Active Learning Regression by Mitigating Domain Drift

John Isak Texas Falk

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master of Research
of
University College London.

Department of Computer Science
University College London

August 15, 2019

I, John Isak Texas Falk, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Active learning enables an algorithm to choose a subset training set of a dataset to label while retaining most of the performance as if training on the full dataset. When labelling data is costly but finding instances cheap, this can lead to massive cost reduction over supervised learning. While active learning for classification has been explored both theoretically and practically, there is much less work done on regression. In this work we focus on noiseless regression where we devise an algorithm to pick instances to label and train a Kernel Ridge Regression on this data. Central to this is the Frank Wolfe algorithm which greedily optimises the Maximum Mean Discrepancy between the full dataset and train set. This leads to an algorithm that optimises an upper bound on the empirical risk of the estimator trained on the built train set. We try to control the true generalisation error but current results from convergence of Frank Wolfe means we can't say whether or not this improves upon random sampling. We apply the algorithm to a wide range of datasets and compare it to leverage score sampling. We show that the algorithm performs competitevely with other algorithms and the benefit of using active learning for regression. We also show that it dominates other methods in the agnostic setting of both regression and classification. We show that using Frank Wolfe to mitigate domain drift is competitive and works well in practice, with learning curves that show greater performance than random sampling on a wide range of datasets in both agnostic and realisable setting.

Impact Statement

The knowledge and algorithms in this dissertation have the potential to have industrial use in any setting where labelling is costly, or when there is a fixed budget of how many instances it is possible to label. This is a setting which occur in many different areas, to name a few, medical healthcare, process engineering, natural language processing and data mining of documents. As this contribution is derived from theoretical considerations, it can be used both locally and internationally, by anyone anywhere.

From an academic point of view, the information and knowledge in this dissertation will act as a springboard for future researchers and students, and add to the existing pool of knowledge in the field of machine learning. The hope it that this will foster further ideas on how to improve the field of active learning for regression, building on results derived here and extending it to more general and different settings.

Acknowledgements

To my parents, thank you.

To Carlo and Massi, Cheers!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

A cornerstone of machine learning is the paradigm of supervised learning. In supervised learning the goal is to learn an input / output (also called label) relationship depicted by $f : \mathcal{X} \to \mathcal{Y}$, where f represents a relationship mapping from the input space $\mathcal{X}$ the output space $\mathcal{Y}$. We learn this f from a collection of n input-output pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ sampled from the underlying phenomenon giving rise to f. However, in supervised learning the algorithm does not have any way of choosing how this dataset is created. This puts some restrictions on its use in practice as certain algorithms are too computationally expensive to train on full datasets or the case when instances are cheap to collect, but expensive to label.

The second setting occurs frequently in practice. Today data is being created in abundance due to human use of social media, email and devices [25], and due to data being collected from industrial processes [44]. Much of this data is unstructured and in most cases the output we want to predict is not available directly. An example of this is the text data being generated by the web; platforms such as Twitter produces massive amounts of free text which is not labelled at all. For example, when trying to do sentiment analysis on sentences from Twitter, it is necessary to heuristically assign a label to an instance or send it to an expert for labelling [43]. This begs the question if we can design strategies for what instances we should label first in order to learn patterns faster, in comparison to labelling instances at random.

Active learning [17] is a machine learning paradigm which generalises supervised learning by allowing the algorithm in some sense to decide what instances to query for the corresponding output. The hope is that this will lead to close-to-optimal performance in much fewer labels compared to a supervised learning algorithm trained on a dataset of the same size. In this way, it is a question of the information content of the dataset. By choosing instances carefully we may create a dataset that is richer in information with respect to the pattern we want to learn than a dataset of same size, given to us from nature.

As such, active learning is similar to semi-supervised learning [12] in that it tries

to make efficient use of all available data. However, where semi-supervised learning uses the unlabelled data to improve performance by inferring aspects of the marginal distribution, active learning builds a dataset by querying the unlabelled data and getting the corresponding output. It is possible to combine the two [60].

As a concrete example, imagine the following problem: We are on $[0, 1] \subseteq \mathbb{R}$ and we know that there exist some true relationship $f(x) = \mathbf{1}(w^* \leq x)$ where $w^* \in (0, 1)$, but we do not observe the output directly but have to ask for it, an operation we call *querying* an instance.

Nature draws instances $x \sim \rho_{\mathcal{X}}$ where $\rho_{\mathcal{X}}$ is some distribution over the domain $\mathcal{X} = [0, 1]$. Our goal is to find a good guess $h_w(x) = \mathbf{1}(w \leq x)$ so that we incur a small loss, $\Pr_{\rho_{\mathcal{X}}}(h_w(x) \neq f(x))$, in as few queries of labels as possible. Since we know the form of the relationship, we also know that if we have queried an instance x_1 which leads to the pair $(x_1, f(x_1))$ where $f(x_1) = 0$ (which means that $x_1 \leq w^*$) then we also know that for any other $x_2 \leq x_1 \leq w^*$, $f(x_2) = 0$. This means that **querying any instance smaller than x_1 is unnecessary**.

It can be shown that for a set of $O(\frac{1}{\epsilon})$ unlabelled instances, an active learning algorithm have to query only $O(\log \frac{1}{\epsilon})$ instances in order to find a h_w such that $\Pr_{\rho_{\mathcal{X}}}(h_w(x) \neq f(x)) < \epsilon$ with high probability. Compared to supervised learning which requires $O(\frac{1}{\epsilon})$ this is an exponential speedup [23, 22]!

In practice, it is often not so simple. Active learning has been shown to not yield better performance than supervised learning when considering worst case scenarios[1] which hints that only when we can identify and exploit structure of the problem, such as the knowledge of the threshold function in the example above, can we hope to improve over normal supervised learning. In the end this has come down to identifying conditions on the data-generating distribution and the target class which enables active learning to improve over supervised learning [6].

1.2 Research Focus

The theoretical foundations of active learning for binary classification is fairly well-understood, with results which are essentially always at least as good as supervised learning [5]. However, results for regression is more sparse and less well-developed. For regression, the assumption is often that the bias term of the error decomposition is negligible, see for example [18]. There has been attempts to control for this directly, but relies on assumptions that basically reduces regression to classification [56]. Thus a natural direction is to consider active learning for regression based on statistical learning theory and investigate how the bias term can be understood and controlled.

In general, statistical learning theory considers how to minimise the risk, which is a much harder problem than minimising the empirical risk. This stems from the problem of generalisation and the fact that good performance on the train set does

[1] Which is done in statistical learning theory.

not necessarily imply good performance on the test set. However, given that we know the data-generating distrubution, active learning can be shown to be closely related to quadrature [10] and optimal experimental design [29]. The main difference is the setting and assumptions, for quadrature the distribution is in general known and optimal experimental design assumes very stringent conditions on the problem at hand.

In this sense we are interested in active learning procedures that are motivated by statistical learning theory and grounded on error bounds through the use of controlling the sampling and approximation error. Ideally we would like to find necessary assumptions for active learning to yield performance improvements over supervised learning, that is, an active learning algorithm with statistical guarantees that performs better than supervised learning in probability. Specifically, in this dissertation we aim to do the following

- Review the literature on active learning for regression in a theoretical setting including necessary results from RKHS theory and optimisation
- Specify and find necessary conditions for enabling error bounds of the risk, which we term the objective
- Derive an active learning algorithm for regression based on optimising objective
- Empirically evaluate and compare this algorithm to other active learning algorithms for regression

1.3 Research Aims and Individual Research Objectives

The aim of this dissertation is to critically analyse and summarise the literature on active learning in the setting of regression and provide conditions and assumption necessary for a theoretically grounded active learning algorithm with guarantees to be possible. We do this by leaning on the well-developed field of RKHS theory, which is in some sense an ideal setting for statistical learning due to various theoretical properties such as the reproducing property and the representer theorem.

We further aim to derive bounds on the quantity of interest, a modified excess risk for active learning, and provide a framework for comparing active learning algorithms to supervised learning algorithms by framing active learning as a generalisation of supervised learning. In addition to this, we will make connections with other fields and use recent advances in machine learning, based on the Frank Wolfe algorithm applied to RKHS's.

Finally we aim to produce an algorithm which satisfy theoretical guarantees which is shown to be empirically competitive with other similar algorithms. This will be done through a rigorous investigation of how the algorithm perform in practice by evaluating the algorithm on regression and finally classification tasks, displaying the

theoretical performance in practice and show that when applied to settings where the theoretical assumptions do not hold, it's still competitive.

1.4 Value of this Research

The contribution of this research will be the following:

- Produce a review of the existing methods and approaches to active learning in regression which is currently lacking to the extent that binary classification active learning is a synonym with active learning
- Investigate and develop an area of active learning that is underdeveloped compared to binary classification
- Synthesise methods from statistical learning theory, kernel theory and optimisation in order to produce novel algorithms for active learning
- Identify failure cases and set a basis for future work to build upon

Chapter 2

Literature Review

In this chapter, we review and compile the work done on active learning for regression based on RKHS theory. We divide this into three parts: the first focusing on the theoretical foundation of RKHS theory, the second active learning theory and previous work using RKHS to do active learning for regression, and finally we give an overview of Kernel Herding (hereafter KH) and its link to the Frank Wolfe (hereafter FW) algorithm for optimising the maximum mean discrepancy efficiently.

2.1 Reproducing Kernel Hilbert Space Theory

2.1.1 A brief history of the what and the why of Reproducing Kernel Hilbert Spaces

Kernels have a long history and the field was arguably started with the work of [2]. Since then it has found use, to name a few, in geostatistics in the form of Kriging [21], machine learning with support vector machines [27] and gaussian processes [57].

A first question might be why we are interested in RKHS's when in many other areas of mathematics and the sciences we consider L^2 spaces. One of the first obstacles is that the space of $L^2(\mathcal{X}, \rho_{\mathcal{X}})$ is actually not a space of functions, but equivalence classes of functions which agree except on a set of measure zero with respect to $\rho_{\mathcal{X}}$. In particular for specific choices of kernel functions, for any compact subset $\mathcal{Z}$ of the input space $\mathcal{X}$, the set of $\overline{\text{span}(K_x, x \in \mathcal{Z})}$ is dense in $C(\mathcal{Z})$, the space of all continuous functions on $\mathcal{Z}$ with respect to the maximum norm, showing the power of RKHS's.

A second reason is that by choosing an RKHS as the hypothesis space turns the regularised empirical risk minimisation objective, when using a convex loss, into a nice convex program through the use of the representer theorem [32, "Large" Reproducing Kernel Hilbert Spaces]. This also allows us to turn optimisation over a space of functions to a dual problem over a finite-dimiensional vector space, reducing it to the domain of linear algebra and giving us access to the tools therein.

However, there are drawbacks of using kernels for learning. It is well-known that most kernel algorithms require inversion of the kernel matrix, including kernel ridge regression, which scales cubically in the number of datapoints [46] making it

prohibitively expensive except for small datasets. Luckily this goes very well with active learning since we are actually interested in making the size of the dataset small while retaining close-to-optimal performance. This means that we are not really giving up scalability as we are already making the aim to label as few instances in the original dataset as possible. Hence RKHS is an ideal space to work with as it gives us nice theoretical properties, a well-developed theory and when used in active learning, few of the drawbacks usually encountered.

2.1.2 Reproducing Kernel Hilbert Spaces

The field of RKHS theory is well-established and there has been much written about it, we will follow the course on RKHS theory taught at UCL 2018-2019 [32] together with introductions of [40, 28].

A Hilbert space is a tuple $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ where $\mathcal{H}$ is a vector space and $\langle\cdot,\cdot\rangle$ is an inner product defined on some set $\mathcal{X}$ such that this space is complete with respect to the metric $\|\cdot\|$ induced by the inner product. An RKHS is a specific type of Hilbert space with the following property:

Definition 1. *Let $\mathcal{X}$ be a non-empty set and let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$, then $\mathcal{H}$ is said to be an RKHS if for any $x \in \mathcal{X}$ the evaluation functional $\delta_x : f \mapsto f(x)$ is continuous or equivalently bounded.*

The above definition is often too abstract in practice and we will show how to equivalently construct RKHs's through the use of specific concepts of *kernels, reproducing kernel* and *positive definite functions*, starting with the definition a kernel,

Definition 2. *Let $\mathcal{X}$ be a non-empty set and K a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Then K is called a kernel on $\mathcal{X}$ if there exists some Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that for any $x, x' \in \mathcal{X}$, $K(x, x') = \langle\phi(x), \phi(x')\rangle_{\mathcal{H}}$.*

Secondly the concept of a *reproducing kernel*,

Definition 3. *Let $\mathcal{X}$ be a non-empty set and let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a reproducing kernel of $\mathcal{H}$ if it satisfies the following properties*

$$\forall x \in \mathcal{X}, \quad K_x = K(\cdot, x) \in \mathcal{H}, \tag{2.1}$$

$$\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \quad \langle f, K_x\rangle_{\mathcal{H}} = f(x), \tag{2.2}$$

that is, all functions K_x are in the space and they have the so called reproducing property.

In particular, we have that

Corollary 1. *For any* $x, x' \in \mathcal{X}$

$$K(x, x') = \left\langle K(\cdot, x), K(\cdot, x') \right\rangle_{\mathcal{H}}.$$

In fact, every reproducing kernel is a kernel

Corollary 2. *Every reproducing kernel is a kernel with explicit feature map* $\phi : x \mapsto K(\cdot, x) \in \mathcal{H}$.

Finally, we have the following theorems which state that an RKHS is characterised by its reproducing kernel function and it is unique:

Theorem 3. *If it exists, a reproducing kernel is unique.*

Theorem 4. $\mathcal{H}$ *is an RKHS if and only if it has a reproducing kernel.*

A reproducing kernel is useful since it enables us to use machinery from linear algebra to bound quantities. For example, we can use cauchy-schwartz (from here on denoted *CS*) to bound $f(x)$ using $\kappa = \sup_{x \in \mathcal{X}} \sqrt{K(x, x)}$,

Example 1. *When* $f \in \mathcal{H}$*, an RKHS with reproducing kernel* K*, then we have the following*

$$\begin{aligned} |f(x)| &= |\langle f, K(\cdot, x) \rangle_{\mathcal{H}}| \\ &\leq \|f\|_{\mathcal{H}} \|K(\cdot, x)\|_{\mathcal{H}} \\ &\leq \|f\|_{\mathcal{H}} \sqrt{K(x, x)} \\ &\leq \|f\|_{\mathcal{H}} \sup_{x \in \mathcal{X}} \sqrt{K(x, x)} \\ &= \|f\|_{\mathcal{H}} \cdot \kappa \end{aligned}$$

where we have used the CS inequality since $\mathcal{H}$ is a Hilbert space. Thus we have

$$f(x) \leq \|f\|_{\mathcal{H}} \cdot \kappa, \tag{2.3}$$

which shows that if a function is in an RKHS it is enough to control the norm $\|f\|_{\mathcal{H}}$ in order to control f at any point $x \in \mathcal{X}$.

The final part we need is that of *positive semi-definiteness*,

Definition 4. *Let* $\mathcal{X}$ *be a non-empty set then* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *is a positive semi-definite function if for any* $n \geq 1$*, for any* $(\alpha_i)_{i=1}^n \in \mathbb{R}^n$*, for any* $(x_i)_{i=1}^n \in \mathcal{X}^n$*,*

$$\sum_{i,j}^{n} \alpha_i \alpha_j K(x_i, x_j) \geq 0,$$

and we say that K *is positive definite if for mutually distinct* x_i*, the inequality is strict for any vector* $(\alpha_i)_{i=1}^n \neq \mathbf{0}$.

Every inner product is a positive semi-definite function, which implies that

Corollary 5. *Every kernel K is a positive semi-definite function.*

At this point we have shown that for any reproducing kernel K of an RKHS $\mathcal{H}$, it is also a kernel and furthermore is positive semi-definite, finally we also have that every positive semi-definite function K is the reproducing kernel for some RKHS $\mathcal{H}$,

Theorem 6 ([2])**.** *Let $\mathcal{X}$ be a non-empty set and let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive semi-definite function. Then there exists a unique RKHS $\mathcal{H} \subset \{f : \mathcal{X} \to \mathbb{R}, f \text{ is measureable}\}$ such that K is the reproducing kernel of this space.*

Looking back, we see that using an RKHS as the space of functions under consideration in machine learning is useful as it enables us to use the *kernel trick*: if an algorithm is defined only in terms of elementary operations over inner products between data vectors, we can replace each inner product $\langle x, x' \rangle$ with a kernel $K(x, x')$ in order to work implicitly in a (potentially) infinite dimensional space.

