}$ in $\mathcal{D}_\mathcal{X}$ can be understood as the radius of the largest ball in $\mathbb{R}^d$ (with $d = 2$ in this example) that is centered on a data point in $\mathcal{D}_\mathcal{X}$ and does not include any points from $\mathcal{L}_\mathcal{X}$.

the pool of data points we wish to label. It can be calculated by considering only the features of the data points.

Definition 5.1 *Given* $\mathcal{D}_\mathcal{X} := \{\boldsymbol{x}_q\}_{q=1}^n \subset \mathbb{R}^d$ *and* $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$*, the* fill distance *of* $\mathcal{L}_\mathcal{X}$ *in* $\mathcal{D}_\mathcal{X}$ *is defined as*

$$h_{\mathcal{L}_\mathcal{X}, \mathcal{D}_\mathcal{X}} := \max_{\boldsymbol{x} \in \mathcal{D}_\mathcal{X}} \min_{\boldsymbol{x}_j \in \mathcal{L}_\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \tag{5.2}$$

where $\|\cdot\|_2$ *is the* L_2*-norm. Put differently, we have that any point* $\boldsymbol{x} \in \mathcal{D}_\mathcal{X}$ *has a point* $\boldsymbol{x}_j \in \mathcal{L}_\mathcal{X}$ *not farther away than* $h_{\mathcal{L}_\mathcal{X}, \mathcal{D}_\mathcal{X}}$*.*

Fig. 5.1 provides an intuitive illustration of the concept of fill distance. Notice that the fill distance depends on the distance metric we consider in the feature space. In this work, for simplicity, we consider the L_2-distance, but the following result can be generalized to other distances.

Next, we formulate two assumptions. The first assumption concerns the data being analyzed and the relationship between data features and labels.

Assumption 5.1 *We assume that for any feature vector* $\boldsymbol{x}_q \in \mathcal{X}$ *we have that*

$$\mathbb{E}\left[\,|Y|\big|\boldsymbol{x}_q\right] := \int_\mathcal{Y} |y|\, p(y|\boldsymbol{x}_q) dy < \infty \tag{5.3}$$

and that there exists $\epsilon \geq 0$ *such that*

$$\mathbb{E}\left[\, |Y - \mathbb{E}[Y|\boldsymbol{x}_q]| \,\middle|\, \boldsymbol{x}_q \right] := \int_{\mathcal{Y}} \left| y - \mathbb{E}[Y|\boldsymbol{x}_q] \right| p(y|\boldsymbol{x}_q) dy \leq \epsilon, \tag{5.4}$$

where

$$p(y|\boldsymbol{x}_q) := \frac{p_{\mathcal{Z}}(\boldsymbol{x}_q, y)}{p_{\mathcal{X}}(\boldsymbol{x}_q)} \quad \textit{and} \quad p_{\mathcal{X}}(\boldsymbol{x}_q) := \int_{\mathcal{Y}} p_{\mathcal{Z}}(\boldsymbol{x}_q, y) dy. \tag{5.5}$$

We refer to "ϵ*" as the* labels' uncertainty*. Moreover, we assume that*

$$\left| \, \mathbb{E}\left[Y|\hat{\boldsymbol{x}}\right] - \mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right] \, \right| \leq \lambda_p \|\hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}}\|_2, \tag{5.6}$$

$\forall\ \hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}} \in \mathcal{X}$*, where* $\lambda_p \in \mathbb{R}^+$*.*

In most applications, it is reasonable to assume that labels are finite, as measurements or experimental outcomes typically fall within some range value. Formula (5.3) formally ensures that for any given feature vector $\boldsymbol{x}_q \in \mathcal{X}$ the expected absolute value of the associated label is finite. This assumption aligns with the practical need for models to operate on data that has finite-valued labels. Formula (5.4) states that given a realization $\boldsymbol{X} = \boldsymbol{x}_q$, the expected absolute difference between the variable Y and its conditional expectation, taken over the distribution of Y given $\boldsymbol{x}_q$, is bounded by a positive scalar ϵ. In simpler words, given a data point location $\boldsymbol{x}_q \in \mathcal{X}$ in the feature space, its associated label value is not fixed. Instead, it tends to be concentrated in a small region of the label space around its conditional expected value, whose size is determined by the positive scalar ϵ. Formula (5.4) models those scenarios where the underlying true mapping between the feature and label spaces is either stochastic in nature or deterministic with error fluctuations of magnitude parameterized by ϵ. The Lipschitz continuity in (5.6) is an assumption on the regularity of the map connecting the feature space $\mathcal{X}$ with the label space $\mathcal{Y}$. It tells us that if two data points have close representations in the feature space, then the conditional expectations of the associated labels are also close, that is, elements closer in $\mathcal{X}$ are more likely to be associated labels close in $\mathcal{Y}$.

The second assumption concerns the error function used to evaluate the performance of the model and the prediction quality of the model on the training set. Firstly, to formalize the notion that the prediction error of a trained model on the training set is bounded. Secondly, to limit our analysis to error functions that exhibit a certain degree of regularity, which also reflects the regularity of the regression model.

Assumption 5.2 *We assume there exist* $\epsilon_{\mathcal{L}} \geq 0$*, depending on the labeled set* $\mathcal{L} \subset \mathcal{D} := \{(\boldsymbol{x}_q, y_q)\}_{q=1}^n \subset \mathcal{X} \times \mathcal{Y}$ *and the trained regression model* $m_{\mathcal{L}}$*, such that for any labeled point* $(\boldsymbol{x}_j, y_j) \in \mathcal{L}$ *we have that*

$$\mathbb{E}[l(\boldsymbol{x}_j, Y, m_{\mathcal{L}}) \big| \boldsymbol{x}_j] \leq \epsilon_{\mathcal{L}}. \tag{5.7}$$

We consider $\epsilon_{\mathcal{L}}$ *as the maximum expected prediction error of the trained model* $m_{\mathcal{L}}$ *on the labeled data* $\mathcal{L}$*. Moreover, we assume that for any* $y \in \mathcal{Y}$ *and* $\mathcal{L} \subset \mathcal{D}$ *the error function* $l(\cdot, y, m_{\mathcal{L}})$ *is* $\lambda_{l_{\mathcal{X}}}$*-Lipschitz and that for any* $\boldsymbol{x} \in \mathcal{X}$ *and* $\mathcal{L} \subset \mathcal{D}$*,* $l(\boldsymbol{x}, \cdot, m_{\mathcal{L}})$ *is* $\lambda_{l_{\mathcal{Y}}}$*-Lipschitz, convex and* $\mathbb{E}[|l(\boldsymbol{x}, Y, m_{\mathcal{L}})| \big| \boldsymbol{x}] < \infty$*.*

With (5.7) we assume that the expected error on the training set is bounded. Moreover, with the Lipschitz continuity assumptions we limit our study to error functions that show a certain regularity. However, these regularity assumptions on the error function are not too restrictive and are connected with the regularity of the evaluated trained model as we show in Remark 5.1. For instance, the $\lambda_{l_{\mathcal{Y}}}$-Lipschitz regularity and the convexity in the second argument are verified by all L_p-norm error functions, with $1 \leq p < \infty$. We also assume that $\mathbb{E}[|l(\boldsymbol{x}, Y, m_{\mathcal{L}})|\big|\boldsymbol{x}] < \infty$ for any given $\boldsymbol{x} \in \mathcal{X}$. This assumption formalizes the intuitive fact that, in applications, independently of the trained model and the feature vector considered, we can expect the error function to take finite values. With that, we formulate the main theoretical result of this chapter, which is a theorem that provides an upper bound for the optimization objective in (5.1), depending linearly on the fill distance of the selected training set.

Theorem 5.1 *Consider random variables $(\boldsymbol{X}, Y)$, taking values in $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ with joint probability measure $p_{\mathcal{Z}}$. Let $\mathcal{D} := \{(\boldsymbol{x}_q, y_q)\}_{q=1}^n = \mathcal{U} \sqcup \mathcal{L} \subset \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^d \times \mathbb{R}$ be a set of independent realizations of $(\boldsymbol{X}, Y)$. Let $m_{\mathcal{L}} \in \mathcal{M}$ be a regression model trained on $\mathcal{L}$, and $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{M} \to \mathbb{R}^+$ an error function. If Assumptions 5.1 and 5.2 hold, then the following result applies:*

$$\max_{\boldsymbol{x} \in \mathcal{U}_{\mathcal{X}}} \mathbb{E}\left[l(\boldsymbol{x}, Y, m_{\mathcal{L}})|\boldsymbol{x}\right] \leq h_{\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}} \left(\lambda_{l_{\mathcal{X}}} + \lambda_{l_{\mathcal{Y}}} \lambda_p\right) + \underbrace{\lambda_{l_{\mathcal{Y}}} \epsilon}_{\substack{\text{labels} \\ \text{uncertainty}}} + \underbrace{\epsilon_{\mathcal{L}},}_{\substack{\text{max error} \\ \text{training set}}} \tag{5.8}$$

where $\mathcal{D}_{\mathcal{X}} := \{\boldsymbol{x}_q\}_{q=1}^n = \mathcal{U}_{\mathcal{X}} \sqcup \mathcal{L}_{\mathcal{X}}$. Moreover, $h_{\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}}$ is the fill distance of $\mathcal{L}_{\mathcal{X}}$ in $\mathcal{D}_{\mathcal{X}}$, ϵ and λ_p are the labels' uncertainty and Lipschitz constant from Assumption 5.1, respectively, $\lambda_{l_{\mathcal{X}}}$ and $\lambda_{l_{\mathcal{Y}}}$ are the Lipschitz constants of the error function, and $\epsilon_{\mathcal{L}}$ is the maximum expected error of the learned model predictions on the labeled set $\mathcal{L}$.

Proof. First we want to find an upper bound for $\mathbb{E}\left[l(\tilde{\boldsymbol{x}}, Y, m_{\mathcal{L}})|\tilde{\boldsymbol{x}}\right]$ for each $\tilde{\boldsymbol{x}} \in \mathcal{U}_{\mathcal{X}}$. Recall that, $\mathcal{U}_{\mathcal{X}} \subset \mathcal{X}$ is the set of data features associated with data points in $\mathcal{U} \subset \mathcal{D}$. Fixed $\tilde{\boldsymbol{x}} \in \mathcal{U}_{\mathcal{X}}$, by the definition of the fill distance we know there exists $\boldsymbol{x}_j \in \mathcal{L}_{\mathcal{X}}$ such that $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}_j\|_2 \leq h_{\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}}$. Next, we note that

$$\begin{aligned}
\mathbb{E}\left[l(\tilde{\boldsymbol{x}}, Y, m_{\mathcal{L}})|\tilde{\boldsymbol{x}}\right] &= \int_{\mathcal{Y}} l(\tilde{\boldsymbol{x}}, y, m_{\mathcal{L}}) p(y|\tilde{\boldsymbol{x}}) dy \\
&\leq \int_{\mathcal{Y}} \big|l(\tilde{\boldsymbol{x}}, y, m_{\mathcal{L}}) - l(\boldsymbol{x}_j, y, m_{\mathcal{L}})\big| p(y|\tilde{\boldsymbol{x}}) dy + \int_{\mathcal{Y}} l(\boldsymbol{x}_j, y, m_{\mathcal{L}}) p(y|\tilde{\boldsymbol{x}}) dy \\
&\leq h_{\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}} \lambda_{l_{\mathcal{X}}} + \int_{\mathcal{Y}} l(\boldsymbol{x}_j, y, m_{\mathcal{L}}) p(y|\tilde{\boldsymbol{x}}) dy
\end{aligned} \tag{5.9}$$

where $\lambda_{l_{\mathcal{X}}}$ is from Assumption 5.2. The second inequality in (5.9) follows from the $\lambda_{l_{\mathcal{X}}}$-Lipschitz continuity of the error function and from the fact that, by how it is defined

$p(y|\tilde{\boldsymbol{x}})$ in (5.5), we have $\int_{\mathcal{Y}} p(y|\tilde{\boldsymbol{x}})dy = 1$. We can bound the remaining term as follows

$$\begin{aligned}
\int_{\mathcal{Y}} l(\boldsymbol{x}_j, y, m_{\mathcal{L}})p(y|\tilde{\boldsymbol{x}})dy \leq & \int_{\mathcal{Y}} \left|l(\boldsymbol{x}_j, y, m_{\mathcal{L}}) - l(\boldsymbol{x}_j, \mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right], m_{\mathcal{L}})\right| p(y|\tilde{\boldsymbol{x}})dy \\
& + \int_{\mathcal{Y}} \left|l(\boldsymbol{x}_j, \mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right], m_{\mathcal{L}}) - l(\boldsymbol{x}_j, \mathbb{E}\left[Y|\boldsymbol{x}_j\right], m_{\mathcal{L}})\right| p(y|\tilde{\boldsymbol{x}})dy \\
& + \int_{\mathcal{Y}} l(\boldsymbol{x}_j, \mathbb{E}\left[Y|\boldsymbol{x}_j\right], m_{\mathcal{L}})p(y|\tilde{\boldsymbol{x}})dy \\
\leq & \lambda_{l_{\mathcal{Y}}} \int_{\mathcal{Y}} \left|y - \mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right]\right| p(y|\tilde{\boldsymbol{x}})dy \\
& + \lambda_{l_{\mathcal{Y}}} \int_{\mathcal{Y}} \left|\mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right] - \mathbb{E}\left[Y|\boldsymbol{x}_j\right]\right| p(y|\tilde{\boldsymbol{x}})dy \\
& + \int_{\mathcal{Y}} \mathbb{E}[l(\boldsymbol{x}_j, Y, m_{\mathcal{L}})|\boldsymbol{x}_j]p(y|\tilde{\boldsymbol{x}})dy \\
\leq & \lambda_{l_{\mathcal{Y}}}\epsilon + \lambda_{l_{\mathcal{Y}}} \int_{\mathcal{Y}} \left(\lambda_p h_{\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}}\right) p(y|\tilde{\boldsymbol{x}})dy + \int_{\mathcal{Y}} \epsilon_{\mathcal{L}}\, p(y|\tilde{\boldsymbol{x}})dy \\
\leq & \lambda_{l_{\mathcal{Y}}}\epsilon + \lambda_{l_{\mathcal{Y}}}\lambda_p h_{\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}} + \epsilon_{\mathcal{L}}.
\end{aligned} \tag{5.10}$$

The second inequality follows from the $\lambda_{l_{\mathcal{Y}}}$-Lipschitz continuity of the error function and Jensen's inequality, which is used to obtain the conditional expectation in the integrand of the last term. The third inequality follows from the definition of labels' uncertainty, the λ_p-Lipschitz continuity of the conditional expectation of the random variable Y and the assumption that the expected error on the training set is bounded by $\epsilon_{\mathcal{L}}$. The fourth inequality is obtained by taking out the constants from the integrals in the second and third terms and noticing that, from the definition of $p(y|\tilde{\boldsymbol{x}})$ in (5.5), we have $\int_{\mathcal{Y}} p(y|\tilde{\boldsymbol{x}})dy = 1$. Since inequalities (5.9) and (5.10) hold for each $\tilde{\boldsymbol{x}} \in \mathcal{U}_{\mathcal{X}}$, we have that

$$\max_{\boldsymbol{x} \in \mathcal{U}_{\mathcal{X}}} \mathbb{E}\left[l(\boldsymbol{x}, Y, m_{\mathcal{L}})|\boldsymbol{x}\right] \leq h_{\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}} \left(\lambda_{l_{\mathcal{X}}} + \lambda_{l_{\mathcal{Y}}}\lambda_p\right) + \lambda_{l_{\mathcal{Y}}}\epsilon + \epsilon_{\mathcal{L}}. \tag{5.11}$$

□

Formula (5.8) provides an upper bound for the minimization objective in (5.1) that is linearly dependent on the fill distance of the training set. Note that our derived bound also depends on the labels' uncertainty "ϵ". In particular, the larger the labels' uncertainty, the larger the bound for a fixed training set fill distance. Assuming that the maximum error on the labeled data ($\epsilon_{\mathcal{L}}$) is negligible, the smaller the fill distance, the smaller the bound for the maximum expected approximation error on the unlabeled set, conditional to the unlabeled data locations. Although $\epsilon_{\mathcal{L}}$ is typically considered to be small, its presence in the formula suggests that the maximum expected error on the unlabeled set is also dependent on the maximum error of the predictions on the labeled set used for training, thus, on how well the trained model fits the training data. Additionally, the connections between the bound and the regularity of the map connecting

the features and the labels, and the chosen error function are highlighted by the presence of the Lipschitz constants λ_p, $\lambda_{l_\mathcal{X}}$ and $\lambda_{l_\mathcal{Y}}$ on the right-hand side of (5.8).

We remark that if we consider the error function to be the absolute value of the difference between true and predicted labels, Theorem 5.1 holds for all Lipschitz continuous learning algorithms, such as kernel ridge regression with the Gaussian kernel and feed forward neural networks.

Remark 5.1 *If the trained model $m_\mathcal{L} \in \mathcal{M}$ is $\lambda_{l_\mathcal{X}}-$Lipschitz continuous, then also the absolute value error function is $\lambda_{l_\mathcal{X}}-$Lipschitz continuous with respect to its first argument. To see this, fix $y \in \mathcal{Y}$, $\mathcal{L} \subset \mathcal{D}$ and $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{X}$. Then we have*

$$|l(\boldsymbol{x}, y, m_\mathcal{L}) - l(\tilde{\boldsymbol{x}}, y, m_\mathcal{L})| = \big||m_\mathcal{L}(\boldsymbol{x}) - y| - |m_\mathcal{L}(\tilde{\boldsymbol{x}}) - y|\big| \leq |m_\mathcal{L}(\boldsymbol{x}) - m_\mathcal{L}(\tilde{\boldsymbol{x}})|.$$

Moreover, the absolute value error function is always $\lambda_{l_\mathcal{Y}}$-Lipschitz with respect to its second argument with $\lambda_{l_\mathcal{Y}} = 1$. As a matter of fact, fixed $\boldsymbol{x} \in \mathcal{X}$, $m_\mathcal{L} \in \mathcal{M}$ and $y, \tilde{y} \in \mathcal{Y}$ we have

$$|l(\boldsymbol{x}, y, m_\mathcal{L}) - l(\boldsymbol{x}, \tilde{y}, m_\mathcal{L})| = \big||m_\mathcal{L}(\boldsymbol{x}) - y| - |m_\mathcal{L}(\boldsymbol{x}) - \tilde{y}|\big| \leq |y - \tilde{y}|.$$

It is worth noticing that the result in Theorem 5.1 is independent of the data dimension in the feature space. That is, the right-hand-side of Formula 5.8 does not explicitly include the data features dimension $d \in \mathbb{N}$. While the independence of the data dimension is a desirable quality of our proposed bound, in practical scenarios, the dimensionality of the data significantly impacts the interpretability of the fill distance and, therefore, the interpretability and effectiveness of the bound, especially when the L_2 distance is used. This issue arises due to the well-known "curse of dimensionality". The curse of dimensionality was first introduced by [Bel61], and it refers to the difficulties that arise when working with high dimensional data. As mentioned in [RNI10], "One aspect of the dimensionality curse is distance concentration, which denotes the tendency of distances between all pairs of points in high-dimensional data to become almost equal". In such a scenario it is clear that the fill distance may not be a good indicator of how well the selected training set represents the data space of interest. Thus, while the theorem provides a dimension-agnostic bound, the practical implications of high-dimensional spaces and the specific effects of the L_2 distance metric cannot be overlooked in practical applications. For more literature on the effects of the high dimensions in ML application we refer the reader to [RNI10].

5.3. Selecting training sets with the farthest point sampling

Theorem 5.1 provides an upper bound for the maximum expected value of the error function on the unlabeled data, conditional to the knowledge of the data features. Our aim is to select a training set by minimizing such a bound. Assuming that the value of the maximum prediction error of the trained regression model on the training set is

negligible, we can attempt the minimization of the upper bound in (5.8) by solving the following minimization problem

$$\min_{\substack{\mathcal{L}\subset\mathcal{D},\\ |\mathcal{L}|=b}} h_{\mathcal{L}_\mathcal{X},\mathcal{D}_\mathcal{X}}, \tag{5.12}$$

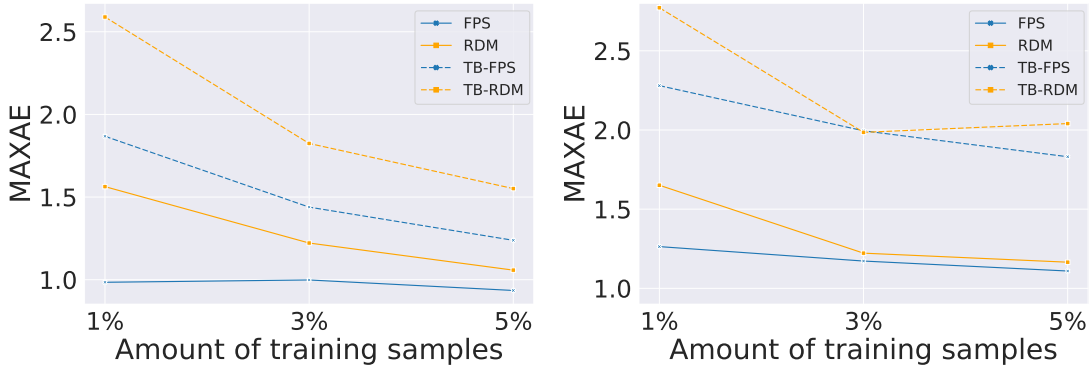
where $\mathcal{D} := \{(\boldsymbol{x}_q, y_q)\}_{q=1}^n \subset \mathcal{X} \times \mathcal{Y}$ is the pool of data points we want to label, and $\mathcal{L} := \{(\boldsymbol{x}_j, y_j)\}_{j=1}^b$ is the set of labeled points we use for training. The minimization problem in (5.12) is equivalent to the k-center clustering problem [HP11]. Given a set of points in a metric space, the k-center clustering problem consists of selecting k points, or centers, from the given set so that the maximum distance between a point in the set and its closest center is minimized, i.e., the fill distance of the k centers in the set is minimized. Unfortunately, the k-center clustering problem is NP-Hard [Hoc84]. However, using the farthest point sampling (FPS), introduced in Section 3.3.2 and described in Algorithm 5, it is possible to obtain sets with fill distance at most a factor of 2 from the minimal fill distance in polynomial time [HP11]. It is worth to recall that reducing the factor of approximation below 2 would require solving an NP-hard problem [HS85]. Thus, FPS provides a suboptimal solution, but obtaining a better approximation with theoretical guarantees would not be feasible in polynomial time. To give a qualitative understanding of the data efficiency of FPS, with our implementation of the FPS algorithm, it takes approximately 70 seconds [1] to select 1000 points from the training dataset provided within the selection-for-vision DataPerf challenge [MBY+22], consisting of circa 3.3 millions points in $\mathbb{R}^{256}$.

5.3.1. An illustrative numerical example

To provide a more empirical understanding of our theoretical result and the effects of minimizing the training set fill distance using FPS, we provide an illustrative example where we empirically estimate the theoretical bound provided in Theorem 5.1 and compare it with the computed maximum error achieved by a given regression model.

We use a set of two-dimensional data points $\hat{\mathcal{D}} := \{\boldsymbol{x}_i\}_{i=1}^{1000} \subset [-1,1]^2$, the function $f(\boldsymbol{x}_i) = x_{i,1} \times x_{i,2}$, which is the function we aim to predict from data, where $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}]$, and a linear regression model. We consider the absolute error as error function, which is the absolute value of the difference between the actual and predicted function evaluated on a specific data point. We note that $f(\boldsymbol{x})$ is Lipschitz continuous with respect to the absolute error function in $[-1,1]^2$ with Lipschitz constant $\lambda_p = \sqrt{2}$. Moreover, the Lipschitz constants associated with the absolute error of the predictions of a linear regression model are $\lambda_{l_\mathcal{X}} = \|\boldsymbol{a}_\mathcal{L}\|_2$ and $\lambda_{l_\mathcal{Y}} = 1$, where $\boldsymbol{a} \in \mathbb{R}^2$ is the vector of the weights learned by the linear model trained on a set $\mathcal{L} \subset \mathcal{D}$. $\lambda_{l_\mathcal{X}}$ coincide with the Lipschitz constant of the regression model as we know from Remark 5.1. We also consider a noisy version of the dataset, by independently adding random noise $\epsilon_i \in \mathbb{R}$ with mean zero and variance 0.1 to each of the data point labels $y_i = f(\boldsymbol{x}_i)$ for $i = 1, \ldots, 1000$. To quantify the amount of noise in the data we compute the data to noise ratio $DTNS := \frac{\frac{1}{1000}\sum_{i=1}^{1000} y_i^2}{\frac{1}{1000}\sum_{i=1}^{1000} \epsilon_i^2}$,

[1] We used a 48-cores CPU with 384 GB RAM.



(a) No noise

(b) data-to-noise ratio: 12.5

Figure 5.2.: Results for regression task on the illustrative example dataset.(a) No noise. (b) Noisy labels. We use a linear regression model trained on sets of various sizes selected uniformly at random (RDM) and with FPS. The amount of data used for training is expressed as a percentage of the available data points. The graphs illustrate the maximum absolute error (MAXAE) of the trained linear regression model and the data-driven estimation of theoretical bound (TB) from (5.8) for the expected maximum error of the linear model. The results suggest that our bound provides an effective qualitative estimate of the expected maximum error. That is, the larger the bound the larger the MAXAE. Moreover, using FPS leads to lower MAXAE.

which in this illustrative experiment is equal to 12.5. Moreover, we approximate the labels' uncertainty with the maximal noise magnitude $\bar{\epsilon} := \max_{1\leq i\leq 1000} |\epsilon_i|$ We train the linear model independently on subsets $\mathcal{L}_i \subset \hat{\mathcal{D}}$, $i = 1, 2, 3$, consisting of 1%, 3% and 5% of the available data points, that is, 10, 30 and 50 points, respectively. Next, we compute the maximum error of the predictions on the training sets,$\{\epsilon_{\mathcal{L}_i}\}_{i=1}^{3}$, and estimate the theoretical bounds (TB) related to each of the selected training sets and respective trained models as follows

$$TB(\mathcal{L}_i, \boldsymbol{a}_{\mathcal{L}_i}) := h_{\mathcal{L}_i,\hat{\mathcal{D}}}\left(\|\boldsymbol{a}_{\mathcal{L}_i}\|_2 + \sqrt{2}\right) + \epsilon_{\mathcal{L}_i} + \bar{\epsilon}.$$

We consider both, training sets selected randomly and with FPS.

Figs. 5.2a-b illustrate the maximum error of the predictions of the linear model trained on randomly selected training sets (in orange) and sets selected with FPS (in blue), for the noiseless and noisy data scenarios, respectively. Moreover, Figs. 5.2a-b illustrate the related TB (dashed lines) for each training set size and selection strategy considered. The figure suggests that, the theoretical bound provides a qualitative estimate of the expected maximum error that can capture the behavior of the true maximum error, independently of the selection strategy, the training set size and whether the data is noisy or not. Furthermore, Fig. 5.2 indicates that selecting the training set by fill distance minimization with the FPS reduces the theoretical bound calculated according to (5.8)

in Theorem 5.1 and the maximum error of predictions, with respect to the randomly selected training sets. Comparing Fig. 5.2a and Fig. 5.2b, we can see that including noise in data increases the maximum prediction error and the theoretical error bound, independently of the selection strategy employed. This can be expected as noise can distort the true underlying relationship between the input features and the target variable, making the regression task more challenging. It is also important to note that including noise does not only affect the bound in terms of the labels' uncertainty, but it also has an effect of other quantities such as the maximum error on the training set and the Lipschitz constant of the trained regression model, which is determined by the learned weights. This is because both the maximum error on the training set and the learned weights also depend on the label values considered in the regression task. Additionally, it is worth highlighting that, with FPS, the maximum prediction error is flat, that is, it converges fast to a plateau value. This is particularly evident in Fig. 5.2a where noise is not included. It can be clearly seen that such a phenomenon may not be reflected in our proposed bound, indicating that our theorem provides a qualitative estimate of the maximum expected error that may be further improved. In Section 7.1.5, we empirically investigate on three different datasets why there can be a fast decay of the maximum expected error when using the FPS and how this is related to the data points distribution in the feature space. Nonetheless, Fig 5.2 suggests that our proposed bound provides an effective qualitative estimate of the expected maximum error and that reducing the training set fill distance benefits the robustness of the trained model. This is evident from both a theoretical perspective, as shown by the decrease in the computed value of the theoretical bound, and from an empirical point of view, as indicated by the reduced maximum prediction error.

5.4. Increased numerical stability of Gaussian kernel regression with FPS

Our theoretical analysis suggests that selecting training sets by fill distance minimization with the FPS leads to a reduction in the expected maximum error of the predictions of Lipschitz continuous models. We note that our analysis relies on the Lipschitz continuity assumption of the loss function, which is related to the regularity of the trained model under consideration, as suggested by Remark 5.1. A natural question is whether we can highlight additional benefits of using the FPS for selecting the training set by tightening the assumption on the regularity of the regression model through the consideration of specific regression models. In this section, we investigate the additional benefit of selecting the training set using FPS for kernel regression models with the Gaussian kernel, a class of regression approaches successfully employed in various applications such as molecular and material sciences [DBB$^+$21], or robotics [DFR15]. In particular, we see how selecting training sets with the FPS increases the model stability for this specific class of regression approaches.

Numerical stability in a regression approach is a key factor in ensuring the robustness of the learning algorithm with respect to noise and therefore its reliability. A standard

criterion for measuring the numerical stability in case of kernel regression is the condition number of the kernel matrix, $\mathbf{K}_{\mathcal{L}} \in \mathbb{R}^{b,b}$. In the specific case of a Gaussian kernel we have that $\mathbf{K}_{\mathcal{L}}(i,j) := e^{-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}$, $\gamma \in \mathbb{R}^+$. The condition number of a matrix is defined as

$$cond(\mathbf{K}_{\mathcal{L}}) := \|\mathbf{K}_{\mathcal{L}}\|_2 \|\mathbf{K}_{\mathcal{L}}^{-1}\|_2 = \frac{\lambda_{max}(\mathbf{K}_{\mathcal{L}})}{\lambda_{min}(\mathbf{K}_{\mathcal{L}})}, \tag{5.13}$$

where $\lambda_{max}(\mathbf{K}_{\mathcal{L}})$ and $\lambda_{min}(\mathbf{K}_{\mathcal{L}})$ are the largest and smallest eigenvalues of $\mathbf{K}_{\mathcal{L}}$, respectively. The smaller the condition number, the more numerically stable the algorithm. For high condition numbers, the numerical computations involving the kernel matrix can suffer from amplification of rounding errors and loss of precision that can lead to numerical instability when performing operations like matrix inversion or solving linear systems involving the kernel matrix. Such phenomena may also lead to instability of the predictions as small variations in the input may lead to significant variations in the output.

To increase the model stability we aim to select a training set that leads to a kernel matrix with a small condition number (5.13). Since the condition number can be expressed as the ratio between the largest and smallest eigenvalues of the kernel, we aim to select a training set that minimizes the largest eigenvalue while maximizing the smallest eigenvalue. In the following, we recapture results related to the stability of the kernel matrix that have been presented in [Wen04] in the context of numerical mathematics. Our goal is to connect such results with the FPS and show that FPS can be used to increase the stability of KRR models by reducing the condition number of the kernel matrix. For further information and analysis on the stability of kernel matrices, we direct the reader to [Wen04], Chapter 12. From the cited literature, we know that the largest eigenvalue of a kernel matrix is mainly dependent on the number of points we consider and not on how we choose them. In particular, the value of the largest eigenvalue can be bounded as follows

$$\lambda_{max}(\mathbf{K}_{\mathcal{L}}) \leq b \max_{q,r=1,\dots,b} |\mathbf{K}_{\mathcal{L}}(q,r)|. \tag{5.14}$$

Thus, the maximum eigenvalue is bounded by a quantity that depends linearly on the number of training samples times the maximal entry of the kernel matrix. Since we are considering Gaussian kernels, the maximal entry of the kernel is bounded. Consequently, the value of the maximal eigenvalue grows at most as fast as the number of points we select, independently of how we choose them.

On the contrary, the value of the smallest eigenvalue is strongly dependent on how we choose the training points. To study that, we use the separation distance, a quantity we can associate to subsets of our pool of unlabeled data points.

Definition 5.2 *Given set $\mathcal{L}_{\mathcal{X}} := \{\boldsymbol{x}_j\}_{j=1}^b \in \mathbb{R}^d$, the* separation distance *of the points in $\mathcal{L}_{\mathcal{X}}$ defined as*

$$s_{\mathcal{L}_{\mathcal{X}}} := \frac{1}{2} \min_{\substack{\boldsymbol{x}_q, \boldsymbol{x}_r \in \mathcal{L}_{\mathcal{X}} \\ q \neq r}} \|\boldsymbol{x}_q - \boldsymbol{x}_r\|_2.$$

In words, the separation distance is half the minimal distance between two points in $\mathcal{L}_{\mathcal{X}}$. Given a training set $\mathcal{L} \subset \mathcal{D}$ we define $s_{\mathcal{L}_{\mathcal{X}}}$ to be its separation distance.

With the concept of separation distance in mind, we observe that the value of the smallest eigenvalue of the Gaussian kernel matrix can be bounded from below as [Wen04]

$$\lambda_{min}(\mathbf{K}_{\mathcal{L}}) \geq C_d \left(\sqrt{2\gamma}\right)^{-d} e^{\frac{-40.71d^2}{(s^2_{\mathcal{L}_{\mathcal{X}}}\gamma)}} s^{-d}_{\mathcal{L}_{\mathcal{X}}}, \tag{5.15}$$

where $d \in \mathbb{N}$ is the training data dimension, which is fixed, $\gamma \in \mathbb{R}^+$ is the Gaussian kernel hyperparameter, representing the width of the Gaussian, and $s_{\mathcal{L}_{\mathcal{X}}} \in \mathbb{R}^+$ is the training set separation distance. It is important to notice that the lower bound of the smallest eigenvalue decreases exponentially as the separation distance of the selected set decreases. Consequently, given two training sets of the same size, a small difference in their separation distance may lead to a large difference between the smallest eigenvalue of their corresponding kernel matrices, thus also in condition number and model stability. We note that the inequality in (5.15) applies to the minimum eigenvalue of the Gaussian Kernel specifically. It is interesting to know that [Wen04, Theorem 12.3] provides a more general lower bound for the minimum eigenvalue, depending on the separation distance, that applies to all kernels that can be characterized by an even conditionally positive definite function that possesses a positive Fourier transform, including the Gaussian kernel.

Given Formula (5.15), in order to increase the model stability of the kernel regression approach with Gaussian kernel, we aim to select a training set that solves the following NP optimization problem

$$\max_{\substack{\mathcal{L}\subset\mathcal{D}\\|\mathcal{L}|=b}} s_{\mathcal{L}_{\mathcal{X}}}. \tag{5.16}$$

Interestingly, the FPS provides sets with separation distance at most a factor of 2 from the maximal separation distance [RRT94]. Specifically, given $Q \subset \mathcal{D}$ such that

$$Q \in \arg\max_{\substack{\mathcal{L}\subset\mathcal{D}\\|\mathcal{L}|=b}} s_{\mathcal{L}_{\mathcal{X}}}.$$

Then, the separation distance of a set $\mathcal{L}^{FPS} \subset \mathcal{D}$, $|\mathcal{L}^{FPS}| = b$, obtained using FPS, is at least the half of the maximal separation distance, that is,

$$\frac{1}{2} s_{Q_{\mathcal{X}}} \leq s_{\mathcal{L}^{FPS}_{\mathcal{X}}}.$$

Moreover, to obtain an approximation factor better than 2, an NP problem must be solved [RRT94]. Thus, the FPS provides the solution with optimal approximation factor in polynomial time to both the problems in (5.12) and (5.16). Consequently, when we consider kernel regression approaches with the Gaussian kernel, selecting the training set with the FPS leads to more robust and stable models

5.5. Alternatives to the fill distance

The fill distance is a scalar that measures how well a subset represents the dataset from which it is selected. From Definition 5.1, the fill distance of a set $\mathcal{L}_\mathcal{X}$ in $\mathcal{D}_\mathcal{X}$ can be understood as the radius of the largest ball in $\mathbb{R}^d$ that is centered on a data point in $\mathcal{D}_\mathcal{X}$ and does not include any points from $\mathcal{L}_\mathcal{X}$ (Fig. 5.1). In what follows, we see how the concept of dispersion generalizes the notion of fill distance by considering other d-dimensional shapes instead of balls.

The concept of dispersion, introduced by [Hla76], was initially used to quantify error estimates in numerical approximation procedures, such as, estimating the maximum value of a function within the unit cube. The notion of dispersion originally coincided with the definition of the fill distance of a point set within the d-dimensional cube. Over time, the concept of dispersion evolved and was expanded, particularly in the context of Quasi-Monte Carlo methods [RT96].

Dispersion is typically defined for the unit cube $Q_d := [0,1]^d$ [LL22]. Let us define the set

$$\mathcal{R}_d := \Big\{ \prod_{i=1}^{d} I_i \mid I_i := [a_i, b_i) \subset [0,1] \Big\}. \tag{5.17}$$

$\mathcal{R}_d$ is the set of all axis-parallel boxes within Q_d. Suppose P is a finite subset of Q_d. The dispersion of P is defined as

$$disp(P) := \sup\{\, Vol_d(B) \mid B \in \mathcal{R}_d,\ B \cap P = \emptyset \}, \tag{5.18}$$

where $Vol_d(B) := \int_B d\boldsymbol{x}$ is the d-dimensional volume of B. Thus, the dispersion is the size of the largest axis-parallel box within Q_d that contains no points from P. The definition of dispersion can be further generalized. For example, instead of computing the volumes of the boxes with the Lebesgue measure, one could consider a probability measure p on Q_d, that is, $Vol_d(B) = \int_B p(\boldsymbol{x}) d\boldsymbol{x}$. Similarly, the family of subsets R_d could be replaced with any family of measurable subsets of Q_d with respect to p [RT96].

