Practical Gaussian Process Regression

Joshua Corneck
Imperial College London

March 24, 2025

Table of contents

Motivating the Gaussian process

Outline of the plan

- Review Bayesian linear regression.
- Consider transforming from input space using feature maps for improved flexibility.
- Examine limitations of basis functions.
- Introduce the kernel trick for allowing to scale the number of basis functions.
- Introduce Mercer's theorem to motivate using kernels directly.
- Discuss the function-space interpretation.

Bayesian linear regression

Consider the standard regression setting $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and posit a linear relationship for each i:

$$y_i = f(\boldsymbol{x}_i; \boldsymbol{\beta}) + \epsilon, \qquad f(\boldsymbol{x}_i; \boldsymbol{\beta}) = \boldsymbol{x}_i^T \boldsymbol{\beta}, \qquad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2).$$

This gives a likelihood of the form

$$p(\boldsymbol{y} \mid \boldsymbol{\beta}, X, \sigma_\epsilon^2) = \prod_{i=1}^n \mathcal{N}(y_i \mid \boldsymbol{x}_i^T \boldsymbol{\beta}, \sigma_\epsilon^2). \tag{1}$$

where $X \in \mathbb{R}^{n \times d}$.

A *frequentist* takes this likelihood and maximises it to get the *maximum likelihood estimator*:

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}. \tag{2}$$

Bayesian linear regression

Computing the posterior

A Bayesian instead places a prior on the model parameters:

$$\boldsymbol{\beta} \sim \mathcal{N}(0, \Sigma),$$

and computes the posterior distribution of $\boldsymbol{\beta}$

$$p(\boldsymbol{\beta} \mid X, \boldsymbol{y}) \propto p(\boldsymbol{y} \mid X, \boldsymbol{\beta}) p(\boldsymbol{\beta}).$$

We can show

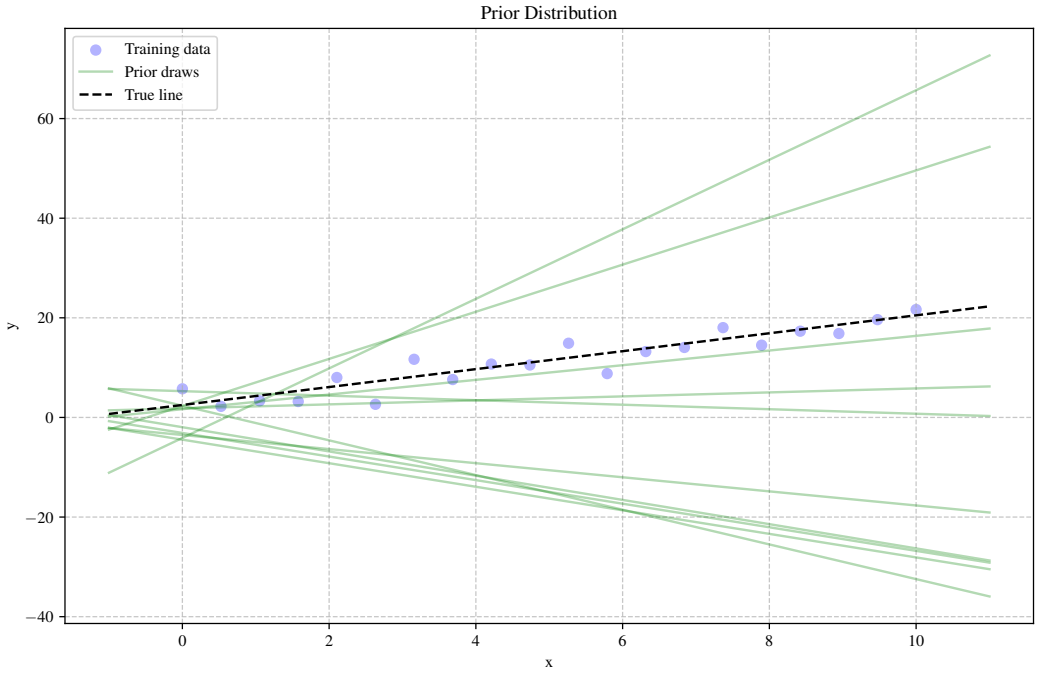
$$\boldsymbol{\beta} \mid X, \boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_\beta, \Lambda_\beta^{-1}\right),$$

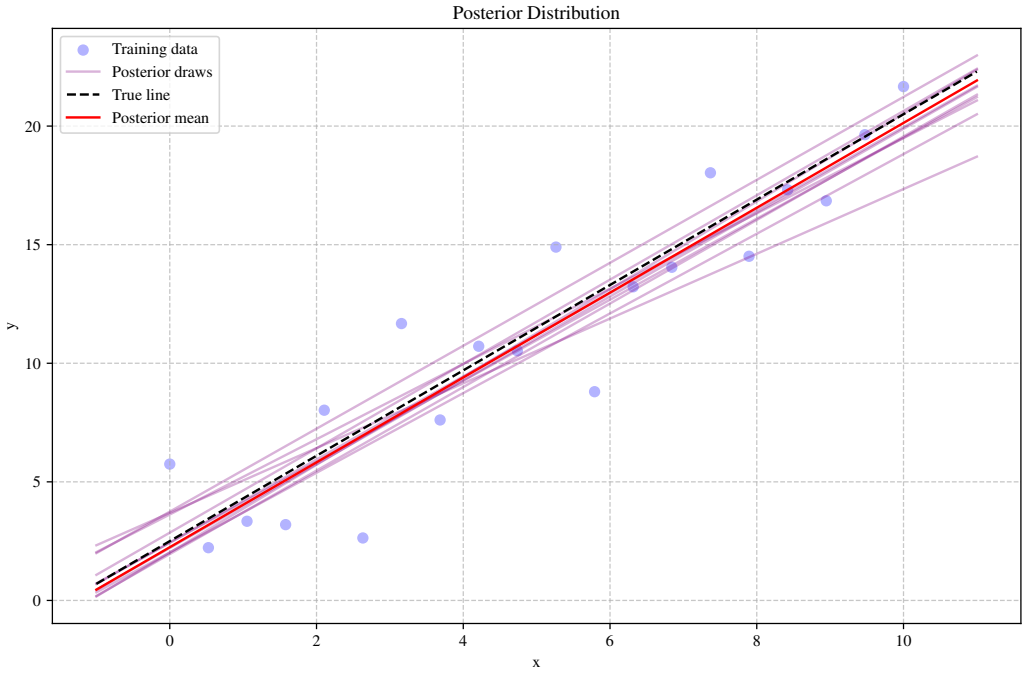
where

$$\boldsymbol{\mu}_\beta = \frac{1}{\sigma_\epsilon^2} \Lambda_\beta^{-1} X^T \boldsymbol{y}, \qquad \Lambda_\beta = \frac{1}{\sigma_\epsilon^2} X^T X + \Sigma^{-1}.$$

Bayesian linear regression

Draws from the prior and posterior





Bayesian linear regression

Relation to the MLE

Consider our posterior distribution:

$$\boldsymbol{\beta} \mid X, \boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_\beta, \Lambda_\beta^{-1}\right),$$

with

$$\boldsymbol{\mu}_\beta = \frac{1}{\sigma_\epsilon^2}\Lambda_\beta^{-1} X^T \boldsymbol{y}, \qquad \Lambda_\beta = \frac{1}{\sigma_\epsilon^2} X^T X + \Sigma^{-1}.$$

How does this relate to the MLE obtained from the frequentist approach?

