Comparing Correlation Functions and Exploring Efficiency Issues for the Gaussian Process

Ivor Walker
Supervised by Dr Michail Papathomas

Abstract

Contents

TODO Introduction

Defining the Gaussian Process

Most statistical methods describe relationships between data by estimating weights that linearly combine the data to produce an output. This is fundamentally different to Gaussian processes (GPs), which describe a distribution over possible functions that describe the data. This section aims to describe what Gaussian processes are in terms of familiar weight-based statistical tools, namely Bayesian regression. We start with an in-depth look into Bayesian linear regression, relating it to frequentist approaches and deriving important results such as the the weight and predictive distributions. By applying the linear model to some non-linear flexible function of the data $\phi(X)$, we make the model more flexible. Then, we generalise our Bayesian linear regression in terms of a distribution of possible functions that ϕ could be and translate our Bayesian linear regression into a Gaussian process. We finish by investigating general properties of Gaussian processes that are analagous to properties of Bayesian linear regression, such as the predictive distribution and log-likelihood, and describe some practical considerations involved in their usage such as training and inference.

2.1 Weight-space view

2.1.1 Standard linear model

The standard linear model summises that we have some data-generating function $f(.)$ that linearily combines training data X, parameters of some model W, and some irreduceable noise ϵ (because y is rarely a perfect observation of $f(X)$, e.g. if y is measured inaccurately) to produce an output y:

$$\begin{aligned} f(X) &= X^T W \\ y &= f(X) + \epsilon \end{aligned} \tag{1}$$

Errors The standard linear model assumes that our irreduceable noise, or errors, ϵ are drawn from a Gaussian distribution:

$$\epsilon \sim N(0, \sigma_n^2 I) \tag{2}$$

We assume our model has zero mean because all the information we have about the data-generating function $f(X)$ is contained in the weights W and the data X. We add a covariance matrix I to describe how the noise for one observation is related to the noise of another observation. In this case, we assume that the noise is independent and identically distributed (i.i.d.) across all observations, so I is an "identity matrix" where each diagonal element is 1 and all off-diagonal elements are 0. The variance σ_n^2 describes how much noise we expect in our observations.

We can combine our expressions for the data-generating function and and assumptions about our noise to produce a conditional distribution that fully describes y, also known as the likelihood of y given X and W:

$$p(y|X, W) = \mathcal{N}(X^T W, \sigma_n^2 I)$$

2.1.2 Determining weights

Our first task is to estimate the weights for our data-generating function W, as we typically do not know these in advance. Frequentist approaches focus on arriving at a single estimate of these weights ($\hat{W}$) via "maximum likelihood estimation" (MLE), whereas Bayesian approaches treat W as a random variable for which a distribution is specified.

Frequentist regression Because our likelihood $p(y|X, W)$ is Gaussian, it is at its highest density around the expected value $X^T W$ and we can use iterative optimisation routines to find a single estimate for W, $\hat{W}$, that maximises our likelihood. Because we assume $E[\epsilon] = 0$ 2, the values of W at the maximum of $p(y|X, W)$ are also the values of W that minimise the squared error $||y - X^T W||^2$. We can communicate uncertainty surrounding our estimations of W by computing "standard errors", or the ratio between the variance of errors and the variance in X. A high variance in errors shows that the variation in y is more noise than signal captured by X, but a broader range of X makes estimating W easier.

Bayesian regression Instead of producing single point estimates for weights and uncertainty in estimating them, Bayesian statistics treats W as a random variable and specifies an expected value and a variance. Placing W in a probabilistic framework allows us to propagate uncertainty throughout the model and to encode beliefs (e.g. from domain experts) about the weights before observing the data.

We start with a "prior" distribution of W, which the Bayesian linear model assumes:

$$p(W) = N(0, \Sigma_p) \tag{3}$$

Then, we observe the data and update our beliefs about the weights using Bayes' theorem to produce a "posterior" distribution $p(W|X, y)$:

$$p(W|X, y) = \frac{p(y|X, W)p(W)}{p(y|X)}$$

$p(y|X, W)$ is the density of the residuals after applying $p(W)$ to X, W under our assumed noise model ϵ, and $p(y|X)$ is our likelihood.

Deriving our posterior To understand the relationship between $p(W|X, y)$ and W, we can ignore terms that do not vary with W (e.g. our marginal likelihood) by absorbing them into a proportionality constant:

$$p(W|X, y) \propto p(y|X, W)p(W) \tag{4}$$

The probability density function (PDF) measures probability per unit length. Integrating the PDF over a range gives us the probability of observing a value in that range. The PDF of a Gaussian distribution is given by:

$$N(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{5}$$

We can get a PDF for our likelihood by representing our errors in squared error form and substituting it into the Gaussian PDF 5:

$$p(y|X, W) = exp\left(-\frac{1}{2\sigma_n^2}||y - X^TW||^2\right) \tag{6}$$

Reframing $p(W)$ as a PDF:

$$p(W) = \frac{1}{[\sqrt{\sigma_p}]\sqrt{2\pi}} exp\left(-\frac{1}{2}\frac{([W]-[0])}{[\Sigma_p]}\right)$$

The first term can be absorbed into the proportionality constant. Rewriting the second term as a negative exponential:

$$p(W) \propto exp\left(-\frac{1}{2}W^T\Sigma_p^{-1}W\right) \tag{7}$$

Putting both expressions for 6 and 7 into 4:

$$p(W|X, y) \propto exp\left(-\frac{1}{2\sigma_n^2}||y - X^TW||^2\right) exp\left(-\frac{1}{2}W^T\Sigma_p^{-1}W\right)$$

Expanding $||y - X^TW||^2$ to $y^Ty - 2y^TXW + W^TX^TXW$:

$$p(W|X, y) \propto exp\left(-\frac{1}{2\sigma_n^2}(y^Ty - 2y^TXW + W^TX^TXW)\right) exp\left(-\frac{1}{2}W^T\Sigma_p^{-1}W\right)$$

Putting both exponentials together by adding their powers:

$$p(W|X, y) \propto exp\left(\frac{1}{\sigma_n^2}(y^Ty - 2y^TXW + W^TX^TXW) + \left(-\frac{1}{2}W^T\Sigma_p^{-1}W\right)\right)$$

Rearranging the inside term to be a quadratic, linear and constant term in W:

$$p(W|X, y) \propto exp\left(\frac{1}{2}W^T\left(\frac{1}{\sigma_n^2}X^TX + \Sigma_p^{-1}\right)W - \left(\frac{1}{\sigma_n^2}y^TX\right)W + \frac{1}{2}y^Ty\right)$$

We can ignore the constant final term. Introducing these terms to simplify this result:

$$\begin{aligned} A &= \Sigma_p^{-1} + \frac{1}{\sigma_n^2}X^TX \\ b &= \frac{1}{\sigma_n^2}y^TX \\ p(W|X, y) &\propto exp\left(-\frac{1}{2}W^TAW + b^TW\right) \end{aligned} \tag{8}$$

Deriving the properties of the posterior by completing the square Now we have a simplified form for the posterior's PDF, we need to get it into a Gaussian form to recover the properties of the posterior distribution.

Bringing all terms inside the exponential to a single term:

$$-\frac{1}{2}W^TAW + b^TW = \frac{1}{2}\left(-W^TAW + 2b^TW\right)$$

Completing the square on our new inner term $W^TAW - 2b^TW$

$$W^TAW - 2b^TW = (W - A^{-1}b)^TA(W - A^{-1}b) - b^TA^{-1}bp(W|X, y) \propto exp\left(-\frac{1}{2}\left((W - A^{-1}b)^TA(W - A^{-1}b) - b^TA^{-1}b\right)\right) \tag{9}$$

Our expression lines up with the Gaussian PDF's "kernel" term $exp\left(-\frac{1}{2}(W-\mu)^T\Sigma^{-1}(W-\mu)\right)$, where $\mu = A^{-1}b$ and $\Sigma = A^{-1}$ ($\Sigma^{-1} = A$). Therefore, 9 can be represented as a Gaussian distribution:

$$p(W|X, y) \sim N(A^{-1}b, A^{-1}) \tag{10}$$

Inside our definition of A at **??**, we are missing an expression for Σ_p. Assuming independence of noise under the linear model 2, our weight variance Σ_p under the Bayesian linear model is "isotropic", meaning it is the same in all directions.

$$\Sigma_p = \tau^2 I$$

I is our familiar identity matrix and τ^2 is a scalar variance term, chosen as a prior.

Substituting the isotropic prior Σ_p into A:

$$A = \Sigma_p^{-1} + \frac{1}{\sigma_n^2} X^T X = \left[\tau^2 I\right]^{-1} + \frac{1}{\sigma_n^2} X^T X = \frac{1}{\tau^2} I + \frac{1}{\sigma_n^2} X^T X$$

Simplifying:

$$A = \frac{1}{\sigma_n^2} \left(X^T X + \frac{\sigma_n^2}{\tau^2} I \right) \tag{11}$$

Gaussian posteriors and ridge regression So far we have worked exclusively within the Bayesian paradigm, but we can draw some value of W from our posterior distribution to relate it to a frequentist framework. For Gaussian posteriors, our expected value of W $A^{-1}b$ is also its mode. This is called the maximum a posteriori (MAP) estimate of W, and is due to symmetries in linear model and posterior and is not the case in general. Our MAP estimate does not matter within the Bayesian framework but is equivelant to our frequentist $\hat{W}$.

Substituting our full expressions for A 11 and b 8 into our MAP estimation:

$$W_{\text{MAP}} = A^{-1} b = \left[\frac{1}{\sigma_n^2} (X^T X + \frac{\sigma_n^2}{\tau^2} I \right]^{-1} \cdot \left[\frac{1}{\sigma_n^2} y^T X \right]$$

Inverting LHS term of A:

$$A^{-1} = \frac{\sigma_n^2}{X^T X + \frac{\sigma_n^2}{\tau^2} I} = \sigma_n^2 \left(X^T X + \frac{\sigma_n^2}{\tau^2} I \right)^{-1}$$

Substituting this back into W_{MAP} cancels out the σ_n^2 term in A with the $\frac{1}{\sigma_n^2}$ term in B:

$$W_{\text{MAP}} = \sigma_n^2 \left(X^T X + \frac{\sigma_n^2}{\tau^2} I \right)^{-1} \cdot \frac{1}{\sigma_n^2} y^T X = \left(X^T X + \frac{\sigma_n^2}{\tau^2} I \right)^{-1} \cdot y^T X$$

This is equivelant to the solution to ridge regression, where $\lambda = \frac{\sigma_n^2}{\tau^2}$.

$$W_{\text{ridge}} = \left(X^T X + \lambda I \right)^{-1} X^T y$$

Ridge regression introduces some bias to lower the variance in a frequentist linear model by shrinking weights, where the λ term controls the amount of shrinkage applied to the weights. This is traditionally useful where variance is particularly high (e.g. multicollinearity) and can be reduced at the cost of little bias.

Our MAP estimation in Bayesian linear regression with isotropic priors is equivalent to ridge regression, where the amount of bias we introduce depends on our confidence in our priors. The more we trust our prior, the higher our λ and the more we shrink our weights towards zero. A lower τ means we are more confident in $p(W)$ and have better priors, whereas a higher σ_n means lower confidence in $p(y|X, W)$ and worse weights that should be shrunk closer to zero.

2.1.3 Predictive distribution

Deriving the predictive distribution Our second task is to make predictions y_* using new input data X_* and our previously learned weights W. Frequentist methods simply multiply $\hat{W}$ by X_*, but this does not propagate uncertainty in W. In this Bayesian framework, we form a "predictive distribution" which we sample from to get our noise-free function evaluations $f(X_*)$ (denoted f_*) and add ϵ to get our noisy predictions y_*.

$$p(f_*|X_*, X, y) = \int p(f_*|X_*, W) \cdot p(W|X, y) dW$$

$p(f_*|X_*, W)$ is what we think the function looks like after producing a prediction using X_* and perfect knowledge of W. $p(W|X, y)$ is our familiar 10 posterior distribution of weights. $p(f_*|X_*, W) \cdot p(W|X, y)$ is the joint distribution of our predictions and our posterior weights, which gets us the conditional distribution $p(f_*, W|X_*, X, y)$ by definition of conditional probability. Because $p(f_*, W|X_*, X, y)$ relies on our perfect knowledge of W, which we lack, we integrate over all possible W to get the final predictive distribution $p(f_*|X_*, X, y)$

$p(f_*|X_*, W)$ is our error distribution, which we assume to be distributed normally and independently with our I identity matrix:

$$p(f_*|X_*, W) = \mathcal{N}(f_*|W^T X_*, \sigma_n^2 I)$$

Substituting into 5 and absorbing the LHS term into the proportionality constant:

$$p(f_*|X_*, w) \propto exp \left(-\frac{1}{2} \frac{1}{\sigma_n^2} (f_* - W^T X_*)^2 \right)$$

Multiplying $P(f_*|X_*, W)$ and $p(W|X, y)$ to get our conditional $p(f_*, W|X_*, X, y)$, and add the exponents:

$$p(f_*, W|X_*, X, y) \propto exp\left(\frac{1}{2}(-W^T AW + 2b^T W) + \left(-\frac{1}{2}\frac{1}{\sigma_n^2}(f_* - W^T X_*)^2\right)\right)$$

Combining the terms inside the exponent:

$$p(f_*, W|X_*, X, y) \propto exp\left(-\frac{1}{2}\left(W^T AW - 2b^T W + \frac{1}{\sigma_n^2}(f_* - W^T X_*)^2\right)\right)$$

Expanding the squared term:

$$p(f_*, W|X_*, X, y) \propto exp\left(-\frac{1}{2}\left(W^T AW - 2b^T W + \frac{1}{\sigma_n^2}(f_*^2 - 2f_* W^T X_* + W^T X_* X_*^T X_*)\right)\right)$$

Similar to our posterior, we can rearrange this to be a quadratic, linear and constant term in W:

$$p(f_*, W|X_*, X, y) \propto exp\left(-\frac{1}{2}\left(W^T\left(A + \frac{1}{\sigma_n^2}X_* X_*^T\right)W - 2\left(b + \frac{1}{\sigma_n^2}f_* X_*\right)^T W + \frac{1}{\sigma_n^2}f_*^2\right)\right) \tag{12}$$

We can define new terms A_* and b_* to simplify this expression:

$$A_* = A + \frac{1}{\sigma_n^2}X_* X_*^T$$
$$b_* = b + \frac{1}{\sigma_n^2}f_* X_*$$

Substituting into 12:

$$p(f_*, W|X_*, X, y) \propto exp\left(-\frac{1}{2}\left(W^T A_* W - 2b_*^T W + \frac{1}{\sigma_n^2}f_*^2\right)\right)$$

Integrating out W to get our predictive distribution:

$$p(f_*|X_*, X, y) = \int p(f_*, W|X_*, X, y)dW \propto \int exp\left(-\frac{1}{2}\left(W^T A_* W - 2b_*^T W + \frac{1}{\sigma_n^2}f_*^2\right)\right)dW \tag{13}$$

Factoring out $\frac{1}{\sigma_n^2}f_*^2$ as it does not depend on W (since $\int exp(X)dX = exp(X)$):

$$= exp\left(-\frac{1}{2}\frac{1}{\sigma_n^2}f_*^2\right) \times \int exp\left(-\frac{1}{2}\left(W^T A_* W - 2b_*^T W\right)\right)dW$$

Evaluating the RHS multivariate Gaussian integral:

$$\int exp\left(-\frac{1}{2}\left(W^T A_* W - 2b_*^T W\right)\right)dW = \frac{(2\pi)^{D/2}}{\sqrt{|A_*|}}exp\left(\frac{1}{2}b_*^T A_*^{-1} b_*\right)$$

Substituting back into 13:

$$p(f_*|X_*, X, y) \propto exp\left(-\frac{1}{2}\frac{1}{\sigma_n^2}f_*^2\right) + \frac{(2\pi)^{D/2}}{\sqrt{|A_*|}} \cdot exp\left(\frac{1}{2}b_*^T A_*^{-1} b_*\right)$$

Now that no part of our expression is dependent on W, we need an expression for everything that depends on f_*.