To apply the concept of dispersion based on axis-parallel boxes to training data selection, some challenges arise. These challenges stem from the fact that, in practice, we often work with a finite dataset $\mathcal{D}_\mathcal{X} := \{\boldsymbol{x}_i\}_{i=1}^n$ in an unknown space $\mathcal{X} \subset \mathbb{R}^d$, which may not be bounded, rather than within a unit cube. The goal is to represent the dataset $\mathcal{D}_\mathcal{X}$ with a subset $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$. One way to address this issue is to redefine the family of axis-parallel boxes $\mathcal{R}_d$ to ensure their centroids lie within $\mathcal{D}_\mathcal{X}$. That is, we can consider

$$\widetilde{\mathcal{R}}_{\mathcal{D}_\mathcal{X}} := \Big\{ \prod_{j=1}^{d} I_k \mid I_k := [a_k, b_k] \subset \mathbb{R} \text{ such that } \exists\, \boldsymbol{x} \in \mathcal{D}_\mathcal{X} \text{ with } x_k := \frac{b_k - a_k}{2} \Big\} \tag{5.19}$$

and define the dispersion of a set $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$ as

$$\widetilde{disp}(\mathcal{L}_\mathcal{X}, \mathcal{D}_\mathcal{X}) := \sup\{\, Vol_d(B) \mid B \in \widetilde{\mathcal{R}}_d,\ B \cap \mathcal{L}_\mathcal{X} = \emptyset \}. \tag{5.20}$$

The scalar in (5.20) represents the volume of the largest axis-parallel box in $\mathbb{R}^d$ with its centroid in $\mathcal{D}_\mathcal{X}$ that does not include any point from $\mathcal{L}_\mathcal{X}$. This provides a measure of how

effectively the selected set $\mathcal{L}_{\mathcal{X}}$ covers $\mathcal{D}_{\mathcal{X}}$. A smaller value of dispersion indicates better coverage of $\mathcal{D}_{\mathcal{X}}$ by $\mathcal{L}_{\mathcal{X}}$. Therefore, given a data budget $b \in \mathbb{N}$, we can select a training set by minimizing the value in (5.20). This can be achieved by solving the following optimization problem:

$$\min_{\substack{\mathcal{L}_{\mathcal{X}} \subset \mathcal{D}_{\mathcal{X}} \\ |\mathcal{L}_{\mathcal{X}}| = b}} \widetilde{disp}(\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}}). \tag{5.21}$$

In principle, we can define $\widetilde{disp}(\mathcal{L}_{\mathcal{X}}, \mathcal{D}_{\mathcal{X}})$ considering more general shapes than axis-parallel boxes. Exploring alternative shapes and examining their theoretical and practical implications could lead to more reliable metrics than the fill distance for evaluating coverage and enhancing the representativeness of selected subsets. We remark that the concept of fill distance can be related to the concept of dispersion based on ball radii instead of volumes. The curse of dimensionality makes concepts of space coverage based on ball radii less effective or meaningful. In high dimensions, the volume of a ball becomes increasingly concentrated near its surface, making the fill distance (radius-based dispersion) less informative about space coverage. This is also one of the facts that led the Quasi-Monte Carlo community to generalize the concept of dispersion to make it more effective and useful in high-dimensional domains. Future research should focus on extending our findings related to fill distance to various concepts of dispersion.

Note that there may be alternatives to the fill distance other than the concept of dispersion. In this work, we use the concept of fill distance based on the L_2 metric. We may choose to consider different distance metrics or concepts of similarity between data points. One notable alternative, which we do not explore in this study, is the density connectivity distance (dc-distance) developed in [BDH+23]. The dc-distance measures the similarity between two data points based on the L_2 distance while also accounting for the overall density distribution of the points. As we see in the next chapter, incorporating data density information into the sampling process may allow the sampling procedure to go beyond only reducing the MAXAE and also improving MAE performances.

6. On minimizing a training set weighted fill distance

In the previous chapter we focused on the low data budget regime scenario. The goal was sampling points to improve the robustness of regression models by prioritizing the performance improvement on the worst-case scenarios. That is, minimizing the maximum prediction error rather than optimizing for average performance. We derived an upper bound for the maximum expected prediction error of Lipschitz continuous regression models that is linearly dependent on the training set fill distance. Moreover, we provided a first example where minimizing the training set fill distance with the FPS significantly reduces the maximum error of the label predictions of a linear regression model.

In this chapter we focus on the medium data budget regime scenario. In particular, we aim to develop a data selection strategy to select training sets that can improve the average prediction quality of ML regression models. We intend to do that by selecting training sets that can capture the distribution of the data in our pool of interest, while ensuring that the whole data space is represented in the training set. To achieve this, we propose a passive sampling, model-agnostic data selection strategy we name "Density-Aware Farthest Point Sampling" (DA-FPS). Our proposed strategy greedily selects sets from a pool of available data points. The primary goal of DA-FPS is selecting training sets that can improve the average prediction performance of Lipschitz continuous regression models on the new points not considered for training. DA-FPS starts analogously to FPS, iteratively sampling data points at a maximal distance from those already selected. However, contrary to FPS, after selecting a portion of the data pool, DA-FPS also considers the underlying data source distribution during the sampling process. This is done by considering weighted distances, where the weights depend on the selected set and enclose information related to the distribution of the selected training points in relation to the distribution of the data points in the available pool. The weights we use to deform the distances, which we later formally define, take higher values in those regions that are not properly represented by the training data distribution, that is, where the training data distribution underestimates the data source distribution. Note that, since the weights we consider depend on the selected set they dynamically change as the selected set changes.

We derive an upper bound for the expected prediction error of Lipschitz continuous regression models on data arising from a fixed source distribution. Our derived bound is linearly dependent on the weighted fill distance, that is, the maximum weighted distance between a point in the feature domain of interest and its closest selected point. We prove that our proposed algorithm, DA-FPS, provides suboptimal minimizers for a data-driven estimation of the weighted fill distance, thereby attempting to minimize our derived

bound.

We are not aware of any other passive and model-agnostic sampling strategy supported by theoretical work that aims at improving the average prediction error of the predictions of a regression model. Furthermore, we are not aware of any approach, other than DA-FPS, that can provide suboptimal minimizers to the weighted k-center problem with dynamics weights as we later define it.

6.1. Problem definition

Recapturing the setting introduced in the previous chapter, we focus on a supervised regression problem defined on the bounded spaces $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}$ determining the feature and label space, respectively. We assume the solution of the regression problem to be on a function space $\mathcal{M} := \{g : \mathcal{X} \to \mathcal{Y}\}$. That is, for each set of weights $\boldsymbol{w} \in \mathbb{R}^m$ we learn by training a given learning algorithm there exists a function in $\mathcal{M}$ associated with it. Additionally, we consider an error function $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{M} \to \mathbb{R}^+$.

Let $\mathcal{D} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be a pool of available data points consisting of i.i.d. realizations of two random variables $\boldsymbol{X}$ and Y taking value on $\mathcal{X}$ and $\mathcal{Y}$, respectively, with joint distribution $p_{\mathcal{D}} \in \mathcal{P} := \{p : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+ | \int_{\mathcal{X}\times\mathcal{Y}} p(\boldsymbol{x}, y) d\boldsymbol{x}dy = 1\}$. Our objective is to select and label a training set $\mathcal{L} := \{(\boldsymbol{x}_{i_j}, y_{i_j})\}_{j=1}^b$ from $\mathcal{D}$, with $i_j \in \{1, \dots, n\}$ $\forall j$, and $b \ll n$, to maximize the average predictive performance of a regression model $m_{\mathcal{L}} \in \mathcal{M}$, trained on $\mathcal{L}$, on the new data points not selected for training. In the following, without loss of generality, we assume $\mathcal{L} := \{(\boldsymbol{x}_j, y_j)\}_{j=1}^b$ and define $\mathcal{L}_{\mathcal{X}} := \{\boldsymbol{x}_j\}_{j=1}^b$. Following along [WY15], we consider a scenario where the training data is selected according to a distribution $p_{\mathcal{L}} \in \mathcal{P}$ with $p_{\mathcal{D}} \neq p_{\mathcal{L}}$ and $p_{\mathcal{D}}(y|\boldsymbol{x}) = p_{\mathcal{L}}(y|\boldsymbol{x})$. In other words, the training data distribution may differ from the data source distribution, but the map connecting a data location $\boldsymbol{x} \in \mathcal{X}$ and its associated label value is independent on how the data is selected. Furthermore, we study a typical scenario involving molecular prediction tasks in which the data source distribution $p_{\mathcal{D}}$ is unknown, and we have only access to the locations $\mathcal{D}_{\mathcal{X}} := \{\boldsymbol{x}_i\}_{i=1}^n$ of the dataset $\mathcal{D}$. Moreover, we are interested in applications in which the labeling process is computationally expensive, therefore, given a budget $b \in \mathbb{N}$ of points to label, the goal is to select a subset $\mathcal{L}_{\mathcal{X}} \subset \mathcal{D}_{\mathcal{X}}$ and label it to obtain a subset $\mathcal{L} \subset \mathcal{D}$ that solves the following optimization problem:

$$\min_{\substack{\mathcal{L}\subset\mathcal{D},\\ |\mathcal{L}|=b}} \mathbb{E}_{p_{\mathcal{D}}}[l(\boldsymbol{X}, Y, m_{\mathcal{L}})], \tag{6.1}$$

In other words, we aim to sample and label a training set $\mathcal{L}$, of cardinality b, so that the expected error on the data distribution $p_{\mathcal{D}}$ associated with the regression function $m_{\mathcal{L}}$ is minimized.

Both the optimization problems in (5.1) and (6.1) focus on minimizing an expected value of the error function by selecting a subset $\mathcal{L}$ from a set of available data points $\mathcal{D}$. The key distinction lies in their objectives: in (5.1), the goal is to minimize the maximum expected error over a finite set of known data locations. In contrast, in (6.1),

the objective is to minimize the average error, defined as the expected value of the error function over the data source distribution $p_{\mathcal{D}}$ on $\mathcal{X} \times \mathcal{Y}$.

In (6.1), an alternative version of the problem could involve minimizing the average expected error over a finite set of data points with known data locations. We decide to address the more general version by considering the expected error over a data distribution $p_{\mathcal{D}}$ defined on $\mathcal{X} \times \mathcal{Y}$, which may be non-finite sets. Addressing the more general problem requires modifications to the formulation of the problem setting. For instance, we now assume the error function is evaluated on the non-zero support of $p_{\mathcal{D}}$, which may contain an uncountable number of points. In particular, the feature space $\mathcal{X}$ may be non-finite. Thus, given the data point locations of a finite training set $\mathcal{L}_{\mathcal{X}} \subset \mathcal{X}$ if we want to use a notion of coverage based on pairwise distances between data features of points in $\mathcal{L}_{\mathcal{X}}$ and those in $\mathcal{X} \subset \mathbb{R}^d$, we must assume $\mathcal{X}$ is bounded. For example, if the fill distance $h_{\mathcal{L}_{\mathcal{X}},\mathcal{X}} := \sup_{\boldsymbol{x}\in\mathcal{X}} \min_{\boldsymbol{x}_j\in\mathcal{L}_{\mathcal{X}}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2$ is used, $\mathcal{X}$ must be bounded to ensure it is finite.

6.2. Bound for the expected prediction error

The optimization problem in (6.1) cannot be solved directly because the labels of the points in $\mathcal{D}$ and the data source distribution $p_{\mathcal{D}}$ are unknown. To address this challenge, we establish an upper bound for the optimization objective in (6.1), depending linearly on a weighted fill distance of the training set. Subsequently, we introduce Density-Aware Farthest Point Sampling (DA-FPS), a novel data selection algorithm aiming at selecting training sets that can minimize the proposed bound. First, we define the weighted fill distance, a quantity we can associate with finite subsets of the feature space $\mathcal{X}$ selected from $p_{\mathcal{X}_{\mathcal{L}}}$, which is the marginal on $\mathcal{X}$ of a distribution $p_{\mathcal{L}} \in \mathcal{P}$, which we call training data distribution.

Definition 6.1 *Consider $\mathcal{X} \subset \mathbb{R}^d$ bounded and $\mathcal{L}_{\mathcal{X}} = \{\boldsymbol{x}_j\}_{j=1}^b \subset \mathcal{X}$, set of locations of data points sampled from $p_{\mathcal{X}_{\mathcal{L}}}$, marginal on $\mathcal{X}$ of a distribution $p_{\mathcal{L}} \in \mathcal{P}$. Moreover, consider $p_{\mathcal{X}_{\mathcal{D}}}$ marginal on $\mathcal{X}$ of a distribution $p_{\mathcal{D}} \in \mathcal{P}$. We define the* weighted fill distance *of $\mathcal{L}_{\mathcal{X}}$ in $\mathcal{X}$ with respect to $p_{\mathcal{X}_{\mathcal{D}}}$ as*

$$W_{\mathcal{L}_{\mathcal{X}},\mathcal{X}}\left(p_{\mathcal{X}_{\mathcal{L}}} || p_{\mathcal{X}_{\mathcal{D}}}\right) := \sup_{\boldsymbol{x}\in\mathcal{X}} \min_{\boldsymbol{x}_j\in\mathcal{L}_{\mathcal{X}}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \psi_{\mathcal{L}_{\mathcal{X}}}(\boldsymbol{x}), \tag{6.2}$$

where $\|\cdot\|_2$ is the L_2-norm and the weight function $\psi_{\mathcal{L}_{\mathcal{X}}} : \mathcal{X} \to \mathbb{R}$ is defined as

$$\psi_{\mathcal{L}_{\mathcal{X}}}(\boldsymbol{x}) := \begin{cases} 1 - \frac{p_{\mathcal{X}_{\mathcal{L}}}(\boldsymbol{x})}{p_{\mathcal{X}_{\mathcal{D}}}(\boldsymbol{x})} & \text{if } p_{\mathcal{X}_{\mathcal{D}}}(\boldsymbol{x}) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

with $p_{\mathcal{X}_{\mathcal{L}}}(\boldsymbol{x}) := \int_{\mathcal{Y}} p_{\mathcal{L}}(\boldsymbol{x}, y)dy$ and $p_{\mathcal{X}_{\mathcal{D}}}(\boldsymbol{x}) := \int_{\mathcal{Y}} p_{\mathcal{D}}(\boldsymbol{x}, y)dy$.

The weighted fill distance depends on the distance metric and on how we define the weight function $\psi_{\mathcal{L}_{\mathcal{X}}}$. Here, the L_2-distance is considered, but the following results can be generalized to other distances. Moreover, we make a specific choice for the weight function $\psi_{\mathcal{L}_{\mathcal{X}}}$, aiming to take into account the relationship between $p_{\mathcal{X}_{\mathcal{D}}}$ and $p_{\mathcal{X}_{\mathcal{L}}}$. The weight

function has higher positive values in regions of the feature space where the marginal of the training data distribution ($p_{\mathcal{X}_\mathcal{L}}$) underestimates the marginal of the data source distribution ($p_{\mathcal{X}_\mathcal{D}}$). On the contrary, it has lower negative values where $p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) > p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$. Note that the weighted fill distance is non-negative as we show in the following remark.

Remark 6.1 *The weighed fill distance defined in (6.2) is non-negative.*

Proof. In the definition of the weighted fill distance in (6.2) the distances are by definition non-negative. Therefore, it is enough to show that there always exists a point $\boldsymbol{x} \in \mathcal{X}$ such that $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) \geq p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})$, that is, there always exists a point where the weight function is non-negative. Let us proceed by contradiction. Let us assume that $\forall\, \boldsymbol{x} \in \mathcal{X}$ we have that $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) < p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})$. Next, let us note that $\int_\mathcal{X} p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})d\boldsymbol{x} = \int_\mathcal{X} p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})d\boldsymbol{x} = 1$, which follows from the fact that both, $p_{\mathcal{X}_\mathcal{D}}$ and $p_{\mathcal{X}_\mathcal{L}}$ are probability distributions on $\mathcal{X}$. However, by our assumption we have that

$$1 = \int_\mathcal{X} p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})d\boldsymbol{x} < \int_\mathcal{X} p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})d\boldsymbol{x} = 1,$$

which is a contradiction. □

Moreover, the weighted fill distance is zero only in two scenarios: First, if $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) = p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})\ \forall\ \boldsymbol{x} \in \mathcal{X}$. In this case, the weight function would always be zero. Second, in the case that $\mathcal{L}_\mathcal{X}$ contains all the points in the non-zero support of $p_{\mathcal{X}_\mathcal{D}}$ where $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) > p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})$, that is, $\{\boldsymbol{x} \in \mathcal{X}$ such that $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) \neq 0$ and $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) > p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})\} \subset \mathcal{L}_\mathcal{X}$. Simply put, the weighted fill distance considers both point distances and distributional differences, and it provides a way to quantify how well the features of points in $\mathcal{L}_\mathcal{X}$ selected from $p_{\mathcal{X}_\mathcal{L}}$ represent the underlying distribution of data points features arising from $p_{\mathcal{X}_\mathcal{D}}$.

We use Assumption 5.1 and Assumption 5.2 from Chapter 5 for our theoretical analysis. In particular, we consider the assumptions to be valid also for $p_\mathcal{D}$ and $p_\mathcal{L}$. Recall that, the first assumption states that given the data feature location $\boldsymbol{X} = \boldsymbol{x}_i$, the associated label value, y_i, is close to the conditional expected value of the random variable Y at that location. This assumption models scenarios where the true feature-label relationship is stochastic or deterministic but subject to random fluctuations with a magnitude parameterized by a fixed scalar ϵ. Further, a Lipschitz continuity assumption relates to the smoothness of the map connecting $\mathcal{X}$ and $\mathcal{Y}$. It implies that when two data points are closer in $\mathcal{X}$, their corresponding labels tend to be closer in $\mathcal{Y}$. The second assumption essentially states that the expected error on the training set is bounded and that standard regularity assumptions on the error function and the model hold in the form of Lipschitz continuity. With that, we are able to establish an upper bound for the optimization objective in (6.1).

Theorem 6.1 *Consider random variables $(\boldsymbol{X}, Y)$ taking value on $\mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^d \times \mathbb{R}$, with $\mathcal{X}$ bounded, data source distribution $p_\mathcal{D} \in \mathcal{P}$, labeled dataset $\mathcal{L} := \{(\boldsymbol{x}_j, y_j)\}_{j=1}^b$ arising from training data distribution $p_\mathcal{L} \in \mathcal{P}$, regression model $m_\mathcal{L} \in \mathcal{M}$ trained on $\mathcal{L}$, and*

error function $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{M} \to \mathbb{R}^+$. *If Assumptions 5.1 and 5.2 are fulfilled, then we have that*

$$\mathbb{E}_{p_{\mathcal{D}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right] \leq C W_{\mathcal{L}_{\mathcal{X}},\mathcal{X}}\left(p_{\mathcal{X}_{\mathcal{L}}} || p_{\mathcal{X}_{\mathcal{D}}}\right) + \underbrace{\lambda_{l_{\mathcal{Y}}}\epsilon}_{\substack{\text{labels} \\ \text{uncertainty}}} + \underbrace{\epsilon_{\mathcal{L}}}_{\substack{\text{max error} \\ \text{training set}}} + \underbrace{\mathbb{E}_{p_{\mathcal{L}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right]}_{\substack{\text{expected error} \\ \text{training distribution}}} \tag{6.4}$$

where $C := \left(\lambda_{l_{\mathcal{X}}} + \lambda_{l_{\mathcal{Y}}}\lambda_p\right)$. λ_p *Lipschitz constant and* ϵ *labels' uncertainty from Assumption 5.1.* $\lambda_{l_{\mathcal{X}}}$ *and* $\lambda_{l_{\mathcal{Y}}}$ *are Lipschitz constants of the error function from Assumption 5.2.* $\epsilon_{\mathcal{L}}$ *is the maximum error of* $m_{\mathcal{L}}$ *on the training set, from Assumption 5.2.* $W_{\mathcal{L}_{\mathcal{X}},\mathcal{X}}\left(p_{\mathcal{X}_{\mathcal{L}}} || p_{\mathcal{X}_{\mathcal{D}}}\right)$ *weighted fill distance defined as in (6.2). Moreover,* $p_{\mathcal{X}_{\mathcal{D}}}$ *and* $p_{\mathcal{X}_{\mathcal{L}}}$ *are the marginals of* $p_{\mathcal{D}}$ *and* $p_{\mathcal{L}}$*, respectively, defined on* $\mathcal{X}$.

Proof. First, let us notice that

$$\begin{aligned}
&\mathbb{E}_{p_{\mathcal{D}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right] \\
&\quad = \mathbb{E}_{p_{\mathcal{D}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right] - \mathbb{E}_{p_{\mathcal{L}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right] + \mathbb{E}_{p_{\mathcal{L}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right] \\
&\quad = \int_{\mathcal{X}} \mathbb{E}\left[l(\boldsymbol{x}, Y, m_{\mathcal{L}})|\boldsymbol{x}\right] p_{\mathcal{X}_{\mathcal{D}}}(\boldsymbol{x}) d\boldsymbol{x} - \int_{\mathcal{X}} \mathbb{E}\left[l(\boldsymbol{x}, Y, m_{\mathcal{L}})|\boldsymbol{x}\right] p_{\mathcal{X}_{\mathcal{L}}}(\boldsymbol{x}) d\boldsymbol{x} + \mathbb{E}_{p_{\mathcal{L}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right] \\
&\quad \leq \int\limits_{\substack{\mathcal{X}, \\ p_{\mathcal{X}_{\mathcal{D}}} \geq p_{\mathcal{X}_{\mathcal{L}}}}} \mathbb{E}\left[l(\boldsymbol{x}, Y, m_{\mathcal{L}})|\boldsymbol{x}\right] \left(p_{\mathcal{X}_{\mathcal{D}}}(\boldsymbol{x}) - p_{\mathcal{X}_{\mathcal{L}}}(\boldsymbol{x})\right) d\boldsymbol{x} + \mathbb{E}_{p_{\mathcal{L}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right].
\end{aligned} \tag{6.5}$$

Note that in (6.5) the expectation $\mathbb{E}\left[l(\boldsymbol{x}, Y, m_{\mathcal{L}})|\boldsymbol{x}\right]$ is independent of $p_{\mathcal{D}}$ and $p_{\mathcal{L}}$. This is because, as mentioned in Section 6.1, we consider a scenario where $p_{\mathcal{D}}$ may differ from $p_{\mathcal{L}}$ but the map connecting a data location $\boldsymbol{x} \in \mathcal{X}$ and its associated label value is independent on how the data is selected, that is, $p_{\mathcal{D}} \neq p_{\mathcal{L}}$ and $p_{\mathcal{D}}(y|\boldsymbol{x}) = p_{\mathcal{L}}(y|\boldsymbol{x})$. In what follows we define $p(y|\boldsymbol{x}) := p_{\mathcal{D}}(y|\boldsymbol{x}) = p_{\mathcal{L}}(y|\boldsymbol{x})$. Next, fixed $\boldsymbol{X} = \tilde{\boldsymbol{x}} \in \mathcal{X}$, we want to bound $\mathbb{E}\left[l(\tilde{\boldsymbol{x}}, Y, m_{\mathcal{L}})|\tilde{\boldsymbol{x}}\right]$. To do that we use a result from the proof of Theorem 5.1: For fixed $\tilde{\boldsymbol{x}} \in \mathcal{X}$ and $\boldsymbol{x}_j \in \mathcal{L}_{\mathcal{X}}$ we have

$$\begin{aligned}
\mathbb{E}\left[l(\tilde{\boldsymbol{x}}, Y, m_{\mathcal{L}})|\tilde{\boldsymbol{x}}\right] &= \int_{\mathcal{Y}} l(\tilde{\boldsymbol{x}}, y, m_{\mathcal{L}}) p(y|\tilde{\boldsymbol{x}}) dy \\
&\leq \int_{\mathcal{Y}} \left|l(\tilde{\boldsymbol{x}}, y, m_{\mathcal{L}}) - l(\boldsymbol{x}_j, y, m_{\mathcal{L}})\right| p(y|\tilde{\boldsymbol{x}}) dy + \int_{\mathcal{Y}} l(\boldsymbol{x}_j, y, m_{\mathcal{L}}) p(y|\tilde{\boldsymbol{x}}) dy \\
&\leq \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_j\|_2 \lambda_{l_{\mathcal{X}}} + \int_{\mathcal{Y}} l(\boldsymbol{x}_j, y, m_{\mathcal{L}}) p(y|\tilde{\boldsymbol{x}}) dy
\end{aligned} \tag{6.6}$$

The second inequality in (6.6) follows from the $\lambda_{l_{\mathcal{X}}}$-Lipschitz continuity of the error function. We can bound the remaining term as follows

$$\int_{\mathcal{Y}} l(\boldsymbol{x}_j, y, m_{\mathcal{L}}) p(y|\tilde{\boldsymbol{x}}) dy$$

$$
\begin{aligned}
&\leq \int_{\mathcal{Y}} |l(\boldsymbol{x}_j, y, m_{\mathcal{L}}) - l(\boldsymbol{x}_j, \mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right], m_{\mathcal{L}})|\, p(y|\tilde{\boldsymbol{x}}) dy \\
&\quad + \int_{\mathcal{Y}} |l(\boldsymbol{x}_j, \mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right], m_{\mathcal{L}}) - l(\boldsymbol{x}_j, \mathbb{E}\left[Y|\boldsymbol{x}_j\right], m_{\mathcal{L}})|\, p(y|\tilde{\boldsymbol{x}}) dy \\
&\quad + \int_{\mathcal{Y}} l(\boldsymbol{x}_j, \mathbb{E}\left[Y|\boldsymbol{x}_j\right], m_{\mathcal{L}}) p(y|\tilde{\boldsymbol{x}}) dy \\
&\leq \lambda_{l_{\mathcal{Y}}} \int_{\mathcal{Y}} |y - \mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right]|\, p(y|\tilde{\boldsymbol{x}}) dy \\
&\quad + \lambda_{l_{\mathcal{Y}}} \int_{\mathcal{Y}} |\mathbb{E}\left[Y|\tilde{\boldsymbol{x}}\right] - \mathbb{E}\left[Y|\boldsymbol{x}_j\right]|\, p(y|\tilde{\boldsymbol{x}}) dy \\
&\quad + \int_{\mathcal{Y}} \mathbb{E}[l(\boldsymbol{x}_j, Y, m_{\mathcal{L}})\big|\boldsymbol{x}_j] p(y|\tilde{\boldsymbol{x}}) dy \\
&\leq \lambda_{l_{\mathcal{Y}}} \epsilon + \lambda_{l_{\mathcal{Y}}} \int_{\mathcal{Y}} (\lambda_p \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_j\|_2)\, p(y|\tilde{\boldsymbol{x}}) dy + \int_{\mathcal{Y}} \epsilon_{\mathcal{L}}\, p(y|\tilde{\boldsymbol{x}}) dy \\
&\leq \lambda_{l_{\mathcal{Y}}} \epsilon + \lambda_{l_{\mathcal{Y}}} \lambda_p \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_j\|_2 + \epsilon_{\mathcal{L}}.
\end{aligned}
$$

Similarly to the proof of Theorem 5.1, the second inequality follows from the $\lambda_{l_{\mathcal{Y}}}$-Lipschitz continuity of the error function and Jensen's inequality, which is used to obtain the conditional expectation in the integrand of the last term. The third inequality follows from the definition of labels' uncertainty, the λ_p-Lipschitz continuity of the conditional expectation of the random variable Y and the assumption that the expected error on the training set is bounded by $\epsilon_{\mathcal{L}}$. The fourth inequality is obtained by taking out the constants from the integrals in the second and third terms and noticing that, from the definition of $p(y|\tilde{\boldsymbol{x}})$, we have $\int_{\mathcal{Y}} p(y|\tilde{\boldsymbol{x}}) dy = 1$.

By taking the minimum over $\boldsymbol{x}_j \in \mathcal{L}_{\mathcal{X}}$ we get

$$
\mathbb{E}\left[l(\tilde{\boldsymbol{x}}, Y, m_{\mathcal{L}})|\tilde{\boldsymbol{x}}\right] \leq \min_{\boldsymbol{x}_j \in \mathcal{L}_{\mathcal{X}}} \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_j\|_2 \left(\lambda_{l_{\mathcal{X}}} + \lambda_{l_{\mathcal{Y}}} \lambda_p\right) + \lambda_{l_{\mathcal{Y}}} \epsilon + \epsilon_{\mathcal{L}}. \tag{6.7}
$$

Next, we define $C := \left(\lambda_{l_{\mathcal{X}}} + \lambda_{l_{\mathcal{Y}}} \lambda_p\right)$ and apply (6.7) to (6.5). Consequently, we have

$$
\begin{aligned}
&\mathbb{E}_{p_{\mathcal{D}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right] \\
&\quad \leq \int\limits_{\substack{\mathcal{X}, \\ p_{\mathcal{X}_{\mathcal{D}}} \geq p_{\mathcal{X}_{\mathcal{L}}}}} \left(\min_{\boldsymbol{x}_j \in \mathcal{L}_{\mathcal{X}}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 C + \lambda_{l_{\mathcal{Y}}} \epsilon + \epsilon_{\mathcal{L}}\right) \left(p_{\mathcal{X}_{\mathcal{D}}}(\boldsymbol{x}) - p_{\mathcal{X}_{\mathcal{L}}}(\boldsymbol{x})\right) d\boldsymbol{x} + \mathbb{E}_{p_{\mathcal{L}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right] \\
&\quad = \int\limits_{\substack{\mathcal{X}, \\ p_{\mathcal{X}_{\mathcal{D}}} \geq p_{\mathcal{X}_{\mathcal{L}}}}} \min_{\boldsymbol{x}_j \in \mathcal{L}_{\mathcal{X}}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 C \left(p_{\mathcal{X}_{\mathcal{D}}}(\boldsymbol{x}) - p_{\mathcal{X}_{\mathcal{L}}}(\boldsymbol{x})\right) d\boldsymbol{x} \\
&\qquad + \int\limits_{\substack{\mathcal{X}, \\ p_{\mathcal{X}_{\mathcal{D}}} \geq p_{\mathcal{X}_{\mathcal{L}}}}} \left(\lambda_{l_{\mathcal{Y}}} \epsilon + \epsilon_{\mathcal{L}}\right) \left(p_{\mathcal{X}_{\mathcal{D}}}(\boldsymbol{x}) - p_{\mathcal{X}_{\mathcal{L}}}(\boldsymbol{x})\right) d\boldsymbol{x} + \mathbb{E}_{p_{\mathcal{L}}}\left[l(\boldsymbol{X}, Y, m_{\mathcal{L}})\right]
\end{aligned}
$$

$$
\begin{aligned}
&\leq \int_{\substack{\mathcal{X}, \\ p_{\mathcal{X}_\mathcal{D}} \geq p_{\mathcal{X}_\mathcal{L}}}} \min_{\boldsymbol{x}_j \in \mathcal{L}_\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 C \left(p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) - p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})\right) d\boldsymbol{x} \\
&\quad + (\lambda_{l_\mathcal{Y}} \epsilon + \epsilon_\mathcal{L}) \int_\mathcal{X} p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) d\boldsymbol{x} + \mathbb{E}_{p_\mathcal{L}} \left[l(\boldsymbol{X}, Y, m_\mathcal{L})\right] \\
&\leq \int_{\substack{\mathcal{X}, \\ p_{\mathcal{X}_\mathcal{D}} \geq p_{\mathcal{X}_\mathcal{L}} \\ p_{\mathcal{X}_\mathcal{D}} > 0}} \min_{\boldsymbol{x}_j \in \mathcal{L}_\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 C \left(1 - \frac{p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})}{p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})}\right) p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) d\boldsymbol{x} + (\lambda_{l_\mathcal{Y}} \epsilon + \epsilon_\mathcal{L}) + \mathbb{E}_{p_\mathcal{L}} \left[l(\boldsymbol{X}, Y, m_\mathcal{L})\right] \\
&\leq C \sup_{\boldsymbol{x} \in \mathcal{X}} \left(\min_{\boldsymbol{x}_j \in \mathcal{L}_\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \left(1 - \frac{p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})}{p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})}\right)\right) + \lambda_{l_\mathcal{Y}} \epsilon + \epsilon_\mathcal{L} + \mathbb{E}_{p_\mathcal{L}} \left[l(\boldsymbol{X}, Y, m_\mathcal{L})\right] \\
&= C W_{\mathcal{L}_\mathcal{X}, \mathcal{X}} \left(p_{\mathcal{X}_\mathcal{L}} || p_{\mathcal{X}_\mathcal{D}}\right) + \lambda_{l_\mathcal{Y}} \epsilon + \epsilon_\mathcal{L} + \mathbb{E}_{p_\mathcal{L}} \left[l(\boldsymbol{X}, Y, m_\mathcal{L})\right].
\end{aligned}
$$

The last inequality follows from taking the supremum over $\boldsymbol{x} \in \mathcal{X}$, taking the constant terms out of the integral and noticing that $\int_{\substack{\mathcal{X}, \\ p_{\mathcal{X}_\mathcal{D}} \geq p_{\mathcal{X}_\mathcal{L}} \\ p_{\mathcal{X}_\mathcal{D}} > 0}} p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) d\boldsymbol{x} \leq \int_\mathcal{X} p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) d\boldsymbol{x} = 1$. □

Theorem 6.1 provides a qualitative upper bound for the expected value of the error function on the data source distribution $p_\mathcal{D}$ depending linearly on the weighted fill distance of the selected training set. Our goal is to design a training data distribution marginal $p_{\mathcal{X}_\mathcal{L}}$ and sample a training set $\mathcal{L}_\mathcal{X}$ to attempt to minimize the only component of the bound that does not involve the data labels and that we can attempt to estimate before training a regression model: the weighted fill distance.

Note that, the bound in (6.4) depends on four main quantities: the maximum error on the training set ($\epsilon_\mathcal{L}$), which we can only compute after training a regression model, the label uncertainty (ϵ), the expected error on the training data distribution ($\mathbb{E}_{p_\mathcal{L}} \left[l(\boldsymbol{X}, Y, m_\mathcal{L})\right]$), which we may not know or be able to compute, and the weighted fill distance, which contrarily to the previous three quantities does not depend on the data labels. The expected error on the training data distribution and the maximum error on the training set indicate that the bound also depends on how well the unknown trained model fits the training data distribution. We do not consider minimizing these quantities because we can not estimate them without training a regression model. We are considering a passive and model-agnostic sampling scenario where we have no knowledge about the data labels and the regression model at the time of selection. In what follows, we work under the general assumption that the employed trained model works well on the training data and these quantities to be negligible. A similar assumption has been considered in previous work [JKW+23]. Note that, under these assumptions, the smaller the weighted fill distance of the selected training set, the smaller the bound for the expected approximation error on data from the source distribution $p_\mathcal{D}$.

Intuitively, a sampling procedure aiming at selecting training sets by minimizing the weighted fill distance should consider how each data point in $\mathcal{X}$ influences its value and select those that minimize it. From (6.2), we know that for each $\boldsymbol{x} \in \mathcal{X}$ the weighted

fill distance considers two quantities: the value of the weight function, $\psi_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})$, and the data point's distance to its closest selected point in $\mathcal{L}_\mathcal{X}$. For each data point, the distance to its nearest selected point in $\mathcal{L}_\mathcal{X}$ tends to be smaller in regions where $p_{\mathcal{X}_\mathcal{L}}$ has higher values. This is because the points in $\mathcal{L}_\mathcal{X}$ are drawn from $p_{\mathcal{X}_\mathcal{L}}$. Consequently, $\mathcal{L}_\mathcal{X}$ is more likely to include data points from regions of the feature space where $p_{\mathcal{X}_\mathcal{L}}$ is larger, thereby reducing the distance factor for points in those areas. Furthermore, the weight function $\psi_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})$ (defined in (6.3)) takes higher values in regions of the feature space where $p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) \leq p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$. As a result, the weighted distances tend to be larger in regions where $p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) \leq p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$ and that are not well represented by the points in the selected set $\mathcal{L}_\mathcal{X}$. Since the weighted fill distance is defined as the supremum of these weighted distances, a sampling procedure that seeks to minimize it should either include in the training sets data points where $p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) \leq p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$ or consider a training data distribution that assigns to them greater importance.

In the next section we propose one possible sampling approach attempting to minimize the weighted fill distance of the selected set.

6.3. Density-Aware Farthest Point Sampling (DA-FPS)

In this section we design a data selection procedure aiming at minimizing the weighted fill distance of the selected set: $W_{\mathcal{L}_\mathcal{X},\mathcal{X}}\left(p_{\mathcal{X}_\mathcal{L}} || p_{\mathcal{X}_\mathcal{D}}\right)$. Unfortunately, given a set $\mathcal{L}_\mathcal{X} \subset \mathcal{X}$, its associated weighted fill distance can not be directly computed as the weights explicitly depend on $p_{\mathcal{X}_\mathcal{D}}$, and we are considering scenarios in which we do not know the data source distribution $p_\mathcal{D}$, and therefore its marginal, $p_{\mathcal{X}_\mathcal{D}}$. Still, we have access to a set $\mathcal{D}_\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ of unlabeled data points arising from $p_{\mathcal{X}_\mathcal{D}}$. Thus, we first compute an estimation $\hat{p}_{\mathcal{X}_\mathcal{D}}$ of $p_{\mathcal{X}_\mathcal{D}}$ from the unlabeled set $\mathcal{D}_\mathcal{X}$. Subsequently, we select a training set

Algorithm 6 Density-Aware Farthest Point Sampling (DA-FPS)

Input Dataset $\mathcal{D}_\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{X}$, data budget $b \in \mathbb{N}$, neighborhood size $k \in \mathbb{N}$, with $b,\ k \ll n$. Set $\mathcal{L}_\mathcal{X} \subset D_\mathcal{X}$ with $|\mathcal{L}_\mathcal{X}| \ll b$, $u \in \mathbb{N}$ with $u < b$
Output Subset $\mathcal{L}_\mathcal{X} \subset D_\mathcal{X}$ with $|\mathcal{L}_\mathcal{X}| = b$.

if $\mathcal{L}_\mathcal{X} = \emptyset$ **then**
    Choose $\hat{\boldsymbol{x}} \in \mathcal{D}_\mathcal{X}$ randomly and set $\mathcal{L}_\mathcal{X} = \{\hat{\boldsymbol{x}}\}$
while $|\mathcal{L}_\mathcal{X}| < b$ **do**
    if $|\mathcal{L}_\mathcal{X}| < u$ **then**
        $\bar{\boldsymbol{x}} = \underset{x \in \mathcal{D}_\mathcal{X}}{\arg\max} \left[\underset{x_j \in \mathcal{L}_\mathcal{X}}{\min} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \right]$.
    else
        Compute $\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})\ \forall\ \boldsymbol{x} \in \mathcal{D}_\mathcal{X} \backslash \mathcal{L}_\mathcal{X}$ as in (6.11).
        $\bar{\boldsymbol{x}} = \underset{x \in \mathcal{D}_\mathcal{X}}{\arg\max} \left[\underset{x_j \in \mathcal{L}_\mathcal{X}}{\min} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x}) \right]$.
    $\mathcal{L}_\mathcal{X} \leftarrow \mathcal{L}_\mathcal{X} \cup \bar{\boldsymbol{x}}$.