Bayesian linear regression

Relation to the MLE

Consider our posterior distribution:

$$\boldsymbol{\beta} \mid X, \boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_\beta, \Lambda_\beta^{-1}\right),$$

with

$$\boldsymbol{\mu}_\beta = \frac{1}{\sigma_\epsilon^2}\Lambda_\beta^{-1} X^T \boldsymbol{y}, \qquad \Lambda_\beta = \frac{1}{\sigma_\epsilon^2} X^T X + \Sigma^{-1}.$$

How does this relate to the MLE obtained from the frequentist approach? Notice

$$\begin{aligned}
\boldsymbol{\mu}_\beta &= \frac{1}{\sigma_\epsilon^2}\Lambda_\beta^{-1} X^T \boldsymbol{y} \\
&= \frac{1}{\sigma_\epsilon^2}\left(\frac{1}{\sigma_\epsilon^2} X^T X + \Sigma^{-1}\right)^{-1} X^T \boldsymbol{y} \\
\left(\text{uninformative prior: } \lim_{\Sigma^{-1}\to 0}\right) \quad &= \frac{1}{\sigma_\epsilon^2}\left(\frac{1}{\sigma_\epsilon^2} X^T X\right)^{-1} X^T \boldsymbol{y} \\
&= \hat{\boldsymbol{\beta}}.
\end{aligned}$$

Bayesian linear regression

Prediction for $\boldsymbol{y}_*$

We now have a new set of inputs $X_* \in \mathbb{R}^{m \times d}$ and want to predict $\boldsymbol{y}_* \in \mathbb{R}^m$.

Bayesian linear regression

Prediction for $\boldsymbol{y}_*$

We now have a new set of inputs $X_* \in \mathbb{R}^{m \times d}$ and want to predict $\boldsymbol{y}_* \in \mathbb{R}^m$.

In a frequentist setting, we use $\hat{\boldsymbol{\beta}}$ to make predictions:

$$\boldsymbol{y}_* = f(X_*; \hat{\boldsymbol{\beta}}).$$

A Bayesian has the advantage of a full distribution, which we leverage by *averaging* over all possible parameter values using our posterior as a weighting:

$$p(\boldsymbol{y}_* \mid X_*, X, \boldsymbol{y}) = \int p(\boldsymbol{y}_* \mid X_*, \boldsymbol{\beta}) p(\boldsymbol{\beta} \mid X, \boldsymbol{y}) d\boldsymbol{\beta},$$

Bayesian linear regression

Prediction for $\boldsymbol{y}_*$

We now have a new set of inputs $X_* \in \mathbb{R}^{m \times d}$ and want to predict $\boldsymbol{y}_* \in \mathbb{R}^m$.

In a frequentist setting, we use $\hat{\boldsymbol{\beta}}$ to make predictions:

$$\boldsymbol{y}_* = f(X_*; \hat{\boldsymbol{\beta}}).$$

A Bayesian has the advantage of a full distribution, which we leverage by *averaging* over all possible parameter values using our posterior as a weighting:

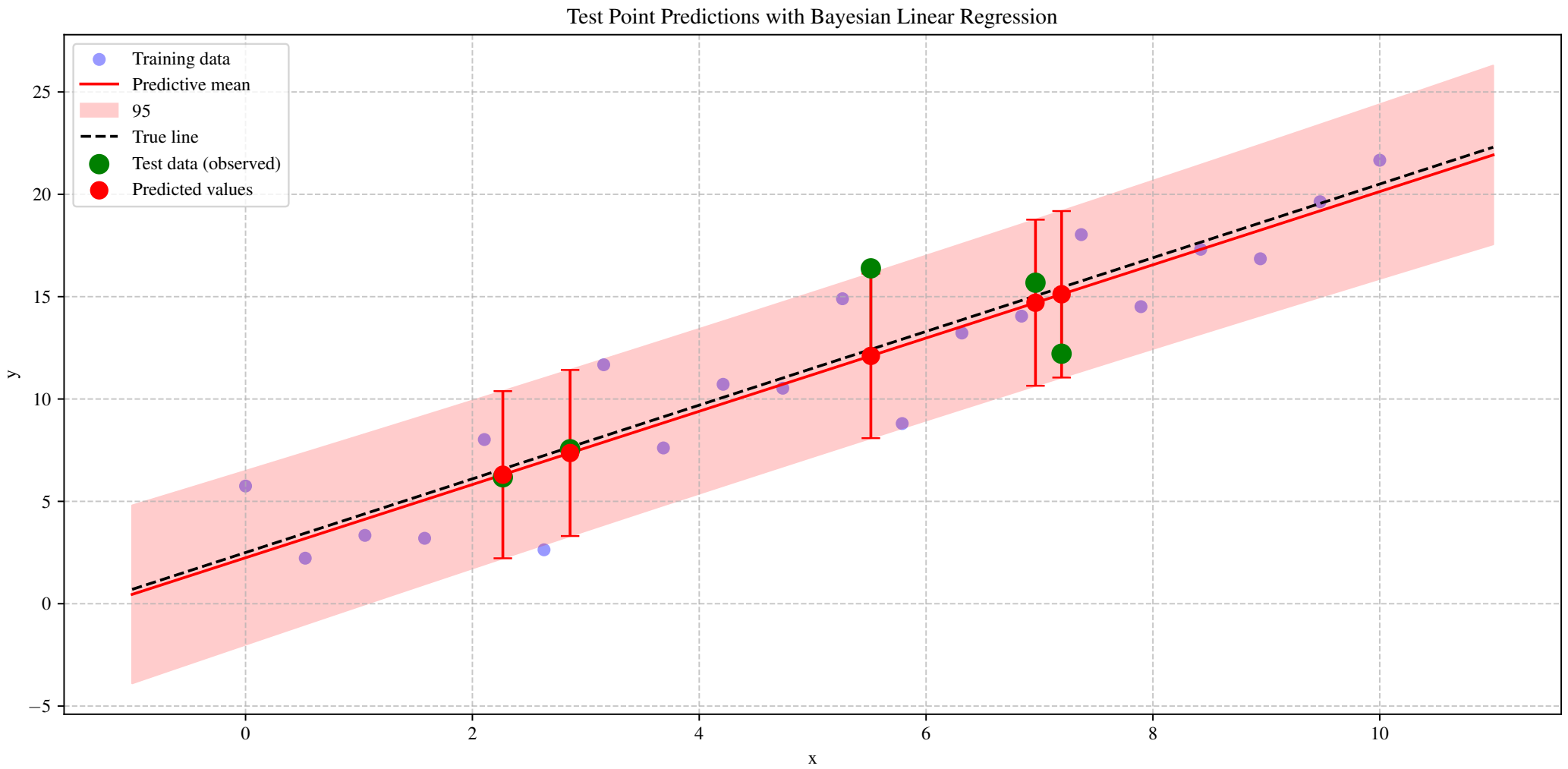
$$p(\boldsymbol{y}_* \mid X_*, X, \boldsymbol{y}) = \int p(\boldsymbol{y}_* \mid X_*, \boldsymbol{\beta}) p(\boldsymbol{\beta} \mid X, \boldsymbol{y}) d\boldsymbol{\beta},$$