Absorbing the second term (since it does not depend on f_*) into the proportionality constant, and combining the remaining exponential terms by adding their powers:

$$p(f_*|X_*, X, y) \propto \exp\left(-\frac{1}{2}\frac{1}{\sigma_n^2}f_*^2 + \frac{1}{2}b_*^T A_*^{-1} b_*\right)$$

Similar to deriving properties from our posterior, we can rearrange this expression and complete the square to derive the properties of our predictive distribution:

$$p(f_*|X_*, W) \sim N(X_*^T A^{-1} b, X_*^T A^{-1} X_*) \tag{14}$$

The variance is quadratic in X_* with A^{-1}, showing that predictive uncertainties grow with size of X_*.

2.1.4 Projections of inputs into feature space

One problem with this model is that it assumes a linear relationship between X and y. We can project our inputs into a higher dimensional feature space and apply a linear model in this space to express non-linear relationships between X and y.

Defining $\phi(X)$ as a function that maps a D-dimensional input vector X into an N dimensional feature space, our standard linear model becomes:

$$f(X) = \phi(X)^T W$$

For example, a scalar x could be projected into the space of powers of x: $\phi(x) = [1, x, x^2, \dots, x^d]^T$ for a polynomial basis expansion of degree d to represent a d-power relationship between x and y. . Substituting $\phi(X)$ for X in 14:

$$p(f_*|X_*, X, y) = N(\phi(X_*)^T A_\phi^{-1} b_\phi, \phi(X_*)^T A_\phi^{-1} \phi(X_*)) \tag{15}$$

Where A_ϕ and b_ϕ are now:

$$A_\phi = \Sigma_p^{-1} + \frac{1}{\sigma_n^2}\phi(X)^T\phi(X) b_\phi = \frac{1}{\sigma_n^2}\phi(X)^T y$$

2.1.5 Computational issues

Avoiding inversion of A_ϕ 15 requires inverting the $N \times N$ matrix A_ϕ, where N is dimension of feature space, to get the expected value and variance.

Typically, matrices are inverted using Gaussian elimination. We need to perform a "forward" pass which requires N pivots on every row and column, N eliminations per pivot, and up to $2N$ columns to update, resulting in an $O(N^3)$ time complexity. Then, we need to perform a backwards pass in the opposite direction which is another $O(N^3)$ operation. Finally, we need to multiply the inverse by the RHS vector b_ϕ, which is an $O(N^2)$ operation but appears trivial next to these two cubic steps.

We can mitigate this for a particular class of high-dimensional $N > n$ problems by restating the predictive distribution in terms of the number of training data points n which would require inverting an $n \times n$ matrix instead. For polynomial basis expansions, N is degree D multiplied by number of features, so N can be very large or even infinite (e.g. SE).

Substituting b_ϕ into our predictive distribution mean:

$$\mathbb{E}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T \cdot A_\phi^{-1} \cdot \left[\frac{1}{\sigma_n^2}\phi(X)^T y\right]$$

Rearranging to isolate $A_\phi^{-1}\phi(X)$

$$= \frac{1}{\sigma_n^2}\left[A_\phi^{-1}\phi(X)\right]^T y$$

We can use the Sherman-Morrison-Woodbury identity (SMW) to get an expression for A_ϕ^{-1} directly, where $K = \phi(X)^T \Sigma_p \phi(X)$, since TODO rank

$$A_\phi^{-1} = \Sigma_p - \Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1}\phi(X)^T\Sigma_p$$

For the mean, we can use the SMW identity again to get an expression for $A_\phi^{-1}\phi(X)$

$$A_\phi^{-1}\phi(X) = \sigma_n^2 \Sigma_p \phi(X)(K + \sigma_n^2 I)^{-1} \tag{16}$$

Substitute in 16 into our 15:

$$\mathbb{E}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)\frac{1}{\sigma_n^2}\left[\sigma_n^2\Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1}\right]^T y$$

$\frac{1}{\sigma_n^2}$ and σ_n^2 cancel out, leaving us with this final expression for the mean:

$$\mathbb{E}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T \cdot \Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1} y \tag{17}$$

For the variance, we cannot use the Sherman-Morrison identity to arrive at an expression for $A_\phi^{-1}\phi(X_*)$ because $\phi(X_*)$ is an arbitrary N-vector, not one of the columns of $\phi(X)$. Instead, we use the A_ϕ^{-1} expression we derived earlier to get an expression for $A_\phi^{-1}\phi(X_*)$:

$$A_\phi^{-1}\phi(X_*) = \Sigma_p \cdot \phi(X_*) - \Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1}\phi(X)^T\Sigma_p \cdot \phi(X_*)$$

Substituting this into 15:

$$\mathrm{Var}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T\Sigma_p\phi(X_*) - \phi(X_*)^T\Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1}\phi(X)^T\Sigma_p\phi(X_*) \tag{18}$$

With our alternative mean 17 and variance 18, we can form an alternative expression for our predictive distribution:

$$\begin{aligned} p(f_*|X_*, X, y) = \mathcal{N}(\\ \phi(X_*)^T\Sigma_p\phi(X)(K+\sigma_n^2I)^{-1}y, \\ \phi(X_*)^T\Sigma_p\phi(X_*) - \phi(X_*)^T\Sigma_p\phi(X)(K+\sigma_n^2I)^{-1}\phi(X)^T\Sigma_p\phi(X_*) \\ ) \end{aligned} \tag{19}$$

With this alternative formulation, we need to invert the $n \times n$ matrix $K + \sigma_n^2 I$ only. Geometrically, n datapoints can span at most n dimensions in the feature space - if $N > n$, the data forms a subspace of the feature space.

Kernels and the kernel trick In 19, $\phi(.)$ is always an inner product of a positive definite correlation matrix Σ_p, but with different arrangements of $\phi(X)$ and $\phi(X_*)$. We can define $k(X, X') = \phi(X)^T\Sigma_p\phi(X')$ as a covariance function or kernel, where X and X' are either X or X_*. For example, in 19 the definition of $K = \phi(X)^T\Sigma_p\phi(X)$ becomes $K = k(X, X)$.

Introducing $\psi(X)$ to better represent $k(X, X')$ as an inner product:

$$\psi(X) = \phi(X)\Sigma_p^{1/2}$$
$$k(X, X') = \psi(X)^T\psi(X')$$

These inner product representations require us to compute $\phi(X)$ and $\phi(X')$ in the feature space. A higher-dimensional feature space requires more compute to evaluate $\phi(X)$ and more memory to store $\phi(X)$ and $\phi(X')$.

Instead, the representer theorem guarantees that we can find an equivelant kernel that does not require us to explicitly compute $\phi(X)$ or $\phi(X')$ in the feature space. With this "kernel trick" we avoid the associated memory and computational costs of explicitly computing $\phi(X)$ and $\phi(X')$. Since computing the kernel directly is more convenient than the feature vectors themselves, these kernels become the object of primary interest.

For example, if we had some polynomial transformation $\phi(X) = [1, x^1, ..., x^D]^T$ and Σ_p as an identity matrix, we could define $k(X, X')$ as inner products:

$$\psi(X) = [1, x^1, ..., x^D]^T k(X, X') = \psi(X)^T\psi(X')$$

This approach requires arranging ϕ and $\phi(X')$ into a D sized vector, then taking the dot product. This is trivial for small D, but as D becomes infinite (e.g. RBF kernel), arranging a D sized vector requires too much memory and the dot product becomes computationally expensive.

Instead, we can define $k(X, X')$ as an equivelant function of X and X' directly:

$$k(X, X') = (1 + X \cdot X')^D$$

This is the polynomial kernel, which is equivalent to our original polynomial basis expansion $\phi(X)$ without explicitly computing $\phi(X)$.

2.2 Function-space view [11]

2.2.1 Gaussian processes (GP)

Bayesian linear model We can define our Bayesian linear model of a real process $f(X)$ entirely in terms of mean function $m(X)$ and covariance function $k(X, X')$:

$$\begin{aligned} m(X) = \phi(X)^T\mathbb{E}[W] = \phi(X)^T[0] = 0 \\ k(X, X') = \phi(X)^T\mathbb{E}[WW^T]\phi(X') = \phi(X)^T\Sigma_P\phi(X') \end{aligned} \tag{20}$$

Our covariance function here is in inner product form. The kernel trick here uses the squared exponential (SE) covariance function, also known as the radial basis function (RBF) or Gaussian kernel:

$$k(f(X), f(X')) = \exp\left(-\frac{1}{2}\frac{|X - X'|^2}{l^2}\right)$$

It can be shown that SE corresponds to a Bayesian linear regression model with infinite basis functions.

Function evaluations to a random function We can choose a subset X_{*1} from our test data X_* and apply it to our model get some function evaluations $f(X_{*1})$. $f(X_{*1})$ can be described as a multivariate Gaussian distribution, e.g. in the Bayesian linear model $f(X_{*1}) \sim N(0, k(X_{*1}, X_{*1}))$. Each output $f(X_{\theta*1})$ in our $f(X_{*1})$ vector is a random variable with mean 0 and covariance with each other $K_{\theta\theta'} = k(X_{\theta*}, X_{\theta'*})$. There exists some random function $g(X_{*1})$ for our subsets such that $f(X_{*1}) = g(X_{*1})$. We only know the value of $g(X_{*1})$ at the points X_{*1}, so $g(X_{*1}) = X_{*1} : f(X_{*1})$. Because $g(X)$ entirely consists of random points, we can think of $g(X_{*1})$ as a random function and our distribution $f(X)$ can be seen as a distribution of these random $g(X)$ functions. We can recover our individual $g(X_{*1}$ thanks to consistency - if we marginalised out our subset from the entire distribution $f(X_*)$, we would recover the subset distribution $N(0, K_*(X_{*1}, X_{*1}))$ that describes our random function $g(X_{*1})$.

Definition of a GP A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. Ultimately, GPs describe a distribution of random functions where each drawn function is a $g(X)$ sample from the GP.

$$f(X) \sim \mathcal{GP}(m(X), k(X, X')),$$
$$m(X) = \mathbb{E}[f(X)],$$
$$k(X, X') = \text{Cov}(f(X), f(X')) = \mathbb{E}[(f(X) - m(X))(f(X') - m(X'))]$$

Consistency requirement This definition implies a consistency requirement - any group of functions drawn from our GP can be described by the same distribution as our GP. For example, if our GP implies that $(f(X_1), f(X_2)) \sim \mathcal{N}(\mu, \Sigma)$, then $(f(X_1) \sim \mathcal{N}(\mu_1, \Sigma) and f(X_2) \sim \mathcal{N}(\mu_2, \Sigma))$ where $\mu_\theta = m(X_\theta)$ and $\Sigma_{\theta\theta} = k(X_\theta, X_\theta)$. This requirement is also called the marginalisation property, because to get the smaller distribution of $f(X_1)$ we marginalise out the larger distribution of $f(X_1), f(X_2)$ by integrating the larger distribution wrt $f(X_2)$. Consistency is automatically gained if our covariance function specifies entries in a covariance matrix.

2.2.2 Predictive distributions with noise-free observations

Prior distribution over functions $f(X)$ and $f(X_*)$ are jointly distributed according to the prior:

$$\begin{pmatrix} f(X) \\ f(X_*) \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right) \tag{21}$$

Posterior distribution of functions To get the posterior distribution of functions given the training data and our prior, we can condition the joint prior distribution on the training data. Intuitively, this is like generating random functions $g(X)$ and rejecting those that do not pass through the training data. Probabilistically, we condition our joint Gaussian prior distribution on the observations $p(f(X_*)|X_*, X, f(X))$.

Substituting $p(W)$ and our conditioning X into the Gaussian multivariate conditioning identity:

$$\begin{aligned} p(f(X_*)|X_*, X, f(X)) \sim N(\\ [0] + [K(X_*, X)][K(X, X)]^{-1}([f(X)] - [0]), \\ [K(X*, X*)] - [K(X_*, X)][K(X, X)]^{-1}[K(X, X_*)] \\ ) \end{aligned}$$

Although we condition on X_*, X, and $f(X)$, we only substitute $f(X)$ because X_* and X are known constants, but $f(X)$ is random because it is a sample from the prior. We also swap $f(X_*)$ and $f(X)$ in our prior to match the conditioning identity, such that our input vector into the conditioning identity is $(f(X_*), f(X))^T$.

Simplifying the last term in the mean:

$$\begin{aligned} p(f(X_*)|X_*, X, f(X)) \sim N(\\ K(X, X_*)K(X, X)^{-1}f(X), \\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*) \\ ) \end{aligned} \tag{22}$$

2.2.3 Predictive distributions with noisy observations

Noisy observations prior It is typical to not have the noise-free function evaluations $f(X)$ as our training data, but instead our noisy observations y. We can simply add ϵ:

$$\text{Cov}(y_p, y_q) = K(X_p, X_q) + \sigma_n^2 \delta_{pq}$$

δ_{pq} represents our independence condition in 1D. This is the Kronecker delta, which returns 1 if indices (p, q) are equal and 0 otherwise. σ_n^2 is the noise variance, which is a constant for all observations.