$\mathcal{L}_\mathcal{X} = \{\boldsymbol{x}_j\}_{j=1}^b \subset \mathcal{D}_\mathcal{X}$ determining an estimation $\hat{p}_{\mathcal{X}_\mathcal{L}}$ of the marginal of the training data distribution $p_{\mathcal{X}_\mathcal{L}}$. The selection procedure we propose aims at selecting a set $\mathcal{L}_\mathcal{X}$ that minimizes a data-driven estimation of the weighted fill distance based on the dataset $\mathcal{D}_\mathcal{X}$, the selected subset $\mathcal{L}_\mathcal{X}$ and the two estimated marginals $\hat{p}_{\mathcal{X}_\mathcal{D}}$ and $\hat{p}_{\mathcal{X}_\mathcal{L}}$.

Before diving into the technicalities of our proposed sampling method, let us briefly describe its core procedure. Our approach involves iteratively sampling data points at a maximal weighted distance from those already selected. Initially the weights are set to one for all available points, thus, in this initial stage, our procedure coincides with FPS. After a portion of the data has been selected, the weights adapt dynamically during the selection process according to the chosen training set. The weights reflect the relationship between $\hat{p}_{\mathcal{X}_\mathcal{D}}$ and $\hat{p}_{\mathcal{X}_\mathcal{L}}$, which we estimate from the data locations. For this reason we call our approach Density-Aware Farthest Point Sampling (DA-FPS). The weights dynamically change because the estimation of the marginal of the training data distribution, $\hat{p}_{\mathcal{X}_\mathcal{L}}$, is data-driven and depends on the current training set. As we add new points to the selected set, $\hat{p}_{\mathcal{X}_\mathcal{L}}$ must be updated.

To estimate the multivariate marginal distributions $p_{\mathcal{X}_\mathcal{D}}$ and $p_{\mathcal{X}_\mathcal{L}}$, we follow along [WKV09] and choose an adaptive k-nearest-neighbor (kNN) density estimation approach. Note that, we opted for kNN density estimators over other strategies for density or density ratio estimation because they are computationally efficient. Using kNNs, we can estimate the density at a specific data point by only considering its distance relationships with its k-nearest neighbors, without having to perform additional comparisons with other points in the data set. This reduces the computational costs involved in the iterative process of density estimation. The multivariate kNN density estimations of marginal distributions, $p_{\mathcal{X}_\mathcal{L}}$ and $p_{\mathcal{X}_\mathcal{D}}$, are based on a selected set $\mathcal{L}_\mathcal{X}$, the dataset $\mathcal{D}_\mathcal{X}$ and a kernel function $K : \mathbb{R}^d \to \mathbb{R}^+$. At a point $\boldsymbol{x} \in \mathcal{X}$ the estimated densities have the form

$$\hat{p}^k_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) := \frac{\sum\limits_{\boldsymbol{x}_i \in \mathcal{D}_\mathcal{X}} K\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})}\right)}{|\mathcal{D}_\mathcal{X}| \left(r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})\right)^d} \quad \text{and} \quad \hat{p}^k_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) := \frac{\sum\limits_{\boldsymbol{x}_j \in \mathcal{L}_\mathcal{X}} K\left(\frac{\boldsymbol{x}-\boldsymbol{x}_j}{r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})}\right)}{|\mathcal{L}_\mathcal{X}| \left(r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})\right)^d}, \tag{6.8}$$

where

$$r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x}) := \min\left\{\min_{\bar{\boldsymbol{x}} \in \mathcal{L}_\mathcal{X}} \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_2 + \frac{\epsilon_\mathcal{X}}{|\mathcal{L}_\mathcal{X}|}, \rho_k(\boldsymbol{x})\right\} \quad \text{and} \quad K(\boldsymbol{x}) := \begin{cases} \frac{1}{V_d}, & \text{if } \|\boldsymbol{x}\|_2 \leq 1 \\ 0, & \text{otherwise} \end{cases}, \tag{6.9}$$

with $\rho_k(\boldsymbol{x})$ the distance between $\boldsymbol{x}$ and its k-nearest neighbor in $\mathcal{D}_\mathcal{X}$, and $V_d := \frac{\pi^{d/2}}{\Gamma(d/2+1)}$ is the volume of the unit ball in $\mathbb{R}^d$, where $\Gamma(r) = (r-1)!$ is the so-called Gamma function. The quantity $r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})$ defines an adaptive neighborhood size, depending on the selected set $\mathcal{L}_\mathcal{X}$, aiming at reducing the estimation bias at finite sample sizes [WKV09]. $\epsilon_\mathcal{X}$ in the definition of $r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})$ is an arbitrary small positive scalar that prevents the denominator in the kNN density estimations from becoming zero on points in $\mathcal{L}_\mathcal{X}$. Note that, we are interested in estimating the densities only for points in $\mathcal{D}_\mathcal{X} \backslash \mathcal{L}_\mathcal{X} := \{\boldsymbol{x} \in \mathcal{D}_\mathcal{X} \text{ such that } \boldsymbol{x} \notin \mathcal{L}_\mathcal{X}\}$, which are not already in $\mathcal{L}_\mathcal{X}$, and we may want to select depending on the value they

would provide, which we quantify with the associated weighted distance. Nonetheless, the kernel estimations in (6.8) are well-defined for all points in $\mathcal{X}$. In Appendix A we show that, under some assumptions, $\hat{p}^k_{\mathcal{X}_\mathcal{L}}$ and $\hat{p}^k_{\mathcal{X}_\mathcal{D}}$ are asymptotically unbiased estimations of $p_{\mathcal{X}_\mathcal{L}}$ and $p_{\mathcal{X}_\mathcal{D}}$, respectively. Considering the density estimations in (6.8), given a set $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$, we compute a data-driven estimate of its weighted fill distance as

$$W_{\mathcal{L}_\mathcal{X},\mathcal{D}_\mathcal{X}}\left(p_{\mathcal{X}_\mathcal{L}}||p_{\mathcal{X}_\mathcal{D}}\right) \approx \max_{\boldsymbol{x}\in\mathcal{D}_\mathcal{X}\setminus\mathcal{L}_\mathcal{X}} \left[\left(1-\frac{\hat{p}^k_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})}{\hat{p}^k_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})}\right)\min_{\boldsymbol{x}_j\in\mathcal{L}_\mathcal{X}} \|\boldsymbol{x}-\boldsymbol{x}_j\|_2\right] \tag{6.10}$$

Unfortunately, the behavior of the estimated weight function in (6.10) is inconsistent with that of the true weight function in (6.3). To see this, let us consider $\mathcal{N}_k(\boldsymbol{x}) \subset \mathcal{D}_\mathcal{X}$ as the set of data points of $\mathcal{D}_\mathcal{X}$ in the k-neighborhood of $\boldsymbol{x}$, with $\boldsymbol{x} \in \mathcal{N}_k(\boldsymbol{x})$, and the weights

$$\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x}) := \begin{cases} |\{\bar{\boldsymbol{x}} \in \mathcal{D}_\mathcal{X} \text{ such that } \|\boldsymbol{x}-\bar{\boldsymbol{x}}\|_2 \leq r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})\}|, & \text{if } |\mathcal{L}_\mathcal{X}|>0 \\ k, & \text{if } \mathcal{L}_\mathcal{X}=\emptyset \end{cases} \tag{6.11}$$

as the number of points in $\mathcal{D}_\mathcal{X}$ contained in the ball centered in $\boldsymbol{x}$ with radius $r^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})$. For completeness, we set the weights associated with the empty set equal to k. Next, let us note that for each $\boldsymbol{x} \in \mathcal{X}$ we have

$$\left(1-\frac{\hat{p}^k_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x})}{\hat{p}^k_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})}\right) := \begin{cases} 1, & \text{if there is no } \boldsymbol{x}_j \in \mathcal{L}_\mathcal{X} \text{ s.t. } \boldsymbol{x}_j \in \mathcal{N}_k(\boldsymbol{x}) \\ 1-\frac{|\mathcal{D}_\mathcal{X}|}{\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})|\mathcal{L}_\mathcal{X}|}, & \text{otherwise} \end{cases} \tag{6.12}$$

Thus, depending on the ratio $\frac{|\mathcal{D}_\mathcal{X}|}{\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})|\mathcal{L}_\mathcal{X}|}$ and the value of k, the estimated weight function in (6.12) may allow for negative values on data points in $\mathcal{D}_\mathcal{X}\setminus\mathcal{L}_\mathcal{X}$, which have not been selected and included in $\mathcal{L}_\mathcal{X}$ where we would expect $\hat{p}^k_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) \leq \hat{p}^k_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$. This is an artifact of the approach we use to estimate the densities' ratio which is not consistent with the behavior of true weight function defined in (6.3), which associates positive weight values with points where $p_{\mathcal{X}_\mathcal{L}}(\boldsymbol{x}) \leq p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$. This inconsistency indicates that the approach we use to compute the densities' ratio may cause scaling issues for the values of the estimated weight function in (6.12). To address this inconsistency, we note that the values of the estimated weight function are directly correlated with those of $\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})$ defined in (6.11). Given $\tilde{\boldsymbol{x}}, \bar{\boldsymbol{x}} \in \mathcal{X}$ we have $\left(1-\frac{\hat{p}^k_{\mathcal{X}_\mathcal{L}}(\tilde{\boldsymbol{x}})}{\hat{p}^k_{\mathcal{X}_\mathcal{D}}(\tilde{\boldsymbol{x}})}\right) > \left(1-\frac{\hat{p}^k_{\mathcal{X}_\mathcal{L}}(\bar{\boldsymbol{x}})}{\hat{p}^k_{\mathcal{X}_\mathcal{D}}(\bar{\boldsymbol{x}})}\right) \Rightarrow \omega^k_{\mathcal{L}_\mathcal{X}}(\tilde{\boldsymbol{x}}) \geq \omega^k_{\mathcal{L}_\mathcal{X}}(\bar{\boldsymbol{x}})$ and $\omega^k_{\mathcal{L}_\mathcal{X}}(\tilde{\boldsymbol{x}}) > \omega^k_{\mathcal{L}_\mathcal{X}}(\bar{\boldsymbol{x}}) \Rightarrow \left(1-\frac{\hat{p}^k_{\mathcal{X}_\mathcal{L}}(\tilde{\boldsymbol{x}})}{\hat{p}^k_{\mathcal{X}_\mathcal{D}}(\tilde{\boldsymbol{x}})}\right) > \left(1-\frac{\hat{p}^k_{\mathcal{X}_\mathcal{L}}(\bar{\boldsymbol{x}})}{\hat{p}^k_{\mathcal{X}_\mathcal{D}}(\bar{\boldsymbol{x}})}\right)$. The weights $\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x})$ are directly associated with the data density around the data points as they quantify the number of points close to a given point. Moreover, they are not affected by issues related to the estimation of the densities' ratio. Based on these observations, instead of attempting the minimization of the quantity in (6.10) we consider the following minimization problem:

$$O_\mathcal{X} \in \arg\min_{\substack{\mathcal{L}_\mathcal{X}\subset\mathcal{D}_\mathcal{X} \\ |\mathcal{L}_\mathcal{X}|=b}} \left[\max_{\boldsymbol{x}\in\mathcal{D}_\mathcal{X}} \left(\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x}) \min_{\boldsymbol{x}_j\in\mathcal{L}_\mathcal{X}} \|\boldsymbol{x}-\boldsymbol{x}_j\|_2\right)\right]. \tag{6.13}$$

For a more compact notation, given a subset $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$, we define the quantity between the squared brackets as follows

$$W^k_{\mathcal{L}_\mathcal{X},\mathcal{D}_\mathcal{X}} := \max_{\boldsymbol{x}\in\mathcal{D}_\mathcal{X}} \left(\omega^k_{\mathcal{L}_\mathcal{X}}(\boldsymbol{x}) \min_{\boldsymbol{x}_j\in\mathcal{L}_\mathcal{X}} \|\boldsymbol{x}-\boldsymbol{x}_j\|_2 \right) \tag{6.14}$$

and refer to it as estimated weighted fill distance of $\mathcal{L}_\mathcal{X}$ in $\mathcal{D}_\mathcal{X}$. The value k in (6.14) is the amount of nearest neighbors we consider to compute the distance weights as in (6.11). The optimization problem in (6.13) is a weighted version of the fill distance minimization problem and is therefore at least NP hard. To address the optimization problem in (6.13) we propose the novel DA-FPS described in Algorithm 6. DA-FPS takes in input a finite dataset $\mathcal{D}_\mathcal{X} \subset \mathbb{R}^d$, a data budget $b \in \mathbb{N}$, a neighborhood size $k \in \mathbb{N}$, with $b,\ k \ll n$, a subset $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$, with $|\mathcal{L}_\mathcal{X}| \ll b$ and an additional hyperparameter $u \in \mathbb{N}$, $u < b$. The input hyperparameter $u \in \mathbb{N}$ allows for a greedy selection with uniform weights until the size of the selected set reaches the value described by the hyperparameter. Later on, the objective of our algorithm is to greedily augment the input subset $\mathcal{L}_\mathcal{X}$ by selecting points from the dataset $\mathcal{D}_\mathcal{X}$ so that the maximum weighted distance of any point in $\mathcal{D}_\mathcal{X}$ to its nearest selected point in $\mathcal{L}_\mathcal{X}$ is minimized. The algorithm stops when the number of elements in the selected set $\mathcal{L}_\mathcal{X}$ reaches the data budget b. After that, $\mathcal{L}_\mathcal{X}$ is provided as output.

6.3.1. An illustrative example of DA-FPS sampling

We provide a simple example to illustrate how DA-FPS sampling works, comparing it with uniform random sampling and FPS. The top-left graph in Fig. 6.1 shows a dataset of 1200 two-dimensional points. The dataset contains a high-density cluster with 600 points and three sparser clusters of 200 points each, differing in both point density and surface area coverage. The top-right graph displays a 2D kernel density estimation (KDE) plot of the points in the dataset. To generate the plot we use the Seaborn python library [Was21]. In the KDE plot, darker blue areas indicate higher data density, and lighter areas indicate lower data density. The graphs in the second, third, and fourth rows of Fig. 6.1 show the locations and distributions of points in subsets selected using uniform random sampling (RDM), FPS, and DA-FPS, respectively. For each sampling approach, we consider three subset sizes (5%, 10% and 20% of the available data points), with each larger subset including the smaller ones selected by the same approach. We initialized DA-FPS with $u = 0$ and $k = 300$.

From a global perspective, DA-FPS balances two objectives: covering the data space evenly while giving greater importance to higher-density regions. Consequently, for all subset sizes, the sampled data approximates the original distribution but with reduced density differences across the data space. More specifically DA-FPS initially provides a rough estimation of the data density distribution, while ensuring all regions are represented, as seen in the 5% sampling plot. Trivially, as the subset size increases, the distribution of sampled points more closely resembles that of the full dataset. Still, the differences in density across regions are less pronounced than in the true data distribution.

In contrast, FPS consistently selects points to provide a uniform representation of the data distribution, while RDM mainly focuses on the high-density cluster, neglecting sparser regions.

Locally, both FPS and DA-FPS tend to select points that are evenly distributed without clustering. This is due to their optimization processes, which lead to maximize pairwise (weighted) distances between the selected points, creating a "repulsion effect".

One of the key advantages of DA-FPS for sampling training sets lies in its ability to balance the representation of both dense and sparse regions in the data. By ensuring that high-density regions are well-represented while also covering sparser areas, DA-FPS mitigates the risk of over-fitting or under-fitting to dominant clusters, which is a common issue when using random sampling and FPS, respectively.

Said that, we recall that sampling strategies are not inherently good or bad. The effectiveness of a sampling approach depends on the specific context and goals of the task. While DA-FPS balances the representation of dense and sparse regions, we can encounter scenarios where RDM or methods that closely reflect the original data distribution may be better suited. For instance, when the data distribution is uniform, DA-FPS may add unnecessary complexity. Additionally, for very small training sets, or very simple distributions, e.g., a dataset with only one high-density cluster and a few outliers, RDM may better reflect the original distribution, while DA-FPS may over-represent the less impactful regions populated by the outliers. Nonetheless, in Chapter 7 we provide examples on various datasets and with different regression models where using DA-FPS for training data selection leads to better average performances compared to various baselines, including RDM and FPS.

6.4. Analysis of DA-FPS

This section studies how well the proposed DA-FPS algorithm can address the optimization problem in (6.13). In particular, the following theorem shows that, if we initialize Algorithm 6 with $\mathcal{L}_\mathcal{X} = \emptyset$ and $u = 0$, it provides a $2k$-approximation to the optimization problem in (6.13), where k is the amount of nearest neighbors we consider to estimate the densities in (6.8) given in input to the algorithm.

Theorem 6.2 *Given set of data locations $\mathcal{D}_\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, subset $O_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$, optimal solution to the problem in (6.13) with $|O_\mathcal{X}| = b \in \mathbb{N}^+$, $b < n$, and $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$, $|\mathcal{L}_\mathcal{X}| = b$, subset selected with Algorithm 6 initialized with $\mathcal{L}_\mathcal{X} = \emptyset$ and $u = 0$, we have*

$$W^k_{\mathcal{L}_\mathcal{X}, \mathcal{D}_\mathcal{X}} \leq 2k W^k_{O_\mathcal{X}, \mathcal{D}_\mathcal{X}}, \tag{6.15}$$

where $W^k_{\mathcal{L}_\mathcal{X}, \mathcal{D}_\mathcal{X}}$ and $W^k_{O_\mathcal{X}, \mathcal{D}_\mathcal{X}}$ are the estimated weighted fill distances of $\mathcal{L}_\mathcal{X}$ and $O_\mathcal{X}$ in $\mathcal{D}_\mathcal{X}$, respectively, defined as in (6.14).

Proof. Let us fix the budget $b \in \mathbb{N}^+$, and define, for each $i = 1, \dots, b+1$, $\mathcal{L}_i := \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_i\}$ as the set of cardinality i obtained after $i-1$ iterations of the "While" loop in line 3 of Algorithm 6. To simplify the notation, for the rest of the proof, for each $\mathcal{L}_\mathcal{X} \subset \mathcal{D}_\mathcal{X}$ we define $W_{\mathcal{L}_\mathcal{X}} := W^k_{\mathcal{L}_\mathcal{X}, \mathcal{D}_\mathcal{X}}$ omitting the k and $\mathcal{D}_\mathcal{X}$ from the notation of the estimated weighted fill distance. The following proof consists of three main steps.

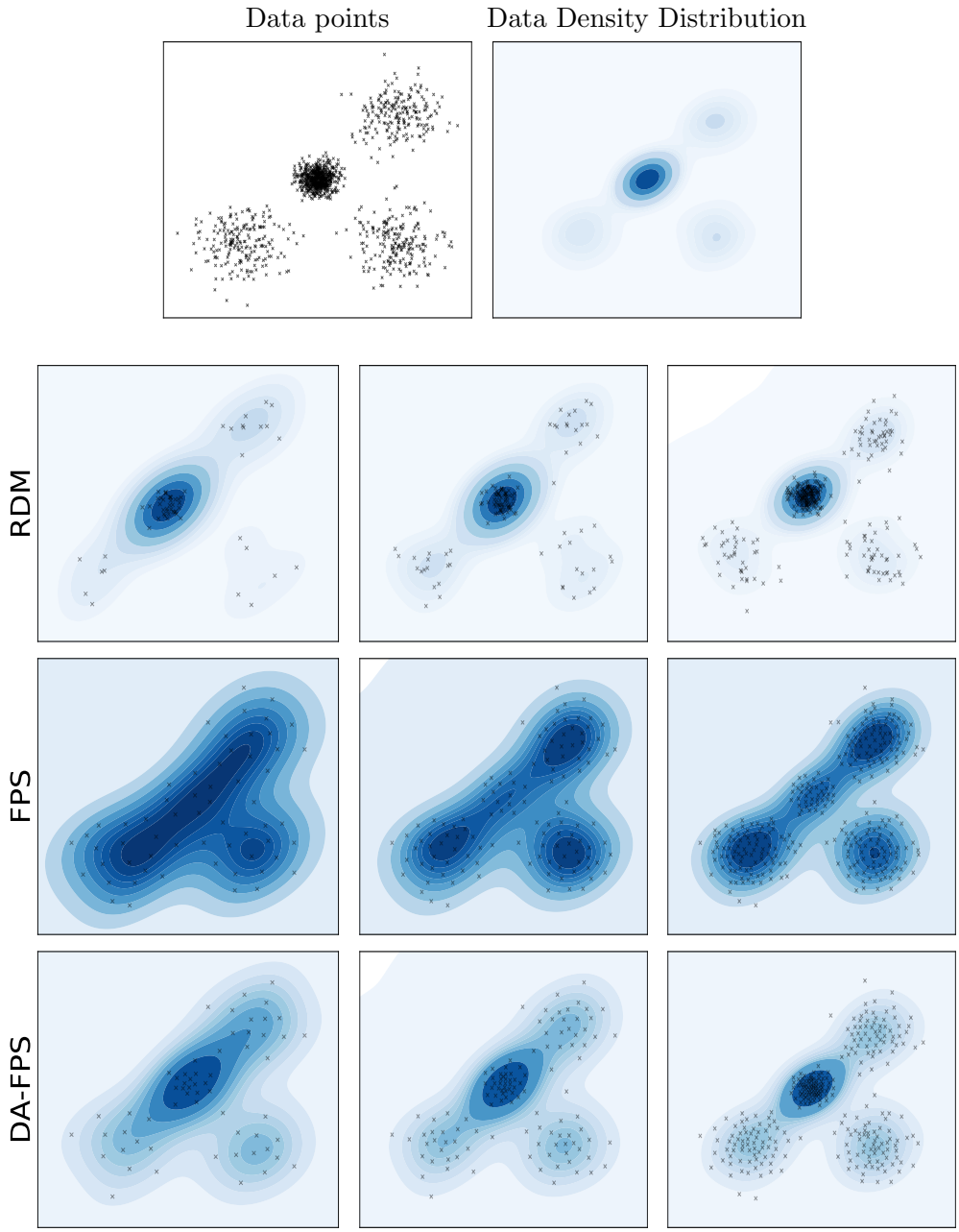


Figure 6.1.: Comparison of DA-FPS, FPS, and RDM. (Top row, left graph) The original dataset of 1200 points, with a high-density cluster (600 points) and three sparser clusters (200 points each). (Top row-right graph) A 2D kernel density estimation (KDE) plot highlighting the density variations. Darker areas indicate higher data density. (Graphs in rows 2-4) Subsets comprising 5%, 10%, and 20% of the available data, selected using RDM, FPS, and DA-FPS.

Step 1 The first step of the proof consists of showing that for each $1 \leq i < b+1$ we have

$$W_{\mathcal{L}_{i+1}} \leq W_{\mathcal{L}_i}.$$

To prove this first step we notice that $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \boldsymbol{x}_{i+1}$. Thus, for each $\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}$ we have that

$$\min_{\boldsymbol{x}_j \in \mathcal{L}_{i+1}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \leq \min_{\boldsymbol{x}_j \in \mathcal{L}_i} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2.$$

This is because adding a point to the selected set $\mathcal{L}_i$ ensures that the distance from any $\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}$ to its closest selected element either remains the same or decreases. Consequently, $\omega^k_{\mathcal{L}_{i+1}}(\boldsymbol{x}) \leq \omega^k_{\mathcal{L}_i}(\boldsymbol{x})$ also holds. To see this, recall that $\omega^k_{\mathcal{L}_i}(\boldsymbol{x})$ is the number of data points in $\mathcal{D}_{\mathcal{X}}$ contained within the ball centered at $\boldsymbol{x}$ with radius $r^k_{\mathcal{L}_i}(\boldsymbol{x}) := \min\left\{\min_{\boldsymbol{x}_j \in \mathcal{L}_i} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 + \frac{\epsilon_{\mathcal{X}}}{|\mathcal{L}_i|}, \rho_k(\boldsymbol{x})\right\}$, where $\rho_k(\boldsymbol{x})$ is the distance between $\boldsymbol{x}$ and its k-th nearest neighbor and $\epsilon_{\mathcal{X}}$ positive scalar value, which we consider arbitrary small. Adding a point to the selected set $\mathcal{L}_i$ ensures that the distance between any $\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}$ and its closest selected element does not increase. As a result, the value of $r^k_{\mathcal{L}_i}(\boldsymbol{x})$ is non-increasing, which in turn implies that the weights $\omega^k_{\mathcal{L}_i}(\boldsymbol{x})$ are also non-increasing. Therefore, we have that for each $1 \leq i < b+1$

$$\min_{\boldsymbol{x}_j \in \mathcal{L}_{i+1}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \omega^k_{\mathcal{L}_{i+1}}(\boldsymbol{x}) \leq \min_{\boldsymbol{x}_j \in \mathcal{L}_i} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \omega^k_{\mathcal{L}_i}(\boldsymbol{x}),$$

Since the above inequality holds for each $\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}$, by taking the maximum over the points in $\mathcal{D}_{\mathcal{X}}$ we prove the first claim.

Step 2 The second step of the proof shows that for each $2 \leq i \leq b+1$ and $1 \leq l < m \leq i$ we have that

$$W_{\mathcal{L}_{i-1}} \leq \|\boldsymbol{x}_m - \boldsymbol{x}_l\|_2 \omega^k_{\mathcal{L}_{m-1}}(\boldsymbol{x}_m), \tag{6.16}$$

where $\boldsymbol{x}_m$ and $\boldsymbol{x}_l$ are the points selected by Algorithm 6 at the m-th and l-th iterations, respectively. To prove this second step we proceed by induction. For the base step we have $i = 2,\ m = 2,\ l = 1$. Then we have

$$W_{\mathcal{L}_1} = \max_{\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}} \|\boldsymbol{x} - \boldsymbol{x}_1\|_2 \omega^k_{\mathcal{L}_1}(\boldsymbol{x}) = \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2 \omega^k_{\mathcal{L}_1}(\boldsymbol{x}_2)$$

which verifies the base step. The second equality follows from how the selection strategy in Algorithm 6 is defined. Next, let us assume the assumption in (6.16) is true for $i-1$. Then we have for each $1 \leq l < m \leq i-1$

$$W_{\mathcal{L}_{i-1}} \leq W_{\mathcal{L}_{i-2}} \leq \|\boldsymbol{x}_m - \boldsymbol{x}_l\|_2 \omega^k_{\mathcal{L}_{m-1}}(\boldsymbol{x}_m).$$

Where the first inequality follows from the first step of the proof and the second inequality is our inductive assumption. Now notice that for each $1 \leq r < i$ we have

$$W_{\mathcal{L}_{i-1}} = \min_{\boldsymbol{x}_j \in \mathcal{L}_{i-1}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i) \leq \|\boldsymbol{x}_i - \boldsymbol{x}_r\|_2 \omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i),$$

which proves the inductive step.

Step 3 Consider now a set $\mathcal{C} \subset \mathcal{D}_{\mathcal{X}}$, with $|\mathcal{C}| = b$. Observe that by the definition of weighted fill distance we have that for each $\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}$ there exists $\boldsymbol{c} \in \mathcal{C}$ such that $\omega_{\mathcal{C}}^{k}(\boldsymbol{x})\|\boldsymbol{x} - \boldsymbol{c}\|_2 \leq W_{\mathcal{C}}$. Next, notice that given $\mathcal{L}_{b+1}$, with $|\mathcal{L}_{b+1}| = b + 1$, selected with Algorithm 6, by the pigeonhole principle we have that there exists $\boldsymbol{x}_m, \boldsymbol{x}_l \in \mathcal{L}_{b+1}$ with $1 \leq l < m \leq b + 1$ that have a common closest element $\bar{\boldsymbol{c}} \in \mathcal{C}$. Therefore, $\max\{\|\boldsymbol{x}_m - \bar{\boldsymbol{c}}\|_2 \omega_{\mathcal{C}}^{k}(\boldsymbol{x}_m),\ \|\boldsymbol{x}_l - \bar{\boldsymbol{c}}\|_2 \omega_{\mathcal{C}}^{k}(\boldsymbol{x}_l)\} \leq W_{\mathcal{C}}$. Thus, we have

$$
\begin{aligned}
W_{\mathcal{L}_b} &\leq \|\boldsymbol{x}_m - \boldsymbol{x}_l\|_2 \omega_{\mathcal{L}_{m-1}}^{k}(\boldsymbol{x}_m) \\
&\leq \left(\|\boldsymbol{x}_m - \bar{\boldsymbol{c}}\|_2 + \|\boldsymbol{x}_l - \bar{\boldsymbol{c}}\|_2\right) \omega_{\mathcal{L}_{m-1}}^{k}(\boldsymbol{x}_m) \\
&\leq \left(\|\boldsymbol{x}_m - \bar{\boldsymbol{c}}\|_2 \omega_{\mathcal{C}}^{k}(\boldsymbol{x}_m) + \|\boldsymbol{x}_l - \bar{\boldsymbol{c}}\|_2 \omega_{\mathcal{C}}^{k}(\boldsymbol{x}_l)\right) \omega_{\mathcal{L}_{m-1}}^{k}(\boldsymbol{x}_m) \\
&\leq 2\omega_{\mathcal{L}_{m-1}}^{k}(\boldsymbol{x}_m) W_{\mathcal{C}} \\
&\leq 2k W_{\mathcal{C}}.
\end{aligned}
\tag{6.17}
$$

The first inequality follows from the second step of the proof, the second inequality follows from the triangular inequality of the distance considered, the third and fifth inequalities follow from the fact that, by its definition, we have $1 \leq \omega_{\mathcal{C}}^{k}(\boldsymbol{x}) \leq k$ for all $\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}$ and $\mathcal{C} \subset \mathcal{D}_{\mathcal{X}}$. Since the above inequality holds for each $\mathcal{C} \subset \mathcal{D}_{\mathcal{X}}$, it holds for the optimal subset $O_{\mathcal{X}}$ as well. □

Note that, Algorithm 6 can be implemented using $\mathcal{O}(|\mathcal{D}|k)$ memory and takes $\mathcal{O}(b|\mathcal{D}|k)$ time. To give a qualitative understanding of DA-FPS efficiency, with our PyTorch [PGM+19] implementation, it takes approximately 330 seconds to select 1000 points from the QM9 [RvDBR12, RDRvL14] dataset consisting 133885 data points of dimension 100.[1] Later in Section 7.2.5 we provide more experimental results showing the efficiency of DA-FPS and how it scales with the dataset size. Additionally, it is important to note that if the nearest neighborhood size to compute the weights as in (6.11) is set to $k = 1$, the optimization problem in (6.13) coincides with the fill distance minimization problem and Algorithm 6 reduces to the well known Farthest Point Sampling algorithm (FPS), thus providing 2-optimal solution [HP11]., which is the best approximation factor attainable in polynomial time with theoretical guarantees [HS85].

We note that, while Theorem 6.2 demonstrates that DA-FPS provides suboptimal solutions to the optimization problem in (6.13), multivariate density estimation theory indicates that smaller selected sets lead to more biased approximations of the corresponding density marginals in (6.8), especially for high-dimensional data [WKV09]. In simpler terms, the smaller the size of the selected set, the less representative are the computed weights of the actual distribution of the points. To mitigate these limitations, in our experiments, we make use of the hyperparameter $u > 0$. During the initial steps of the sampling process we consider constant weights, which we then iteratively update according to (6.11) after selecting the first u points. Thus, in early stages, DA-FPS coincides with FPS. That is, it focuses on minimizing the maximum distance between any point in the dataset $\mathcal{D}_{\mathcal{X}}$ and its closest selected element. This approach ensures

[1] We used a 48-cores CPU with 384 GB RAM.

broader coverage of the data space without initially accounting for the density of the data. The value of u plays a crucial role in the effectiveness of the selection strategy for training data development in regression task. If u is set too small, sparse regions of the dataset may be underrepresented in the training set, as the sampling will primarily target higher-density regions. This imbalance can result in low model performances in sparse regions of the dataset with a consequential increase in the average error.

Theorem 6.2 states that DA-FPS provides a $2k$-optimal result to the optimization problem in (6.13), which considers dynamic-weights, that is, the weights are iteratively updated any time a new point is selected. We can consider a simplified version of the optimization problem in (6.13) by considering for each data point $\boldsymbol{x} \in \mathcal{D}_{\mathcal{X}}$ a fixed weight $\omega(\boldsymbol{x}) \in \mathbb{R}^+$, which is determined a-priori and does not depend on the selected set. Such a simplified scenario has been already studied in literature. In particular, the authors of [DF85] show that it is possible to find solutions that are σ-optimal, with $\sigma := \min\{3, 1+\alpha\}$, where $\alpha := \frac{\max_{\boldsymbol{x}\in\mathcal{D}_{\mathcal{X}}} \omega(\boldsymbol{x})}{\min_{\boldsymbol{x}\in\mathcal{D}_{\mathcal{X}}} \omega(\boldsymbol{x})}$. That is, it is possible to find solutions that are at least 3-optimal. With the following theorem we attempt to extend the result provided in [DF85] into our scenario with dynamic weights.

Theorem 6.3 *Given set $\mathcal{D}_{\mathcal{X}} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, subset $O_{\mathcal{X}} \subset \mathcal{D}_{\mathcal{X}}$, optimal solution to the problem in (6.13) with $|O_{\mathcal{X}}| = b \in \mathbb{N}^+$, $b < n$, and $\mathcal{L}_{\mathcal{X}} \subset \mathcal{D}_{\mathcal{X}}$, $|\mathcal{L}_{\mathcal{X}}| = b$, subset selected with Algorithm 6 initialized with $\mathcal{L}_{\mathcal{X}} = \emptyset$ and $u = 0$, we have*

$$W^k_{\mathcal{L}_{\mathcal{X}},\mathcal{D}_{\mathcal{X}}} \leq \sigma\gamma W^k_{O_{\mathcal{X}},\mathcal{D}_{\mathcal{X}}}, \tag{6.18}$$

where $W^k_{\mathcal{L}_{\mathcal{X}},\mathcal{D}_{\mathcal{X}}}$ and $W^k_{O_{\mathcal{X}},\mathcal{D}_{\mathcal{X}}}$ are the estimated weighted fill distances of $\mathcal{L}_{\mathcal{X}}$ and $O_{\mathcal{X}}$ in $\mathcal{D}_{\mathcal{X}}$, respectively, defined as in (6.14). Moreover,

$$\gamma := \max_{j=1,\dots,b+1} \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_j)} \tag{6.19}$$

and

$$\sigma := \min\{3, 1+\alpha\} \quad \textit{with} \quad \alpha := \max_{\substack{i,j=1,\dots,b+1 \\ i<j}} \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)}. \tag{6.20}$$

For each $j = 1, \dots, b$, $\mathcal{L}_j := \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_j\}$ is the set of cardinality j obtained with Algorithm 6. We set $\mathcal{L}_0 = \emptyset$. The weights $\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)$ and $\omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_j)$ in (6.19) and (6.20) are computed according to the same principle as in (6.11).

Proof. Let $O_{\mathcal{X}} := \{\boldsymbol{o}_1, \dots, \boldsymbol{o}_b\}$ be an optimal solution to the optimization problem in (6.13). Moreover, let $\mathcal{L}_{b+1} := \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{b+1}\}$ be the set of cardinality $b+1$ obtained with Algorithm 6. First, note that by pigeonhole principle there exist $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{L}_{b+1}$, with $1 \leq i < j \leq b+1$ such that there exists a common closest element $\boldsymbol{o}_c \in O_{\mathcal{X}}$. Therefore, $\max\{\|\boldsymbol{x}_i - \boldsymbol{o}_c\|_2 \omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_i), \|\boldsymbol{x}_j - \boldsymbol{o}_c\|_2 \omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_j)\} \leq W_{O_{\mathcal{X}},\mathcal{D}_{\mathcal{X}}}$. Next, we define the quantity

$$\beta := \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)}$$

and consider two scenarios $\beta \leq 2$ and $\beta > 2$.

First scenario: $\beta \leq 2$. If we assume $\beta \leq 2$ we can prove the Theorem as follows

$$\begin{aligned}
W^k_{\mathcal{L}_b,\mathcal{D}_\mathcal{X}} \leq & \omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j) \min_{\boldsymbol{x}\in\mathcal{L}_{j-1}} \|\boldsymbol{x}-\boldsymbol{x}_j\|_2 \\
\leq & \omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2 \\
\leq & \omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)\left(\|\boldsymbol{x}_i-\boldsymbol{o}_c\|_2+\|\boldsymbol{x}_j-\boldsymbol{o}_c\|_2\right) \\
\leq & \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{O_\mathcal{X}}(\boldsymbol{x}_j)}\omega^k_{O_\mathcal{X}}(\boldsymbol{x}_j)\|\boldsymbol{x}_j-\boldsymbol{o}_c\|_2 \\
& + \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)}\frac{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)}{\omega^k_{O_\mathcal{X}}(\boldsymbol{x}_i)}\omega^k_{O_\mathcal{X}}(\boldsymbol{x}_i)\|\boldsymbol{x}_i-\boldsymbol{o}_c\|_2 \\
\leq & \gamma(1+\beta)W^k_{O_\mathcal{X},\mathcal{D}_\mathcal{X}} \\
\leq & \sigma\gamma W^k_{O_\mathcal{X},\mathcal{D}_\mathcal{X}}.
\end{aligned}$$

The first inequality follows from the fact that $W^k_{\mathcal{L}_{i+1},\mathcal{D}_\mathcal{X}} \leq W^k_{\mathcal{L}_i,\mathcal{D}_\mathcal{X}}$ for all $i = 1,\ldots,b$. This is shown in Step 1 of the proof of Theorem 6.2. The second inequality follows from the fact that $\boldsymbol{x}_i \in \mathcal{L}_{j-1}$ since $i < j$. The last inequality follows from the assumption that $\beta \leq 2$, thus, we have that $\beta \leq \min\{\alpha, 2\}$ which implies that $1+\beta \leq \sigma$.