We can show that this is also Gaussian:

$$\begin{aligned}
\boldsymbol{y}_* \mid X_*, X, \boldsymbol{y} &\sim \mathcal{N}\left(X_* \boldsymbol{\mu}_\beta, X_* \Lambda_\beta^{-1} X_*^T + \sigma_\epsilon^2 I_m\right) \\
&= \mathcal{N}\left(\frac{1}{\sigma_\epsilon^2} X_* \left(\frac{1}{\sigma_\epsilon^2} X^T X + \Sigma^{-1}\right)^{-1} X^T \boldsymbol{y}, X_* \left(\frac{1}{\sigma_\epsilon^2} X^T X + \Sigma^{-1}\right)^{-1} X_*^T + \sigma_\epsilon^2 I_m\right).
\end{aligned}$$

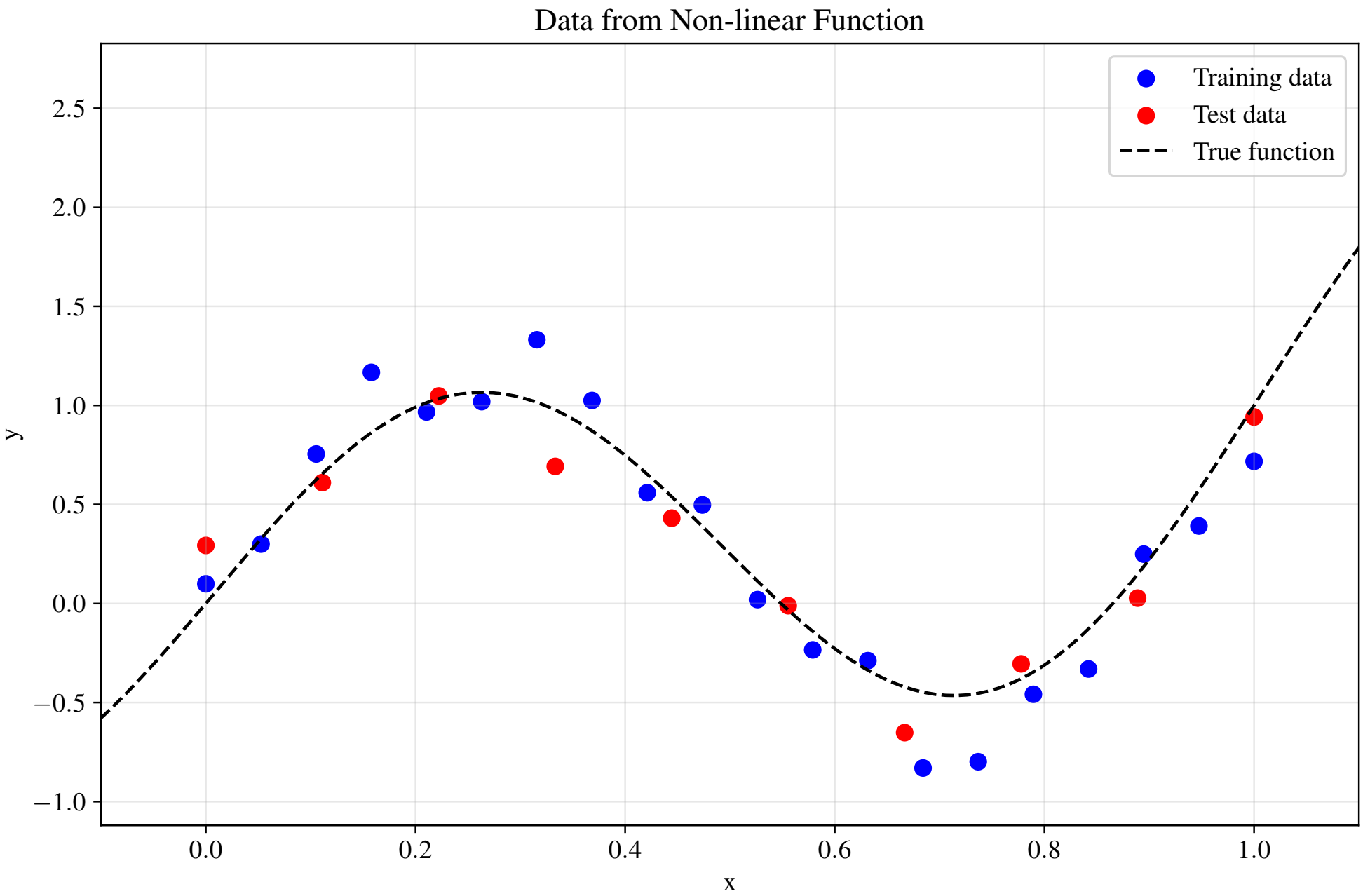
Bayesian linear regression

Predictive distribution confidence intervals



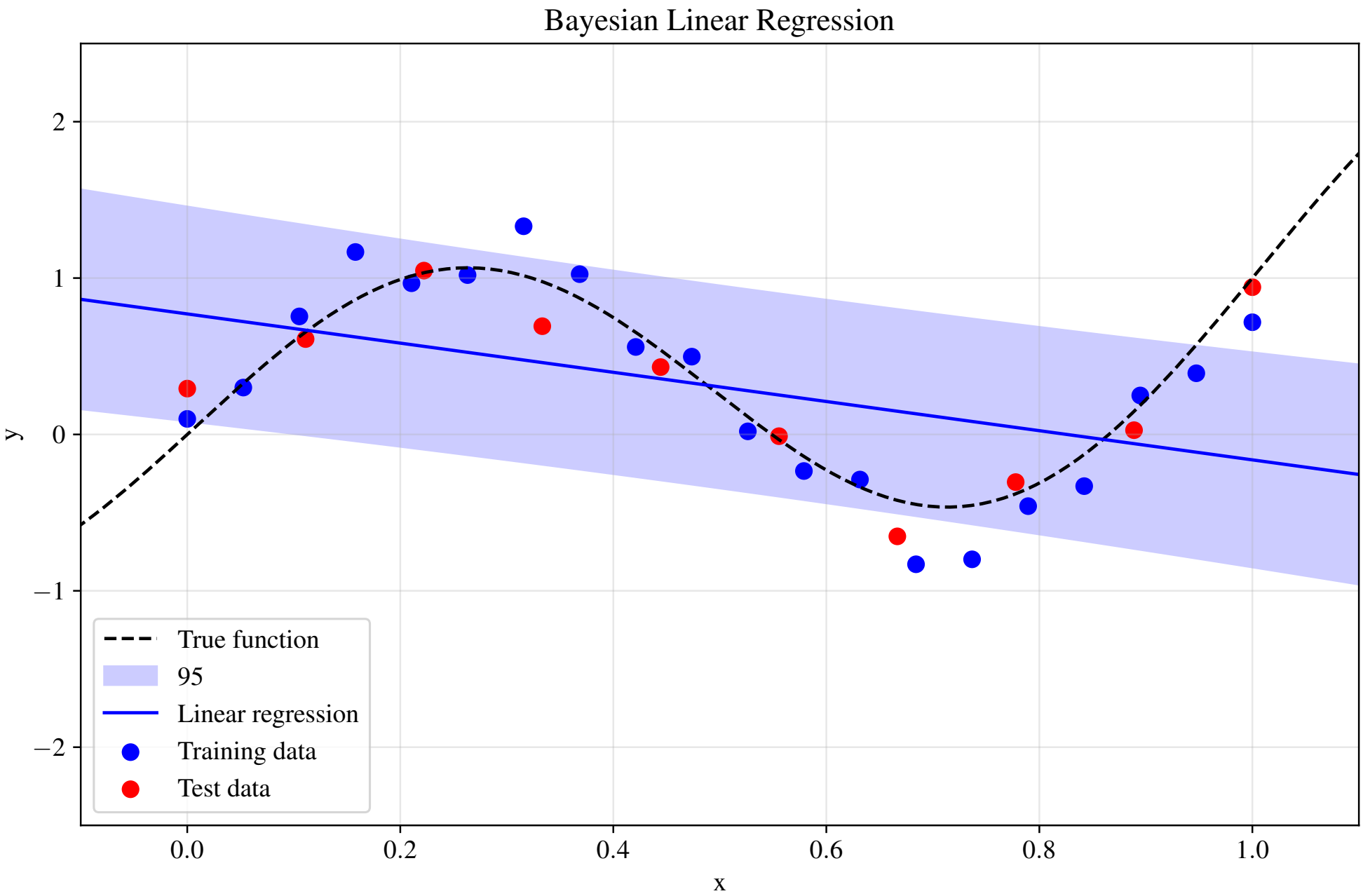
Bayesian linear regression

Polynomial feature map



Bayesian linear regression

Polynomial feature map



Bayesian linear regression

Feature maps for increased flexibility

To increase flexibility, we consider a *mapping* $\boldsymbol{\phi}(\cdot) : \mathcal{X} \to \mathcal{V}$ from our input domain $\mathcal{X}$ (here $\mathbb{R}^d$) to some new space $\mathcal{V}$, which may or may not be finite.

Rather than considering $f(\boldsymbol{x}; \boldsymbol{\beta}) = \boldsymbol{x}^T\boldsymbol{\beta}$, we instead consider $f(\boldsymbol{x}; \boldsymbol{\beta}) = \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\beta}$:

1. $\boldsymbol{\phi}(\boldsymbol{x}_i) = \boldsymbol{x}_i$,
2. $\boldsymbol{\phi}(\boldsymbol{x}_i) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots, \sqrt{2}x_{d-1}x_d)^T$,