In matrix form:

$$\text{Cov}(Y) = k(X, X) + \sigma_n^2 I$$

This gives us this prior:

$$\begin{pmatrix} Y \\ f(X_*) \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right)$$

Noisy observations posterior As before, we can form a predictive distribution using the Gaussian multivariate conditioning identity:

$$\begin{aligned} p(f(X_*)|X_*, X, Y) \sim N(\\ K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} Y, \\ K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*) \\ ) \end{aligned}$$

Substituting $k(X,X') = \phi(X)^T\Sigma_p\phi(X')$ into here gives us the exact same result as 19.

Our variance is independent of the targets y and only depends on our inputs X and X_*. Our variance is two terms: our prior covariance $K(X_*,X_*)$, a term representing the information the observations give us about the function. As before, we can compute the predictive distribution of y_* by adding the noise term $\sigma_n^2 I$ to the variance.

2.2.4 Marginal likelihood

Even though we are working within the Bayesian paradigm, certain practical subroutines e.g. kernel hyperparameter optimisation algorithms need a likelihood to maximise. The marginal likelihood $p(Y|X)$ is a measure of how well our GP fits the data:

$$p(Y|X) = \int p(Y|f,X)p(f|X)df \tag{23}$$

$p(f|X)$ is our familiar prior distribution over weights $\sim N(0,K)$ which we use here to represent the complexity of f. $p(y|f,X)$ is the likelihood of our observations given $p(y|f,X) \sim N(f,\sigma_n^2 I)$, and represents how well f maps X to y.

Our weight's mean will always be the same under our prior, but our $K(X,X')$ tells us how "wiggly" our function is. The closer our $p(f|X)$ distribution is to the true complexity of the function, the higher our marginal likelihood. For example, for SE if our data is close together, then $|X - X'|$ becomes small and our covariance $k(X,X')$ on our function distribution prior is large. Therefore, we get a high variety of functions and a higher probability of sampling a more complex function.

We can express $p(Y|X)$ as a Gaussian integral over the joint distribution of f and Y ($p(Y,f|X)$), and marginalise out f to get this PDF:

$$logp(Y|X) = -\frac{1}{2}Y^T(K(X,X)+\sigma_n^2 I)^{-1}Y - \frac{1}{2}log|K+\sigma_n^2 I| - \frac{n}{2}log(2\pi) \tag{24}$$

Alternatively, from 1 we know that y is Gaussian. Since both y and f are Gaussian, we can simply add their means and variances:

$$p(Y|X) = N(0, K + \sigma_n^2 I)$$

We can plug these mean and variances into the Gaussian PDF 5 to get 24. Assembling the likelihood is necessary for training to select hyperparameters, but the $K(X,X)+\sigma_n^2 I$ inversion costs $O(n^3)$ to compute.

2.2.5 Algorithm for Gaussian processes

Here is an algorithm for computing the predictive distribution for predictions and log-likelihood for training that uses Cholesky decomposition **??** to address the matrix inversion requirement.

1. Take in inputs X, outputs y, covariance function k, noise level σ_n^2, and test input X_*
2. $L = \text{cholesky}(K(X,X) + \sigma_n^2 I)$
   - Compute the Cholesky factor 38 for the $[K(X,X)+\sigma_n^2 I]$ matrix that needs to be inverted for training and inference
3. $alpha = L^T \backslash (L \backslash y)$
   - Find $[K(X,X)+\sigma_n^2 I]^{-1}y$ using Cholesky decomposition **??**, which is equivelant to solving the linear system $L^T L \cdot alpha = y$.
4. $mu = K(X_*,X)^T \cdot alpha$
   - Compute the mean
5. $v = L \backslash K(X_*,X)^T$
   - Compute $v = L^{-1}K(X_*,X)$
6. $var = K(X_*,X_*) - v^T v$
   - Plug v into the variance restated in terms of the Cholesky factors, where $v^T v = (L^{-1}K(X_*,X))^T(L^{-1}K(X_*,X)) = K(X_*,X)^T[K(X,X)+\sigma_n^2 I]^{-1}K(X_*,X)$
7. $\text{log}p(Y|X) = -\frac{1}{2}y^T \cdot alpha - \frac{1}{2}log|K(X,X)+\sigma_n^2 I| - \frac{n}{2}log(2\pi)$
   - Compute the log marginal likelihood
8. Return the mean mu, variance var, and log marginal likelihood

\ denotes a linear system solver, i.e. $L\backslash y$ solves the linear system $Lx = y$ for x.

Exploring Covariance Functions

The behaviour of the covariance function and its resulting covariance matrix is crucial to understanding the behaviour of the Gaussian process - the covariance matrix appears in every term of the predictive distribution and the marginal likelihood. In particular, the choice of covariance function affects the smoothness of the overall GP. This section describes the mechanisms that the covariance function has at its disposal to vary smoothness, namely its degree of differentiability and the choice of length scale. We investigate in detail the stationary family of covariance functions, discuss why they are the most widely used, and develop a new tool to analyse them. Finally, we introduce the most widespread stationary covariance functions, their properties, and the types of data they are best suited for.

3.1 Characteristics of covariance functions [11]

3.1.1 Covariance matrices

The inner product representations of the covariance for our Bayesian model in function-space 20, weight-space 19, and covariance matrices in general can be represented as a Gram matrices. Given a vector of input points $X_i | i = 1, ..., n$ and a covariance function k, a Gram matrix K is the $n \times n$ matrix whose (i, j)-th entry is $k(x_i, x_j)$. This requires any proposed covariance matrix to exhibit the properties of a Gram matrix, namely symmetry and positive semidefiniteness:

$$K_{ij} = K_{ji}$$
$$X^T K X \geq 0$$

Symmetry is inherent from the inner product property since inner products are symmetric, i.e. $X_i^T X_j = X_j^T X_i$, and imposes $cov(X_i, X_j) = cov(X_j, X_i)$. K satisfies positive semidefiniteness if, for any vector V of inputs, $V^T K V \geq 0$ since:

$$V^T K V = (KV)^T (KV) = ||KV||^2 \geq 0$$

This means that the covariance matrix must not produce negative variances for any linear combination of inputs.

3.1.2 Eigenvalue and eigenfunctions of covariance matrices

Integral operators If we wanted to multiply a vector Φ by a matrix $K(X, X')$, we would use the following matrix-vector multiplication to produce a discrete vector $[K\phi]$:

$$[K\phi]_i = \sum_{j=1}^{n} K_{ij} \Phi_j$$

If we wanted to multiply a function $\phi(x)$ by our matrix $K(X, X')$, we could use an "integral operator" K to produce a new contnuous function $[K\phi](x')$:

$$[K\phi](x') = \int K(x, x')\phi(x)dx$$

This specific usage of $K(X, X')$ is where covariance matrices are known as kernels, but the terms "kernel" and "covariance matrix" are used interchangably outside of this context. For example, using the linear kernel $K(x, x') = x \cdot x'$, a function $\phi(x) = x^2$, and the limits of our domain $a = 0$ and $b = 1$:

$$[K\phi](x') = \int x \cdot x' x^2 dx = x' \int_0^1 x^3 dx = x' [\frac{x^4}{4}]_0^1 = x' \cdot \frac{1}{4}$$

$[K\phi](x')$ becomes x' scaled by the RHS integral - the first term depends on our choice of kernel, and the size of the scale depends on $\phi(x)$ and a, b.

We can refer to some vector Φ as the eigenvector and λ as the eigenvalue of K if it obeys this equation:

$$[K\Phi] = \lambda\Phi$$

Since our integral operator K is like an infinite-dimensional matrix whose entries are $k(x, x')$, we can refer to a function $\phi(x)$ as the eigenfunction and λ as the eigenvalue of our integral operator K if:

$$\int K(x, x')\phi(x)d\mu(x) = \lambda\phi(x')$$

where μ is some measure over the domain of x upon which the integration will occur thus defining the limits of integration a, b.

Mercer's theorem If our kernel satisfies the conditions of a Gram matrix (i.e. symmetry and positive semidefiniteness) and our measure μ is any compact subset $C \subset \mathbb{R}^n$ (e.g. the probability space $[0, 1]$), then it conforms to Mercer's theorem. Mercer's theorem states that these eigenvalues $[\lambda_i]_{i=1}^{\infty}$ are non-negative and decay to zero, and our kernel itself can be decomposed into a sum of eigenfunctions ϕ_i and eigenvalues λ_i:

$$K(X, X') = \sum_{i=1}^{\infty} \lambda_i \phi_i(X) \phi_i(X') \tag{25}$$

A kernel with infinite rank, or a covariance function that has infinite basis functions (e.g. SE), is called a nondegenerate kernel, else it is degenerate. For example, our N-dimensional linear kernel $K(X, X') = X \cdot X'$ results in a degenerate kernel with at most N non-zero eigenvalues.

3.1.3 Varying the length scale

Our covariance functions has some hyperparameters, e.g. the full form of SE in one dimension contains some free parameters σ_f^2, σ_n^2, and l:

$$k_y(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{|x - x'|^2}{l^2}\right) + \sigma_n^2 \delta_{x,x'}$$

Note that our covariance function is for k_y as it is for the noisy targets y, not the function f. σ_f^2 is the signal variance, which controls the overall scale of the function. σ_n^2 is the noise variance, which controls the amount of noise in the observations. $\delta_{X,X'}$ is the Kronecker delta which represents our independence of noise assumption.

l is a "length scale" hyperparameter that controls how sensitive our functions are - if we specify a lower l, we can "artificially" get a high $k(X, X')$. One way to determine l is by the expected number of "upcrossings" that our kernel is expected to make for a given level u. A function performs an upcrossing for u when $u = f(x)$ and $dy/dx > 0$. For example, with $u = 2$ and $y = x^2$, there exists one upcrossing at $(2, \sqrt{2})$ where $dy/dx = 4$, and a downcrossing at $(2, -\sqrt{2})$ where $dy/dx = -4$. For our zero-mean Gaussian processes, the expected number of upcrossings of our (stationary) kernel for a level $0 < u < 1$ is:

$$\mathbb{E}[N_u] = \frac{1}{2\pi}\sqrt{\frac{-k''(0)}{k(0)}} \exp\left(-\frac{u^2}{2k(0)}\right)$$

We can empirically count the number of upcrossings between 0 and 1 $\hat{N}_u$ and set this equal to the expected number of upcrossings to get a value for l:

$$\frac{1}{2\pi}\sqrt{\frac{-k''(0)}{k(0)}} \exp\left(-\frac{u^2}{2k(0)}\right) = \hat{N}_u \tag{26}$$

A large amount of upcrossings implies our data-generating function wiggles rapidly, so our l becomes smaller to produce a covariance function that in turn produces more flexible functions.

3.1.4 Mean square continuity and differentiability

To understand how smooth the functions drawn from a Gaussian process are, we need to understand how differentiable and continuous they are. A more differentiable function implies that the function contains higher order polynomials which makes it smoother, and a continuous function avoids any reductions in smoothness produced by discontinuities.

Because the functions drawn from the Gaussian distribution are random functions between datapoints, there are infinitely many possible functions and determining if they are all continuous or differentiable is impossible. TODO

Continuity Let x be an infinite-length vector of inputs $x_1, x_2, ...$ whose values linearly approach some stable fixed point x_* as the sequence progresses. Formally:

$$\lim_{k\to\infty} |x_k - x_*| = 0$$

A regular function f is continuous at x_* if three conditions are met. x_* must exist in the domain of f, e.g. $f(X) = 1/x$ is not continuous at $f(0)$ because 0 is not in the domain of f. There must be a single limit the function approaches.

$$f(X) = \begin{cases} 0 & \text{if } X < 0 \\ 1 & \text{if } X > 0 \\ 2 & \text{if } X = 0 \end{cases}$$

In this example, at $f(0)$ x is not continuous at $x = 0$ since f approaches both 0 and 1. This single limit needs to be the same as the function evaluation at the limit. If we removed the 0 if $X < 0$ case to produce a single limit, it would still not be continuous at $x = 0$ since the limit would be 1 but $f(0) = 2$. Formally summarising these three ideas:

$$\lim_{x\to x_*} |f(x) - f(x_*)| = 0$$

A Gaussian process f is MS continuous at x_* if the expected function evaluations $E[f(x)]$ quadratically approach $f(x_*)$ as $x \to x_*$. Formally:

$$\lim_{x \to x_*} \mathbb{E}[(f(x) - f(x_*))^2] = 0$$

Thanks to the quadratic term, we can expand our error term directly into kernel terms. The covariance function k is defined as the expected value of the product of two function evaluations:

$$\begin{aligned}\mathbb{E}[(f(x) - f(x_*))^2] &= \mathbb{E}[f(x)^2] + \mathbb{E}[f(x_*)^2] - 2\mathbb{E}[f(x)f(x_*)] \\ &= k(x,x) + k(x_*,x_*) - 2k(x,x_*)\end{aligned}$$

Using this expansion in our definition of MS continuity:

$$\lim_{x \to x_*} |k(x,x) - 2k(x,x_*) + k(x_*,x_*)| = 0 \tag{27}$$

Because x_* is given, $k(x_*,x_*)$ is a constant. This limit requires that both $k(x,x)$ and $k(x,x_*) \to k(x_*,x_*)$ as $x \to x_*$, which is the very definition of continuity of k at x_* - they guarantee that x_* is in the domain of k, that it has a single limit, and this limit of $k(x,x_*)$ is $k(x_*,x_*)$ as $x \to x_*$. Therefore, a Gaussian process is MS continuous at x_* if and only if the covariance function k is continuous at x_*.

Differentiability f is MS differentiable at x_* in the ith direction of our input vector X if:

$$\lim_{h \to 0} \mathbb{E}\left[|\frac{(f(x_* + he_i) - f(x_*))}{h} - \frac{\partial f}{\partial x_{*i}}(x_*)|\right]^2 = 0$$

$e_i = (0_0, ..., 0_{i-1}, 1_i, 0_{i+1}, ..., 0_n)$ is a unit vector that constrains the change in x to only the ith dimension. We assume $m(x_*) = 0$ for simplicity, but a non-zero mean does not change the structure of this proof and leads to the same result.

