Comparing Correlation Functions and Exploring Efficiency and Identifiability Issues for the Gaussian Process

Ivor Walker
Supervised by Dr Michail Papathomas

Abstract

Contents

1 Introduction **3**

2 Defining the Gaussian Process **3**
2.1 Weight-space view [10] . . . 3
2.1.1 Standard linear model . . . 3
2.1.2 Posterior distribution . . . 3
2.1.3 Predictive distribution . . . 5
2.1.4 Projections of inputs into feature space . . . 6
2.1.5 Computational issues . . . 7
2.2 Function-space view [10] . . . 8
2.2.1 Gaussian processes (GP) . . . 8
2.2.2 Predictive distributions with noise-free observations . . . 9
2.2.3 Predictive distributions with noisy observations . . . 9
2.2.4 Marginal likelihood . . . 9
2.2.5 Algorithm for predictive distribution . . . 10
2.3 Varying the hyperparameters [10] . . . 10
2.4 Smoothing and equivelant kernels [10] . . . 10
2.4.1 Linear predictors and smoothers . . . 11
2.4.2 Explicit basis functions . . . 11

3 Exploring Covariance Functions **11**
3.1 Covariance functions, chapter 4 [10] . . . 11
3.2 Simulations of different kernels [6] . . . 11
3.2.1 Continuous . . . 11
3.2.2 Periodic . . . 11
3.2.3 Factors . . . 11
3.3 Deriving kernels [14] . . . 12
3.4 Learning best kernel from data [1] . . . 12
3.5 Additive covariance kernels for high-dimensional learning [5] . . . 12
3.6 Hierarchical Bayesian covariance function for hierarchical modelling [11] . . . 12
3.7 Free-form covariance matrix for multi-task learning [3] . . . 12
3.8 Combining different kernels for multi-task learning [8] . . . 12

4 Extensions of the Gaussian Process **12**
4.1 Gaussian process regression networks [15] . . . 12
4.2 Variational Gaussian process [13] . . . 12

5 Computational Issues **12**
5.1 Matrix inversion [7] . . . 12
5.1.1 Global approximations . . . 12
5.1.2 Local approximations . . . 12
5.1.3 Improvements . . . 12
5.1.4 Extensions . . . 13

6 Applying a Gaussian process to TBD **13**
6.1 Materials science [4] . . . 13
6.2 Cosmography [12] . . . 13
6.3 Statistical emulators [2] . . . 13
6.4 Signals processing [9] . . . 13

7 Conclusion **13**

1 Introduction

2 Defining the Gaussian Process

2.1 Weight-space view [10]

2.1.1 Standard linear model

The standard linear model neatly summarises how all data ever produced in the universe has been generated. It summises that we have some data-generating function $f(.)$ that combines training data X and parameters of the model W to produce an output y:

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

y is rarely a perfect observation of $f(X)$, e.g. because of measurement error, so we add some noise ϵ. The standard linear model assumes that our noise term can be drawn from a Gaussian distribution $\epsilon \sim N(0, \sigma_n^2 I)$. We add a covariance matrix I to describe how the noise for one observation is related to the noise of another observation, e.g. how a burst of noise affects some observations but not others.

We can combine our expressions and assumptions for $f(X)$ and ϵ to produce the distribution of errors, or the distribution from which y is drawn from after knowing perfectly X and W:

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

Our first task is to find W as we typically do not know it in advance. Frequentist statistics focuses on arriving at a single estimate of W ($\hat{W}$) via MLE (maximum likelihood estimation). $p(y|X, W)$ is at its highest density around our error $y|X^T W$ and we assumed our error zero, so we can use optimisation methods to find $\hat{W}$ to find the highest density of the distribution $N(0, \sigma_n^2 I)$, which is the same as minimising the squared error $||y - X^T W||^2$. However, a single $\hat{W}$ does not tell us how confident we are in our estimate, or how much uncertainty there is in the model.

Bayesian statistics addresses this by treating W as a random variable. We start with a "prior" distribution of W, which the Bayesian linear model assumes:

$$p(W) \sim N(0, \Sigma_p) \tag{2}$$

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 our priors $p(W)$ to the data X, W under our assumed noise model ϵ, $p(W)$ is the prior distribution of the weights, and $p(y|X)$ is the marginal likelihood, or how likely the data is given the model.