Second scenario: $\beta > 2$. Consider $1 = i < j \leq b+1$. Note that by how the weights are defined in (6.11), we have that $\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i) = \omega^k_{\mathcal{L}_0}(\boldsymbol{x}_1) = \omega^k_{\emptyset}(\boldsymbol{x}_1) = k$ and that for each $j = 2,\ldots,b+1$ we have $1 \leq \omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j) \leq k$. Thus, if $1 = i < j \leq b+1$, it follows that

$$\beta = \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)} = \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{k} \leq 1,$$

which contradicts the assumption $\beta > 2$, so it holds $i > 1$. Next, consider $1 \leq l < i < j \leq b+1$, and $\boldsymbol{x}_l$ to be the closest point to $\boldsymbol{x}_j$ when $\boldsymbol{x}_i$ is chosen, then we have that

$$\begin{aligned}
W^k_{\mathcal{L}_b,\mathcal{D}_\mathcal{X}} \leq & \omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j) \min_{\boldsymbol{x}\in\mathcal{L}_{j-1}} \|\boldsymbol{x}-\boldsymbol{x}_j\|_2 \\
\leq & \omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)\|\boldsymbol{x}_l-\boldsymbol{x}_j\|_2 \\
\leq & \left|\left\{\bar{\boldsymbol{x}} \in \mathcal{D}_\mathcal{X} \text{ such that } \|\boldsymbol{x}_j-\bar{\boldsymbol{x}}\|_2 \leq \min\{\|\boldsymbol{x}_j-\boldsymbol{x}_l\|_2 + \frac{\epsilon_\mathcal{X}}{|\mathcal{L}_{i-1}|}, \rho_k(\boldsymbol{x}_j)\}\right\}\right|\|\boldsymbol{x}_l-\boldsymbol{x}_j\|_2 \\
= & \omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_j) \min_{\boldsymbol{x}\in\mathcal{L}_{i-1}} \|\boldsymbol{x}-\boldsymbol{x}_j\|_2 \\
\leq & \omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i) \min_{\boldsymbol{x}\in\mathcal{L}_{i-1}} \|\boldsymbol{x}-\boldsymbol{x}_i\|_2 \\
\leq & \omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)\|\boldsymbol{x}_i-\boldsymbol{x}_l\|_2 \\
\leq & \omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)\left(\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2+\|\boldsymbol{x}_j-\boldsymbol{x}_l\|_2\right).
\end{aligned} \tag{6.21}$$

The second inequality follows from the fact that $\boldsymbol{x}_l \in \mathcal{L}_{j-1}$, thus the distance between $\boldsymbol{x}_j$ and the closest element in $\mathcal{L}_{j-1}$ is smaller than $\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2$. This is also relevant for the third inequality: The value of the weight $\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)$, which is the amount of data points in the ball centered in $\boldsymbol{x}_j$ with radius $r^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j) := \min\left\{ \min_{\boldsymbol{x}\in\mathcal{L}_{j-1}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 + \frac{\epsilon_{\mathcal{X}}}{|\mathcal{L}_{j-1}|}, \rho_k(\boldsymbol{x}_j)\right\}$, is less or equal the amount of data points contained in the ball centered in $\boldsymbol{x}_j$ with the larger radius of $\min\{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2 + \frac{\epsilon_{\mathcal{X}}}{|\mathcal{L}_{i-1}|}, \rho_k(\boldsymbol{x}_j)\}$. The equality follows from the fact that we assume $\boldsymbol{x}_l$ to be the closest point to $\boldsymbol{x}_j$ when $\boldsymbol{x}_i$ is chosen, that is, $r^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_j) = \min\{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2 + \frac{\epsilon_{\mathcal{X}}}{|\mathcal{L}_{i-1}|}, \rho_k(\boldsymbol{x}_j)\}$. The fourth inequality is true because if we assume it was false then $\boldsymbol{x}_j$ would have been selected before $\boldsymbol{x}_i$.

If we now assume that $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \leq \|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2$ by the inequalities in (6.21) we would have

$$\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2 \leq 2\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2,$$

which implies that

$$\frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)} \leq 2,$$

which is a contradiction since we are assuming $\beta > 2$. Thus, we have that $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 > \|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2$. Therefore, from equation (6.21), we have that

$$\begin{aligned}
W^k_{\mathcal{L}_b,\mathcal{D}_{\mathcal{X}}} \leq & 2\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \\
\leq & 2\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)\left(\|\boldsymbol{x}_i - \boldsymbol{o}_c\|_2 + \|\boldsymbol{x}_j - \boldsymbol{o}_c\|_2\right) \\
\leq & 2\frac{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)}{\omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_i)}\omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_i)\|\boldsymbol{x}_i - \boldsymbol{o}_c\|_2 \\
& + 2\frac{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)}{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}\frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_j)}\omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_j)\|\boldsymbol{x}_j - \boldsymbol{o}_c\|_2 \\
\leq & 2\gamma(1+\beta^{-1})W^k_{O_{\mathcal{X}},\mathcal{D}_{\mathcal{X}}} \\
\leq & \sigma\gamma W^k_{O_{\mathcal{X}},\mathcal{D}_{\mathcal{X}}}
\end{aligned}$$

The last inequality follows from the facts that $\beta > 2 \Rightarrow \alpha > 2 \Rightarrow 1 + \alpha > 3 \Rightarrow \sigma = 3$. Thus, since $2(1 + \beta^{-1}) \leq 3$, we have that $\sigma \geq 2(1 + \beta^{-1})$. □

With Theorem 6.3 we provide an alternative result to Theorem 6.2 for the optimality of the solution provided by DA-FPS. Note that in Theorem 6.3 we explicitly link the quality of approximation of DA-FPS with the ratio between the computed and optimal weights. In particular, one of the terms in the approximation factor is $\gamma := \max_{j=1,\ldots,b+1} \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{O_{\mathcal{X}}}(\boldsymbol{x}_j)}$, that is, the ratio between the weights related to the set selected with DA-FPS and those of an optimal set. Note that the simplest upper bound for γ is $\gamma \leq k$. This is because, for how we defined them in (6.11), the weights value is at least 1 and at the most k, independently of the set considered to define them. Thus, using the simplest upper bound for γ, according to Theorem 6.3, DA-FPS achieves approximations that

are $3k$-optimal. This rate represents a less favorable worst-case scenario compared to the $2k$-optimal rate given by Theorem 6.2. Nonetheless, we think this result is relevant because it showcases that there may be other ways to determine the approximation factor of DA-FPS than the one provided in Theorem 6.2. Moreover, by explicitly linking the quality of DA-FPS's approximation to γ and σ, we aim to highlight a possible path for improving the optimality constant by identifying a bound for either of these two quantities. Future work should focus on this direction.

7. Numerical Results

In this chapter we provide empirical validations for the theoretical results from Chapter 5 and Chapter 6. It is organized in two main sections. In the first, we consider the low data budget regime scenario and report experiments showing that selecting the training set by fill distance minimization using FPS substantially reduces the maximum prediction error for Lipschitz continuous regression models. In the second section we focus on the medium data budget regime and show that by minimizing the introduced weighted fill distance of the training set with the DA-PFS, we can improve the average prediction performances of Lipschitz continuous regression models. The datasets, regression models, and data processing procedures we employ in our experiments related to chemistry applications have been introduced and described in detail in Chapter 4. Our GitHub repository (https://github.com/ins-uni-bonn/PassiveSamplingML) contains all the necessary code for downloading, reading, and preprocessing the datasets, our implementations of FPS and DA-FPS, regression models, and evaluation procedures.

7.1. Minimizing the fill distance with FPS

In this section we investigate the effects of minimizing the training set fill distance on regression tasks from quantum chemistry. We predict molecular properties on the QM7, QM8 and QM9 datasets. In particular, we study the performance of FPS in comparison to various passive and model-agnostic sampling baselines while using two machine learning models for prediction, KRR and FNN. Additionally, we empirically investigate the potential benefit of minimizing the training set fill distance for multivariate regression tasks, that is, regression tasks where the label value to predict is multidimensional. More precisely, we focus on the force-field prediction task on molecular trajectories from the rMD17 dataset. The molecular force-field consists of the per-atom forces in a molecule.

7.1.1. Baseline sampling strategies for FPS

We compare the effects of minimizing the training set fill distance through the FPS algorithm with three sampling techniques. We selected the three baselines method among those analyzed in well-established papers related to data selection [SS18, GZ19, MBL20, KSRI21], that fit our application scenario and problem constraints. That is, passive and model-agnostic data selection techniques that do not rely on the knowledge of the labels. Specifically, we consider uniform random sampling (RDM), the facility location algorithm and k-medoids++, which we introduced and described in Chapter 3 and briefly recapture next. RDM is considered the natural benchmark for all the other coreset sampling strategies [Fel19], and consists of choosing the points to label and use for training

uniformly at random from the available pool of data points. Facility location [Fri74] is a greedy algorithm that aims at minimizing the sum of the squared distances between the points in the pool and their closest selected element. k-medoids++ [MJH$^+$11] is a variant of the k-means++ [AV07], that partitions the data points into k clusters and, for each cluster, selects one data point as the cluster center by minimizing the distance between points labeled to be in a cluster and the point designated as the center of that cluster. Both, facility location and k-medoids++, attempt to minimize a sum of pairwise distances. However, the fundamental difference is that facility location is a greedy technique, while k-medoids++ is based on a segmentation of the data points into clusters.

7.1.2. Experimental setting with FPS

This section reports experiments related to molecular property prediction tasks on four different datasets: QM7, QM8 and QM9 for univariate regression and rMD17 for multivariate regression. Detailed information on the datasets, preprocessing procedures, descriptors and regression models are provided in Section 4.2. Here we introduce them briefly. QM7 [BR09, RTMvL12] consists of 7165 molecules with up to 23 atoms that we represent as vectors in $\mathbb{R}^{529}$. The vector-valued descriptors provide information related to the molecules' geometrical structure. The label value to predict is the atomization energy, measured in electronvolt (eV). QM8 [RvDBR12, RHTvL15] consists of 21786 molecules with up to 8 heavy atoms that we represent as vectors in $\mathbb{R}^{1296}$ describing the molecules' topological structure. The label values to predict is the lowest singlet transition energy (E1), measured in eV. QM9 [RvDBR12, RDRvL14] has 133885 molecules with up to 9 heavy atoms that we represent as vectors in $\mathbb{R}^{1307}$ describing the molecules' topological structure. The label values to predict is the HOMO-LUMO energy, measured in eV. The revised MD17 [CvL20a, CvL20b] (rMD17) consists of temporal trajectories of various small organic molecules. We study the trajectories of the Benzene with 9 atoms, Uracil and Malonaldehyde each consisting of 12 atoms. We represent the molecules using the vector-valued descriptors defined in (4.2) providing information on the molecules' geometrical structure. We use the rMD17 for a multivariate regression task, where the vector-valued labels to predict are the per atom forces measured in $\frac{\text{kcal}}{\text{mol}\times\text{ångstrong}}$.

We consider two regression models for the univariate tasks: KRR with the Gaussian kernel and FNN. The hyperparameters of the KRR are selected following the grid-search procedure described in Section 4.3, by varying the hyperparameters on a tensor product grid with 12 points per dimension between 10^{-14} and 10^{-2}. To train the FNN, we use the procedure described in Section 4.3. We consider 1000 epochs for the QM7 and 250 epochs for the QM8 and QM9. QM8 and QM9 are much larger than QM7, and the feature vectors used for the QM8 and QM9 have higher dimension than those used for QM7. Thus, we use fewer epochs on the QM8 and QM9 to make the experiments computationally affordable. For the multivariate regression tasks on the rMD17 we use the gradient-domain machine learning (GDML) method introduced in Section 4.3.

The experiments we perform involve testing the predictive accuracy of each trained model. For measuring the predictive accuracy of the regression models we use various

evaluation metrics. For the molecular property prediction tasks on the QM datasets, with univariate labels, we consider the Maximum Absolute Error (MAXAE), Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). For the force-field prediction tasks on the rMD17, with vector-valued labels, we consider MAXAE_F, MAXMAE_F and MAE_F. Recall that, in the univariate label scenario, given true target values $\{y_i\}_{i=1}^n$ and the predicted values $\{\tilde{y}_i\}_{i=1}^n$ we have

$$\text{MAXAE} := \max_{1\leq i\leq n} |y_i - \tilde{y}_i|, \tag{7.1}$$

$$\text{MAE} := \frac{1}{n}\sum_{i=1}^{n} |y_i - \tilde{y}_i|, \tag{7.2}$$

$$\text{RMSE} := \sqrt{\frac{1}{n}\sum_{i=1}^{n} |y_i - \tilde{y}_i|^2}. \tag{7.3}$$

In the multivariate scenario, consider the true target values $\{\boldsymbol{F}_i\}_{i=1}^n \subset \mathbb{R}^{3n_a}$, describing the per-atom force of the analyzed molecule, with n_a atoms and $\boldsymbol{F}_i = [F_{i,1}, F_{i,2}, \ldots, F_{i,3n_a}]$, along its trajectory. Moreover, consider the predicted values $\{\tilde{\boldsymbol{F}}_i\}_{i=1}^n \subset \mathbb{R}^{3n_a}$, then we define

$$\text{MAXAE}_F := \max_{1\leq i\leq n} \max_{1\leq j\leq 3n_a} |F_{i,j} - \tilde{F}_{i,j}|, \tag{7.4}$$

$$\text{MAXMAE}_F := \max_{1\leq i\leq n} \left(\frac{1}{3n_a} \sum_{j=1}^{3n_a} |F_{i,j} - \tilde{F}_{i,j}| \right), \tag{7.5}$$

$$\text{MAE}_F := \frac{1}{3nn_a} \sum_{i=1}^{n} \sum_{j=1}^{3n_a} |F_{i,j} - \tilde{F}_{i,j}|. \tag{7.6}$$

See Section 4.4 for a detailed description and interpretation of the above-mentioned evaluation metrics.

For each of the data sampling strategy we consider, we construct multiple training sets consisting of different amounts of samples. For each sampling strategy and training set size, the training set selection process is independently run five times. In the case of RDM, points are independently and uniformly selected at each run, while for the other sampling techniques, the initial point to initialize is randomly selected at each run. Consequently, for each selection strategy and training set size, each analyzed model is independently trained and tested five times. Each time, the performance of the models are tested on all the points in the dataset not used for training. The reported test results are the average of the five runs. We also plot error bars, which, unless otherwise specified, represent the standard deviation of the results. We remark that the final goal of our experiments is to empirically show the benefits of using FPS compared to other model-agnostic state-of-the-art sampling approaches. We do not make any claims on the general prediction quality of the employed models on any of the studied datasets.

7.1.3. Experiments with FPS: Molecular property prediction

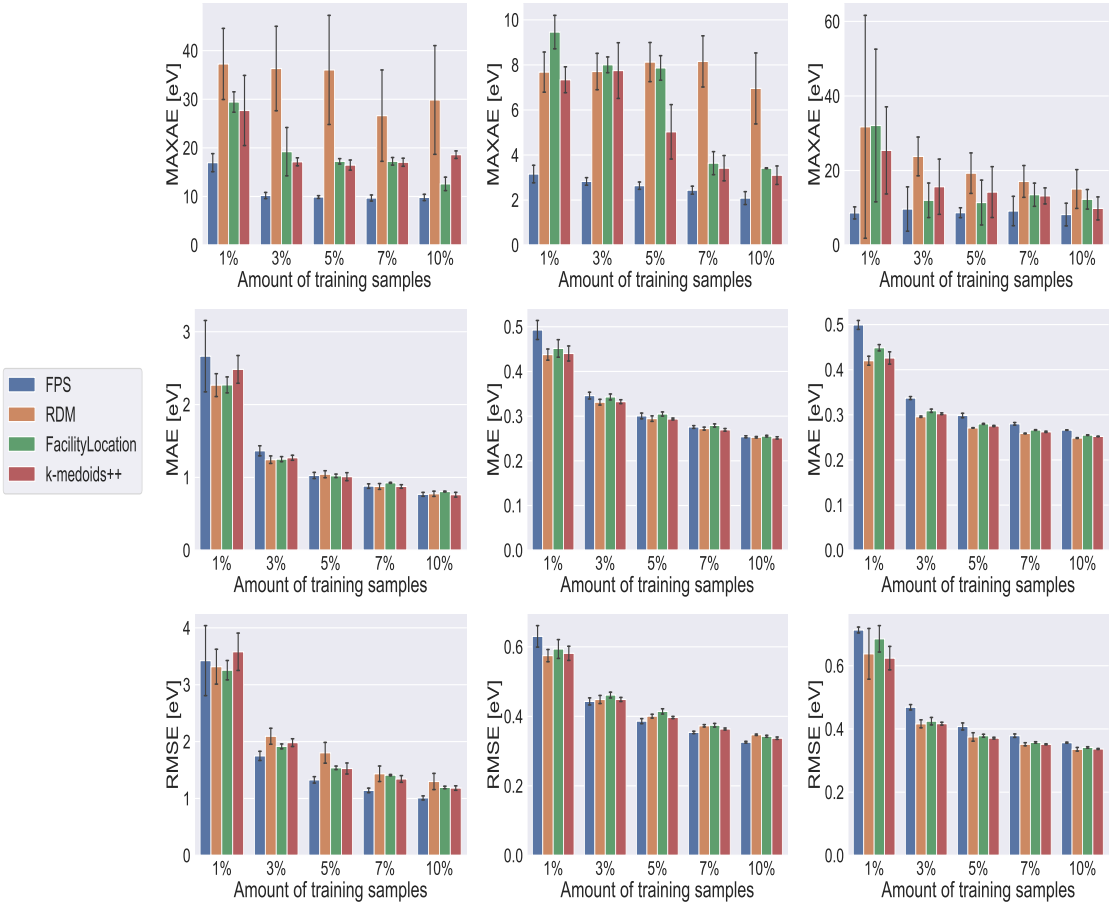
Fig. 7.1 and Fig. 7.2 show the results for the regression tasks with scalar label values on the QM7, QM8 and QM9 datasets using KRR with the Gaussian kernel and FNN, respectively. The graphs on the top rows of Fig. 7.1 and Fig. 7.2 illustrate the maximum absolute error (MAXAE) of the predictions on the unlabeled points. The results suggest that, independently of the dataset and the regression model, selecting the training set by fill distance minimization using FPS, we can perform better than the other baselines in terms of the maximum error of the predictions. FPS consistently leads to lower MAXAE values than the other baselines.

The graphs on the middle row of Fig. 7.1 and Fig. 7.2 show the MAE of the predictions on the QM7, QM8 and QM9 datasets for KRR and FNN, respectively. These graphs indicate that selecting training sets with FPS does not drastically reduce the MAE of the predictions on the unlabeled points with respect to the baselines, independently of the dataset and regression model. On the contrary, we observe examples where FPS performs worse than at least one of the baselines, e.g., with the FNN on the QM7, QM8 and QM9 when trained with 5% of the available data points. These experiments suggest that, contrary to what has been shown for classification [SS18], selecting training sets by fill distance minimization does not provide any significant advantage compared to the baselines in terms of the average error in the low data budget scenario. This marks a fundamental difference between regression and classification tasks regarding the benefits of reducing the training set fill distance.

The graphs on the bottom row of Fig. 7.1 and Fig. 7.2 show the RMSE of the predictions on the QM7, QM8 and QM9 datasets for KRR and FNN, respectively. It can be clearly seen that selecting training sets with FPS may not reduce the RMSE of the predictions on the unlabeled points with respect to the baselines. For instance, with KRR on the QM9 FPS consistently leads to the worst performance. Nonetheless, there are scenarios where FPS does indeed lead to an improvement with respect to the baselines. For instance, with KKR on the QM7 when at least 3% of the available data points have been selected and on the QM8 when at least 5% the data points have been selected. Another example is with FNN on the QM7, where FPS consistently outperforms all the baselines. On the one hand, we think that this is due to the fact that the RMSE penalizes large errors more than the MAE, thus giving more relevance to outlier error values, as mentioned in Section 4.4. On the other hand, from Chapter 5 we know that minimizing the training set fill distance with the FPS leads to a substantial decrease in maximum prediction error and hence reduce the amount and magnitude of outlier error values. For these reasons, it follows that FPS may lead to an improvement with respect to the RMSE.

7.1.4. Increased numerical stability of kernel ridge regression with FPS

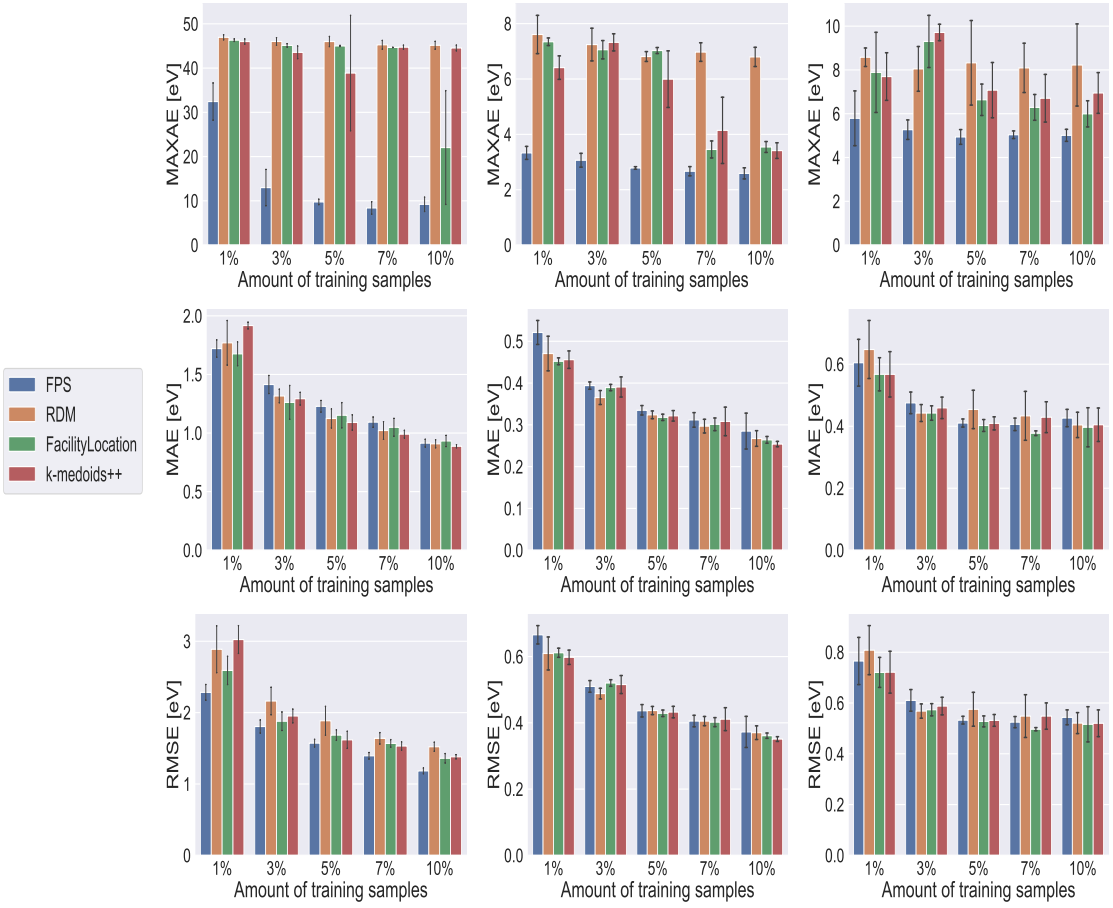
Next, we show that selecting training sets with FPS increases the stability of KRR models with Gaussian kernel by reducing the condition number of the kernel matrix, as we show theoretically in Section 5.4. The graphs in the bottom row of Fig. 7.3 show the condition number of the regularized kernel matrices generated during training of the



(a) KRR on QM7 (b) KRR on QM8 (c) KRR on QM9

Figure 7.1.: Results for regression tasks on QM7, QM8 and QM9 using KRR with the Gaussian kernel trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. MAXAE (top row), MAE (middle row) and RMSE (bottom row) are shown for each training set size and sampling approach. Error bars represent the standard deviation of the results over five runs. The results related to the MAXAE in the top row indicate that selecting the training set using FPS (blue bars) leads to improved performances, independently of the dataset. The graphs illustrating the MAE and RMSE show that selecting training sets with FPS may not lead to better predictions on average.

KRR approach and used to calculate the regression parameters as shown in (4.6). For the QM9, the condition number appears not to be affected by the training dataset choice, while for the QM7 and QM8, choosing training sets with FPS reduces the condition



(a) FNN on QM7 (b) FNN on QM8 (c) FNN on QM9

Figure 7.2.: Results for regression tasks on QM7, QM8 and QM9 using FNN trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. MAXAE (top row), MAE (middle row) and RMSE (bottom row) are shown for each training set size and sampling approach. Error bars represent the standard deviation of the results over five runs. The results related to the MAXAE in the top row indicate that selecting the training set using FPS (blue bars) leads to improved performances, independently of the dataset. The graphs illustrating the MAE and RMSE show that selecting training sets with FPS may not lead to better predictions on average.

number of the regularized kernel, particularly in the low data regime, leading to improved stability of the learned model. We remark that the graphs in the bottom row of Fig. 7.3 depict the condition number of the regularized kernel matrix $\boldsymbol{K}_{\mathcal{L}} + \lambda \boldsymbol{I}$, where $\boldsymbol{K}_{\mathcal{L}}$ is the

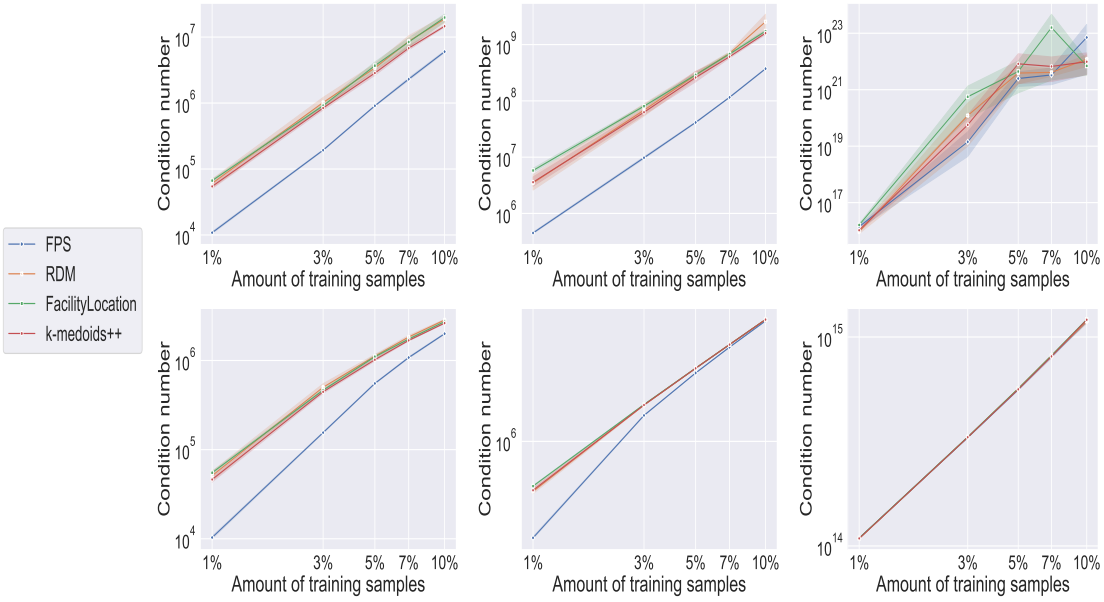
Gaussian kernel matrix built from the training data and $\lambda \boldsymbol{I}$ is the regularization term, introduced in (4.5), used to address ill-conditioning problems. The hyperparameter λ has been chosen following a procedure based on cross-validation on randomly selected subsets of the available data pool, as explained in Subsections 4.3.1 and 7.1.2. The values of λ we employed are $1.90 \cdot 10^{-4}$, $3.64 \cdot 10^{-4}$ and $1.19 \cdot 10^{-11}$ for the QM7, QM8 and QM9, respectively. The top row of Fig. 7.3 illustrates the condition numbers of the non-regularized kernels and shows that, if no regularization is applied, for the QM7 and the QM8 the difference between the condition numbers of the matrices obtained with the FPS and those obtained using the benchmark strategies is close to an order of magnitude, as expected from (5.15). As for the QM9, we still see a lower condition number when using the FPS in the low data limit, until 7% of the data is employed for training. Notice that, for the QM9, the magnitude of the condition number is significantly higher than for the other datasets due to the larger size of the kernel matrix.

It is also important to mention that, in our experiments, differences in the conditions numbers are mainly due to differences in the minimum eigenvalues, which aligns with the theory reported in Section 5.4. The top and bottom rows of Fig. 7.4 show the minimum and maximum eigenvalues of the non-regularized kernel matrices obtained from the training data from experiments in Fig. 7.3, respectively. It can be clearly seen that the kernels have similar maximum eigenvalues independently of how the training data was selected. Moreover, the maximum eigenvalues grow linearly with the size of the kernel matrix, while the minimum eigenvalues decrease exponentially. The y-axis of the minimum eigenvalue plots is on a logarithmic scale. For the QM7 and QM8, the differences between the minimum eigenvalues of the kernel matrices from training sets selected with FPS and those from the baselines are measured in orders of magnitude and reflect the differences between the relative condition numbers in Fig 7.3.

7.1.5. Empirical analysis and discussion

This section further examines the empirical results presented in 7.1.3. Specifically, we identify the overall trends of the MAXAE and relate them to our theoretical study, focusing on their connection with the concept of fill distance of the training set. Moreover, we emphasize what we think should be the practical application of our theoretical result in Theorem 5.1. Next, we analyze the benefit of employing the FPS from a more empirical perspective, focusing on understanding how the FPS selection process works, that is, what points are prioritized during the selection process, how they are distributed, and what consequence this has on the learning process of a given regression model. After that, we discuss the limitations of the FPS.

Interestingly, with FPS, the MAXAE converges fast to a plateau value for all datasets and regression models (Fig. 7.1- 7.2). Differently, with the baseline approaches, the MAXAE has much larger values in the low data regime and tends to decrease gradually as the size of the training sets increases. Note that, these trends of the MAXAE are directly correlated with the fill distances of the respective labeled sets used for training, illustrated in Fig. 7.5a. The figure clearly shows that independently of the dataset considered, with FPS, the fill distances are consistently lower even for small data budgets, while with

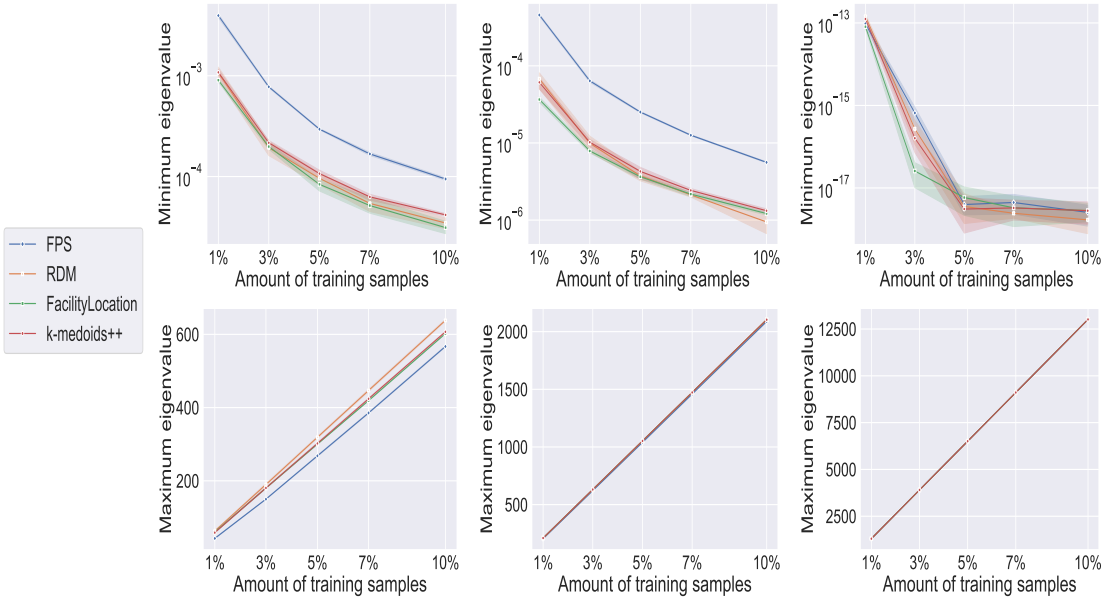


(a) QM7 (b) QM8 (c) QM9

Figure 7.3.: Condition number of the non-regularized (top row) and regularized (bottom row) Gaussian kernels are shown for each QM dataset, training set size and sampling approach. The graphs are on log-log scale and the error bands represent the confidence interval over five runs of the experiments. For the QM7 and QM8, choosing the training sets with FPS (blue lines) reduces the condition number of both regularized and non-regularized kernel, particularly in the low data regime, leading to improved stability of the learned KRR models. For the QM9, the condition number of the regularized kernels appears not to be affected by how the training set is chosen, while for the non-regularized kernels FPS appears to be beneficial mainly for lower training set sizes.

the benchmarks, the fill distances are much larger in the low data regime and gradually decrease as the size of the training set increases. These observations indicate that the training set fill distance is directly correlated with the predictions MAXAE, in line with our theoretical result. Nevertheless, from Theorem 5.1 we know that the training set fill distance is only linked to the maximum expected value of the error function. Moreover, the bound proposed in the theorem also depends on other quantities we may not know or that we cannot compute a priori: The labels' uncertainty and the maximum prediction error on the training set, quantifying how well the trained regression model fits the training data. Thus, we think that the training set fill distance should not be considered as the only parameter to obtain an a priori quantitative evaluation of the MAXAE of the predictions, but as a qualitative indicator of the model robustness that, if minimized, leads to a substantial reduction of the MAXAE.

The bound derived in Theorem 5.1 also depends on the labels' uncertainty and the

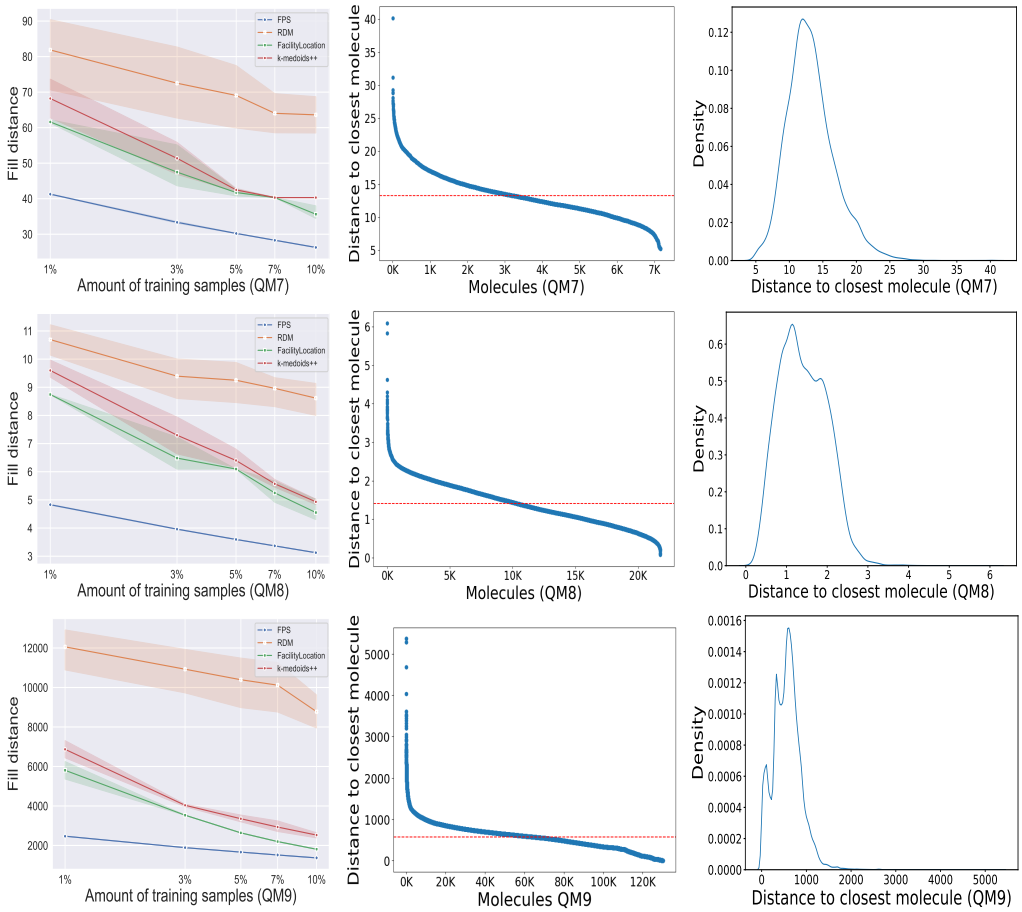


(a) QM7 (b) QM8 (c) QM9

Figure 7.4.: Minimum eigenvalues (top row) and maximum eigenvalues (bottom row) of the non-regularized Gaussian kernels are shown for each dataset, training set size and sampling approach. The y-axis of the graphs of the minimum eigenvalues are on the log-scale and the error bands represent the standard error over five independent runs of the experiments.

maximum error on the training set. We provide additional insight on these two quantities. Unfortunately, the labels' uncertainty is an intrinsic property of the dataset that we cannot compute or estimate unless we know the true solution of the regression problem, or we have an estimate of the error performed by the numerical procedure used to obtain the data labels. Concerning the maximum error on the training sets, we can compute it post-training and evaluate its behaviors with respect to the different selection strategies and regression models we consider. In what follows we analyze the maximum error on the training sets selected for the experiments in 7.1.3 related to the best and worst performing selection strategies, that is, we consider FPS and random sampling, respectively.