There are many different kernels that can be used and can be combined in various ways according to rules we won't go into here, yielding a sort of calculus of kernels. We will only consider the so called Gaussian kernel,

Definition 5. *Let $\mathcal{X} \subseteq \mathbb{R}^d$ for some $d > 0$, the gaussian kernel $K_\sigma : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ has the form of*

$$K_\sigma(x, x') = \exp\left(-\frac{\|x - x'\|_2^2}{2\sigma^2}\right), \tag{2.4}$$

where $\sigma > 0$.

The Gaussian kernel is ubiqituos in machine learning and other fields which use RKHS's, in particular it has the following properties

Theorem 7. *For any $\sigma > 0$, the Gaussian kernel is*

- *Characteristic*
- *Universal*
- *Stationary*

.

2.1.3 Kernel Mean Embedding

Kernel Mean Embedding (hereafter *KME*) generalises the reproducer for $x \in \mathcal{X}$, K_x, to that of a distribution $\rho \in M_1^+(\mathcal{X})$. We will follow the review of [41] and the notes of [32].

Definition 6. *Let $\mathcal{X}$ be a non-empty set and let $M_1^+(X)$ be the space of distributions on a measurable space $(\mathcal{X}, \Sigma)$. The kernel mean embedding of a distribution $\rho \in M_1^+(X)$ into a RKHS $\mathcal{H}$ endowed with a reproducing kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by a mapping*

$$\mu : M_1^+(\mathcal{X}) \to \mathcal{H}, \quad \mu(\rho) = \int_{\mathcal{X}} K_x \mathrm{d}\rho(x). \tag{2.5}$$

We will use the shorthand $\mu_\rho \coloneqq \mu(\rho)$. Note that this generalises the definition of K_x since a dirac delta distribution $\delta_{x'}$ gives us $\mu(\delta_{x'}) = \int_{\mathcal{X}} K_x \mathrm{d}\delta_{x'} = K_{x'}$. In this way μ can be seen as the analogue to ϕ when working with distributions rather than points.

The following lemma gives necessary condition for the element to be in $\mathcal{H}$. We reproduce the proof as it exposes techniques we will be using later.

Theorem 8. *Let $\mathcal{H}$ be an RKHS with reproducing kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. If $\mathbb{E}_\rho[\sqrt{K(X,X)}] < \infty$ and $f \in \mathcal{H}$, then $\mu_\rho \in \mathcal{H}$ and $\mathbb{E}_\rho[f(X)] = \langle f, \mu_\rho \rangle_{\mathcal{H}}$.*

Proof. Let L_ρ be the linear operator defined as $L_\rho(f) = \mathbb{E}_\rho[f(X)]$. Then

$$\begin{aligned} |L_\rho(f)| &= |\mathbb{E}_\rho[f(X)]| \\ &\leq \mathbb{E}_\rho[|f(X)|] \\ &= \mathbb{E}_\rho[|\langle f, K_X \rangle|] \\ &\leq \mathbb{E}_\rho[\sqrt{K(X,X)}\|f\|_{\mathcal{H}}] \end{aligned}$$

where we first apply Jensen's inequality then the CS inequlity. By Riesz representation theorem, there exists an element $\lambda_\rho \in \mathcal{H}$ such that $L_\rho(f) = \langle \lambda_\rho, f \rangle_{\mathcal{H}}$. Letting $f = K_x$ for some $x \in \mathcal{X}$, then $\lambda_\rho(x) = L_\rho(K_x) = \int_{\mathcal{X}} K(x, x') \mathrm{d}\rho(x')$ which shows that $\lambda_\rho = \mu_\rho$. □

Trivially we have the following corollary

Corollary 9. *If $\kappa = \sup_{x \in \mathcal{X}} \sqrt{K(X,X)} < +\infty$ then for any $\rho \in M_1^+(\mathcal{X})$, $\mu_\rho \in \mathcal{H}$.*

Proof. Since $\mathbb{E}_\rho[\sqrt{K(X,X)}\|f\|_{\mathcal{H}}] \leq \kappa \|f\|_{\mathcal{H}}$ we are done. □

We finish by considering the KME of the empirical distribution of a dataset. Let $S = (x_i)_{i=1}^n$ be a sequence of n datapoints from some set $\mathcal{X}$, then the empirical distribution of S, call this ρ_S, is defined to be

$$\rho_S = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \tag{2.6}$$

and the corresponding mean embedding is $\mu_{\rho_S} = \int_S K_x \mathrm{d}\rho_S = \frac{1}{n}\sum_{i=1}^n K_{x_i}$. If S is the set of n samples sampled *iid* from some distribution ρ, then we say

$$\hat{\mu}_\rho \coloneqq \mu_{\rho_S} \tag{2.7}$$

as it is an *empirical estimate* of the mean embedding μ_ρ.

2.1.4 Maximum Mean Discrepancy

Integral Probability Metrics (hereafter *IPM*) [42] are distance-like functions defined in the following way

Definition 7. *Let $\mathcal{F}$ be a space of real-valued bounded measurable functions on a non-empty set $\mathcal{X}$. Given two probability measures ρ, ξ with support of $\mathcal{X}$, then*

$$\mathrm{IPM}_\mathcal{F}(\xi, \rho) = \sup_{f\in\mathcal{F}} \left(\int_\mathcal{X} f(x)\mathrm{d}\xi(x) - \int_\mathcal{X} f(x)\mathrm{d}\rho(x) \right). \tag{2.8}$$

Any space $\mathcal{F}$ defines an IPM, we will only consider the case when $\mathcal{F}$ is a unit ball in an RKHS, in which case we call this metric the Maximum Mean Discrepancy (hereafter *MMD*). It's simple to show that in this case the MMD takes the following simple form,

Theorem 10. *Let B be the unit ball of an RKHS with reproducing kernel K, then we can express the MMD as*

$$\mathrm{MMD}_B(\xi, \rho) = \|\mu_\xi - \mu_\rho\|_\mathcal{H}. \tag{2.9}$$

This leads to the following expression of the MMD as the expectation over kernels,

$$\mathrm{MMD}_B(\xi, \rho)^2 = \mathbb{E}_{X,X'\sim\xi}[K(X, X')] - 2\mathbb{E}_{X\sim\xi, Y\sim\rho}[K(X, Y)] + \mathbb{E}_{Y,Y'\sim\rho}[K(Y, Y')], \tag{2.10}$$

where the notation $\mathbb{E}_{X,X'\sim\xi}[K(X, X')]$ indicates that we draw two independent identically distributed copies form ξ. When the distributions are discrete, this turns into sums over kernel evaluations,

Corollary 11. *When $\xi = \sum_{i=1}^n \alpha_i \delta_{x_i}$ and $\rho = \sum_{j=1}^m \beta_j \delta_{y_j}$, where we collect $(\alpha_i)_{i=1}^n, (\beta_j)_{j=1}^m$ into vectors α, β, then we have*

$$\mathrm{MMD}_B(\xi, \rho)^2 = \sum_{i,i'}^n \alpha_i \alpha_{i'} K(x_i, x_{i'}) - 2\sum_{i,j}^{n,m} \alpha_i \beta_j K(x_i, y_j) + \sum_{j,j'}^n \beta_j \beta_{j'} K(y_j, y_{j'}) \tag{2.11}$$

$$= \alpha^T \mathbf{K}_{nn} \alpha - 2\alpha^T \mathbf{K}_{nm} \beta + \beta^T \mathbf{K}_{mm} \beta, \tag{2.12}$$

where $\mathbf{K}_{nn} \in \mathbb{R}^{n\times n}, \mathbf{K}_{nm} \in \mathbb{R}^{n\times m}, \mathbf{K}_{mm} \in \mathbb{R}^{m\times m}$ are matrices such that $(\mathbf{K}_{nn})_{lt} = K(x_l, x_t), (\mathbf{K}_{nm})_{lt} = K(x_l, y_t), (\mathbf{K}_{mm})_{lt} = K(y_l, y_t)$.

When the kernel is characteristic, the MMD is a true metric

Theorem 12. *When K is a characteristic kernel, MMD is a metric on distributions in $M_1^+(\mathcal{X})$.*

2.2 Active Learning

We introduce the necessary concepts of Active Learning by first introducing Supervised Learning (hereafter *SL*) under the Statistical Learning Theory (hereafter *SLT*) model following the book [49] and the notes from the Learning Theory part of the course in Advanced Topics in Machine Learning given at UCL 2019 [15]. After this we review active learning as a field, looking at the historical developments and sub-divisions. We will then focus on the so called *pool-based active learning*, showing how it relates to supervised learning and critically analyse the works using RKHS theory. We lean on the comprehensive introduction of [47].

2.2.1 Statistical Learning Theory and Supervised Learning

SLT aims to introduce a framework grounded on probability theory that answers such questions as *what does it mean for an algorithm to learn* and *How do we design good learning algorithms* [15, Lecture 1].

The SLT framework introduces the following concepts,

Domain set An arbitrary set, $\mathcal{X}$. The set of objects that we wish to find the output for. We also refer to this set as the *input space* and call a point $x \in \mathcal{X}$ an *instance* or *input*. We assume that $\mathcal{X} \subseteq \mathbb{R}^D$ for some $D \in \mathbb{N}$.

Co-domain set An arbitrary set, $\mathcal{Y}$. Our goal is to find a good mapping from $\mathcal{X}$ to $\mathcal{Y}$ in the sense that we should be able to predict the output in $\mathcal{Y}$ for an arbitrary instance $x \in \mathcal{X}$, hence why it's called the *output space* or *target space*, occasionally we will call y a *label*. We will focus on regression, assuming that $\mathcal{Y} \subseteq \mathbb{R}$.

Train set $S = (x_i, y_i)_{i=1}^n$ is a finite sequence of pair, where each pair $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

Hypothesis space A space of functions $\mathcal{H} \subseteq \{h : \mathcal{X} \to \mathcal{Y},\ h \text{ measurable}\}$ meant to represent all of the possible input-output rules that we consider. Note that we have used $\mathcal{H}$ for both an RKHS and hypothesis space, in practice this doesn't matter as we will always use an RKHS as the hypothesis space.

Learning algorithm A function that takes as input a train set and maps to a hypothesis space,

$$\mathcal{A} : \cup_{i=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{H}. \tag{2.13}$$

Data generating distribution We assume there exists some distribution ρ with support on $\mathcal{X} \times \mathcal{Y}$ such that $\rho(x, y) = \rho(y|x)\rho_{\mathcal{X}}(x)$. The train set $S = (x_i, y_i)_{i=1}^n$

is sampled *iid* from ρ, that is, each pair $(x_i, y_i) \in S$ is sampled independently from ρ and we write $S \sim \rho^n$. We do not have access to ρ directly, but only through S.

Measure of performance We choose a *loss function*, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ that quantifies the loss of predicting $\hat{y}$ when the output is y through $\ell(y, y')$. The goal of SLT is then to minimize the *risk*, which is a function of an estimator h,

$$\mathcal{E}(h) = \mathbb{E}_\rho[\ell(h(x), y)], \tag{2.14}$$

and we will be writing $\mathcal{E}_\rho(h)$ to indicate the risk with respect to what measure.

The goal of SLT is to find an algorithm $\mathcal{A}$ that with high probability over the sampled train set S is close in performance to the optimal estimator in the hypothesis space. This is formalised by the concept of *excess risk*,

Definition 8. *Given the above, we define the excess risk to be the quantity*

$$\mathcal{E}_\rho(h) - \mathcal{E}_\rho(h^*), \tag{2.15}$$

where $h^* \in \arg\min_{h' \in \mathcal{H}} \mathcal{E}_\rho(h')$.

The excess risk measures the additional risk that we take on by using h rather than the optimal estimator in the hypothesis space. In practice, the estimator h will be random since it will be the output of an algorithm $\mathcal{A}$, which maps from a train set S to h[1], making the excess risk itself random due to the random sampling of $S \sim \rho^n$.

In order to control the excess risk in probability, statements are made of the following form

Example 2. *With probability larger than* $1 - \delta$ *taken over the sampling of the train set* $S \sim \rho^n$*, for any* $\delta \in [0, 1]$*, we have that the excess risk is bounded,*

$$\mathcal{E}_\rho(\hat{h}_n) - \mathcal{E}_\rho(h^*) \leq \epsilon$$

and these bounds are derived using error decomposition and controlling for the error in probabiliy using assumptions on the hypothesis class, data generating distribution etc. One famous example of this is Empirical Risk Minimization (hereafter *ERM*)[53] where the algorithm $\mathcal{A}$ is defined as the minimizer of the empirical risk, which in our setting reduces to the following definition,

Definition 9. *Given an input and an output space* $\mathcal{X}, \mathcal{Y}$*, an RKHS* $\mathcal{H}$ *acting as our hypothesis space, with loss* ℓ *and a dataset* $S = (x_i, y_i)_{i=1}^n$*, Empirical Risk*

[1] When the function h is the output of an algorithm based on a train set S of size n we denote $\hat{h}_n := \mathcal{A}(S)$. If the size of S is obvious we suppress the n and simply write $\hat{h}$.

Minimization is an algorithm

$$\mathcal{A}(S) = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i). \tag{2.16}$$

In this work, we will focus on Regularised Empirical Risk Minimization (herafter *RERM*) specialised to our setting,

Definition 10. *Given input and output space $\mathcal{X}, \mathcal{Y}$, an RKHS $\mathcal{H}$ acting as our hypothesis space, with loss ℓ and a dataset $S = (x_i, y_i)_{i=1}^n$, Regularised Empirical Risk Minimization is an algorithm*

$$\mathcal{A}(S) = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i) + \lambda \|h\|_{\mathcal{H}}^2. \tag{2.17}$$

Kernel Ridge Regression (hereafter *KRR*) is a generalisation of Ridge Regression [36] where all of the inner products are lifted to the feature space through the use of the kernel function. KRR can be seen as the solution to the RERM when we have an RKHS $\mathcal{H}$ and mean squared error $\ell(y, y') = (y - y')^2$ with $\mathcal{X} \subseteq \mathbb{R}^D$ and $\mathcal{Y} \subseteq \mathbb{R}$, so that

$$\mathcal{A}(S) = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 + \lambda \|h\|_{\mathcal{H}}^2. \tag{2.18}$$

It can be shown that KRR has the following solution

$$\mathcal{A}(S)(x) = \mathbf{k}_x^T \alpha_* = \mathbf{k}_x^T (\mathbf{K}_{nn} + \lambda n \mathbf{I}_n)^{-1} \mathbf{Y}, \tag{2.19}$$

where

$$\mathbf{k}_x = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_n), \end{bmatrix} \tag{2.20}$$

and

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n, \end{bmatrix} \tag{2.21}$$

.

We judge the quality of an algorithm with respect to its *error bound*. An error bound gives us probabilistic guarantees on the magnitude of the excess risk as defined in 8,

Definition 11. *A supervised learning error bound $\epsilon(n, \delta)$ is a function depending on an algorithm $\mathcal{A}$ that given a $\delta \in [0, 1]$ assures that the excess risk will be less than ϵ*

with probability greater than $1 - \delta$*. Or as a mathematical expression,*

$$\Pr_{S \sim \rho^n} (\mathcal{E}(\mathcal{A}(S)) - \mathcal{E}(h^*) \leq \epsilon(n, \delta)) \geq 1 - \delta. \tag{2.22}$$

2.2.2 Active Learning Theory Review

Active learning generalises the setting of supervised learning by introducing an aspect of cost to acquire labels of instances. According to Settles

> Active learning is any form of learning in which the learning program has some control over the inputs on which it trains [47].

In the SL setting we are *given* a train set S consisting of instance and output-pairs sampled from ρ. Compare this to a setting where we are only given a large set of *instances* but attaining the outputs of an instances is costly, it seems possible that we could attain an acceptable level of accuracy or risk by just asking for a few outputs rather than all of them. The latter example can be formalised as the setting of Active Learning and was hinted at in the Background section.

