Gaussian processes and their equivalent Bayesian linear regressions

Douglas Mason[1]

[1]Koyote Science, LLC *

September 2021

1   Introduction

Finding the equivalent Bayesian linear regression (BLR) to a Gaussian process (GP) can provide substantial benefits in modeling including (1) enormous efficiency savings with large amounts of data, (2) access to incremental (sequential or online) computation methods, (3) access to vetted approximation methods for efficiency (such as the Laplace approximation for regressors, not just classifiers), (4) easier interpretability, and (5) the ability to effortlessly to combine multiple models into one without using bespoke software libraries. However, while the three major textbooks on the subject [2, 4, 5] cover the subject extensively, it surprised me that they didn't provide a practical side-by-side comparison or recipe for converting the kernel used to generate a GP to an equivalent set of basis functions for a BLR. Moreover, we can look at Gaussian processes as an extension of Bayesian linear regression into the territory of small data, where the number of training points is less than the number of features, which is an important problem space for optimization procedures where evaluating a function may take a substantial amount of time. Bringing together the best of both worlds is enormously beneficial to anyone working within the optimization problem space.

Why this subject isn't lucidly explained in any established textbooks or lectures notes we could find remains a mystery to me, but it actually takes a lot of work to get the equivalence just right. This document covers the recipe we arrived at after working through the problem, and the insights we learned along the way. You are free to explore for yourself using the Github repository which can be accessed at https://github.com/KoyoteScience/GP-BLR.

Studying BLR and GP side-by-side is critical for developing any Bayesian application, since these are among only a few models in circulation that allow

---

*http://www.koyotescience.com

you to learn the parameters using explicit and deterministic update rules. In other words, if we want to put Bayesian models into production with calibrated uncertainty predictions, these two are the robust, reliable baselines from which all other approaches are compared. Moreover, because the computation of these models is deterministic, we avoid the horrible pitfalls of Markov chain Monte Carlo, and because they are exact, we avoid the pitfalls of variational inference. As we will cover, both are actually parts of the same continuum between high featurization and large amounts of data.

A quick glossary: scalars are presented as upper- or lower-case English or Greek letters with normal weight, such as n or N, while vectors are presented as lower-case English or Greek letters in bold, such as $\mathbf{v}$ or $\boldsymbol{\phi}$. Row vectors are presented normally, and column vectors with the transpose operator $\mathbf{v}^\top$. The dot or inner product is two vectors multiplied as $\mathbf{v}^\top \mathbf{v}$ and the outer product is two vectors multiplied as $\mathbf{v}\mathbf{v}^\top$. Matrices are presented as upper-case letters with bold weight, such as $\mathbf{M}$ or $\boldsymbol{\Phi}$. For variables that refer to the posterior (after N training samples), their prior counterparts are denoted by the p subscript.

2   Bayesian linear regression

Bayesian linear regression can be defined by modeling a predictor

$$y = f(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} \tag{1}$$

given a prior distribution for our model parameters $\boldsymbol{\theta}$ defined as

$$\boldsymbol{\theta}_p \sim \mathcal{N}(\mathbf{0}, \Sigma_p) \tag{2}$$

using the convention whereby the normal distribution is defined by two arguments, the means in each dimension and the covariance among the dimensions. Given a set of N observations, over the independent space $\mathcal{X}$ and the dependent space $\mathcal{Y}$, gives us the matrix $\mathbf{X}$ of observations in the independent space and the vector $\mathbf{y}$ of corresponding observations in the dependent space.

$$\boldsymbol{\Sigma}^{-1} = \sigma^{-2}\mathbf{X}\mathbf{X}^\top + \boldsymbol{\Sigma}_p^{-1} \tag{3}$$

This gives us

$$\boldsymbol{\theta} \sim \mathcal{N}(\sigma^{-2}\boldsymbol{\Sigma}\mathbf{X}^\top\mathbf{y}, \boldsymbol{\Sigma}) \tag{4}$$

and

$$\mathbf{y} \sim \mathcal{N}\left(\sigma^{-2}\mathbf{X}\boldsymbol{\Sigma}\mathbf{X}^\top\mathbf{y}, \mathbf{X}^\top\boldsymbol{\Sigma}\mathbf{X} + \sigma^2\,\mathbf{I}_N\right) \tag{5}$$

where σ is the noise parameter, or the square-root of the sum of squared residuals., and the term $\sigma^2\mathbf{I}$ provides the uncertainty for repeated measurements of y at the same $\mathbf{x}$.

The equivalent perspective of $L2$ regularization gives us a regularization constant

$$\lambda = \sigma^2/\sigma_p^2 \tag{6}$$

where σ_p represents the noise term for the prior. This implies that we more strongly regularize our results (draw weights towards zero) as we decrease the noise parameter in the prior compared to the noise parameter in the posterior.

The above formulation makes it clear how the covariance matrix of our parameters evolves over inference from $\mathbf{\Sigma}_p$ to $\mathbf{\Sigma}$ and how this affects our predictions. If we don't keep σ fixed but instead learn it from our data or use linear algebra identities to define it, we may prefer to work with the quantity

$$\mathbf{S}^{-1} = \sigma^2 \mathbf{\Sigma}^{-1} = \mathbf{X}\mathbf{X}^\top + \sigma^2 \mathbf{S}_p^{-1} \tag{7}$$

giving us

$$\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{S}\mathbf{X}^\top \mathbf{y}, \sigma^2 \mathbf{S}) \tag{8}$$

and

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{X}\mathbf{S}\mathbf{X}^\top \mathbf{y}, \sigma^2 \left(\mathbf{X}^\top \mathbf{S}\mathbf{X} + \mathbf{I}_N\right)\right) \tag{9}$$

which makes it far clearer that the covariance matrices for both our weights and predictions are proportional to the noise parameter squared, while the mean weights and predictions only depend upon it through Equation 7. This formulation also makes the connection to kernels in Gaussian processes clearer in later sections.

3 Bayesian linear regression with basis functions

We now transform our independent space using a set of basis function $\boldsymbol{\phi}(\mathbf{x})$ and assume that

$$y_* = f(\mathbf{x}_*) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_*) \tag{10}$$

Given N training samples, N_* test samples, and an N_B-dimensional basis vector, we arrive at the notation $\Phi = [\boldsymbol{\phi}(\mathbf{x}_0), \boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_N))]^\top$, which is a $N \times N_B$ matrix, and $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\mathbf{x}_*)$, which is a $N_* \times N_B$ matrix. This gives us the covariance function:

$$\mathbf{\Sigma}^{-1} = \sigma^{-2} \mathbf{\Phi}\mathbf{\Phi}^\top + \mathbf{\Sigma}_p^{-1} \tag{11}$$

where we note that the calculation in involves matrix multiplications on the scale $N_B \times (N \times N) \times N_B + N_B \times N_B$. In other words, we perform a summation (integration or dot product) over the training data points.