Fig. 7.6 shows the maximum absolute prediction error on the training sets for QM7, QM8 and QM9 using KRR (top row) and FNN (bottom row). The training sets we consider are the same we used for the experiments in Fig. 7.1 and Fig. 7.2. It can be clearly seen that the maximum error tends to increase as the size of the training set increases, which is an expected result since the more data samples we have, the more difficult it becomes for the model to accurately fit all the data points. Moreover, it is worth noticing that the behavior of the maximum error on the training set for the KRR is consistent across different datasets and training set sizes, independently of the selection strategy we consider. In particular, Fig. 7.6 suggests that, for the KRR, maximum errors on the training sets selected with the FPS and randomly are comparable. This



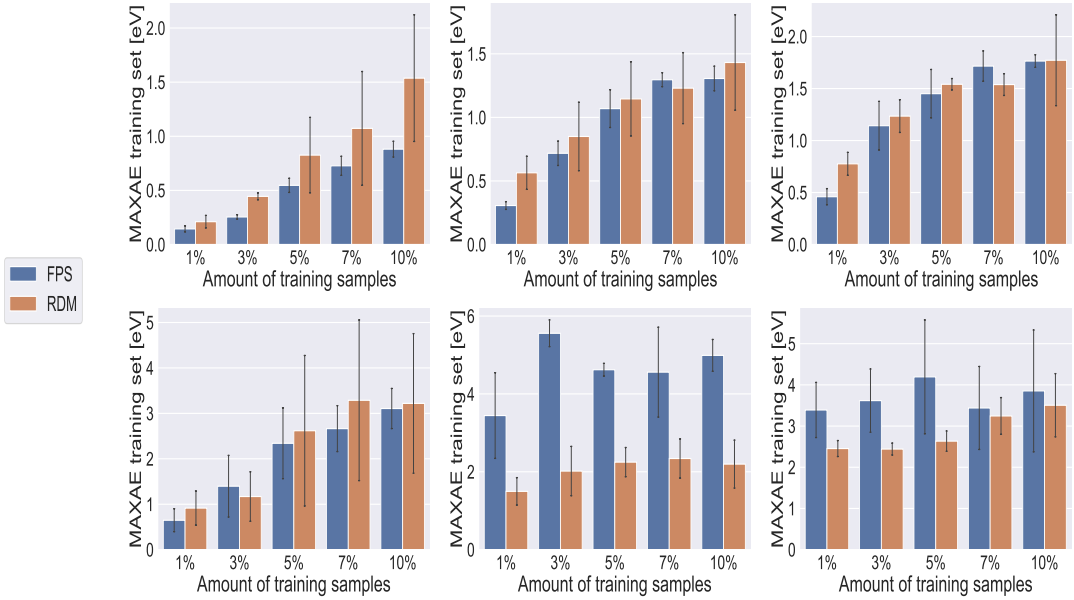
(a) Training sets fill distances.

(b) Distances to the nearest neighbor.

(c) Density of distances to the nearest neighbor.

Figure 7.5.: (a) Fill distances of the selected training sets. (b) Euclidean distances to the nearest neighbor and (c) density of such distances for molecules in the QM7 (top row), QM8 (middle row) and QM9 (bottom row). In (b) the red lines are the average distances between the molecules in the datasets and their nearest neighbor and the molecules are sequentially numbered such that the distances decrease in magnitude as the associated molecule numbers increase.

observation also holds for the FNN on QM7. This fact implies that differences in the maximum error on the relative test sets are mainly due to the other quantities appearing in the bound in Theorem 5.1, such as the training set fill distance. However, when comparing the FPS and random sampling (RDM) on QM8 and QM9 with FNN, we notice more pronounced differences. Notably, on the training sets selected with RDM the maximum error tends to be smaller than on those selected with FPS. It is also interesting to compare Fig 7.6, which illustrates the error on the training sets, with



(a) QM7 (b) QM8 (c) QM9

Figure 7.6.: MAXAE on training set for QM7, QM8 and QM9 using KRR (top row) and FNN (bottom row) trained on sets of various sizes, expressed as a percentage of the available data points, and selected uniformly at random (RDM) or with FPS.

Fig. 7.2 and Fig. 7.1, illustrating the error on the relative test sets. From this comparison, we notice that for KRR, the maximum error on the training set is negligible or much smaller compared to the maximum error on the test set, particularly for QM8 and QM9. This is not the case for the FNN regression model. Specifically, with FNN on QM8, the maximum error on the training sets selected with FPS can be larger than on the test set.

As a matter of fact, we think that the effectiveness of FPS is also due to its ability to sample, even for small training sets sizes, those points that are at the tails of the data distribution and that are convenient to label, as the predictive accuracy of the learning methods on those points would be limited due to the lack of data information in the portions of the feature space where data points are more sparsely distributed. To see this empirically, let us first consider Fig. 7.5b and Fig. 7.5c, showing for each molecule the Euclidean distance to the respective closest molecule and the density of such distances, respectively, for the QM7, QM8 and QM9 datasets. Fig. 7.5b shows that, in all the analyzed datasets, there are "isolated" molecules for which the Euclidean distance to the nearest molecule is more than twice the average distance between the molecules in the dataset and their nearest neighbor, represented by the red line in the graphs. Fig. 7.5c, representing the density distribution of the distances of the molecules to their closest data point, indicates that the "isolated" molecules are only a very small portion of the dataset and, therefore, represent the tail of the data distribution. We now see that FPS, contrary to the other baselines, can effectively sample the isolated molecules even for

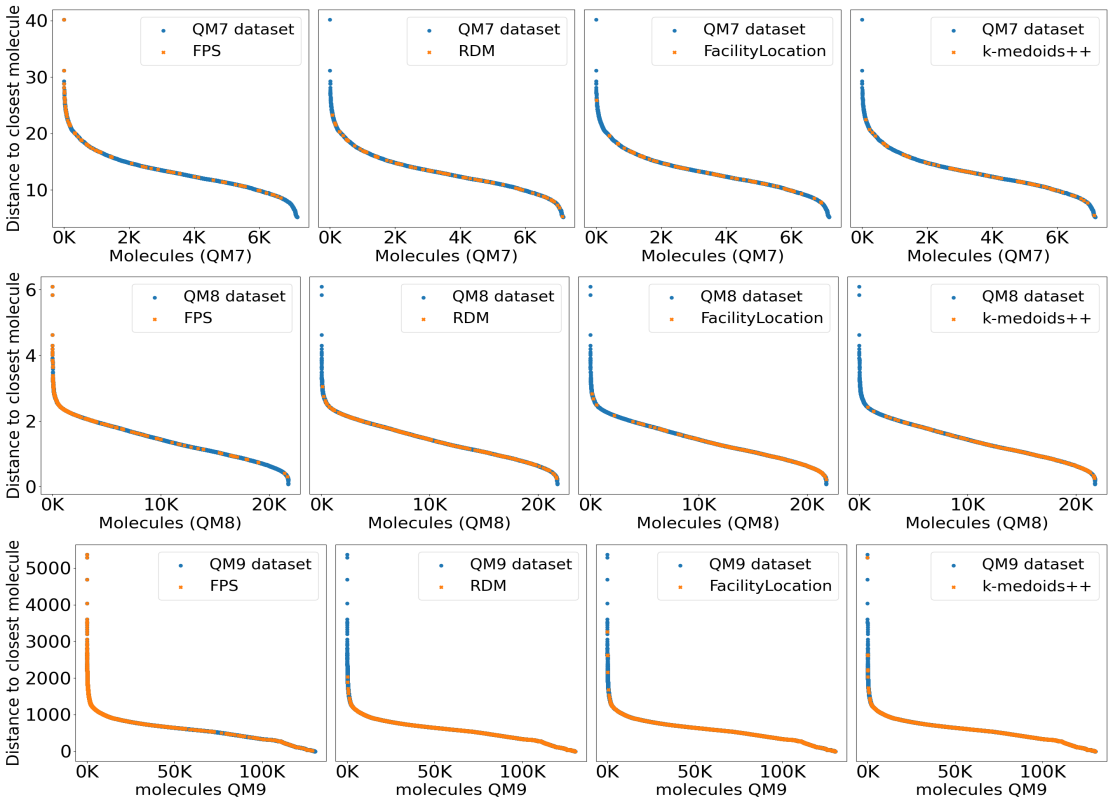
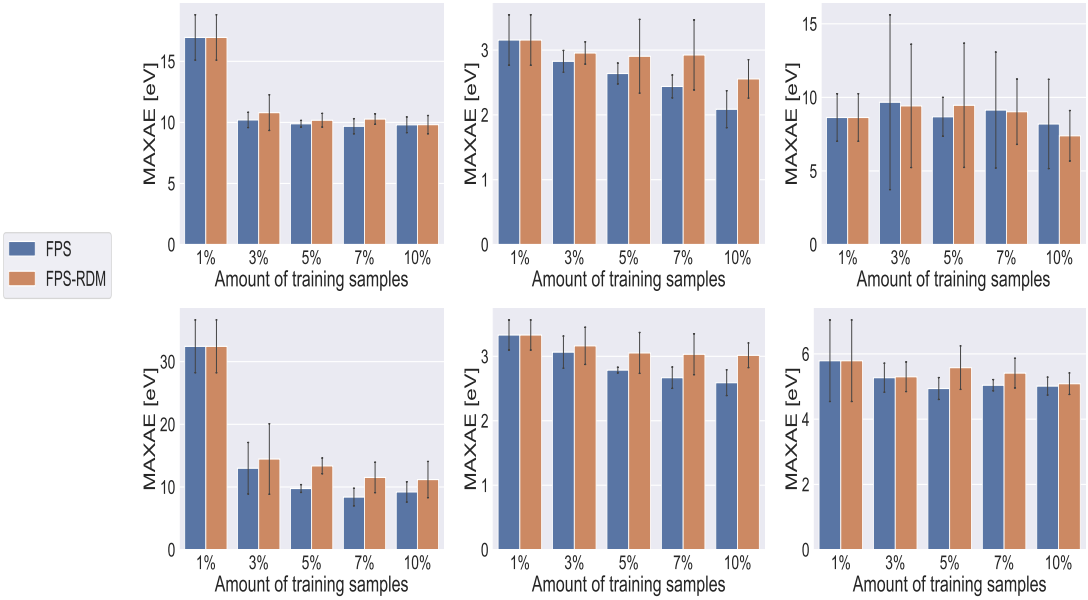


Figure 7.7.: In blue, the Euclidean distances to the nearest neighbor for molecules in the QM7 (top row), QM8 (middle row) and QM9 (bottom row). In orange are highlighted the molecules selected with FPS and the other baselines. For each dataset we selected 1% of available data points.

a low training data budget. Fig. 7.7 highlights the Euclidean distances to the closest neighbor for molecules selected with FPS, and the other baseline strategies, from all the analyzed datasets. The size of the selected sets is 1% of the available data points. Specifically, we are analyzing the same elements selected in the lowest training data budget we considered for the regression tasks in Fig. 7.1 and Fig. 7.2. Fig. 7.7 clearly illustrates that, independently of the dataset, FPS selects points across the whole density spectrum. On the contrary, the baseline methods mainly sample points that have a closer nearest neighbor and that are nearer to the center of the data distribution (Fig. 7.5c).

The observation that selecting isolated molecules is beneficial in terms of the MAXAE reduction is also in line with Theorem 5.1. We know that a sampling strategy that aims to reduce the maximum error of the predictions should minimize the fill distance of the training set. Thus, it should include the isolated molecules in the training set, as their distance to the nearest neighbor is much larger than the average

Our empirical analysis indicates that using FPS can be advantageous in the low training data budget, as it allows including early in the sampling process the “isolated” molecules. However, we think that once the data points at the tails of the data distribution have been



(a) QM7 (b) QM8 (c) QM9

Figure 7.8.: Results for regression tasks on QM7, QM8 and QM9 using KRR with the Gaussian kernel (top row) and FNN (bottom row) trained on sets of various sizes and selected with FPS and with FPS combined with random sampling (RDM) after 2% of the available data have been selected. The graphs illustrate the MAXAE of the predictions for each training set size and sampling approach. Error bars represent the standard deviation of the results over five runs. Selecting the initial 2% of data points with FPS, followed by RDM, produces results comparable to using FPS alone.

included, there may be more convenient sampling strategies than FPS to select points at the center of the distribution, where more information is available. To empirically support the observation that FPS is mostly beneficial in the low data limit, Fig. 7.8 illustrates the MAXAE of the predictions on the QM7, QM8 and QM9, for the KRR and FNN trained on sets selected with the FPS and on sets selected initially with the FPS, the first 2%, and then selected randomly. The figure clearly illustrates that after FPS has sampled the first 2% of the dataset, the MAXAE of the predictions does not tend to decrease or increase significantly for larger training set sizes, even if the later samples are selected randomly, independently of the datasets and regression model considered. This fact further suggests that FPS is mainly beneficial in the low data limit and is strongly connected with the ability of this sampling strategy to select samples at the tail of the data distribution.

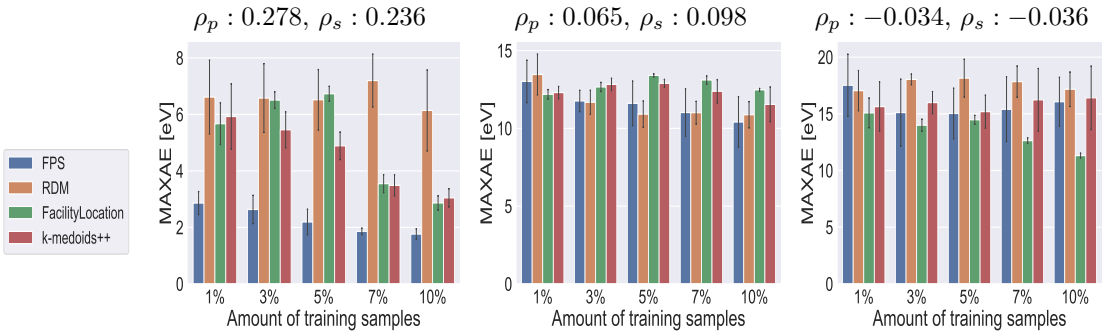
Importance of the data assumptions

We now highlight the importance of the data assumptions in ensuring that a fill distance minimization strategy leads to a significant reduction of the MAXAE, in correspondence to the theoretical result proposed in Theorem 5.1. The focus is on Assumption 5.1, Formula (5.6), indicating that if two data points have close representations in the feature space, then the conditional expectations of the associated labels are also close. Simply put, this assumption states that if two data points have similar features, their labels are more likely to be similar as well. Therefore, we expect the pairwise distances in the feature and label spaces to be directly correlated for the experiments to ensure consistency with the theory.

One approach to test such a correlation on a given dataset is to calculate the Euclidean distance in the feature and label spaces for each pair of points and then calculate Pearson's (ρ_p) or Spearman's (ρ_s) correlation coefficient [Bos08] to assess the strength and direction of the correlation between the pairwise distances. These coefficients measure how closely correlated two quantities are, with values ranging from -1 to 1. Pearson's coefficient captures linear relationships between variables, whereas Spearman's coefficient, measures monotonic relationships regardless of linearity. A positive value indicates a positive correlation, while a negative value indicates a negative correlation. Following along [SBS18], we define the correlation between the two analyzed quantities to be negligible if the considered correlation coefficient ρ is such that $|\rho| \leq 0.1$, otherwise we consider the correlation positive or negative, depending on the sign of ρ.

We calculate Pearson's (ρ_p) and Spearman's (ρ_s) coefficients for the datasets used in the experiments we perform in Section 7.1.3 and illustrate in Fig. 7.1 and Fig. 7.2. Due to memory issue in storing the distance matrix, for the QM9 we computed the correlation coefficients on 50% of randomly selected data points. We selected random subsets and computed the associated correlation coefficients for five times. The results we report for the QM9 are the average over the five runs. The computed coefficients are 0.15, 0.22, and 0.27 for ρ_p, and 0.28, 0.19, and 0.22 for ρ_s, for QM7, QM8, and QM9, respectively. These numbers indicate that in all experiments where the fill distance minimization approach is successful in significantly reducing the maximum prediction error, there is a positive correlation between the pairwise distances of the data features and labels.

Moreover, we also want to show that if the correlation between the pairwise distances in the feature and label space is negligible, the fill distance minimization approach may not lead to a significant reduction in the maximum prediction error. To illustrate this, we perform additional experiments on the QM8 dataset. In these experiments, we examine various labels not yet considered in this work while considering the same data features we previously used. The labels we now consider are the second singlet transition energy (E2), measured in eV, and the first and second oscillator strengths (f1 and f2), measured in atomic units (a.u.). Our computations reveal a Pearson's and Spearman's correlation coefficient of 0.278 and 0.236 for E2, respectively. As for correlations with f1 and f2, Pearson's coefficients are 0.065 and -0.034, respectively, while Spearman's coefficients are 0.098 and -0.036, accordingly. These results suggest a positive correlation between pairwise distances in the feature and label space when considering E2 as the label value,



(a) second singlet transition energy (E2)

(b) first oscillator strength (f1)

(c) second oscillator strength (f2)

Figure 7.9.: Results for regression tasks on QM8 considering three different labels: (a) second singlet transition energy, (b) first oscillator strength, (c) second oscillator strength. Regression performed using KRR trained on sets of various sizes and selected with different sampling strategies. ρ_p and ρ_s are Pearson's and Spearman's correlation coefficients of the data points pairwise distance in the feature and label space.

but negligible correlations for f1 and f2. Such a negligible correlation for f1 and f2 indicates that our initial assumptions about the data properties may not hold true when considering these two labels.

Fig. 7.9 shows the results for the regression tasks on the QM8 dataset considering E2, f1 and f2 as labels, and using the KRR as regression model. Specifically, Fig. 7.9b and 7.9c illustrate the MAXAE of the predictions for the regression tasks with f1 and f2, respectively, and suggest that selecting the training set by fill distance minimization using FPS, does not lead to a significant reduction in the maximum prediction error when the correlation between the pairwise distances in the feature and label space is negligible. On the contrary, Fig. 7.9a, illustrating the results on the E2 regression task, provides further evidence that the fill distance minimization approach is effective when the correlation is positive. Note that, for the case of the QM8 dataset with f1 or f2 as labels, where the correlation between pairwise distances in the features and label space is negligible, selecting training sets by fill distance minimization approach with the FPS is either comparable or better than randomly choosing the points in terms of the MAXAE of the predictions. Moreover, no benchmark approach can consistently perform better than FPS. For instance, the facility location approach performs best on the f2 regression task for training set sizes of 7% and 10%, but is the worst performing on the f1 regression task for all training set sizes other than 1%.

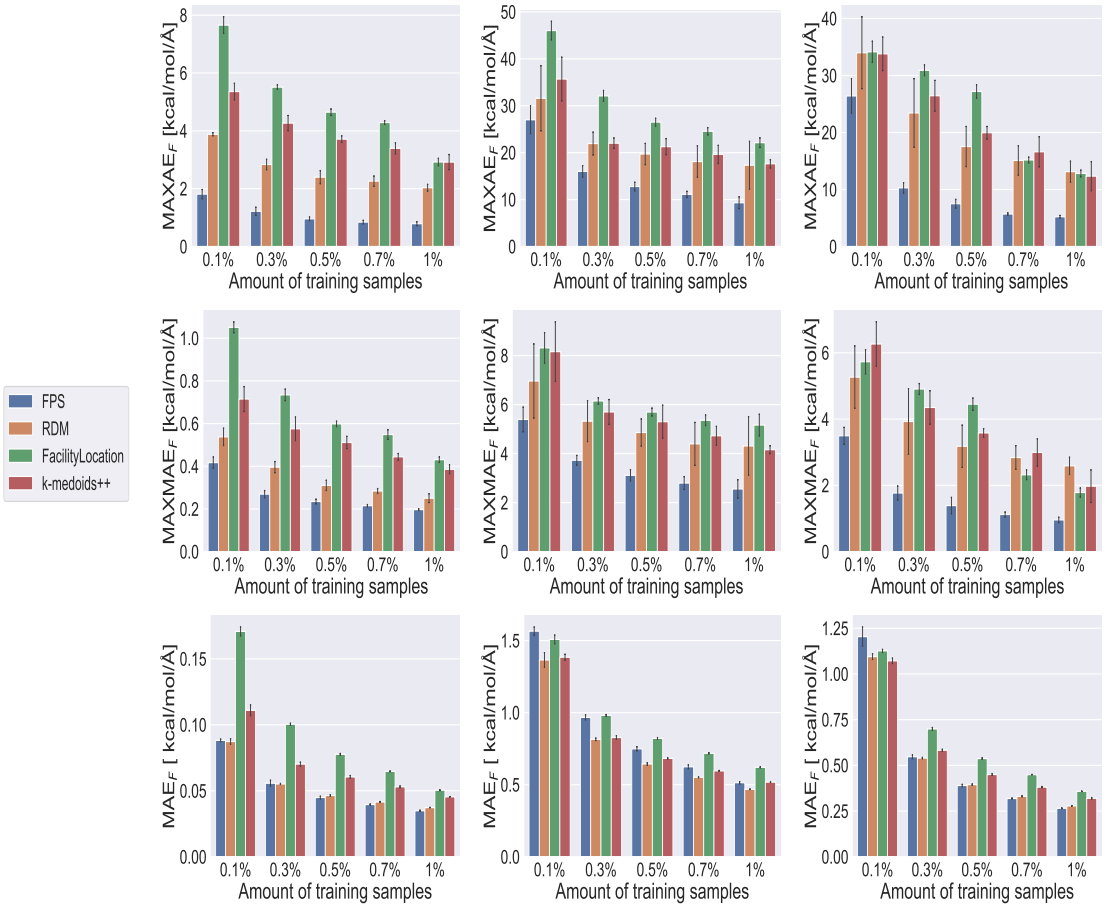
7.1.6. Force-field prediction on the rMD17 dataset

In this section, we empirically investigate the effects of minimizing the training set fill distance for multivariate regression tasks on the rMD17 dataset. The label value to predict is the molecular force-field along a molecule's trajectory. We note that for the

experiments on the rMD17 we analyze a different range for the size of the training sets, from 0.1% to 1% of the available points, instead of the range 1%- 10%. This change was made because the data points associated with each molecule in the rMD17 are taken from time series and may have a high degree of correlation. For this reason the authors of the rMD17 suggest "DO NOT train a model on more than 1000 samples from this dataset" [CvL20a]. Since each trajectory in our analysis consists of 100000 points, limiting ourselves to at most 1% of the available data ensures that we respect the constraint set by the authors. Fig. 7.10 shows the results for the force-field regression tasks on the trajectories of Benzene, Malonaldehyde and Uracil using GDML. The graphs on the top and middle rows of Fig. 7.10 illustrate the MAXAE_F and MAXMAE_F of the predictions on the unlabeled points. The results suggest that, independently of the trajectory considered, selecting the training set by fill distance minimization using FPS we can perform better than the baselines in terms of these two metrics quantifying the robustness of the model predictions. The graphs on the bottom row of Fig. 7.10 show the MAE_F of the predictions. These graphs indicate that selecting training sets with FPS may not improve the MAE_F of the predictions on the unlabeled points with respect to the baselines, similarly to the experiments with scalar labels. In particular, we observe examples where FPS performs worse than one of the baselines, e.g., on the trajectory of Malonaldehyde FPS performs consistently worse than random sampling. Nonetheless, these experiments suggest that selecting training set by fill distance minimization with the FPS can be beneficial for multivariate regression tasks, increasing models robustness by reducing of the entry-wise maximum error of the predictions. We remark that the theoretical analysis we propose in Chapter 5 is limited to regression tasks with scalar label values, and Lipschitz continuous models. Therefore, while the results illustrated in Fig. 7.10 show promising potential for the FPS in increasing model robustness in multivariate regression tasks, they are not supported by a solid theoretical result.

7.1.7. Section highlights

Our experiments indicate that selecting the training set using FPS significantly reduces the MAXAE of the predictions for KRR and FNN (Fig. 7.1 and Fig. 7.2). The empirical results align with the theoretical analysis from Chapter 5. Our empirical analysis (Section 7.1.5) suggests that FPS leads to a reduction of the MAXAE by maximizing data space coverage, ensuring the inclusion in the training set of points from the tails of the data distribution. FPS enhances model stability for KRR with the Gaussian kernel (Fig. 7.3), as expected from the theoretical results from numerical mathematics reported in Section 5.4. FPS is mainly beneficial in scenarios with limited training data. We find that selecting the initial 2% of data points with FPS, followed by random sampling, yields results comparable to using FPS alone in terms of the MAXAE (Fig. 7.8). Recall that our theoretical results from Chapter 5 focus on regression tasks with scalar labels, but our experiments show FPS can also improve the model robustness in tasks with multidimensional labels (Fig. 7.10). FPS does not seem to offer any advantage in terms of the MAE compared to random sampling and the other baselines. This suggests that alternative sampling strategies may be better suited for decreasing the MAE.



(a) Benzene (b) Malonaldehyde (c) Uracil

Figure 7.10.: Results for multivariate regression tasks on trajectories of Benzene (a), Malonaldehyde (b) and Uracil (c) from the rMD17 dataset. We use GDML trained on sets of various sizes, expressed as a percentage of the available data points in each trajectory, and selected with different sampling strategies. MAXAE$_F$ (top row), MAXMAE$_F$ (middle row) and MAE$_F$ (bottom row) are shown for each training set size and sampling approach. Error bars represent the standard deviation of the results over five runs. The results with the MAXAE$_F$ and MAXMAE$_F$, quantifying the robustness of the model predictions, suggest that selecting the training set using FPS (blue bars) leads to better performances than with the baselines, independently of the trajectory considered. The graphs on the bottom row showing the MAE$_F$ indicate that selecting training sets with FPS may not reduce the average error of the predictions with respect to the baselines.

7.2. Minimizing the weighted fill distance with DA-FPS

In this section we empirically validate the theoretical results presented in Chapter 6. That is, we investigate the effects of minimizing the training set weighted fill distance using DA-FPS for training data selection on regression tasks. In the previous section we saw that FPS is mainly beneficial in the low data budget scenario, reducing the magnitude of the worst-case prediction error. Moreover, the experiments suggested that FPS does not lead to any significant advantage in terms of the models MAE. In what follows, we consider a medium training data budget regime, instead of the low data budget scenario analyzed in the previous section. Thus, we focus on larger training set sizes and investigate a data regime where employing the FPS does not show any significant advantages in terms of the MAE, compared to classical passive and model-agnostic sampling approaches. We empirically show that in this data budget scenario using DA-FPS for training data selection reduces the MAE of the predictions of Lipschitz continuous regression models. Additionally, we empirically investigate the benefits of combining the FPS with other well-established passive and model-agnostic sampling strategies and show that augmenting such approaches with the FPS during the initial steps of the sampling process consistently leads to a decrease in the MAE of the predictions.

We consider the same datasets, regression models and tasks used in Section 7.1 for univariate regression. That is, we perform experiments where we predict molecular properties on the QM7, QM8 and QM9 datasets using the KRR with the Gaussian kernel and FNN as regression models.

Let us remark that the final goal of our experiments is to empirically show the benefits of using DA-FPS compared to other model-agnostic sampling approaches and investigate the benefits of complementing classical passive and model-agnostic sampling approaches with the FPS. We do not make any claims on the general prediction quality of the employed models on any of the studied datasets.

7.2.1. Baseline sampling strategies for DA-FPS

We benchmark DA-FPS with the four sampling techniques used in the previous section. Specifically, we consider uniform random sampling (RDM), Farthest Point Sampling (FPS), the facility location algorithm [Fri74] and k-medoids++ [MJH$^+$11]. Additionally, to empirically investigate the benefits of combining a given passive and model-agnostic sampling technique with FPS, we also consider modified versions of the RDM, facility location and k-medoids++. These modified versions of the baselines first select a prefixed amount of points with FPS, the same amount we consider for DA-FPS, and then augment the selected set by sampling from the remaining points in the data pool according to their specific criteria. Consequently, in the early stage of the sampling process, the sets selected with our proposed approach coincide with those selected with FPS and the modified baselines. We refer to the modified baselines as FPS-RDM, FPS-FacLoc and FPS-k-medoids++.

7.2.2. Experimental setup with DA-FPS

We analyze the effectiveness of DA-FPS considering training sets of sizes from 5% up to 20% of the available points in the data pool of interest. Note that, previously we investigated the benefits of the FPS considering a lower data budget regime, with training set sizes from 1% up to 10% of the available data pool. On the one hand, we now focus on larger training set sizes to limit the issues affecting the effectiveness of DA-FPS, related to high-dimensional density estimation with limited data points. On the other hand, by considering larger training set sizes we can analyze scenarios where only employing FPS may not provide any significant advantages compared to classical passive and model-agnostic sampling approaches in terms of the MAE. Our experiments involve the QM datasets for molecular energy prediction tasks. The data preprocessing procedure is similar to that used in the previous section and described in Section 4.2. But, we use lower-dimensional feature vectors (up to 276 dimensions instead of up to 1300) again to mitigate issues related to high-dimensional data density estimations in (6.8), affecting the effectiveness of DA-FPS. To describe molecules in the QM7 [BR09, RTMvL12] we consider the upper triangular entries of the Coulomb matrix. Thus, in the case of QM7 each molecule is represented as vector in $\mathbb{R}^{276}$. The label value to predict is the atomization energy, measured in electronvolt (eV). For QM8 [RvDBR12, RHTvL15] and QM9 [RvDBR12, RDRvL14] we follow the same preprocessing procedure as in Section 4.2. In addition to that we scale the entries of the molecular descriptors of the QM9 dataset in the interval (0,1) and perform dimensionality reduction applying principal component analysis (PCA) [JC16] on both, QM8 and QM9. Thus, molecules in the QM8 and QM9 datasets are represented by feature vectors in $\mathbb{R}^{100}$ describing the molecules topological structure. The label values to predict are the lowest singlet transition energy, and HOMO-LUMO energy, measured in eV, for the QM8 and QM9, respectively.

We use KRR with the Gaussian Kernel and FNN as regression models. We adapt the hyperparameter grid search for the KRR and the number of epochs to train the FNN to the new data scenario consisting of different training set sizes and lower dimensional molecular descriptor. Specifically, for the hyperparameter grid search related to KRR, we vary the hyperparameters on a smaller tensor product grid of 6 points per dimension between 10^{-6} and 10^{-2}. To train the FNN, we use the procedure described in Section 4.3 considering 1000 epochs independently of the dataset. In this setup, the lower-dimensional feature vectors result in fewer weights for the FNN to learn. This allows us to train for more epochs than what we consider in the previous Section 7.1 without significant computational costs. Training the FNN on the QM9 dataset with all selected subsets took about six days (the most computationally demanding case). The training was conducted sequentially on a 48-core CPU with 384 GB of RAM.

We test the predictive accuracy of each trained model on all data points not used for training in terms of the MAE, RMSE and MAXAE. These quantities have been already recalled in the previous section and interpreted in Section 4.4. We remark that, the main metric of interest for these experiments the MAE. However, analyzing RMSE and MAXAE, may provide further insights into the performance of DA-FPS compared to the other baselines.

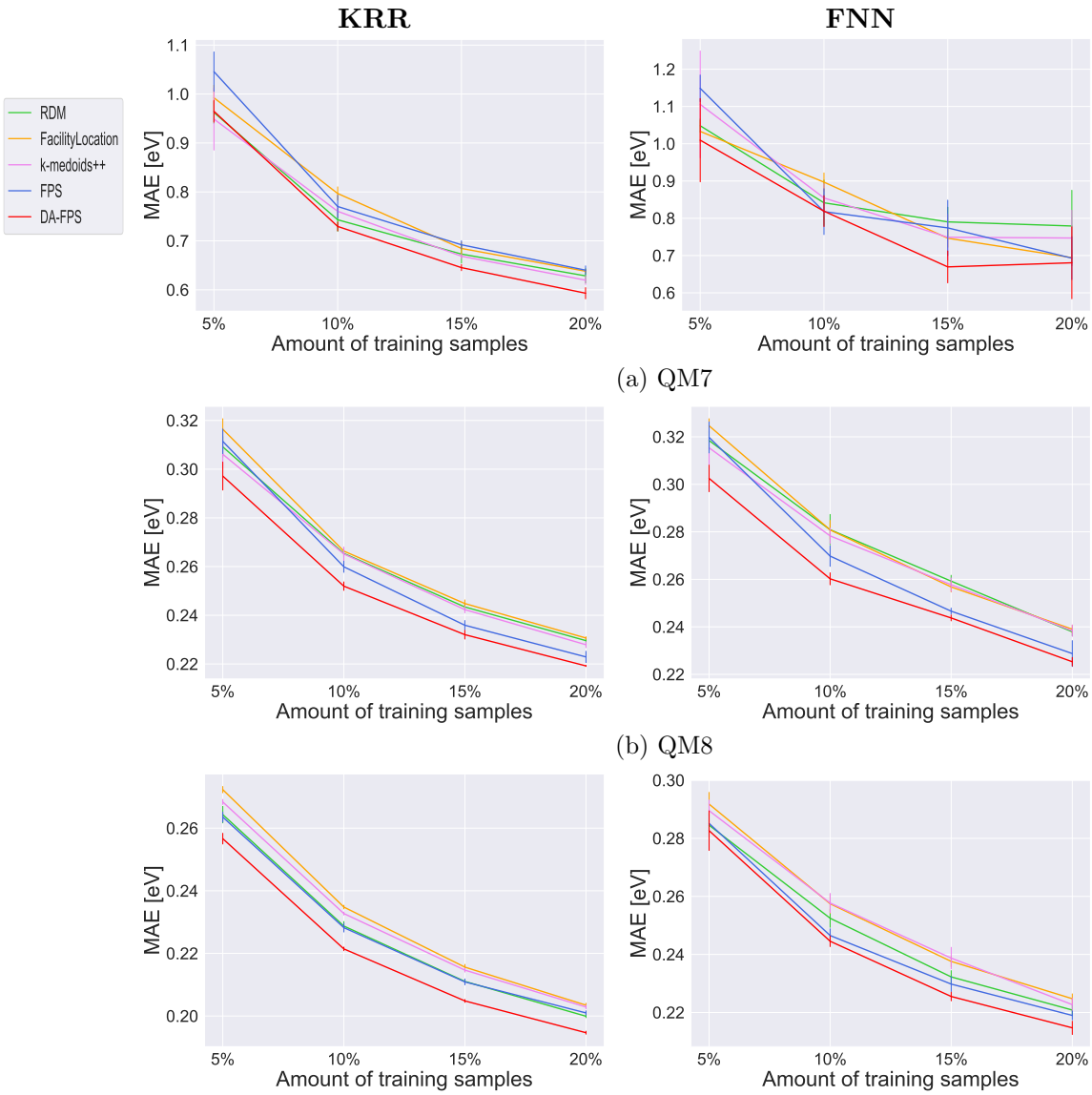
We consider various training set sizes for each sampling strategy. For each sampling strategy and set size, the training set selection process is independently run five times considering different initializations, that is, different initial point or random seed. Accordingly, we report for each analyzed model the average of the results for five runs. We also plot error bars, which represent the standard deviation over the five runs.

The number of k-NNs we consider to initialize DA-FPS (Algorithm 6) is 100. We think that a good choice for the value of k is data dependent and relies on various factors, such as data dimensions and distribution. Nonetheless, while we think that there may be better choices, we could find a value of k suitable for all the QM datasets. Determining the optimal value of k requires further research as it is closely tied to the problem of high-dimensional density approximation, which is beyond the scope of this study. The input parameter u is set to be 3% of the available data points, for each of the considered datasets. Thus, we first sample 3% of the available data considering constant weights, as done by FPS. We note again that setting the parameter u to be 3% of the available data points implies that our proposed approach coincides with FPS, FPS-RDM, FPS-FacLoc and FPS-k-medoids++ until 3% of the available data has been sampled.

In this section, we use line plots to visualize results. In the previous section, we used bar plots to highlight the large differences in the magnitude of predictions' MAXAE. The value of the MAE tends to be more similar across various training sets because it averages errors, smoothing out extremes. We find that line plots are better suited for capturing trends and subtle variations of the MAE, particularly when comparing up to six different strategies for each training set size.

7.2.3. Experiments with DA-FPS: Molecular property prediction

The graphs on Fig. 7.11 show the MAE for the regression tasks on the QM7, QM8 and QM9 datasets using KRR (left column) and FNN (right column). Our experiments indicate that selecting the training sets using DA-FPS we can perform better than the baselines RDM, facility location, FPS and k-medoids++ in terms of the MAE of the predictions, particularly on the larger QM8 and QM9 and for larger training set sizes ($> 5\%$ of the available data points). Comparing the MAE obtained with KRR and FNN, reveals that the standard deviation of the MAE, represented by the error bars, tends to be larger for the FNN, especially on the smaller QM7. This occurs because the FNN training process is data hungry, and it is sensitive to the limited amount of training data available with the QM7. More specifically, the FNN determines its regression parameters by solving a non-linear optimization problem using stochastic gradient descent. This training approach is inherently prone to variability and instability. The stochastic nature of gradient descent introduces randomness in parameter updates, and the complexity of the non-linear optimization landscape, with infinite possible solution, can lead to greater fluctuations in performance. As a result, metrics like the MAE often exhibit larger standard deviations, particularly for smaller datasets where the amount of data is insufficient to stabilize the training process. In contrast, KRR solves a convex optimization problem with a closed-form solution (4.5). This deterministic approach results in lower variability in predictions and lower standard deviation of the MAE.