3. $\boldsymbol{\phi}(\boldsymbol{x}_i) = (\phi_1(\boldsymbol{x}_i), \ldots, \phi_p(\boldsymbol{x}_i))^T$ with $\phi_j(\boldsymbol{x}_i) = \exp\left\{-\frac{\|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2}{2\sigma^2}\right\}$.

Bayesian linear regression

Feature maps for increased flexibility

Importantly, these functions are *fixed* and *independent of* $\boldsymbol{\beta}$, so we can treat is as before:

$$y_i = f(\boldsymbol{x}_i; \boldsymbol{\beta}) + \epsilon, \qquad f(\boldsymbol{x}_i) = \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\beta}, \qquad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2).$$

Everything is the same, we just replace X with $\Phi(X)$:

$$\boldsymbol{\beta} \mid X, \boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_\beta, \Lambda_\beta^{-1}\right),$$

where

$$\begin{aligned}
\boldsymbol{\mu}_\beta &= \frac{1}{\sigma_\epsilon^2}\left(\frac{1}{\sigma_\epsilon^2}\Phi^T\Phi + \Sigma^{-1}\right)^{-1}\Phi^T\boldsymbol{y}, \\
\Lambda_\beta &= \frac{1}{\sigma_\epsilon^2}\Phi^T\Phi + \Sigma^{-1},
\end{aligned}$$

where we write Φ to mean $\Phi(X)$.

The predictive distribution becomes:

$$\boldsymbol{y}_* \mid \Phi_*, \Phi, \boldsymbol{y} \sim \mathcal{N}\left(\Phi_*\boldsymbol{\mu}_\beta, \Phi_*\Lambda_\beta^{-1}\Phi_*^T + \sigma_\epsilon^2 I_m\right),$$

Bayesian linear regression

Let's choose a feature map to develop a more flexible function.

We will work with the *squared exponential function*:

$$\phi_j(x) = \sigma^2 \exp\left\{-\frac{(x - c_j)^2}{2\ell^2}\right\},$$

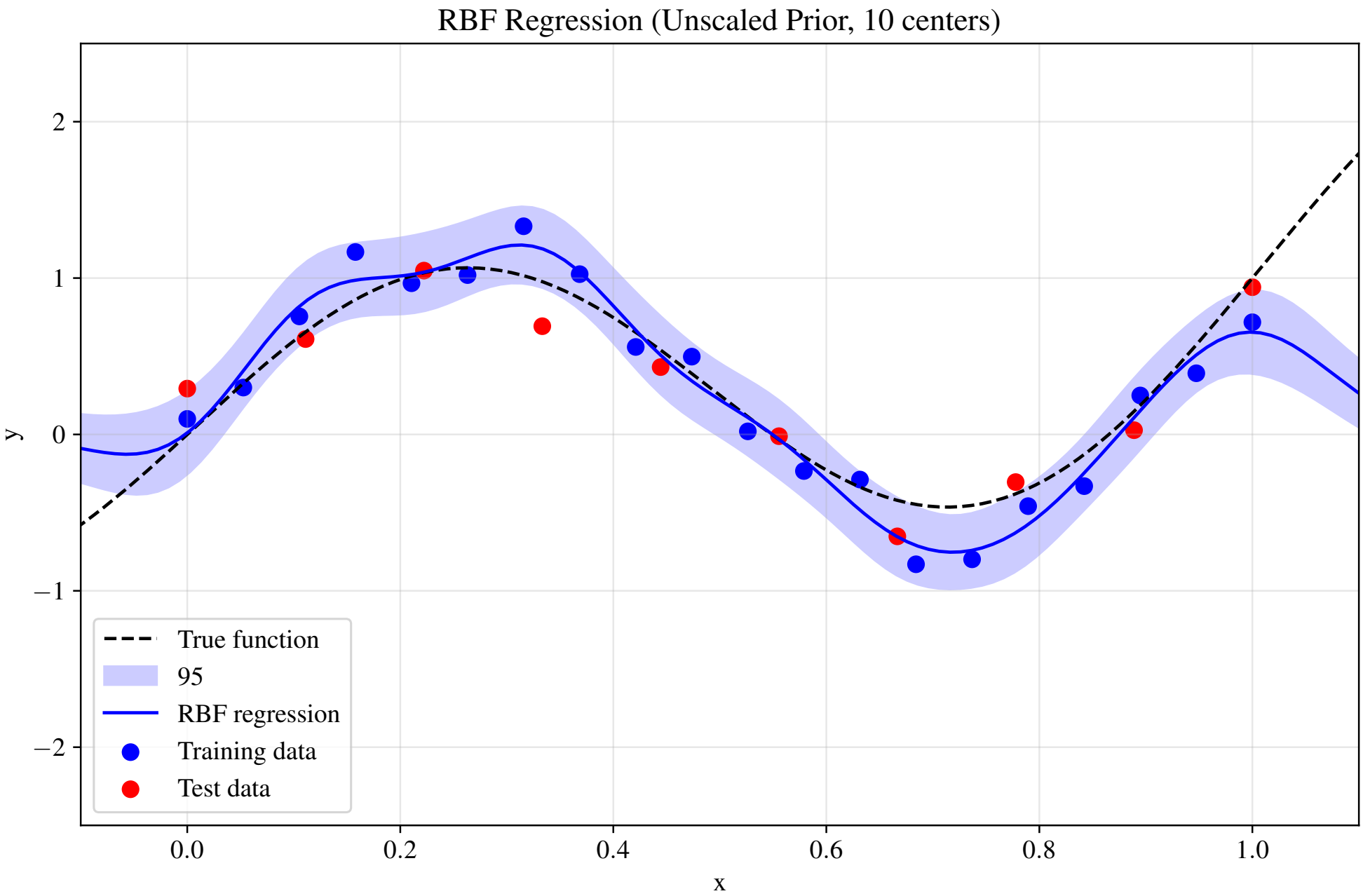
for some *centre* $c_j \in \mathbb{R}$.