Similarly,

$$\boldsymbol{\theta} \sim \mathcal{N}(\sigma^{-2} \mathbf{\Sigma}\mathbf{\Phi}^\top \mathbf{y}, \mathbf{\Sigma}) \tag{12}$$

involves matrix multiplications on the scale $N_B \times (N_B \times N_B) \times (N \times N)$ for the mean, that is, we perform a summation over the basis vector and the training data. We now have model parameters for each N_B dimension in the basis functions. This gives us

$$\mathbf{y}(\mathbf{x}) \sim \mathcal{N}\left(\sigma^{-2} \mathbf{\Phi}\mathbf{\Sigma}\mathbf{\Phi}^\top \mathbf{y}, \mathbf{\Phi}^\top \mathbf{\Sigma}\mathbf{\Phi} + \sigma^2 \mathbf{I}_N\right) \tag{13}$$

and

$$y_*(\mathbf{x}_*) \sim \mathcal{N}\left(\sigma^{-2} \boldsymbol{\phi}_* \mathbf{\Sigma}\mathbf{\Phi}^\top \mathbf{y}, \boldsymbol{\phi}_*^\top \mathbf{\Sigma}\boldsymbol{\phi}_* + \sigma^2\right) \tag{14}$$

with the associated matrix multiplications on the scale $N_* \times (N_B \times N_B) \times (N_B \times N_B) \times (N \times N)$ for the mean and $N_* \times (N_B \times N_B) \times (N_B \times N_B) \times N_*$ for the covariance. In other words, we most of our summations are over the basis dimensionality N_B and some over the training set dimensionality N.

Meanwhile, the alternative formulation gives us

$$\mathbf{S}^{-1} = \sigma^2 \boldsymbol{\Sigma}^{-1} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2 \mathbf{S}_p^{-1} \tag{15}$$

$$\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{S}\boldsymbol{\Phi}^\top \mathbf{y}, \sigma^2 \mathbf{S}) \tag{16}$$

$$\mathbf{y}(\mathbf{x}) \sim \mathcal{N}\left(\boldsymbol{\Phi}\mathbf{S}\boldsymbol{\Phi}^\top \mathbf{y}, \sigma^2 \left(\boldsymbol{\Phi}^\top \mathbf{S}\boldsymbol{\Phi} + \mathbf{I}_N\right)\right) \tag{17}$$

$$\mathbf{y}_*(\mathbf{x}_*) \sim \mathcal{N}\left(\boldsymbol{\phi}_* \mathbf{S}\boldsymbol{\Phi}^\top \mathbf{y}, \sigma^2 \left(\boldsymbol{\phi}_*^\top \mathbf{S}\boldsymbol{\phi}_* + \mathbf{I}_{N_*}\right)\right) \tag{18}$$

4   Gaussian processes

The Gaussian Process (GP) is defined differently. Instead of considering a set of multi-valued basis function vectors defined for each point in independent space $\mathbf{x}$, we consider a single-valued function that takes as two points in the independent space as its arguments. This function is the kernel $k(\mathbf{x}, \mathbf{x}')$ which can be extrapolated to multiple observations at once as the matrix $\mathbf{K}(\mathbf{X}, \mathbf{X}')$. We thus draw the associated observations in the dependent space $\mathbf{y}$ according to the normal distribution using the prior distribution:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right) \tag{19}$$

Let us simplify this notation a bit:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix}\right) \tag{20}$$

This is a distribution over the observations and the new test points (denoted with the asterisk) jointly. What does it looks like from the new test point's perspective after we account for the observations? We can solve this using Section 4.3.4 and Equation 4.69 in [4], obtaining the posterior distribution:

$$\mathbf{y}_* \sim \mathcal{N}\left(\mathbf{K}_*^\top \mathbf{K}^{-1}\mathbf{y}, \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1}\mathbf{K}_*\right) \tag{21}$$

Note that the Gaussian process formalism presented here assumes that the variance of our targets $\sigma_\mathbf{y}$ is equal to one. However, we can adjust for this by regressing against $\mathbf{y}/\sigma_\mathbf{y}$ and then multiplying our predictions $\sigma_\mathbf{y}$ (and similarly for the prediction standard deviations). The calculations for a Gaussian process require summations in the matrix multiplication at the scale of $N_* \times (N \times N) \times (N \times N)$ for the mean and $N_ast \times (N \times N) \times (N \times N) \times N_*$ for the covariance, thus all summations are over the training set dimensionality N rather than the basis set dimensionality N_B, which never appears in the GP. This is the essence of the "kernel trick". An example Gaussian process regression can be seen in Figure 1.
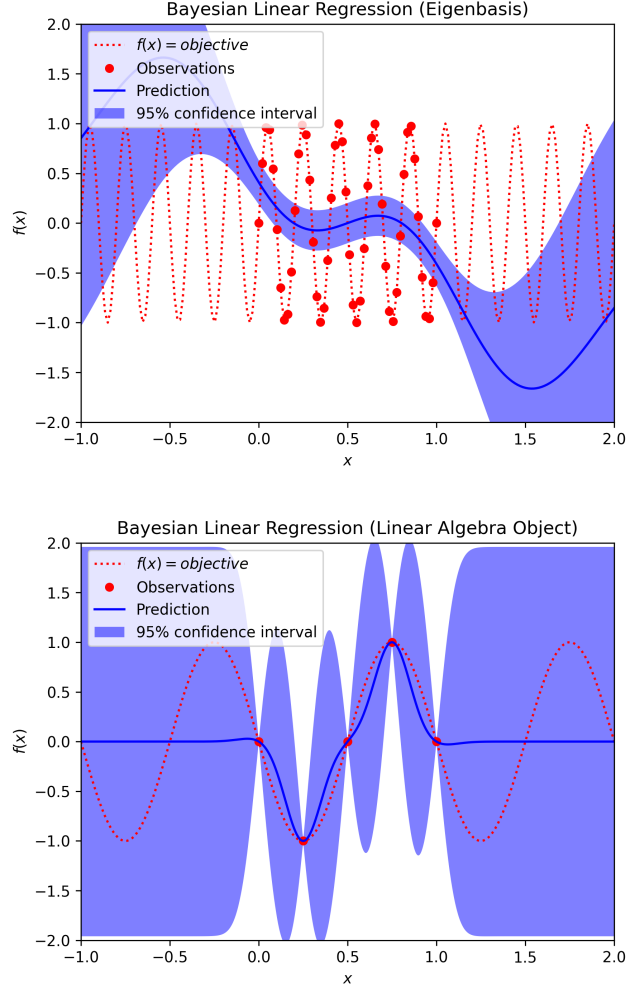
4

Figure 1: Regression results for all three approaches for an objective function that is a sine wave. Top: with wavelength set to 0.2, a Gaussian kernel with length scale set to 0.5, the noise parameter set to 0.1, with 1000 basis centers spread across a domain from -2 to 3, and 50 training data points spread equidistantly over a domain from 0 to 1. Bottom: with wavelength set to 1, length scale set to 0.1, noise parameter set to 0.01, with 1000 basis centers spread across a domain from -5 to 5, and 5 training data points spread equidistantly over a domain from 0 to 1. Two standard deviations of the prediction distribution are shown in blue. This figure was created with the associated code on Github.

