Kernel Methods in Statistical Learning

Contents

1 Introduction and Preliminaries

This report will outline the use of *kernel* methods for solving problems in the field of statistical learning. The primary focus of the field of Statistical Learning is how to use a finite amount of noisy data to approximate, or make inferences about the true and unknown data generating process. The importance of this is that one may then use these approximations to make predictions about new, unseen data, and use these predictions to make informed decisions. One approach is to nominate a large class of functions defined on the data input space, and search over this class for a specific function that minimizes a chosen approximation error. One interesting class of functions that has been widely used is the Reproducing Kernel Hilbert Space (RKHS) of functions, which is a Hilbert space augmented with the extra structure of reproducing kernel. These spaces provide a large amount of flexibility in

modelling the data generating process, and they are often taken to be infinite dimensional. This added flexibility comes with the added cost of having to now search over an infinite dimensional space. This is addressed by the Representer Theorem [KW71], which states that under mild conditions, the optimization over an RKHS in fact reduces to a finite dimensional optimization problem.

This essay will first provide an introduction to the Statistical Learning problem. It will provide intuition by building up a concrete example of Bayesian regression, and extending this example to a Gaussian Process throughout the report. It will also provide a self contained exposition of RKHS theory with a particular focus on the construction and application of a RKHS to solve the statistical learning problem. In the remainder of sections 1 and 2, we define and motivate the need for RKHS theory in statistical learning. Then in section 3 we introduce and provide a proof of the Representer theorem, and implement a concrete example of RKHS regression. Finally, in section 5 we consider another popular kernel based method in the Statistical Learning literature, called the Gaussian Process.

2 Motivation

We begin with a motivating example of the Statistical Learning problem through an informal discussion of Bayesian Linear Regression as presented in [Ras04]. Consider an input set, $\mathcal{X}$, and a target set, $\mathcal{Y}$. For each input, $\mathbf{x} \in \mathcal{X}$, there is a corresponding target value, $y \in \mathcal{Y}$. For example, if we take $\mathcal{X} = \mathbb{R}^n$, with each element representing some measurement of a company's performance, then we may want to assign to each company an element $y \in \mathcal{Y}$ representing its stock price. Usually, we are given access to a set of N independent training points, $D = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^N$, and we must use this information to infer the relationship between $\mathcal{X}$ and $\mathcal{Y}$. It must be noted however that we cannot trust this data fully, and we need to assume that there exists some noise or error in the training data. The simplest possible approach when $\mathcal{Y} \subset \mathbb{R}$ is then to assume that this relationship is linear, and model it by f where:

$$f(\mathbf{x}) = x_1\theta_1 + x_2\theta_2 + \cdots + x_N\theta_N = \mathbf{x}^T\theta.$$

where θ is a vector of parameters, or weights that are not known a priori. We assume that the error in our approximation, $y - f(x)$, is random, and in particular follows a Gaussian distribution, that is:

$$y = f(\mathbf{x}) + \epsilon,$$

Note that $x_1 = 1$ is standard, so that θ_1 represents the constant/bias of our linear equation. Further, we assume $\epsilon \sim \mathcal{N}(0, \sigma^2)$, that is, ϵ is a random variable that follows a Gaussian distribution. In general, if a random variable $V \in \mathbb{R}^n$ follows a Gaussian distribution with mean vector $\mu_V \in \mathbb{R}^n$ and symmetric positive-definite covariance matrix $\Sigma_V \in \mathbb{R}^{n \times n}$, then its probability density function (pdf) is given by:

$$p(v; \mu_V, \Sigma_V) = \frac{1}{(2\pi)^{n/2}|\Sigma_V^{-1}|^{1/2}} \exp\left(-\frac{1}{2}(v - \mu_v)^T \Sigma_V (v - \mu_v)\right),$$

where the T superscript denotes the vector transpose, and $|\cdot|$ denotes the determinant. Given the distribution of ϵ, and assuming that θ is given, our assumption is that:

$$y|\mathbf{x}, \theta = \mathbf{x}^T\theta + \epsilon \sim \mathcal{N}(y|\mathbf{x}^T\theta, \sigma^2\mathcal{I}),$$

where $\mathcal{I}$ denotes the identity matrix and we have used the linearity of the Gaussian distribution. The goal is then to pick a vector θ that best fits our data, according to some objective that should measure how close our approximation is to the true data. The Bayesian approach is to assume that θ itself is random, and assign it a distribution, referred to as the prior distribution. The prior distribution encodes our initial beliefs before (prior) to seeing any of the data. When we do not have information about the problem at hand, it is common to use an uninformative prior, such as a zero mean Gaussian with an Identity matrix as the covariance. Using such a prior is the same as saying that we do not have any information that should influence the learning process, and so we are letting the data determine the model completely. Let us asume such an informative prior on θ:

$$\theta \sim \mathcal{N}(\theta|\mathbf{0}, \Sigma_\theta).$$

Given that the training data is assumed to be independent, we write $\mathbf{y} = (y_1, y_2, \ldots, y_N)^T$, and $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)^T$ as collections of the N targets and inputs respectively. We may then compute the likelihood:

$$\begin{aligned}
p(\mathbf{y}|X, \theta) &= \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \theta) \\
&= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i^T\theta)^2\right) \\
&= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y} - X^T\theta\|_2^2\right) \\
&= \mathcal{N}(\mathbf{y}|X^T\theta, \sigma^2\mathcal{I}),
\end{aligned}$$

where $\|\cdot\|_2$ denotes the Euclidean norm, that is, for any $y \in \mathbb{R}^n$, $\|y\|_2 = \sqrt{y_1^2 + y_2^2 + \ldots y_n^2}$.

At this point, we have our prior belief about θ, and we now also have the likelihood of our data. We must incorporate the information from the data to update our belief about θ. In other words, we need to calculate the conditional distribution $p(\theta|X, \mathbf{y})$. To do so, we need to make use of Bayes' rule:

$$p(\theta|X, \mathbf{y}) = \frac{p(\theta, X, \mathbf{y})}{p(X, \mathbf{y})} = \frac{p(\mathbf{y}|X, \theta)p(\theta|X)p(X)}{p(\mathbf{y}|X)p(X)} \propto p(\mathbf{y}|X, \theta)p(\theta),$$

where $\propto$ denotes proportionality. Since probabilities must sum to 1, we will often remove terms that do not involve parameters of interest, since we can re-normalize later. In this case, we are only interested in θ since that is the argument of the initial distribution, and

so we drop any terms independent of θ. Using this formula along with our assumption that $p(\theta) \sim \mathcal{N}(\theta|\mathbf{0}, \Sigma_p)$ and $p(\mathbf{y}|X, \theta) = \mathcal{N}(\mathbf{y}|X^T\theta, \sigma^2\mathbb{I})$, we calculate:

$$\begin{aligned}
p(\theta|X, \mathbf{y}) &\propto p(\mathbf{y}|X, \theta) \times p(\theta) \\
&= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y} - X^T\theta\|_2^2\right) \times \frac{1}{(2\pi)|\Sigma_p|^{1/2}} \exp\left(-\frac{1}{2}\theta^T\Sigma_p\theta\right) \\
&\propto \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y} - X^T\theta\|_2^2\right) \times \exp\left(-\frac{1}{2}\theta^T\Sigma_p\theta\right) \\
&\propto \exp\left(-\frac{1}{2}(\theta - \overline{\theta})^T\left(\frac{1}{\sigma^2}XX^T + \Sigma_p^{-1}\right)(\theta - \overline{\theta})\right),
\end{aligned}$$

with $\overline{\theta} = \frac{1}{\sigma^2}(\frac{1}{\sigma^2}XX^T + \Sigma_p^{-1})^{-1}X\mathbf{y} := \frac{1}{\sigma^2}AX\mathbf{y}$. We can see that up to constants, this is itself a Gaussian random variable, that is:

$$p(\theta|X, \mathbf{y}) \sim \mathcal{N}\left(\theta|\overline{\theta}, A^{-1}\right).$$

We call this updated belief over θ the posterior distribution. The posterior distribution represents our new belief about the best parameter vector θ for our linear model. Given a previously unseen test point x_* we would like to calculate a prediction for its corresponding target, denoted $f_* := f(\mathbf{x}_*)$. In the Bayesian sense, what we actually want to do is calculate a distribution over f_*, known as the predictive distribution. We obtain this distribution by taking a weighted average with respect to the posterior distribution, that is:

$$\begin{aligned}
p(f_*|\mathbf{x}_*, X, \mathbf{y}) &= \int_\theta p(f_*|\mathbf{x}_*, X, \mathbf{y}, \theta)p(\theta|X, \mathbf{y})d\theta \\
&= \int_\theta p(f_*|\mathbf{x}_*, \theta)p(\theta|X, \mathbf{y})d\theta \\
&= \mathcal{N}\left(f_*\middle|\frac{1}{\sigma^2}\mathbf{x}_*^T A^{-1}X\mathbf{y}, \mathbf{x}_*^T A^{-1}\mathbf{x}_*\right).
\end{aligned}$$

Note that in the second line, once we condition on θ, the distribution of y_* no longer depends on $X, \mathbf{y}$. This is because θ encodes all of the relevant information about the training data needed to make predictions for new inputs.