Expanding and creating simplifying A and B terms to represent the limit and the proposed derivative respectively:

$$\begin{aligned}\lim_{h \to 0} \mathbb{E}\left[(A_h - B)^2\right] = \mathbb{E}[A_h^2] + \mathbb{E}[B^2] - 2\mathbb{E}[A_h B] = 0 \\ A_h = \frac{f(x_* + he_i) - f(x_*)}{h} \\ B = \frac{\partial f}{\partial x_{*i}}(x_*)\end{aligned}$$

Similar to MS continuity, we expand A_h^2 into kernel terms:

$$\mathbb{E}[A_h^2] = \frac{1}{h^2}\left(k(x_* + he_i, x_* + he_i) + k(x_*,x_*) - 2k(x_* + he_i, x_*)\right) \tag{28}$$

$k(x_*,x_*)$ is a constant, but the other terms vary with h. We can use a second-order Taylor series to approximate these terms if k is twice continuously differentiable at x_*. Starting with the univariate $k(x_* + he_i, x_*)$:

$$\begin{aligned}k(x_* + he_i, x_*) = \\ k(x_*,x_*) \\ +h[\partial_{u_i} k](x_*,x_*) \\ +\frac{h^2}{2}[\partial^2_{u_i u_i} k](x_*,x_*) \\ +O(h^3)\end{aligned}$$

u_i and v_i denote the first $x_* + he_i$ and second x_* arguments respectively in the ith direction. TODO explain O and why it's in terms of h^3 Then, the multivariate $k(x_* + he_i, x_* + he_i)$:

$$\begin{aligned}k(x_* + he_i, x_* + he_i) = \\ k(x_*,x_*) \\ +h[\partial_{u_i} k + \partial_{v_i} k](x_*,x_*) \\ +\frac{h^2}{2}[\partial^2_{u_i u_i} k + \partial^2_{u_i v_i} k + \partial^2_{v_i v_i} k](x_*,x_*) \\ +O(h^3)\end{aligned}$$

Substituting these approximations into 28:

$$\begin{aligned}\mathbb{E}[A_h^2] = \frac{1}{h^2} \\ k(x_*,x_*) + k(x_*,x_*) - 2k(x_*,x_*) \\ +h[\partial_{u_i} k + \partial_{v_i} k](x_*,x_*) - 2h[\partial_{u_i} k](x_*,x_*) \\ +\frac{h^2}{2}[\partial^2_{u_i u_i} k + \partial^2_{u_i v_i} k + \partial^2_{v_i v_i} k](x_*,x_*) - 2\frac{h^2}{2}[\partial^2_{u_i u_i} k](x_*,x_*) \\ +O(h^3)\end{aligned}$$

The constant $k(x_*, x_*)$ terms cancel and the $O(h^3)$ remainders combine but remain the same order.

We can combine the linear partial derivatives to form a single function $[\partial_{u_i} k + \partial_{v_i} k - 2\partial_{v_i} k]$. Thanks to the symmetry of the covariance matrix inherited from the Gram matrix, $k(u, v) = k(v, u)$ and $\partial_{u_i} k = \partial_{v_i} k$ and substituting in either into our linear term cancels it completely.

Similarly, we can combine the quadratic derivatives:

$$E[A^2] = \frac{1}{h^2} \frac{h^2}{2} [\partial^2_{u_i u_i} k + \partial^2_{u_i u_i} k + \partial^2_{v_i v_i} k](x_*, x_*) - h^2 [\partial^2_{u_i v_i} k](x_*, x_*)$$

Cancelling $\frac{1}{h^2}$ and $\frac{h^2}{2}$ and simplifying:

$$\begin{aligned} &= [\partial^2_{u_i v_i} k + \partial^2_{u_i u_i} k + \partial^2_{v_i v_i} k - \partial^2_{u_i v_i} k](x_*, x_*) \\ &= [\partial^2_{u_i u_i} k + \partial_{u_i v_i} k - \partial^2_{v_i v_i} k](x_*, x_*) \end{aligned}$$

Similarly to the linear terms, using the symmetry of k to cancel $\partial^2_{u_i u_i}$ and $\partial^2_{v_i v_i}$ yields the final expression for $\mathbb{E}[A_h^2]$:

$$\mathbb{E}[A_h^2] = \partial^2_{u_i v_i} k(x_*, x_*) + O(h^3) \tag{29}$$

Because B is Gaussian, its square is simply its variance:

$$\begin{aligned} \mathbb{E}[B^2] &= \mathrm{Var}\left(\frac{\partial f}{\partial x_{*i}}(x_*)\right) \\ &= \mathrm{Cov}\left(\frac{\partial f}{\partial x_{*i}}(x_*), \frac{\partial f}{\partial x_{*i}}(x_*)\right) \end{aligned}$$

We can substitute our regular limit definition of differentiability outside of MS space, with different small increments h and h':

$$= \mathrm{Cov}\left(\frac{f(x_* + he_i) - f(x_*)}{h}, \frac{f(x_* + h'e_i) - f(x_*)}{h'}\right)$$

The h and h' denominators merely scale the covariance, so we can take them out of the covariance expression:

$$= \frac{1}{hh'} \mathrm{Cov}\left(f(x_* + he_i) - f(x_*), f(x_* + h'e_i) - f(x_*)\right)$$

Using additivity TODO explain:

$$\mathbb{E}[B^2] = \frac{1}{hh'}\left(k(x_* + he_i, x_* + h'e_i) - k(x_*, x_* + h'e_i) - k(x_* + he_i, x_*) + k(x_*, x_*)\right)$$

We already have $k(x_* + he_i, x_*)$ from **??** directly, and $k(x_*, x_* + he_i)$ by symmetry and substituting h' for h. Our multivariate Taylor series $k(x_* + he_i, x_* + h'e_i)$ looks similar to **??**:

$$\begin{aligned} k(x_* + he_i, x_* + h'e_i) = & \\ k(x_*, x_*) & \\ +h[\delta_{u_i} k](x_*, x_*) + h'[\delta_{v_i} k](x_*, x_*) & \\ +\frac{h^2}{2}[partial^2_{u_i u_i} k](x_*, x_*) & \\ +hh'[\partial^2_{u_i v_i} k](x_*, x_*) & \\ +\frac{h'^2}{2}[\partial^2_{v_i v_i} k](x_*, x_*) & \\ +O(||(h, h')||^3) & \end{aligned}$$

Substituting these results into $\mathbb{E}[B^2]$:

$$\begin{aligned} \mathbb{E}[B^2] = \frac{1}{hh'} & \\ k(x_*, x_*) & \\ +k(x_*, x_*) + k(x_*, x_*) - k(x_*, x_*) & \\ +h[\delta_{u_i} k](x_*, x_*) - h[\delta_{u_i} k](x_*, x_*) & \\ +h'[\delta_{v_i} k](x_*, x_*) - h'[\delta_{v_i} k](x_*, x_*) & \\ +\frac{h^2}{2}[\partial^2_{u_i u_i} k](x_*, x_*) - \frac{h^2}{2}[\partial^2_{u_i u_i} k](x_*, x_*) & \\ +hh'[\partial^2_{u_i v_i} k](x_*, x_*) & \\ +\frac{h'^2}{2}[\partial^2_{v_i v_i} k](x_*, x_*) - \frac{h'^2}{2}[\partial^2_{v_i v_i} k](x_*, x_*) & \\ +O(h^3) + O(h'^3) + O(||(h, h')||^3) & \end{aligned}$$

All constant, linear h and h', and h^2 and h'^2 terms cancel, leaving the mixed term and the remainders. $\frac{1}{hh'}$ cancel and the remainders can be combined since TODO:

$$\mathbb{E}[B^2] = \partial^2_{u_i v_i} k(x_*, x_*) + O(h^2) \tag{30}$$

For $\mathbb{E}[A_h B]$, assuming zero-mean:

$$\mathbb{E}[A_h B] = \mathrm{Cov}\left(\frac{f(x_* + he_i) - f(x_*)}{h}, \frac{\partial f}{\partial x_{*i}}(x_*) \right)$$

Similarly to B, we can remove the scaling factor $\frac{1}{h}$:

$$= \frac{1}{h} \mathrm{Cov}\left(f(x_* + he_i) - f(x_*), \frac{\partial f}{\partial x_{*i}}(x_*) \right)$$

By linearity of covariance TODO elabourate:

$$= \frac{1}{h} \left[\mathrm{Cov}(f(x_* + he_i), \frac{\partial f}{\partial x_{*i}}(x_*)) - \mathrm{Cov}(f(x_*), \frac{\partial f}{\partial x_{*i}}(x_*)) \right]$$

A fundamental identities for Gaussian fields TODO elabourate is:

$$\mathrm{Cov}(f(u), \delta_{u_i} f(v)) = \delta_{u_i} k(u, v)$$

Applying it to both terms:

$$\mathrm{Cov}(f(x_* + he_i), \frac{\partial f}{\partial x_{*i}}(x_*)) = \delta_{u_i} k(x_* + he_i, x_*)$$

$$\mathrm{Cov}(f(x_*), \frac{\partial f}{\partial x_{*i}}(x_*)) = \delta_{v_i} k(x_*, x_*)$$

Taylor expanding our first term:

$$\begin{aligned} \delta_{u_i} k(x_* + he_i, x_*) = \\ \delta_{u_i} k(x_*, x_*) \\ + h\delta^2_{u_i v_i} k(x_*, x_*) \\ + \frac{h^2}{2} \delta^3_{u_i u_i u_i} k(x_*, x_*) \\ + O(h^3) \end{aligned}$$

We can take advantage of symmetry of k to cancel out the constant term using the second term in $\mathbb{E}[A_h B]$ - when $u = v$, $\delta_{v_i} k(x_*, x_*) = \delta_{u_i} k(x_*, x_*)$. Substituting this into $\mathbb{E}[A_h B]$:

$$\begin{aligned} \mathbb{E}[A_h B] = \frac{1}{h} \\ h\delta^2_{u_i v_i} k(x_*, x_*) + \frac{h^2}{2} \delta^3_{u_i u_i v_i} k(x_*, x_*) \\ + O(h^3) \end{aligned}$$

Dividing by h reduces the order of the remainder. Simplifying for a final expression:

$$\mathbb{E}[A_h B] = \delta^2_{u_i v_i} k(x_*, x_*) + \frac{h}{2} \delta^3_{u_i u_i u_i} k(x_*, x_*) + O(h^2) \tag{31}$$

Substituting 29, 30, and 31 into our original limit definition of MS differentiability:

$$\lim_{h \to 0} \delta^2_{u_i v_i} k(x_*, x_*) + O(h^3) + \delta^2_{u_i v_i} k(x_*, x_*) + O(h^2) - 2 \left(\delta^2_{u_i v_i} k(x_*, x_*) + \frac{h}{2} \delta^3_{u_i u_i u_i} k(x_*, x_*) + O(h^2) \right) = 0$$

Redefining $A = \delta^2_{u_i v_i} k(x_*, x_*)$ and $B = \frac{h}{2} \delta^3_{u_i u_i v_i} k(x_*, x_*)$ for brevity:

$$\lim_{h \to 0} A + O(h^3) + A + O(h^2) - 2(A + \frac{h}{2} B + O(h^2)) = 0$$

Expanding the last term:

$$\lim_{h \to 0} A + O(h^3) + A + O(h^2) - 2A - hB - 2O(h^2) = 0$$

All A terms cancel. Combining and reducing the remainders to the lowest order:

$$\lim_{h \to 0} hB + O(h) = 0$$

We can aborsb hB into $O(h)$ since it is a linear term in h:

$$\lim_{h \to 0} O(h) = 0 \tag{32}$$

$O(h)$ approaches 0 as $h \to 0$, so our limit definition of MS differentiability is met using a second-order Taylor expansion. We can extend this to any k-th derivative in MS space by using a $2k$-th order Taylor expansion of the covariance functions, which requires the covariance function to be $2k$-times continuously differentiable at x_*.

3.2 Stationary covariance functions [11]

3.2.1 Stationarity and isotropicism

A stationary covariance function $k(X - X')$ is some function of $X - X'$, and is invariant to the exact locations of X and X'. An isotropic covariance function $k(|X - X'|)$ is a function of $|X - X'|$, and is is invariant to the direction of $X - X'$. For example, SE [**eq:se**] is both stationary and isotropic because it is a function of $|X - X'|$.

3.2.2 Spectral density

Bochner's theorem states that the covariance function of a stationary process can be represented as the Fourier transform of some positive finite measure μ [5]:

$$k(x, x') = \int_{-\infty}^{\infty} e^{2\pi i\omega(x-x')} d\mu(\omega) \tag{33}$$

The Fourier transform decomposes our covariance function into a sum of sines and cosines that represents the frequences that make it up, written in a compact form as a complex exponential:

$$e^{-2\pi i\omega(x-x')} = \cos(2\pi\omega(x - x')) - i\sin(2\pi\omega(x - x')) \tag{34}$$

By applying this representation to the covariance function, we get its "spectral density" $S(\omega)$:

$$S(\omega) = \int_{-\infty}^{\infty} k(x, x')e^{2\pi i\omega(x-x')} d(x - x') \tag{35}$$

$S(\omega)$ is a complex number representing how much total variance is allocated per unit of frequency around our given frequency ω.

The Weiner-Khinchin theorem governs the inversion of the Fourier transform to recover the original covariance function - it states that our covariance function and its spectral density are Fourier duals of each other:

$$k(x, x') = \int_{-\infty}^{\infty} S(\omega)e^{-2\pi i\omega(x-x')} d\omega \tag{36}$$

Smoothness To remain integratable, our spectral density needs to approach 0 as $\omega \to \infty$. The faster it approaches 0, the less power it contains at high frequencies, and the smoother our resultant GP will be. Ritter et. al. [12] formalised this by by taking moments of the spectral density and connecting them to the MS differentiability of our resultant GP. The $2m$-th ω-th moment of the spectral density is:

$$\int_{-\infty}^{\infty} |\omega|^2 m S(\omega) d\omega$$

If this $2m$ moment exists and is finite, our covariance function is $2m$ differentiable thus our GP is m-times MS differentiable.

3.2.3 GPs from stationary covariance functions in MS space

Putting the stationary kernel $k(x, x') = k(x - x')$ into 27:

$$\lim_{x \to x_*} |k(x - x) - 2k(x - x_*) + k(x_* - x_*)| = 0$$

Simplifying the terms inside the kernels and combining like terms:

$$\lim_{x \to x_*} 2|k(0) - k(x - x_*)| = 0$$

2 is a constant factor and can be ignored. Because $|k(0) - k(x - x_*)|$ is invariant to direction, we can swap them to align with the definition of continuity at 0:

$$\lim_{x \to x_*} |k(x_* - x) - k(0)| = 0$$

Thus, a stationary covariance function is MS continuous at x_* if and only if it is continuous at $x_* = 0$. Similarly, it can be shown that stationary covariance functions are MS differentiable at x_* if and only if they are MS differentiable at $x_* = 0$.