(a) QM7

(b) QM8

(c) QM9

Figure 7.11.: MAE for regression tasks on QM datasets using KRR with Gaussian kernel (left column) and FNN (right column) trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. Error bars represent the standard deviation over five runs. DA-FPS (red lines) outperforms the baselines. FNN exhibits larger MAE standard deviations than KRR, particularly on QM7. This is due to its data-intensive stochastic gradient descent training, which increases variability in parameters and predictions. In contrast, KRR's deterministic closed-form solution results in smaller variability.

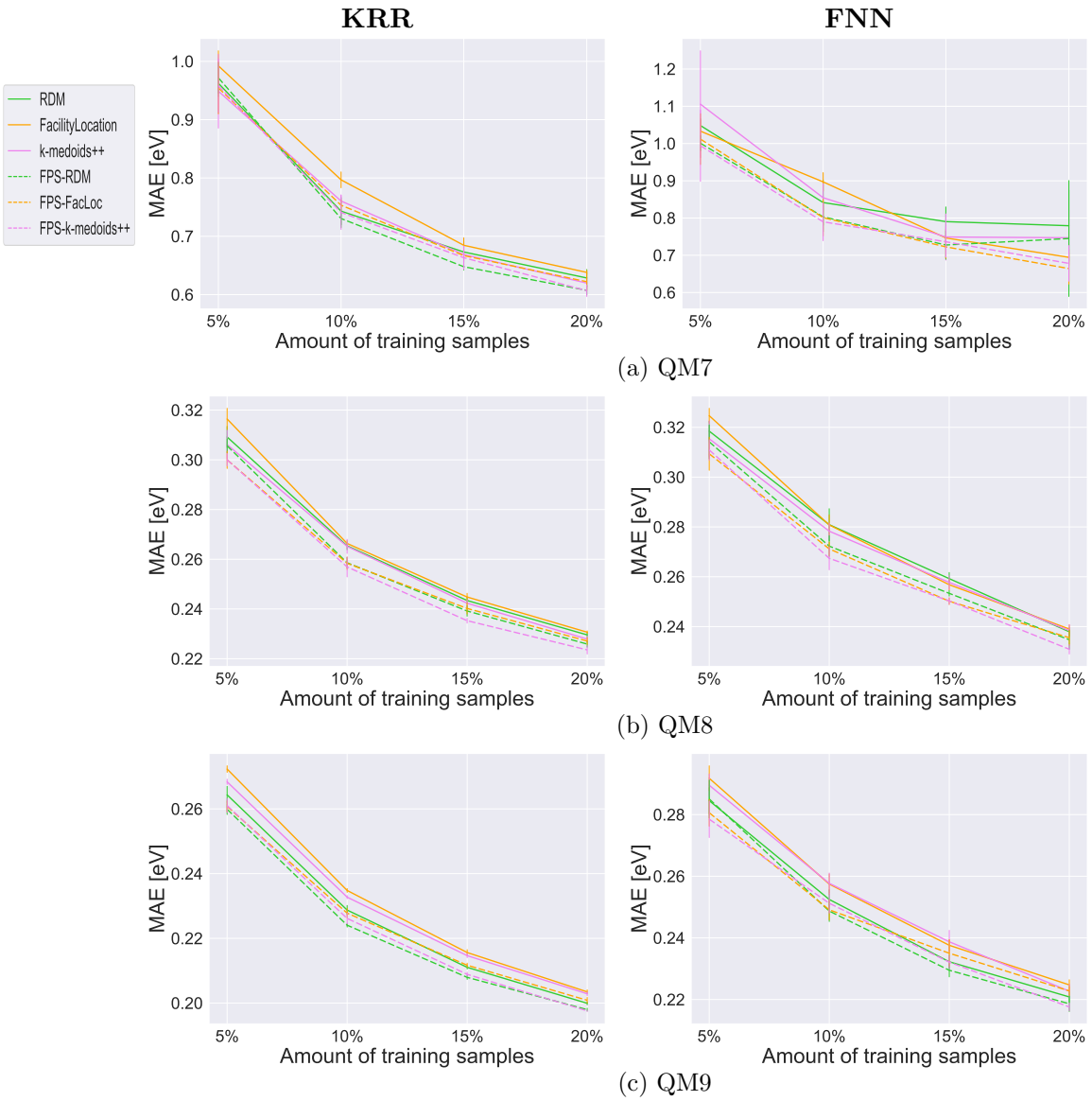
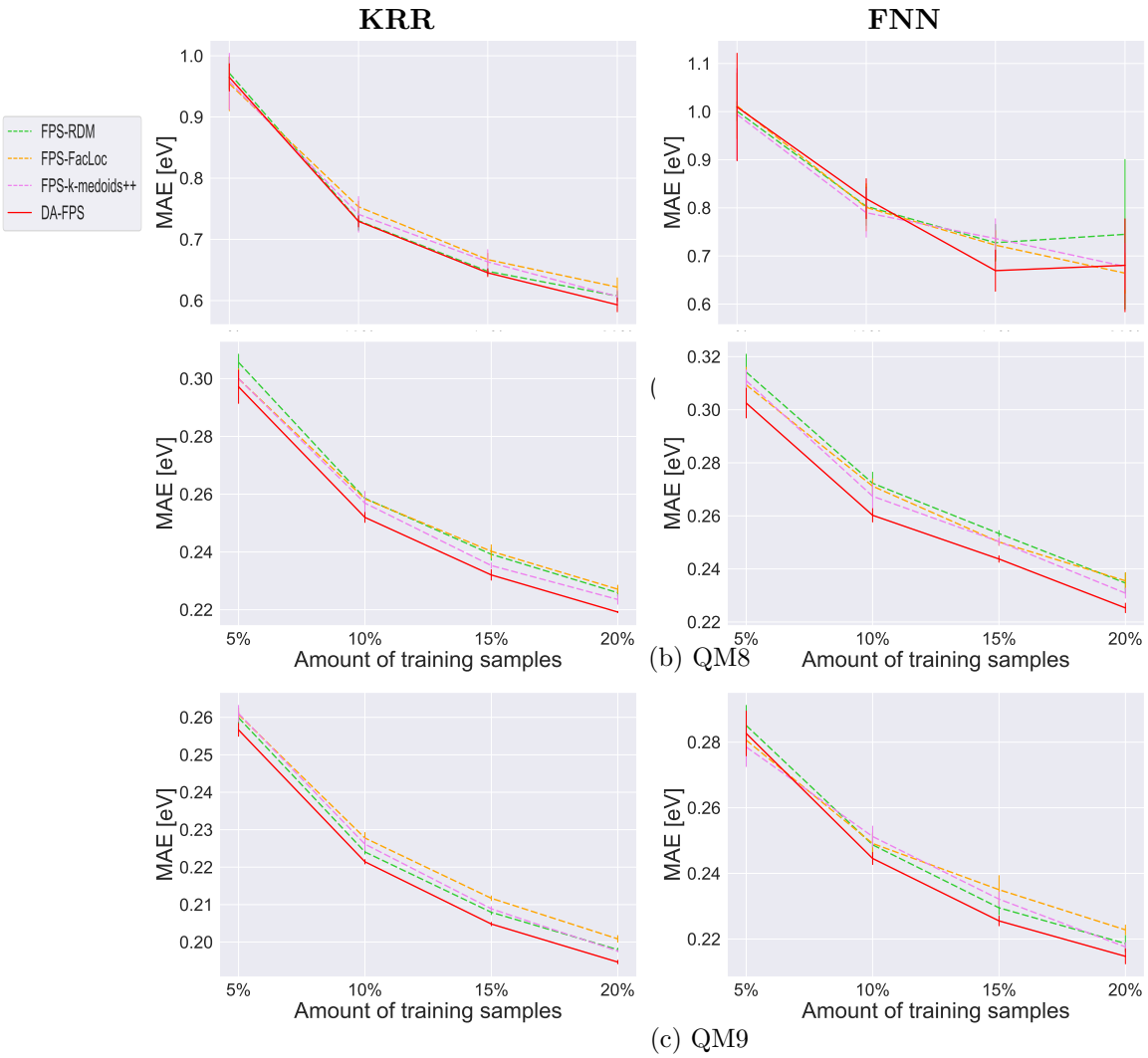


Figure 7.12.: MAE for regression tasks on QM datasets using KRR with Gaussian kernel (left column) and FNN (right column) trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. Error bars represent the standard deviation over five runs. The modified versions of the baselines (dashed lines) lead to better performance than the respective original baselines (solid lines).

Fig. 7.12 shows that FPS-RDM, FPS-FacLoc and FPS-k-medoids++ consistently outperform the associated baselines RDM, facility location and k-medoids++ . These results suggest that modifying the baselines by initially sampling with FPS leads to a reduction of the MAE independently of the dataset and trained model.



(b) QM8

(c) QM9

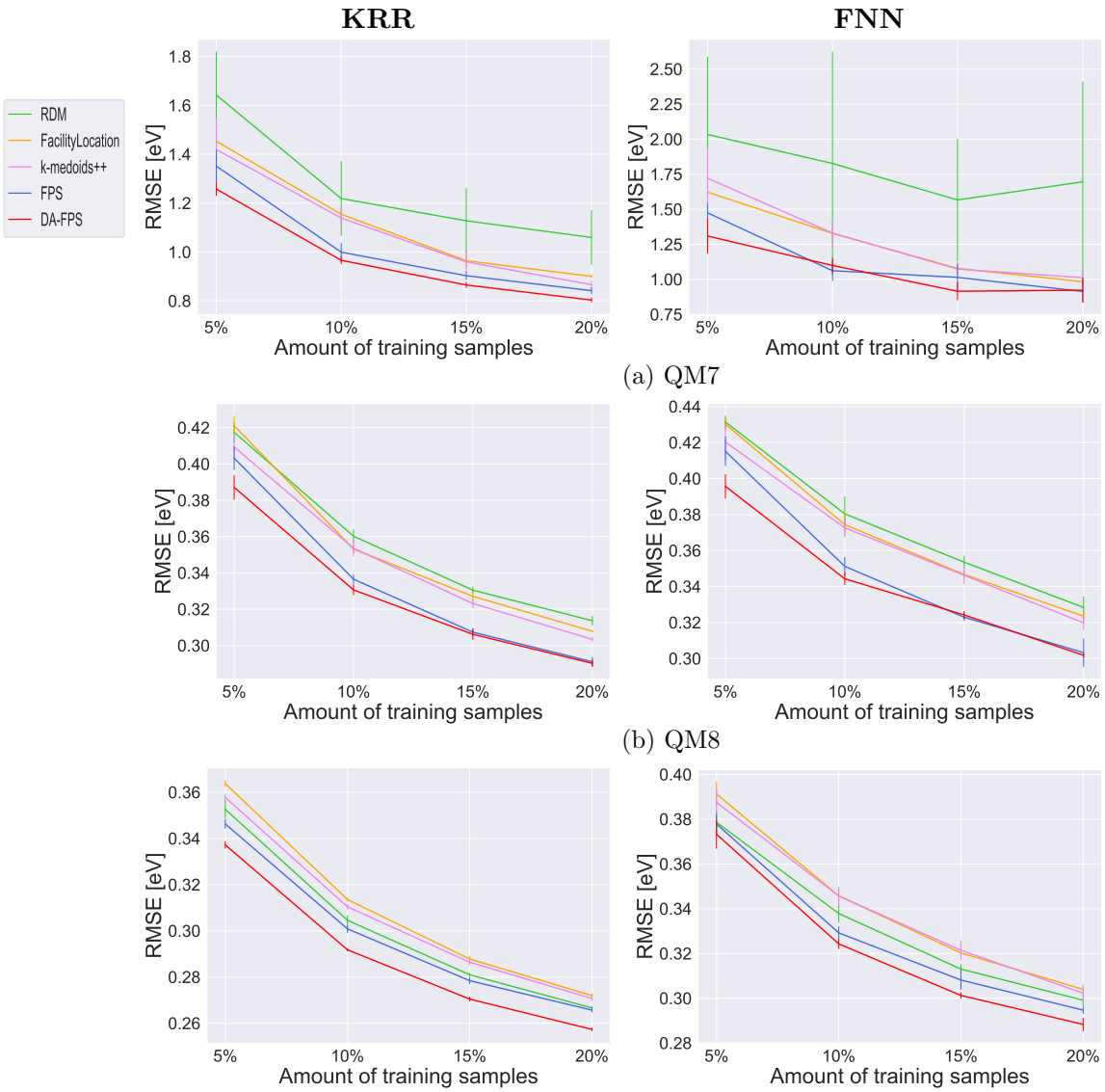
Figure 7.13.: MAE for regression tasks on QM datasets using KRR with Gaussian kernel (left column) and FNN (right column) trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. Error bars represent the standard deviation over five runs. DA-FPS (red lines) outperforms the modified baselines. FNN exhibits larger MAE standard deviations than KRR, particularly on QM7. This is due to its data-intensive stochastic gradient descent training, which increases variability in parameters and predictions. In contrast, KRR's deterministic closed-form solution results in smaller variability.

Fig. 7.13 compares DA-FPS with the modified baselines. Overall, DA-FPS leads to lower MAE of the regression models.

Looking at the results in Fig. 7.13 obtained with FNN, we can highlight two scenarios where DA-FPS is outperformed the modified baselines: on the smaller QM7 and on the QM9 for the 5% training set size. These results suggest that, the advantages of using DA-FPS with respect to the modified baselines may be less evident on smaller datasets and training set sizes ($\leq 5\%$ of the available points), when using the FNN as learning model. Recall that, FNN predictions are prone to instability for lower training set sizes. Nonetheless, our experiments still indicate that, overall, DA-FPS is more competitive than the modified baselines in terms of the MAE of the predictions. In particular, no modified baseline can consistently outperform DA-FPS in any of the datasets considered. Moreover, according to our experiments, the comparative effectiveness of the modified baselines, in terms of the MAE, depends on the dataset considered, that is, on the underlying data distribution. For instance, in Fig. 7.13, if we consider KRR as the regression model, FPS-RDM outperforms FPS-k-medoids++ on QM9. The opposite is true on QM8. The results with DA-FPS appear to be more robust to changes in the datasets. We think this is because DA-FPS considers data density during the sampling process, allowing it to adapt effectively to changes in the underlying data distribution.

The graphs in Fig. 7.14 compare the performances of DA-FPS with the baselines and illustrate the RMSE of the predictions for the KRR and FNN. The illustrated results suggest that the performances of FPS may be closer to that of DA-FPS when we consider the RMSE. This is particularly evident on the smaller QM7 and for larger set sizes on the QM8. Note that the RMSE penalizes large errors more than the MAE, thus giving more relevance to outlier error values. As we know from the results in Section 7.1.3, FPS leads to a substantial decrease in maximum prediction error and hence reduces the amount and magnitude of large error values. Nevertheless, DA-FPS still leads competitive performances in terms of the RMSE, highlighting its robustness against large errors. Fig. 7.15 compares DA-FPS with the modified baselines in terms of the RMSE. The performances of the modified baselines may be closer to the performance of DA-FPS than their non-modified version, e.g., on QM7 with KRR as regression model. Still, DA-FPS remains the most competitive approach. No modified baseline can consistently outperform DA-FPS in any of the datasets.

Figure 7.16 shows the MAXAE of predictions on the QM datasets using KRR (right) and FNN (left) as regression models. The results indicate that minimizing the training set fill distance with FPS remains effective for reducing MAXAE, even with larger training sets and lower-dimensional feature vectors than those in Section 7.1.3. FPS consistently outperforms all other methods across scenarios. DA-FPS achieves a MAXAE of a similar magnitude to FPS. Additionally, we note that, modifying baseline approaches by starting with FPS significantly reduces MAXAE, as suggested by the empirical analysis in Section 7.1.5 and the experiments in Fig. 7.8, where we compare the results obtained with RDM and FPS-RDM. This was also confirmed by additional experiments that we do not report in this work to avoid supplementary experiments that do not provide any additional information.

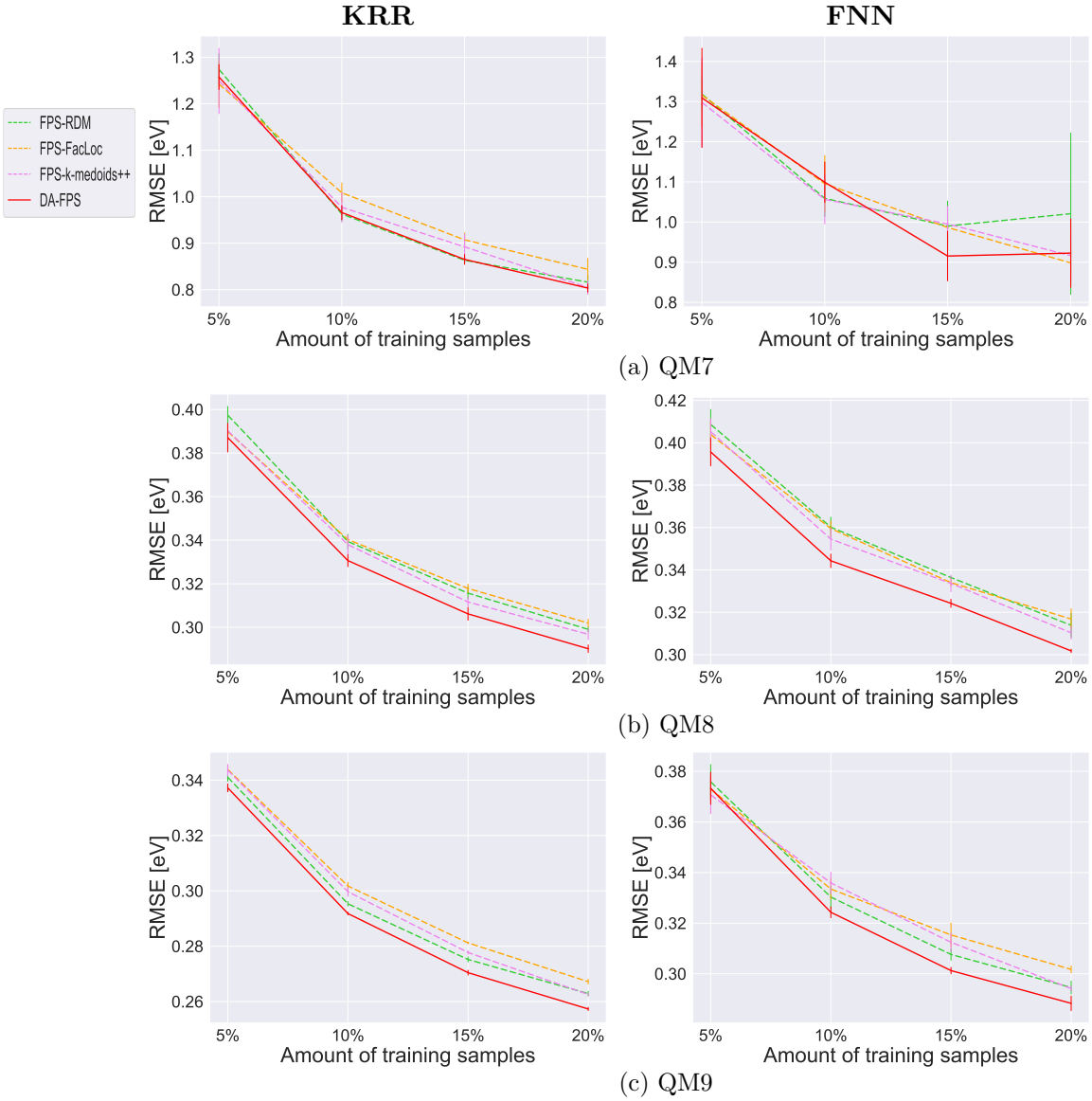


(a) QM7

(b) QM8

(c) QM9

Figure 7.14.: RMSE for regression tasks on QM datasets using KRR with Gaussian kernel (left column) and FNN (right column) trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. Error bars represent the standard deviation over five runs. The performances of FPS (blue lines) may be close to that of DA-FPS (red lines) when we consider the RMSE, particularly for larger training set sizes. This is particularly evident on the smaller QM7 and for larger set sizes on the QM8. Nevertheless, DA-FPS still leads to the most competitive performances across datasets.

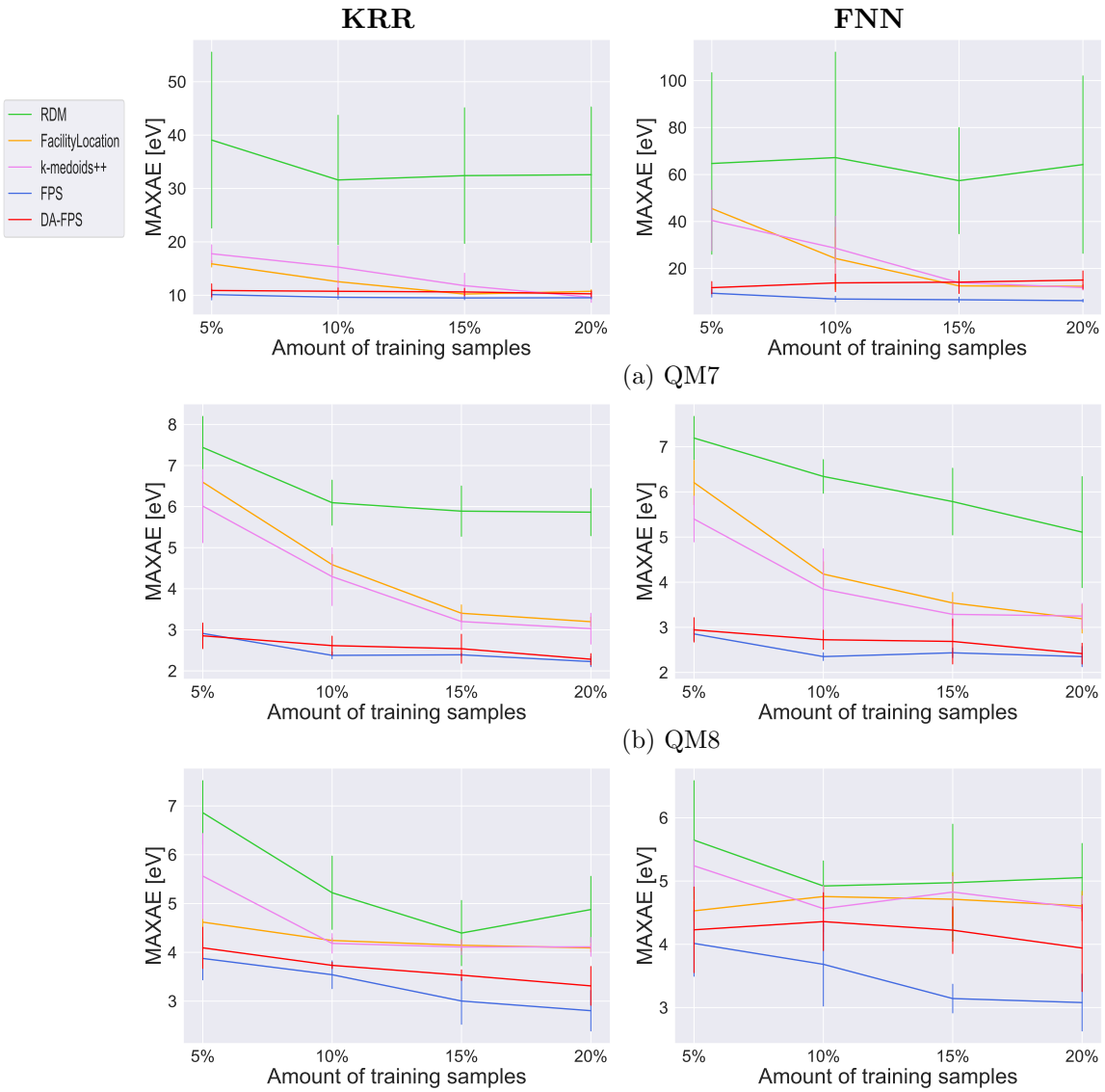


(a) QM7

(b) QM8

(c) QM9

Figure 7.15.: RMSE for regression tasks on QM datasets using KRR with Gaussian kernel (left column) and FNN (right column) trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. Error bars represent the standard deviation over five runs. DA-FPS (red lines) is the most competitive approach. No modified baseline (dashed lines) can consistently outperform DA-FPS in any of the datasets.



(a) QM7

(b) QM8

(c) QM9

Figure 7.16.: MAXAE for regression tasks on QM datasets using KRR with Gaussian kernel (left column) and FNN (right column) trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. Error bars represent the standard deviation over five runs. FPS (blue lines) consistently outperforms all other methods. DA-FPS (red lines) achieves a MAXAE of a similar magnitude to FPS.

7.2.4. Ablation study DA-FPS hyperparameters on ZINC dataset

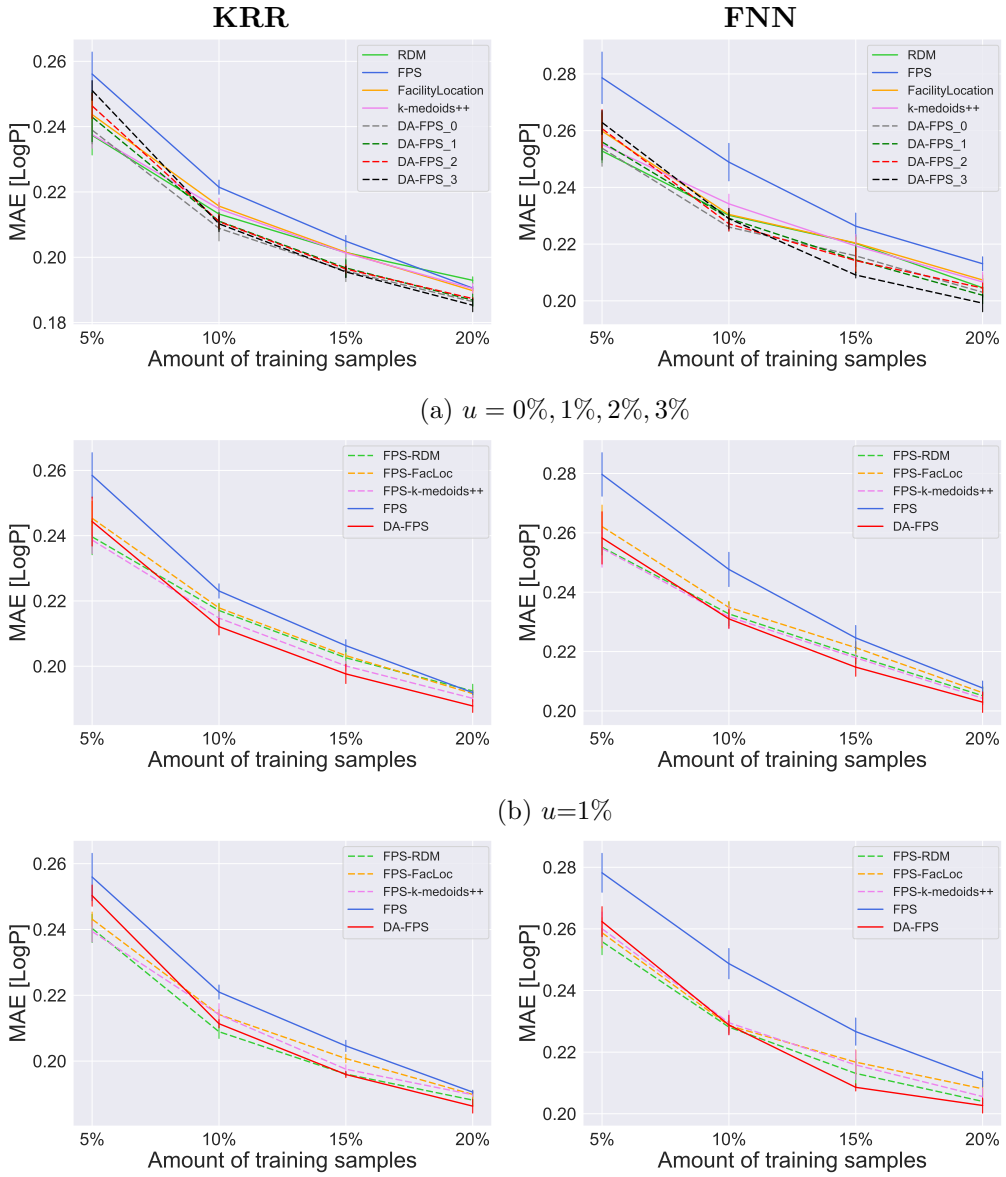
Next, we analyze how the performances of DA-FPS may be affected as we vary the hyperparameter u, defining the amount of samples initially selected with uniform weights and the hyperparameter k, defining the amount of k-nearest neighbors considered for computing the weights. We perform this analysis on the ZINC dataset and use the KRR and FNN as regression models. We consider a version of the ZINC dataset consisting of 24000 molecules represented as vectors in $\mathbb{R}^{100}$. We aim to predict the molecules' LogP value, describing the molecules' solubility. In Section 4.2 we provide the details on the descriptors, label values, and preprocessing procedures we use. ZINC provides one additional application scenario to those already considered in previous sections.

Hyperparameter "u"

To study how DA-FPS performs as the hyperparameter u varies, we fix $k = 300$. The graphs in Fig. 7.17a illustrate how the performance of DA-FPS (dashed lines) changes as the parameter u varies, compared to the classical baseline approaches (solid lines). We consider $u = 0\%, 1\%, 2\%, 3\%$. For the low data budget of 5%, we see that random sampling and k-medoids++ lead to better results than DA-FPS, particularly if we consider larger values of u for DA-FPS. This is more evident with KRR. On the contrary, for the larger training set sizes of 10%, 15% and 20%, DA-FPS consistently outperforms all the baselines independently of the choice of u. Moreover, for the smallest training set size of 5% we see that the smaller the value of u the more accurate the average predictions obtained considering training set selected with DA-FPS, independently of the regression model. We note that, such a trend may change or even be reverted if we consider larger training set sizes of 10%, 15% and 20%. This is particularly evident in the graph of Fig. 7.17a related to the FNN model, where, for training set sizes of 15% and 20%, the larger the value of u the more accurate the average predictions with DA-FPS. Thus, from our experiments, we see that the parameter u may affect the performance of DA-FPS differently, depending on the training set size and regression model considered.

The graphs in Fig. 7.17b compare DA-FPS and the modified baselines. They show the results obtained by initializing DA-FPS and the modified baselines with $u = 1\%$ of the available data points. That is, DA-FPS and the modified baselines coincide with FPS until 1% of the available data points has been selected. The results align with those of the experiments performed in Section 7.2.3. In particular, the graphs in Fig. 7.17b show that, DA-FPS tends to outperform the modified baselines in terms of the MAE of the regression models, particularly for larger training set sizes ($> 5\%$).

Fig. 7.17c illustrates the performance of DA-FPS and the modified baselines considering $u = 3\%$ of the available data. The graphs suggest that, by increasing the parameter u form 1% to 3% the gap between the modified baselines and DA-FPS reduces for KRR and increases FNN. This indicates that the choice of u has an impact on the relative effectiveness of the modified baselines and DA-FPS and that such impact also depends on the model used for the regression task. Nonetheless, independently of the choice u, DA-FPS is the best or second best performing for the larger training set sizes ($> 10\%$).

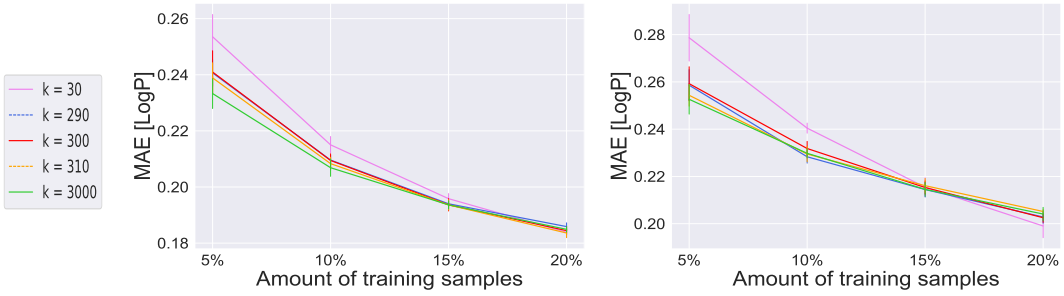


(a) $u = 0\%, 1\%, 2\%, 3\%$

(b) u=1%

(c) u=3%

Figure 7.17.: Sensitivity study DA-FPS hyperparameter "u": MAE for regression tasks on ZINC using KRR (left column) and FNN (right column) trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. DA-FPS is implemented with $k = 300$ and considering various values for u. In (a) DA-FPS is implemented with $u = 0\%, 1\%, 2\%, 3\%$. DA-FPS and the modified baselines are implemented with $u = 1\%$ in (b) and $u = 3\%$ in (c). Error bars represent the standard deviation over five runs.



(a) KRR (b) FNN

Figure 7.18.: Sensitivity study DA-FPS hyperparameter "k": MAE for regression tasks on ZINC using KRR with Gaussian kernel (a) and FNN (b) trained on sets of various sizes, and selected with DA-FPS considering $u = 1\%$ and $k = 30, 290, 300, 310, 3000$. Error bars represent the standard deviation over five runs. The performance of DA-FPS remains stable when the hyperparameter k is within a certain range ($k = 290, 300, 310$). Choosing k too small may lead to a notable performance decline.

Hyperparameter "k"

To study how DA-FPS performs as the hyperparameter k varies, we fix u=1%. We choose u=1% because, according to the results in Fig. 7.17, it provides the better overall results across the various training set sizes considered. Fig. 7.18 reports experiments where we investigate the effects of varying the DA-FPS parameter k for regression tasks with KRR and FNN on the ZINC dataset. We select training sets of various sizes with DA-FPS, considering different values of k ($k = 30, 290, 300, 310, 3000$).

These exemplary results suggest that the proposed value $k = 300$ (used for the experiments in Fig. 7.17) falls within a range where small changes would not negatively impact the effectiveness of our approach on the ZINC datasets. Using $k = 290$ and 310 does not lead to substantial differences, independently of the training set size. However, applying changes of an order of magnitude to k could potentially affect the approach. For instance, changing the value of $k = 300$ by a factor of 10 to $k = 30$ leads to a significant decrease in performance for the training set size of 5%, independently of the regression model. The rough magnitude of k likely depends on the data dimension and distribution and is generally investigated in the context of density estimation. We expect further insights from that domain.

Data assumptions

So far, in this section, we empirically validated the theoretical results presented in Chapter 6. That is, we saw that attempting to minimize the estimated training set weighted fill distance using DA-FPS leads to an improvement of the average prediction performance of regression models. We note that, for the experiments to be consistent with the theory developed in Chapter 6, the datasets we use should respect the data

assumptions required in Theorem 6.1. We focus on Assumption 5.2, more specifically Formula (5.6), indicating that if two data points have close representations in the feature space, then the conditional expectations of the associated labels are also close. This assumption is necessary to attain the theoretical result in Theorem 6.1.

For the QM datasets, we have already tested the assumption in Section 7.2 where we perform experiments related to FPS. However, for the experiments with DA-FPS we use different feature vectors. Thus, it is worth to verify whether the new feature vectors we consider still respect the required assumptions or not.

We use the same procedure employed in Section 7.2 and study the correlation between pairwise distances in the feature and labels spaces by computing the Pearson's (ρ_p) and Spearman's (ρ_s) correlation coefficients. We recall that these coefficients measure and quantify the correlation between the quantities of interest. We compute the correlation coefficients for the pairwise distances in the feature and label spaces on the QM7, QM8, QM9 and ZINC. We consider the features and labels used for the experiments illustrated in Section 7.2.3 and Section 7.2.4.

Due to memory issue in storing the distance matrix, for the QM9 we computed the correlation coefficients on 50% of randomly selected data points. We selected random subsets and computed the associated correlation coefficients for five times. The results we report for the QM9 are the average over the five runs. The computed coefficients are 0.15, 0.22, 0.26 and 0.22 for ρ_p, and 0.27, 0.19, 0.22 and 0.19 forρ_s, for QM7, QM8, QM9 and ZINC, respectively. Thus, both the Pearson's and Spearman's coefficients indicate a positive correlation between the pairwise distances of the data features and labels, suggesting that the data assumption considered in Theorem 5.1 is respected for each of the considered datasets.

We remark that this check on the data assumption based on the correlation coefficient between pairwise distances in the feature and label space only provides a qualitative understanding of the data regularity. That is, it indicates that the data respects the assumption, but it does not confirm it. Moreover, it relies on the knowledge of the data labels. We are interested in scenarios where the labels and the map connecting the features and the labels are unknown. At this moment in time, we do not see any numerical approach that can provide a qualitative or quantitative understanding of the regularity of the mapping connecting data locations and associated labels without the knowledge of the labels. Thus, we think that researchers should rely on domain knowledge, as was our case, to understand whether data assumptions are respected or not.

7.2.5. Computational efficiency DA-FPS

Given the novelty of DA-FPS, it is important to investigate its computational complexity. DA-FPS can be implemented using $\mathcal{O}(|\mathcal{D}|k)$ memory and the greedy selection takes $\mathcal{O}(db|\mathcal{D}|k)$ time. $|\mathcal{D}|$ is the amount of available data points, k the amount of nearest neighbors we consider for the density approximation, b is the amount of points we select and d is the dimension of the data points. The computational cost of DA-FPS is determined by the weights update (line 7 in Algorithm 6) taking $\mathcal{O}(d|\mathcal{D}|k)$ at each of the b iterations. In the current implementation the weights update involves iterating over all

points in $\mathcal{D}$ and compute the distances between the new selected point and the points' k-nearest neighbors, which costs $\mathcal{O}(d|\mathcal{D}|k)$.

Note that, initializing DA-FPS requires the computation of the k-nearest neighbors matrix, which can be a potential bottleneck. In our implementation we query the k-nearest neighbors using the cKDtree algorithm from the SciPy python library [VGO+20]. The algorithm takes $\mathcal{O}(d|\mathcal{D}|\log|\mathcal{D}|)$ for building the balanced tree in the worst case scenario. After that, it queries the k-nearest neighbors with a worst-case cost of $\mathcal{O}(|\mathcal{D}|^{1-\frac{1}{d}})$ and an average cost of $\mathcal{O}(\log|\mathcal{D}|)$.