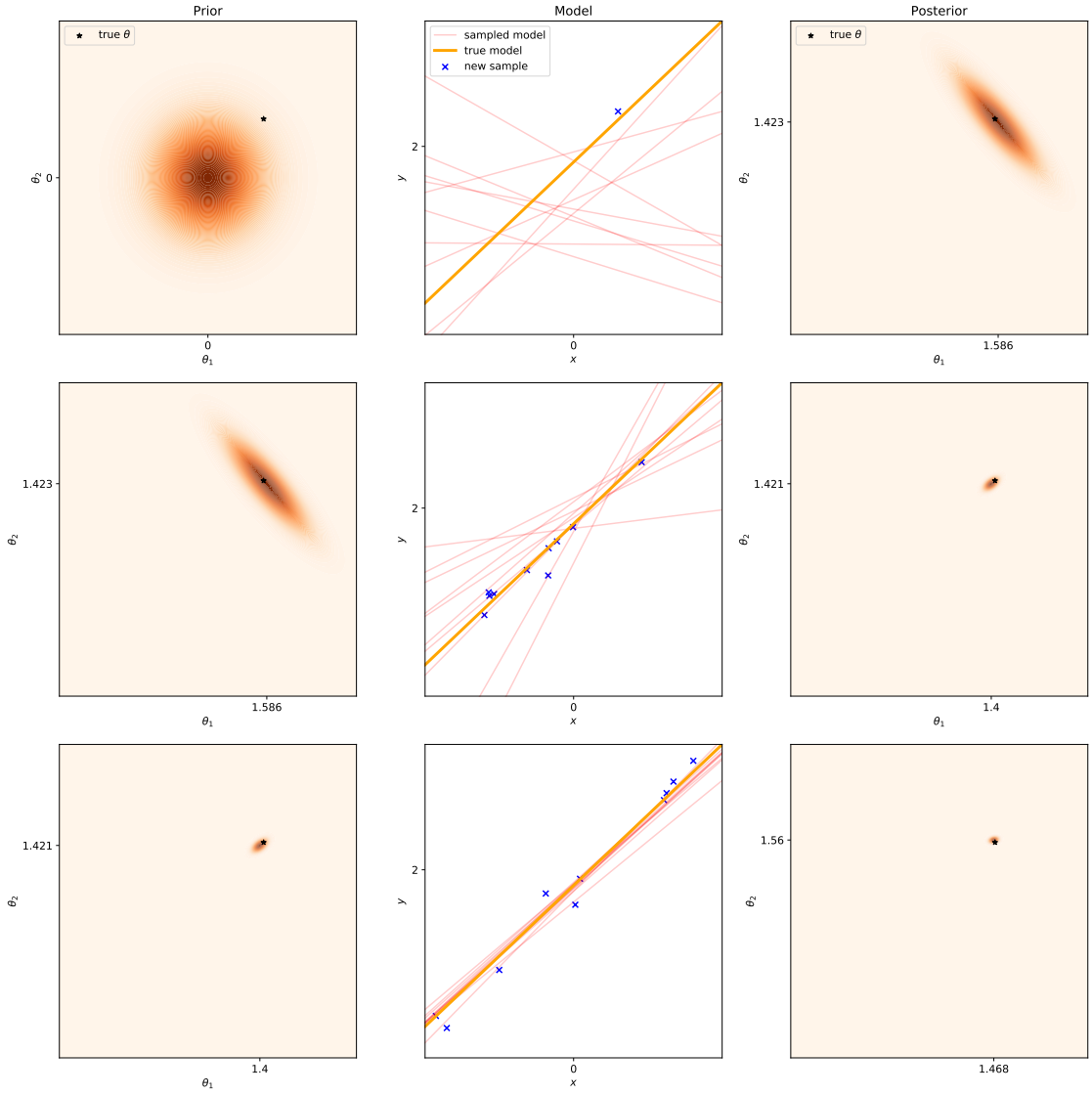

Figure 1: An example of sequential Bayesian linear regression.

This procedure gives us a very simple algorithm for Bayesian Linear regression that can be used in a sequential manner. For example, we may nominate a prior distribution, observe a subset of the training data of size $m < N$ and subsequently update our posterior to reflect our change in beliefs due to the observed data. We then observe more data, and so we treat our old posterior as a new prior, and update this to calculate a new posterior. This is a form of on-line, or stream learning. The name of this method comes from the fact that traditionally, statistical models are treated as static procedures which have access to the entire training set immediately. However, the Bayesian framework allows for a more

flexible approach.

We close this section with an implementation of Bayesian regression[I] depicted graphically in Figure 1. We assume that $\theta = (\theta_1, \theta_2) \in \mathbb{R}^2$. In this case, our model is a straight line with equation:

$$f(x) = \theta_1 + \theta_2 x.$$

We choose a true model, $\hat{\theta} = (\frac{3}{2}, \frac{3}{2})$, and begin with an uninformative prior - a Gaussian with zero mean and identity covariance matrix, whose contour plot is represented in the top-left plot. In the top-middle plot, the thick orange line represents the true model, whilst the lighter lines are random samples from our prior belief. We then assume that we have observed a single sample from the true model, represented by the blue cross. The top-right panel is then the posterior distribution, which has updated to reflect what we have observed. We then treat this posterior as our new prior, and repeat the process, this time sampling more points. Note that as we see more data, our uncertainty regarding the true parameters drops to zero, and the sampled models are very close to the true model.

2.1 A Formal Description

We proceed with a more formal exposition of the Statistical Learning problem, and motivate the use of concepts from functional analysis to solve some of these problems. The field of Statistical Learning is very large, therefore a complete exposition is difficult and the reader may wish to refer to [CS02, FHT01, EPP00] which were the primary sources for this section.

We begin by considering a set of *inputs*, $\mathcal{X}$, and a set of *targets*, $\mathcal{Y}$, and assume there exists a probability distribution, $P(x, y)$ defined on $\mathcal{X} \times \mathcal{Y}$, that specifies an unknown relationship between the two, so that any given input $x \in \mathcal{X}$ determines a probability distribution of potential targets on $\mathcal{Y}$. When the target is continuous, such as in the Bayesian Linear Regression example, we are dealing with a *regression* problem. If instead, the targets are discrete or categorical, then we call this a *classification* problem.

The difficulty of the problem is compounded by the fact that we only observe the relationship between $\mathcal{X}$ and $\mathcal{Y}$ a finite number of times. That is, we are constrained to a finite set of N data points, or *examples*, $D = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^{N}$, collected by sampling the set $\mathcal{X} \times \mathcal{Y}$ a total of N times according to $P(x, y)$. From here, one approach often taken in classical statistics is to directly specify the structure of the data generating process, for example by assuming that P follows a Gaussian distribution, and to use this as a framework to make inferences about the relationship between inputs and targets. The somewhat more pragmatic approach adopted in Statistical Learning is to assume that P is a black-box that can never be known. Instead, we nominate some sufficiently rich, compact class of functions

[I] code included in bayesian_reg.py, this can be run from terminal with the command: 'python3 bayesian_reg.py', assuming Python version 3 or later as well as the availability of recent versions of the Numpy and Matplotlib modules.

from $\mathcal{X}$ into $\mathcal{Y}$, denoted by $\mathcal{F}$, and approximate, or *learn*, a function $g \in \mathcal{F}$ that is able to predict an appropriate target for any given input, $x \in \mathcal{X}$. The inherent difficulty is that g is required to not only perform well on the provided examples, but it must also generalize to new inputs that may not have been encountered in the learning phase. To this end, we introduce a *loss* function, $L : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, that quantifies the error of a prediction $g(x)$ when the true target is y, i.e. $L(y, g(x))$. Typical choices for L in regression problems are the squared-loss, $(y - g(x))^2$, and the absolute-error loss, $|y - g(x)|$, and in the case of classification problems, a typical choice is the zero-one loss, $\mathbb{I}(y = g(x))$[II]. The average amount of error associated with a specific g is then captured by the *risk functional*, $\mathcal{R} : \mathcal{F} \to \mathbb{R}$, defined by:

$$\mathcal{R}(g) := \int_{\mathcal{X}\times\mathcal{Y}} L(y, g(x))P(x, y)\mathrm{d}x\mathrm{d}y,$$

and so our objective then becomes the identification of a minimizer, $g_\star := \arg\min_{g\in\mathcal{F}} \mathcal{R}(f)$. Whilst an objective has now been formally identified, the optimization is impossible to carry out given the dependence of $\mathcal{R}$ on the unknown distribution P. This motivates us to consider an approximation of the risk functional, called the *empirical risk functional*, which is defined in terms of the available examples as:

$$\mathcal{R}_e(g; N) := \frac{1}{N}\sum_{n=1}^{N} L(y_i, g(x_i)),$$

which quantifies for a specific g and number of examples, N, the average loss achieved.

Let $g_N := \arg\min_{g\in\mathcal{F}} \mathcal{R}_e(g; N)$, and note that from the definition, $g_\star$ is optimal over the entirety of $\mathcal{X}$, and so we need not worry about the ability of $g_\star$ to generalize beyond D. On the contrary, g_N is defined with respect to D, and so we can always find a function class rich enough to drive $\mathcal{R}_e(g_N)$ to zero. To see why, consider letting $\mathcal{F} = \mathcal{P}_{N-1}$, the space of polynomials of degree at most $N - 1$, then g_N will simply be the unique polynomial of degree $N - 1$ that interpolates the N examples in D, obtaining an average error of zero. This is of course not useful for any practical purposes, since this function will have been *overfit* to D and it is very unlikely that the true relationship has been learned, therefore not satisfying our requirement for good generalization to new data. The goal then cannot be simply to search for a function that minimizes the empirical risk over any given class $\mathcal{F}$, but to find a function in such an $\mathcal{F}$ so that the performance of this function in the limit approaches that of $g_\star$, that is, with probability one under P:

$$\lim_{N\to\infty} \mathcal{R}_e(g_N; N) = \lim_{N\to\infty} \mathcal{R}(g_N) = \mathcal{R}(g_\star).$$

We note that this does not imply that we seek to find $g_\star$, but merely a function that imitates the error of $g_\star$ as the number of examples increases, and it indeed may be the case that $g_\star \notin \mathcal{F}$.

[II] $\mathbb{I}(\cdot)$ is the indicator function that takes the value one if its argument is true, and zero otherwise.

We are therefore faced with a significant trade-off in choosing $\mathcal{F}$. On the one hand, we could pick $\mathcal{F}$ to be very rich and complex, but this will always lead to over-fitting the examples and not generalizing well to new data. Alternatively, we could restrict $\mathcal{F}$ to be simple, for example, take $\mathcal{F} = \mathcal{P}_1$, the space of linear functions which causes the risk of over-fitting to be virtually non-existent, however, this class will often be too crude an estimate of the natural phenomenon generating the data. This leads us to the idea of *regularization*, which proposes to consider a sufficiently rich class, $\mathcal{F}$, but to augment the empirical loss with a regularization functional, $h : \mathcal{F} \to \mathbb{R}$, that penalizes the complexity of the minimizer. That is, we now consider the regularized empirical loss:

$$\mathcal{R}_e^L(g; N, \lambda) := \mathcal{R}_e(g; N) + \lambda h(f) = \underbrace{\frac{1}{N}\sum_{n=1}^{N} L(y_i, g(x_i))}_{\text{goodness of fit}} \quad + \quad \underbrace{\lambda h(f)}_{\text{complexity of fit}}, \qquad (\star)$$

where we also introduce a tuning parameter, $\lambda > 0$, to control the amount of penalization applied. With this new objective, and a fixed λ, the function $\hat{g} := \arg\min_{g \in \mathcal{F}} \mathcal{R}_e^L(g; N, \lambda)$ will balance fitting the data sufficiently well while minimizing complexity, in other words, for two functions in $\mathcal{F}$ that fit the data well, we now have a preference for the simpler of the two. It is often the case that the function h is taken to be the norm, $h := \|\cdot\|_{\mathcal{F}}$ on the space of functions under consideration. It turns out that when we take $\mathcal{F}$ to be a RKHS, a structure to be introduced in the next section, the norm corresponds to a notion of smoothness, which can be viewed as a measure of complexity. We now therefore consider the theory behind RKHS, with the view of using a RKHS as our class of candidate functions over which to optimize $\mathcal{R}_e^L$.