2.1.2 Posterior distribution

2.1.2.1 Deriving our posterior Because we need to understand how our posterior varies with our weights only, we can ignore terms that do not vary with W such as our marginal likelihood by absorbing it into the proportionality constant.

$$p(W|X, y) \propto p(y|X, W)p(W) \tag{3}$$

Starting with $p(Y|X, W)$, we can write it as the distribution of errors for each data point i:

$$p(y|X, W) = \prod_{i=1}^{N} \mathcal{N}(y_i|X^T W, \sigma_n^2)$$

We can find $p(y|X, W)$ by multiplying the Gaussian density $-\frac{1}{2\sigma_n^2}$ and the squared errors from our model $||y - X^T W||^2$ to get a final likelihood:

$$p(y|X, W) = exp\left(-\frac{1}{2\sigma_n^2}||y - X^T W||^2\right) \tag{4}$$

Then, we can rewrite our $p(W)$ as its density function:

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

The first term can be absorbed into the proportionality constant, so rewriting the second term as negative exponential:

$$p(W) \propto exp\left(-\frac{1}{2}W^T \Sigma_p^{-1} W\right) \tag{5}$$

Putting both expressions for 4 and 5 back into the posterior:

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

To simplify, first we can expand $||y - X^T W||^2$ to $y^T y - 2y^T XW + W^T X^T XW$, and insert this expanded expression:

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

Then, we put both exponentials together by adding their powers:

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

We can rearrange the inside term to be a quadratic, linear and constant term in W:

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

We can ignore the constant final term, and introduce these terms:

$$\begin{aligned} A &= \Sigma_p^{-1} + \frac{1}{\sigma_n^2}X^T X \\ b &= \frac{1}{\sigma_n^2}y^T X \end{aligned} \tag{6}$$

To get our final posterior's PDF:

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

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

Firstly, we can bring all terms inside the exponential to a single term:

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

We can "complete the square" on this term $W^T AW - 2b^T W$ to rewrite it in a form that is easier to interpret:

$$W^T AW - 2b^T W = (W - A^{-1}b)^T A(W - A^{-1}b) - b^T A^{-1}b$$

Substituting this back into our posterior density:

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

If we look at our Gaussian probability density function (PDF):

$$N(W|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}}exp\left(-\frac{1}{2}(W-\mu)^T\Sigma^{-1}(W-\mu)\right) \tag{7}$$

We can see that our expression lines up with the RHS Gaussian "kernel" term $exp\left(-\frac{1}{2}(W-\mu)^T\Sigma^{-1}(W-\mu)\right)$, where $\mu = A^{-1}b$ and $\Sigma^{-1} = A$ thus $\Sigma = A^{-1}$ So we can write our posterior density in Gaussian form

$$p(W|X,y) \sim N(A^{-1}b, A^{-1}) \tag{8}$$

Finally, we lack an expression for Σ_p in A. The standard linear model assumes independence of noise, so our weight variance Σ_p under the Bayesian linear model is "isotropic", meaning it is the same in all directions.

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

Because we assume independence, I is an "identity matrix where each diagonal element is 1 and all off-diagonal elements are 0. τ^2 is a scalar variance term, chosen as a prior.

We can substitute our 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 this gives us our final expression for A:

$$A = \frac{1}{\sigma_n^2}\left(X^T X + \frac{\sigma_n^2}{\tau^2}I\right) \tag{9}$$

2.1.2.3 Gaussian posteriors and ridge regression For Gaussian posteriors, our mean $A^{-1}b$ is also its mode, called the maximum a posteriori (MAP) estimate of W. This is due to symmetries in linear model and posterior and is not the case in general. Our MAP point is equivelant to our frequentist $\hat{W}$.

We can substitute our full expressions for A 9 and b 6 into our MAP estimation to understand how our expressions relate to frequentist approaches:

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

Inverting LHS term of A:

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

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:

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

The solution to frequentist ridge regression is very similar

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

In ridge regression, λ is a regularisation parameter that controls the amount of shrinkage which is usually selected to maximise likelihood/minimise error. Therefore, our MAP estimation in Bayesian linear regression with isotropic priors is equivalent to ridge regression with $\lambda = \frac{\sigma_n^2}{\tau^2}$.

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.