From literature there are proposed criteria for what a strategy for choosing instances to label should consider and allows us to classify algorithms according to how they implement each of these. [58] argue that a strategy should take into account the *informativeness*, *representativeness* and *diversity* of the instances chosen. This corresponds well with how the objective for choosing what instance to label often decompose into terms that can be interpreted as encouraging informative instances but discouraging instances similar to instances already labeled (for example [35, Equation 4] and [13, Equation 7]).

It is possible to define an active learning algorithm by specifying an *active learning tuple* $(\mathcal{A}, \mathcal{Q})$ where $\mathcal{A}$ is the algorithm of SL as defined in 2.13 and $\mathcal{Q}$ is the strategy for how to choose instances to query. The various setting of active learning then depends on what we need $\mathcal{Q}$ to do and according to what objective we want to have good performance on. This leads to the subsettings of active learning.

Active learning can be split up into three subsettings, being *Query Synthesis*, *Stream-Based Selective Sampling* and *Pool-based Active Learning* [47]. These can be further divided into sequential and batch-mode active learning where the first considers instances one-by-one while the second instead presents the learner with sets of instances [35]. The dominating stream of the two is that of sequential learning due to the simplicity of deriving results but also due to the combinatorial aspect of choosing subsets from a mother set leading to explosion in computational complexity unless using approximations. Furthermore, these approximations can sometimes be shown to be close-to-optimal. [38] shows that in an active learning setting where the goal is to optimally place sensors in predefined locations and casting this as a problem of maximising the Mutual Information between the sensor locations and the rest of

the locations, there is a polynomial-time greedy procedure that is within $(1 - 1/e)$ of optimum of the NP-hard combinatorial objective problem. All this indicates that the batch-setting is underexplored as the benefits are small over sequential active learning.

All active learning theory from here on will be considered in the sequential setting, and note that all batch-mode active learning can be rendered sequential by making the set of size 1. Active learning then operates in a loop where t is the time step at the start of the iteration, and the active learning algorithm gets an input to the querying strategy $\mathcal{Q}$ after which $\mathcal{Q}$ chooses what to do depending on the current setting, thereafter the algorithm is free to predict or do whatever needed for $\mathcal{Q}$ to choose the next point at time $t + 1$. The reason for the flexibility is that in general the instance chosen at $t + 1$ may depend on the output and predicted output of the trained model up to time t, that is the full history up until t.

2.2.3 Query synthesis

When the algorithm is allowed to synthesis instances *de novo* so that the instance chosen by $\mathcal{Q}$ at time t may be any point in $\mathcal{X}$ we call this query synthesis, or learning with membership queries [1]. One prominent issue with this is that we do not have access to $\rho_{\mathcal{X}}$ at all and thus cannot hope to minimise any risk unless this is supposed to be known a-priori.

[18] does query synthesis by assuming that $\rho_{\mathcal{X}}$ is uniform to make the problem tractable, learning the dynamics of an idealised robot arm. This can be a fruitful direction to take if the actual goal is about learning a concept in itself rather than aiming for low risk. However, there are failure cases when the instances are actually from some underlying distribution $\rho_{\mathcal{X}}$, which is expanded upon in [7] where they show that using humans to label synthesised images of MNIST lead to strange artifacts as the algorithm does not take into account the fact that the true marginal distribution on image only has support for a small subspace of all possible images of digits. This makes it problematic since we are trying to reduce the risk and need access to samples from $\rho_{\mathcal{X}}$ in order to learn about the marginal distribution.

2.2.4 Stream-Based active learning

This is also called selective sampling [17] due to the querying strategy deciding at each time step whether to query an instance, which arrive in an on-line fashion, or discard it and query a later instance. The assumption is that the source of the instances produce these virtually for free so cost only enters through querying. The example of learning a threshold concept in Background can be cast in this setting, where we reject any instance shown to us to be less than the biggest instance currently labeled 0. In this way, the querying strategy is a function $\mathcal{Q} : \mathcal{X} \to \{0, 1\}$ where 0 represents a decision to not label and 1 to label the current instance streamed.

Approaches of when to query often corresponds to the criteria of an instance

being informative, representative and to add to the diversity of the already queried instances so far. The most straightforward approach is to define a measure of utility of information content and bias the sampling towards instances with high information content. A second approach is that of using a version space [52].

This setting contains many theoretically grounded algorithms, for example [24]. They also point out some common problems with general active learning algorithms; they rely upon computations which are not feasible in practice or requires solving problems which are intractable in practice.

2.2.5 Pool-based active learning

When given a set of datapoints $S \sim \rho^n$ we may assume that we know the instances and that the labels exist, but are hidden from us until we query them for the corresponding outputs. When the querying comes with a cost we want to only query as many instances as needed to reach some level of performance before stopping. This is the pool-based active learning setting [39] and it differs from query synthesis in that the query strategy $\mathcal{Q}$ depends on the dataset S of which it only has access to the instances and labels from previously queried instances to pick which instance to query next.

Pool-based active learning has a lot of overlap with the information retrieval and data mining communities, due to the straightforward representation of databases with instances such as documents as a pool set where labels need to be produced by human experts [39, 52]. [48] go into how various utility based ways to choose what instance to label. Most of these depend on the output of the current best hypothesis (or in the case of Query By Committee, a set of hypotheses), which we will see is unnecessary in the setting we will consider.

One limitation of pool-based active learning is that if the pool of samples is big, the querying strategy theoretically need to calculate the utility of each instance and then pick the instance which maximises the utility. In practice, this is less of a concern as it's possible to approximate this scheme by for example first subsample a smaller unlabeled dataset and label the most informative instance in this. This poses a problem for us though, as unless we can show that making this approximation leads to guarantees in probability, it's not simple to see how this can be overcome.

We now define the necessary components for Pool-based active learning for regression. For a set of input-output pairs $P = (x_i, y_i)_{i=1}^n$ we denote the set of only instance as $P^x = (x_i)_{i=1}^n$. Pool-based active learning operates in a loop and we let t denote the current time-step, incremented at the end of each loop. We have the following

Pool set A set of instance-output pairs sampled from ρ before the start of the active learning loop, denoted by $P_0 = (x_i, y_i)_{i=1}^n \sim \rho^n$ and the corresponding set of only instances as P_0^x. The pool set acts as the pool of available instances that

the algorithm can query at each t, so that P_t^x are the instances not labeled at start of iteration t. At time t the algorithm only have access to P_t^x and will ask for labels of an instance from P_t^x.

Oracle The oracle is a function which takes as an input an instance and samples from the conditional distribution $\rho(Y|X = x)$. We will assume no output noise in our derivation, so the conditional distribution is deterministic. Formally we define $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{O}(x) = f(x)$ where $f(x)$ is the true relationship.

Querying strategy The strategy of how we decide what instance to label at time t which we denote $\mathcal{Q}$. We will make the assumption that $\mathcal{Q}$ chooses instances at t through a non-adaptive utility function[2] $\upsilon : \mathcal{X} \to \mathbb{R}$, which implicitly depends on the instances already queried although we omit this for notational ease, and that $\mathcal{Q}$ chooses an instance $x_t \in \arg\max_{x \in P_t^x} \nu(x)$. We call the chosen instance x_t^q and send it to the oracle to label giving us $y_t^q = \mathcal{O}(x_t^q)$.

Train set This is the analogue of the train set used in supervised learning. We denote this by S_t and the corresponding set of instances as S_t^x. The reason for including t as a subscript is that S_t is not static but is grown by one input-output pair at the end of each iteration.

In this way a pool-based active learning algorithm is defined by a tuple $(\mathcal{A}, \mathcal{Q})$ where $\mathcal{A}$ has the same role as in supervised learning, outputting a hypothesis $\hat{h}_t$ depending on the current train set S_t, but which is now built depending on the querying strategy $\mathcal{Q}$.

Algorithm 1 Pool based active learning

```
1:  procedure ActiveLearning(P_0, A, Q)
2:      S_0 ← ∅
3:      n ← |P_0|
4:      t ← 1
5:      while t ≤ n do
6:          x_t^q ← Q(P_t^x)
7:          y_t^q ← O(x_t^q)
8:          S_t ← S_{t−1} ∪ {(x_t^q, y_t^q)}
9:          P_t ← P_t \ {(x_t^q, y_t^q)}
10:         t ← t + 1
11:     end while
12: end procedure
```

The performance of an active learning algorithm will be considered to be its *error bound* which is defined similarly as in 11, but we now also let this depend on the size of S_t in addition to the size original pool set

[2]This simply means that the querying strategy is independent of the history of outputs from the queried instances.

Definition 12. *An active learning error bound $\epsilon(n, t, \delta)$ is a function depending on an algorithm $\mathcal{A}$ and a querying strategy $\mathcal{Q}$ that given a $\delta \in [0, 1]$ assures that the excess risk will be less than ϵ with probability greater than $1 - \delta$. Or as a mathematical expression, let S_t be the train set at the end of time t,*

$$\Pr_{S \sim \rho^n} (\mathcal{E}(\mathcal{A}(S_t)) - \mathcal{E}(h^*) \leq \epsilon(n, t, \delta)) \geq 1 - \delta. \tag{2.23}$$

This gives us a way to compare active learning algorithms to each other, and note that this also gives us a way to compare supervised learning to active learning by specifying $(\mathcal{A}, \mathcal{Q})$ and comparing the active learning error bound $\epsilon(n_{al}, t, \delta)$ to the supervised learning bound $\epsilon(n_{sl}, \delta)$ where $n_{al} = |P_0|$ is the number of original samples in the pool and $n_{sl} = |S|$ is the number of samples in the train set of supervised learning.

For active learning there has been several works based on generalisation bounds [31, 59, 34] but these are aimed at classification or graph-based prediction. For the setting of regression, where a normal choice is to consider square loss and kernel ridge regresion there is recently some promising results of [54] where they derive a bound of the form

$$\mathcal{E}_{P_0}(h) \leq \mathcal{E}_{S_t}(h) + \mathrm{MMD}(P_0^x, S_t^x) + \eta_{MMD} \tag{2.24}$$

and propose to base the querying strategy of optimising the MMD term. By making certain assumptions on the space for the MMD it can be shown that $\eta_{MMD} = 0$ and that the active learning bound decomposes into term which may be interpreted as the empirical fit and the drift from the instance set S_t^x to the original set of instances P_0^x. This makes it very clear that active learning in this setting is closely related to domain adaptation and domain drift. [20] show other measures in addition to MMD which may be used to control this domain drift through upper bounding the empirical risk in similar ways, but MMD is an appealing choice since it has an analytical form and analysing it comes down to qualities of the kernel matrix.

From a domain adaptation perspective, active learning under no noise is equivalent to the case of *covariate shift* [33].

2.3 Herding and Conditional Gradient Methods in RKHS's

2.3.1 Kernel Herding

Optimising the MMD term globally in the sense of picking k instances out of P_0^x can be shown to reduce to a quadratic binary programming problem and is thus NP-hard in general [11]. In [13] they relax this problem in two ways to choose instances, one through solving a convex quadratic programming problem and one through solving a linear programming problem, both which can be solved efficiently. While they

consider classification, since the algorithm is non-adaptive and based on controlling the same MMD term as in 2.24 it is directly comparable to what we are trying to do. We will insted consider optimising this term through the use of Frank-Wolfe [30, 37] of which herding can be shown to be a special case [4].

Herding started as an approach to generate deterministic *pseudo-samples* from observed moments of a posited model, without needing to first find the maximum likelihood estimates of the model parameters, but was limited in that $\mathcal{X}$ had to be finite [55]. The Herding algorithm was later generalised to arbitrary domains $\mathcal{X}$ by [14] noting that we can let $\mathbf{w}$, the parameters of the model, be an element of a **finite-dimensional** RKHS $\mathcal{H}$ with ϕ being the feature map of the corresponding reproducing kernel K. Given a probability distribution $\rho_{\mathcal{X}}$ on $\mathcal{X}$, kernel herding is the following

Algorithm 2 KernelHerding

procedure KernelHerding($\mathcal{H}$, $\rho_{\mathcal{X}}$, $\mathbf{w}_0 \in \mathcal{H}$)
  $\mu_{\rho\mathcal{X}} = \mathbb{E}_{X\sim\rho_{\mathcal{X}}}[\phi(X)]$
  $t \leftarrow 1$
  while $t < \infty$ **do**
    $x_t \leftarrow \arg\max_{x\in\mathcal{X}} \langle \mathbf{w}_{t-1}, \phi(x)\rangle_{\mathcal{H}}$
    $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \hat{\mu} - \phi(x_t)$
    $t \leftarrow t + 1$
  end while
end procedure

While this is very similar to the original herding algorithm, they also show that we can view the herding algorithm as a way of sequentially minimizing the squared error

$$\left\| \mu_{\rho\mathcal{X}} - \frac{1}{t}\sum_{j=1}^{t} \phi(x_j) \right\|_{\mathcal{H}}^2 . \tag{2.25}$$

If we look at the form of this, we can see that by 2.9 this is the same as

$$\mathrm{MMD}_{\mathcal{H}}(\rho_{\mathcal{X}}, \hat{\rho}_t)^2, \tag{2.26}$$

where $\hat{\rho}_t = \frac{1}{t}\sum_{j=1}^{t} \delta_{x_j}$. Furthermore, while Monte-Carlo sampling of $\hat{\rho}_t$, that is sampling *iid* from $\rho_{\mathcal{X}}$, decreases the squared MMD as $O(\frac{1}{t})$, KH improves upon this with $O(\frac{1}{t^2})$ which can be explained by the KH algorithm using *negative* auto-correlations to explore parts of the space where samples haven't been drawn from before, linking it to Quasi Monte Carlo methods and Low Discrepancy Sequences [14, 4]. They state the following which relates how herding and KH can be used in active learning,

Corollary 13 ([14], Corollary 6)**.** *An active learning algorithm selecting labels in accordance with the herding algorithm has guaranteed rate of convergence in terms of*

$O(\frac{1}{t})$. Moreover, the submodular greedy algorithm of [38] has therefor also at least the same approximation rate since it is within a constant fraction $(1 - e^{-1})$ of optimality.

which is in essence what we will be doing in this dissertation by building S_t using KH and FW, but trying to extend it to the generalisation error instead of considering the empirical excess risk.

2.3.2 Frank-Wolfe

Frank Wolfe [30] is an algorithm for solving general constrained convex optimization problems of the form

$$\min_{z \in \mathcal{C}} J(z), \tag{2.27}$$

where $\mathcal{C}$ is some compact convex subset of a Hilbert space $\mathcal{Z}$ and J is assumed to be convex over $\mathcal{C}$ and continuously differentiable. FW sets itself apart as it only requires access to the gradient ∇J on $\mathcal{C}$ and be able to find the minimiser of quantities of the form $\langle s, \nabla J(z) \rangle_{\mathcal{Z}}$, and does not require any projection steps while converging to $J(z^*) = \min_{z \in \mathcal{C}} J(z)$ as $O(\frac{1}{t})$ given certain regularity conditions [37].

Algorithm 3 FrankWolfe

```
procedure FrankWolfe(𝒵, 𝒞, J, ρ_t)
    t ← 1
    while t ≤ T do
        g̃_t ← arg min_{z∈𝒵} ⟨z, ∇J(x_{t−1})⟩_𝒳
        g_t ← (1 − ρ_{t−1}) g_{t−1} + ρ_{t−1} g̃_t
        t ← t + 1
    end while
end procedure
```

By rewriting the Kernel Herding algorithm, [4] showed that KH is a subset of a family of Frank-Wolfe algorithms over an RKHS $\mathcal{H}$. This is the setting we will work in and we describe the assumptions they use which will also be ours.