2.2 Hilbert Spaces

We proceed with some preliminary material regarding Hilbert Spaces. This section is a summary of some of the results in [SS09, Chapter 4] and is included for completeness. A Hilbert space $\mathcal{H}$ is a vector space over a scalar field, $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$, equipped with an inner product, $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{F}$, which is linear in its *second* argument, conjugate symmetric, and for any $f \in \mathcal{H}$, $\langle f, f \rangle_{\mathcal{H}} \geq 0$. The inner product induces a norm on $\mathcal{H}$, defined for any $f \in \mathcal{H}$ by $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$, and satisfies all the usual requirements of a norm. Finally, we assume that $\mathcal{H}$ is complete in the metric induced by the norm, $d_{\mathcal{H}}(f, g) = \|f - g\|_{\mathcal{H}}$ for any $f, g \in \mathcal{H}$, and that $\mathcal{H}$ is separable, so that there exists a countable collection $\{f_k\} \subset \mathcal{H}$ such that their linear span is dense in $\mathcal{H}$. We also define the notion of a pre-Hilbert space, $\mathcal{H}_0$, which satisfies all properties of a Hilbert space except for completeness. We note that a pre-Hilbert space can always be *completed*. The completion of $\mathcal{H}_0$ is a Hilbert-space $\mathcal{H}$, with $\mathcal{H}_0 \subset \mathcal{H}$ and $\mathcal{H}_0$ dense in $\mathcal{H}$. Further, the restriction of the inner-product on $\mathcal{H}$ to elements of $\mathcal{H}_0$ is identical to the inner product of those elements under the $\mathcal{H}_0$ inner product. For a closed subspace $\mathcal{S} \subset \mathcal{H}$, we always have the decomposition $\mathcal{H} = \mathcal{S} \oplus \mathcal{S}^{\perp}$, where $\mathcal{S}^{\perp}$ denotes the orthogonal complement of $\mathcal{S}$. Therefore, any element $h \in \mathcal{H}$ can be written as $h = f + g$, where $f \in \mathcal{S}$, $g \in \mathcal{S}^{\perp}$. A linear functional, T, is a linear map from $\mathcal{H}$ to the underlying

scalar field, $\mathbb{F}$, and the continuity of a linear functional implies its boundedness, and vice versa. We denote by $\mathcal{H}^*$ the dual of $\mathcal{H}$, that is, the space of bounded linear functionals on $\mathcal{H}$. The Riesz Representation theorem states that for any $T \in \mathcal{H}^*$, there exists a unique $g \in \mathcal{H}$ such that $T(f) = \langle f, g \rangle_{\mathcal{H}}$ for any $f \in \mathcal{H}$, and moreover, $\|T\| = \|f\|_{\mathcal{H}}$, where $\|\cdot\|$ denotes the operator norm, $\|T\| = \sup\{\|Th\| \mid h \in \mathcal{H},\ \|h\|_{\mathcal{H}} = 1\}$. Finally, we denote the space of *square-summable* sequences by $\ell_2(\mathbb{N}) := \left\{(a_n)_{n=1}^{\infty} \mid a_n \in \mathbb{N},\ \sum_{n=1}^{\infty} |a_n|^2 < \infty\right\}$.

3 Kernel Methods

Given an input set, $\mathcal{X}$, many common algorithms in Statistical Learning, such as the Support Vector Machine and the Perceptron consider a notion of similarity, captured by the inner product on $\mathcal{X}$, that is $\langle x, x' \rangle$ for $x, x' \in \mathcal{X}$. This can often be restrictive since the inner product is linear, and we are therefore constrained to linear representations of the data. One common approach is to project $\mathcal{X}$ into a feature space, Υ, and then consider inner products of the transformed elements. This allows for non-linear and hence more flexible representations of $\mathcal{X}$, that is, we seek a *feature map*, $\phi : \mathcal{X} \to \Upsilon$, where Υ represents a higher dimensional feature space. We would then like to compute the similarity of the transformed points in the feature space, that is:

$$k(x, x') := \langle \phi(x), \phi(x') \rangle_{\Upsilon} \quad \forall x, x' \in \mathcal{X},$$

where we have introduced the *kernel function*, k, as a non-linear similarity metric in place of the standard inner product on $\mathcal{X}$. We therefore have a more flexible representation of our data that is found by finding a feature map and then taking inner products. For example, we may take $\mathcal{X} = \mathbb{R}^2$ and can consider the feature map:

$$\phi : \mathcal{X} \to \Upsilon, \quad \phi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2] \quad \forall x = (x_1, x_2) \in \mathcal{X}.$$

Under this particular feature map, the feature space is a 6-dimensional representation of $\mathcal{X}$. The question then arises of how one might come up with useful feature maps - the manual approach can be to choose many non-linear transformations of the input points and combine them into a map. However, building the feature map explicitly and taking inner products on Υ can be computationally costly, especially when we have feature maps that map to higher dimension feature spaces. We can bypass this step by noticing that we can write the inner product in the previous example in the following way for any $x, x' \in \mathcal{X}$:

$$\begin{aligned}\langle \phi(x), \phi(x') \rangle_{\Upsilon} &= 1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_1'x_2x_2' \\ &= (1 + \langle x, x' \rangle_{\mathcal{X}})^2.\end{aligned}$$

What this tells us is that the inner product on the feature space can be written explicitly as a function of the inner product on the original space, $\mathcal{X}$.

With this in mind, we can approach the problem in the reverse order: Given a kernel function, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is a function of the inner product on the original space,

can we decompose this into an explicit feature map, so that $k(x, x') = \langle \phi(x), \phi(x') \rangle_\Upsilon$? The significance of this is that we can bypass the explicit feature mappings and computation of inner products in high dimensional spaces by studying only this function. It turns out that such functions are guaranteed to exist, and the above example is an instance of the polynomial kernel, which can be generalized as:

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad k(x, x') = (m + \langle x, x' \rangle_\mathcal{X})^d, \quad m \in \mathbb{R}, d \in \mathbb{N},$$

which may be viewed as determining a d-dimensional feature space, Υ. We therefore have that for any algorithm defined purely in terms of inner products in the input set can then be lifted into a much higher dimensional space through the use of the kernel, without much extra work. This is often referred to as the 'kernel trick'. The true power of the kernel trick is that for many popular kernels, the feature space is infinite dimensional. This means that we get an infinite dimensional representation of our data space without ever having to explicitly compute the feature map, nor work with infinite dimensional objects.

For the remainder of this section, we keep the kernel trick in mind and introduce the mathematical fundamentals of Reproducing Kernel Hilbert Spaces, which will be seen as the spaces of functions induced by special types of kernels, called Reproducing Kernels. We will see that these kernels, defined on the product space $\mathcal{X} \times \mathcal{X}$, induce a space of functions defined as the linear span of the corresponding feature maps, $\phi := k(x, \cdot)$. We also introduce various types of kernels and discuss their properties.

3.1 Kernelizing Bayesian Linear Regression

Before moving on to the theory, we first go through a deeper example of kernelizing a Statistical Learning algorithm, namely the Bayesian Linear Regression algorithm, as presented in [Ras04], introduced at the start of this report. This example will serve to motivate the wide reach of the kernel trick introduced in the previous section. We consider a K dimensional input space, $\mathcal{X}$, and we restrict ourselves to feature spaces, Υ, of dimension at most $M > K$. So we have the feature map, ϕ, that maps subsets of $\mathbb{R}^N$ into subsets of $\mathbb{R}^M$. Such a feature map is independent of the parameter vector, θ, and so the algorithm for Bayesian Regression can be extended easily by replacing instances of the data matrix X, with the transformed data under ϕ. That is, for a set of N data points, we replace:

$$X = [x_1, x_2, \ldots, x_N]^T \in \mathbb{R}^{N \times K} \iff \Phi(X) = [\phi(x_1), \phi(x_2), \ldots, \phi(x_N)]^T \in \mathbb{R}^{N \times M}.$$

The Bayesian Linear Regression model for $\mathbf{x} \in \mathbb{R}^N$ then simply becomes:

$$y = \phi(x_1)\theta_1 + \phi(x_2)\theta_2 + \cdots + \phi(x_N)\theta_N + \epsilon = \phi(\mathbf{x})^T\theta + \epsilon.$$

If we introduce the notation $\phi_* := \phi(x_*), \Phi := \Phi(X)$, it follows easily then that:

$$\overline{\theta} = \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}\Phi\Phi^T + \Sigma_p^{-1}\right)^{-1}\Phi\mathbf{y},$$

and the predictive distribution for new data point $f_* = f(x_*)$ is simply:

$$p(f_*|\mathbf{x}_*, X, \mathbf{y}) = \mathcal{N}\left(f_* \middle| \frac{1}{\sigma^2}\phi_*^T \left(\frac{1}{\sigma^2}\Phi\Phi^T + \Sigma_p^{-1}\right)^{-1} \Phi\mathbf{y},\ \phi_*^T \left(\frac{1}{\sigma^2}\Phi\Phi^T + \Sigma_p^{-1}\right)^{-1} \phi_*\right).$$

Now, recalling[III] the following two matrix inversion lemmas for invertible matrices $A, Z \in \mathbb{R}^{n\times n}$, invertible matrix $W \in \mathbb{R}^{m\times m}$ and matrices $B, U, V \in \mathbb{R}^{n\times m}$:

$$(A + BB^T)^{-1}B = A^{-1}B(\mathcal{I} + B^TA^{-1}B)^{-1}, \tag{1}$$
$$(Z + UWV^T)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^TZ^{-1}U)^{-1}V^TZ^{-1}, \tag{2}$$

we may rewrite the mean of the predictive distribution as:

$$\begin{aligned}
\frac{1}{\sigma^2}\phi_*^T \left(\frac{1}{\sigma^2}\Phi\Phi^T + \Sigma_p^{-1}\right)^{-1} \Phi\mathbf{y} &= \frac{1}{\sigma}\phi_*^T \left(\left(\frac{1}{\sigma}\Phi\right)\left(\frac{1}{\sigma}\Phi\right)^T + \Sigma_p^{-1}\right)^{-1} \left(\frac{1}{\sigma}\Phi\right)\mathbf{y} \\
&= \frac{1}{\sigma}\phi_*^T (BB^T + A)^{-1}B\mathbf{y} \\
&= \frac{1}{\sigma}\phi_*^T A^{-1}B(\mathcal{I} + B^TA^{-1}B)^{-1}\mathbf{y} && \text{By (1)} \\
&= \frac{1}{\sigma}\phi_*^T \Sigma_p \left(\frac{1}{\sigma}\Phi\right)\left(\mathcal{I} + \left(\frac{1}{\sigma}\Phi\right)^T \Sigma_p \left(\frac{1}{\sigma}\Phi\right)\right)^{-1}\mathbf{y} \\
&= \left(\Phi^T\Sigma_p\phi_*\right)^T \left(\sigma^2\mathcal{I} + \Phi^T\Sigma_p\Phi\right)^{-1}\mathbf{y} \\
&= K_*^T \left(\sigma^2\mathcal{I} + K\right)^{-1}\mathbf{y},
\end{aligned}$$

where the definitions of A and B are clear from the context, and we have introduced the matrices $K := \Phi^T\Sigma_p\Phi$ and vector $K_* = \Phi^T\Sigma_p\phi_*$. Further, we may rewrite the variance of the predictive distribution as:

$$\begin{aligned}
\phi_*^T \left(\frac{1}{\sigma^2}\Phi\Phi^T + \Sigma_p^{-1}\right)^{-1} \phi_* &= \phi_*^T \left(\Phi\left(\frac{1}{\sigma^2}\mathcal{I}\right)\Phi^T + \Sigma_p^{-1}\right)^{-1} \phi_* \\
&= \phi_*^T \left(UWV^T + Z\right)^{-1} \phi_* \\
&= \phi_*^T (Z^{-1} - Z^{-1}U(W^{-1} + V^TZ^{-1}U)^{-1}V^TZ^{-1})\phi_* && \text{by (2)} \\
&= \phi_*^T (\Sigma_p - \Sigma_p\Phi(\sigma^2\mathcal{I} + \Phi^T\Sigma_p\Phi)^{-1}\Phi^T\Sigma_p)\phi_* \\
&= \phi_*^T\Sigma_p\phi_* - \phi_*^T\Sigma_p\Phi(\sigma^2\mathcal{I} + \Phi^T\Sigma_p\Phi)^{-1}\Phi^T\Sigma_p\phi_* \\
&= K_{**} - K_*^T(\sigma^2\mathcal{I} + K)^{-1}K_*,
\end{aligned}$$

where we have introduced the notation $K_{**} := \phi_*^T\Sigma_p\phi_*$. In summary, we can now write:

[III] See, for example, the Matrix Cookbook.

$$p(f_*|\mathbf{x}_*X,\mathbf{y}) = \mathcal{N}(f_*|K_*^T(K+\sigma^2\mathcal{I})^{-1}\mathbf{y}, K_{**} - K_*^T(K+\sigma^2\mathcal{I})^{-1}K_*).$$

Note that the reason we have done this work to rewrite the predictive distribution is that in the current form, the feature representation of the data enters the algorithm only in the 'K' terms, that is the terms: K, K_* and K_{**}. We can define the kernel:

$$k(\mathbf{x},\mathbf{x}') := \phi(\mathbf{x})\Sigma_p\phi(\mathbf{x}'),$$

and note that from this kernel we may compute the elements of all the K terms directly. It will be important to keep this in mind moving forward as we generalize the Bayesian Linear Regression to the Gaussian Process, in which we are primarily interested in the kernel, and how it determines the predictive distribution, the opposite of the approach we have taken in this example.

3.2 The Reproducing Kernel Hilbert Space

We now take a short detour to describe the mathematical fundamentals of kernel theory. We first introduce the Reproducing Kernel Hilbert Space (RKHS) and connect the theory back to the kernel trick. The following two sub-sections provide the core definitions and results of RKHS theory, and the primary source was the text [BTA11], unless otherwise stated. Given an arbitrary input set, $\mathcal{X}$, we can define a Hilbert space $\mathcal{H}$ of functions from the input space $\mathcal{X}$ to an underlying scalar field, $\mathbb{F}$.

Definition 3.1 (Evaluation Functional)**.** *An evaluation functional on a Hilbert space of functions is the map that associates to each element in $\mathcal{H}$ its evaluation at a fixed point $x \in \mathcal{X}$. Formally, it is the map:*

$$e_x : \mathcal{H} \to \mathbb{F}, \qquad e_x(f) = f(x) \quad \forall f \in \mathcal{H}.$$

Next, we formalize the idea of a kernel, which is a map from $\mathcal{X} \times \mathcal{X}$ to the underlying scalar field $\mathbb{F}$. We have already noted that the kernel can be taken to be a similarity metric on the product space $\mathcal{X} \times \mathcal{X}$. One special class of kernels that will be the focus of our study are the reproducing kernels.

Definition 3.2 (Reproducing Kernel)**.** *Given a set $\mathcal{X}$, and a kernel, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{F}$, then k is a reproducing kernel of the Hilbert space $\mathcal{H}$ if the following two properties hold for any $x \in \mathcal{X}$ and any $f \in \mathcal{H}$:*

(i) $k(\cdot, x) \in \mathcal{H}$,

(ii) $\langle k(\cdot, x), f\rangle_{\mathcal{H}} = f(x)$.

The latter of the two is known as the *reproducing property* of k, since it allows us to reproduce the value of f at x by an inner product on $\mathcal{H}$. For a fixed $x \in \mathcal{X}$, the quantity $k(\cdot, x)$ is a function on $\mathcal{H}$ that represents x by its distance from all other points under a chosen notion of distance determined by the choice of k, and it follows from the two properties of a reproducing kernel that for any $(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$, $k(x_1, x_2) = \langle k(\cdot, x_1), k(\cdot, x_2)\rangle_{\mathcal{H}}$.

Definition 3.3 (Reproducing Kernel Hilbert Space)**.** *A Hilbert space, $\mathcal{H}$, together with a reproducing kernel, k, is known as a Reproducing Kernel Hilbert Space (RKHS), and is denoted $\mathcal{H}_k$ to signify that $\mathcal{H}$ is a RKHS with respect to the specific kernel, k.*

We now consider a simple example of a RKHS, taken from [BTA11].

Example 3.4. *Let $\mathcal{H} = \ell_2(\mathbb{N})$ and consider the kernel:*

$$k : \ell_2(\mathbb{N}) \times \ell_2(\mathbb{N}) \to \{0, 1\}, \qquad k(m, n) = \mathbb{I}(m = n).$$

This is a reproducing kernel since for any $m \in \mathbb{N}$ and $x := (x_i)_{i=1}^{\infty} \in \ell_2(\mathbb{N})$:

(i) $k(\cdot, m) = (0, 0, \dots, 0, 1, 0, \dots) \in \ell_2(\mathbb{N})$ *where the* 1 *is at the m-th index.*

(ii) $\langle k(\cdot, m), x\rangle_{\ell_2(\mathbb{N})} = \sum_{i=1}^{\infty} x_i \mathbb{I}(i = m) = x_m$.

Therefore, $\ell_2(\mathbb{N})$ is a RKHS with respect to this kernel.

Given that we are projecting our input space $\mathcal{X}$ into a feature space $\mathcal{H}$, it would be ideal to have that elements that are similar in $\mathcal{H}$ are also close in $\mathcal{X}$, otherwise we lose the structure in our data. The first notion of similarity is closeness with respect to the norm on $\mathcal{H}$, and the second notion is pointwise. In general, norm convergence need not imply pointwise convergence and vice versa in a Hilbert space. It turns out that it is precisely in a RKHS that all evaluation functionals are continuous, and therefore these spaces have the nice property that pointwise convergence is implied by norm convergence.

Theorem 3.5. *A Hilbert space $\mathcal{H}$ over complex valued functions on a set $\mathcal{X} \neq \emptyset$ has a reproducing kernel if and only if all of its evaluation functionals are continuous.*

Proof. In one direction, assume that there exists a reproducing kernel on $\mathcal{H}$, and denote it by k. Fix $x \in \mathcal{X}$ and consider the function $k(\cdot, x) \in \mathcal{H}$. Then, by the reproducing property, we have for any $f \in \mathcal{H}$ that $\langle f, k(\cdot, x)\rangle_{\mathcal{H}} = f(x)$. It follows that:

$$\begin{aligned} |e_x f| = |f(x)| &= |\langle f, k(\cdot, x)\rangle_{\mathcal{H}}| \\ &\leq \|f\|_{\mathcal{H}} \|k(\cdot, x)\|_{\mathcal{H}} && \text{(Cauchy-Schwarz)} \\ &= \|f\|_{\mathcal{H}} \sqrt{\langle k(\cdot, x), k(\cdot, x)\rangle_{\mathcal{H}}} \\ &= \|f\|_{\mathcal{H}} \sqrt{k(x, x)} < \infty. \end{aligned}$$

Hence, e_x is bounded and thus continuous, and since it is clearly linear, we have that $e_x \in \mathcal{H}^*$ for any $x \in \mathcal{X}$. In the reverse direction, assume that $e_x \in \mathcal{H}^*$ for any $x \in \mathcal{X}$. Then, by the Riesz-Representation theorem, there exists $g_x \in \mathcal{H}$ such that for any $f \in \mathcal{H}$, we have $e_x f = \langle f, g_x\rangle_{\mathcal{H}}$. Since this holds for any x, we can take $k(\cdot, x) := g_x$ as our reproducing kernel, and the proof is complete. □

An immediate consequence of this is that for any sequence $(f_n)_{n=1}^{\infty} \subset \mathcal{H}$ converging in norm to some $f \in \mathcal{H}$, we also get pointwise convergence, since for any $x \in \mathcal{X}$:

$$|f_n(x) - f(x)| = |e_x(f_n) - e_x(f)| \leq \|e_x\|_{\mathcal{H}^*} \|f_n - f\|_{\mathcal{H}} \longrightarrow 0,$$

by the fact that e_x is bounded. From this we may conclude that the RKHS is a suitable feature space, since it preserves distances between data points.

3.3 From Kernels to RKHS

In the previous section, we defined the notion of a RKHS, however, this definition tells us very little on how one can identify a suitable RKHS when all we have at our disposal is a finite set of examples D. The obvious first step is to be able to identify reproducing kernels on $\mathcal{X} \times \mathcal{X}$, for arbitrary $\mathcal{X}$, and so we seek a simpler characterization of reproducing kernels than currently provided by their formal definition. To achieve this, we first need to introduce the concept of a positive-definite function, which generalizes positive definite matrices.

Definition 3.6 (Positive-Definite Functions)**.** *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ be a conjugate-symmetric function. Then, we say that k is positive definite if for any $(x_i)_{i=1}^N \subset \mathcal{X}$ and any $(\alpha_i)_{i=1}^N \subset \mathbb{F}$ then:*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \overline{\alpha}_i \alpha_j k(x_i, x'_j) \geq 0.$$

The following lemma then allows us to easily construct positive-definite functions.

Lemma 3.7. *Suppose $\mathcal{X}$ is a set and $\mathcal{H}$ is a Hilbert space, with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Let $\phi : \mathcal{X} \to \mathcal{H}$, and let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{F}$ be defined as $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$. Then k is positive definite.*

Proof. The proof follows almost immediately from the definition: let $(x_i)_{i=1}^N \subset \mathcal{X}$, $(\alpha_i)_{i=1}^N \subset \mathbb{F}$ be arbitrarily chosen, then

$$\begin{aligned}
\sum_{i=1}^{N}\sum_{j=1}^{N} \overline{\alpha}_i \alpha_j k(x_i, x'_j) &= \sum_{i=1}^{N}\sum_{j=1}^{N} \langle \alpha_i \phi(x_i), \alpha_j \phi(x_j) \rangle_{\mathcal{H}} \\
&= \left\langle \sum_{i=1}^{N} \alpha_i \phi(x_i), \sum_{j=1}^{N} \alpha_j \phi(x_j) \right\rangle_{\mathcal{H}} \\
&= \left\| \sum_{i=1}^{N} \alpha_i \phi(x_i) \right\|_{\mathcal{H}}^2 \geq 0.
\end{aligned}$$

□

With these two results, we now state a fundamental result in the study of RKHS, known as the Moore-Aronsajn Theorem [Aro50], which provides the link between positive definite functions, reproducing kernels and the corresponding RKHS.