2.1.3 Predictive distribution

2.1.3.1 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 properly propagate the 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 8 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 errors, which we assume to be distributed normally and independently with our I identity matrix:

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

Substituting into 7 and absorbing the LHS term into the proportionality constant:

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

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

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

We can further combine these with a single factor of $\frac{1}{2}$:

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

Expanding the squared term:

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

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

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

We can define:

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

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

And substitute these into our expression to get our final conditional distribution of f_*, W:

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

Lastly, we have to integrate this wrt W to get our predictive distribution $p(f_*|X_*, X, y)$:

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

We can factor out the $\frac{1}{\sigma_n^2}f_*^2$ term from the integral, as it does not depend on W (since $\int exp(X)dX = exp(X)$):

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

The RHS term is a multivariate Gaussian integral which evaluates to:

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

Substituting this back into our predictive distribution:

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

Note that no part of our expression is now dependent on W. Now we need an expression of everything that depends on f_*. We absorb the second term, since it does not depend on f_* into the proportionality constant, and combine the remaining exponential terms by adding their powers, to get our PDF:

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

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

$$p(f_*|X_*, W) \sim N(X_*^T A^{-1} b, X_*^T A^{-1} X_*) \tag{10}$$

Note that 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

These linear models suffer from limited expressiveness due to the linearity of the model. We can project our inputs into a higher dimensional feature space and apply a linear model in this space, e.g. a scalar x could be projected into the space of powers of x: $\phi(x) = [1, x, x^2, \ldots, x^d]^T$ for a polynomial basis expansion of degree d. $\phi(X)$ maps a D-dimensional input vector X into an N dimensional feature space - choosing $\phi(X)$ is done by the Gaussian process, but for now is a given. Our standard linear model is now:

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

Substituting in $\phi(X)$ for X in 10 gives our new predictive distribution:

$$p(f_*|X_*, X, y) = N(\phi(X_*)^T A_\phi^{-1} b_\phi, \phi(X_*)^T A_\phi^{-1} \phi(X_*)) \tag{11}$$

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

2.1.5.1 Avoiding inversion of A_ϕ 11 inverts the $N \times N$ matrix A_ϕ, where N is dimension of feature space, to get the expected value and variance. Inverting matrices scales poorly for large N ($O(N^3)$), so we need to restate our predictive distribution in a form that avoids this inversion.

Substitute 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 identity to get an expression for A_ϕ^{-1} directly, where $K = \phi(X)^T\Sigma_p\phi(X)$

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

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

$$A_\phi^{-1}\phi(X) = \sigma_n^2\Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1} \tag{12}$$

Substitute in 12 into our 11:

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

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

$$\mathbb{E}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T \cdot \Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1} y \tag{13}$$

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 11:

$$\mathrm{Var}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T\Sigma_p\phi(X_*) - \phi(X_*)^T\Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1}\phi(X)^T\Sigma_p\phi(X_*) \tag{14}$$

With our alternative mean 13 and variance 14, we can form an alternative expression for our predictive distribution:

$$\begin{aligned} p(f_*|X_*, X, y) = \mathcal{N}(\\ \phi(X_*)^T\Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1} y, \\ \phi(X_*)^T\Sigma_p\phi(X_*) - \phi(X_*)^T\Sigma_p\phi(X)(K + \sigma_n^2 I)^{-1}\phi(X)^T\Sigma_p\phi(X_*) \\ ) \end{aligned} \tag{15}$$

With this alternative formulation, we need to invert $n \times n$ matrix $K + \sigma_n^2 I$, where n is the number of training data points, instead of A_{phi}. Inverting $K + \sigma_n^2 I$ is faster if $n < N$. For polynomial basis expansions, N is degree D multiplied by number of features, so N can be very large. Some kernels (e.g. SE) have infinite dimensional feature spaces, so N is infinite. Some data domains (e.g. text classification, genomic data) have very high dimensional feature spaces.

Geometrically, note that n datapoints can span at most n dimensions in the feature space - if $N > n$, the data forms a subspace of the feature space.

2.1.5.2 Kernels and the kernel trick In 15, note that ϕ is always in the same general form but with different combinations of $\phi(X)$ and $\phi(X_*)$, e.g. in the definition of $K = \phi(X)^T\Sigma_p\phi(X)$. We can generalise this to $\phi(X)^T\Sigma_p\phi(X')$, where X and X' are either X or X_*, and define $k(X, X') = \phi(X)^T\Sigma_p\phi(X')$ as a covariance function or kernel.