Assumption 1. *We have a set $\mathcal{X}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$, where $\mathcal{H}$ is an RKHS with reproducing kernel K. We assume that the data is uniformly bounded in the feature space, which means that for any $x \in \mathcal{X}$, $\|\phi(x)\|_{\mathcal{H}} \leq R$ for some $R > 0$.*

We denote $\mathcal{M}$ to be the marginal polytope, which is the convex hull of all vectors $\phi(x)$ for $x \in X$ a finite set $X \subseteq \mathcal{X}$. For any $f \in \mathcal{H}$ the following holds

Fact 14. *The supremum of $f(x)$ over $\mathcal{C}$ is attained by an extremum,*

$$\sup_{x \in \mathcal{X}} f(x) = \sup_{g \in \mathcal{M}} \langle f, g \rangle_{\mathcal{H}}. \tag{2.28}$$

the authors then show equivalence between KH and FW by considering the

optimization problem where we equate $\mathcal{H}$ with $\mathcal{Z}$ and $\mathcal{M}$ with $\mathcal{C}$

$$\min_{g\in\mathcal{M}} J(g) = \min_{g\in\mathcal{M}} \frac{1}{2}\|g - \mu_{\rho_X}\|^2_{\mathcal{H}} \tag{2.29}$$

where μ_{ρ_X} is the mean embedding of ρ_X into $\mathcal{H}$, leading to the FW update equations

$$\tilde{g}_t = \arg\min_{g\in\mathcal{M}} \langle g_{t-1} - \mu_X, g\rangle_{\mathcal{H}} \tag{2.30}$$

$$g_t = (1-\rho_{t-1})g_{t-1} + \rho_{t-1}\tilde{g}_t \tag{2.31}$$

and using $\rho_t = \frac{1}{t+1}$ we recover the KH algorithm. Note that for each t in 2.30 the chosen $\tilde{g}_t$ is assured to lie on the extremum according to 14. This shows that as X is a finite set of n points, running FW produces a sequence $(x_t)_{t=1}^n$ where x_t is associated with the corner of $\mathcal{M}$ such that $\tilde{g}_t = \phi(x_t)$ which means that when $X = P_0^x$ we can use the FW algorithm to choose points. In the utility framework, we let υ be the function

$$\upsilon(x) = -\left\langle g_{t-1} - \mu_{\rho_{P_0^x}}, \phi(x)\right\rangle_{\mathcal{H}}, \tag{2.32}$$

and using this we can create an algorithm by letting $\mathcal{Q}$ be the querying strategy that has utility function 2.32 of 1. We call this family of algorithms Frank-Wolfe Active Learning (FWAL), where we indicate the algorithm by specifying ρ_t.

2.3.3 Convergence rates for Frank-Wolfe

While KH had a convergence rate of $O(\frac{1}{t})$, FW has different rates depending on the step-size chosen at each t. In fact, FW can have much faster convergence if we use line-search as opposed to a fixed step-size. Let the constant

$$d = \arg\min_{g\in\partial_{\text{RelInt}}\,\mathcal{M}} \|g - \mu_{\rho_{\mathcal{X}}}\|_{\mathcal{H}} \tag{2.33}$$

where $\partial_{\text{RelInt}}\,\mathcal{M}$ is the relative boundary of $\mathcal{M}$. Note that this depends on $\mathcal{M}$ and we can have arbitrarily small d. The following table denotes upper bounds on the rate of convergence rates of $J(g_t) = \frac{1}{2}\|g_t - \mu_{\rho_{\mathcal{X}}}\|^2_{\mathcal{H}}$ using FW (any), FW-kh ($\rho_t = \frac{1}{t+1}$), FW-line search (hereafter *FW-ls*), where g_t is the output from the algorithms after t steps and sufficient (but not necessary) conditions for this to hold is laid out in the following table

Table 2.1: Upper bound on convergence rates using Frank Wolfe

Algorithm	Convergence (upper bound)	Additional Condition to 1
FW (any)	$4\frac{R^2}{t}$	None
FW-kh	$\frac{2R^4}{d^2t^2}$	$d > 0$
FW-ls	$R^2\exp\left(-\frac{d^2t}{R^2}\right)$	$d > 0$

Finally, [4] prove the following two propositions

Proposition 15. *Assume that $\mathcal{H}$ is finite-dimensional, that $\mathcal{X}$ is a compact topological measure space with a continuous kernel function K and that the distribution $\rho_{\mathcal{X}}$ has full support on $\mathcal{X}$. Then $d > 0$.*

Proposition 16. *Assume that $\mathcal{H}$ is infinite-dimensional, that $\mathcal{X}$ is a compact subspace of $\mathbb{R}^D$ with a continuous kernel function K on $\mathcal{X} \times \mathcal{X}$. Then $d = 0$.*

Chapter 3

Methodology

In this chapter we propose an active learning algorithm based on the bound of [54] and use FW to optimise this bound. In particular we use the tools laid out in the Literature Review in order to attempt to derive bounds on the excess active learning risk. We show that FW gives us guarantees of reducing the empiricial risk but that it's inconclusive given current results on rate of convergence if it improves on the generalisation error compared to random sampling.

3.1 Setting

The setting largely follows that of [54] and [8] as we fuse these two works and lean on results of both. We first specify all of our assumptions for the bound.

Assumption 2. *We assume that $\mathcal{X} \subseteq \mathbb{R}^D$ for some $D \in \mathbb{N}$ and $\mathcal{Y} \subseteq \mathbb{R}$. We assume there exists a deterministic labeling function $f : \mathcal{X} \to \mathcal{Y}$ and that there is some unknown distribution ρ on $\mathcal{X} \times \mathcal{Y}$. Furthermore, we assume that there is no output noise such that $\rho(x, y) = \rho_{\mathcal{X}}(x)\mathbf{1}(f(x) = y)$. At the start we get a pool of iid samples from ρ^n, which we call $P_0 = (x_i, y_i)_{i=1}^n$, and we start with an empty train set S_0.*

We furthermore make the following assumptions

Assumption 3. *We assume that $\mathcal{X}$ is compact.*

Assumption 4. *We assume that the output it bounded in magnitude,*

$$\sup_{y \in \mathcal{Y}} |y| = M_y < \infty.$$

Assumption 5. *The function $f \in B_\gamma$ where $B_\gamma \coloneqq B_\gamma(0) \subseteq \mathcal{H}$ where $\mathcal{H}$ is an RKHS with kernel function K.*

Assumption 6. *We are in realisable setting, our algorithm $\mathcal{A}$ is such that $f \in Ran(\mathcal{A})$.*

We will use the squared loss $\ell(y, y') = (y - y')^2$. We will consider the hypothesis space to be a ball in an RKHS $\mathcal{H}$ with reproducing kernel K, where the radius $\gamma < \gamma'$ so that f is in this hypothesis space. We introduce an auxiliary RKHS $\mathcal{H}'$ which is such that the reproducing kernel K' is $K'(x, x') = K(x, x')^2$, which will be associated with the functional space used for MMD calculations and in extension KH and FW.

We make the following assumptions, $H' = B_\eta \subseteq \mathcal{H}'$, and from 3, if we assume that K, K' are continuous on $\mathcal{X} \times \mathcal{X}$ we have that the norm in RKHS feature mappings, $\phi(x)$ are bounded in the RKHS for both $\mathcal{H}, \mathcal{H}'$ and we denote $\kappa = \sup_{x \in \mathcal{X}} \sqrt{K(x, x)} = \sup_{x \in \mathcal{X}} \|\phi(x)\|_{\mathcal{H}} < \infty$, and $\kappa' = \sup_{x \in \mathcal{X}} \sqrt{K'(x, x)} < \infty$.

3.2 MMD empirical generalisation bound

We now state results from [54] that will be used.

Theorem 17 ([54], Extension: Theorem 1)**.** *Let ℓ be any loss, and let $g_{f,h}(x) = \ell(h(x), f(x))$, then for any function $h \in H \subseteq \mathcal{H}$, H being an arbitrary subset of $\mathcal{H}$, any train set $S \subseteq P_0$ and let ρ_{S^x} be any distribution with support on S^x with ρ_S the corresponding distribution on $(x, f(x))_{x \in S}$, then for any $H' \subseteq \mathcal{H}'$*

$$\mathcal{E}_{P_0}(h) \leq \mathcal{E}_{\rho_S}(h) + \mathrm{MMD}_{H'}(P_0^x, S^x) + \eta_{MMD} \tag{3.1}$$

where $\eta_{MMD} = 2 \min_{\tilde{g} \in H'} \max_{h \in H, x \in P_0^x} |g_{f,h}(x) - \tilde{g}(x)|$.

Proof. This is essentially [54][Proof of Theorem 1]. Let $g_{f,h}(x) = \ell(f(x), h(x))$ and let $\tilde{g}$ be any function in H'.

Consider bounding the following quantity

$$|\mathcal{E}_{P_0}(h) - \mathcal{E}_{\rho_S}(h)|.$$

We decompose this as follows, letting $w_j = \rho_{S^x}(x_j)$ where $x_j \in S^x$ and $n = |P_0|, m = |S|$,

$$\begin{aligned}
|\mathcal{E}_{P_0}(h) - \mathcal{E}_{\rho_S}(h)| &= \left| \mathcal{E}_{P_0}(h) - \frac{1}{n} \sum_{i=1}^{n} \tilde{g}(x_i) + \frac{1}{n} \sum_{i=1}^{n} \tilde{g}(x_i) - \sum_{j=1}^{m} w_j \tilde{g}(x_j) + \sum_{j=1}^{m} w_j \tilde{g}(x_j) - \mathcal{E}_{\rho_S}(h) \right| \\
&\leq \underbrace{\left| \mathcal{E}_{P_0}(h) - \frac{1}{n} \sum_{i=1}^{n} \tilde{g}(x_i) \right|}_{(a)} + \underbrace{\left| \frac{1}{n} \sum_{i=1}^{n} \tilde{g}(x_i) - \sum_{j=1}^{m} w_j \tilde{g}(x_j) \right|}_{(b)} + \underbrace{\left| \mathcal{E}_{\rho_S}(h) - \sum_{j=1}^{m} w_j \tilde{g}(x_j) \right|}_{(c)}.
\end{aligned}$$

For (a) we can bound this as

$$\left|\mathcal{E}_{P_0}(h) + \frac{1}{n}\sum_{i=1}^{n}\tilde{g}(x_i)\right| \leq \frac{1}{n}\sum_{i=1}^{n}|g_{h,f}(x_i) - \tilde{g}(x_i)| \leq \max_{x\in P_0^x}|g_{h,f}(x) - \tilde{g}(x)|,$$

and for (c) we can similarly bound this as

$$\begin{aligned}\left|\mathcal{E}_{\rho_S}(h) - \sum_{j=1}^{m} w_j\tilde{g}(x_j)\right| &\leq \sum_{j=1}^{m} w_j|g_{h,f}(x_j) - \tilde{g}(x_j)| \leq \sum_{j=1}^{m} w_j \max_{x\in S^x}|g_{h,f}(x) - \tilde{g}(x)| \\ &= \max_{x\in S^x}|g_{h,f}(x) - \tilde{g}(x)| \leq \max_{x\in P_0^x}|g_{h,f}(x) - \tilde{g}(x)|,\end{aligned}$$

which follows from the fact that $S^x \subseteq P_0^x$.

For (b) we can bound this as follows

$$\left|\frac{1}{n}\sum_{i=1}^{n}\tilde{g}(x_i) - \sum_{j=1}^{m} w_j\tilde{g}(x_j)\right| \leq \sup_{\tilde{g}\in H'}\left|\frac{1}{n}\sum_{i=1}^{n}\tilde{g}(x_i) - \sum_{j=1}^{m} w_j\tilde{g}(x_j)\right| = \mathrm{MMD}_{H'}(\rho_{P_0^x}, \rho_{S^x}),$$

and altogether we have that

$$\begin{aligned}|\mathcal{E}_{P_0}(h) - \mathcal{E}_{\rho_S}(h)| &\leq \mathrm{MMD}_{H'}(\rho_{P_0^x}, \rho_{S^x}) + 2\max_{x\in P_0^x}|g_{h,f}(x) - \tilde{g}(x)| \\ &\leq \mathrm{MMD}_{H'}(\rho_{P_0^x}, \rho_{S^x}) + 2\min_{\tilde{g}\in H'}\max_{h\in H, x\in P_0^x}|g_{h,f}(x) - \tilde{g}(x)|\end{aligned}$$

.

Setting $\rho_{S^x} = \frac{1}{m}\sum_{j=1}^{m}\delta_{x_j}$ recovers the original proof. □

We note that 17 works for empirical distributions, but also reweighted distributions. This is useful as it allows us to use FW algorithms that outputs a distribution over S_t^x that is non-uniform, for example when using FW line search which gives a weighted distribution.

We have the following result

Theorem 18 ([54], Theorem 2)**.** *Let ℓ be the squared loss and assume $f \in B_\gamma$. Let K, K' be the kernel functions of the RKHS's $\mathcal{H}, \mathcal{H}'$ respectively. If $K'(x,x') = K(x,x')^2$ and $\eta \geq 4\gamma^2$, then it is guaranteed that $g(\cdot) = \ell(h(\cdot), f(\cdot)) \in H'$ and thus $\eta_{MMD} = 0$.*

When using a Gaussian kernel, this reduces to the following corollary

Corollary 19. *Let $f \in B_\gamma$ and let $K(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$, the gaussian kernel with bandwidth σ. If $K'(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma'^2}\right)$ with bandwidth $\sigma' = \frac{\sigma}{\sqrt{2}}$ and $\eta = 4\gamma^2$ then $\eta_{MMD} = 0$.*

Proof. A sufficient condition given by 18 is that if $K'(x,x') = K(x,x')^2$ and $\eta = 4\gamma^2$ then $\eta_{MMD} = 0$. As we are using gaussian kernels, this means that

$$\exp\left(-\frac{\|x-x'\|^2}{2\sigma'^2}\right) = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)^2 = \exp\left(-\frac{\|x-x'\|^2}{\sigma^2}\right)$$

which means that $\sigma' = \frac{\sigma}{\sqrt{2}}$. □

For η_{MMD} to be zero, we need the output of our algorithm to be contained in the ball B_γ. While KRR and Ivanov Regression is linked as KRR solves a lagrangian formulation of the Ivanov Regression problem, we need to make sure that the output $\hat{h} = \mathcal{A}(S)$ when doing KRR is such that $\left\|\hat{h}\right\|_{\mathcal{H}} \leq \gamma$. Following the proof of [19, Lemma 1] we have the following

Lemma 20. *The output of the KRR algorithm when applied to $S = (x_i, y_i)_{i=1}^n$ is such that for $\hat{h} = \mathcal{A}(S)$ we have $\left\|\hat{h}\right\|_{\mathcal{H}} \leq \frac{2M_y}{\gamma}$ and if $\lambda = \frac{2M_y\kappa}{\gamma}$ then the hypothesis space is contained within B_γ.*

Proof. Since $\hat{h} = \mathcal{A}(S)$ and this is the minimiser of the KRR objective, we see that

$$\mathcal{E}_S(\hat{h}) + \lambda\left\|\hat{h}\right\|_{\mathcal{H}}^2 \leq \mathcal{E}_S(0)$$

and we can then rearrange this by

$$\begin{aligned}
\lambda\left\|\hat{h}\right\|_{\mathcal{H}}^2 &= \mathcal{E}_S(0) - \mathcal{E}_S(\hat{h}) \\
&= \frac{1}{n}\sum_{i=1}^n (y_i^2 - (\hat{h}(x_i) - y_i)^2) \\
&= \frac{1}{n}\sum_{i=1}^n (2y_i\hat{h}(x_i) - \hat{h}(x_i)^2) \\
&\leq \frac{2}{n}\sum_{i=1}^n y_i\hat{h}(x_i) \\
&\leq \frac{2}{n}\sum_{i=1}^n M_y\left\|\hat{h}\right\|_{\mathcal{H}}\kappa \\
&= 2M_y\kappa\left\|\hat{h}\right\|_{\mathcal{H}}.
\end{aligned}$$

Assume that $\hat{h} \neq 0$, otherwise this is an equality, then we can cancel out the norm showing that

$$\left\|\hat{h}\right\|_{\mathcal{H}} \leq \frac{2M_y\kappa}{\lambda}. \tag{3.2}$$

Setting $\gamma = \frac{2M_y\kappa}{\lambda}$, we see that if $\lambda = \frac{2M_y\kappa}{\gamma}$ then the output of KRR will be in B_γ. □

Lemma 20 assures us that ifo we pick λ large enough, then the output will be contained within the ball B_γ so that 19 holds and we can assume $\eta_{MMD} = 0$.

3.3 Frank-Wolfe Active Learning algorithm

Considering the bound of the empirical risk over the sampled set P_0 in 17 we see that there are three terms to control, the bias term $\mathcal{E}_{\rho_{S_t}}(h)$, the term representing domain drift, $\text{MMD}_{H'}(P_0^x, S_t^x)$, and the leftover term η_{MMD} which becomes 0 according to 19 by setting $H' = B_{4\gamma^2}$ and $\sigma' = \frac{\sigma}{\sqrt{2}}$.