We can use a number of these functions to form a flexible mapping:

$$\boldsymbol{\phi}(x) = (\phi_1(x), \ldots, \phi_p(x))^T.$$

Bayesian linear regression

Polynomial feature map



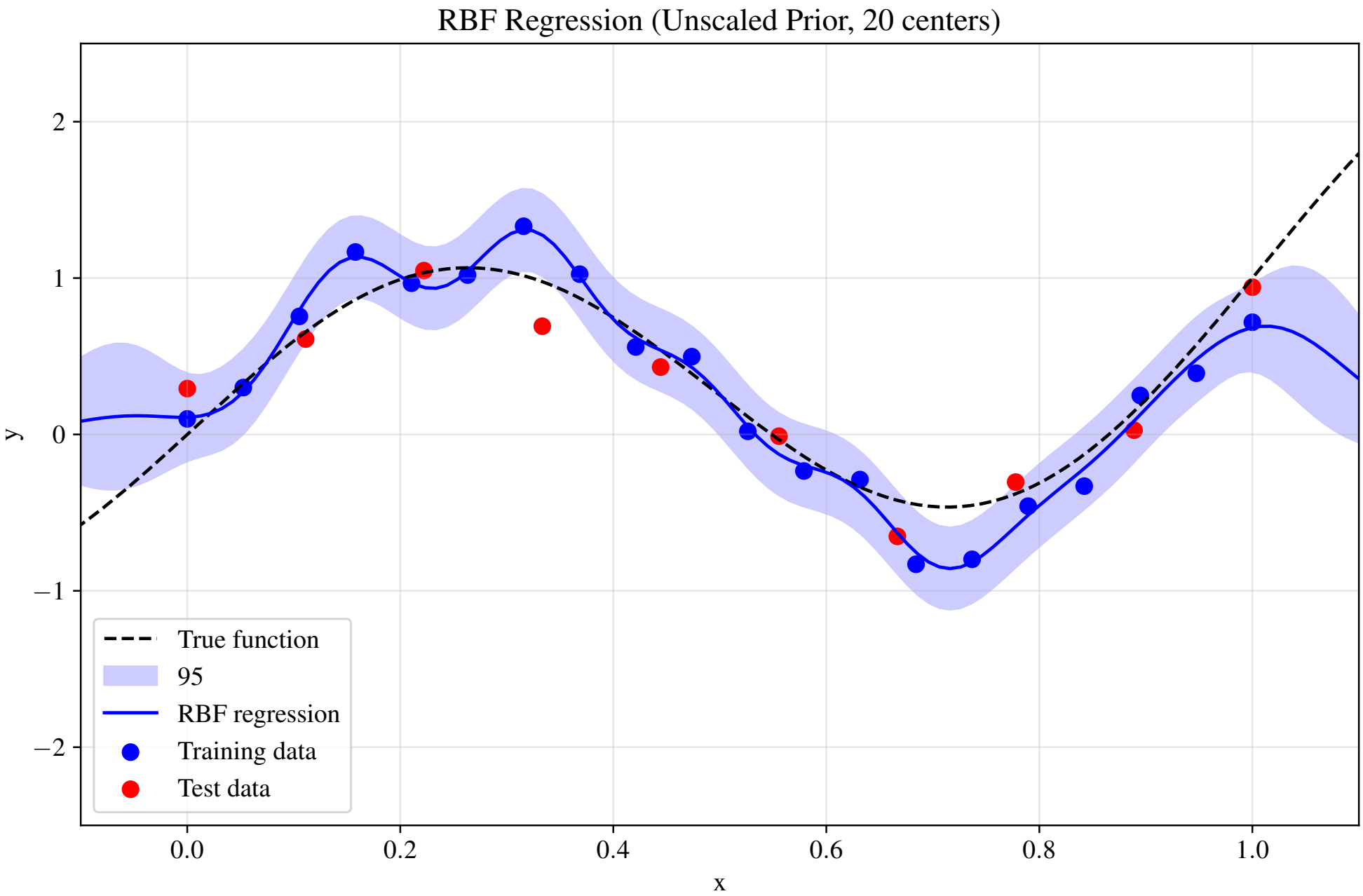
Bayesian linear regression

By transforming from input space, we are able to better capture the shape of the data.

Why don't we just let our feature map grow in size?