5 Equating the Bayesian linear regression with the Gaussian process for a squared-exponential kernel

We aim to find set of basis functions that relate to the Gaussian process prior kernel according to

$$k_{\text{GP}}(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \phi^\top(\mathbf{x})\phi(\mathbf{x})$$
$$= \sigma_0^2 \sum_{i=1}^{N_B} k_{\text{BLR}}(\mathbf{x}, \mathbf{x}_i'')k_{\text{BLR}}(\mathbf{x}', \mathbf{x}_i'') \tag{22}$$
$$\mathbf{K} = k_{\text{GP}}(\mathbf{X}, \mathbf{X}) = \sigma_0^2 \mathbf{\Phi}^\top \mathbf{\Phi} + \sigma^2 \mathbf{I}_N$$

where N_B is the number of basis dimensions, our kernel bases are normalized so that $\phi^\top \phi = 1$, and the desired prediction standard deviation far from the training data, but without any additional noise σ, is defined as σ_0.

By Equation 21, varying σ_0 changes our prediction covariance but not our prediction means. Note that while Equation 22 resembles Equation 15, the definition for $\mathbf{K}$ uses the inner product, while the definition of $\mathbf{S}$ uses the outer product. Also, most definitions of the Gaussian process set $\sigma_0 \to 1$, which is usually justified by scaling the training data so that the empirical standard deviation of our targets, $\sigma_{\mathbf{y}}$, also goes 1 and therefore $\sigma_0 = \sigma_{\mathbf{y}}$, but the truth is that the scale can be arbitrarily chosen. We include the distinction here to help with the equivalence with BLR shortly. For now, what our choice of kernel shows is that if we scale our training targets, and expect σ_0 to scale accordingly so that $\sigma_0 \propto \sigma_{\mathbf{y}}$, then it is the ratio $\frac{\sigma}{\sigma_{\mathbf{y}}}$ that defines the relative noise of our kernel.

According to the reproducing kernel Hilbert space (RKHS) theory, there is one set of ψ that will accomplish this goal, but what is it? We consider the common case of

$$k_{\text{GP}}(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \exp\left(\frac{(\mathbf{x}' - \mathbf{x})^2}{2l^2}\right) + \sigma^2 \delta(\mathbf{x}, \mathbf{x}') \tag{23}$$

which is a squared-exponential kernel with white noise added for the variation we expect in repeated measurements at the same point. Note that l defines a length scale for the kernel.

Using the identity

$$\int_{-\infty}^{\infty} e^{-ax^2 + bx + c} dx = \sqrt{\frac{\pi}{a}} e^{\frac{b^2}{4a} + c} \tag{24}$$

we observe that

$$\int_{-\infty}^{\infty} \exp\left(\frac{(x_0 - x_0'')^2)}{2l^2}\right) \exp\left(\frac{(x_0' - x_0'')^2)}{2l^2}\right) dx'' \propto \exp\left(\frac{x_0 - x_0'}{4l^2}\right) \tag{25}$$

That is, if we perform an inner product over all possible $\mathbf{x}''$s for a basis function defined by a kernel with length scale l at points $\mathbf{x}$ and $\mathbf{x}'$, this is equivalent to computing a different kernel between x and $\mathbf{x}''$. To create a basis function set that will return a Gaussian kernel with length scale l, the kernel we use to generate that basis function set has the same form but with a length scale proportional to $l/\sqrt{2}$.

Looking back at Equations 18 and 21, we can make the following correspondences for the means

$$\overline{\mathbf{y}}_* \leftrightarrow \sigma^{-2}\phi_*\mathbf{\Sigma}\mathbf{\Phi}^\top\mathbf{y} \leftrightarrow \mathbf{K}_*^\top\mathbf{K}^{-1}\mathbf{y} \tag{26}$$

and the covariances

$$\mathrm{cov}(\mathbf{y}_*) \leftrightarrow \phi_*\mathbf{\Sigma}\phi_*^\top + \sigma^2\mathbf{I}_{N_*} \leftrightarrow \mathbf{K}_{**} - \mathbf{K}_*^\top\mathbf{K}^{-1}\mathbf{K}_* \tag{27}$$

. The corresponding expressions certainly look similar, but there are noticeable differences. Moreover, they involve matrix multiplications on totally different dimensions: over the training data for GPs, but over the basis dimensions for BLRs. It begs the question, what if there were some kernel $k_{\mathrm{BLR}}(\mathbf{x},\mathbf{x}')$ that could define a basis set

$$\phi(\mathbf{x}) = Z^{-1}(\mathbf{x})(k_{\mathrm{BLR}}(\mathbf{x},\mathbf{x}_0), k_{\mathrm{BLR}}(\mathbf{x},\mathbf{x}_1), \ldots, k_{\mathrm{BLR}}(\mathbf{x},\mathbf{x}_{N_B})) \tag{28}$$

with normalization constant $Z(\mathbf{x})$ such that the above correspondences could become identities? We normalize according to

$$Z(\mathbf{x}) = \sqrt{\sum_{i=0}^{N_B} k_{\mathrm{BLR}}(\mathbf{x},\mathbf{x}_i)^2} \tag{29}$$
$$1 = \phi(\mathbf{x})^\top\phi(\mathbf{x})$$

and by setting the prior covariance for the Bayesian linear regression proportional to the desired prediction standard deviation σ_0 far from the training data

$$\mathbf{\Sigma}_p = \sigma_0^2\mathbf{I} \tag{30}$$
$$\sigma_p = \sigma_0$$

we achieve the equivalence we've been looking for. In other words, we assume our prior weights have a mean of zero, are uncorrelated, and have a spread in their means equal to σ_0 far from the training data regardless of k_{BLR}, since the basis functions $\phi(\mathbf{x})$ are normalized by Equation 29.