3.3 Stationary covariance functions [11]

3.3.1 Squared exponential (SE)

Here is the already introduced SE:

$$k(X, X*) = \exp\left(-\frac{|X - X'|^2}{2l^2}\right)$$

This covariance function is infinitely differentiable at $x - x' = 0$ thanks to the squared difference between X and X' in the exponent, so a GP using SE is infinitely mean-squared differentiable which produces extremely smooth functions, which is often not useful for approximating real-world function. The three major attempts to generalise the SE to vary the level of smoothness are the rational quadratic, γ-exponential, and Matern classes of covariance functions.

From feature space to SE We can derive this form by expanding X into our feature space ϕ defined by Gaussian-shaped basis functions centred densely in X. Defining this basis function:

$$\phi_c(x) = \exp\left(-\frac{|x-c|^2}{2l^2}\right)$$

c the centre of our basis functions. With our familiar isotropic Gaussian prior on the weights $W \sim \mathcal{N}(0, \sigma_p^2 I)$, we get our familiar covariance function in weight-space:

$$k(x,x') = \sigma_p^2 \sum_{c=1}^{N} \phi_c(x)\phi_c(x')$$

N represents the number of these basis functions. If $N = \infty$ with centres everywhere between some interval c_{min} and c_{max}, we can simply integrate over the interval:

$$\lim_{N\to\infty} \sigma_p^2 \sum_{c=1}^{N} \phi_c(x)\phi_c(x') = \sigma_p^2 \int_{c_{min}}^{c_{max}} \phi_c(x)\phi_c(x')dc$$

Plugging in our basis function:

$$k(x,x') = \sigma_p^2 \int_{c_{min}}^{c_{max}} \exp\left(-\frac{|x-c|^2}{2l^2}\right)\exp\left(-\frac{|x'-c|^2}{2l^2}\right)dc$$

Combining the exponentials:

$$= \sigma_p^2 \int_{c_{min}}^{c_{max}} \exp\left(-\frac{|x-c|^2 - |x'-c|^2}{2l^2}\right)dc$$

Expanding the squared terms, rearranging the fraction and simplifying:

$$= \sigma_p^2 \int_{c_{min}}^{c_{max}} \exp\left(-\frac{1}{l^2}\left[c^2 - c(x+x') + \frac{x^2+x'^2}{2}\right]\right)dc$$

We can complete the square on the c terms to get a product of two exponentials:

$$= \sigma_p^2 \int_{c_{min}}^{c_{max}} \exp\left(-\frac{1}{l^2}\left[(c - \frac{x+x'}{2})^2 - \frac{(x+x')^2}{4}\right]\right)dc$$

Our second term in the exponential does not vary with c and can be safely factored out of the integral:

$$= \sigma_p^2 \exp\left(\frac{(x+x')^2}{4l^2}\right)\int_{c_{min}}^{c_{max}} \exp\left(-\frac{1}{l^2}(c - \frac{x+x'}{2})^2\right)dc$$

If we let our c_{min} and c_{max} approach infinity, we can use the standard Gaussian integral:

$$\int_{-\infty}^{\infty} \exp\left(-\frac{1}{l^2}(c - \frac{x+x'}{2})^2\right)dc = l\sqrt{\pi}$$

Substituting this in:

$$k(x,x') = \sigma_p^2 l\sqrt{\pi}\exp\left(-\frac{(x-x')^2}{4l^2}\right)$$

We can absorb the $l\sqrt{\pi}$ into the σ_p^2 term to produce our familiar SE with a $\sqrt{2}$ longer length scale:

$$k(x,x') = \sigma_p^2 \exp\left(-\frac{(x-x')^2}{2(\sqrt{2}l)^2}\right)$$

Length scale We can observe the role l plays in SE by finding the value for l analytically and rearranging it for $\hat{N}_u$ to see its effect on smoothness.

Using 26:

$$l = \frac{1}{2\pi\hat{N}_u}\exp\left(-\frac{u^2}{2\sigma^2}\right)$$

Setting $u = 0$ makes our term inside the exponential equal to zero:

$$l = \frac{1}{2\pi\hat{N}_u}$$

Rearranging for $\hat{N}_u$:

$$\hat{N}_u = \frac{1}{2\pi l}$$

Here, l behaves as a length scale - a larger l reduces the number of upcrossings, stretching out the features over longer distances and producing smoother sample paths.

./img/gaussian_variances.png

Figure 1: Plot of a Gaussian process using SE applied to a toy dataset. The toy dataset ($n = 15$) is a data-generating function in blue with some Gaussian noise applied to produce the datapoints in black. The black line represents the expected function from the Gaussian process. The green line represents the 90% confidence interval around the predictive distribution without the σ_n^2 term, representing the uncertainty surrounding predictions of the noise-free mean function $f(X)$. The red line represents the 90% confidence interval with σ_n^2, representing the uncertainty surrounding predictions of the noisy observations y.
TODO explain

./img/gaussian_draws.png

Figure 2: Plots of functions from a Gaussian process using SE applied to the same toy dataset. The blue line and black datapoints and lines are as before, but the green lines here are a sample of functions drawn from the Gaussian process.
TODO explain

3.3.2 Rational quadratic (RQ)

RQ controls smoothness by using a mixture of different length scales to capture multiple trends in the data with different frequencies.

$$k(X, X') = \left(1 + \frac{|X - X'|^2}{2\alpha l^2}\right)^{-\alpha}$$

We can derive this form by creating an infinite sum of different instances of SE with a vector of different length-scales l and some weights w_i attached to each instance:

$$k(X, X') = \sum_{i=1}^{\infty} w_i \exp\left(-\frac{|X - X'|^2}{2l_i^2}\right)$$

This is a valid covariance function long as the sum remains less than ∞ to maintain PSD. We can turn this into a continuous representation if our l is very dense, where $p(l)$ is the PDF for l:

$$k(X, X') = \int_0^\infty p(l) \exp\left(-\frac{|X - X'|^2}{2l^2}\right) dl$$

Defining $\tau = l^{-2}$, $l = \tau^{-1/2}$. Putting this into SE:

$$k_{SE}(X, X') = \exp(-\frac{|X - X'|^2}{2[\tau^{-1}]}) = \exp(-\frac{1}{2}\tau|X - X'|^2)$$

Before putting this back into the above integral which is in terms of l, we need to account for the change of variables from l to τ. The Jacobian factor of this transformation is:

$$|\partial_\tau l| = \frac{1}{2}\tau^{-3/2} d\tau$$

Defining the transformation $l(x) = \frac{1}{\sqrt{x}}$, putting these into the integral changes the PDF from $p(l)$ to $p(l(\tau))|\partial_\tau l|$:

$$k(X, X') = \int_0^\infty p(l(\tau)) \frac{1}{2}\tau^{-3/2} \exp\left(-\frac{1}{2}\tau|X - X'|^2\right) d\tau$$

We need an expression for the PDF $p(\tau)$ that integrates to 1 and is PSD. A common choice is the Gamma distribution with shape α and scale β parameters: $\tau \sim \Gamma(\alpha, \frac{\beta}{\alpha})$

$$p(\tau) \propto \tau^{\alpha-1} \exp\left(-\frac{\alpha}{\beta}\tau\right)$$

Putting this into the integral:

$$k(X, X') \propto \int_0^\infty \left[\tau^{\alpha-1} \exp\left(-\frac{\alpha}{\beta}\tau\right)\right] \frac{1}{2}\tau^{-3/2} \exp\left(\tau - \frac{1}{2}\tau|X - X'|^2\right) d\tau$$

$\frac{1}{2}\tau^{-3/2}$ from dl and the new τ^{a-1} combine, and the new exponential joins the existing one:

$$k(X, X') \propto \int_0^\infty \frac{1}{2}\tau^{\alpha-\frac{5}{2}} \exp\left[-\tau\left(-\tau\frac{\alpha}{\beta} + \tau\frac{|X - X'|^2}{2}\right)\right] d\tau$$

We can use the Gamma integral here, but need some rearranging. Absorbing $\frac{1}{2}$ into the proportionality, setting $a = \alpha - \frac{3}{2}$ and factoring τ out of the exponent for $c = \frac{\alpha}{\beta}\tau + \frac{|X-X'|^2}{2}$:

$$k(X, X') \propto \int_0^\infty \tau^{a-1} \exp(-\tau c) d\tau$$

Applying the Gamma integral and absorbing $\Gamma(a)$ into proportionality:

$$k(X, X') \propto c^{-a}\Gamma(a) \propto \left[\frac{\alpha}{\beta} + \frac{|X - X'|^2}{2}\right]^{-([\alpha-\frac{3}{2}])}$$

Now we need to decide on the parameter values of our Gamma distribution. Since the Gamma distribution strictly takes in one Typically, $\beta = l^-2$ such that $\beta^{-1} = l^2$ and $\frac{\alpha}{\beta} = a \cdot \beta^{-1} = \alpha l^2$. Substituting:

$$k(X, X') = \left(al^2 + \frac{|X - X'|^2}{2}\right)^{-\alpha-\frac{3}{2}}$$

Factoring out al^2 and absorbing it into the proportionality constant to get our final form:

$$k(X, X') = \left(1 + \frac{|X - X'|^2}{2\alpha l^2}\right)^{-\alpha - \frac{3}{2}}$$

This derivation starts from the discrete length scale l for illustrative purposes, so includes $\frac{3}{2}$ to account for the change from length scale to τ. In practice, derivations start from τ so never have to change units and never include the $\frac{3}{2}$ term in the exponent.

RQ is infinitely MS differentiable, so produces infinitely smooth functions no matter what the shape parameter of our distribution of length scales α is. As $\alpha \to \infty$, this distribution becomes sharply peaked at the single analytically chosen length scale l and RQ becomes SE. However, as $\alpha \to 1$, we get a wider spread of length scales so our functions capture trends at different frequencies.

./img/ratquad_variances.png

Figure 3: Plot of a Gaussian process using rational quadratic applied to the toy dataset. For this dataset, the R package GauPro [3] has obtained $\alpha = 100$ via MLE which produces a kernel close to SE, reflecting the lack of trends in this dataset.

./img/ratquad_draws.png

Figure 4: Plots of functions from a Gaussian process using rational quadratic applied to the same toy dataset.

3.3.3 γ-exponential and exponential

SE and RQ have no parameter controlling or reducing its MS differentiability and its smoothness, rendering it a poor approximation of the less smooth functions we often encounter in the real world. We can introduce a differentiability parameter γ to control how differentiable the covariance function is:

$$k(X, X') = \exp\left(-\frac{|X - X'|^\gamma}{l^\gamma}\right)$$

where $0 < \gamma \geq 2$ controls the smoothness of the covariance function. $\gamma = 2$ produces SE for maximal smoothness. For all other values of γ other than 2, the covariance function is continuous but not differentiable at all at $|x - x'| = 0$, which coincides with the undifferentiable turning point of the modulus function at $x - x' = 0$, and produces the roughest functions of all covariance functions we consider.

Figure 5: Plot of a Gaussian process using γ-exponential applied to the toy dataset. For this dataset, the R package GauPro [3] has used $\gamma = 1.688043$ obtained via MLE.
There are no green lines here because TODO. γ-exponential produces much rougher functions than SE, so its expected function has a lot more wiggle and conforms to the datapoints much closer than SE. In this case, our data-generating function is smooth so SE is closer to the data-generating function than γ-exponential, but real world data-generating functions are often much rougher and would benefit from a kernel flexible enough to pass through the datapoints.

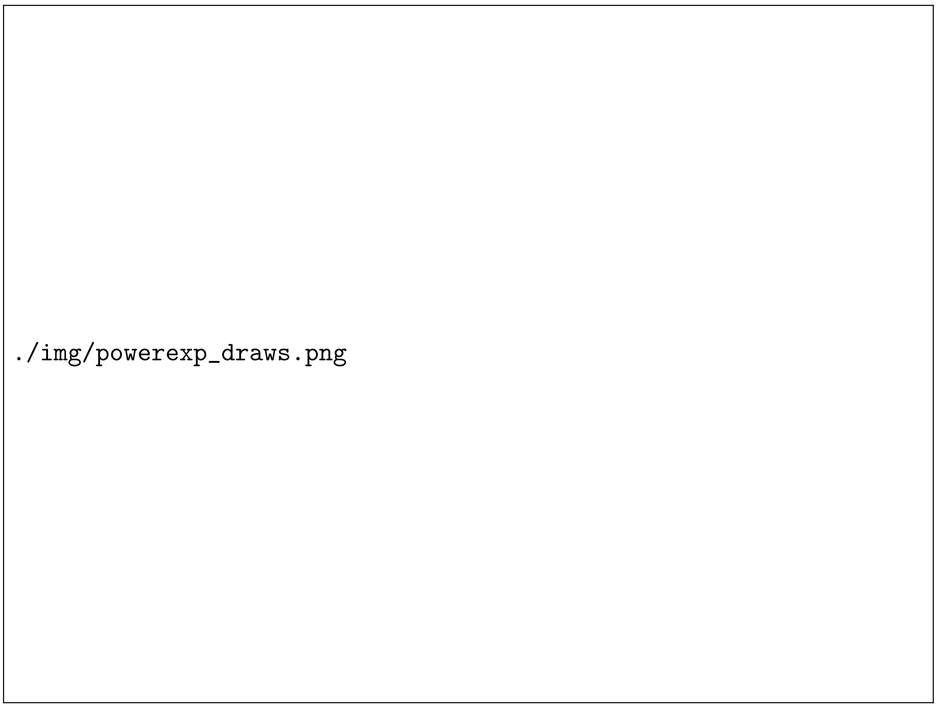

Figure 6: Plots of functions from a Gaussian process using γ-exponential applied to the same toy dataset.
These sample function draws are much rougher than SE.

Exponential covariance function $\gamma = 1$ produces the exponential covariance function:

$$k(X, X') = \exp\left(-\frac{|X - X'|}{l}\right)$$

Restating:

$$k(X, X') = a \exp\left(-c|X - X'|\right)$$

Setting $a = 1$ and $c = \frac{1}{l}$ recovers our familiar exponential kernel **??**. Rybicki and Press [**fast-exp**] showed that any zero-mean GP with this covariance function is the unique stationary solution of the stochastic differential equation (SDE):

$$dX_t = -\lambda X_t dt + \sqrt{2\lambda} dW_t$$

TODO explain parts of equation. This corresponds to the form of a strong Markov process:

$$dX_t = a(X_t)dt + b(X_t)dW_t$$

TODO explain Markov processes. No other covariance function produces an SDE of order 1.