Theorem 3.8 (Moore-Aronsajn)**.** *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{F}$ be a positive definite kernel, then there exists a unique RKHS, $\mathcal{H}_k$, with reproducing kernel k.*

The implication of this theorem when combined with Lemma 3.7 gives us a lot to work with. First, since every positive-definite kernel is a reproducing kernel, the map $\phi : \mathcal{X} \to \mathcal{H}$

in Lemma 3.7 is simply $\phi(x) = k(\cdot, x)$ which we have seen multiple examples of. The feature map maps the raw input into a feature space, and represents x by its feature vector, $\phi(x)$, which captures the similarity of x to all other points in the input space $\mathcal{X}$. Even though this is an extremely rich representation, we are still able to take inner products in the feature space by the definition of the reproducing kernel. The Moore-Aronsajn theorem tells us that for each choice of positive definite k, there is a unique corresponding RKHS, $\mathcal{H}_k$ that contains these representations. We may therefore approach the notion of a RKHS from the opposite direction of what was presented in the preceding section, by nominating a positive definite kernel and constructing the RKHS. The following is an informal discussion of how $\mathcal{H}_k$ can be constructed and is based on the arguments in [SHS01, KHSS18].

Assume that we are given a non-empty input set, $\mathcal{X}$, and a positive definite kernel, k, and define the corresponding feature map, $\phi(x) = k(\cdot, x)$. We can construct an inner-product space by first considering the linear span of the images of the inputs in $\mathcal{X}$ under ϕ. Let:

$$\mathcal{H}_0 := \text{span}\{k(\cdot, x) \mid x \in \mathcal{X}\} = \left\{ f = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i) \;\middle|\; n \in \mathbb{N},\ (\alpha_i)_{i=1}^n \subset \mathbb{F},\ (x_i)_{i=1}^n \subset \mathcal{X} \right\}.$$

Then we can endow $\mathcal{H}_0$ with an inner-product, and hence convert it into a pre-Hilbert space. Let $m, n \in \mathbb{N}$, $(\gamma_i)_{i=1}^n$, $(\beta_j)_{j=1}^m \subset \mathbb{F}$ and $(x_i)_{i=1}^n$, $(x'_j)_{j=1}^m \subset \mathcal{X}$, then take $f, g \in \mathcal{H}_0$ with $f(\cdot) = \sum_{i=1}^n \gamma_i k(\cdot, x_i)$ and $g(\cdot) = \sum_{j=1}^m \beta_j k(\cdot, x'_j)$, and define:

$$\langle f, g \rangle_{\mathcal{H}_0} := \sum_{i=1}^{n} \sum_{j=1}^{m} \overline{\gamma}_i \beta_j \langle k(\cdot, x_i), k(\cdot, x'_j) \rangle_{\mathcal{H}_0} = \sum_{i=1}^{n} \sum_{j=1}^{m} \overline{\gamma}_i \beta_j k(x_i, x'_j).$$

This is well defined since $\langle f, g \rangle_{\mathcal{H}_0} = \sum_{i=1}^n \overline{\gamma}_i g(x_i) = \sum_{j=1}^m \beta_j \overline{f(x'_j)}$ and satisfies the properties of an inner-product, so induces a norm in the usual way, denoted $\|f\|_{\mathcal{H}_0}$. The corresponding RKHS, $\mathcal{H}_k$, will be the completion of this space:

$$\mathcal{H}_k := \overline{\mathcal{H}}_0 = \left\{ f = \sum_{i=1}^{\infty} \alpha_i k(\cdot, x_i) \;\middle|\; (\alpha_i)_{i=1}^\infty \subset \mathbb{F},\ (x_i)_{i=1}^\infty \subset \mathcal{X},\ \text{s.t. } \|f\|_{\mathcal{H}_k} < \infty \right\},$$

with $\|f\|^2_{\mathcal{H}_k} := \lim_{n\to\infty} \|\sum_{i=1}^n \alpha_i k(\cdot, x_i)\|^2_{\mathcal{H}_0} = \sum_{i,j=1}^\infty \overline{\alpha}_i \alpha_j k(x_i, x_j)$. From this construction, it is clear that functions living in the RKHS, $\mathcal{H}_k$, inherit their properties from the defining kernel, k, since they are linear combinations of this kernel. Therefore, the choice of kernel governs the type of function class we are searching over. Some examples of real-valued reproducing kernels collected from [Rob14, chapter 14] and [KHSS18] are presented in Table 1[IV] We also visually present some of these kernels in Figure 2 where $k(0, x)$ is plotted for various kernels and hyper-parameter settings. Whilst reproducing kernels may be defined for any non-empty input X, it is most often the case that we are dealing with real-valued

[IV] See also the Kernel Cookbook

data, or real-valued statistics calculated from non-standard input, see [FHT01, Chapter 9].

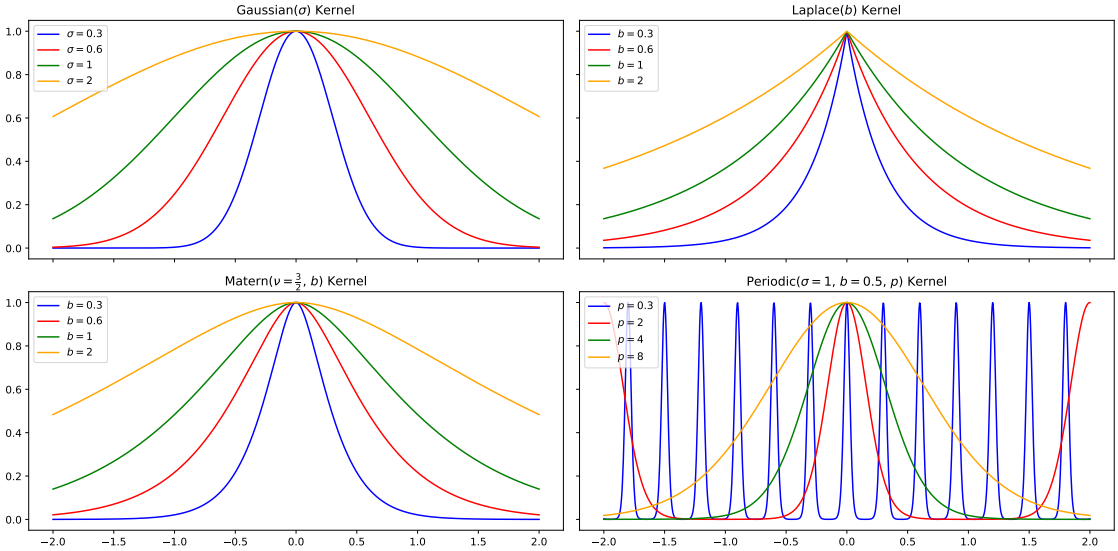

Figure 2: Plots of various kernels, introduced in Table 1 for different hyper-parameter settings.

The kernels in Table 1 are parameterized by the corresponding hyper-parameters, so in fact each definition corresponds to a collection of kernels, with each hyper-parameter setting defining a specific kernel.[V] As we saw earlier, kernels allow us to lift our input data into higher dimensional feature spaces, and for many of the kernels in 1, this feature space is in fact infinite dimensional. This is particularly remarkable since the kernel gives us the ability to build this representation and take inner products on this space without ever explicitly writing down the feature map. From Lemma 3.7, every positive definite function defines a kernel, and every kernel in turn defines a feature map and corresponding RKHS feature space. The following example, taken from an exercise in [Rob14, Chapter 14] illustrates this point.

Example 3.9 (Gaussian Feature Maps)**.** *We previously considered the polynomial feature map, which projects the input space into d-dimensional feature space, where d is always finite. We now consider the Gaussian kernel, which projects data into infinite dimensional*

[V] For the Matern kernel, $\Gamma(\cdot)$ is the Gamma function, $K_\nu(\cdot)$ is the modified Bessel function of the second kind.

Name	$k(x_1, x_2)$	hyper-parameters
Polynomial	$(x_1^T x_2 + m)^d$	$m \geq 0,\ d \in \mathbb{N}$
Gaussian	$\exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$	$\sigma > 0$
Sigmoidal	$\tanh(\gamma x_1^T x_2 + r)$	$\gamma > 0,\ r \geq 0$
Laplace	$\exp\left(-\frac{\|x_1 - x_2\|}{b}\right)$	$b > 0$
Matern	$\frac{2^{1-\nu}}{\Gamma(\nu)}\left(-\frac{\sqrt{2\nu}\|x_1 - x_2\|}{b}\right)^{\nu} \mathrm{K}_\nu\left(\frac{\sqrt{2\nu}\|x_1 - x_2\|}{b}\right)$	$\nu \geq 0,\ b > 0$
Periodic	$\sigma^2 \exp\left(-\frac{2}{b^2}\sin^2\left(\frac{\pi\|x_1 - x_2\|}{p}\right)\right)$	$\sigma > 0,\ p > 0,\ b > 0$

Table 1: Collection of popular reproducing kernels in the RKHS literature. For these examples, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with $\mathcal{X} \subset \mathbb{R}^N$

space. To see this, let $X = \mathbb{R}$, and let $x_1, x_2 \in X$, $\sigma > 0$, then:

$$\begin{aligned} k(x_1, x_2) &= \exp\left(-\frac{1}{2\sigma^2}(x_1 - x_2)^2\right) \\ &= \exp\left(-\frac{x_1^2}{2\sigma^2}\right)\exp\left(-\frac{x_2^2}{2\sigma^2}\right)\sum_{k=0}^{\infty}\frac{(x_1 x_2)^k}{\sigma^{2k}k!} \qquad \left(\textit{Taylor expansion of } \exp\left(\frac{x_1 x_2}{\sigma^2}\right)\right) \\ &= \exp\left(-\frac{x_1^2}{2\sigma^2}\right)\exp\left(-\frac{x_2^2}{2\sigma^2}\right)\sum_{k=0}^{\infty}\frac{x_1^k}{\sigma^k\sqrt{k!}}\frac{x_2^k}{\sigma^k\sqrt{k!}}, \end{aligned}$$

which can be written as $\langle \phi(x_1), \phi(x_1) \rangle = \phi(x_1)^T \phi(x_2)$ where:

$$\phi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right)\left[1, \quad \frac{x}{\sigma\sqrt{1!}}, \quad \frac{x^2}{\sigma^2\sqrt{2!}}, \quad \frac{x^3}{\sigma^3\sqrt{3!}}, \quad \dots\right]^T.$$

It is therefore impossible to explicitly represent $\phi(x)$, for the Gaussian case, and for many popular kernels such as the Matern and Sigmoidal families. However, we never actually need to, since we can always calculate distances in the feature space simply by evaluating k.

4 The Representer Theorem

We have so far introduced the notion of a RKHS, and showed that to each positive-definite function on the product $\mathcal{X} \times \mathcal{X}$, there exists a corresponding RKHS. However, we have also noted that this RKHS is often an infinite dimensional vector space, and whilst this can

lead to very flexible representations, it introduces a new challenge when considering the Statistical Learning optimization problem defined by ($\star$), since we must now search over an infinite space of functions. The Representer theorem is the remarkable fact that when this infinite dimensional space is taken to be a RKHS, and the regularization functional is taken to be a monotonically increasing function, that the function that minimizes ($\star$) can always be represented as a finite linear combination of of feature maps of inputs contained in the training set, D. The theorem was first presented in [KW71], and subsequently generalized in [SHS01]. We consider the latter generalized version.