We can deepen our intuition behind the prior noise parameter by first observing that close to the training data our prediction variances are close to zero, but far from the training data, the variance is equal to $\sigma_0^2 + \sigma^2$. To show this for the Gaussian process, we observe that $\mathbf{K}_0 \to 0$ so that

$$\mathrm{cov}(\mathbf{y}_0) = \mathbf{K}_{00} - \mathbf{K}_0^\top\mathbf{K}^{-1}\mathbf{K}_0 \tag{31}$$
$$\to \mathbf{K}_{00} = \sigma_0^2 + \sigma^2$$

To show this for the BLR, we start $\boldsymbol{\phi}_*^\top \boldsymbol{\Phi} \approx 0$ far from the training data. Due to the Woodbury matrix identity, we can derive

$$(I + UV)^{-1} = I - U(I + VU)^{-1}V \tag{32}$$

so that when we look far from the training data, the prediction covariance is

$$
\begin{aligned}
\text{cov}(\mathbf{y}_0) &= \boldsymbol{\phi}_0^\top \boldsymbol{\Sigma} \boldsymbol{\phi}_0 + \sigma^2 \\
&= \sigma^2 \left(\boldsymbol{\phi}_0^\top \left(\frac{\sigma^2}{\sigma_p^2} \mathbf{I}_{N_B} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \boldsymbol{\phi}_0 + 1 \right) \\
&= \sigma^2 \left(\frac{\sigma_p^2}{\sigma^2} \boldsymbol{\phi}_0^\top \left(\mathbf{I}_{N_B} + \frac{\sigma_p^2}{\sigma^2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \boldsymbol{\phi}_0 + 1 \right) \\
&= \sigma^2 \left(\frac{\sigma_p^2}{\sigma^2} \boldsymbol{\phi}_0^\top \left(\mathbf{I}_{N_B} - \frac{\sigma_p^2}{\sigma^2} \boldsymbol{\Phi} \left(\mathbf{I}_{N_B} + \frac{\sigma_p^2}{\sigma^2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \boldsymbol{\Phi}^\top \right) \boldsymbol{\phi}_0 + 1 \right) \\
&= \sigma_p^2 \boldsymbol{\phi}_0^\top \boldsymbol{\phi}_0 + \sigma^2 - \frac{\sigma_p^4}{\sigma^2} \boldsymbol{\phi}_0^\top \boldsymbol{\Phi} \left(\mathbf{I}_{N_B} + \frac{\sigma_p^2}{\sigma^2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\phi}_0 \\
&\approx \sigma_p^2 \boldsymbol{\phi}_0^\top \boldsymbol{\phi}_0 + \sigma^2
\end{aligned}
\tag{33}
$$

By Equations 30 and 29, we obtain

$$\text{cov}(\mathbf{y}_0) \approx \sigma_0^2 + \sigma^2 \tag{34}$$

In other words, the prior noise parameter determines the prediction variance far from our training data, since it determines the spread of our weights, multiplied by normalized basis functions, when they are not perturbed by our training data. Note that because of the basis normalization and our priors, if we include multiple kernels, for example a different length scale for each input dimension in a model that sums their weighted contributions rather than multiplies them, their concatenated basis functions need to be normalized by Equation 29, rather than each kernel being normalized independently.

If we mix kernel features with non-kernel features (e.g., linear, polynomial, etc.) in one model, we may normalize the kernels together, set the prior noise parameter for the kernel features to $\sigma_{\mathbf{y}}$, and set the L2 regularization constant to a predefined constant for the remaining features. In such a model, the normalized kernel features contribute a maximum prediction variance equal to the desired variance of our targets far from the training data, while the non-kernel features may contribute a prediction variance that far exceeds this value outside the domain of our training data, just as happens with standard Bayesian linear regression. Alternatively, one may apply an automatic relevance detection (ARD) regression and learn the effective prior noise parameters for each feature independently. However, a substantial drawback of ARD regression is that it doesn't have guarantees on convergence and is prone to underfitting, which is

also why there may be challenges putting automatic tuning of σ_p into production. We can, however, learn σ reliably using a sample of our training data, and we can learn σ_p or λ with cross-validation or other function minimization procedures.

Finally, we honor Equation 25 by setting

$$k_{\mathrm{BLR}}(x_0, x_0') = \exp\left(\frac{x_0 - x_0'}{4l^2}\right) \tag{35}$$

and we arrive at a Bayesian linear regression that produces the same predictions as the Gaussian process regression with kernel

$$k_{\mathrm{GP}}(x_0, x_0') = \exp\left(\frac{x_0 - x_0'}{2l^2}\right) \tag{36}$$

due to the normalization in Equation 29.

Note that in this case, the posterior covariance isn't proportional to k_{GP} for the same reason that it isn't proportional to K_{**} in Equation 21 or Σ_p^{-1} in Equation 18. Rather, if we describe the posterior covariance as a pure kernel function, it would look something like

$$\mathrm{cov}(\mathbf{y}_*) = k_{\mathrm{post.}}(\mathbf{x}_*, \mathbf{x}_*) = k_{\mathrm{equiv.}}(\mathbf{x}_*, \mathbf{x}) k_{\mathrm{equiv.}}^\top(\mathbf{x}_*, \mathbf{x}) \tag{37}$$

$$k_{\mathrm{equiv.}} = C\sigma\boldsymbol{\Sigma}^{1/2} k_{\mathrm{BLR}}^\top(\mathbf{x}_*, \mathbf{x}) \tag{38}$$

which means that inference on the posterior mixes up the basis functions defined by k_{BLR} according to the operator $\boldsymbol{\Sigma}^{1/2}$, which is defined by performing a summation over the training points (just like with a Gaussian process). For examples of this basis function set, see Figure 2.

What's a little weird about this equation is that it implies our covariance matrix would see its exponents divide by two due to the square root, but because of the identity of integrating over the product of two Gaussians in Equation 24, it actually multiplies by two. Similarly, in Section 7, the derivation for computing the log marginal likelihood of the BLR (Equation 58) requires integrating over the product of two multivariate norms. The result inverts its proportionality with the determinant of the covariance matrix compared to either of the original multivariate norms.

6 Equating the Bayesian linear regression with the Gaussian process for arbitrary kernels

The previous section covered how to accomplish the equivalence between the Bayesian linear regression and the Gaussian process for a squared-exponential kernel for the GP. One take-away was that the equivalent basis in the BLR has a different length scale and prefactor, but since we can just normalize out prefactors, all that matters is the difference in length scales between the two

formulations. The other main take-away is that we can use any arbitrary combination of our basis set and the BLR will learn new coefficients to account for that transformation, as evidenced in Equation 38.

The drawback with the previous approach was that, while intuitive, it leaves us in the dark for performing this equivalence for arbitrary kernels. However, we can use the fact that we can use any linear combination of basis functions to achieve the same result. If that's the case, then let's eschew the manual integration and instead define our basis as the eigenvectors of the kernel used in the GP evaluated at the same basis centers used in the previous analysis.