We focus on the domain drift term and propose to optimise this using FW. Outside of the degenerate case this enables us to do active learning in an online fashion by running the FW algorithm on $\text{MMD}_{H'}(P_0^x, S_t^x)$ which will give a convergence of $O(\frac{1}{t})$ compared to $O(\frac{1}{\sqrt{t}})$ if we were to sample points randomly.

We now state our first active learning algorithm which uses the active learning tuple $(\mathcal{A}, \mathcal{Q})$ where $\mathcal{A}$ is KRR 2.19 with regularisation parameter λ and $\mathcal{Q}$ is the querying strategy that runs FW on P_0^x and chooses the point x_t corresponding to $\tilde{g}_t$ in 2.30. The algorithm follows the recipy of 1 and we call this FW-KRR for Frank-Wolfe Kernel Ridge Regression.

Note that in general we can create ad-hoc active learning algorithms by letting $\mathcal{Q}$ being FW and $\mathcal{A}$ any algorithm. In this way we can create a family of FW-based active learning algorithms. We introduce the following active learning algorithm for classification (which we will call FW-KRRC), let $\mathcal{Q}$ be FW and let $\mathcal{A}$ be the KRR applied to classification through application of the method expanded upon in [16].

3.4 Attempting to Bound the Generalisation Error

With inspiration of error decomposition of the generalisation error in supervised learning we try to bound the following quantity $\mathcal{E}_\rho(\hat{h})$ - $\mathcal{E}_\rho(f)$ where as before $\hat{h} = \mathcal{A}(S)$. We work with Ivanov regularisation instead of Kernel Ridge Regression, where the algorithm is the following

$$\mathcal{A}(S) = \underset{h \in B_\gamma \subseteq \mathcal{H}}{\arg\min} \frac{1}{m} \sum_{j=1}^{m} (h(x_j) - y_j)^2, \tag{3.3}$$

as this can easily be extended to the KRR case.

3.4.1 Error decomposition

We decompose the error as follows

$$\mathcal{E}_\rho(\hat{h}) - \mathcal{E}_\rho(f) = \underbrace{\mathcal{E}_\rho(\hat{h}) - \mathcal{E}_S(\hat{h})}_{(a)} + \underbrace{\mathcal{E}_S(\hat{h}) - \mathcal{E}_S(h_\gamma)}_{\leq 0} + \underbrace{\mathcal{E}_S(h_\gamma) - \mathcal{E}_\rho(h_\gamma)}_{(b)} + \underbrace{\mathcal{E}_\rho(h_\gamma) - \mathcal{E}_\rho(f)}_{\text{Approximation error}},$$

where $h_\gamma = \arg\min_{h \in \mathcal{H}_\gamma} \mathcal{E}(h)$ and $\hat{h} = \mathcal{A}(S)$.

We then try to control each term, first we have that (a) can be decomposed as

$$\mathcal{E}_\rho(\hat{h}) - \mathcal{E}_S(\hat{h}) = \underbrace{\mathcal{E}_\rho(\hat{h}) - \mathcal{E}_{P_0}(\hat{h})}_{\text{Generalization error}} + \underbrace{\mathcal{E}_{P_0}(\hat{h}) - \mathcal{E}_S(\hat{h})}_{\text{Empirical drift generalization error}},$$

where we have called the second quantity *Empirical drift generalisation error* to highlight the fact that we are training on S which is a biased sample drifting from P_0.

Secondly, we can decompose (b) as follows,

$$\mathcal{E}_S(h_\gamma) - \mathcal{E}_\rho(h_\gamma) = \underbrace{\mathcal{E}_S(h_\gamma) - \mathcal{E}_{P_0}(h_\gamma)}_{\text{Empirical drift generalization error}} + \underbrace{\mathcal{E}_{P_0}(h_\gamma) - \mathcal{E}_\rho(h_\gamma)}_{\text{Generalization error}}.$$

We now investigate each of these expressions in turn. We start with the generalisation error for both.

Assume that $h \in B_\gamma$ is any function and consider the following

$$\begin{aligned}
|\mathcal{E}_{P_0}(h) - \mathcal{E}_\rho(h)| &= \left| \frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2 - \mathbb{E}_\rho[(h(x) - y)^2] \right| \\
&= \left| \frac{1}{n}\sum_{i=1}^{n}(h(x_i)^2 - 2h(x_i)y_i + y_i^2) - \mathbb{E}_\rho[h(x)^2 - 2h(x)y + y^2] \right| \\
&= \left| \frac{1}{n}\sum_{i=1}^{n} h(x_i)^2 - \mathbb{E}_{\rho_\mathcal{X}}[h(x)^2] + 2(\mathbb{E}_\rho[h(x)y] - \frac{1}{n}\sum_{i=1}^{n} h(x_i)y_i) + \frac{1}{n}\sum_{i=1}^{n} y_i^2 - \mathbb{E}_{\rho_\mathcal{Y}}[y^2] \right| \\
&\leq \underbrace{\left| \frac{1}{n}\sum_{i=1}^{n} h(x_i)^2 - \mathbb{E}_{\rho_\mathcal{X}}[h(x)^2] \right|}_{(i)} + 2\underbrace{\left| \frac{1}{n}\sum_{i=1}^{n} h(x_i)y_i - \mathbb{E}_\rho[h(x)y] \right|}_{(ii)} + \underbrace{\left| \frac{1}{n}\sum_{i=1}^{n} y_i^2 - \mathbb{E}_{\rho_\mathcal{Y}}[y^2] \right|}_{(iii)}.
\end{aligned}$$

Using the reproducing property of 3 we can express $f(x) = \langle f, K_x \rangle_\mathcal{H}$ for any $x \in \mathcal{X}$. Consider each term individually

(i)

$$\begin{aligned}
\left|\frac{1}{n}\sum_{i=1}^{n} h(x_i)^2 - \mathbb{E}_{\rho_{\mathcal{X}}}[h(x)^2]\right| &= \left|\frac{1}{n}\sum_{i=1}^{n} \langle h, \phi(x_i)\rangle_{\mathcal{H}}^2 - \mathbb{E}_{\rho_{\mathcal{X}}}[\langle h, \phi(x)\rangle_{\mathcal{H}}^2]\right| \\
&= \left|\frac{1}{n}\sum_{i=1}^{n} \langle h, \langle h, \phi(x_i)\rangle_{\mathcal{H}}\, \phi(x_i)\rangle_{\mathcal{H}} - \mathbb{E}_{\rho_{\mathcal{X}}}[\langle h, \langle h, \phi(x)\rangle_{\mathcal{H}}\, \phi(x)\rangle_{\mathcal{H}}]\right| \\
&= \left|\frac{1}{n}\sum_{i=1}^{n} \langle h, (\phi(x_i)\otimes\phi(x_i))h\rangle_{\mathcal{H}} - \mathbb{E}_{\rho_{\mathcal{X}}}[\langle h, (\phi(x)\otimes\phi(x))h\rangle_{\mathcal{H}}]\right| \\
&= \left|\left\langle h, \frac{1}{n}\sum_{i=1}^{n}(\phi(x_i)\otimes\phi(x_i))h\right\rangle_{\mathcal{H}} - \langle h, \mathbb{E}_{\rho_{\mathcal{X}}}[(\phi(x)\otimes\phi(x))]h\rangle_{\mathcal{H}}\right| \\
&= \left|\left\langle h, (\hat{C}_{P_0^x} - C_{\rho_{\mathcal{X}}})h\right\rangle_{\mathcal{H}}\right| \\
&= \|h\|_{\mathcal{H}}^2 \left\|\hat{C}_{P_0^x} - C_{\rho_{\mathcal{X}}}\right\|_{op},
\end{aligned}$$

where $C_{\rho_{\mathcal{X}}}, \hat{C}_{P_0^x}$, are the covariate and empirical covariate operator in $\mathcal{H}$ for $\rho_{\mathcal{X}}$. Since $\hat{C}_{P_0^x}, C_{\rho_{\mathcal{X}}} \in HS(\mathcal{H})$, which follows as $\kappa < \infty$, the operator norm can be bounded by the HS (Frobenius) norm

$$\left\|\hat{C}_{P_0^x} - C_{\rho_{\mathcal{X}}}\right\|_{op} \leq \left\|\hat{C}_{P_0^x} - C_{\rho_{\mathcal{X}}}\right\|_{HS}. \tag{3.4}$$

We will now use the following lemma

Lemma 21 ([61], Lemma 1)**.** *Suppose that $\kappa^2 < M$. Given any $\rho_{\mathcal{X}}$ let $\{x_i\}_{i=1}^n$ be a set sampled iid from $\rho_{\mathcal{X}}$. If $C, \hat{C}_n$ are the true and empirical covariance operators, then with probability greater than $1-\delta$,*

$$\left\|\hat{C}_n - C\right\|_{HS} \leq \frac{2M}{\sqrt{n}}\left(1 + \sqrt{\frac{\log(1/\delta)}{2}}\right). \tag{3.5}$$

using 21 together with 3.4 we have

Corollary 22. *With probability greater than $1-\delta$,*

$$\left\|\hat{C}_{P_0^x} - C_{\rho_{\mathcal{X}}}\right\|_{op} \leq \frac{2\kappa^2}{\sqrt{n}}\left(1 + \sqrt{\frac{\log(1/\delta)}{2}}\right) \tag{3.6}$$

(ii)

$$\begin{aligned}
\left|\frac{1}{n}\sum_{i=1}^{n} h(x_i)y_i - \mathbb{E}_{\rho}[h(x)y]\right| &= \left|\frac{1}{n}\sum_{i=1}^{n} \langle h, \phi(x_i)\rangle\, y_i - \mathbb{E}_{\rho}[\langle h, \phi(x)\rangle\, y]\right| \\
&= \left|\left\langle h, \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)y_i - \mathbb{E}_{\rho}[\phi(x)y]\right\rangle\right|
\end{aligned}$$

Let $z_{P_0} = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)y_i$ and $z_\rho = \mathbb{E}_\rho[\phi(x)y]$. Applying the CS inequality, this

leads to

$$\left|\left\langle h, \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)y_i - \mathbb{E}_\rho[\phi(x)y]\right\rangle\right| = \|h\|_{\mathcal{H}}\|z_{P_0} - z_\rho\|_{\mathcal{H}}$$

The following lemma will let us control the norm of $z_{P_0} - z_\rho$,

Lemma 23 ([50], Lemma 2)**.** *let $\mathcal{H}$ be a Hilbert space and let ξ be a random variable on $(\mathcal{X}\times\mathcal{Y}, \rho)$, with values in $\mathcal{H}$. Assume $\|\xi\|_{\mathcal{H}} \leq M < \infty$ almost surely. Denote $\sigma^2(\xi) = \mathbb{E}[\|\xi\|^2_{\mathcal{H}}]$. Let $\{z_i\}_{i=1}^n$ be independent random draws of ρ. For any $0 < \delta < 1$, with probability $1-\delta$ over the draws,*

$$\left\|\frac{1}{n}\sum_{i=1}^{n}\xi(z_i) - \mathbb{E}[\xi(z)]\right\|_{\mathcal{H}} \leq \frac{2M\log(2/\delta)}{n} + \sqrt{\frac{2\sigma^2(\xi)\log(2/\delta)}{n}} \tag{3.7}$$

We will use 23 in order to bound the quantity $\|z_{S_0} - z_\rho\|_{\mathcal{H}}$ in probability. By assumption 4 $\sup_{y\in\mathcal{Y}}|y| = M_y < \infty$ which means that $z_i = y_i\phi(x_i)$ will be an element of $\mathcal{H}$. These z's will correspond to the ξ's of lemma 23. We can bound them through

$$\begin{aligned}
\|z\|_{\mathcal{H}} &\leq \|y\phi(x)\|_{\mathcal{H}} \\
&\leq \|y\|\|\phi(x)\|_{\mathcal{H}} \\
&\leq M_y\sqrt{\langle\phi(x),\phi(x)\rangle_{\mathcal{H}}} \\
&\leq M_y\sqrt{K(x,x)} \\
&\leq M_y\sup_{x\in\mathcal{X}}\sqrt{K(x,x)} \\
&\leq M_y\kappa,
\end{aligned}$$

hence we can identify M in the lemma with $M_y\kappa$.

We also have that

$$\begin{aligned}
\sigma^2 &= \mathbb{E}[\|y\phi(x)\|^2_{\mathcal{H}}] \\
&= \mathbb{E}[|y|^2\langle\phi(x),\phi(x)\rangle_{\mathcal{H}}] \\
&= \mathbb{E}[|y|^2K(x,x)] \\
&\leq M_y^2\kappa^2 \\
&< \infty.
\end{aligned}$$

Finally as P_0 is a set of input-output pairs sampled *iid* from ρ we have that the $z_i = y_i\phi(x_i)$ are sampled *iid* as well. Thus all conditions of the lemma 23 are satisfied.

Thus we have that for any $\delta\in(0,1)$, with probability $1-\delta$ the following holds

$$\|z_{P_0} - z_\rho\|_{\mathcal{H}} \leq \frac{2M_y\kappa\log(2/\delta)}{n} + \sqrt{\frac{2M_y^2\kappa^2\log(2/\delta)}{n}} = M_y\kappa\left(\frac{2\log(2/\delta)}{n} + \sqrt{\frac{2\log(2/\delta)}{n}}\right). \tag{3.8}$$

(iii)

$$\left|\frac{1}{n}\sum_{i=1}^{n} y_i^2 - \mathbb{E}_{\rho_{\mathcal{Y}}}[y^2]\right|$$

We can bound this using assumption 4 which says that any Y is such that $y \leq M_y$. First we introduce Hoeffding's inequality,

Theorem 24 ([15], Hoeffdings Inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [a_i, b_i]$ with probability 1. Let $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then*

$$\Pr\left(\left|\overline{X} - \mathbb{E}[\overline{X}]\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{2n^2\epsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right). \tag{3.9}$$

Applying 24 to the random variables $\{y_i\}_{i=1}^n$ which are sampled *iid* and using that $|y|^2 \leq M_y^2$ we have that for any $\delta \in (0,1)$, with probability at least $1-\delta$ over the samples the following holds

$$\left|\frac{1}{n}\sum_{i=1}^{n} y_i^2 - \mathbb{E}_{\rho_y}[y^2]\right| \leq \sqrt{\frac{M_y^4 \log(2/\delta)}{2n}} = M_y^2\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{3.10}$$

Combining $(i), (ii)$ and (iii) we proceed to bound this uniformly using the assumption that $h \in B_\gamma$,

$$\begin{aligned}\sup_{\|h\|_{\mathcal{H}}\leq\gamma} |\mathcal{E}_{P_0}(h) - \mathcal{E}_\rho(h)| &\leq \sup_{\|h\|_{\mathcal{H}}\leq\gamma} \|h\|_{\mathcal{H}}^2 \left\|\hat{C}_{S_0^x} - C_{\rho_{\mathcal{X}}}\right\|_{op} + \|h\|_{\mathcal{H}}\left\|z_{S_0^x} - z_\rho\right\|_{\mathcal{H}} + \left|\frac{1}{n}\sum_{i=1}^{n} y_i^2 - \mathbb{E}_{\rho_{\mathcal{Y}}}[y^2]\right| \\ &= \gamma^2\left\|\hat{C}_{S_0^x} - C_{\rho_{\mathcal{X}}}\right\|_{op} + \gamma\left\|z_{S_0^x} - z_\rho\right\|_{\mathcal{H}} + \left|\frac{1}{n}\sum_{i=1}^{n} y_i^2 - \mathbb{E}_{\rho(y)}[y^2]\right|.\end{aligned}$$

Combining this with results 3.6, 3.8, 3.10 we have that with probability $1-3\delta$ over P_0 we have that

$$\sup_{\|h\|_{\mathcal{H}}\leq\gamma} |\mathcal{E}_{P_0}(h) - \mathcal{E}(h)| \leq \frac{2\kappa^2\gamma^2}{\sqrt{n}}\left(1 + \sqrt{\frac{\log\frac{1}{\delta}}{2}}\right) + \frac{2M_y\gamma\kappa\log(2/\delta)}{n} + \sqrt{\frac{2M_y^2\kappa^2\log(2/\delta)}{n}} + M_y^2\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{3.11}$$

Since $\hat{h}, h_\gamma \in B_\gamma$ we have bounded both of the generalisation terms in $(a), (b)$.