Theorem 4.1 (Representer Theorem)**.** *Let $\mathcal{X} \neq \emptyset$ be an arbitrary set of inputs, $\mathcal{Y} \subset \mathbb{F}$ a set of corresponding targets, and $D = \{(x_i, y_i)_{i=1}^N \subset X \times \mathcal{Y}\}$ a finite set of examples. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{F}$ be any positive definite kernel, and $\Omega : [0, \infty) \to \mathbb{R}$ a strictly increasing function. For any loss function, $L : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and a class of functions taken to be the unique RKHS with kernel k, $\mathcal{F} := \mathcal{H}_k$. Then, any function $\hat{f}$ that satisfies*

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \left(\frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \Omega(\|f\|_{\mathcal{H}_k}) \right) \tag{$\dagger$}$$

also admits the representation:

$$\hat{f}(\cdot) = \sum_{i=1}^{N} \alpha_i k(\cdot, x_i).$$

Proof. First, let $S := \text{span}\{k(\cdot, x_i) \mid i = 1, \ldots, N\}$, and since $\mathcal{F}$ is a Hilbert space, we have the decomposition $\mathcal{F} = S \oplus S^{\perp}$, and so any $f \in \mathcal{F}$ may be written uniquely as:

$$f = \sum_{i=1}^{N} \alpha_i k(\cdot, x_i) + s', \qquad (\alpha_i)_{i=1}^N \subset \mathbb{F},\ s' \in S^{\perp}.$$

Then, noting that $\langle k(\cdot, x_i), s' \rangle_{\mathcal{F}} = 0$ for $i = 1, \ldots, N$, we may write for any x_j belonging to an element of D:

$$\begin{aligned}
f(x_j) &= \langle k(\cdot, x_j), f) \rangle_{\mathcal{F}} && \text{(reproducing property of } k\text{)} \\
&= \left\langle k(\cdot, x_j), \sum_{i=1}^{N} \alpha_i k(\cdot, x_i) + s' \right\rangle_{\mathcal{F}} \\
&= \left\langle k(\cdot, x_j), \sum_{i=1}^{N} \alpha_i k(\cdot, x_i) \right\rangle_{\mathcal{F}} + \langle k(\cdot, x_j), s' \rangle_{\mathcal{F}} \\
&= \sum_{i=1}^{N} \alpha_i \langle k(\cdot, x_j), k(\cdot, x_i) \rangle_{\mathcal{F}} \\
&= \sum_{i=1}^{N} \alpha_i k(x_i, x_j),
\end{aligned}$$

so that the evaluation of any $f \in \mathcal{F}$ at x_j is independent of s'. This in turn implies that the first component of (†) is therefore also independent of s'. Then, for the second component, we have:

$$\begin{aligned}
\Omega(\|f\|_{\mathcal{F}}) &= \Omega\left(\left\|\sum_{i=1}^{N}\alpha_i k(\cdot,x_i) + s'\right\|_{\mathcal{F}}\right) \\
&= \Omega\left(\sqrt{\left\|\sum_{i=1}^{N}\alpha_i k(\cdot,x_i)\right\|_{\mathcal{F}}^2 + \|s'\|_{\mathcal{F}}^2}\right) && \text{(triangle inequality)} \\
&\geq \Omega\left(\left\|\sum_{i=1}^{N}\alpha_i k(\cdot,x_i)\right\|_{\mathcal{F}}\right) && (\Omega \text{ increasing}),
\end{aligned}$$

with equality in the last line only in the case that $s' = 0$. Since the first term of (†) is independent of s', and $s' = 0$ minimizes the second term, $\hat{f}$ must be of the form $\hat{f}(\cdot) = \sum_{i=1}^{N}\alpha_i k(\cdot,x_i)$. □

4.1 RKHS Regression

We now apply the former theory in a small example of RKHS regression [VI] to demonstrate the concepts covered, and specifically to show the representer theorem at work. The example here is an implementation and extension of the discussion around RKHS regression in [FHT01, Chapter 5.8].

First, $N = 30$ examples are generated by considering a function, $f : [0,4] \to \mathbb{R}$, $f(x) = -\sin(2x) - \frac{1}{2}$, whose choice was essentially arbitrary.[VII] Randomness is induced by perturbing realizations from this function by Gaussian random noise. So, for any element (x_i, y_i) in our set of examples, we have for the x value a corresponding y value: $-\sin(2x) - \frac{1}{2} + \epsilon$ with $\epsilon \sim \mathcal{N}(0,\tau^2)$. In this case, we choose $\tau = \frac{3}{5}$ for our example, so that the observed (noisy) value of y is within $\pm\frac{3}{5}$ of the true y. Next, we nominate an arbitrary positive-definite kernel, k, and its corresponding RKHS, $\mathcal{H}_k$. We wish to find $f \in \mathcal{H}_k$ that minimizes the regularized loss (⋆). This optimization corresponds to the loss covered by the representer theorem, (†), with $\Omega(\cdot) = \lambda \times \mathcal{I}(\cdot)$.[VIII] So, we want to find $\hat{f} = \arg\min_{f\in\mathcal{H}_k} \mathcal{R}_e^L(f; N, \lambda)$, and by the representer theorem, $\hat{f}(\cdot) = \sum_{j=1}^{N}\alpha_j k(\cdot, x_j)$. Assuming we take L to be the squared error loss, this optimization problem is reduced to an optimization over $\mathbb{R}^N$ for $\alpha := (\alpha_1, \ldots, \alpha_N)^T$. To ease the notation, we consider the Gram matrix for D, denoted G, with i,j-th element equal to $k(x_i, x_j)$. We also denote the i-th row of G by $G_{i,:}$ and j-th

[VI] code included in rkhs_reg.py, this can be run from terminal with the command: 'python3 rkhs_reg.py', assuming Python version 3 or later as well as the availability of recent versions of the Numpy and Matplotlib modules.

[VII] The Python code provided generates identical plots for any user defined function on the same domain.

[VIII] $\mathcal{I}$ denotes the identity operator.

column by $G_{:,j}$. Finally, we also define $y = (y_1, \ldots, y_N)^T \in \mathbb{R}^N$. Now, we can rewrite the minimization as:

$$\begin{aligned}
&\min_{\alpha \in \mathbb{R}^N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1} \alpha_j k(x_i, x_j) \right)^2 + \lambda \left\| \sum_{j=1}^{N} \alpha_j k(\cdot, x_j) \right\|_{\mathcal{H}_k}^2 \\
&= \min_{\alpha \in \mathbb{R}^N} \sum_{i=1}^{N} (y_i - G_{i,:}\alpha)^2 + \lambda \|G_{:,j}\alpha\|_{\mathcal{H}_k}^2 \\
&= \min_{\alpha \in \mathbb{R}^N} \|y - G\alpha\|_{\mathbb{R}^N}^2 + \lambda \alpha^T G \alpha.
\end{aligned}$$

Differentiating with respect to α and setting equal to zero yields:

$$\begin{aligned}
0 &:= -2y^T G + 2G^T G\alpha + \lambda \alpha^T G \alpha, \\
\text{thus } \hat{\alpha} &= (G + \lambda I)^{-1} y,
\end{aligned}$$

where $\hat{\alpha}$ is the particular $\alpha \in \mathbb{R}^N$ corresponding to $\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i k(x, x_i)$ for all $x \in [0, 4]$.

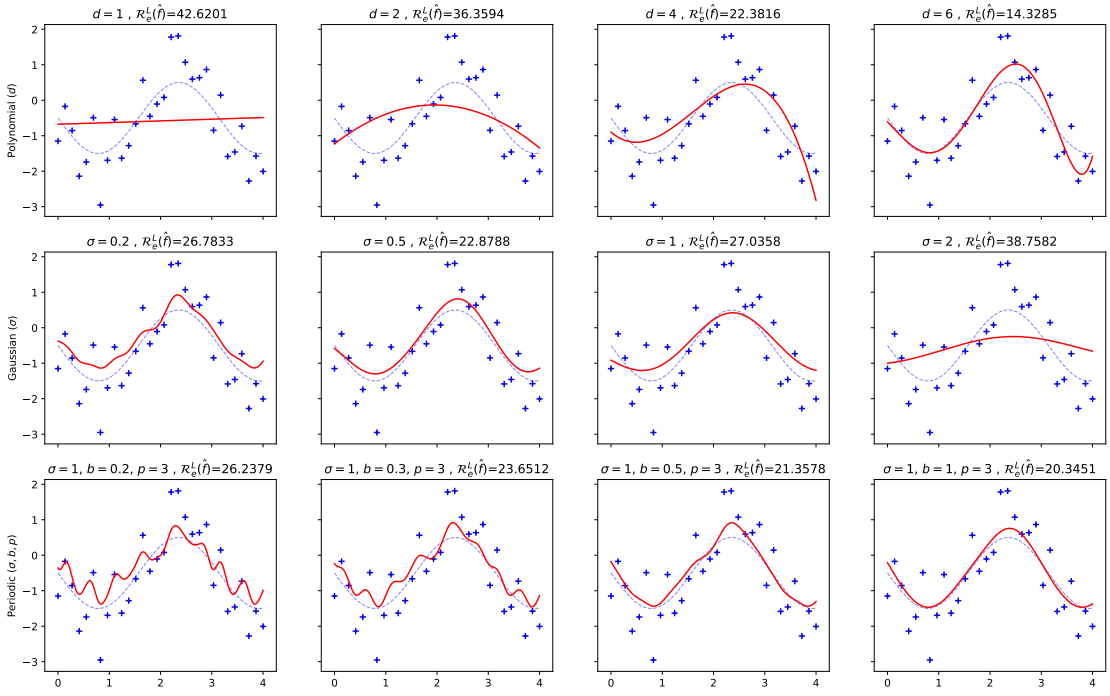

Figure 3: $\hat{f}$ under RKHS regression for various kernel hyper-parameters. $\hat{f}$ is plotted in red, the true data generating function, $f(x) = -\sin(2x) - \frac{1}{2}$ in blue

Note that the result here is quite general, and holds for *any* positive definite kernel. However, in practice there is still quite a bit of work to do around choosing the correct kernel

for the data at hand, which in particular includes choosing the correct hyper-parameters governing the kernel, as well as choosing an appropriate amount of penalization, λ. This discussion falls under the general umbrella of *cross validation*, in which all parameters, including the kernel, can be learned from the data. For our case, we will side-step this discussion and study the results under three kernels with varying parameters.

The results are outlined in Figure 3, where each row corresponds to a specific kernel, specified on the y-axis, and each cell corresponds to a specific configuration of hyper-parameter. For example, in the second cell of the second row, $\hat{f}$ defined for $\hat{\alpha}$ with Gaussian kernel with $\sigma = \frac{1}{2}$ is shown in red, with the true data generating function in light blue. Note that none of these results is optimal as we have not chosen the hyper-parameters in an optimal way, but we can nonetheless see that the periodic kernel does well, in terms of regularized loss, $\mathcal{R}_e^L(\hat{f})$, as is to be expected from the definition of the true function. We can also see that the polynomial kernel does well, however, one would expect that as the domain is taken over a larger subset than $[0, 4]$, it will be more difficult for the polynomial kernel to achieve good results relative to the periodic kernel.