./img/exp_variances.png

Figure 7: Plot of a Gaussian process using exponential applied to the toy dataset.TODO explain

./img/exp_draws.png

Figure 8: Plots of functions from a Gaussian process using exponential applied to the same toy dataset.
These sample function draws are much rougher than SE and γ-exponential.

3.3.4 Matern-class

The γ-exponential generalisation is not useful due to its brittleness - it can only produce two covariance functions representing the extremes of the smoothness-roughness scale. The Matern class of covariance functions addresses this by introducing a parameter $\nu > 0$ that controls the smoothness of the function. The Matern class is defined as:

$$k(X, X') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|X - X'|}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}|X - X'|}{l} \right)$$

l is our familiar length scale hyperparameter. TODO Bessel function K_ν, background needed. Therefore, a Gaussian process using a Matern class kernel is k-times MS differentiable if and only if $\nu > k$.

Using half-integers, i.e. $\nu = p + 1/2$ where p is a non-negative integer, the covariance function becomes a product of a polynomial and an exponential:

$$k_{\nu=p+1/2}(X, X') = \exp\left(-\frac{\sqrt{2\nu}|X - X'|}{l} \right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^{p} \frac{(p+i)!}{i!(p-i)!} \left(\frac{\sqrt{8\nu r}}{l} \right)^{p-i}$$

$\nu = 1/2$ is equivelant to the exponential covariance function. As $\nu \to \infty$, the Matern covariance function approaches the SE covariance function. Practically, $\nu \geq 7/2$ produces functions that are as smooth as SE, leaving us with two cases of interest: $\nu = 3/2$ and $\nu = 5/2$.

Matern 3/2 $\nu = 3/2$ produces the following covariance function:

$$k_{3/2}(X, X') = \left(1 + \frac{\sqrt{3}|X - X'|}{l} \right) \exp\left(-\frac{\sqrt{3}|X - X'|}{l} \right) \tag{37}$$

3/2 is one-time MS differentiable, leading to much rougher functions than SE.

./img/matern32_variances.png

Figure 9: Plot of a Gaussian process using Matern 3/2 applied to the toy dataset.TODO explain

./img/matern32_draws.png

Figure 10: Plots of functions from a Gaussian process using Matern 3/2 applied to the same toy dataset.
TODO explain

Matern 5/2 $\nu = 5/2$ produces the following covariance function:

$$k_{5/2}(X, X') = \left(1 + \frac{\sqrt{5}|X - X'|}{l} + \frac{5|X - X'|^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}|X - X'|}{l}\right)$$

5/2 is twice MS differentiable, producing functions that are slightly smoother than 3/2.

./img/matern52_variances.png

Figure 11: Plot of a Gaussian process using Matern 5/2 applied to the toy dataset.
The expected function deviates very slightly from datapoints compared to 3/2, but the big difference compared to 3/2 is in the variances. 5/2 has narrower variances than 3/2, producing smoother functions and functions that are closer to the expected function. SE is still better in this instance because it is the most smooth and our data-generating function is unusually smooth. However, 5/2 is generally the most popular covariance function because it achieves this parsimonious balance between smoothness, to still reach some reasonable approximation of the data-generating function, and flexibility, to still pass through the datapoints.

./img/matern52_draws.png

Figure 12: Plots of functions from a Gaussian process using Matern 5/2 applied to the same toy dataset. These sample function draws conform more to the expected function than 3/2 due to the narrower variances, but are still much rougher than SE and approach the data-points.

Computational Issues

Training Gaussian processes and producing predictions using them both require inverting the $n \times n$ $[K(X,X) + \sigma_n^2 I]$ matrix. Naively inverting this matrix X has a computational complexity of $O(n^3)$. Unfortunately, this cubic complexity is shared by most covariance functions and renders Gaussian processes unusable in practice - a covariance matrix with $n = 1000$ would require over one billion operations to invert, and this cost would need to be paid for each hyperparameter optimisation step and each prediction. One approach is to approximate this inversion for any covariance matrix. These strategies for any covariance function fall into two categories: those that produce a single approximation for the entire dataset, or those that produce several approximations that are ”experts” in a particular region of the dataset and combine these local approximations to form a global approximation. This section focuses on the global approximations because TODO. Another approach is to design a specific covariance function that produces a covariance matrix which has certain exploitable properties that allow for faster inversion. Generally speaking, these structured approaches are much more accurate than global approximations but are more computationally complex, cannot be used for all covariance functions, and cannot use more than one regressor. We introduce one structured approach, the celerite model, and discuss the origins of these limitations in detail.

4.1 Cholesky decomposition

Cholesky decomposition [11] is a matrix factorisation technique specialised for Gram matrices. Analagous to taking the square root of a Gram matrix, Cholesky finds a unique lower-traingular matrix L with positive diagonal entries for some Gram matrix A such that:

$$A = LL^T \tag{38}$$

For example, let:

$$A = \begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix}$$

L would be:

$$L = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix}$$

such that:

$$LL^T = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix}$$

We can assemble L using the Cholesky algorithm. [10] For diagonal entries in A, this costs $O(n)$:

$$L_{i,i} = \sqrt{A_{i,i} - \sum_{k=1}^{i-1} L_{i,k}^2}$$

For off-diagonal entries:

$$L_{i,j} = \begin{cases} \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right) & \text{if } i > j \\ 0 & \text{if } i < j \end{cases}$$

The computational cost of obtaining the Cholesky factorisation is $O(\frac{1}{3}n^3)$ thanks to the $\sum$ terms in the off-diagonal entries where $i > j$.

Once we have L, we can invert A by solving the two triangular systems:

$$A^{-1} = L^{-T} L^{-1}$$

L^{-1} comes from solving the forward substitution problem $LY = I$, where I is the identity matrix, and L^{-T} comes from solving the backward substitution problem $L^T X = Y$. Inverting L costs $O(n^3)$, for a total complexity of $O(n^3)$ for Cholesky factorisation and inversion.

The inversions of $K(X,X)$ that we need to form 22 and 24 exist as products of other terms ($K(X,X)^{-1}Y$ or $K(X,X)^{-1}K(X,X_*)$). We can obtain these solutions by solving these traingular systems:

$$K(X,X)^{-1}Y = L^{-1}Y$$

Y is the product $K(X,X)^{-1}$.

Cholesky is the preferred method of handling the inversion of covariance matrices because of its extreme numerical stability (thanks to the positive diagonal entries), and its guarantee to produce a valid Gram matrix.

4.2 Sparse kernels

We can improve on the cubic complexity of Cholesky decomposition by using sparse kernels, which are designed to produce sparse covariance matrices that can be inverted more efficiently.

A sparse kernel [10] is a particularly designed kernel that imposes $k(X, X') = 0$ if $|X - X'|$ is larger than some threshold d to create a sparse covariance matrix. This reduces the number of calculations that need to be performed and computational complexity to $O(an^3)$, where a is the proportion of non-zero entries remaining, but the kernel needs to be carefully designed to work with zeroes and ensure all entries are PSD. TODO sparse RBF

4.3 Eigenfunction and eigenvalue approximation of covariance matrices [11]

TODO why not sparse kernels? Mercer's representation of the kernel 25 provides a way of approximating any arbitrary Gram matrix. Instead of summing all the eigenfunctions from 1 to ∞, we can approximate our kernel to any degree of accuracy $M < N$ by restricting our sum to the largest M eigenvalues:

$$K(X, X') \approx \sum_{i=1}^{M} \lambda_i \phi_i(X) \phi_i(X') \tag{39}$$

and invert this smaller matrix at a lower computational cost $O(M^3)$. However, obtaining the eigenfunctions and eigenvalues of an N-size covariance matrix matrix is $O(N^3)$ for a total cost of $O(N^3) + O(M^3)$. We can reduce the cost of obtaining these eigenfunctions and eigenvalues by solving our eigenproblem on a smaller random sample of X. These approximations, and subsequent approximations, come with irreduceable error which could create a kernel that no longer satisfies the Gram matrix constraint of positive semidefiniteness, violating Mercer's theorem and making them invalid covariance matrices.

Instead of our integral weighing our kernel and eigenfunction with the measure μ as before, they become weighed in terms of the PDF of the training data $p(x)$:

$$\lambda_i \phi_i(x') = \int K(x, x') p(x) \phi_i(x) dx$$

We can approximate this continuous representation and make it discrete by representing our selected samples as a vector X_l of size l:

$$\lambda_i \phi_i(X') \approx \frac{1}{l} \sum_{j=1}^{l} K(x_j, X') \phi_i(x_j)$$

Defining a vector Φ_i of all eigenfunction evaluations $\phi_i(x_j)$ on the X_l sample:

$$\lambda_i \phi_i(X') \frac{1}{l} K(X_j, X') \Phi_i$$

Multiplying both sides by l:

$$n \lambda_i \phi_i(X') = K(X_j, X') \Phi_i$$

Setting $\lambda_i^{mat} = l \lambda_i$:

$$K(X_l, X') \phi_i(X') = \lambda_i^{mat} \Phi_i$$

This arrangement requires our covariance matrix to be evaluated with some X'. Without creating another sample or defeating the point of this approximation by using the whole of X, the only candidate for X' is the X_l sample. Setting $X' = X_l$, our $\phi_i(X')$ becomes the definition of Φ_i:

$$K(X_l, X_l) \Phi_i = \lambda_i^{mat} \Phi_i$$

Here, Φ_i also plays the role of the eigenvector U_i of $K(X_l, X_l)$, and λ_i^{mat} plays the eigenvalue of $K(X_l, X_l)$.

Although we introduced $\lambda_i^{mat} = l \lambda_i$, λ_i^{mat} are the eigenvalues of the eigenproblem produced by sampling, whereas λ_i are the eigenvalues from the original eigenproblem. Thus, it acts as an approximation:

$$\lim_{l \to \infty} \lambda_i = \frac{1}{l} \lambda_i^{mat}$$

4.3.1 Approximating Φ_i and U_i with Nystrom

Solving this eigenproblem is $O(l^3)$ - an improvement on the $O(n^3)$ cost of solving the previous eigenproblem, and the resulting $O(l^3) + O(M^3)$ could be lower than the naive $O(n^3)$ inversion cost (if we use sufficiently low degrees of accuracy l and M). We can improve this further by approximating Φ_i.

Φ_i and U_i are not exactly equivelant thanks to differing normalisations. Φ_i's normalisation comes from the original Monte Carlo approximation: $\frac{1}{l} \sum_{j=1}^{l} K(x_j, X') \phi_i(x_j) \approx 1$, thus the sum and our $||\Phi_i||_2^2 \approx l$ and $||\Phi_i|| \approx \sqrt{l}$. U_i's normalisation comes from solving the eigenproblem, which returns $||U_i||_2^2 = 1$ and $||U_i|| = 1$. We can convert between the two by scaling Φ_i by $\sqrt{l}$ to produce U_i.

The Nyström method [1] extends this approximation from the sample points X_l to the full set of points X:

$$\phi_i(X) \approx \frac{1}{\sqrt{l}}\sqrt{n}K(X_l, X)\Phi_i$$

This approximation further reduces the cost of solving the eigenproblem and inverting the low-rank matrix to $O(l^2n) + O(M^3)$.

4.4 Subset-of-data (SoD) [10]

Matrix approximations like the Nystrom approximation can produce kernels that violate PSD due to the irreduceable error involved. Instead of producing a low-rank approximation of the covariance matrix from a full GP, we could guarantee PSD by applying the GP to a subset M of X to reduce the cost of inversion to $O(m^3)$, where m is the number of training points in M. A theoretical graphon analysis showed that choosing M randomly gives an accuracy of $O(log^{-1/4}m)$ for the predictive mean and variance, which produces more accurate predictions with faster runtimes than sparse approximations as n increases. [6] Subset-of-data also requires no analytic assumptions about the covariance functions.

We can reduce m needed to achieve the same level of accuracy with a "greedy" approach by determining the gain in likelihood from including each data point x_i in X, adding the maximum gain in likelihood point to M and repeating until the size of M reaches m. However, computational savings from reducing m are smaller than the cost of searching X for these centroids $O(n^2m)$. Instead, we can use a "matching pursuit" approach - maintain a cache of the already precomputed kernel values, and use these to compute the gain in likelihood for each point in X in $O(nm^2)$ time. [9]

4.5 Sparse approximations [10]

SoD throws away all data not in the selected subset of real training data m, losing information and reducing accuracy. Sparse approximation approaches are more accurate than SoD whilst achieving the same computational cost $O(nm^2+m^3)$, by using potentially imaginary inducing points X_m that best approximate the full covariance matrix $K(X, X)$.

Joining a new prior on $f(X_m)$ to our previous prior on $f(X)$ 21 gives us a new joint prior:

$$\begin{pmatrix} f(X) f(X_m) \end{pmatrix} \sim \mathcal{N}(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(X,X) & K(X,X_m) \\ K(X_m,X) & K(X_m,X_m) \end{pmatrix})$$

Similarly to 22, we can use the Gaussian conditioning identity to condition this joint prior on our inducing points $f(X_m)$ and obtain a posterior distribution over $f(X)$:

$$\begin{aligned} p(f(X)|X, X_m, f(X_m)) \sim \mathcal{N}(\\ K(X,X_m)K(X_m,X_m)^{-1}f(X_m), \\ K(X,X) - K(X,X_m)K(X_m,X_m)^{-1}K(X_m,X) \\ ) \end{aligned} \tag{40}$$

Marginalise out $f(X_m)$:

$$p(f(X)) = \int p(f(X)|f(X_m))p(f(X_m))df(X_m)$$

Without any approximations, it can be shown [10] that this integral recovers our full priors on $f(X)$ **??**:

$$p(f(X)) = \mathcal{N}(0, K(X,X))$$

We can also assemble our predictive distribution by forming a new joint prior and a posterior with the testing data X_* by substituting X for X_*:

$$p(f(X_*)|X_*, X, f(X_m)) \sim \mathcal{N}(K(X_*,X_m)K(X_m,X_m)^{-1}f(X_m), K(X_*,X_*) - K(X_*,X_m)K(X_m,X_m)^{-1}K(X_m,X_*))$$

As with training data, marginalising out X_m yields our original full prior on the training data $f(X_*) \sim \mathcal{N}(0, K(X_*,X_*))$.

We have introduced a new first step into the assembly of the predictive distribution that reduces the cost of prediction from $O(n^3)$ to $O(mn^2 + m^3)$, but the cost of assembling the marginal likelihood remains $O(n^3)$ in training. Approximations aimed at reducing the complexity of computing our marginal likelihoods are divided into two categories: prior approximations which change $p(f(X)|f(X_m))$, and posterior approximations which change $p(f(X), f(X_m)|y)$.