3.4.2 Bounding in probability

From 3.1 we have that for $S \subseteq P_0$ and letting $\rho_{S^x} = \frac{1}{m}\sum_{x\in S^x}\delta_x$, both of the empirical drift generalisation errors in $(a), (b)$ can be bounded above by $\text{MMD}_{B_\eta}(P_0^x, S^x) + \eta_{MMD}$, and from 19 if we let $\eta = 4\gamma^2$ and assume that $f \in B_\gamma$ then $\eta_{MMD} = 0$. We will choose K' such that 19 holds.

Combining this we have that

Theorem 25. *Given that $f \in B_\gamma$, $\eta = 4\gamma^2$ and for any $x, x' \in \mathcal{X}$, $K'(x,x') = K(x,x')^2$ the following bound holds for $P_0 \sim \rho^n$ and any $S \subset P_0$*

$$\mathcal{E}_\rho(\hat{h}) - \mathcal{E}_\rho(f) \leq 2 \sup_{\|h\| \leq \gamma} |\mathcal{E}_\rho(h) - \mathcal{E}_{P_0}(h)| + 2\mathrm{MMD}_{B_\eta}(P_0, S). \tag{3.12}$$

Proof. This is simply an application of the results developed in rror Decomposition]. We have the following

$$\begin{aligned}
\mathcal{E}_\rho(\hat{h}) - \mathcal{E}_\rho(f) &= \mathcal{E}_\rho(\hat{h}) - \mathcal{E}_S(\hat{h}) + \underbrace{\mathcal{E}_S(\hat{h}) - \mathcal{E}_S(h_\gamma)}_{\leq 0} + \mathcal{E}_S(h_\gamma) - \mathcal{E}_\rho(h_\gamma) + \mathcal{E}_\rho(h_\gamma) - \mathcal{E}_\rho(f) \\
&\leq \mathcal{E}_\rho(\hat{h}) - \mathcal{E}_S(\hat{h}) + \mathcal{E}_S(h_\gamma) - \mathcal{E}_\rho(h_\gamma) + \mathcal{E}_\rho(h_\gamma) - \mathcal{E}_\rho(f) \\
&\leq \left| \mathcal{E}_\rho(\hat{h}) - \mathcal{E}_S(\hat{h}) + \mathcal{E}_S(h_\gamma) - \mathcal{E}_\rho(h_\gamma) + \mathcal{E}_\rho(h_\gamma) - \mathcal{E}_\rho(f) \right| \\
&\leq \left| \mathcal{E}_\rho(\hat{h}) - \mathcal{E}_S(\hat{h}) \right| + |\mathcal{E}_S(h_\gamma) - \mathcal{E}_\rho(h_\gamma)| + \underbrace{|\mathcal{E}_\rho(h_\gamma) - \mathcal{E}_\rho(f)|}_{=0} \\
&\leq 2 \left(\sup_{\|h\| \leq \gamma} |\mathcal{E}_\rho(h) - \mathcal{E}_{P_0}(h)| + \mathrm{MMD}_{B_\eta}(P_0, S) \right),
\end{aligned}$$

where in the final line we have used the bounds on the generalisation and empirical drift generalisation error. □

From the first result in the table 2.1 we can bound the quantity $\mathrm{MMD}_{B_\eta}(P_0^x, S^x)$ as follows, let S_t be the set built from P_0 by running FW on $\frac{1}{2}\left\|\mu_{P_0^x} - \mu_{S_t^x}\right\|^2_{\mathcal{H}'}$ for t iterations, then

$$\begin{aligned}
\mathrm{MMD}_{B_\eta}(P_0^x, S_t^x) &= \sqrt{2\eta \frac{1}{2}\left\|\mu_{P_0^x} - \mu_{S_t^x}\right\|^2_{\mathcal{H}'}} \\
&\leq \sqrt{2\eta \frac{4R^2}{t}}.
\end{aligned}$$

When using the Gaussian kernel, $R = 1$ and the bound reduces to $\mathrm{MMD}_{B_\eta}(P_0^x, S_t^x) = 2\sqrt{\frac{2\eta}{t}}$. For the Gaussian kernel case we have the following,

Theorem 26. *In addition to Assumptions 2 to 6, assume that $\eta = 4\gamma^2$, $K'(x,x') = K(x,x')^2$ and that S_t is built using FW, $\hat{h} = \mathcal{A}(S_t)$ is Ivanov Kernel Regression. Then with probability greater than $1 - (3+2)\delta) = (1 - 5\delta)$ the following bound holds*

$$\begin{aligned}
\mathcal{E}_\rho(\hat{h}) - \mathcal{E}_\rho(f) \leq \frac{4\gamma^2}{\sqrt{n}} \left(1 + \sqrt{\frac{\log(1/\delta)}{2}} \right) + \frac{4M_y \gamma \log(2/\delta)}{n} + 2\sqrt{\frac{2M_y^2 \log(2/\delta)}{n}} + 2M_y^2 \sqrt{\frac{\log(2/\delta)}{2n}} \\
+ 4\sqrt{\frac{2\eta}{t}}
\end{aligned}$$

While this is a bound, it is not an improvement over random (MC) sampling.

Consider a sampling strategy that uniformly at random picks datapoints in P_0, this is the same as doing uniformly sampling from $poolset_0$ without replacement. Since the set S_t^x is now an empirical iid sample from S_0^x we have that $\mathrm{MMD}_{\mathcal{H}'}(P_0^x, S_t^x) = O_P(\frac{1}{\sqrt{t}})$ [51] which is of the same order as using FW. This is the baseline we would like to improve upon.

3.4.3 Lower bounding the radius

In order to have a generalisation bound which is an improvement over MC sampling we would have to show that in probablity we can lower bound the quantity 2.33 so that we would have a convergence rate of the second entry of table 2.1. We proceed as follows, let $A = \{\alpha \succeq 0, \|\alpha\|_1 = 1, \|\alpha\|_0 < n\} \subseteq \mathbb{R}^n$, the faces of the simplex $\triangle_{n-1}$, then the radius d as a function of P_0^x can be bounded as follows

$$\begin{aligned} d(P_0^x)^2 &= \min_{\varphi \in \partial_{relint}\mathcal{M}} \left\|\mu_{P_0^x} - \varphi\right\|_{\mathcal{H}}^2 \\ &= \min_A \left\| \frac{1}{n}\sum_{i=1}^n \phi(x_i) - \sum_{i=1}^n \alpha_i \phi(x_i) \right\|_{\mathcal{H}}^2 \\ &= \min_A \left\| \mathbf{K}_n^{1/2}(\mathbf{1}/n - \alpha) \right\|_2^2 \\ &\geq \lambda_{min}(\mathbf{K}_n) \min_A \|\mathbf{1}/n - \alpha\|_2^2. \end{aligned}$$

We first consider the term $\min_A \|\mathbf{1}/n - \alpha\|_2^2$. Note that this is just the minimum of the distances from the barycenter $\mathbf{1}/n$ onto any of the faces of $\triangle_{n-1}$. Now, the simplex is symmetric, so the result will be the same up to permutation of what entries in α that is set to zero, so without loss of generality we will denote $\alpha^k = (\alpha_1, \ldots, \alpha_k, 0, \ldots, 0)^T \in A$. Let F_k be the face generated by k non-zero vectors α^k, then $F_k \subseteq F_l$ when $k < l$. Thus, we can write

$$\min_{F_k, k \in [n-1]} \|\mathbf{1}/n - \alpha\|_2^2 = \min_{F_{n-1}} \|\mathbf{1}/n - \alpha\|_2^2 \tag{3.13}$$

where we used the fact that for any $k < n$, $F_k \subseteq F_{n-1}$ and so we only have to minimise over F_{n-1} which is a convex set.

We claim that the optimum over the affine hull of F_{n-1} of 3.13 is attained on F_{n-1}, that is $\left\|\mathbf{1}/n - \alpha^{n-1}\right\|_2^2$ can be optimised over $A_{n-1} := \mathrm{AffHull}(F_{n-1})$. To see this consider the family of vectors $\beta^{n-1} = (\beta_1, \ldots, \beta_{n-1}, 0)^T$ where each $\beta_i \in \mathbb{R}$ and $\sum_{i=1}^{n-1} \beta_i = 1$, then we see that

$$\min_{\beta \in A_{n-1}} \|\mathbf{1}/n - \beta\|_2^2 = \min_{\beta \in A_{n-1}} \sum_{i=1}^{n-1} (n^{-1} - \beta_i)^2 + n^{-1}. \tag{3.14}$$

We can solve this using lagrange multiplier. The corresponding lagrangian to

3.14 is

$$\mathcal{L}(\beta,\gamma) = \sum_{i=1}^{n-1}(n^{-1}-\beta_i)^2 + \gamma(\sum_{i=1}^{n-1}\beta_i - 1)$$

and using the KKT conditions, we can replace the primal variables β. Consider,

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\beta_i} &= 2(\beta_i - n^{-1}) + \gamma \\ &= 0\end{aligned}$$

which means that we can set $\beta_i = n^{-1} - \frac{1}{2}\gamma$. We then have the following

$$\begin{aligned}\mathcal{L}(\gamma) &= \sum_{i=1}^{n-1}(\frac{1}{2}\gamma)^2 + \gamma(\sum_{i=1}^{n-1}(n^{-1}-\gamma) - 1) \\ &= \frac{1}{4}(n-1)\gamma^2 + \gamma((1-n^{-1}) - \frac{n-1}{2}\gamma - 1) \\ &= \frac{1}{4}(n-1)\gamma^2 - \frac{1}{2}(n-1)\gamma^2 - n^{-1}\gamma \\ &= -\frac{1}{4}(n-1)\gamma^2 - n^{-1}\gamma.\end{aligned}$$

This expression has the maximum at $\gamma^* = -\frac{2}{n(n-1)}$ which gives that each $\beta_i = n^{-1} + n^{-1}(n-1)^{-1} = \frac{1}{n-1}$. It's clear that this solution is in F_{n-1} and hence we have solved the problem. Substituting this in for the α in 3.13, we have that

$$\begin{aligned}\min_{F_{n-1}} \|\mathbf{1}/n - \alpha\|_2^2 &= \sum_{i=1}^{n-1}(n^{-1} - (n-1)^{-1})^2 + n^{-2} \\ &= (n-1)^{-1}n^{-2} + n^{-2} \\ &= n^{-2}(1 + (n-1)^{-1}) \\ &= \frac{1}{n(n-1)} \\ &> \frac{1}{n^2}\end{aligned}$$

We have thus shown that we can bound

$$d(P_0^x) \geq \frac{\sqrt{\lambda_{min}(\mathbf{K}_n)}}{n}. \tag{3.15}$$

Unfortunately this bound is vacuous as for this to be of relevance to us we require $\frac{\sqrt{\lambda_{min}(\mathbf{K}_n)}}{n} \geq c > 0$ for some scalar c. This would mean that $\lambda_{min}(\mathbf{K}_n) = \Omega(n^2)$ which does not go well with well-established assumptions on decay of the eigenvalues of the kernel matrix, for example that $\text{tr}(\mathbf{K}_n) = \Theta(n)$ according to [3].

Chapter 4

Experiments

4.1 Setup

As before, we will use a gaussian kernel $K_\sigma(x, x') = \exp\left(-\frac{\|x-x'\|_2^2}{2\sigma^2}\right)$. We split the experiments into *agnostic* and *realisable*. All datasets were normalised in both input and output over the whole dataset. We produce learning curves on the test set in order to analyse the generalisation performance of the active learning algorithms.

For agnostic we do the following: D is the total dataset Due to computational feasibility, we have a threshold n_{sub}, if the size of D is greater than this, we randomly subsample n_{sub} datapoints from D and let D be this smaller dataset. D is then split into two sets, $S_{train+test}$ (80%) and S_{val} (20%). We use S_{val} to do 5-fold cross validation (kCV) and get optimal hyperparamters $\sigma_{opt}, \lambda_{opt}$ for algorithm $\mathcal{A}$ which will be KRR.

The dataset $S_{train+test}$ is split into 5 folds $[S_1, \ldots, S_5]$ and for each $i \in \{1, \ldots, 5\}$ a train set is chosen by letting $S_{train} = S_i$, and test set $S_{test} = S_{train+test} \setminus S_{train}$. The train set will correspond to P_0. $S_0 = \emptyset$ and for the first iteration, $t = 1$, we add the first instance $x_1 \in P_0$ so that $S_1 = \{(x_1, y_1)\}$ and after this we use the querying strategy $\mathcal{Q}$ at each $t > 1$ to choose what instance to query[1]. At each time t, after querying an instance, we fit the algorithm $\mathcal{A}$ to the current train set S_t giving us a hypothesis $\hat{h}_t$ with which we get the current loss $l_t = \mathcal{E}_{S_{test}}(\hat{h}_t)$ where the loss will be the squared loss for regression (MSE) and the zero-one loss for classification (Accuracy).

For the realisable setting, we proceed almost in the same way. The only difference is that after potentially subsampling D, we get the optimal hyperparameters doing 5-fold kCV on the dataset D, fit a ground-truth hypothesis using KRR with these hyperparameters, which we'll call $h_{realisable}$. Finally we replace the outputs of $D = \{(x_i, y_i)_{i=1}^n\}$ with $h_{realisable}(x_i)$. This will ensure that the assumptions of no noise and realisability are satisfied. The rest is the same as in the agnostic setting.

We compare the FW (KH, step-size $\rho_t = \frac{1}{1+t}$) algorithm with random subsampling (MC) and Leverage Score sampling (Levscore (deterministic)) [45], where we

[1] This is necessary as the algorithms require an instance for the updates to be well-defined.

calculate the leverage scores and pick the instances in descending order of the leverage scores (we choose the λ of the leverage scores to be the same as λ_{opt} gotten through kCV).

4.2 Datasets

We evaluate the algorithms on 5 regression datasets and 2 classification datasets. These were sourced from the UCI dataset repository [26] and through the sklearn dataset API [26]. Some of the datasets were changed to remove non-continuous variables.

4.3 Analysis

In this section we will analyse a subset of the learning curves produced. All of the learning curves can be found in the appendix for the agnostic regression, realisable regression and agnostic classification. The rest of the learning curves are similar and do not retract from the analysis in any significant way.

Since there is no standardised way on how to evaluate the performance of active learning algorithms against each other empirically, we will analyse the curves directly.

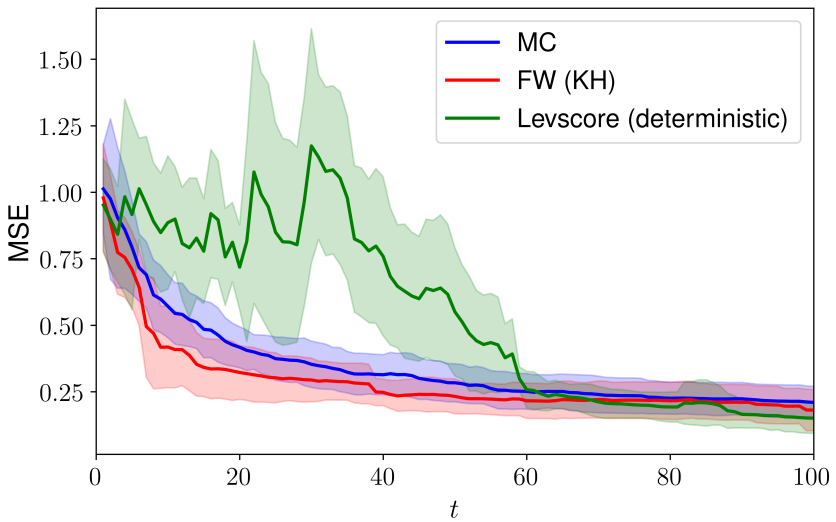


Figure 4.1: Learning curve (bold line is mean with shaded region being ± 1 standard deviation over the 5 folds) for agnostic regression on Boston dataset comparing FW (KH) with MC and Levscore. The y-axis is MSE (Mean Squared Error), x-axis is t (number of datapoints in current active learning train set). A good algorithm will have a trajectory that goes down quickly with t. MC is baseline.