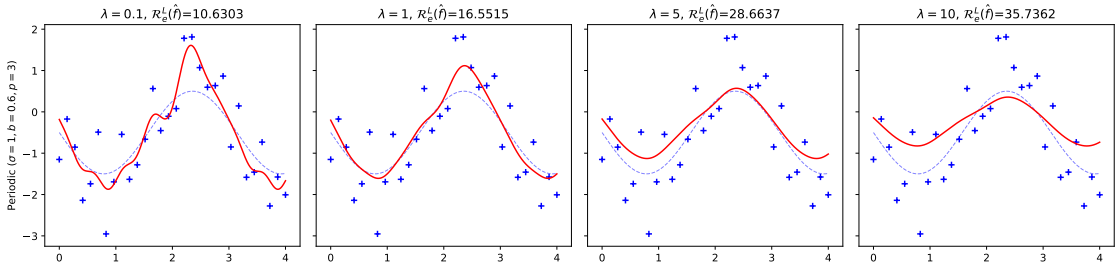

Figure 4: $\hat{f}$ under RKHS regression for various λ. $\hat{f}$ is plotted in red, the true data generating function, $f(x) = -\sin(2x) - \frac{1}{2}$ in blue

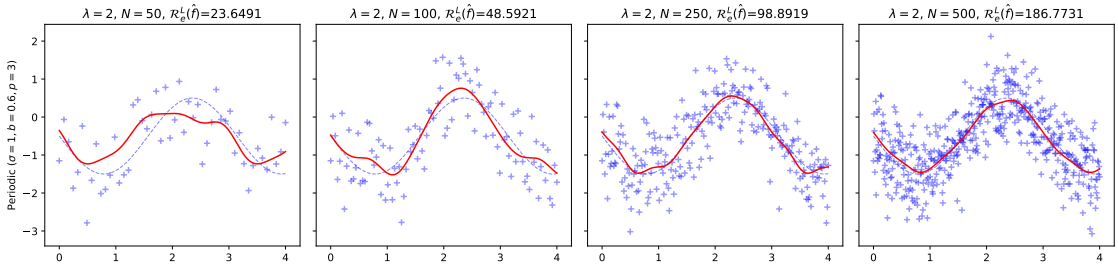

Figure 5: $\hat{f}$ under RKHS regression for various N. $\hat{f}$ is plotted in red, the true data generating function, $f(x) = -\sin(2x) - \frac{1}{2}$ in blue

Finally, we also consider the results of varying the penalization parameter, λ, on the

periodic kernel in Figure 4, as well as the effect of increasing the number of data points in our training data set, in 5. We can see that as λ is increased, the complexity of the optimal $\hat{f}$ is reduced, evidenced by smoother functions that exhibit fewer oscillations.

5 Gaussian Processes

We close the report with a look at one of the most famous kernel methods, namely the Gaussian Process (GP). The Gaussian Process can be seen as a direct extension of Bayesian Linear Regression covered in this report. Instead of considering a distribution over the parameter vector θ however, we now look at a distribution over infinite dimensional function spaces instead. This gives us a lot more flexibility in the Statistical Learning problem. The primary[IX] reference for this section is [Ras04, Chapter 2]. We begin with the formal definition of a GP.

Definition 5.1 (Gaussian Process)**.** *A Gaussian Process is a collection of random variables, any finite subset of which have a joint Gaussian distribution. A Gaussian Process defined on an input set $\mathcal{X}$, is a stochastic process, that is completely characterized by its mean and covariance functions. For any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, we write:*

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')),$$

where:

$$\begin{aligned} \mu(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))f(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned}$$

The primary focus of the GP literature is the covariance function, which can be thought of as a positive-definite kernel, with all the previous results in this report holding. For this reason, we tend to set $m(\mathbf{x}) = 0$ without any loss of generality. Stochastic Processes are often thought of as functions of time, however, here we consider the index set to be the input space of raw data. Therefore, the value $f(\mathbf{x})$ represents the value of the function at a particular input $\mathbf{x} \in \mathcal{X}$. If we consider multiple points, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathcal{X}$, then the GP evaluated at those points will be denoted as $f_1, f_2, \ldots, f_N$. Note then that from the definition, we directly have that:

$$\mathbf{f} := (f_1, f_2, \ldots, f_N)^T \sim \mathcal{N}(\mathbf{f}|\mu, \Sigma),$$

where $\mu \in \mathbb{R}^N$ is a mean vector, and $\Sigma \in \mathbb{R}^{N \times N}$ is a covariance matrix. By the definition it also follows that the GP must be consistent, in the sense that if we look at any subset $A \subset \{1, 2, \ldots, N\}$ and corresponding subset $\mathbf{f}_A$ of $\mathbf{f}$, then:

$$p(f_A) = \mathcal{N}(f_A|\mu_A, \Sigma_A),$$

[IX] For the modelling, the GPFlow Regression tutorial was used extensively.

where μ_A is the collection of means from the original mean vector μ which correspond to the indices A, and Σ_A is the appropriate sub-matrix of Σ, which is the covariance matrix if we restrict attention to the indices A.

Further, from the definition we see that the covariance matrix Σ is determined by the covariance function k. Therefore, for a set of N training examples, we would have:

$$\Sigma_k = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_N) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & k(\mathbf{x}_N, \mathbf{x}_2) & \dots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix},$$

and, identically to the case in RKHS regression, we call this the Gram matrix determined by k and the training points $\mathbf{x}_1, \dots, \mathbf{x}_N$. In order Σ_k to be a valid covariance matrix, it must be symmetric positive-definite. In order for our covariance matrix to achieve this, it must then a positive-definite symmetric function, in other words, the covariance function is simply the kernel function. We can therefore use any of the kernel families introduced earlier in modelling our GP, and this choice depends on the properties we desire our resulting approximating functions to possess.

5.1 Gaussian Process Regression

We now return to the Bayesian Regression example that motivated this report. We still consider a linear model with additive noise:

$$y = f(\mathbf{x}) + \epsilon,$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. However, instead of assuming that $f(\mathbf{x}) = \mathbf{x}^T\theta$ and performing inference on the parameter vector θ, we now consider $f(\mathbf{x})$ to follow a Gaussian Process. In other words, instead of having a zero-mean prior distribution on θ, we place a zero-mean GP prior on f, that is:

$$f(\mathbf{x}) \sim \mathrm{GP}(\mathbf{0}, k(\mathbf{x}, \mathbf{x}')).$$

To see what these functions look like, we can generate a random set of $\mathbf{x}$ values and create a corresponding covariance matrix of kernel evaluations at the generated points. We can then simply generate a vector from a multivariate Gaussian:

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \Sigma_k),$$

and plot these points.[X] We do this for various popular kernels in the GP literature with different hyper-parameter settings in Figure 6. Note that the sampled vector $\mathbf{f}$ is a combination of discrete points on the grid, and we have interpolated these points to generate the plots.

[X] code included in GP_plots.py, this can be run from terminal with the command: 'python3 GP_plots.py', assuming Python version 3 or later as well as the availability of recent versions of the Numpy, Matplotlib, GPFlow and Tensorflow modules.

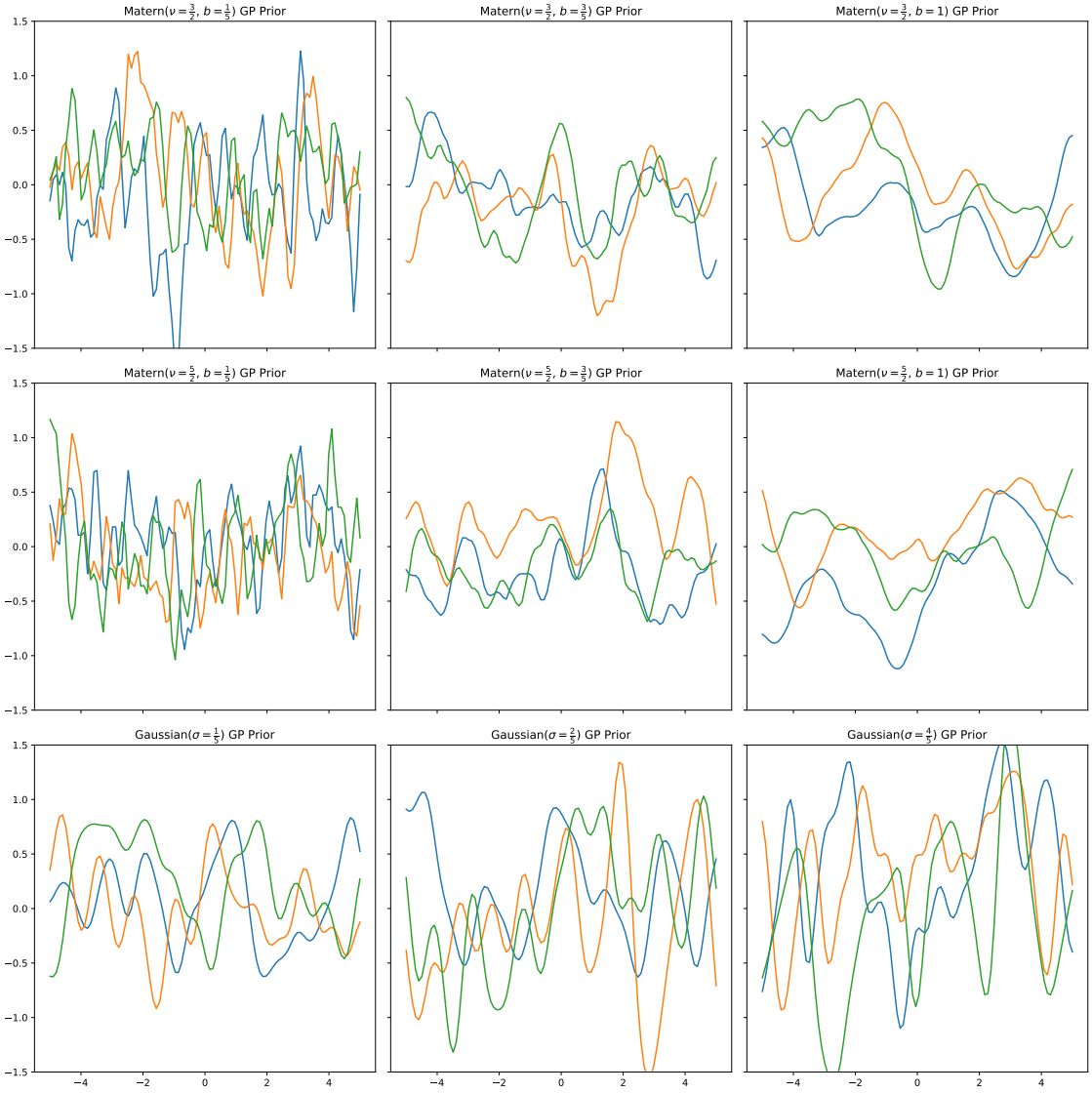

Figure 6: Random samples from Gaussian Process priors for multiple kernels.