4.5.1 Prior approximations

Subset-of-Regression (SoR) [8] SoR sets the variance of 40 to zero:

$$p_{\text{SoR}}(f(X_*)|f(X_m)) \sim \mathcal{N}(K(X_*,X_m)K(X_m,X_m)^{-1}f(X_m), 0)$$

Producing a marginal prior:

$$p_{\text{SoR}}(f(X_*)) = \int p_{\text{SoR}}(f(X_*)|f(X_m))p(f(X_m))df(X_m)$$

Evaluating this integral and representing the result as a Gaussian distribution:

$$p_{\text{SoR}}(f(X_*)) = \mathcal{N}(0, K(X_*, X_m)K(X_m, X_m)^{-1}K(X_m, X_*))$$

Inverting the m-rank $K(X_m, X_M)^{-1}$ costs $O(m^3)$, and multiplying it by the n-rank $K(X_*, X_m)$ and $K(X_m, X_*)$ costs $O(n^2m)$, sped up using SMW 16 to yield $O(nm^2)$, for a predictive distribution complexity of $O(nm^2+m^3)$. The resulting likelihood:

$$\begin{aligned} p_{\text{SoR}}(Y) = -\frac{1}{2}\log(2\pi\sigma_n^2) \\ -\frac{1}{2\sigma_n^2}Y^TY \\ +\frac{1}{2\sigma_n^4}Y^TK(X, X_m)A^{-1}K(X_m, X)Y \\ -\frac{1}{2}\log|A| \\ +\frac{1}{2}\log|K(X_m, X_m)| \end{aligned}$$

The only inversion here is the rank-m $A = K(X_m, X_m) + \sigma_n^{-2}K(X_m, X)K(X, X_m)$ matrix, which costs $O(m^3)$. This achieves the same computational cost as the Nystrom approximation but does so in a probabilistic framework that guarantees PSD. However, setting our prior's variance to zero produces a predictive distribution that severely underestimates uncertainty, producing GPs that are too confident far from X_m.

Fully independent training conditional (FITC) [13] FITC assumes that the new data $f(X_*)$ is independent given the inducing points $f(X_m)$. Formally:

$$p(f(X_*)|f(X_m)) = \prod_{i=1}^{n} p(f(x_{i*})|f(X_m))$$

TODO more motivation on prior yields. Our new prior distribution is:

$$p_{\text{FITC}}(f(X_*)|f(X_m)) = \mathcal{N}(f(X_*)|K(x_i, X_m)K(X_m, X_m)^{-1}f(X_m), \text{diag}[V_{nn}])$$

$V_{nn} = K(X_*, X_*) - K(X_*, X_m)K(X_m, X_m)^{-1}K(X_m, X_*)$. Our new predictive distribution becomes:

$$\begin{aligned} p_{\text{FITC}}(f(X_*)) \sim \mathcal{N}(\\ k(X_*, X_m)\lambda^{-1}K(X_m, X_m)f(X_m), \\ K(X_*, X_*) - K(X_*, X_*) - K(X_*, X_m)K(X_m, X_m)^{-1}K(X_m, X_*) + K(X_*, X_m)\lambda^{-1}K(X_m, X_*) \\ ) \end{aligned}$$

$\lambda = K(X_m, X_m) + K(X_m, X_*)\text{diag}[V_{nn}]^{-1}K(X_*, X_m)$.

There are three matrices being assembled here: the at-most rank n diag$[V]$, the rank-m λ and the final predictive variance of rank m. V is similar to SoR and costs $O(mn^2)$, whilst getting its diagonal costs $O(nm)$, and assembling λ and the predictive variance is also $O(nm^2)$. We need to invert λ, $K(X_m, X_m)$ and diag$[V_{nn}]$ - the first two are rank M, and inverting a diagonal only costs $O(n)$. These operations reduce to an overall inference complexity of $O(nm^2 + m^3)$, and the resulting likelihood:

$$\begin{aligned} p_{\text{FITC}}(Y) = -\frac{n}{2}\log(2\pi) \\ -\frac{1}{2}\log|\lambda + \sigma_n^2I_n| \\ -\frac{1}{2}\log|K(X_m, X_m) + K(X_m, X)[\lambda + \sigma_n^2I_n]^{-1}K(X, X_m)| \\ +\frac{1}{2}\log|K(X_m, X_m)| \\ -\frac{1}{2}y^T[\lambda + \sigma_n^2I_n]^{-1}y \\ +\frac{1}{2}y^T[\lambda + \sigma_n^2I_n]^{-1}K(X, X_m)[K(X_m, X_m) + K(X_m, X)[\lambda + \sigma_n^2I_n]^{-1}K(X_m, X)[\lambda + \sigma_n^2I_n]^{-1}y \end{aligned}$$

requires an $O(m^3)$ inversion of $K(X_m, X_m) + K(X_m, X)[\lambda + \sigma_n^2I_n]^{-1}K(X_m, X)$ and an $O(nm^2)$ inversion of $\lambda + \sigma_n^2I_n$, for an overall training complexity of $O(nm^2 + m^3)$.

FITC requires the same training and inference complexity as SoR but achieves a more accurate approximation for variance. Additionally, Bauer et. al. [2] found that the diagonal correlation diag$[V_{nn}]$ represents the posterior variances of $f(X_*)$ given $f(X_m)$ enables predictive variances to grow in regions far from inducing points, capturing a form of heteroskedasticity. However, Titsias [14] found that approximate priors produce marginal likelihoods that are far from the marginal likelihood produced by the full prior, which degrades the accuracy of GPs trained using this goal.

4.5.2 Posterior approximations

Variational free energy (VFE) Titsias [14] introduced a variational distribution to approximate the noisy posterior:

$$q(f(X), f(X_m)|y) = p(f(X)|f(X_m)) \cdot q(f(X_m|y))$$

The Kullback-Leibler (KL) divergence measures how different two probability distributions are. The KL divergence between the true posterior $p(f(X), f(X_m)|y)$ and our variational distribution $q(f(X), f(X_m)|y)$ is:

$$D_{KL}(q(f(X), f(X_m)|y)||p(f(X), f(X_m)|y)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)}{p(f(X), f(X_m)|y)} df(X)df(X_m)$$

We need this expression in terms of $\log p(y)$ to form an approximation for the likelihood. We can introduce this term into D_{KL} by Bayes' theorem:

$$D_{KL}(q(.)||p(.)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)}{p(y, f(X), f(X_m)|y)/p(y)} df(X)df(X_m)$$

Simplifying:

$$D_{KL}(q(.)||p(.)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)p(y)}{p(y, f(X), f(X_m)|y)} df(X)df(X_m)$$

Splitting the log term:

$$D_{KL}(q(.)||p(.)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)}{p(y, f(X), f(X_m)|y)} df(X)df(X_m) + \int q(f(X), f(X_m)|y) \log p(y) df(X)df(X_m)$$

$\log p(y)$ is a constant with respect to $f(X)$ and $f(X_m)$ and $\int q(f(X), f(X_m)|y)df(X)df(X_m) = 1$, so our second integral is just $\log p(y)$. Thus:

$$D_{KL}(q(.)||p(.)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)}{p(y, f(X), f(X_m)|y)} df(X)df(X_m) + \log p(y)$$

Rewriting to isolate $\log p(y)$:

$$\log p(y) = \int q(f(X), f(X_m)|y) \log \frac{p(y, f(X), f(X_m))}{q(f(X), f(X_m)|y)} df(X)df(X_m) + D_{KL}(q(.)||p(.)) \tag{41}$$

Our first term F $= \int q(f(X), f(X_m)|y) \log \frac{p(y,f(X),f(X_m))}{q(f(X),f(X_m)|y)} df(X)df(X_m)$ is called the Evidence Lower Bound (ELBO), or the VFE term. $D_{KL}(q(.)||p(.)) \geq 0$, so $\log p(y) \geq$ F and F represents the lowest possible likelihood and raising F produces a raise in $log(y)$. Maximising F using the model hyperparameters directly improves the model, and maximising it using the parameters of the variational distribution makes it a better approximation which improves the hyperparameter tuning methods.

The resulting likelihood after computing the Gaussian integrals in 41 is:

$$\begin{aligned} p_{\text{VFE}}(Y) = &-\frac{1}{2}\log(2\pi\sigma_n^2) \\ &-\frac{1}{2\sigma_n^2} Y^T Y \\ &+\frac{1}{2\sigma_n^4} Y^T K(X, X_m)[K(X_m, X_m) \\ &+\sigma_n^{-2} K(X_m, X) K(X, X_m)]^{-1} K(X_m, X) Y \\ &-\frac{1}{2}\log|K(X_m, X_m) + \sigma_n^{-2} K(X_m, X) K(X, X_m)| \\ &+\frac{1}{2}\log|K(X_m, X_m)| \\ &-\frac{1}{2\sigma_n^2} \text{tr}\left[K(X, X) - K(X_n, X_m) K(X_m, X_m)^{-1} K(X_m, X)\right] \end{aligned}$$

The trace term tr[.] represents the residual variance the variational distribution has left unexplained. The terms in the likelihood that dominate computational complexity are the $y^T K(X, X_m)[K(X_m, X_m) + \sigma_n^{-2} K(X_m, X) K(X, X_m)]^{-1} K(X_m, X) y$ and the $K(X_m, X_m) + \sigma_n^{-2} K(X_m, X) K(X, X_m)$ inversions, which cost $O(nm^2)$ and $O(m^3)$ respectively, for an overall training complexity of $O(nm^2 + m^3)$.

For inference, this posterior produces the variational predictive distribution:

$$\begin{aligned} q(f(X_*)) = \mathcal{N}(\\ K(X_*, X_m) K(X_m, X_m)^{-1} \mu^*, \\ K(X_*, X_*) - K(X_*, X_m) K(X_m, X_m)^{-1} \Sigma^* K(X_m, X_m)^{-1} K(X_m, X_*) \\ ) \end{aligned}$$

μ^* and Σ^* are the mean and covariance of the optimal $q(f(X_m)|y)$ variational distribution. This posterior approximation produces a likelihood where closed form solutions for μ^* and Σ^* at $\partial\frac{p(y)}{\partial\mu} = 0$ and $\partial\frac{p(y)}{\partial\Sigma} = 0$ are available:

$$\Sigma^* = \left[K(X_m, X_m)^{-1} + \sigma_n^{-2} K(X_m, X_m)^{-1} K(X_m, X) K(X, X_m) K(X_m, X_m)^{-1}\right]^{-1} \mu^* = \sigma_n^{-2} \Sigma^* K(X_m, X) y$$

Assembling Σ^* and μ^* requires inverting $K(X_m, X_m)$, which costs $O(m^3)$, and multiplying it by the n-rank $K(X_m, X)$ and $K(X, X_m)$, which costs $O(nm^2)$, for a total complexity of $O(nm^2+m^3)$. VFE achieves a more accurate approximation of the marginal likelihood than FITC without additional computational complexity.

Deriving FITC through the VFE framework [fitc-vfe-unifier] Bui et. al. [**fitc-vfe-unifier**] found that applying expectation propagation, or minimising the "inclusive" KL where p and q are swapped: $D_{KL}(p(.)||q(.))$ recovers the FITC prior exactly:

$$\begin{aligned} q(f) = N(\\ 0, \\ K(X, X_m) K(X_m, X_m)^{-1} K(X_m, X) + \text{diag}[V_{nn}] \\ ) \end{aligned}$$

They defined the power expectation propagation (PEP) objective:

$$\log q_{\text{PEP}}(y) = \log q(y) - \frac{1-\alpha}{2\alpha} \text{tr} \left[\log(I_n + \frac{\alpha}{\sigma_n^2} V_{nn}\right]$$

$q(y)$ represents our FITC prior and the trace term represents VFE. When $\alpha = 1$, the trace term vanishes and we recover FITC exactly. As $\alpha \to 0$, the expansion of the $\frac{1-\alpha}{/}\alpha \log(I + \alpha A) \to tr[A]$ and we recover VFE in its collapsed form. TODO improve explanation

Stochastic varational GP (SVGP) Hensman et. al. [7] removed VFE's closed form solutions for the variational distribution's mean and variance and directly optimised them for lower computational complexity. They used a simple posterior approximation:

$$q(f(X_m)|y) = \mathcal{N}(f(X_m)|M, S)$$

M and S are the vector of means and the covariance matrix respectively, and are treated as free parameters to be optimised to maximise F. This produces a simpler ELBO:

$$F = \mathbf{E}_{q(f(X))}\left[\log p(y|f(X))\right] - D_{KL}(q(f(X_m)|y)||p(f(X_m)))$$

The KL distance between our approximation and the prior is:

$$D_{KL}(q(f(X_m)|y)||p(f(X_m))) = \frac{1}{2}\left(\log \frac{|K(X_m, X_m)|}{|S} - m + \text{tr}(K(X_m, X_m)^{-1} S) + M^T K(X_m, X_m)^{-1} M\right)$$

We can use the posterior produced by these approximations to obtain $logp(y|f(X))$. The posterior is:

$$\begin{aligned} p(f(X)|X) = \mathcal{N}(\\ K(X, X_m) K(X_m, X_m)^{-1} M, \\ K(X, X) - K(X, X_m) K(X_m, X_m)^{-1} (K(X_m, X_m) - S) K(X_m, X_m)^{-1} K(X_m, X) \\ ) \end{aligned}$$

For the i-th point, we can find the squared difference between the true y_i and the predicted $f(x_i)$ by producing predictions using the posterior mean μ_i and variance Σ_{ii} at i:

$$\mathbf{E}_{q(f(X_i))}\left[\log p(y_i|f(X_i))\right] = \mathbf{E}\left[(y_i - f(x_i))^2\right] = (y_i - \mu_i)^2 + \Sigma_{ii}$$

Assuming these remainders are Gaussian and independent of each other, we can sum all i remainders together and plug them into the Gaussian likelihood to get a global expression:

$$\log p(y|f(X)) = -\frac{n}{2}\log(2\pi\sigma_n^2) - \frac{1}{2\sigma_n^2}\sum_{i=1}^{n}\left[(y_i - \mu_i)^2 + \Sigma ii\right]$$

The only reason both training and inference using this approximation is $O(mn^2 + m^3)$ is because the posterior means and variances currently cost $O(nm^2)$ to compute. Spending this during inference is unavoidable, but Stochastic Gradient Descent (SGD) [**adam**] suggests that we can reduce training costs by only summing a batch of b points at a time:

$$\log p(y|f(X)) = -\frac{n}{2}\log(2\pi\sigma_n^2) - \frac{1}{2\sigma_n^2}\frac{n}{b}\sum_{i=1}^{b}\left[(y_i - \mu_i)^2 + \Sigma ii\right]$$

This results in a training complexity of $O(bm^2 + m^3)$, which is significantly lower than the $O(nm^2 + m^3)$ complexity of VFE and FITC. TODO give full advantages & disadvantages

4.6 Celerite kernels [4]

4.6.1 The celerite model

Rybicki and Press [**fast-exp**] used the Markov property of the exponential covariance function to produce a fast algorithm for computing its inverse. Since the exponential kernel is Markov, the only non-zero covariance at i is $i-1$ and $i+1$, and the covariance matrix is tridiagonal whose inverse can be computed in $O(n)$ time.