We first calculate our eigenvectors (rows of V) and eigenvalues (diagonal elements of D) of the kernel without the noise parameter for an arbitrary set of points in the independent space X' as follows:

$$\mathbf{V}\mathbf{D}\mathbf{V}^\top = k_{\mathrm{GP}}(\mathbf{X}', \mathbf{X}') - \sigma^2\mathbf{I} = \mathbf{K}' - \sigma^2\mathbf{I}_{N'} \tag{39}$$

giving us

$$\mathbf{V}\mathbf{D}\mathbf{V}^\top + \sigma^2\mathbf{I}$$
$$= \begin{bmatrix} k_{\mathrm{GP}}(\mathbf{x}_0, \mathbf{x}_0) & k_{\mathrm{GP}}(\mathbf{x}_0, \mathbf{x}_1) & \ldots & k_{\mathrm{GP}}(\mathbf{x}_0, \mathbf{x}_{N_B}) \\ k_{\mathrm{GP}}(\mathbf{x}_1, \mathbf{x}_0) & k_{\mathrm{GP}}(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k_{\mathrm{GP}}(\mathbf{x}_1, \mathbf{x}_{N_B}) \\ \vdots & \vdots & \ddots & \vdots \\ k_{\mathrm{GP}}(\mathbf{x}_{N_B}, \mathbf{x}_0) & k_{\mathrm{GP}}(\mathbf{x}_{N_B}, \mathbf{x}_1) & \ldots & k_{\mathrm{GP}}(\mathbf{x}_{N_B}, \mathbf{x}_{N_B}) \end{bmatrix} \tag{40}$$

Note that previous equations look for the eigendecomposition of the kernel evaluated at an arbitrary set of points in independent space, omitting the white noise (diagonal entries in matrix form) but keeping the radial basis function (a Gaussian in this case) as well as the mean function $m(\mathbf{x})$ or $\mathbf{m}(\mathbf{X})$. The solution presented in this section works from these eigenvectors. To obtain a non-zero (mean) prediction far away from the data, we can either add subtract off the mean and add it back in during prediction, fit the intercept as a fixed hyperparameter, or concatenate a constant (usually 1) to the feature vectors $\phi(\mathbf{x})$ and otherwise fit as normal. By comparison, GP practitioners either subtract off the mean or fit the mean through a constant added to every pair of points in the kernel (see Section 7. In both cases, adding the mean prediction increases the prediction variance at each point, in the BLR case due to the variance in the intercept term, and in the GP through the constant added to the kernel.

To project our observations into this space, we convolve them with an normalized infinitesimal-width Gaussian kernel, that is, a dirac delta function δ, but for computational purposes we choose a finite width that is equivalent to the distance between adjacent points in our sampling of the independent space $\mathbf{X}'$. While the choice is arbitrary, it makes sense to reuse the centers to generate our basis set, that is, to set $\mathbf{X}' = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{N_B})$.
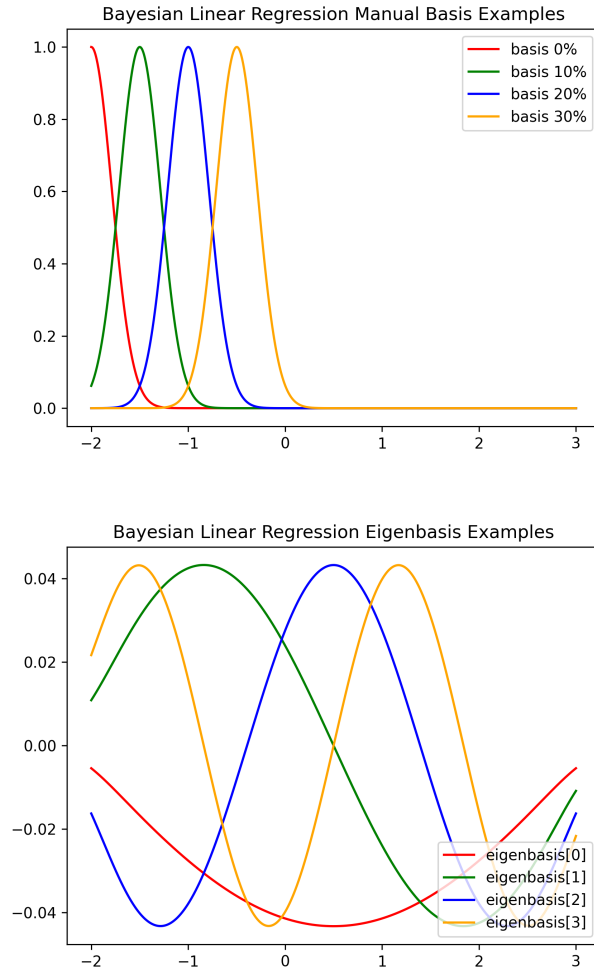
We thus perform this projections as follows:

Figure 2: Top: The manual basis functions, which are length-scale-adjusted Gaussian distributions, centered at 0%, 10%, 20%, and 30% along the domain. Bottom: The eigenbasis functions V for the largest four eigenvalues. Note that the sign of each function is arbitrary (can be flipped over the x-axis), and that these functions resemble the harmonic modes for an equivalent one-dimensional thick string with fixed endpoints one length scale outside the domain of our basis centers (off-plot).
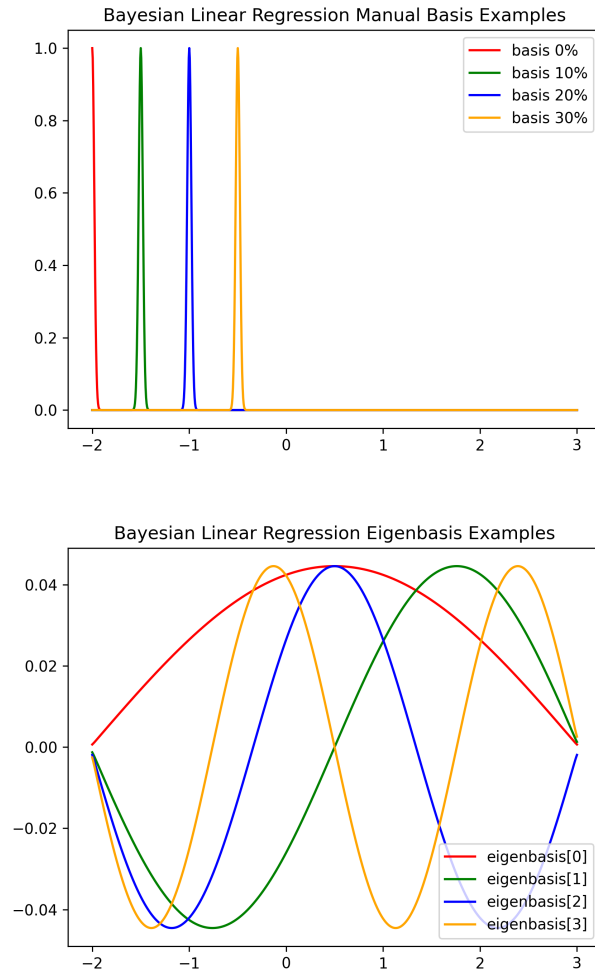
Figure 3: Same as Figure 2, but for a kernel with a length scale $1/10^{\text{th}}$ the size. We see that the fixed endpoints of the hypothetical string are now right at the end of the basis center domain since the length scale of the kernel is so small.

$$\mathbf{X}_\delta = \begin{bmatrix} \delta(\mathbf{x}_0, \mathbf{x}_0) & \delta(\mathbf{x}_0, \mathbf{x}_1) & \dots & \delta(\mathbf{x}_0, \mathbf{x}_{N_B}) \\ \delta(\mathbf{x}_1, \mathbf{x}_0) & \delta(\mathbf{x}_1, \mathbf{x}_1) & \dots & \delta(\mathbf{x}_1, \mathbf{x}_{N_B}) \\ \vdots & \vdots & \ddots & \vdots \\ \delta(\mathbf{x}_N, \mathbf{x}_0) & \delta(\mathbf{x}_N, \mathbf{x}_1) & \dots & \delta(\mathbf{x}_N, \mathbf{x}_{N_B}) \end{bmatrix} \tag{41}$$

and then project onto the eigenbasis:

$$\begin{aligned} \mathbf{\Phi} &= \mathbf{X}_\delta \sqrt{\mathbf{D}} \mathbf{V} \\ \mathbf{\Phi}\mathbf{\Phi}^\top &= \mathbf{V}\mathbf{D}\mathbf{V}^\top \\ &= \mathbf{K}' - \sigma^2 \mathbf{I}_{N'} \end{aligned} \tag{42}$$

and similarly for our test data points at $\mathbf{X}_*$. We then perform the same procedure as for Bayesian linear regression (Equation 18). For examples of this basis function set, see Figure 2.