Note that this kernel is an inner product with a positive definite correlation matrix Σ_p. This lets us define $\psi(X) = \Sigma_p^{1/2}\phi(X)$ Substituting $\psi(X)$ back into our kernel allows the $\Sigma_p^{1/2}$ to cancel out, giving us a simple dot product representation

$$k(X, X') = \psi(X) \cdot \psi(X')$$

Now we have defined a kernel solely in terms of inner products in the input space, we know that there must be another equivelant $k(X, X'$ that does not require us to explicitly compute $\phi(X)$ or $\phi(X')$ in the feature space.

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 $k(X, X')$ using ψ:

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

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

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

This is the polynomial kernel, which is equivalent to the polynomial basis expansion $\phi(X)$.

We still need to perform the same calculations but with this "kernel trick" we avoid the memory cost of explicitly computing $\phi(X)$ and $\phi(X')$ in the feature space. It is mostly more convenient to compute the kernel directly rather than the feature vectors themselves, leading to the kernel being the object of primary interest.

2.2 Function-space view [10]

2.2.1 Gaussian processes (GP)

2.2.1.1 Definition A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. GPs describe a distribution over functions, where each function is a sample from the GP.

Defined completely by its mean function $m(X)$ and covariance function $k(X, X')$ of a real process $f(X)$:

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

These random variables represent the value of $f(X)$ at location X, e.g. often Gaussian processes are defined over time so X can be a time point. The covariance function specifies the covariance between pairs of random variables.

2.2.1.2 Consistency requirement This definition implies a consistency requirement, i.e. examining a larger set does not change the distribution of a smaller set. For example, if our GP implies that $(f(X_1), f(X_2)) \sim \mathcal{N}(\mu, \Sigma)$, then it must specify $(f(X_1) \sim \mathcal{N}(\mu_1, \Sigma), f(X_2) \sim \mathcal{N}(\mu_2, \Sigma))$ 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 the covariance matrix.

2.2.1.3 Bayesian linear regression as a GP We can view our Bayesian linear regression model with $p(W) \sim N(0, \Sigma_P)$ as a GP:

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

We will use a squared exponential (SE) covariance function, also known as the radial basis function (RBF) or Gaussian kernel:

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

To exploit the kernel trick, $k(f(X), f(X')$ is written as a function of X and X'.

For SE, covariance is almost unity between outputs whose inputs are close together, but decays exponentially as inputs get further apart. l is a "length scale" hyperparameter that controls the rate of decay of the covariance function. It can be shown that SE corresponds to a Bayesian linear regression model with infinite basis functions. The Mercer theorem states that for every positive definite covariance function $k(X, X')$, there exists a possibly infinite set of basis functions. SE can also be obtained from the linear combination of infinite Gaussian-shaped basis functions. Because SE is infinitely differentiable, it produces smooth functions.

2.2.1.4 Function evaluations to a random function We can choose a subset of five inputs X_{*1} from our test data X_* and apply a GP to get five outputs $f(X_{*1})$. $f(X_{*1})$ 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.

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 $g(X_{*1})$. Therefore, the specification of the covariance function imples that our GP can also be seen as a distribution of o, where each sample produces a random function $g(X_*)$ that passes through the points $f(X_{*1})$.

2.2.2 Predictive distributions with noise-free observations

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

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

2.2.2.2 Posterior distribution over functions We are not interested in drawing random functions from the prior, but incorporating the knowledge that the training data provides about the function. To get the posterior distribution over functions given the training data, we can condition the joint prior distribution on the training data. Intuitively, this is like generating random functions g and rejecting those that do not pass through the training data. Probabilistically, we need to condition our joint Gaussian prior distribution on the observations $p(f(X_*)|X_*, X, f(X))$.

We can substitute $p(W)$ and our conditioning X into the Gaussian multivariate conditioning identity:

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

Note that 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, we get our final expression for the posterior distribution:

$$\begin{aligned} p(f(X_*)|X_*, X, f(X)) \sim N(\\ K(X,X_*)K(X,X)^{-1}f(X), \\ K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*) \\ ) \end{aligned} \tag{16}$$

2.2.3 Predictive distributions with noisy observations

2.2.3.1 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:

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

2.2.3.2 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 15.