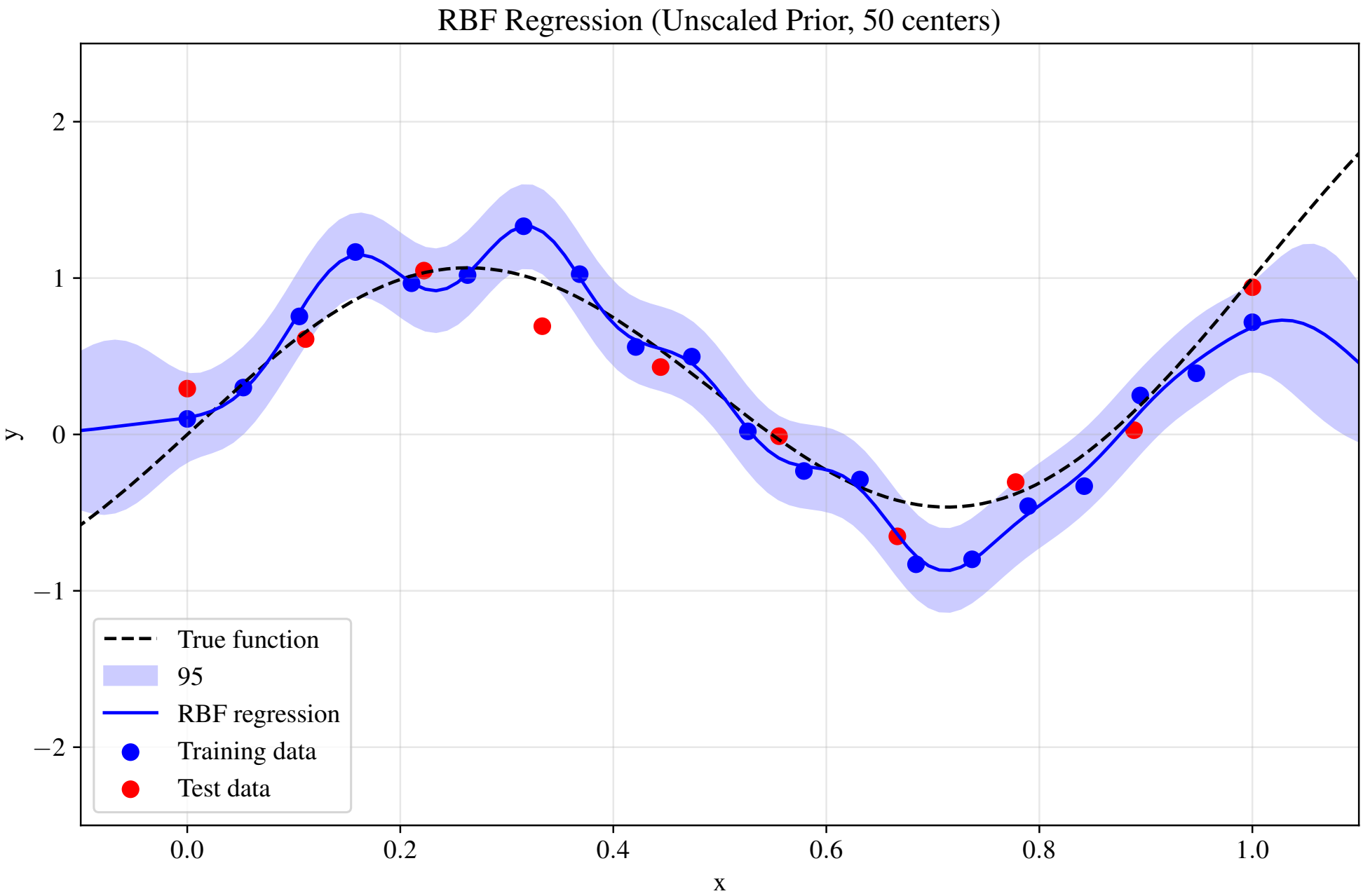
Bayesian linear regression

Polynomial feature map



Bayesian linear regression

Polynomial feature map

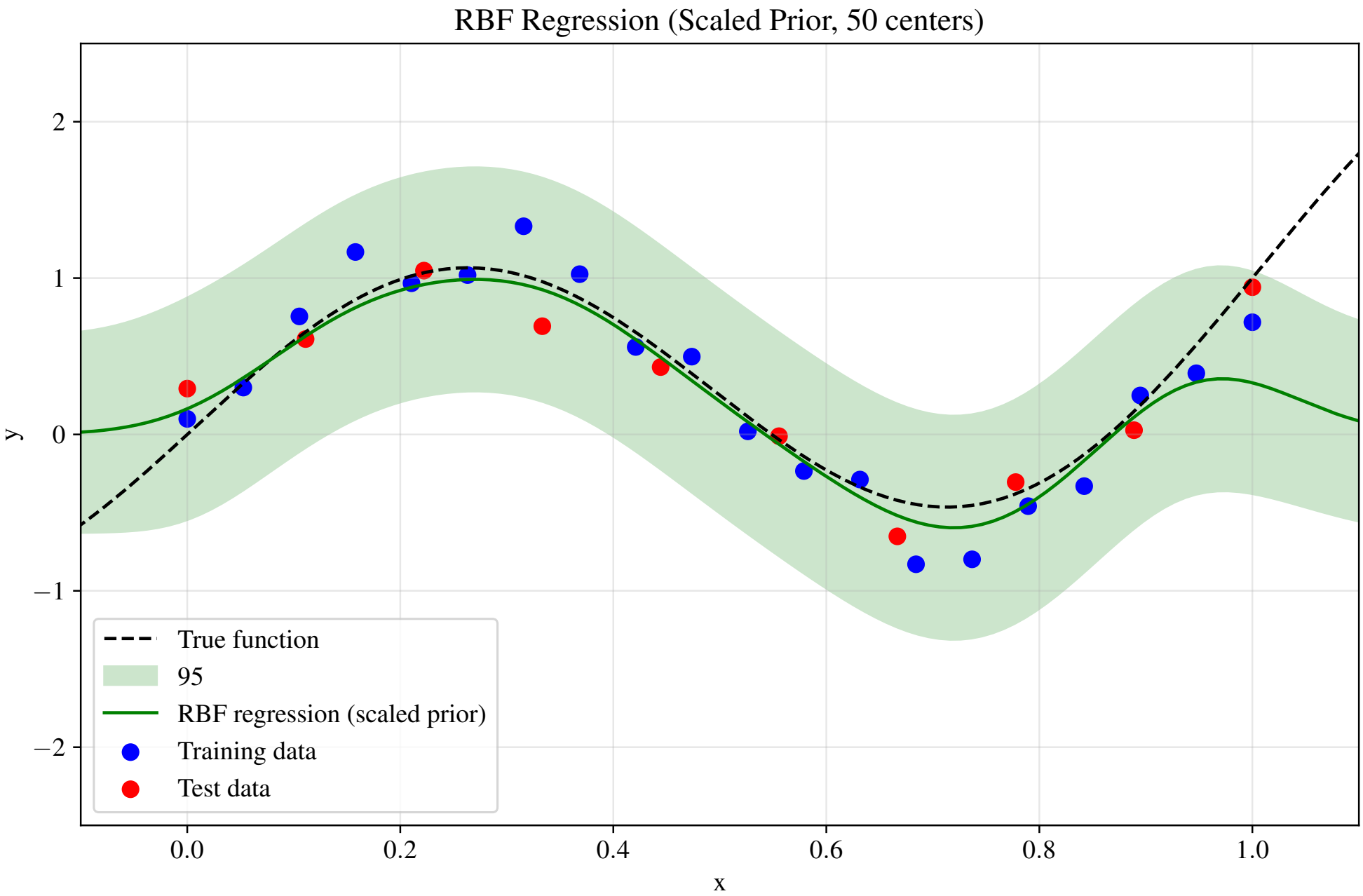


Bayesian linear regression

- The problem with letting p grow is that we *overfit* the data - we lose any useful predictive properties and the shape of the true generating function.
- However, Bayesian statistics comes to the rescue: the prior $\boldsymbol{\beta} \sim \mathcal{N}(0, \Sigma)$ controls the prior *smoothness* of $\boldsymbol{\beta}$.
- If we increase how "smooth forcing" our prior is (i.e. decrease the magnitude of the entries of Σ) we will force smoothness in $\boldsymbol{\beta}$ (less overfitting).
- We can do this by writing $\boldsymbol{\beta} \sim \mathcal{N}(0, \Sigma/p)$ where p is the length of $\boldsymbol{\beta}$.

Bayesian linear regression

Polynomial feature map



Bayesian linear regression

We have a much smoother function now! Adding more basis functions improves our flexibility *if* we regularise.

Questions:

1. How do we know where to centre our basis functions?
2. Can we let p grow indefinitely? Does this cause any problems?

The first question is answered by placing basis functions everywhere! But what issue does this cause?

Bayesian linear regression

Problems with lots of basis functions

We would like to have many (many many) more basis functions to allow us to extrapolate outside of the training range!

This task is not computationally feasible at the moment:

$$\boldsymbol{\mu}_\beta = \frac{1}{\sigma_\epsilon^2}\Lambda_\beta^{-1}\Phi^T\boldsymbol{y},$$

$$\Lambda_\beta^{-1} = \left(\frac{1}{\sigma_\epsilon^2}\underbrace{\Phi^T\Phi}_{p\times p} + \Sigma^{-1}\right)^{-1},$$

where $\Phi^T\Phi \in \mathbb{R}^{p\times p}$. This will be too large if we want lots of basis functions!

Bayesian linear regression

From $p \times p$ to $n \times n$

A crucial observation is that we can rewrite the mean and covariance from before:

$$\boldsymbol{\mu}_\beta = \frac{1}{\sigma_\epsilon^2}\left(\frac{1}{\sigma_\epsilon^2}\Phi^T\Phi + \Sigma^{-1}\right)^{-1}\Phi^T\boldsymbol{y} \quad \rightarrow \quad \boldsymbol{\mu}_\beta = \Sigma\Phi^T\left(\Phi\Sigma\Phi^T + \sigma_\epsilon^2 I_n\right)^{-1}\boldsymbol{y},$$

$$\Lambda_\beta^{-1} = \left(\frac{1}{\sigma_\epsilon^2}\Phi^T\Phi + \Sigma^{-1}\right)^{-1} \quad \rightarrow \quad \Lambda_\beta^{-1} = \Sigma - \Sigma\Phi^T(\Phi\Sigma\Phi^T + \sigma_\epsilon^2 I_n)^{-1}\Phi\Sigma.$$

Why is this better?