What happens to the eigenbasis when we change the width of the kernel? For small length scales (see Figure 3), the eigenbasis resembles the vibrational modes of a string (in one dimension, a plane in two dimensions, etc.) bounded by the domain of the basis centers, as if the string is perfectly fixed at its endpoints like on a guitar. As the length scale grows larger, as in Figure 2, we see that the eigenbases actually taper to the edges of the basis center domain with a length scale proportional to the one used for the kernel. The best possible approximation extends the domain of basis centers to plus and minus infinity, and when we take this approach in our calculations, we find that the same intuition applies but the interpretation changes slightly. The eigenbases with the highest eigenvalues resemble sine-waves in the training data domain and the wavelengths of adjacent eigenbases increment proportionally to the kernel wavelength. Similarly, the Taylor expansion of a Gaussian involves sine and cosine waves of differing wavelengths whose differences are proportional to the Gaussian length scale.

There is meaning held in the eigenvalues diag($\mathbf{D}$). If we look at the eigenvalues for the linear system

$$\left(\sigma_p^{-2} \mathbf{\Phi}\mathbf{\Phi}^\top \right) \mathbf{u}_i = \mathbf{u}_i \tag{43}$$

defined by $(\lambda_1, \lambda_2, \dots, \lambda_N) = \sigma_p^{-2} \text{diag}(\mathbf{D})$, by Section 3.5.3 of [2], we see that the quantity

$$\gamma = \sum_{i=1}^{N} \frac{\lambda_i}{\sigma_p^{-2} + \lambda_i} \tag{44}$$

defines the effective number of parameters by which we can alter our noise parameter

$$\sigma_{\text{unbaised}} = \sigma \frac{N}{N - \gamma} \tag{45}$$

to get the unbaised covariance matrix

$$\mathbf{\Sigma}_{\text{unbaised}}^{-1} = \sigma_{\text{unbiased}}^{-2} \mathbf{\Phi}\mathbf{\Phi}^\top + \mathbf{\Sigma}_p^{-1} \tag{46}$$

13

of our model parameters $\boldsymbol{\theta}$.

Note that whether we use the biased or unbiased noise parameter and covariance matrix does not affect our mean predictions, as shown in Equations 15 through 18 , at least from the perspective where the regularization constant defined in Equation 6 is kept constant (not to be confused with the eigenvalues λ_i). Also note that Equation 44 establishes an implicit solution for σ_p which can be used to maximize the marginal likelihood over σ_p through iteration if we don't set it to $\sigma_{\mathbf{y}}$, although instabilities arise if the iterations for both σ_p and σ run concurrently, causing the uninformative result $\sigma_p \to 0$, $\sigma \to \sigma_{\mathbf{y}}$, and $\overline{\boldsymbol{\theta}} \to \mathbf{0}$.

When [2] examines Equation eq:gamma, the text is mostly interested in how γ alters with σ_p by way of the regularization constant λ. However, we fix $\sigma_p = \sigma_{\mathbf{y}}$ when equating BLR with GP so we are more interested in how γ alters with our other model hyperparameters. Given the same set of training data of N rows, we find that we can cause γ to asymptotically approach N with two changes: increase the number of basis functions (centers of Gaussian distributions), and/or decrease the noise parameter σ. In other words, both changes to the problem increase the ability and need of the model to resolve more information than can be offered by just a few (well-chosen) basis functions. In the regime of many data points and fewer basis functions, γ will asymptotically approach the number of basis functions as we increase σ.

We can perform a couple of simplifications to the procedure outlined in this section by considering two ways to reduce the computational complexity: (1) reduce the number of basis centers so that they are spaced according to the length scales of the original kernel and (2) reduce the number of eigenvectors (by filtering out those with small eigenvalues). The first works well, except when the variation in the training set y-values is greater than can be afforded by the length scale of the kernel. A GP or equivalent BLR will fit more closely to these widely-varying observations than one with a sparser basis set.

Similarly, the second approach loses spatial resolution in proportion to how many eigenvectors are discarded, and in our experience, we found this approach to be very prone to meaningfully changing your calculations. Note that just as is the case with the manually-derived basis functions, the BLR will only produce equivalent predictions within the domain that the basis functions are not infinitesimally small.

7 Tuning the noise parameter and kernel hyperparameters in both formalisms

There are only a few models in common circulation that allow you to learn the parameters using explicit and deterministic update rules. Of the two most prominent, well-studied, and easy-to-implement algorithms, one is the Bayesian linear regression with a fixed prior noise parameter or L2 regularization constant, and optionally with a fixed data noise parameter (the current discussion cover the case of fixed noise parameter, [1] covers the learned case). The other is

the Gaussian process, with a fixed data noise parameter, as discussed in this document. As we have covered, both are actually parts of the same continuum between high featurization and large amounts of data. The secret ingredient for these models is that due to their assumptions, it is possible to analytically integrate the prediction

$$p_{\boldsymbol{\theta}}(\mathbf{y}_*|\mathbf{X}_*, \boldsymbol{\sigma}) = \int p(\mathbf{y}_*|\mathbf{X}_*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\sigma})d\boldsymbol{\theta} \tag{47}$$

when $p(\mathbf{y}_*|\mathbf{X}_*, \boldsymbol{\theta})$ is defined by a multivariate norm centered on a function defining the mean (and mode) and another function defining the covariance. Here, we represent the hyperparameters of the model by the vector of noise parameters $\boldsymbol{\sigma}$ which could include σ, σ_p, or both. In fact, the hyperparameters could also include the lengths scales of our kernel or basis functions, and even individual prior noise parameters for each data point. Note that the model parameters $\boldsymbol{\theta}$ are defined by the measured function values $\mathbf{f}(\mathbf{X})$ for the Gaussian process, while in the Bayesian linear regression they are the coefficients of the basis functions.