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

We need some measure of how well our GP fits the data, which we can get by computing the marginal likelihood $p(Y|X)$:

$$p(Y|X) = \int p(Y|f, X)p(f|X)df$$

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

$$logp(Y|X) = -\frac{1}{2}Y^T(K + \sigma_n^2 I)^{-1}Y - \frac{1}{2}log|K + \sigma_n^2 I| - \frac{n}{2}log(2\pi) \tag{17}$$

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 Gaussian PDF 7 to get 17.

2.2.5 Algorithm for predictive distribution

1. Take in inputs X, outputs y, covariance function k, noise level σ_n^2, and test input X_*
2. $L = \text{cholesky}(K(X,X) + \sigma_n^2 I)$
   - Invert our $[K(X,X) + \sigma_n^2 I]$ matrix needed for mean and variance using Cholesky decomposition
3. $alpha = L^T \backslash (L \backslash y)$
   - Prepare the mean of our predictive distribution in linear combination form by computing the α vector
4. $mu = K(X_*, X)^T \cdot alpha$
   - Compute the mean
5. $v = L \backslash K(X_*, X)^T$
   - Prepare to compute variance by computing v, the form in which L is used in the variance
6. $var = K(X_*, X_*) - v^T v$
   - Compute the variance
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

TODO Choelsky decomposition if needed

2.3 Varying the hyperparameters [10]

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. l is the length scale, which controls how quickly the covariance decays with distance - if we specify a lower l, we can "artificially" get a high $k(X,X')$. $\delta_{X,X'}$ is the Kronecker delta which represents our independence of noise assumption.

2.4 Smoothing and equivelant kernels [10]

TODO missing background

2.4.1 Linear predictors and smoothers

Understanding how our predictive distribution's mean varies with its inputs is difficult because of the first term $[(K(X,X)+\sigma_n^2 I)]^-1$: it is dependent on exact values of X, and it requires the inversion of $K(X,X)+\sigma_n^2 I$. Instead, we can reformulate $E_{p(f(X_*)|X_*,X,Y)}[f(X_*)]$ as an "equivelant kernel" to remove the dependence on X. which uses a kernel smoother (also known as the Nadaraya-Watson estimator)

Firstly, we rewrite our mean such that our mean function is a "linear smoother", or a function of the covariance function and our training data labels Y:

$$\begin{aligned} H(X_*) &= [K(X,X)+\sigma_n^2 I]^{-1} K(X_*,X) \\ \mathbb{E}_{y_*|X_*,X,y)}[f(X_*)] &= H(X_*)^T Y \end{aligned} \tag{18}$$

This vector of functions $H(X_*)$ is called a weight function and contains our problematic term.

Instead of inputting the values of x_i directly into the kernel function, we can use x_i centred around X_*:

$$k_i = k(|x_i - x_{*i}|/l)$$

Where l is some length scale hyperparameter.

TODO explain what we gain from this

Thus, our predictive distribution mean becomes:

$$\mathbb{E}_{p(f(X_*)|X_*,X,Y)}[f(X_*)] = \sum_{i=1}^{n} w_i y_i$$

where $w_i = k_i / \sum_{j=1}^{n} k_j$ is a TODO.

2.4.2 Explicit basis functions

TODO

3 Exploring Covariance Functions

3.1 Covariance functions, chapter 4 [10]

TODO

3.2 Simulations of different kernels [6]

3.2.1 Continuous

TODO Gaussian, mattern, exponential, SE, triangle, power exponential

3.2.2 Periodic

TODO periodic, mattern

3.2.3 Factors

TODO latent factor, ordered factor, factor, Gower factor, ignore indices

3.3 Deriving kernels [14]

3.4 Learning best kernel from data [1]

3.5 Additive covariance kernels for high-dimensional learning [5]

3.6 Hierarchical Bayesian covariance function for hierarchical modelling [11]

3.7 Free-form covariance matrix for multi-task learning [3]

3.8 Combining different kernels for multi-task learning [8]

4 Extensions of the Gaussian Process

4.1 Gaussian process regression networks [15]

4.2 Variational Gaussian process [13]

5 Computational Issues

5.1 Matrix inversion [7]

Inverting the $[K(X,X)+\sigma_n^2 I]$ matrix in our predictive distribution scales poorly with the number of training data points n, as inverting the $n \times n$ matrix X that represents our training data is $O(n^3)$. Strategies to approximate the result of this inversion 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.