Bayesian linear regression

From $p \times p$ to $n \times n$

A crucial observation is that we can rewrite the mean and covariance from before:

$$\boldsymbol{\mu}_\beta = \frac{1}{\sigma_\epsilon^2}\left(\frac{1}{\sigma_\epsilon^2}\Phi^T\Phi + \Sigma^{-1}\right)^{-1}\Phi^T\boldsymbol{y} \quad \rightarrow \quad \boldsymbol{\mu}_\beta = \Sigma\Phi^T\left(\Phi\Sigma\Phi^T + \sigma_\epsilon^2 I_n\right)^{-1}\boldsymbol{y},$$

$$\Lambda_\beta^{-1} = \left(\frac{1}{\sigma_\epsilon^2}\Phi^T\Phi + \Sigma^{-1}\right)^{-1} \quad \rightarrow \quad \Lambda_\beta^{-1} = \Sigma - \Sigma\Phi^T(\Phi\Sigma\Phi^T + \sigma_\epsilon^2 I_n)^{-1}\Phi\Sigma.$$

Why is this better?

$$\left(\frac{1}{\sigma_\epsilon^2}\underbrace{\Phi^T\Phi}_{p\times p} + \Sigma^{-1}\right)^{-1} \quad \rightarrow \quad \left(\underbrace{\Phi\Sigma\Phi^T}_{n\times n} + \sigma_\epsilon^2 I_n\right)^{-1},$$

With this trick, everything can now be written very compactly...

Bayesian linear regression

Predictive distribution with feature maps

Recall the form of our predictive distribution:

$$\boldsymbol{y}_* \mid \Phi_*, \Phi, \boldsymbol{y} \sim \mathcal{N}\left(\Phi_* \boldsymbol{\mu}_\beta, \Phi_* \Lambda_\beta^{-1} \Phi_*^T + \sigma_\epsilon^2 I_m\right).$$

With our new expressions for $\boldsymbol{\mu}_\beta$ and Λ_β^{-1}, these become:

$$\begin{aligned}
\Phi_* \boldsymbol{\mu}_\beta &= \Phi_* \Sigma \Phi^T (\Phi \Sigma \Phi^T + \sigma_\epsilon^2 I_n)^{-1} \boldsymbol{y} \\
\Phi_* \Lambda_\beta^{-1} \Phi_*^T &= \Phi_* \Sigma \Phi_*^T - \Phi_* \Sigma \Phi^T \left(\Phi \Sigma \Phi^T + \sigma_\epsilon^2 I\right)^{-1} \Phi \Sigma \Phi_*.
\end{aligned}$$

Notice how everything is a function of one of the following:

$$\Phi \Sigma \Phi^T, \qquad \Phi_* \Sigma \Phi^T, \qquad \Phi \Sigma \Phi_*^T, \qquad \Phi_* \Sigma \Phi_*^T.$$

Each entry of these matrices will be of the form $\boldsymbol{\phi}(\boldsymbol{x})^T \Sigma \boldsymbol{\phi}(\boldsymbol{x}')$.

Bayesian linear regression

Predictive distribution with feature maps

Let's define a function:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^T \Sigma \boldsymbol{\phi}(\boldsymbol{x}') = \boldsymbol{\psi}(\boldsymbol{x})^T \boldsymbol{\psi}(\boldsymbol{x}'),$$

where we write $\boldsymbol{\psi}(\boldsymbol{x}) = \Sigma^{1/2}\boldsymbol{\phi}(x)$, which is possible because Σ is positive definite.

With this function defined, the expression is simpler:

$$\begin{aligned} \Phi_* \Sigma \Phi^T (\Phi \Sigma \Phi^T + \sigma_\epsilon^2 I_n)^{-1} \boldsymbol{y} \quad &\rightarrow \quad K_{*\cdot}(K_{\cdot\cdot} + \sigma_\epsilon^2 I_n)^{-1} \boldsymbol{y} \\ \Phi_* \Sigma \Phi_*^T - \Phi_* \Sigma \Phi^T \left(\Phi \Sigma \Phi^T + \sigma_\epsilon^2 I\right)^{-1} \Phi \Sigma \Phi_* \quad &\rightarrow \quad K_{**} - K_{*\cdot}(K_{\cdot\cdot} + \sigma_\epsilon^2 I_n)^{-1} K_{\cdot *}. \end{aligned}$$

Here we are writing:

$$K_{\cdot\cdot} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}, \qquad K_{\cdot *} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_{*,1}) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_{*,m}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_{*,1}) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_{*,m}) \end{pmatrix}$$

Bayesian linear regression

OK, what's the point?

We've introduced some succinct notation... so what? How does this help?

Mercer's theorem

James Mercer showed that if $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a *continuous, symmetric and positive semi-definite* function, then there exists a *basis function* or *feature map* $\boldsymbol{\psi} : \mathcal{X} \to \mathcal{V}$ such that

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\psi}(\boldsymbol{x})^T \boldsymbol{\psi}(\boldsymbol{x}'),$$

where $\langle \cdot, \cdot \rangle$ is an inner product on $\mathcal{V}$.

In fact, this statement is an *if and only if*:

any such function defines a feature map, and any choice of feature map defines a function!

Bayesian linear regression

Why is this useful?

For example, consider the feature map:

$$\boldsymbol{\psi}(\boldsymbol{x}) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots, \sqrt{2}x_{d-1}x_d)^T.$$

This has dimension $\mathcal{O}(d^2)$, but notice that

$$\boldsymbol{\psi}(\boldsymbol{x})^T\boldsymbol{\psi}(\boldsymbol{x}') = 1 + \sum_{i=1}^{d} 2x_ix_i' + \sum_{i,j=1}^{d} x_ix_jx_i'x_j' = (1 + \boldsymbol{x}^T\boldsymbol{x}')^2.$$

Explicit computation (the middle part) requires $\mathcal{O}(d^2 + d)$ operations, but implicit computation (the last part) requires $\mathcal{O}(d)$.

If we define the *kernel* $K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^T\boldsymbol{x}')^2$, we obtain the same output but without having to compute the feature map!

Bayesian linear regression

Kernel functions

We are describing a function that is:

- **continuous**,
- **symmetric**: $k(x, y) = k(y, x)$,
- **positive semi-definite**: for any finite set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$ and any real numbers $\{c_1, \ldots, c_n\}$, we have

$$\sum_{i,j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.$$

We call such a function a **kernel** or **covariance** function.

Bayesian linear regression

This is important: let's be clear

- We want to have a set of flexible basis functions defined through a feature map $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{V}$ to model complex data.
- The problem is that having lots of basis functions isn't computationally feasible (we can't let $|\mathcal{V}| \to \infty$).
- BUT: we have found a "trick" to rewrite the posterior and predictive distributions so that we never need to actually evaluate the feature map directly.
- By defining a kernel function $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, we never need to work with $\mathcal{V}$ at all!
- By choosing a kernel k that matches the choice of $\boldsymbol{\phi}$, we simply compute $K(X, X) = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j=1}^n$ and we are able to evaluate the posterior and predictive for $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\beta} + \epsilon$, with $\boldsymbol{\beta} \sim \mathcal{N}(0, \Sigma)$, without ever having to explicitly evaluate $\{\boldsymbol{\phi}(\boldsymbol{x}_i)\}_{i=1}^n$!