Alternatively, many models for which the prediction integral is intractable simply perform one prediction at the set of parameters $\boldsymbol{\theta}$ at the maximum a posteriori (MAP) when there is a prior $p(\boldsymbol{\theta})$, or the maximum likelihood estimate (MLE) when there is no prior or it is uniform and uninformative. In the Laplace approximation, which treats all models like they were Bayesian linear regressions, the spread in the probability mass for the model parameters around the MAP or MLE is modeled as a multivariate Gaussian whose covariance matrix is related to the Fisher information matrix, allowing us to compute a tractable prediction integral again.

The posterior distribution for our targets is provided by

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y})p(\mathbf{y})}{p_{\mathbf{y}}(\boldsymbol{\theta}|\mathbf{X})} \tag{48}$$

or equivalently

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\sigma}) = \frac{p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}, \boldsymbol{\sigma})p(\mathbf{y}|\boldsymbol{\sigma})}{p_{\mathbf{y}}(\boldsymbol{\theta}|\mathbf{X}, \boldsymbol{\sigma})} \tag{49}$$

and the inverse relationship is

$$p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}, \boldsymbol{\sigma}) = \frac{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\sigma})p(\boldsymbol{\theta}|\boldsymbol{\sigma})}{p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma})} \tag{50}$$

where

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma}) = \int p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\sigma})d\boldsymbol{\theta} \tag{51}$$

is the normalization factor, marginal likelihood, or the model evidence from Bayes' theorem (posterior = likelihood × prior / evidence). Since we are usually interested in maximizing the posterior with respect to $\mathbf{y}$ or $\boldsymbol{\theta}$, we usually ignore the denominators in the above equations, unless we decide to also optimize our hyperparameters, which we will now discuss.

We can consider this model to be hierarchical – first we set the model hyperparameters $\boldsymbol{\sigma}$, giving us our prior distribution over the model parameters $\boldsymbol{\theta}$, and then we compute our posteriors. While we expect our model to learn its parameters, we may also want a model to learn the hyperparameters. This requires us to go up one level in the hierarchy, where we look at the posterior distribution of our hyperparameters

$$p_{\boldsymbol{\theta}}(\boldsymbol{\sigma}|\mathbf{X}, \mathbf{y}) = \frac{p_{\boldsymbol{\theta}(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma})}p(\boldsymbol{\sigma})}{p_{\boldsymbol{\sigma}}(\mathbf{y}|\mathbf{X})} \tag{52}$$

where

$$p_{\boldsymbol{\sigma}}(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma})p(\boldsymbol{\sigma})d\boldsymbol{\sigma} \tag{53}$$

and the inverse relationship (which we don't use) is

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma}) = \frac{p_{\boldsymbol{\theta}(\boldsymbol{\sigma}|\mathbf{X}, \mathbf{y})}p(\mathbf{y})}{p_{\mathbf{y}}(\boldsymbol{\sigma}|\mathbf{X})} \tag{54}$$

That is, the MAP or MLE of the hyperparameters will occur when we maximize the marginal likelihood in Equation 51, which is also known as the evidence of the model itself when we integrate over the parameters.

The above discussion leaves you to wonder that, since there is uncertainty in your hyperparameters when we don't fix them in advance and instead fit them along with the model parameters, how their uncertainty would add to the uncertainty of your predictions. Since Equation 53 is generally intractable, we use the Type-II maximum likelihood estimation, or empirical Bayes approximation, whereby we keep our hyperparameters fixed for several different guesses and then learn the MLE of the influenced parameters, and maximize with respect to the hyperparameters we explore over. If we are comfortable with using the MAP or MLE for our mean and the Laplace approximation (defining the spread in hyperparamters by the Fisher information matrix or local curvature), we can stop here. A well-thought out discussion of why we should be satisfied with this approximation can be found in [3].

Using Equation 52, we can compare two models with hyperparameters $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$,

$$\frac{p(\boldsymbol{\sigma}_1|\mathbf{X}, \mathbf{y})}{p(\boldsymbol{\sigma}_2|\mathbf{X}, \mathbf{y})} = \overbrace{\frac{p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma}_1)}{p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma}_2)}}^{\text{Bayes' factor}} \frac{p(\boldsymbol{\sigma}_1)}{p(\boldsymbol{\sigma}_2)} \tag{55}$$

giving us Bayes' factors (used in the Bayesian information criterion, or BIC). This tells us that we can compare them using their marginal likelihoods without having to perform the nasty integrals from the denominator, assuming that the prior distributions of the hyperparameters are identical.

We compute the marginal likelihood by integrating the product of the multivariate normal likelihood distribution dependent on $\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}, \boldsymbol{\Sigma}, \sigma$, and the prior multivariate normal prior distribution of the parameters dependent on σ_p, over the model parameters $\boldsymbol{\theta}$. Meanwhile, the normalized multivariate distribution

16

$\mathbf{y}|\mathbf{X} \sim \mathcal{N}(\mathbf{y}|\mathbf{m}, \boldsymbol{\Sigma}_f)$ is given by

$$\mathcal{N}(\mathbf{y}|\mathbf{m}, \boldsymbol{\Sigma}_f) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}_f|}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{m})^\top \boldsymbol{\Sigma}_f^{-1}(\mathbf{y} - \mathbf{m})\right)$$

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}(\mathbf{y} - \mathbf{m})^\top \boldsymbol{\Sigma}_f^{-1}(\mathbf{y} - \mathbf{m}) - \frac{1}{2}\log|\boldsymbol{\Sigma}_f| - \frac{N}{2}\log 2\pi$$

$$(56)$$

We thus express our marginal likelihood as

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma}) = \int p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\sigma})d\boldsymbol{\theta}$$

$$= \int \mathcal{N}(\mathbf{y}|\boldsymbol{\Phi}\boldsymbol{\theta}, \sigma^2 \mathbf{I}_N)\mathcal{N}(\boldsymbol{\theta}|\mathbf{0}_{N_B}, \sigma^2 \mathbf{I}_{N_B})d\boldsymbol{\theta}$$

$$(57)$$

After completing the squares on the two squared exponentials and normalizing[1], we arrive a Equation 3.86 from [2]:

$$\log p_{\boldsymbol{\theta}}(\mathbf{y}|\sigma_p, \sigma)$$

$$= -\frac{1}{2\sigma^2}\left(||\mathbf{y} - \boldsymbol{\Phi}\overline{\boldsymbol{\theta}}||^2 + \frac{\sigma^2}{\sigma_p^2}\overline{\boldsymbol{\theta}}^\top \overline{\boldsymbol{\theta}}\right)$$

$$- \frac{1}{2}\left(\log|\boldsymbol{\Sigma}^{-1}| + 2N\log\sigma + 2N_B\log\sigma_p\right)$$

$$- \frac{N}{2}\log 2\pi$$

$$(58)$$

Note that the data term $-\frac{1}{2\sigma^2}||\mathbf{y} - \boldsymbol{\Phi}\overline{\boldsymbol{\theta}}||^2$ becomes a constant $(-\frac{1}{2})$ if we learn σ from the residuals, and that we are expressing our solution in terms of the mean (learned) model parameters $\overline{\boldsymbol{\theta}} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top$ from Equation 16.