5.1.1 Global approximations

5.1.1.1 Subset-of-data The simplest strategy is to use 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. Although this approach does not address the issues of matrix inversion directly,a theoretical graphon analysis proves 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. [**random-subsampling**] Subset-of-data also requires no analytic assumptions about the kernel.

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. [**matching-pursuit**]

5.1.1.2 Sparse kernels A sparse kernel 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 positive definite to satisfy completeness. TODO sparse RBF

5.1.1.3 Sparse approximations

5.1.1.3.1 Prior approximation

5.1.1.3.2 Posterior approximation

5.1.1.3.3 Structured sparse approximation

5.1.2 Local approximations

5.1.2.1 Naive-local-experts

5.1.2.2 Mixture-of-experts

5.1.2.3 Product-of-experts

5.1.3 Improvements

5.1.3.1 Scalability

5.1.3.2 Capability

5.1.4 Extensions

5.1.4.1 Scalable manifold GP

5.1.4.2 Scalable deep GP

5.1.4.3 Scalable online GP

5.1.4.4 Scalable multi-task GP

5.1.4.5 Scalable recurrent GP

5.1.4.6 Scalable GP classification

6 Applying a Gaussian process to TBD

Possible domains:

6.1 Materials science [4]

6.2 Cosmography [12]

6.3 Statistical emulators [2]

6.4 Signals processing [9]

7 Conclusion

References

[1] Jean-Luc Akian et al. "Learning "best" kernels from data in Gaussian process regression. With application to aerodynamics". In: *Journal of Computational Physics* 470 (Aug. 2022), p. 111595. DOI: 10.1016/j.jcp.2022.111595.

[2] Leonardo S Bastos and Anthony O'hagan. "Diagnostics for Gaussian process emulators". In: *Technometrics* 51.4 (2009), pp. 425–438.

[3] Edwin V Bonilla, Kian Chai, and Christopher Williams. "Multi-task Gaussian Process Prediction". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt et al. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[4] Volker L Deringer et al. "Gaussian process regression for materials and molecules". In: *Chemical Reviews* 121.16 (2021), pp. 10073–10141.

[5] Nicolas Durrande et al. *Additive Covariance Kernels for High-Dimensional Gaussian Process Modeling*. 2011. arXiv: 1111.6233 [stat.ML]. URL: https://arxiv.org/abs/1111.6233.

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

[7] 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.

[8] Arman Melkumyan and Fabio Ramos. "Multi-kernel Gaussian processes". In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Vol. 22. 1. 2011, p. 1408.

[9] Fernando Perez-Cruz et al. "Gaussian Processes for Nonlinear Signal Processing: An Overview of Recent Advances". In: *IEEE Signal Processing Magazine* 30.4 (July 2013), pp. 40–50. ISSN: 1558-0792. DOI: 10.1109/MSP.2013.2250352.

[10] 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.

[11] Anton Schwaighofer, Volker Tresp, and Kai Yu. "Learning Gaussian Process Kernels via Hierarchical Bayes". In: *Advances in Neural Information Processing Systems*. Ed. by L. Saul, Y. Weiss, and L. Bottou. Vol. 17. MIT Press, 2004. URL: https://proceedings.neurips.cc/paper_files/paper/2004/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

[12] Arman Shafieloo, Alex G. Kim, and Eric V. Linder. "Gaussian process cosmography". In: *Phys. Rev. D* 85 (12 June 2012), p. 123530. DOI: 10.1103/PhysRevD.85.123530. URL: https://link.aps.org/doi/10.1103/PhysRevD.85.123530.

[13] Dustin Tran, Rajesh Ranganath, and David M Blei. "The variational Gaussian process". In: *arXiv preprint arXiv:1511.06499* (2015).

[14] Andrew Wilson and Ryan Adams. "Gaussian Process Kernels for Pattern Discovery and Extrapolation". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1067–1075. URL: https://proceedings.mlr.press/v28/wilson13.html.

[15] Andrew Gordon Wilson, David A Knowles, and Zoubin Ghahramani. "Gaussian process regression networks". In: *arXiv preprint arXiv:1110.4411* (2011).