For agnostic regression (here the Boston dataset), in general we see that FW (KH) performs well, going down quickly in the start and outperforming MC (in mean) at all t. Levscore performs poorly for the first 50 t which can be explained by the fact that it does not take into account the already queried instances, only focusing on those with high leverage. This means that while it focuses on *informativeness* of the instances chosen, it does not implement a way to choose *representative* instances.

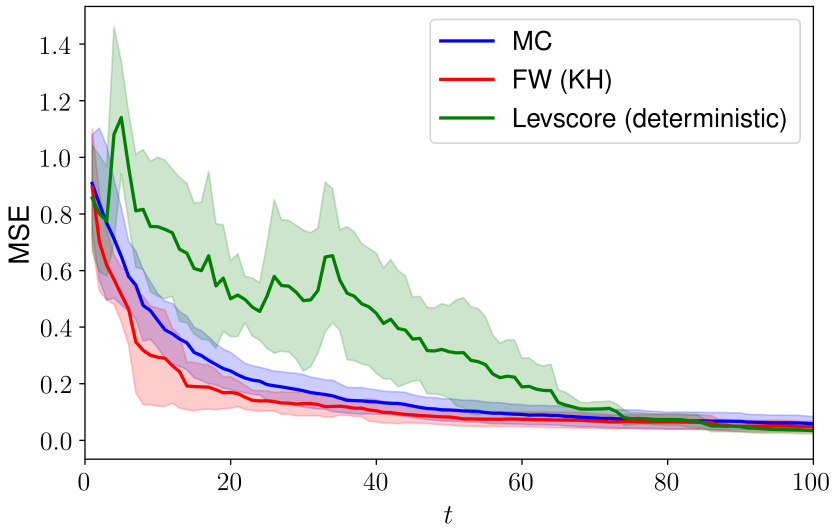


Figure 4.2: Learning curve (bold line is mean with shaded region being ±1 standard deviation over the 5 folds) for realisable regression on Boston dataset comparing FW (KH) with MC and Levscore. The y-axis is MSE (Mean Squared Error), x-axis is t (number of datapoints in current active learning train set). A good algorithm will have a trajectory that goes down quickly with t. MC is baseline.

Nevertheless it performs well towards the end of the plot. MC goes down slowly which is what you would expect, since it does not care about informativeness but only about representativeness. FW (KH) arguably care about both as it chooses instances in such a way that the set S_t^x resembles P_0^x which means that it will focus early on building a train set that contains the instances that can explain the statistical properties of P_0^x. The analysis for the realisable setting (as in here for Boston dataset) is essentially the same, although we see that the curves goes towards 0 which is to be expected since now the function f is in the hypothesis class which is not true for the agnostic case.

For agnostic classification (here the mnist dataset) we see that FW (KH) does a very good job at choosing good instances. The red curve outperforms MC and Levscore essentially everywhere except for in the very beginning. Levscores poor performance can be explained that it is mainly a method to choose instances that has high leverage for regression, but this does not apply when doing classification. An explanation for the good performance of FW (KH) is that it is practically *mode seeking* as can be seen in this figure.

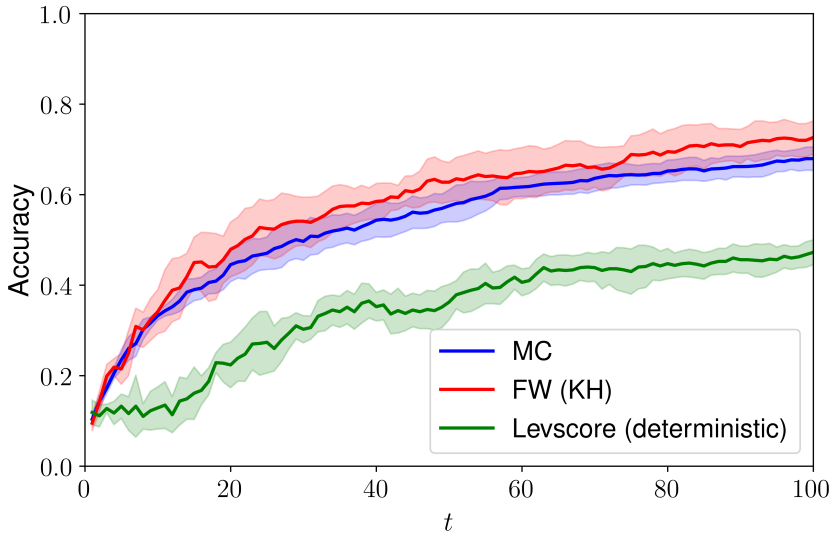


Figure 4.3: Learning curve (bold line is mean with shaded region being ± 1 standard deviation over the 5 folds) for agnostic classification on mnist comparing FW (KH) with MC and Levscore. The y-axis is Accuracy, x-axis is t (number of datapoints in current active learning train set). A good algorithm will have a trajectory that goes up quickly with t. MC is baseline.

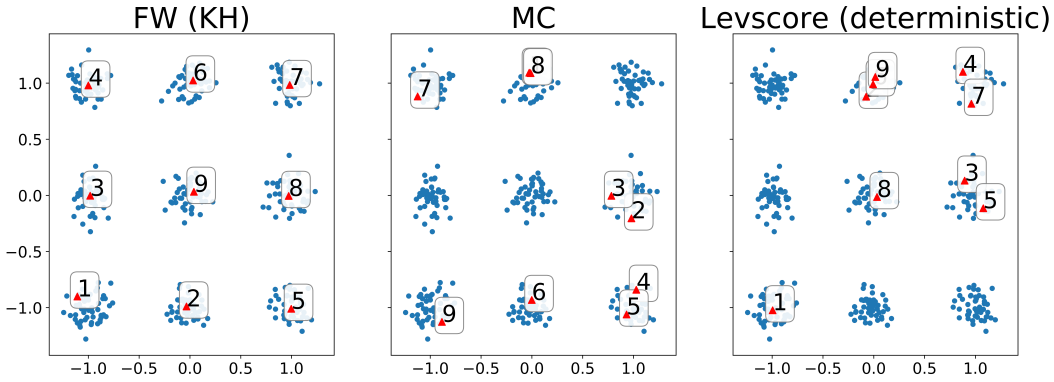


Figure 4.4: Comparison of the first 9 instances chosen by FW (KH), MC and Levscore. The data was generated by sampling 50 datapoints from 9 gaussian distributions with means from the set $\{(i,j)|i,j \in \{-1,0,1\}\}$ and covariance matrix being the identity matrix scaled by 0.01. The kernel matrix was created by using a gaussian kernel with $\sigma^2 = 0.02$ which was fine-tuned to exhibit this phenomenon.

Chapter 5

Conclusions and future directions

In this dissertation we have done the following; Introduced the field of active learning, where we produced an overview of the field in general before diving deep into the current work on active learning for regression using RKHS theory, statistical learning theory and optimisation in the form of kernel herding and frank Wolfe. Since the work married ideas from different fields we also provided a review of the necessary concepts of RKHS theory from kernels to MMD and the necessary concepts for supervised learning from a statistical learning point of view.

From this we identified work in the field of active learning regression upon which we base our work and relate it to. Using an error bound which split up into a data-fitting term and a domain drift term in the form of the MMD between the original sampled dataset and the build train set, we applied FW to this in order to optimise it, which guarantees a theoretical convergence which dominates random sampling.

We decompose the generalisation error of the estimator of the active learning algorithm and derived an upper bound on this which holds with high probability. However, given the current work, we show that this do not yield an improvement over random sampling, however this might change in the future, since the conditions given for FW to converge faster are sufficient, but not necessary.

We investigate the performance empirically and see that FW in the form of KH does very well when compared to the random sampling baseline (MC) and against leverage scores which is a commonly used method in regression for subsampling a dataset or doing dimensionality reduction. We show that this holds both for the realisable regression case where we can control the complexity term in the empirical bound by making it zero, but also in the agnostic regression and classification. For classification this can potentially be explained by KH being mode seeking.

We conclude that KH and in general FW is a competitive way of choosing which instance to label for active learning and that it has good properties whenever there is an MMD bound to optimise. They have a convergence rate which is provably faster over random sampling which enables us to reduce the empirical risk fast, and work well in practice on both regression and classification. This is as far as we are aware

the only place where the theoretical properties of FW has been used to show in theory how FW can give guarantees when optimising this empirical error bound.

There are various avenues for further research. It would be fruitful to investigate the setting of output noise and extend the framework to see how FW can be leveraged. In addition to this it'd also be interesting and helpful to consider losses other than the squared error loss which would require taking the output of the algorithm into account, making it adaptive. Finally, I believe there is a potential direction of using dimensionality reduction together with active learning akin to what is done in [45] in order to reduce the complexity of the training and prediction of KRR, meaning we could apply the active learning algorithm to bigger datasets without the need to subsample.

Chapter 6

Appendix

6.1 Proofs

6.2 Convex Analysis

We introduce the necessary concepts from convex analysis used in the analysis of convergence of KH and FW. We will follow [9].

Definition 13. *The Convex Hull of a set $\mathcal{C}$ is denoted by ConvHull(C) and is defined as*

$$ConvHull(\mathcal{C}) = \{\sum_{i\in I}^{n} \theta_i x_i | \theta_i \geq 0, \forall i \in I, \sum_{i\in I} \theta_i = 1\} \tag{6.1}$$

where I is an index set of elements in $\mathcal{C}$.

Definition 14. *The Affine Hull of a set $\mathcal{C}$ is denoted by AffHull(C) and is defined as*

$$AffHull(\mathcal{C}) = \{\sum_{i\in I}^{n} \theta_i x_i | \theta_i \in \mathbb{R}, \forall i \in I, \sum_{i\in I} \theta_i = 1\} \tag{6.2}$$

where I is an index set of elements in $\mathcal{C}$.

Definition 15. *The Relative Interior of a set $\mathcal{C}$ is denoted by RelInt(C) and is defined as*

$$RelInt(\mathcal{C}) = \{x \in \mathcal{C} | B_r(x) \cup AffHull(\mathcal{C}) \subseteq \mathcal{C}, \text{ for some } r \geq 0\} \tag{6.3}$$

Definition 16. *The Relative boundary of a set $\mathcal{C}$ is denoted by $\partial_{RelInt}\mathcal{C}$ and is defined as*

$$\partial_{RelInt}\, \mathcal{C} = Clos(\mathcal{C}) \setminus RelInt(\mathcal{C}) \tag{6.4}$$

where Clos($\mathcal{C}$) is the closure of $\mathcal{C}$.

6.3 Experiments

6.3.1 Procedures

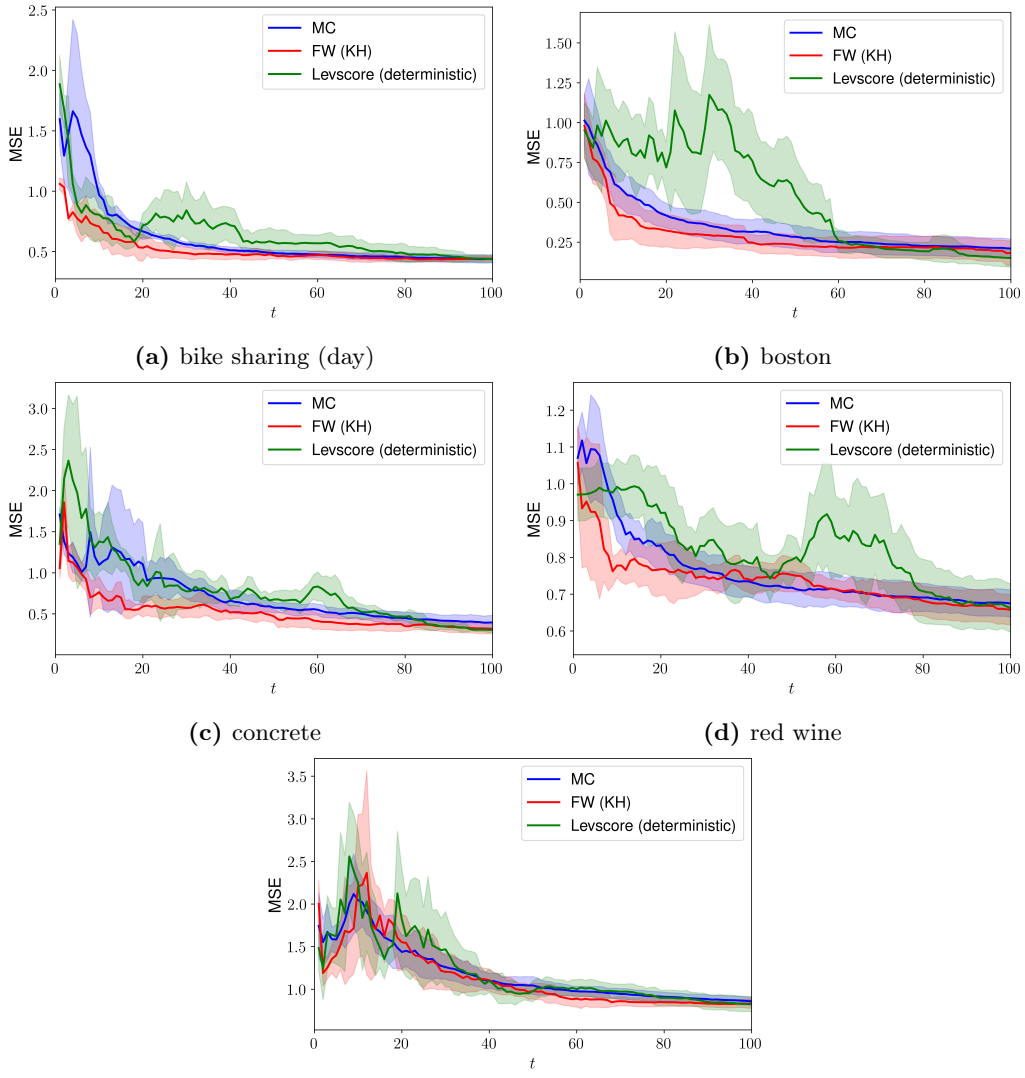
Algorithm 4 Experimental procedure (Agnostic)

procedure EXPERIMENT(D, $\mathcal{A}$, $\mathcal{Q}$, ℓ, k_{cv}, k_ℓ, n_{sub}, n_{val}) ▷ D is the dataset, $\mathcal{A}$ is the algorithm, $\mathcal{Q}$ is the querying strategy, k_{cv} is the number of folds in kCV, k_l is the number of folds over $D_{train+test}$ that we run to get k_l learning curves, n_{sub} is threshold for subsampling and n_{val} is the size of the validation split.
  $\mathbf{L} \leftarrow \mathbf{0}_{k,n}$ ▷ k_l by n empty matrix for losses for timesteps $t = 1 : n$ for folds $i = 1 : k_l$
  $n_D \leftarrow |D|$
  if $n_D > n_{sub}$ **then** ▷ If dataset too big, subsample
    $D \leftarrow \text{subsample}(D, n_{sub})$
    $n_D \leftarrow |D|$
  end if
  $S_{val} \leftarrow D_{1:n_{val}}$
  $\sigma_{opt}, \lambda_{opt} \leftarrow kCV(S_{val}, \mathcal{A}, k_{cv})$ ▷ Get optimal hyperparameters through k-fold CV
  $S_{train+test} \leftarrow S_{n_{val}+1:n}$
  $\sigma'_{opt} \leftarrow \frac{\sigma_{opt}}{\sqrt{2}}$ ▷ Set σ'_{opt} according to 19
  $[S_1, S_2, \ldots, S_k] \leftarrow S_{train+test}$ ▷ Split train / test data in k folds
  for $i = 1 : k_l$ **do** ▷ Get learning curve for fold i
    $S_{test} \leftarrow S_i$
    $P_0 \leftarrow S_{train+test} \setminus S_{test}$
    $n \leftarrow |P_0|$
    for $t = 1 : n$ **do** ▷ Run active learning algorithm
      $x_t^q \leftarrow \mathcal{Q}(P_{t-1}^x)$
      $y_t^q \leftarrow \mathcal{O}(x_t^q)$
      $S_t \leftarrow S_{t-1} \cup \{(x_t^q, y_t^q)\}$
      $P_t \leftarrow P_{t-1} \setminus \{(x_t^q, y_t^q)\}$
      $\hat{h}_t \leftarrow \mathcal{A}(S_t)$
      $\mathbf{L}_{i,t} \leftarrow \mathcal{E}_{S_{test}}(\hat{h}_t)$ ▷ Fill out loss matrix for current fold and step
    end for
  end for
end procedure