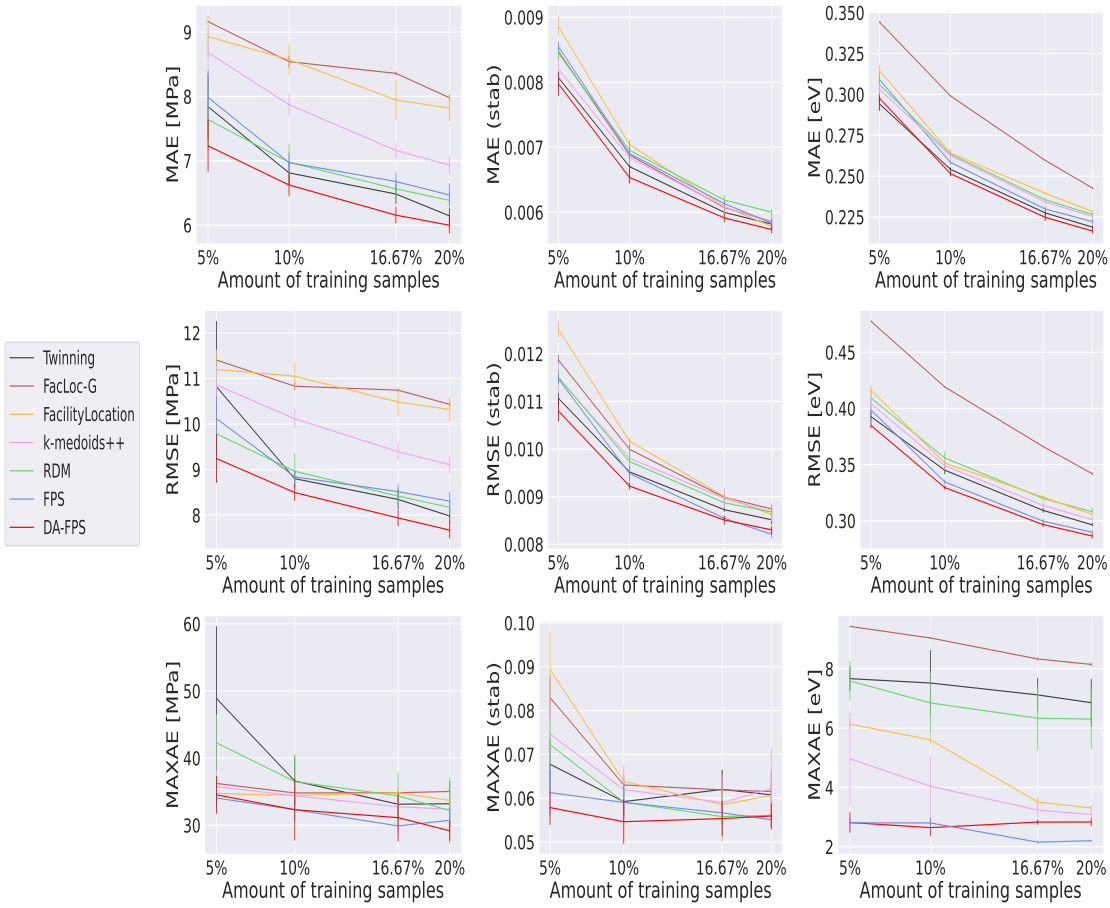
We implemented two versions of DA-FPS: one using NumPy [vdWCV11] and the other with PyTorch [PGM+19]. The average computation times (over five runs) to select 20% of data points from QM7, QM8, QM9, and ZINC are respectively as follows: NumPy - 6, 48, 1974, and 70 seconds; PyTorch - 4, 31, 968, and 33 seconds. This was conducted on a 48-core CPU with 384 GB RAM. DA-FPS was initialized with $u = 0$ and $k = 100$ independently of the dataset. DA-FPS' PyTorch implementation is faster than the NumPy implementation. This is because PyTorch can run computations exploiting multiple CPU cores. It uses libraries like OpenMP[DM98] and Math Kernel Library [WZS+14] to perform operations on multiple CPU cores, leading to faster computations.

7.2.6. Additional experiments

In this section we present experiments considering two additional datasets unrelated to quantum chemistry, two additional sampling strategies, and one additional kernel method. The new datasets are the Concrete Compressive Strength dataset and the Electrical Grid Stability Simulated Dataset, from the public UCI ML repository [DG17]. The Electrical Grid Stability dataset contains 10000 data points represented by 12-dimensional vectors. The target variable for regression is the stability margin (stab), reflecting grid stability. The Concrete dataset consists of 1030 data points of described by 8-dimensional vectors. The target variable is the compressive strength in megapascals (MPa). In Appendix B.2 we provide additional details on the datasets and preprocessing procedures. We also consider the QM8 to evaluate the performance of the additional sampling strategies and regression model in a quantum chemistry context.

The additional sampling strategies are the Twinning algorithm [VJ22], introduced in Section 3.5, and the facility location with a Gaussian similarity function (FacLocG) as defined in (3.13). Fine-tuning is required for the Gaussian width of the similarity function in facility location, and we follow the methodology outlined in [BCD+24], with complete details and hyperparameters described in Appendix B.3. Appendix B.3 also includes the hyperparameters used for DA-FPS. In these experiments, we compare DA-FPS with the additional sampling strategies alongside the baseline methods used in the previous section, including RDM, k-medoids++, facility location and FPS. We intentionally exclude the modified baselines analyzed earlier to keep the error plots clear and avoid introducing unnecessary complexity.

The Twinning algorithm implementation from [VJ22] only allows the selection of subsets of the size that can be expressed as an integer "r" representing the inverse of



(a) Concrete (b) Electrical Grid (c) QM8

Figure 7.19.: Results for regression tasks on the Concrete Compressive Strength, Electrical Grid Stability, and QM8 datasets. We use KRR with the Cauchy kernel trained on sets of various sizes, expressed as a percentage of the available data points, and selected with different sampling strategies. MAE (top row), RMSE (middle row) and MAXAE (bottom row) are shown for each training set size and sampling approach. Error bars represent the standard deviation of the results over five runs. DA-FPS (red lines) consistently showcases competitive performances across all metrics. For MAE, DA-FPS generally outperforms other methods, except Twinning (black lines) at 5% training set size on QM8. Twinning is the second-best method in terms of the MAE. For the RMSE, DA-FPS consistently ranks as the best or second-best. MAXAE results confirm DA-FPS as the best or second-approach, with FPS (blue lines) as the other most competitive approach. As for Twinning, despite strong MAE performance, it under-performs in MAXAE, sometimes worse than random sampling (green lines). Overall, DA-FPS delivers competitive performances across all metrics.

the partitioning ratio, i.e., for obtaining a training subset consisting of 20% of the data points, we must set $r = \frac{100}{20} = 5$. Consequently, to use the Twinning algorithm, we consider training set sizes similar to those considered in the previous section, but that can be expressed as an integer ratio. Specifically, we consider training set sizes of 5%, 10%, 16.67% and 20% associated with a ratio r of 20, 10, 6 and 5, respectively.

We use KRR with a Cauchy kernel as an additional regression method. We take the definition of Cauchy kernel used in [Bas08]. We describe the Cauchy kernel, its hyperparameters and the hyperparameters' optimization process in Appendix B.4.

For each sampling strategy and set size, the training set selection process is independently run five times considering different initializations, that is, different initial point or random seed. Accordingly, we report for each analyzed model the average and the standard deviation of the results for five runs.

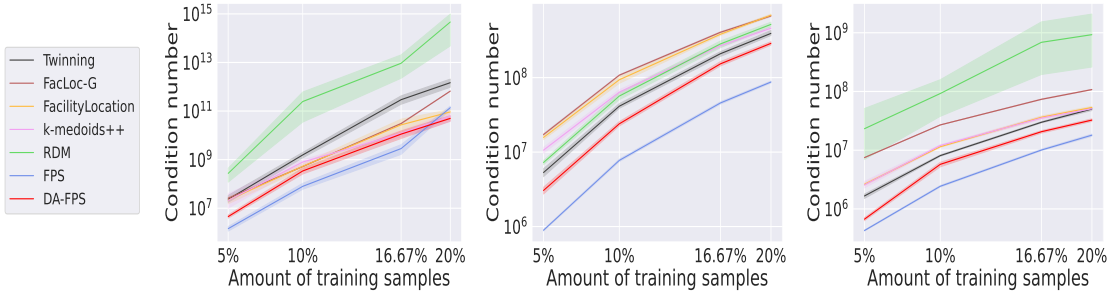
Fig. 7.19 presents the regression task results on the Concrete dataset, the Electrical Grid dataset, and the QM8 dataset, using KRR with a Cauchy kernel trained on datasets of various sizes, selected with different strategies. The top row of the figure shows the MAE of the predictions, the primary metric of interest. These results suggest that, overall, DA-FPS outperforms other methods across all datasets. The only exception occurs with a training set size of 5% on the QM8 dataset, where the Twinning approach performs slightly better. Nonetheless, the MAE plots indicate that Twinning is generally the second-best method regarding MAE.

The middle row of Fig. 7.19 presents the RMSE results, which indicate that DA-FPS consistently achieves strong performance, ranking as either the best or second-best method across all scenarios. Unlike the MAE results, where Twinning is the second best-performing approach, the RMSE results highlight FPS as the method most closely aligned with DA-FPS in terms of performance. This distinction underscores the reliability of DA-FPS across varying evaluation metrics. Since RMSE assigns greater weight to larger prediction errors compared to MAE, these findings emphasize the capability of DA-FPS to manage and mitigate significant prediction errors effectively.

The bottom row of Fig. 7.19 illustrates the MAXAE, further indicating DA-FPS as either the best or second-best method in every scenario. FPS emerges as the other best performing. It is worth to note that while the Twinning approach is the second best performing approach it terms of the MAE, it performs poorly in terms of MAXAE, especially when compared to DA-FPS and FPS. For example, Twinning is the worst-performing algorithm on the Concrete dataset with a training set size of 5% and the second-worst on the QM8 dataset, regardless of training set size. In both cases, Twinning performs even worse than random sampling.

The results in Fig. 7.19 underscore the effectiveness and adaptability of DA-FPS across different datasets, sampling strategies, and error metrics. These additional experiments further indicate that DA-FPS is a robust and versatile approach for various regression tasks not only constrained to the quantum-chemistry domain.

Next, we compute the condition number of the Cauchy kernel to examine the impact of different sampling strategies on the robustness of KRR predictions. From the theoretical observations presented in Section 5.4 we know that the lower the condition number



(a) Concrete (b) Electrical grid (c) QM8

Figure 7.20.: Condition number of the non-regularized Cauchy kernels are shown for each dataset, training set size and sampling approach. The y-axes of the graphs are on log scale and the error bands represent the confidence interval over five runs of the experiments. The figure shows that selecting training sets using FPS (blue lines) leads to kernels with lower condition number, with respect to the other sampling approaches. The second-best-performing strategy is DA-FPS (red lines).

of the kernel, the higher the robustness of the predictions. The graphs in Fig. 7.20 show that selecting training sets using FPS tends to reduce the condition number of the kernel, which is in line with the experimental results in Section 7.1.4. The only case where FPS does not outperform all other strategies occurs with the Concrete dataset for the largest training set size analyzed. It is notable that for the Concrete dataset, the condition numbers are significantly larger than those on other datasets, regardless of the sampling strategy. This observation highlights the reduced effectiveness of FPS in high condition number regimes. Similar behavior was observed in experiments with the QM9 dataset using the Gaussian kernel, as detailed in Section 7.1.4. Interestingly, the second-best-performing strategy across all cases is DA-FPS, regardless of the dataset or selection approach used.

7.2.7. Section highlights

Our experiments indicate that selecting the training sets using DA-FPS reduces the predictions' MAE for KRR and FNN. In particular, DA-FPS outperforms various baselines, including RDM, facility location, FPS, and k-medoids++ (Fig. 7.11 and Fig. 7.19). Moreover, our experiments suggest that modifying the baselines by initially sampling with FPS consistently enhances their average performance (MAE), regardless of the dataset or model used (Fig. 7.12). Still, DA-FPS remains the most competitive approach (Fig. 7.13). Additionally, we find that FPS may perform similarly to DA-FPS in terms of RMSE, particularly for large training set sizes. Nonetheless, DA-FPS is still the most competitive approach across datasets and training set sizes also when considering RMSE as the evaluation metric (Fig. 7.14, Fig. 7.15). Furthermore, we analyze how the performances of DA-FPS may be affected as we vary the hyperparameter u, defining

the number of samples initially selected with uniform weights, and the hyperparameter k, defining the amount of k-nearest neighbors considered for computing the weights. Our experiments on the ZINC dataset suggest that the impact of u on the prediction performance varies, depending on the training set size and regression model (KRR or FNN). Nonetheless, for larger training set sizes ($> 10\%$ of the available data points), DA-FPS shows to be the most competitive approach regardless of u (Fig. 7.17). Our experiments also indicate that choosing for DA-FPS the hyperparameter k too small may lead to a notable performance decline in the MAE of the predictions, especially for smaller training sets (Fig. 7.18).

8. Conclusion

This work addressed issues related to training data selection with a limited labeling budget. We studied passive and model-agnostic approaches to select training sets that can benefit various learning models and prediction tasks. We had two main objectives. The first was selecting training sets aiming to maximize the robustness of regression models by reducing their maximum prediction error. The second was selecting training sets to improve the average performance of regression models by decreasing their mean absolute prediction error. Our experiments mainly focused on molecular property prediction tasks relevant to drug discovery and material design. In what follows we summarize the contributions of this work highlighting the main findings and results.

8.1. Summary, contributions, and findings

First, we focused on selecting training sets that can improve the robustness of the regression models by reducing the maximum error of their predictions. We derived an upper bound for the maximum expected prediction error of Lipschitz continuous regression models that is linearly dependent on the training set fill distance (Theorem 5.1). In Section 5.3 we saw that the Farthest Point Sampling algorithm (FPS) provides solution with optimal approximation factor in polynomial time to the fill distance minimization problem. We showed that selecting the training sets by aiming to minimize their fill distance using FPS, thereby aiming to minimize our derived bound, significantly decreases the MAXAE of the predictions of Lipschitz continuous regression models, such as KRR and FNN. We compared the FPS approach with other passive and model-agnostic sampling techniques and demonstrated its superiority for low training set budgets in terms of maximum absolute error (MAXAE) reduction (Fig. 7.1, Fig. 7.2, and Fig. 7.10). The error bound in Theorem 5.1 focuses on regression tasks with scalar labels, but we also reported experiments showing that FPS can reduce MAXAE in tasks with multidimensional labels (Fig. 7.10). Our experiments indicated that FPS does not lead to any significant advantage in average prediction quality compared to random sampling or other baseline approaches. We provided further theoretical examinations and empirical investigations to show additional advantages of selecting training sets with FPS for KRR with the Gaussian kernel. Specifically, our findings indicated that employing FPS for selecting training sets enhances the stability of this particular class of models (Fig. 7.3) and of other kernel methods, such as KRR with the Cauchy kernel (Fig. 7.20). An additional important finding of this study relates to the impact of the training set size on the performance of FPS. Our experiments demonstrated that FPS is particularly advantageous in terms of MAXAE reduction, especially in scenarios with limited training

data. However, we found that after a certain number of points have been selected with FPS, there is no significant advantage in continuing the sampling with it compared to other model-agnostic strategies. Specifically, we found that selecting the initial 2% of data points with FPS, followed by random sampling, produces results comparable to using FPS alone in terms of MAXAE reduction (Fig. 7.8).

Next, we focused on selecting training sets that can improve the average prediction performance of the regression models. We derived an upper bound for the expected prediction error of Lipschitz continuous regression models (Theorem 6.1). The bound is linear in the weighted fill distance of the training set, which we defined as the maximum weighted distance between a point in the data space of interest and its closest selected point (Definition 6.1). Furthermore, we introduced "Density-Aware Farthest Point Sampling" (DA-FPS) a novel passive and model-agnostic data selection strategy (Algorithm 6). DA-FPS greedily selects sets from a pool of available data points. We proved that DA-FPS provides suboptimal minimizers for a data-driven estimation of the weighted fill distance (Theorems 6.2 and 6.3), thereby aiming to minimize our derived bound. We reported experimental results showing that selecting training sets using DA-FPS increases the average prediction quality of Lipschitz continuous regression models, such as KRR and FNN. In particular, we compared DA-FPS with other model-agnostic sampling techniques, including FPS, and demonstrated its superiority in terms of mean absolute error (MAE) reduction (Fig. 7.11, Fig. 7.13, and Fig. 7.19). We found that the training set size has an impact on the effectiveness of DA-FPS. Our experiments indicated that DA-FPS is advantageous in terms of MAE reduction with respect to the baselines, particularly for medium-to-large data budget regimes, that is, after a portion of the data has been selected ($> 5\%$ of the available points). Furthermore, we empirically investigated the benefits of combining FPS with other well-established passive and model-agnostic sampling strategies. We showed that augmenting such approaches with FPS during the initial steps of the sampling process consistently led to a decrease in the predictions' MAE (Fig. 7.12).

This work highlighted the importance of training data selection in improving the performance of regression models, particularly under limited data labeling budgets. By analyzing and evaluating FPS and DA-FPS, we theoretically motivated and empirically illustrated that these approaches can effectively reduce prediction errors by optimizing training set selection. FPS is effective in minimizing maximum prediction error with small training set sizes, while DA-FPS enhances average prediction quality, particularly with medium-to-large data labeling budgets.

8.2. Challenges, limitations, and possible future directions

We can think of passive model-agnostic data sampling strategies as existing on a spectrum. On one end are methods that select subsets to uniformly cover the dataset without considering the data distribution. On the other end are approaches that prioritize selecting points that closely reflect the overall data distribution. A key challenge is to design a data selection strategy that balances covering the dataset and capturing

its density distribution to select training sets that can optimize the performance of a regression model, according to a specific evaluation metric, under the constraint of a fixed labeling budget. In what follows, we mention the limitations of this work related to balancing the broad coverage of the dataset by sampling with FPS, and capturing the underlying data density using DA-FPS. Additionally, we mention limitations related to the data density estimations we consider to develop DA-FPS. We also propose future research directions to address these limitations and further extend this work.

Our experiments revealed that using FPS is highly effective in reducing MAXAE in the low training data budget regime. Alternatively, DA-FPS proves particularly beneficial in reducing MAE in medium-to-large data regimes. However, the definitions we provided in Section 2.4.2 of "low", "medium", and "large" data regimes are only qualitative. In our experiments we quantified such data regimes based on the range of effectiveness of FPS and DA-FPS. We determined such ranges using a heuristic approach, where we performed various experiments with different training set sizes. Ideally, given an unlabeled pool of data points, we would like a quantitative approach to define a-priori the training set size ranges for which FPS and DA-FPS can be effective. At this moment of time it is not clear to us how to determine a-priori until when it is beneficial to sample aiming at covering the dataset with FPS, in order to reduce the MAXAE, and when it is beneficial to switch to capturing the data distribution with DA-FPS, to reduce the MAE. Finding an optimal balance between broad coverage of the dataset with FPS and accurately capturing the underlying data density with DA-FPS is still an open challenge. Future work should focus on addressing it. Intuitively, the solution to this problem depends on the distribution and dimension of the points in the available dataset.

In the experiments of Section 7.2 we address balancing data space coverage with FPS and the emphasis on density distribution with DA-FPS by heuristically tuning the DA-FPS hyperparameter "u", which determines the amount of samples to select while considering uniform weights. Recall that DA-FPS initially selects points considering uniform weights, that is, DA-FPS initially coincides with FPS. By sampling as with FPS during the initial phase of the sampling process, DA-FPS reduces the MAXAE and avoids large outlier error values, which helps to reduce the MAE for larger training set sizes. The shift from sampling as with FPS to iteratively updating the weights, taking into account the data density during the sampling process, is determined by the input hyperparameter "u", which we tune following a heuristic approach. Ideally, we would like to have a principled approach to determine a-priori an optimal value of "u" for a given dataset.

DA-FPS also takes in input the hyperparameter "k", determining the number of nearest neighbors considered to compute the weights. In this work, we optimized "k" by performing an a-posteriori analysis where we considered various values for "k" and selected those that provided better results. The lack of principled approaches to determine effective values "k" is a limitation of this work. We think that future work should also focus on the problem of identifying a-priori the rough magnitude of effective values for k. Determining an optimal value is closely tied to the problem of high-dimensional density approximation, which was beyond the scope of this study.

Additional research directions should also investigate alternatives for the data density estimations we use to compute the weights, as we defined them in the definition of weighted fill distance in (6.2). In (6.8) we proposed one possible approach based on k-nearest neighbors, but others more effective may be developed. Further insights and methodologies to investigate this direction may be found in the field of data density approximation in high dimensions.

In Section 5.5 we discussed how the concept of dispersion, commonly used in Quasi-Monte Carlo methods, can provide an alternative to the fill distance. The dispersion quantifies the representativeness of the selected training sets by focusing on volumes instead of distances. Both, FPS and DA-FPS, could be adapted to use dispersion. This would change the principles according to which these two approaches attempt to represent the data space of interest, potentially leading to a performance improvement. Further investigation into this approach is recommended

This work focused on selecting training data for improving the performance of regression models. Our motivation relates to the fact that different training sets can lead to different performances of the same regression models. Another important aspect to consider while developing ML regression models is the evaluation process. Model evaluation is a crucial aspect for assessing the effectiveness of regression models before deploying them into the real word. Typically, regression models are evaluated by measuring their performance on a fixed test set, for which the labels are known. Different test sets can lead to different evaluations of the same model. Therefore, future research could also focus on developing test sets for effective and interpretable evaluations. In particular, it would be valuable to develop test sets that can assess specific aspects of the regression models, such as robustness. Note that test sets need to be labeled. Therefore, selecting small test sets from large pools of unlabeled points to evaluate specific aspects of regression models can be beneficial in scenarios with limited computational resources and high labeling costs.

8.3. Final thoughts

The study of training data selection for regression models is a crucial area that merges theoretical insights with practical applications, particularly when resources for labeling data are limited. This work contributes to this evolving domain by providing clear theoretical guidelines, analyzing sampling strategies like FPS and the novel DA-FPS, and empirically validating their effectiveness across various tasks and data regimes.

Our findings and contributions advance the understanding of how data selection impacts regression performance. FPS and DA-FPS offer valuable tools for optimizing training data selection under data availability constraints.

Despite these advances, the research field is far from exhausted. The challenge of balancing uniform data coverage with density-aware sampling remains open. Our work identifies promising research directions, such as finding principled approaches for DA-FPS' hyperparameters selection, investigating alternative approaches to density approximation, and adapting concepts like dispersion from Quasi-Monte Carlo methods. These research directions could lead to the refinement of the sampling methodologies investigated in this

work or even to the creation of new ones.

In summary, the contributions of this work underscore the potential of data selection strategies in ML regression. By providing theoretical insights, empirical validations, and future research directions, this work aims to inspire further investigations in this field, ultimately leading to cheaper, more reliable and effective ML systems.

Bibliography

[AFK+14] Peshawa Jamal Muhammad Ali, Rezhna Hassan Faraj, Erbil Koya, Peshawa J Muhammad Ali, and Rezhna H Faraj. Data normalization and standardization: a technical report. *Mach Learn Tech Rep*, 1(1):1–6, 2014.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[Bas08] Jayanta Basak. A least square kernel machine with box constraints. In *2008 19th International Conference on Pattern Recognition*, page 1–4. IEEE, December 2008.

[BCD+24] Gantavya Bhatt, Yifang Chen, Arnav Das, Jifan Zhang, Sang Truong, Stephen Mussmann, Yinglun Zhu, Jeff Bilmes, Simon Du, Kevin Jamieson, Jordan Ash, and Robert Nowak. An experimental design framework for label-efficient supervised finetuning of large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6549–6560. Association for Computational Linguistics, 2024.

[BDH+23] Anna Beer, Andrew Draganov, Ellen Hohma, Philipp Jahn, Christian M.M. Frey, and Ira Assent. Connecting the dots – density-connectivity distance unifies dbscan, k-center and spectral clustering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 80–92. ACM, August 2023.

[Bel61] Richard E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, 1961.

[BFPK20] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. Dataset discovery in data lakes. In *ICDE*, pages 709–720, 2020.

[Bil22] Jeff Bilmes. Submodularity in machine learning and artificial intelligence, 2022.

[BJL19] Christopher Baik, H. V. Jagadish, and Yunyao Li. Bridging the semantic gap with sql query logs in natural language interfaces to databases. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 374–385. IEEE, April 2019.

[BKC13] Albert P. Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, May 2013.

[Bos08] Sarah Boslaugh. *Statistics in a nutshell*. O'Reilly, 2008.

[BR09] L. C. Blum and J.-L. Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, 131:8732, 2009.

[BW13] Michael Burch and Daniel Weiskopf. *On the Benefits and Drawbacks of Radial Diagrams*, pages 429–451. Springer New York, June 2013.

[Cac66] Theophilos Cacoullos. Estimation of a multivariate density. *Annals of the Institute of Statistical Mathematics*, 18(1):179–189, December 1966.

[CAC+09] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, March 2009.

[CBC+23] Seamless Communication, Loïc Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, Christopher Klaiber, Pengwei Li, Daniel Licht, Jean Maillard, Alice Rakotoarison, Kaushik Ram Sadagopan, Guillaume Wenzek, Ethan Ye, Bapi Akula, Peng-Jen Chen, Naji El Hachem, Brian Ellis, Gabriel Mejia Gonzalez, Justin Haaheim, Prangthip Hansanti, Russ Howes, Bernie Huang, Min-Jae Hwang, Hirofumi Inaguma, Somya Jain, Elahe Kalbassi, Amanda Kallet, Ilia Kulikov, Janice Lam, Daniel Li, Xutai Ma, Ruslan Mavlyutov, Benjamin Peloquin, Mohamed Ramadan, Abinesh Ramakrishnan, Anna Sun, Kevin Tran, Tuan Tran, Igor Tufanov, Vish Vogeti, Carleigh Wood, Yilin Yang, Bokai Yu, Pierre Andrews, Can Balioglu, Marta R. Costa-jussà, Onur Celebi, Maha Elbayad, Cynthia Gao, Francisco Guzmán, Justine Kao, Ann Lee, Alexandre Mourachko, Juan Pino, Sravya Popuri, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, Paden Tomasello, Changhan Wang, Jeff Wang, and Skyler Wang. Seamlessm4t: Massively multilingual and multimodal machine translation, 2023.

[CF23] Eric Cancès and Gero Friesecke. *Density Functional Theory: Modeling, Mathematical Analysis, Computational Methods, and Applications*. Springer International Publishing, 2023.

[CG24] Paolo Climaco and Jochen Garcke. On minimizing the training set fill distance in machine learning regression. *Journal of Data-centric Machine Learning Research*, 1(14):1–36, 2024.

[CHE+21] Rose K Cersonsky, Benjamin A Helfrecht, Edgar A Engel, Sergei Kliavinek, and Michele Ceriotti. Improving sample and feature selection with principal covariates regression. *Machine Learning: Science and Technology*, 2(3):035038, jul 2021.

[CTS+17] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5), may 2017.

[CvL20a] Anders S. Christensen and Anatole von Lilienfeld. Figshare: Revised md17 dataset (rmd17). `https://figshare.com/articles/dataset/Revised_MD17_dataset_rMD17_/12672038`, July 21 2020. Accessed: April 27, 2024.

[CvL20b] Anders S Christensen and O Anatole von Lilienfeld. On the role of gradients for machine learning of molecular energies and forces. *Machine Learning: Science and Technology*, 1(4):045018, October 2020.

[CWF+12] Rita Chattopadhyay, Zheng Wang, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Batch mode active sampling based on marginal probability distribution matching. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 741–749. ACM, August 2012.

[DBB+21] Volker L. Deringer, Albert P. Bartók, Noam Bernstein, David M. Wilkins, Michele Ceriotti, and Gábor Csányi. Gaussian process regression for materials and molecules. *Chemical Reviews*, 121(16):10073–10141, aug 2021.

[DF85] M.E Dyer and A.M Frieze. A simple heuristic for the p-centre problem. *Operations Research Letters*, 3(6):285–288, February 1985.

[DFR15] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, feb 2015.

[DG17] Dheeru Dua and Casey Graff. Uci machine learning repository, 2017. Accessed: November 1, 2024.

[DJL+23] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

[DM98] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1):46–55, 1998.

[EC91] M. Eugénia Captivo. Fast primal and dual heuristics for the p-median location problem. *European Journal of Operational Research*, 52(1):65–74, May 1991.

[ELPZ94] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 2 - Conference B: Computer Vision & Image Processing. (Cat. No.94CH3440-5)*, pages 93–97 vol.3, 1994.

[Eng14] Berthold-G. Englert. *Semiclassical Theory of Atoms*. Springer, 2014.

[ENV17] Alina Ene, Huy Nguyen, and László A. Végh. Decomposable submodular function minimization: Discrete and continuous. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[FAKA+18] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 289–293, 2018.

[Fel19] Dan Feldman. Core-sets: Updated survey. In *Sampling Techniques for Supervised or Unsupervised Tasks*, pages 23–44. Springer International Publishing, oct 2019.

[FHH+17] Felix A. Faber, Luke Hutchison, Bing Huang, Justin Gilmer, Samuel S. Schoenholz, George E. Dahl, Oriol Vinyals, Steven Kearnes, Patrick F. Riley, and O. Anatole von Lilienfeld. Prediction errors of molecular machine learning models lower than hybrid DFT error. *Journal of Chemical Theory and Computation*, 13(11):5255–5264, oct 2017.

[Fri74] A. M. Frieze. A cost function property for plant location problems. *Mathematical Programming*, 7(1):245–248, dec 1974.

[Fuj05] Satoru Fujishige. *Submodular Functions and Optimization, Volume 58*. Elsevier Science, 2005.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GBP+20] Will Gerrard, Lars A. Bratholm, Martin J. Packer, Adrian J. Mulholland, David R. Glowacki, and Craig P. Butts. Impression – prediction of nmr parameters for 3-dimensional chemical structures using machine learning with near quantum chemical accuracy. *Chemical Science*, 11(2):508–515, 2020.

[GBWD+18] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, January 2018.

[GFPC20] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, dec 2020.

[GSS+22] Johannes Gasteiger, Muhammed Shuaibi, Anuroop Sriram, Stephan Günnemann, Zachary Ward Ulissi, C. Lawrence Zitnick, and Abhishek Das. Gemnet-OC: Developing graph neural networks for large and diverse molecular simulation datasets. *Transactions on Machine Learning Research*, 2022.

[GZ19] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR, 09–15 Jun 2019.

[HBR+15] Katja Hansen, Franziska Biegler, Raghunathan Ramakrishnan, Wiktor Pronobis, O. Anatole von Lilienfeld, Klaus-Robert Müller, and Alexandre Tkatchenko. Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. *The Journal of Physical Chemistry Letters*, 6(12):2326–2331, jun 2015.

[Hla76] Edmund Hlawka. Abschätzung von trigonometrischen summen mittels diophantischer approximationen. *Österreich. Akad. Wiss. Math.-Naturwiss. Kl. S.-B. II*, 185(1-3):43–50, 1976.

[HMB+13] Katja Hansen, Grégoire Montavon, Franziska Biegler, Siamac Fazli, Matthias Rupp, Matthias Scheffler, O. Anatole von Lilienfeld, Alexandre Tkatchenko, and Klaus-Robert Müller. Assessment and validation of machine learning methods for predicting molecular atomization energies. *Journal of Chemical Theory and Computation*, 9(8):3404–3419, July 2013.

[Hoc84] Dorit S. Hochbaum. When are NP-hard location problems easy? *Annals of Operations Research*, 1(3):201–214, oct 1984.

[HP11] Sariel Har-Peled. *Geometric approximation algorithms*. American Mathematical Society, 2011.

[HS85] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, may 1985.

[HSK19] Bruno Miranda Henrique, Vinicius Amorim Sobreiro, and Herbert Kimura. Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, 124:226–251, June 2019.

[HvL16] Bing Huang and O. Anatole von Lilienfeld. Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity. *The Journal of Chemical Physics*, 145(16), October 2016.

[HvL20] Bing Huang and O. Anatole von Lilienfeld. Quantum machine learning using atom-in-molecule-based fragments selected on the fly. *Nature Chemistry*, 12(10):945–951, September 2020.

[HvL21] Bing Huang and O. Anatole von Lilienfeld. Ab initio machine learning in chemical compound space. *Chemical Reviews*, 121(16):10001–10036, August 2021.

[HvLKB23] Bing Huang, O. Anatole von Lilienfeld, Jaron T. Krogel, and Anouar Benali. Toward dmc accuracy across chemical space with scalable δ-qml. *Journal of Chemical Theory and Computation*, 19(6):1711–1721, March 2023.

[Iwa07] Satoru Iwata. Submodular function minimization. *Mathematical Programming*, 112(1):45–64, January 2007.

[JC16] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, April 2016.

[JD75] R. C. St. John and N. R. Draper. D-optimality for regression designs: A review. *Technometrics*, 17(1):15–23, feb 1975.

[JG18] Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[JH17] Xin Jin and Jiawei Han. *K-Medoids Clustering*, pages 697–700. Springer US, Boston, MA, 2017.

[JKW+23] Hoang Anh Just, Feiyang Kang, Tianhao Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. LAVA: Data valuation without pre-specified learning algorithms. In *The Eleventh International Conference on Learning Representations*, 2023.

[JMG23] Mohammad Hossein Jarrahi, Ali Memariani, and Shion Guha. The principles of data-centric ai. *Communications of the ACM*, 66(8):84–92, July 2023.

[JMY90] M.E. Johnson, L.M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148, October 1990.

[KBN+21] Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online, June 2021. Association for Computational Linguistics.

[KBP96] W. Kohn, A. D. Becke, and R. G. Parr. Density functional theory of electronic structure. *The Journal of Physical Chemistry*, 100(31):12974–12980, January 1996.

[KE04] Peter Kirkpatrick and Clare Ellis. Chemical space. *Nature*, 432(7019):823–823, December 2004.

[KF18] Angelos Katharopoulos and Francois Fleuret. Not all samples are created equal: Deep learning with importance sampling. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2525–2534. PMLR, 10–15 Jul 2018.

[KG14] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3:71–104, 2014.

[KHN+20] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100string representation. *Machine Learning: Science and Technology*, 1(4):045024, October 2020.

[KMLE20] Mucahid Kutlu, Tyler McDonnell, Matthew Lease, and Tamer Elsayed. Annotator rationales for labeling tasks in crowdsourcing. *Journal of Artificial Intelligence Research*, 69:143–189, September 2020.

[KNTB09] Yoshinobu Kawahara, Kiyohito Nagano, Koji Tsuda, and Jeff A Bilmes. Submodularity cuts and applications. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[Kon23] Alex Konrad. Cleanlab raises 25 million to help solve ai models' data mess. *Forbes*, 10 Oct 2023.

[KR86] Leonard Kaufman and Peter J. Rousseeuw. *CLUSTERING LARGE DATA SETS*, pages 425–437. Elsevier, 1986.

[Kra10] Andreas Krause. Sfo: A toolbox for submodular function optimization. *Journal of Machine Learning Research*, 11:1141–1144, 2010.

[KRI22] Vishal Kaushal, Ganesh Ramakrishnan, and Rishabh Iyer. Submodlib: A submodular optimization library, 2022.

[KSG08] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 02 2008.

[KSM+21] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR, 18–24 Jul 2021.

[KSRI21] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):8110–8118, May 2021.

[Lan12] G. Landrum. Rdkit:. *Open-source cheminformatics*, 2012.

[LB11] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, page 510–520, USA, 2011. Association for Computational Linguistics.

[LBM+18] Konstantinos Liakos, Patrizia Busato, Dimitrios Moshou, Simon Pearson, and Dionysis Bochtis. Machine learning in agriculture: A review. *Sensors*, 18(8):2674, August 2018.

[LCW+17] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys*, 50(6):1–45, December 2017.

[LCW+22] Ann Lee, Peng-Jen Chen, Changhan Wang, Jiatao Gu, Sravya Popuri, Xutai Ma, Adam Polyak, Yossi Adi, Qing He, Yun Tang, Juan Pino, and Wei-Ning Hsu. Direct speech-to-speech translation with discrete units. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022.

[LK90] Peter J. Rousseeuw Leonard Kaufman. *Partitioning Around Medoids (Program PAM)*, chapter 2, pages 68–125. John Wiley & Sons, Ltd, 1990.

[LL22] A.E. Litvak and G.V. Livshyts. New bounds on the minimal dispersion. *Journal of Complexity*, 72:101648, October 2022.

[LPP14] Jefrey Lijffijt, Panagiotis Papapetrou, and Kai Puolamäki. Size matters: choosing the most informative set of window lengths for mining patterns in event sequences. *Data Mining and Knowledge Discovery*, 29(6):1838–1864, December 2014.

[LPW23] Jourdain Lamperski, Oleg A. Prokopyev, and Luca G. Wrabetz. Min-max-min optimization with smooth and strongly convex objectives. *SIAM Journal on Optimization*, 33(3):2435–2456, September 2023.

[LSB18] Nicholas Lubbers, Justin S. Smith, and Kipton Barros. Hierarchical modeling of molecular energies using a deep neural network. *The Journal of Chemical Physics*, 148(24), March 2018.

[LWL+22] Yi Liu, Limei Wang, Meng Liu, Yuchao Lin, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d molecular graphs. In *International Conference on Learning Representations*, 2022.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the fifth berkeley symposium on mathematical statistics and probability*, pages 281–297. University of California Press, 1967.

[MBK+15] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas Krause. Lazier than lazy greedy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), February 2015.

[MBL20] Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6950–6960. PMLR, 2020.

[MBY+22] Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Douwe Kiela, David Jurado, David Kanter, Rafael Mosquera, Juan Ciro, Lora

Aroyo, Bilge Acun, Sabri Eyuboglu, Amirata Ghorbani, Emmett Goodman, Tariq Kane, Christine R. Kirkpatrick, Tzu-Sheng Kuo, Jonas Mueller, Tristan Thrush, Joaquin Vanschoren, Margaret Warren, Adina Williams, Serena Yeung, Newsha Ardalani, Praveen Paritosh, Ce Zhang, James Zou, Carole-Jean Wu, Cody Coleman, Andrew Ng, Peter Mattson, and Vijay Janapa Reddi. Dataperf: Benchmarks for data-centric ai development, 2022.