Therefore, given a set of N training examples, $D = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^{N}$, we compute the vector achieved by constructing the $N \times N$ covariance matrix corresponding to our chosen kernel, k. We therefore have that:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, k(X, X)),$$

where the notation $k(X, X)$ denotes the covariance matrix corresponding to the points in the collection $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$ under kernel k. Under our regression assumption, we then

have that:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathcal{I}).$$

Now, recalling the following facts about the Gaussian distribution:

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \mathcal{N}(\mu|\mathbf{x}, \Sigma), \tag{3}$$

$$\mathcal{N}(\mathbf{x}|\mu_1, \Sigma_1) \times \mathcal{N}(\mathbf{x}|\mu_2, \Sigma_2) = \mathcal{N}(\mathbf{x}|\nu, \Omega)\mathcal{N}(\mu_1|\mu_2, \Sigma_1 + \Sigma_2) \tag{4}$$

where ν, Ω are some arbitrary mean vector and covariance matrices that we need not write down explicitly. We can then compute the distribution of $\mathbf{y}$ under our model:

$$\begin{aligned}
p(\mathbf{y}) &= \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} \\
&= \int \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathcal{I})\mathcal{N}(\mathbf{f}|\mathbf{0}, k(X, X))d\mathbf{f} \\
&= \int \mathcal{N}(\mathbf{f}|\mathbf{y}, \sigma^2\mathcal{I})\mathcal{N}(\mathbf{f}|\mathbf{0}, k(X, X))d\mathbf{f} && \text{by (3)} \\
&= \mathcal{N}(\mathbf{y}|\mathbf{0}, k(X, X) + \sigma^2\mathcal{I}) \int \mathcal{N}(\mathbf{f}|\nu, \Omega)d\mathbf{f} && \text{by (4)} \\
&= \mathcal{N}(\mathbf{y}|\mathbf{0}, k(X, X) + \sigma^2\mathcal{I}).
\end{aligned}$$

We now have our marginal distribution for the noisy observed data. The next step is to build predictions for a new previously unobserved observation, denoted by $\mathbf{x}_*$, with corresponding prediction value $f_* := f(\mathbf{x}_*)$. By definition, the joint distribution of any finite number of points of the GP are multivariate Gaussian, so it immediately follows that:

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(X, X) + \sigma^2\mathcal{I} & k(X, \mathbf{x}^*) \\ k(\mathbf{x}^*, X) & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right),$$

where $k(X, \mathbf{x}_*)$ denotes the vector of kernel evaluations between the observed data in the collection X, and the new data, $\mathbf{x}_*$. Given this, and the consistency of the Gaussian distribution, we now seek to find the predictive distribution. Recall the given a Gaussian vector:

$$\begin{bmatrix} w \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_w \\ \mu_z \end{bmatrix}, \begin{bmatrix} \Sigma_{zz} & \Sigma_{zw} \\ \Sigma_{wz} & \Sigma_{ww} \end{bmatrix}\right),$$

then we may partition:

$$w|z \sim \mathcal{N}(\mu_{w|z}, \Sigma_{w|z}),$$

where:

$$\begin{aligned}
\mu_{z|w} &:= \mu_w + \Sigma_{zw}^T\Sigma_{ww}^{-1}(w - \mu_w) \\
\Sigma_{z|w} &:= \Sigma_{zz} - \Sigma_{zw}^T\Sigma_{ww}^{-1}\Sigma_{zw}.
\end{aligned}$$

Applying this to our joint distribution of $[\mathbf{y}, f_*]^T$ yields:

$$f_*|\mathbf{y} \sim \mathcal{N}(\mu_{f_*|\mathbf{y}}, \Sigma_{f_*|\mathbf{y}}),$$

where:

$$\begin{aligned}\mu_{f_*|\mathbf{y}} &= \mathbf{0} + k^T(X, \mathbf{x}_*)(k(X,X) + \sigma^2\mathcal{I})^{-1}(\mathbf{y} - \mathbf{0}) \\ &= k^T(X, \mathbf{x}_*)(k(X,X) + \sigma^2\mathcal{I})^{-1}\mathbf{y},\end{aligned}$$

and:

$$\Sigma_{f_*|\mathbf{y}} = k(\mathbf{x}_*, \mathbf{x}_*) - k^T(X, \mathbf{x}_*)(k(X,X) + \sigma^2\mathcal{I})^{-1}k(X, \mathbf{x}_*).$$

Note then that the expectation can actually be written as a finite linear combination of kernel evaluations, even though the GP may be represented as an infinite expansion, just as with the RKHS regression example since:

$$\begin{aligned}\mathbb{E}(f_*|\mathbf{y}) &= k^T(X, \mathbf{x}_*)(k(X,X) + \sigma^2\mathcal{I})^{-1}\mathbf{y} \\ &= [k^T(\mathbf{x}_1, \mathbf{x}_*), k^T(\mathbf{x}_2, \mathbf{x}_*), \ldots, k^T(\mathbf{x}_N, \mathbf{x}_*)](k(X,X) + \sigma^2\mathcal{I})^{-1}\mathbf{y} \\ &= \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_*, \mathbf{x}_i),\end{aligned}$$

where we have denoted elements of the vector $(k(X,X) + \sigma^2\mathcal{I})^{-1}\mathbf{y}$ by the α terms. Note that this is a manifestation of the representer theorem in which the cost functional is taken to be the negative log-likelihood of the Gaussian. Note that we have derived here is the GP posterior process, and in analog to the Bayesian Regression case, the posterior process may be treated as our new prior when dealing with sequential updates.

We close the report by returning to our approximation of the function $f(x) = -\sin(2x) - \frac{1}{2}$ introduced in the RKHS regression example. Figure 7 displays the results of GP regression for various kernels with $N = 20$ samples with noise from our true function.[XI] The predictive distribution is sampled multiple times, and these random samples are plotted in green, whilst the mean of the predictive distribution is plotted in blue. Give that we have a distribution on our predictive distribution, we also have variance estimates that can be plotted, so that we now have an error score to go along with out predictions. The 95% confidence interval of the predictive distribution is therefore plotted in the light blue net across each plot.

[XI] code included in GP_reg.py, this can be run from terminal with the command: 'python3 GP_reg.py', assuming Python version 3 or later as well as the availability of recent versions of the Numpy, Matplotlib, GPFlow and Tensorflow modules.

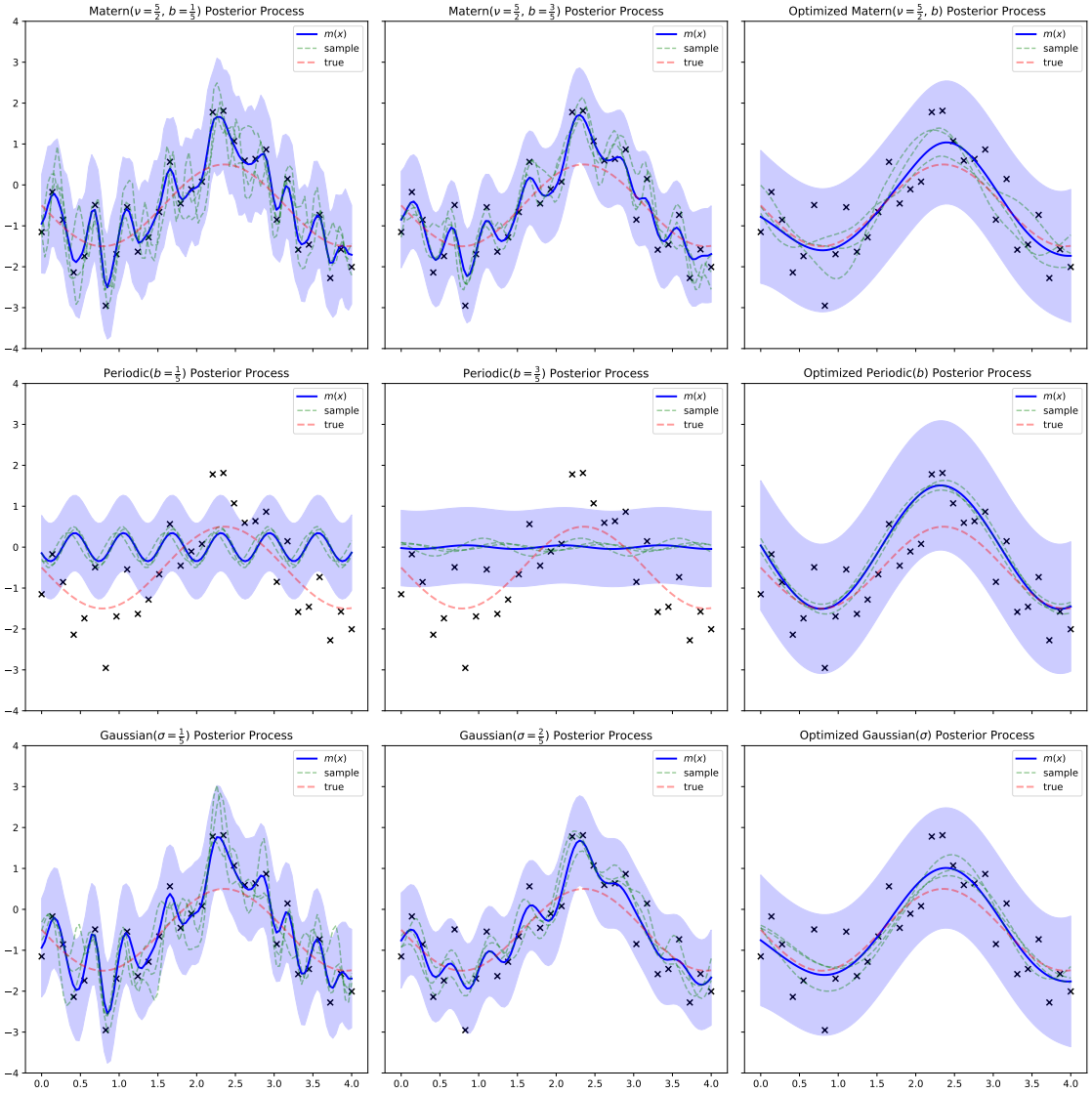

Figure 7: Gaussian Process Regression with true function $f(x) = -\sin(2x) - \frac{1}{2}$ displayed in red and various kernels. The last column corresponds to the optimized hyper-parameters of the respective kernel.

6 Conclusion

This essay was a short but detailed look at the use of Kernel methods for algorithms in Statistical Learning. We introduced and formalized the Statistical Learning problem, and used an extensive example of Bayesian Regression which was extended twice, first by ker-

nelizing the algorithm and then finally by considering a prior over functions, namely, by using a Gaussian Process. This report also covered a detailed explanation of the mathematics underlying kernel methods, by discussing the Reproducing Kernel Hilbert Space, and considering the Representer Theorem, a fundamental result that applies to a very large number of optimization problems that arise when using kernel methods. We also included multiple simulation examples of Bayesian Regression, RKHS Regression and GP Regression to illustrate the theory.

References

[Aro50] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[BTA11] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[CS02] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

[EPP00] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1, 2000.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[KHSS18] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[KW71] George Kimeldorf and Grace Wahba. Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95, 1971.

[Ras04] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[Rob14] Christian Robert. Machine learning, a probabilistic perspective, 2014.

[SHS01] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[SS09] Elias M Stein and Rami Shakarchi. *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009.