Algorithm 5 Experimental procedure (Realisable)

procedure EXPERIMENT(D, $\mathcal{A}$, $\mathcal{Q}$, ℓ, k_{cv}, k_ℓ, n_{sub}, n_{val}) ▷ D is the dataset, $\mathcal{A}$ is the algorithm, $\mathcal{Q}$ is the querying strategy, k_{cv} is the number of folds in kCV, k_l is the number of folds over $D_{train+test}$ that we run to get k_l learning curves, n_{sub} is threshold for subsampling and n_{val} is the size of the validation split.
  $\mathbf{L} \leftarrow \mathbf{0}_{k,n}$ ▷ k_l by n empty matrix for losses for timesteps $t = 1 : n$ for folds $i = 1 : k_l$
  $n_D \leftarrow |D|$
  if $n_D > n_{sub}$ **then** ▷ If dataset too big, subsample
    $D \leftarrow \text{subsample}(D, n_{sub})$
    $n_D \leftarrow |D|$
  end if
  $\sigma_{opt}, \lambda_{opt} \leftarrow kCV(D, \mathcal{A}, k_{cv})$ ▷ Get optimal hyperparameters through k-fold CV
  $h_{realisable} = \mathcal{A}(D)$ ▷ Fit new hypothesis and create new labels
  $D \leftarrow \{(x_i, h_{realisable}(x_i)\}_{i=1}^{n}$
  $S_{val} \leftarrow D_{1:n_{val}}$
  $S_{train+test} \leftarrow S_{n_{val}+1:n}$
  $\sigma'_{opt} \leftarrow \frac{\sigma_{opt}}{\sqrt{2}}$ ▷ Set σ'_{opt} according to 19
  $[S_1, S_2, \ldots, S_k] \leftarrow S_{train+test}$ ▷ Split train / test data in k folds
  for $i = 1 : k_l$ **do** ▷ Get learning curve for fold i
    $S_{test} \leftarrow S_i$
    $P_0 \leftarrow S_{train+test} \setminus S_{test}$
    $n \leftarrow |P_0|$
    for $t = 1 : n$ **do** ▷ Run active learning algorithm
      $x_t^q \leftarrow \mathcal{Q}(P_{t-1}^x)$
      $y_t^q \leftarrow \mathcal{O}(x_t^q)$
      $S_t \leftarrow S_{t-1} \cup \{(x_t^q, y_t^q)\}$
      $P_t \leftarrow P_{t-1} \setminus \{(x_t^q, y_t^q)\}$
      $\hat{h}_t \leftarrow \mathcal{A}(S_t)$
      $\mathbf{L}_{i,t} \leftarrow \mathcal{E}_{S_{test}}(\hat{h}_t)$ ▷ Fill out loss matrix for current fold and step
    end for
  end for
end procedure

6.3.2 Plots



(a) bike sharing (day)

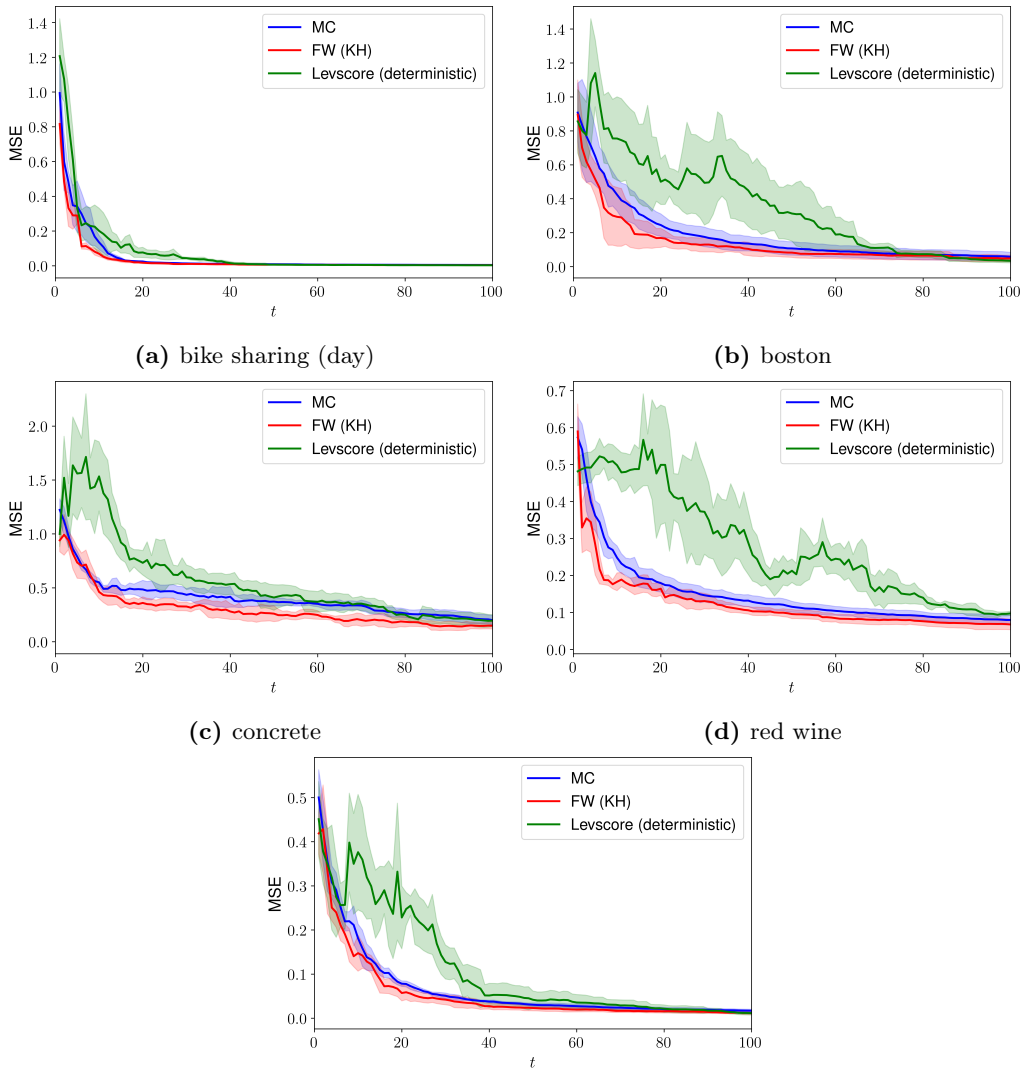
(b) boston

(c) concrete

(d) red wine

(e) white wine

Figure 6.1: Learning curves (bold line is mean with shaded region being ± 1 standard deviation over the 5 folds) for agnostic regression comparing FW (KH) with MC and Levscore. The y-axis is MSE (Mean Squared Error), x-axis is t (number of datapoints in current active learning train set). A good algorithm will have a trajectory that goes down quickly with t. MC is baseline.



(a) bike sharing (day)

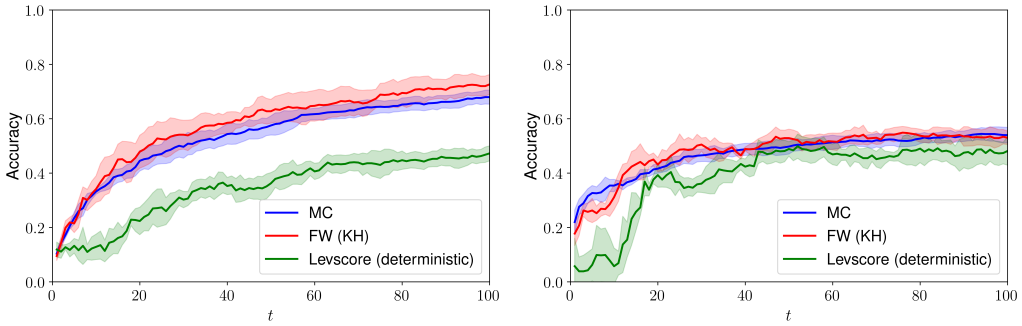
(b) boston

(c) concrete

(d) red wine

(e) white wine

Figure 6.2: Learning curves (bold line is mean with shaded region being ± 1 standard deviation over the 5 folds) for regression comparing FW (KH) with MC and Levscore. The y-axis is MSE (Mean Squared Error), x-axis is t (number of datapoints in current active learning train set). A good algorithm will have a trajectory that goes down quickly with t. MC is baseline.



(a) mnist **(b)** yeast

Figure 6.3: Learning curves (bold line is mean with shaded region being ± 1 standard deviation over the 5 folds) for agnostic classification comparing FW (KH) with MC and Levscore. The y-axis is Accuracy, x-axis is t (number of datapoints in current active learning train set). A good algorithm will have a trajectory that goes up quickly with t. MC is baseline.

6.3.3 Datasets and Hyperparameters

Dataset	n	d	λ_{opt}	σ_{opt}
bike sharing (day)	731	6	0.001	3.60
red wine	1000	11	0.01	2.88
concrete	1000	8	1e-05	6.47
boston	506	13	0.001	3.28
white wine	1000	11	0.0001	13.30

Table 6.1: Table for Agnostic Regression containing dataset information and hyperparameters. First column is the name of the dataset, n is the size, d the number of dimensions, λ_{opt} and σ_{opt} the hyperparameters chosen for KRR using kCV

Dataset	n	d	λ_{opt}	σ_{opt}
white wine	1000	11	0.0001	9.16
red wine	1000	11	0.0001	3.80
concrete	1000	8	1e-06	3.63
bike sharing (day)	731	6	0.0001	4.25
boston	506	13	0.0001	3.45

Table 6.2: Table for Realisable Regression containing dataset information and hyperparameters. First column is the name of the dataset, n is the size, d the number of dimensions, λ_{opt} and σ_{opt} the hyperparameters chosen for KRR using kCV

Dataset	n	d	λ_{opt}	σ_{opt}
yeast	1000	8	0.001	7.87
mnist	1000	784	0.001	79.96

Table 6.3: Table for Agnostic Classification containing dataset information and hyperparameters. First column is the name of the dataset, n is the size, d the number of dimensions, λ_{opt} and σ_{opt} the hyperparameters chosen for KRR using kCV

Bibliography

[1] D. Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[3] F. Bach. Sharp analysis of low-rank kernel matrix approximations. In *Conference on Learning Theory*, pages 185–209, 2013.

[4] F. Bach, S. Lacoste-Julien, and G. Obozinski. On the equivalence between herding and conditional gradient algorithms. *arXiv preprint arXiv:1203.4523*, 2012.

[5] M.-F. Balcan, S. Hanneke, and J. W. Vaughan. The true sample complexity of active learning. *Machine learning*, 80(2-3):111–139, 2010.

[6] M.-F. Balkan and R. Urner. Active learning - modern learning theory. Active learning survey, January 2015.

[7] E. B. Baum and K. Lang. Query learning can work poorly when a human oracle is used. In *International joint conference on neural networks*, volume 8, page 8, 1992.

[8] A. Beck and M. Teboulle. A conditional gradient method with linear rate of convergence for solving convex linear systems. *Mathematical Methods of Operations Research*, 59(2):235–247, 2004.

[9] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[10] F.-X. Briol, C. Oates, M. Girolami, and M. A. Osborne. Frank-wolfe bayesian quadrature: Probabilistic integration with theoretical guarantees. In *Advances in Neural Information Processing Systems*, pages 1162–1170, 2015.

[11] W. A. Chaovalitwongse, I. P. Androulakis, and P. M. Pardalos. Quadratic integer programming: Complexity and equivalent forms. *Encyclopedia of optimization*, pages 3153–3159, 2009.

[12] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[13] R. Chattopadhyay, Z. Wang, W. Fan, I. Davidson, S. Panchanathan, and J. Ye. Batch mode active sampling based on marginal probability distribution matching. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3):13, 2013.

[14] Y. Chen, M. Welling, and A. Smola. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012.

[15] C. Ciliberto. Advanced topics in machine learning: Introduction to statistical learning theory. https://cciliber.github.io/intro-slt/, 2018.

[16] C. Ciliberto, L. Rosasco, and A. Rudi. A consistent regularization approach for structured prediction. In *Advances in neural information processing systems*, pages 4412–4420, 2016.

[17] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, May 1994.

[18] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4(nil):129–145, 1996.

[19] C. Cortes and M. Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126, 2014.

[20] C. Cortes, M. Mohri, and A. M. Medina. Adaptation based on generalized discrepancy. *Machine Learning Research, forthcoming. URL http://www. cs. nyu. edu/˜ mohri/pub/daj. pdf*, 2019.

[21] N. Cressie. The origins of kriging. *Mathematical geology*, 22(3):239–252, 1990.

[22] S. Dasgupta. Analysis of a greedy active learning strategy. In *Advances in neural information processing systems*, pages 337–344, 2005.

[23] S. Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in neural information processing systems*, pages 235–242, 2006.

[24] S. Dasgupta, D. J. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Advances in neural information processing systems*, pages 353–360, 2008.

[25] J. Desjardins. How much data is generated each day? https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f, 2019. Accessed: 2019-07-29.

[26] D. Dua and C. Graff. UCI machine learning repository, 2017.

[27] T. Evgeniou and M. Pontil. Support vector machines: Theory and applications. In *Advanced Course on Artificial Intelligence*, pages 249–257. Springer, 1999.

[28] G. E. Fasshauer. Positive definite kernels: Past, present and future. *Dolomite Research Notes on Approximation*, 4:21–63, 2011.

[29] V. Fedorov. Optimal experimental design. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):581–589, 2010.

[30] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[31] R. Ganti and A. Gray. Upal: Unbiased pool based active learning. In *Artificial Intelligence and Statistics*, pages 422–431, 2012.

[32] A. Gretton and D. Seijdinovich. Advanced topics in machine learning: Reproducing kernel hilbert spaces in machine learning. http://www.gatsby.ucl.ac.uk/~gretton/coursefiles/rkhscourse.html, 2018.

[33] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

[34] Q. Gu and J. Han. Towards active learning on graphs: An error bound minimization approach. In *2012 IEEE 12th International Conference on Data Mining*, pages 882–887. IEEE, 2012.

[35] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *Advances in neural information processing systems*, pages 593–600, 2008.

[36] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[37] M. Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[38] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

[39] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[40] J. H. Manton, P.-O. Amblard, et al. A primer on reproducing kernel hilbert spaces. *Foundations and Trends® in Signal Processing*, 8(1–2):1–126, 2015.

[41] K. Muandet, K. Fukumizu, B. Sriperumbudur, B. Schölkopf, et al. Kernel mean embedding of distributions: a review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

[42] A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[43] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326, 2010.

[44] S. J. Qin. Process data analytics in the era of big data. *AIChE Journal*, 60(9):3092–3100, 2014.

[45] A. Rudi, D. Calandriello, L. Carratino, and L. Rosasco. On fast leverage score sampling and optimal learning. In *Advances in Neural Information Processing Systems*, pages 5672–5682, 2018.

[46] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. 1998.

[47] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, Jun 2012.

[48] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[49] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[50] S. Smale and D.-X. Zhou. Learning theory estimates via integral operators and their approximations. *Constructive approximation*, 26(2):153–172, 2007.

[51] I. Tolstikhin, B. K. Sriperumbudur, and K. Muandet. Minimax estimation of kernel mean embeddings. *The Journal of Machine Learning Research*, 18(1):3002–3048, 2017.

[52] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[53] V. Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

[54] T. J. Viering, J. H. Krijthe, and M. Loog. Nuclear discrepancy for active learning, 2017.

[55] M. Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128. ACM, 2009.

[56] R. Willett, R. Nowak, and R. M. Castro. Faster rates in regression via active learning. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 179–186. MIT Press, 2006.

[57] C. K. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in neural information processing systems*, pages 514–520, 1996.

[58] D. Wu. Pool-based sequential active learning for regression. *IEEE transactions on neural networks and learning systems*, 30(5):1348–1359, 2018.

[59] Z. Xu, T. Yu, and S. Sra. Towards efficient evaluation of risk via herding. 2019.

[60] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, volume 3, 2003.

[61] L. Zwald and G. Blanchard. On the convergence of eigenspaces in kernel principal component analysis. In *Advances in neural information processing systems*, pages 1649–1656, 2006.