[MBY+23] Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William A Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Alicia Parrish, Hannah Rose Kirk, Jessica Quaye, Charvi Rastogi, Douwe Kiela, David Jurado, David Kanter, Rafael Mosquera, Will Cukierski, Juan Ciro, Lora Aroyo, Bilge Acun, Lingjiao Chen, Mehul Smriti Raje, Max Bartolo, Sabri Eyuboglu, Amirata Ghorbani, Emmett Daniel Goodman, Addison Howard, Oana Inel, Tariq Kane, Christine Kirkpatrick, D. Sculley, Tzu-Sheng Kuo, Jonas Mueller, Tristan Thrush, Joaquin Vanschoren, Margaret Warren, Adina Williams, Serena Yeung, Newsha Ardalani, Praveen Paritosh, Ce Zhang, James Y. Zou, Carole-Jean Wu, Cody Coleman, Andrew Ng, Peter Mattson, and Vijay Janapa Reddi. Dataperf: Benchmarks for data-centric AI development. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[MCL20] Baharan Mirzasoleiman, Kaidi Cao, and Jure Leskovec. Coresets for robust training of neural networks against noisy labels. *Advances in Neural Information Processing Systems*, 33, 2020.

[MD09] Michael W. Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, January 2009.

[MDFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[MFP+24] Nestor Maslej, Loredana Fattorini, Raymond Perrault, Vanessa Parli, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, and Jack Clark. Artificial intelligence index report 2024, 2024.

[Min78] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In J. Stoer, editor, *Optimization Techniques*, pages 234–243, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

[MJH+11] Shie Mannor, Xin Jin, Jiawei Han, Xin Jin, Jiawei Han, Xin Jin, Jiawei Han, and Xinhua Zhang. K-medoids clustering. In *Encyclopedia of Machine Learning*, pages 564–565. Springer US, 2011.

[MKSK16] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. *Journal of Machine Learning Research*, 17(235):1–44, 2016.

[Mou12] Jonathan E. Moussa. Comment on "fast and accurate modeling of molecular atomization energies with machine learning". *Physical Review Letters*, 109(5):059801, August 2012.

[MRG+13] Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9):095003, sep 2013.

[MTKT18] Hirotomo Moriwaki, Yu-Shi Tian, Norihito Kawashita, and Tatsuya Takagi. Mordred: a molecular descriptor calculator. *Journal of Cheminformatics*, 10(1), feb 2018.

[MWW+17] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, May 2017.

[NH02] R.T. Ng and Jiawei Han. Clarans: a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, September 2002.

[Nov19] Andrei V. Novikov. Pyclustering: Data mining library. *Journal of Open Source Software*, 4(36):1230, 2019.

[NW81] G.L. Nemhauser and L.A. Wolsey. *Maximizing Submodular Set Functions: Formulations and Analysis of Algorithms*, pages 279–301. Elsevier, 1981.

[NWF78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, December 1978.

[PDM22] Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. Adaptive second order coresets for data-efficient machine learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17848–17869. PMLR, 17–23 Jul 2022.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner,

Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library.* Curran Associates Inc., Red Hook, NY, USA, 2019.

[PJ09] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, March 2009.

[PMS+20] Gabriel A. Pinheiro, Johnatan Mucelini, Marinalva D. Soares, Ronaldo C. Prati, Juarez L. F. Da Silva, and Marcos G. Quiles. Machine learning prediction of nine molecular properties based on the SMILES representation of the QM9 quantum-chemistry dataset. *The Journal of Physical Chemistry A*, 124(47):9854–9866, nov 2020.

[PSGA+24] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. Probabilistic weather forecasting with machine learning. *Nature*, December 2024.

[PSTM18] Wiktor Pronobis, Kristof T. Schütt, Alexandre Tkatchenko, and Klaus-Robert Müller. Capturing intensive and extensive DFT/TDDFT molecular properties with machine learning. *The European Physical Journal B*, 91(8), aug 2018.

[PTM18] Wiktor Pronobis, Alexandre Tkatchenko, and Klaus-Robert Müller. Many-body descriptors for predicting molecular properties with machine learning: Analysis of pairwise and three-body interactions in molecules. *Journal of Chemical Theory and Computation*, 14(6):2991–3003, May 2018.

[PYM+23] Neha Prakriya, Yu Yang, Baharan Mirzasoleiman, Cho-Jui Hsieh, and Jason Cong. Nessa: Near-storage data selection for accelerated machine learning training. *ACM Workshop on Hot Topics in Storage and File Systems (HotStorage)*, 2023.

[PZ21] Neoklis Polyzotis and Matei Zaharia. What can data-centric ai learn from data and ml engineering?, 2021.

[RBE+19] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: rapid training data creation with weak supervision. *The VLDB Journal*, 29(2–3):709–730, July 2019.

[RDRvL14] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

[REW+19] Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life*

Sciences. O'Reilly Media, 2019. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

[RHTvL15] Raghunathan Ramakrishnan, Mia Hartmann, Enrico Tapavicza, and O. Anatole von Lilienfeld. Electronic spectra from TDDFT and machine learning in chemical space. *The Journal of Chemical Physics*, 143(8), aug 2015.

[RLJ20] Jonathan G. Richens, Ciarán M. Lee, and Saurabh Johri. Improving the accuracy of medical diagnosis with causal machine learning. *Nature Communications*, 11(1), August 2020.

[RNI10] Miloš Radovanovic;, Alexandros Nanopoulos, and Mirjana Ivanovic;. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(86):2487–2531, 2010.

[RRT94] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, apr 1994.

[RT96] G. Rote and R.F. Tichy. Quasi-monte-carlo methods and the dispersion of point sequences. *Mathematical and Computer Modelling*, 23(8–9):9–23, April 1996.

[RTMvL12] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108(5):058301, January 2012.

[RvDBR12] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, nov 2012.

[RvL17] Raghunathan Ramakrishnan and O. Anatole von Lilienfeld. Machine learning, quantum chemistry, and chemical space. In *Reviews in Computational Chemistry*, pages 225–256. John Wiley & Sons, Inc., apr 2017.

[RXC+21] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys*, 54(9):1–40, October 2021.

[SBG+20] Christopher Sutton, Mario Boley, Luca M. Ghiringhelli, Matthias Rupp, Jilles Vreeken, and Matthias Scheffler. Identifying domains of applicability of machine learning models for materials science. *Nature Communications*, 11(1):4428, sep 2020.

[SBN20] Jacob Schreiber, Jeffrey Bilmes, and William Stafford Noble. apricot: Submodular selection for data summarization in python. *Journal of Machine Learning Research*, 21(161):1–6, 2020.

[SBS18] Patrick Schober, Christa Boer, and Lothar A. Schwarte. Correlation coefficients: Appropriate use and interpretation. *Anesthesia & Analgesia*, 126(5):1763–1768, may 2018.

[sci21] Scikit-learn-extra: An experimental scikit-learn-compatible package. `https://github.com/scikit-learn-contrib/scikit-learn-extra`, 2021. Version 0.3dev.

[SDD+18] Shazia Sadiq, Tamraparni Dasu, Xin Luna Dong, Juliana Freire, Ihab F. Ilyas, Sebastian Link, Miller J. Miller, Felix Naumann, Xiaofang Zhou, and Divesh Srivastava. Data quality: The role of empiricism. *ACM SIGMOD Record*, 46(4):35–43, February 2018.

[SDF+24] Nore Stolte, János Daru, Harald Forbert, Dominik Marx, and Jörg Behler. Random sampling versus active learning algorithms for machine learning potentials of quantum liquid water, 2024.

[SDK15] Dravyansh Sharma, Amit Deshpande, and Ashish Kapoor. On greedy maximization of entropy. In *Proceedings of the 32nd International Conference on on Machine Learning - Volume 37*, ICML'15, page 1330–1338, Lille, France, 2015. JMLR.org.

[Set12] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[SI15] Teague Sterling and John J. Irwin. Zinc 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, November 2015.

[SIS24] Nabeel Seedat, Fergus Imrie, and Mihaela van der Schaar. Navigating data-centric artificial intelligence with dc-check: Advances, challenges, and opportunities. *IEEE Transactions on Artificial Intelligence*, 5(6):2589–2603, June 2024.

[SJ19] Ayodeji Olalekan Salau and Shruti Jain. Feature extraction: A survey of the types, techniques, applications. In *2019 International Conference on Signal Processing and Communication (ICSC)*, pages 158–164. IEEE, March 2019.

[SKSF+17] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus,

S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[SL22] Erich Schubert and Lars Lenssen. Fast k-medoids clustering in rust and python. *Journal of Open Source Software*, 7(75):4183, July 2022.

[SR19] Erich Schubert and Peter J. Rousseeuw. *Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms*, pages 171–187. Springer International Publishing, 2019.

[SR21] Erich Schubert and Peter J. Rousseeuw. Fast and eager k-medoids clustering: $o(k)$ runtime improvement of the pam, clara, and clarans algorithms. *Information Systems*, 101:101804, November 2021.

[SS18] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[STR+19] Annika Stuke, Milica Todorović, Matthias Rupp, Christian Kunkel, Kunal Ghosh, Lauri Himanen, and Patrick Rinke. Chemical diversity in molecular orbital energy predictions with kernel ridge regression. *The Journal of Chemical Physics*, 150(20):204121, may 2019.

[SUG21] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9377–9388. PMLR, 18–24 Jul 2021.

[SV18] Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3839–3848, Red Hook, NY, USA, 2018. Curran Associates Inc.

[SZ19] Erich Schubert and Arthur Zimek. Elki: A large open-source library for data analysis - elki release 0.7.5 "heidelberg", 2019.

[TMO23] Murad Tukan, Alaa Maalouf, and Margarita Osadchy. Dataset distillation meets provable subset selection. July 2023.

[UCS+21] Oliver T. Unke, Stefan Chmiela, Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, Alexandre Tkatchenko, and Klaus-Robert Müller. Machine learning force fields. *Chemical Reviews*, 121(16):10142–10186, mar 2021.

[Van21] Jonathan Vanian. This hot startup is now valued at 1 billion for its a.i. skills. *Fortune*, 9 August 2021.

[VCC+19] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, and Shanrong Zhao. Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, 18(6):463–477, April 2019.

[VdLPB03] Mark Van der Laan, Katherine Pollard, and Jennifer Bryan. A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73(8):575–584, August 2003.

[vdWCV11] Stéfan van der Walt, S Chris Colbert, and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & amp; Engineering*, 13(2):22–30, March 2011.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[VJ22] Akhil Vakayil and V. Roshan Joseph. Data twinning. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 15(5):598–610, February 2022.

[VKL19] Tom J. Viering, Jesse H. Krijthe, and Marco Loog. Nuclear discrepancy for single-shot batch active learning. *Machine Learning*, 108(8–9):1561–1599, June 2019.

[vLMT20] O. Anatole von Lilienfeld, Klaus-Robert Müller, and Alexandre Tkatchenko. Exploring chemical compound space with quantum-based machine learning. *Nature Reviews Chemistry*, 4(7):347–358, June 2020.

[VSH21] Gaurav Vishwakarma, Aditya Sonpal, and Johannes Hachmann. Metrics for benchmarking and uncertainty quantification: Quality, applicability, and best practices for machine learning in chemistry. *Trends in Chemistry*, 3(2):146–156, February 2021.

[Was21] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[WBG21] George Wynne, François-Xavier Briol, and Mark Girolami. Convergence guarantees for gaussian process means with misspecified likelihoods and smoothness. *J. Mach. Learn. Res.*, 22(1), January 2021.

[Wei88] David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, 28(1):31–36, feb 1988.

[Wen04] Holger Wendland. *Scattered Data Approximation*. Cambridge University Press, dec 2004.

[WIB15] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1954–1963, Lille, France, 07–09 Jul 2015. PMLR.

[WKV09] Qing Wang, Sanjeev R. Kulkarni, and Sergio Verdu. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405, May 2009.

[WLH19] Dongrui Wu, Chin-Teng Lin, and Jian Huang. Active learning for regression using greedy sampling. *Information Sciences*, 474:90–105, feb 2019.

[WLKB14] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Unsupervised submodular subset selection for speech data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2014.

[WRF+18] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[WY15] Zheng Wang and Jieping Ye. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data*, 9(3):1–23, feb 2015.

[WZS+14] Endong Wang, Qing Zhang, Bo Shen, Guangyong Zhang, Xiaowei Lu, Qing Wu, and Yajuan Wang. *Intel Math Kernel Library*, page 167–188. Springer International Publishing, 2014.

[XSML23] Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[Yap10] Chun Wei Yap. Padel-descriptor: An open source software to calculate molecular descriptors and fingerprints. *Journal of Computational Chemistry*, 32(7):1466–1474, December 2010.

[YBT06] Kai Yu, Jinbo Bi, and Volker Tresp. Active learning via transductive experimental design. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*. ACM Press, 2006.

[Yeh98] I-Cheng Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28:1797–1808, 1998.

[YK10] Hwanjo Yu and Sungchul Kim. Passive sampling for regression. In *2010 IEEE International Conference on Data Mining*. IEEE, dec 2010.

[ZBL+25] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, Zhimeng Jiang, Shaochen Zhong, and Xia Hu. Data-centric artificial intelligence: A survey. *ACM Computing Surveys*, January 2025.

[ZC05] Qiaoping Zhang and Isabelle Couloigner. A new and efficient k-medoid algorithm for spatial clustering. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *Computational Science and Its Applications – ICCSA 2005*, pages 181–189, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[ZHSK22] Viktor Zaverkin, David Holzmüller, Ingo Steinwart, and Johannes Kästner. Exploring chemical and conformational spaces by batch mode deep active learning. *Digital Discovery*, 2022.

[ZLB+23] Xin Zheng, Yixin Liu, Zhifeng Bao, Meng Fang, Xia Hu, Alan Wee-Chung Liew, and Shirui Pan. Towards data-centric graph machine learning: Review and outlook, 2023.

A. Appendix

In this appendix we show that the kNN data-driven estimates $\hat{p}^k_{\mathcal{X}_\mathcal{D}}$ and $\hat{p}^k_{\mathcal{X}_\mathcal{L}}$, defined in (6.8) and derived from the finite sets $\mathcal{D}_\mathcal{X}, \mathcal{L}_\mathcal{X} \subset \mathbb{R}^d$, are asymptotically unbiased estimations of some densities $p_{\mathcal{X}_\mathcal{D}}$ and $p_{\mathcal{X}_\mathcal{L}}$, respectively, for any $2 \leq k < n$. We assume that $p_{\mathcal{X}_\mathcal{D}}$ and $p_{\mathcal{X}_\mathcal{L}}$ are uniformly continuous and that $\mathcal{D}_\mathcal{X}$ and $\mathcal{L}_\mathcal{X}$ consist of random samples drawn from $p_{\mathcal{X}_\mathcal{D}}$ and $p_{\mathcal{X}_\mathcal{L}}$, respectively.

To show this we use results from [Cac66], which we recapture in the following theorem in a formulation that is more suitable for our purposes.

Theorem A.1 *Let us consider a d-dimensional Euclidean space $\mathcal{X}$ and $K : \mathcal{X} \to \mathbb{R}$, Borel function on $\mathcal{X}$, such that*

$$\sup_{\boldsymbol{x}\in\mathcal{X}} |K(\boldsymbol{x})| < \infty, \int_\mathcal{X} |K(\boldsymbol{x})| d\boldsymbol{x} < \infty, \int_\mathcal{X} K(\boldsymbol{x}) d\boldsymbol{x} = 1 \quad \textit{and} \quad \lim_{\|\boldsymbol{x}\|_2\to\infty} \|\boldsymbol{x}\|_2^d |K(\boldsymbol{x})| = 0. \tag{A.1}$$

Additionally, let us consider a sequence of positive scalar numbers $\{r_i\}_{i=1}^\infty \subset \mathbb{R}$ such that $\lim_{i\to\infty} r_i = 0$. Let us also consider $\{\boldsymbol{x}_i\}_{i=1}^m \subset \mathcal{X}$, $m \in \mathbb{N}^+$, set of m independent realization of a random variable $\boldsymbol{X}$ with uniformly continuous distribution density p. Then, for any $\boldsymbol{x} \in \mathcal{X}$ in the non-zero support of p we have that

$$p_m(\boldsymbol{x}) = \frac{1}{m(r_m)^d} \sum_{i=1}^m K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{r_m}\right) \tag{A.2}$$

is an asymptotically unbiased estimator of $p(\boldsymbol{x})$, i.e., $\lim_{m\to\infty} \mathbb{E}_p[p_m(\boldsymbol{x})] = p(\boldsymbol{x})$.

Proof. The theorem recaptures results from Theorem 3.1 of [Cac66], which is proved using results from Theorem 2.1 and Lemma 2.1 of the same paper. The sketch of the proof is as follows: First, fix $\boldsymbol{x} \in \mathcal{X}$, and let

$$\varepsilon_m(\boldsymbol{x}) := \frac{1}{r_m} K\left(\frac{\boldsymbol{x} - \boldsymbol{X}}{r_m}\right). \tag{A.3}$$

Then, under the mentioned assumptions on K and p, given a positive integer l, it is possible to show that

$$\lim_{m\to\infty} r_m^{d(l-1)} \mathbb{E}_p[\varepsilon_m^l(\boldsymbol{x})] = p(\boldsymbol{x}) \int K^l(\boldsymbol{x}) d\boldsymbol{x}. \tag{A.4}$$

Next, the theorem follows from noting that

$$\mathbb{E}_p[p_m(\boldsymbol{x})] = \mathbb{E}_p[\varepsilon_m(\boldsymbol{x})]. \tag{A.5}$$

□

Next we provide a corollary of the above theorem showing that the data-driven kNN density estimations $\hat{p}^k_{\mathcal{X}_\mathcal{D}}$ and $\hat{p}^k_{\mathcal{X}_\mathcal{L}}$ are asymptotically unbiased estimators of $p_{\mathcal{X}_\mathcal{D}}$ and $p_{\mathcal{X}_\mathcal{L}}$, respectively, for any $2 \leq k < n$.

Corollary A.1 *Let us consider $\mathcal{X} \subset \mathbb{R}^d$ and the function $K : \mathbb{R}^d \to \mathbb{R}^+$ defined in (6.9). Let us also consider $\mathcal{D}_n, \mathcal{L}_b \subset \mathcal{X}$, $n, b \in \mathbb{N}^+$ such that $\mathcal{D}_n := \{\boldsymbol{x}_i\}_{i=1}^n$ and $\mathcal{L}_b := \{\bar{\boldsymbol{x}}_j\}_{j=1}^b$ are independent realizations of random variables $\boldsymbol{X}_\mathcal{D}$ and $\boldsymbol{X}_\mathcal{L}$ with uniformly continuous densities $p_{\mathcal{X}_\mathcal{D}}$ and $p_{\mathcal{X}_\mathcal{L}}$, respectively. Next, consider the points $\boldsymbol{x}, \bar{\boldsymbol{x}} \in \mathbb{R}^d$ in the non-zero support of $p_{\mathcal{X}_\mathcal{D}}$ and $p_{\mathcal{X}_\mathcal{L}}$, respectively, and the kNN data-driven density estimations $\hat{p}^k_{\mathcal{X}_{\mathcal{D}_n}}(\boldsymbol{x})$ and $\hat{p}^k_{\mathcal{X}_{\mathcal{L}_b}}(\bar{\boldsymbol{x}})$ defined as follows*

$$\hat{p}^k_{\mathcal{X}_{\mathcal{D}_n}}(\boldsymbol{x}) := \frac{\sum\limits_{\boldsymbol{x}_i \in \mathcal{D}_n} K\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{r^k_{b,n}(\boldsymbol{x})}\right)}{n\left(r^k_{b,n}(\boldsymbol{x})\right)^d} \quad \text{and} \quad \hat{p}^k_{\mathcal{X}_{\mathcal{L}_b}}(\bar{\boldsymbol{x}}) := \frac{\sum\limits_{\bar{\boldsymbol{x}}_j \in \mathcal{L}_b} K\left(\frac{\bar{\boldsymbol{x}}-\bar{\boldsymbol{x}}_j}{r^k_{b,n}(\bar{\boldsymbol{x}})}\right)}{b\left(r^k_{b,n}(\bar{\boldsymbol{x}})\right)^d}, \tag{A.6}$$

where, for each $\tilde{\boldsymbol{x}} \in \mathbb{R}^d$, $r^k_{b,n}(\tilde{\boldsymbol{x}}) := \min\left\{\min\limits_{\bar{\boldsymbol{x}}_j \in \mathcal{L}_b} \|\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}}_j\|_2 + \frac{\epsilon_\mathcal{X}}{b}, \rho_{k,n}(\tilde{\boldsymbol{x}})\right\}$, with the scalar $\epsilon_\mathcal{X} > 0$ arbitrary small and $\rho_{k,n}(\tilde{\boldsymbol{x}})$ the distance between $\tilde{\boldsymbol{x}}$ and its k-nearest neighbor in $\mathcal{D}_n$. Then we have that, for any $2 \leq k < n$, $\hat{p}^k_{\mathcal{X}_{\mathcal{D}_n}}(\boldsymbol{x})$ and $\hat{p}^k_{\mathcal{X}_{\mathcal{L}_b}}(\bar{\boldsymbol{x}})$ are asymptotically unbiased estimators of $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$ and $p_{\mathcal{X}_\mathcal{L}}(\bar{\boldsymbol{x}})$, respectively, i.e.,

$$\lim_{n\to\infty} \mathbb{E}_{p_{\mathcal{X}_\mathcal{D}}}[\hat{p}^k_{\mathcal{X}_{\mathcal{D}_n}}(\boldsymbol{x})] = p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x}) \quad \text{and} \quad \lim_{b\to\infty} \mathbb{E}_{p_{\mathcal{X}_\mathcal{L}}}[\hat{p}^k_{\mathcal{X}_{\mathcal{L}_b}}(\bar{\boldsymbol{x}})] = p_{\mathcal{X}_\mathcal{L}}(\bar{\boldsymbol{x}}). \tag{A.7}$$

Proof. First note that the function $K : \mathbb{R}^d \to \mathbb{R}^+$ defined in (6.9) satisfies all the requirements in (A.1). In particular, we have that

$$\sup_{\boldsymbol{x}\in\mathbb{R}^d} |K(\boldsymbol{x})| = \frac{1}{V_d} \quad \text{and} \quad \int_{\mathbb{R}^d} K(\boldsymbol{x})d\boldsymbol{x} = \frac{1}{V_d}\int_{B^d(0,1)} 1 d\boldsymbol{x} = 1, \tag{A.8}$$

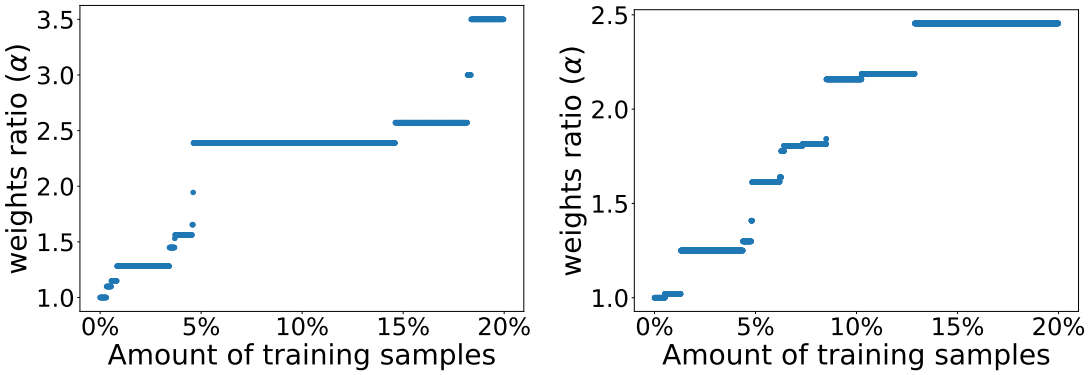
where V_d is the volume of the d-dimensional unit ball, which we write as $B^d(0,1) := \{\boldsymbol{x} \in \mathbb{R}^d \text{ such that } \|\boldsymbol{x}\|_2 \leq 1\}$. Moreover, we have that $|K(\boldsymbol{x})| = 0$ for $\|\boldsymbol{x}\|_2 \geq 1$, thus, $\lim\limits_{\|\boldsymbol{x}\|_2\to\infty} \|\boldsymbol{x}\|_2^d |K(\boldsymbol{x})| = 0$. The only thing left to show to apply Theorem A.1 to $\hat{p}^k_{\mathcal{X}_{\mathcal{D}_n}}(\boldsymbol{x})$ and $\hat{p}^k_{\mathcal{X}_{\mathcal{L}_b}}(\bar{\boldsymbol{x}})$ is that $\lim\limits_{b\to\infty} r^k_{b,n}(\bar{\boldsymbol{x}}) = 0$ and $\lim\limits_{n\to\infty} r^k_{b,n}(\boldsymbol{x}) = 0$.

To show that $\lim\limits_{b\to\infty} r^k_{b,n}(\bar{\boldsymbol{x}}) = 0$ it is sufficient to observe that $\lim\limits_{b\to\infty} \min\limits_{\bar{\boldsymbol{x}}_j \in \mathcal{L}_b} \|\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}}_j\|_2 = 0$ due to the fact that as the number of samples in $\mathcal{L}_b$ increases, the distance between $\bar{\boldsymbol{x}}$ and its nearest neighbor in $\mathcal{L}_b$ decreases, tending to zero. Moreover, $\lim\limits_{b\to\infty} \frac{\epsilon_\mathcal{X}}{b} = 0$. Similarly, we have that $\lim\limits_{n\to\infty} r^k_{b,n}(\boldsymbol{x}) = 0$ from the fact that $\lim\limits_{n\to\infty} \rho_{k,n}(\boldsymbol{x}) = 0$ which follows from the observation that as the number of samples in $\mathcal{D}_n$ increases, the distance between $\boldsymbol{x}$ and its k-nearest neighbor decreases, tending to zero. Thus, we can apply Theorem A.1 to $\hat{p}^k_{\mathcal{X}_{\mathcal{D}_n}}(\boldsymbol{x})$ and $\hat{p}^k_{\mathcal{X}_{\mathcal{L}_b}}(\bar{\boldsymbol{x}})$ showing that they are asymptotically unbiased estimators of $p_{\mathcal{X}_\mathcal{D}}(\boldsymbol{x})$ and $p_{\mathcal{X}_\mathcal{L}}(\bar{\boldsymbol{x}})$, respectively, for any $2 \leq k < n$.

□

B. Appendix

B.1. Investigation weights ratio



(a) QM7 (b) QM8

Figure B.1.: Computation of parameter $\alpha := \max_{\substack{i,j=1,\dots,b+1 \\ i<j}} \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)}$ representing the ratio of the weights computed by the DA-FPS algorithm implemented on the QM7 (a) and QM8 (b) datasets. For each dataset we initialized DA-FPS with $k = 100$ and $u = 0\%$

Figure B.1 shows the parameter $\alpha := \max_{\substack{i,j=1,\dots,b+1 \\ i<j}} \frac{\omega^k_{\mathcal{L}_{j-1}}(\boldsymbol{x}_j)}{\omega^k_{\mathcal{L}_{i-1}}(\boldsymbol{x}_i)}$, which is a component of the DA-FPS approximation factor described in Theorem 6.3. We calculated α using the weights generated by applying DA-FPS to the QM7 and QM8 datasets. The value of α depends on the size of the selected subset and increases as the subset grows.

The relevant component of the approximation factor in Theorem 6.3 involving α is $\sigma = \min\{3, 1 + \alpha\}$ This implies that when $\alpha < 2$, the approximation bound is better than the worst-case scenario of 3. The graphs in the figure illustrate that, in the early stages of the sampling process, the approximation factor remains below 3. Specifically, this holds up to 5% of the selected set size for QM7 and up to 10% for QM8.

B.2. Datasets for additional experiments

This section provides a more detailed description of the additional datasets unrelated to quantum chemistry used for experiments in Section 7.2.6, including information on the

preprocessing procedures.

The Concrete Compressive Strength dataset [Yeh98] downloaded from the UCI Machine Learning Repository [DG17] contains 1030 data points and is used for regression tasks. It includes eight features: the amounts of cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate, as well as the age of the concrete in days. We remove 34 data points having identical descriptors as at least one other point in the dataset, obtaining a reduced dataset of 996 data points. Furthermore, we normalize the features to scale them independently in the interval (0, 1). The target variable is the compressive strength of the concrete, measured in megapascals (MPa). This dataset is used to test machine learning models to predict material properties.

In the experiments described in Section 7.2.6, we use the Twinning algorithm implementation from [VJ22], which selects subsets based on an integer r, the inverse of the partitioning ratio. Since the algorithm strictly partitions the dataset according to this ratio, we remove 6 points from the Concrete dataset, resulting in a reduced dataset with 990 points. The points were selected randomly. This adjustment ensures that the subset size determined by the Twinning algorithm matches the percentages used to select the subsets, eliminating any discrepancies. Similarly, for the experiments in Section 7.2.6, we remove 66 points from the QM8 dataset, creating a reduced dataset of 21700 points. QM8 is preprocessed with the same procedure used Section 7.2.3

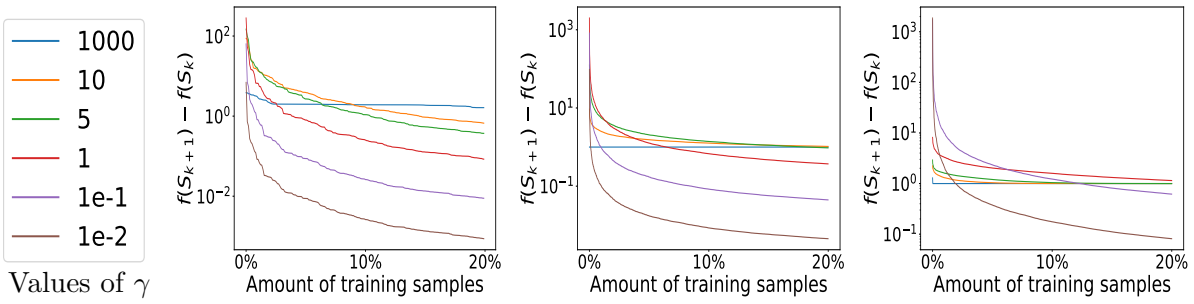
The Electrical Grid Stability Simulated dataset from the UCI Machine Learning Repository [DG17] contains 10000 data points and is designed for both classification and regression tasks. Each data point in this dataset is represented by 12 features that describe characteristics of a simulated power grid. We normalize the features to scale them independently in the interval (0, 1). For regression tasks, the target variable is the stability margin, which quantifies how stable the power grid is.

B.3. Hyperparameters for additional experiments

In this section we follow along [BCD+24] and describe the fine-tuning process for optimizing the facility location selection with the Gaussian similarity function. In addition, we report the hyperparameters considered for DA-FPS used for the experiments in Section 7.2.6.

For the facility location method using the Gaussian similarity function defined in (3.13), the fine-tuning process involves selecting an appropriate kernel width, denoted as γ, to prevent the function's gains from saturating when new data points are added to the training set. That is, given $f(S_k)$ value of the facility location function evaluated on the set S_k, we aim to choose γ to maximize the gains $f(S_{k+1}) - f(S_k)$. The optimization procedure consists of computing the gains for various values of γ and analyzing their behavior. The optimal value γ is chosen to maximize gains for larger training sets while maintaining the ability to capture interactions between data points.

Figure B.2 illustrates the gains obtained from adding new elements to the selected sets for the Concrete, Electrical grid, and QM8 datasets. We initialize the greedy selection process with the same data point, independently of the value of γ. Low values of γ result



(a) Concrete (b) Electrical grid (c) QM8

Figure B.2.: Gains from adding new elements to the selected sets for the QM8, Concrete, and Electrical Grid datasets using the facility location method with the Gaussian similarity function. Gains are shown for various values of the kernel width, $\gamma = 1000, 10, 5, 1, 0.1$ and 0.01.

in diminishing gains as the training set size grows, while excessively high values, such as $\gamma = 1000$, cause the kernel to approximate a diagonal matrix, failing to capture data point interactions. Based on the experimental results, we set γ to 1 for QM8, 10 for the Concrete dataset, and 10 for the Electrical grid dataset.

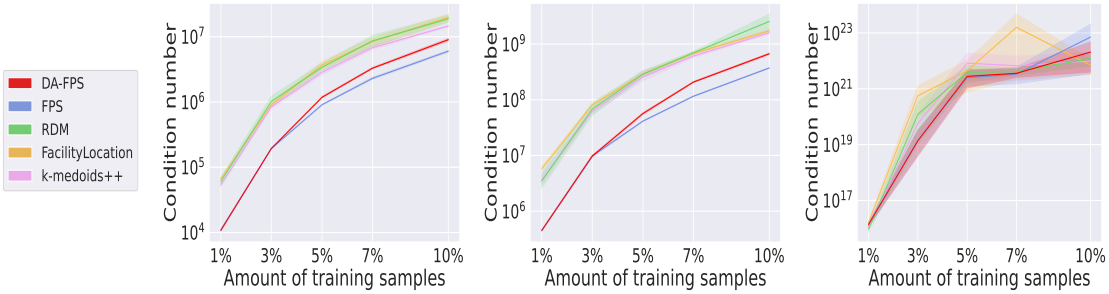
For the DA-FPS we follow the same heuristic approach used for experiments in Section 7.2 and set $u = 3\%$, 1% and 3% and $k = 100$, 300 and 300 for the QM8, Concrete dataset and electricity dataset, respectively.

B.4. Cauchy Kernel

In this section we define the Cauchy kernel used in the additional experiments in Section 7.2.6 and describe the optimization process implemented to fine-tune the kernel hyperparameter and the regularization hyperparameter for the kernel ridge regression weights optimization problem. Given data points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^d$, we follow along [Bas08] and define the Cauchy kernel as follows:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{1 + \left(\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}{\gamma_c}\right)^2} \tag{B.1}$$

The hyperparameter γ_c of the Cauchy kernel and λ, the regularization parameter of kernel ridge regression problem in (4.5), are fine-tuned using the following process: first, we conduct a cross-validation grid search to identify the best hyperparameters for each training set size used in the experiments. The training subsets are obtained through random sampling. Then, we calculate the average of the best hyperparameter pairs across all training set sizes, which is subsequently used to build the final model. The grid search explores 6 points for each hyperparameter, ranging from 10^{-6} to 10^{-2}. It is important to note that in our experiments we do not use an optimal set of hyperparameters for each selection strategy and training set size. This choice ensures that we focus on analyzing the



(a) QM7 (b) QM8 (c) QM9

Figure B.3.: Condition number of the non-regularized Gaussian kernel obtained from the QM datasets selecting training sets of various sizes and according to different sampling strategies. The reported experiments consider the experimental setup described in Section 7.1.2. The y-axes of the graphs are on log scale and the error bands represent the confidence interval over five independent runs of the experiments. The figure extends the results illustrated in the top row of Fig. 7.3 including DA-FPS initialized with $k = 100$ and $u = 3\%$.

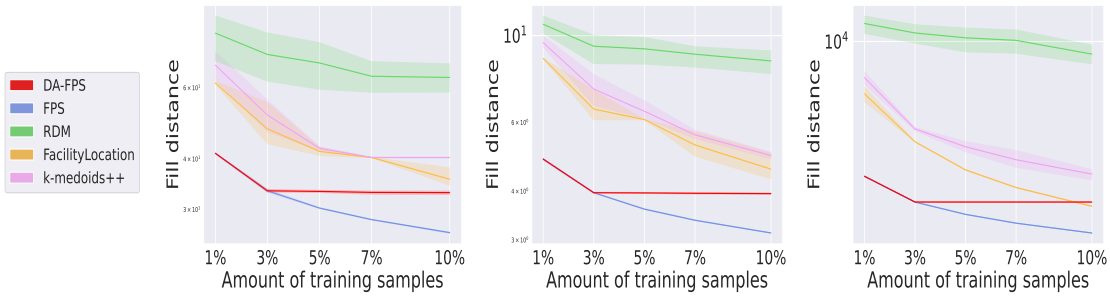
qualitative behavior of a fixed model, where the only variable influencing the prediction quality is the selection of the training set.

B.5. DA-FPS on FPS setting

In this section we compare the performance DA-FPS with FPS and other baseline methods by examining the condition number of the Gaussian kernel and the fill distance with the experimental setup described in Section 7.1.2.

Fig. B.3 expands on the graphs in the top row of Fig. 7.3 by including results obtained using DA-FPS. Specifically, it illustrates the condition number of the non-regularized Gaussian kernels. The kernels were constructed under the same experimental setup described in Section 7.1.2, using the same datasets, data descriptors, training dataset sizes and baseline sampling approaches. DA-FPS was initialized with $k = 100$ and u=3%, regardless of the dataset. This means that DA-FPS and FPS are identical until 3% of the data is sampled. The figure demonstrates that, as expected, DA-FPS leads to worse performance than FPS. However, it still outperforms other baseline approaches.

Fig. B.4 extends the results shown in Fig. 7.5a by illustrating the fill distance of the training sets obtained using DA-FPS compared to that of other baseline sampling strategies. Also in this case we consider the same experimental setup as in Section 7.1.2. Interestingly, with DA-FPS, the fill distance decreases until 3% of the data is selected, during which DA-FPS is equivalent to FPS, since we set $u = 3\%$. Once DA-FPS starts the weights update process, the fill distance for the set it selects remains constant. This indicates that after the initial 3% of elements are selected using FPS, DA-FPS sampling shifts its focus from space coverage to better representing the underlying data distribution.



(a) QM7 (b) QM8 (c) QM9

Figure B.4.: Fill distance of training sets obtained using DA-FPS and other baseline sampling strategies on the QM datasets. The experiments in the figures are performed with the same experimental setup described in Section 7.1.2. The figure extends the results illustrated in Fig. 7.5a including DA-FPS initialized with $k = 100$ and $u = 3\%$. The y-axes of the graphs are on log scale and the error bands represent the confidence interval over five independent runs of the experiments.