For the Gaussian process, we also obtain the log marginal likelihood through integration over two Gaussians. In this case, however, we integrate over the measured input data $\mathbf{f}(\mathbf{X})$, instead of a set of explicit model parameters $\boldsymbol{\theta}$, so that

$$p_{\mathbf{f}}(\mathbf{y}|\mathbf{X}, \boldsymbol{\sigma}) = \int p(\mathbf{y}|\mathbf{X}, \mathbf{f})p(\mathbf{f}|\mathbf{X}, \boldsymbol{\sigma})d\mathbf{f}$$

$$= \int \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 \mathbf{I}_N)\mathcal{N}(\mathbf{f}|\mathbf{0}_N, \mathbf{K}_{\mathrm{RBF}})d\mathbf{f}$$

$$(59)$$

Using the identity in Equation A.7 from [5] with the values

$$\mathbf{a} = \mathbf{f}$$

$$\mathbf{A} = \sigma^2 \mathbf{I}_N$$

$$\mathbf{b} = \mathbf{0}_N$$

$$\mathbf{B} = \mathbf{K}_{\mathrm{RBF}}$$

$$(60)$$

---

[1]Similarly, you can evaluate the integral from Equation 3.77 using Equation 2.115 in Exercise 3.16 from [2]. Bishop's book doesn't cover the derivation, but you can find solutions for Exercise 3.16 online. A detailed and satisfying derivation can be found at http://www.utstat.utoronto.ca/ radford/sta414.S11/week4a.pdf

and noting that $\mathcal{N}(\mathbf{x}|\mathbf{y}, \boldsymbol{\Sigma}_f) = \mathcal{N}(\mathbf{y}|\mathbf{x}, \boldsymbol{\Sigma}_f)$, we arrive at Equations 2.30 and 5.8 in [5]:

$$
\begin{aligned}
\log p(\mathbf{y}|\mathbf{X}, \mathbf{K}_{\mathrm{RBF}}) = & -\frac{1}{2}\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y} \\
& -\frac{1}{2}\log|\mathbf{K}| \\
& -\frac{N}{2}\log 2\pi
\end{aligned}
\tag{61}
$$

where the vertical bars represent the determinant operator and $\mathbf{K} = \sigma_0^2 \mathbf{K}_{\mathrm{RBF}} + \sigma^2 \mathbf{I}_N$. We can make $\bar{\mathbf{f}}$ non-zero by adding a constant between every pair of points to the kernel $\mathbf{K}_{\mathrm{RBF}}$. Given these likelihoods, you can also optimize the noise parameter and length scales of the equivalent BLR the same way that is canon with GPs: you perform multiple gradient descents in the negative log marginal likelihood, since the landscape is pretty smooth, although not convex, and therefore may have multiple local minima.

Diving in more deeply, we see that the first terms, which represent the data fit, are equivalent:

$$
\sigma^{-2}||\mathbf{y} - \overline{\boldsymbol{\theta}}^\top \Phi||^2 + \sigma_p^{-2}\overline{\boldsymbol{\theta}}^\top \overline{\boldsymbol{\theta}} = \mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}
\tag{62}
$$

and at first it may seem surprising to include the regularization term proportional to $\overline{\boldsymbol{\theta}}^\top \overline{\boldsymbol{\theta}}$ in this equation. Keep in mind that the regularization term is enforced in the BLR by the prior with uncorrelated weights with a spread equivalent to a pre-defined noise parameter σ_p, and while the regularization is enforced in the GP by the prior kernel which includes the noise parameter in its white noise component, that is, it is the sum of the white noise kernel and the radial basis function kernel.

The second terms, which represent model complexity, are also equivalent since

$$
\begin{aligned}
\frac{1}{\sigma^2}\mathbf{K} &= \mathbf{I}_N + \frac{\sigma_0^2}{\sigma^2}\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \\
\sigma_p^2 \boldsymbol{\Sigma}^{-1} &= \mathbf{I}_{N_B} + \frac{\sigma_p^2}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top
\end{aligned}
\tag{63}
$$

By the identity $|\mathbf{I}_{N_{\mathrm{inner}}} + \mathbf{U}\mathbf{V}^\top| = |\mathbf{I}_{N_{\mathrm{outer}}} + \mathbf{V}^\top \mathbf{U}|$ and by setting $\sigma_p = \sigma_0$ we obtain

$$
|\boldsymbol{\Sigma}^{-1}|\sigma_p^{2N_B} = |\mathbf{K}|\sigma^{-2N}
\tag{64}
$$

$$
-\log|\boldsymbol{\Sigma}| + 2N_B \log \sigma_p = \log|\mathbf{K}| + 2N \log \sigma
\tag{65}
$$

$$
\log|\boldsymbol{\Sigma}^{-1}| + 2N \log \sigma + 2N_B \log \sigma_p = \log|\mathbf{K}|
\tag{66}
$$

The third terms are trivially equivalent.

There are many ways to learn the noise parameter σ, in particular because it doesn't affect the other learned model parameters in a BLR except through

regularization when σ_p is fixed or learned. First, when inputs or basis functions are orthogonal (which is not the case for the Gaussian basis functions considered in this document), when the regularization constant is fixed, and during updates one-row-at-a-time, σ can be learned using linear algebra identities according to Equation 3.73 in [4]. Second, with either fixed λ or σ_p, we can determine σ empirically, using a randomized subset of the training data as an approximation if necessary. Lastly, in all cases, we can learn the noise parameter for the BLR by minimizing the negative log marginal likelihood, using a gradient descent method for example, just as we can learn the kernel hyperparameters in a GP. These methods work very well for single hyperparameters, but become less reliable for multiple ones, since they require multiple gradient descents and are prone to finding local minima rather than the global minimum.

For learning just λ or σ_p with fixed σ, there are several implicit (circular) solutions, such as the ones presented in Sections 3.5.2 and 9.3.4 from [2], but they require multiple iterations over the dataset. There is a silver lining, however. Online, incremental, or sequential versions of the BLR can update λ or σ_p at each step from an initial guess (usually taken to be close to zero) based on the observed or fixed σ determined by the previous guess. Thus, the implicit solution is iterated over automatically as we add each new row of data. The drawback with this approach is that updating your BLR posteriors one row at a time is computationally inefficient compared to batch processing. On the other hand, there are many applications, such as contextual multi-armed bandits and hyperparameter tuning, where obtaining each row of data is time-intensive, so that incremental solutions are preferred. In these cases, feast on that free lunch all you want.

References

[1]   URL: https://github.com/KoyoteScience/BayesianLinearRegressor.

[2]   Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[3]   D. J. C. MacKay. "Comparison of Approximate Methods for Handling Hyperparameters". In: *Neural Computation* 11.5 (1999), pp. 1035–1068.

[4]   Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN: 9780262018029 0262018020.

[5]   CE. Rasmussen and CKI. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, Jan. 2006, p. 248.