Kelly et. al. [**general-exp**] generalised this to J arbitrary mixtures of exponentials:

$$K(X, X') = \sum j = 1^J a_j \exp(-c_j|X - X'|)$$

Although the inverse of this is dense, Kelly et. al. [**general-exp**] showed that it can be computed in $O(NJ^2)$ time.

Foreman and Mackay [4] used the Bochner representation 33 of this generalised covariance function:

$$\begin{aligned} K(X, X') = \sum_{j=1}^{J} \\ frac12(a_j + ib_j)\exp(-(c_j + id_j)|X - X'|) \\ + \frac{1}{2}(a_j - ib_j)\exp(-(c_j - id_j)|X - X'|) \end{aligned}$$

Using the Fourier transform 34:

$$\begin{aligned} K(X, X') = \sum_{j=1}^{J} \\ a_j \exp(-c_j|X - X'|)\cos(d_j|X - X'|) \\ + b_j \exp(-c_j|X - X'|)\sin(d_j|X - X'|) \end{aligned} \tag{42}$$

The argument within this sum is called a celerite term. Setting b_j and d_j recovers the original exponential kernel, but Foreman and Mackay's [4] generalisation can be extended to approximate more complex kernels than sums of exponentials whilst maintaining Kelly et. al. [**general-exp**]'s $O(NJ^2)$ complexity.

4.6.2 Building Matern 3/2 from celerite

Applying the Fourier representation to the kernel to obtain its spectral density 35:

$$S(\omega) = \sum_{j=1}^{J} \sqrt{\frac{2}{\pi}} \frac{(a_j c_j + b_j d_j)(c_j^2 + d_j^2) + (a_j c_j - b_j d_j)\omega^2}{\omega^4 + 2(c_j^2 - d_j^2)\omega^2 + (c_j^2 + d_j^2)^2}$$

A stochastically-driven dampened simple harmonic oscillator (SHO) is a system that oscillates with a frequency ω and is dampened by a factor c. The differential equation for this system is:

$$\left[\frac{d^2}{dt^2} + \frac{\omega_0}{Q}\frac{d}{dt} + \omega_0^2\right] y(t) = \eta(t)$$

Q is the "quality factor" of the oscillator, which determines how quickly it loses energy, and $\eta(t)$ is the stochastic force driving the oscillation. Anderson et. al. [**sho-spectral-density**] found that if $\eta(t)$ is white noise, the spectral density of the SHO is:

$$S(\omega) = \sqrt{\frac{2}{\pi}} \frac{S_0 \omega_0^4}{(\omega^2 - \omega_0^2)^2 + \frac{\omega_0^2 \omega^2}{Q^2}}$$

where S_0 is proportional to the power at $\omega = \omega_0$, $S(\omega_0) = \sqrt{\frac{2}{\pi}} S_0 Q^2$ TODO explain.

Foreman and Mackay [4] showed that the celerite spectral density matched the SHO spectral density if $Q \geq \frac{1}{2}$:

$$\begin{aligned} a_j &= S_0 \omega_0 Q \\ b_j &= \frac{S_0 \omega_0 Q}{\sqrt{4Q^2 - 1}} \\ c_j &= \frac{\omega_0}{2Q} \\ d_j &= \frac{\omega_0}{2Q}\sqrt{4Q^2 - 1} \end{aligned}$$

For $0 < Q \leq \frac{1}{2}$, Foreman and Mackay [4] used a pair of celerite terms with parameters:

$$\begin{aligned}
a_{j\pm} &= \frac{1}{2} S_0 \omega_0 Q \left[1 \pm \frac{1}{\sqrt{1-4Q^2}}\right] \\
b_{j\pm} &= 0 \\
c_{j\pm} &= \frac{\omega_0}{2Q}\left[1 \pm \sqrt{1-4Q^2}\right] \\
d_{j\pm} &= 0
\end{aligned}$$

As $Q \to 0$, our spectral density indicates that the process has no oscillatory behaviour and represents a white noise process, e.g. $Q = 1/\sqrt{2}$ recovers an astrophysics model [**sho-noise-astro**] for background granulation noise in stars. Conversely, the process exhibits strong oscillatory behaviour as $Q \to 1$.

We can use the Weiner-Khinchin theorem 36 to obtain the SHO kernel from these spectral densities:

$$K(X, X'; S_0, Q, \omega_0) = S_0 \omega_0 Q \exp^{-\frac{\omega_0 |X-X'|}{2Q}} \begin{cases} \cosh(\eta\omega_0|X-X'|) + \frac{1}{2\eta Q}\sinh(\eta\omega_0|X-X'|) & 0 < Q < \frac{1}{2} \\ 2(1+\omega_0|X-X'|) & Q = \frac{1}{2} \\ \cos(\eta\omega_0|X-X'|) + \frac{1}{2\eta Q}\sin(\eta\omega_0|X-X'|) & Q > \frac{1}{2} \end{cases}$$

$\eta = |1-(4Q^2)^{-1}|^{1/2}$.

At $Q = \frac{1}{2}$, the kernel becomes:

$$K(X, X')_{Q=\frac{1}{2}} = S_0 \omega_0 \exp^{-\omega_0|X-X'|}(1+\omega_0|X-X'|)$$

Setting $\omega_0 = \frac{\sqrt{3}}{l}$ recovers the Matern 3/2 kernel .

4.6.3 Cholesky factorisation and inversion of celerite kernels

Foreman and Mackay [4] develop a Cholesky factorisation method for these celerite kernels in $O(NR^2)$ complexity by exploiting their semiseperable structure. This methods reduces the $O(n^3)$ complexity of standard Cholesky factorisation **??** whilst retaining its numerical stability.

Semiseperable matrices A rank-R semiseperable matrix K is a matrix that can be expressed as a sum of a diagonal matrix A and two low-rank matrices U and V:

$$K_= \begin{cases} \sum_{r=1}^{R} U_{nr}V_{mr} & m < n \\ A_{nn} & m = n \\ \sum_{r=1}^{R} U_{mr}V_{nr} & m > n \end{cases} \tag{43}$$

In matrix notation:

$$K = A + \text{tri}_{\text{upper}}(UV^T) + \text{tri}_{\text{lower}}(VU^T) \tag{44}$$

A contains all the diagonal elements of the matrix $i = j$, and the second and third terms cover all the elements below and above the diagonal respectively.

Celerite kernels as semiseperable matrices For celerite kernels 42, the diagonal component A is simply:

$$A_{nn} = \sum_{j=1}^{J} a_j$$

[4] For the off-diagonal components, defining $R = 2J$, we can assemble U and V:

$$\begin{aligned}
U_{n,2j-1} &= a_j \exp^{-c_j t_n}\cos(d_j t_n) + b_j \exp^{-c_j t_n}\sin(d_j t_n) \\
U_{n,2j} &= a_j \exp^{-c_j t_n}\sin(d_j t_n) - b_j \exp^{-c_j t_n}\cos(d_j t_n) \\
V_{m,2j-1} &= \exp^{+c_j t_m}\cos(d_j t_m) \\
V_{m,2j} &= \exp^{+c_j t_m}\sin(d_j t_m)
\end{aligned}$$

[4] The resulting covariance matrix K_{nm} can be assembled using the rules for semiseperable matrices **??**.

Cholesky factorisation of semiseperable matrices The goal of Cholesky factorisation **??** remains to find the square root of the matrix. For standard Cholesky factorisation, the diagonal is already baked into L. However, for semiseperable matrices the diagonal is not part of L and needs to be taken into account in the Cholesky representation of K:

$$K = LDL^T \tag{45}$$

Foreman and Mackay [4] use the ansatz that our Cholesky factor has the form:

$$L = I + \text{tri}_{\text{lower}}(UW^T)$$

I is the identity matrix, U is from the definition of the semiseperable matrix 44. After we determine the unknown $N \times N$ matrix W, we can determine our Cholesky factor. Substituting this ansatz into 45:

$$K = [I + \text{tri}_{\text{lower}}(UW^T)]D[I + \text{tri}_{\text{upper}}(WU^T)]$$

Simplifying:

$$K = D + \text{tri}_{\text{lower}}(UW^T)D + D\text{tri}_{\text{upper}}(WU^T) + \text{tri}_{\text{lower}}(UW^T)D\text{tri}_{\text{upper}}(WU^T)$$

Setting each element on the right-hand side equal to the corresponding element on the left-hand side of 43 (e.g. $A = D$), we derive the following recurive definition W [4]:

$$\begin{aligned} S_{n,j,k} &= S_{n-1,j,k} + D_{n-1,n-1}W_{n-1,j}W_{n-1,k} \\ D_{n,n} &= A_{n,n} - \sum_{j=1}^{R}\sum_{k=1}^{R} U_{n,j}S_{n,j,k}U_{n,k} \\ W_{n,j} &= \frac{1}{D_{n,n}}\left[V_{n,j} - \sum_{k=1}^{R} U_{n,k}S_{n,j,k}\right] \end{aligned}$$

Each iteration of this recursion costs $O(R^2)$, and this needs to be run for all N rows in W for an overall complexity of $O(NR^2)$. This cost dominates the $O(N)$ cost of obtaining the triangle matrix U and the diagonal matrix D.

Inversion of semiseperable matrices with a Cholesky factorisation As with standard regular Cholesky inversion, we can avoid the full cost of inversion since GPs only require the inverse multiplied by a product. This process is also cheaper than under regular Cholesky factorisation. As before, our goal is to solve $z = K^{-1}y$:

$$z = K^{-1}y = L^{T^{-1}}D^{-1}L^{-1}y$$

Recursively solving for $L^{-1}y$ using forward substitution, and defining the solution as z':

$$\begin{aligned} f_{n,j} &= f_{n-1,j} + W_{n-1,j}z'_{n-1} \\ z'_n &= y_n - \sum_{j=1}^{R} U_{n,j}f_{n,j} \end{aligned}$$

where $f_{0,j} = 0$ for all j. Using z' and backward substitution, we can compute z recursively:

$$\begin{aligned} g_{n,j} &= g_{n+1,} + U_{n+1,j}z_{n+1} \\ z_n &= \frac{z'_n}{D_{n,n}} - \sum_{j=1}^{R} W_{n,j}g_{n,j} \end{aligned}$$

The total cost of both the forwards and backwards substitution is $O(NR)$.

TODO Applying a Gaussian Process to Astrostatistics

5.1 Introduction

5.1.1 Astrological background [galaxy-spectra-101]

5.1.2 Using GP [galaxy-gp-noise]

5.2 Methodology

5.2.1 Approaches considered

Stochastic variational Gaussian processes (SVGP) [7]

Structured kernel interpolation (SKI) [ski]

Celerite kernels [4]

5.2.2 Computational speed

5.2.3 Accuracy

5.3 Results

5.4 Discussion

5.5 Conclusion

TODO Conclusion

References

[1] Christopher T H Baker. *The numerical treatment of integral equations*. en. Monographs on Numerical Analysis. London, England: Oxford University Press, Jan. 1978, pp. 67, 67.

[2] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. *Understanding Probabilistic Sparse Gaussian Process Approximations*. 2017. arXiv: 1606.04820 [stat.ML]. URL: https://arxiv.org/abs/1606.04820.

[3] Collin Erickson. *GauPro: Gaussian Process Fitting*. R package version 0.2.15.9000, commit 6c7a7a8055506229b6d02650aca966 2025. URL: https://github.com/CollinErickson/GauPro.

[4] D. Foreman-Mackey et al. "Fast and Scalable Gaussian Process Modeling with Applications to Astronomical Time Series". In: *aj* 154 (Dec. 2017), p. 220. DOI: 10.3847/1538-3881/aa9332.

[5] Iosif I Gikhman and Anatoli V Skorokhod. *The theory of stochastic processes I*. en. Classics in mathematics. Berlin, Germany: Springer, Mar. 2004.

[6] Kohei Hayashi, Masaaki Imaizumi, and Yuichi Yoshida. "On Random Subsampling of Gaussian Process Regression: A Graphon-Based Analysis". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 26–28 Aug 2020, pp. 2055–2065. URL: https://proceedings.mlr.press/v108/hayashi20a.html.

[7] James Hensman, Nicolo Fusi, and Neil D. Lawrence. *Gaussian Processes for Big Data*. 2013. arXiv: 1309.6835 [cs.LG]. URL: https://arxiv.org/abs/1309.6835.

[8] Vladimir Joukov and Dana Kulić. "Fast Approximate Multioutput Gaussian Processes". In: *IEEE Intelligent Systems* 37.4 (July 2022), pp. 56–69. ISSN: 1541-1672. DOI: 10.1109/MIS.2022.3169036. URL: https://doi.org/10.1109/MIS.2022.3169036.

[9] Sathiya Keerthi and Wei Chu. "A matching pursuit approach to sparse Gaussian process regression". In: *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper_files/paper/2005/file/1ec3e7af38e33222bde173fecaef6bfa-Paper.pdf.

[10] Haitao Liu et al. "When Gaussian Process Meets Big Data: A Review of Scalable GPs". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11 (2020), pp. 4405–4423. DOI: 10.1109/TNNLS.2019.2957109.

[11] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005. ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001. eprint: https://direct.mit.edu/book-pdf/2514321/book_9780262256834.pdf. URL: https://doi.org/10.7551/mitpress/3206.001.0001.

[12] Klaus Ritter, Grzegorz W. Wasilkowski, and Henryk Wozniakowski. "Multivariate integration and approximation for random fields satisfying Sacks-Ylvisaker conditions". In: *The Annals of Applied Probability* 5.2 (May 1995). DOI: 10.1214/aoap/1177004776.

[13] Edward Snelson and Zoubin Ghahramani. "Sparse Gaussian Processes using Pseudo-inputs". In: *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper_files/paper/2005/file/4491777b1aa8b5b32c2e8666dbe1a495-Paper.pdf.

[14] Michalis Titsias. "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 567–574. URL: https://proceedings.mlr.press/v5/titsias09a.html.