Bayesian linear regression

Working with infinitely many basis functions

Let's set $\mathcal{X} = \mathbb{R}$ and re-consider our squared exponential functions:

$$\phi_m(x) = \exp\left(-\frac{(x - c_m)^2}{2\ell^2}\right),$$

where c_m is the *centre* of this basis function. Define the feature map $\boldsymbol{\psi}(x) = (\phi_1(x), \ldots, \phi_p(x))^T$. For the prior $\boldsymbol{\beta} \sim \mathcal{N}(0, \sigma^2 I_p/p)$, we now know this gives rise to a kernel function of the form:

$$K(x, x') = \frac{\sigma^2}{p} \sum_{m=1}^{p} \phi_m(x)\phi_m(x')$$

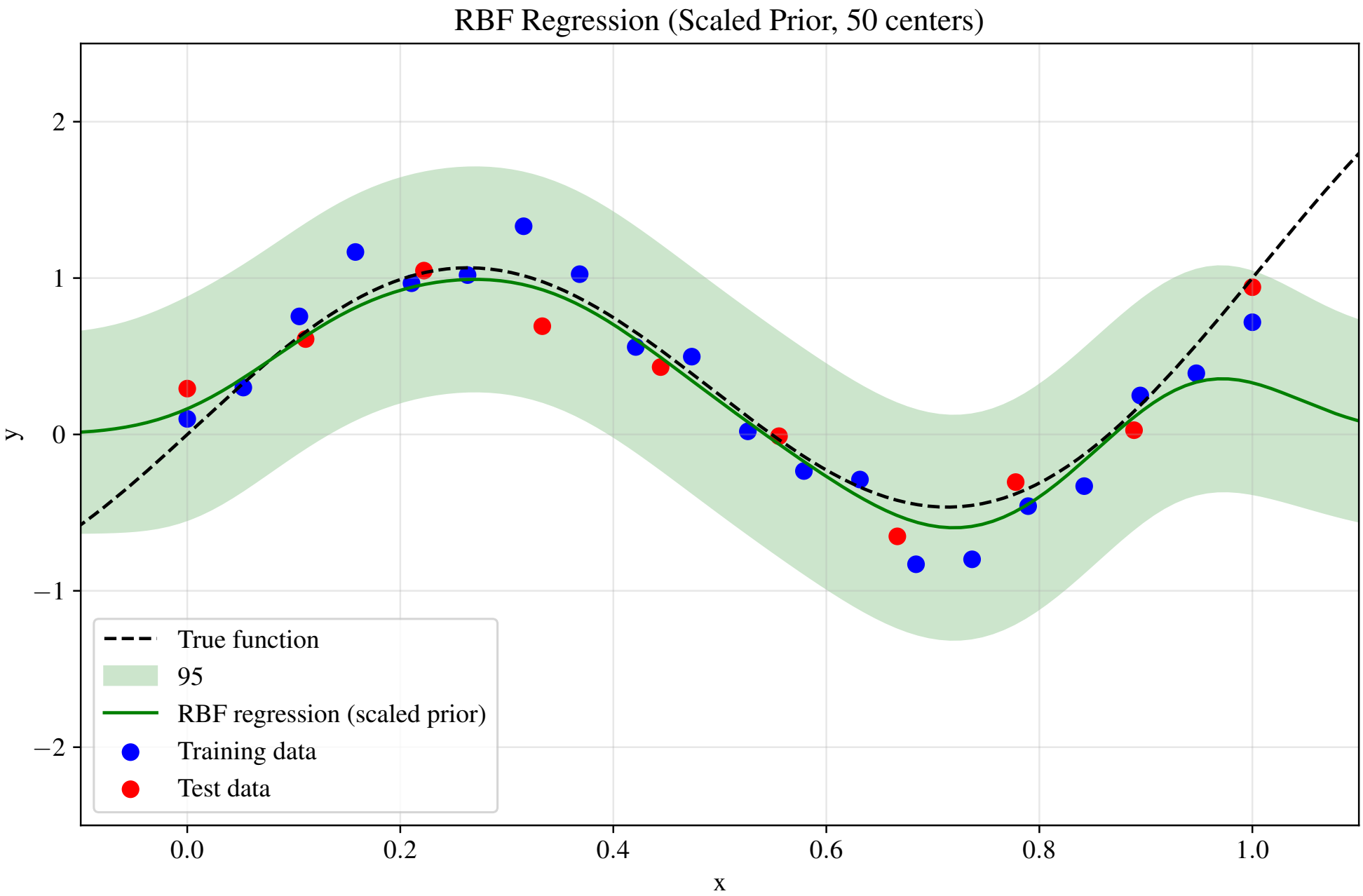
Now let's consider what happens when we put a basis function *everywhere*:

$$\begin{aligned}\lim_{p\to\infty} \frac{\sigma^2}{p} \sum_{m=1}^{p} \phi_m(x)\phi_m(x') &= \int_{-\infty}^{\infty} \exp\left(-\frac{(x - c)^2}{2\ell^2}\right) \exp\left(-\frac{(x - c)^2}{2\ell^2}\right) \mathrm{d}c \\ &= \sigma^2\sqrt{\pi}\ell \exp\left(-\frac{(x - x')^2}{4\ell^2}\right)\end{aligned}$$

which is called the radial basis function (RBF) kernel and it is equivalent to placing RBFs everywhere.

Bayesian linear regression

Polynomial feature map



A change of perspectives

Switching to a function-space view

1. We consider raw data $X \in \mathcal{X}^n$ and $\boldsymbol{y} \in \mathcal{Y}^n$ and test data $X_* \in \mathcal{X}^m$. We want to predict $\boldsymbol{y}_* \in \mathcal{Y}^m$.
2. We considered an explicit parameterisation of a function $f(\cdot; \boldsymbol{\beta}) : \mathcal{X} \to \mathcal{Y}$ as $f(\cdot; \boldsymbol{\beta}) = \boldsymbol{\phi}(\cdot)^T \boldsymbol{\beta}$ - explicitly depending on a parameter $\boldsymbol{\beta}$ and some function $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{V}$.
3. We placed a *Gaussian prior* on $\boldsymbol{\beta}$ and found that observations at a new set of inputs $X_* \in \mathcal{X}^m$ followed a normal distribution:

$$\boldsymbol{y}_* \mid \Phi_*, \Phi, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{y}_*}, K_{\boldsymbol{y}_*} + \sigma_\epsilon^2 I_m).$$

4. We discovered that we could abstract away from $\boldsymbol{\phi}$, so the natural question is:

Can we abstract away from $\boldsymbol{\beta}$?

Moving away from $\boldsymbol{\beta}$

Linear combinations of Gaussians are Gaussian!

Having $\boldsymbol{\beta} \sim \mathcal{N}(0, \Sigma)$ gives that $f(\cdot; \boldsymbol{\beta}) = \boldsymbol{\phi}(\cdot)^T \boldsymbol{\beta}$ is univariate normal. We can compute the expectation

$$\mathbb{E}\{f\} = \mathbb{E}\{\boldsymbol{\phi}^T \boldsymbol{\beta}\} = \boldsymbol{\phi}^T \mathbb{E}\{\boldsymbol{\beta}\} = 0,$$

the variance

$$\mathbb{V}\{f\} = \boldsymbol{\phi}^T \mathbb{V}(\boldsymbol{\beta}) \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T (\boldsymbol{x}) \Sigma \boldsymbol{\phi}(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}),$$

and the covariance

$$\text{Cov}\{f(\boldsymbol{x}), f(\boldsymbol{x}')\} = \boldsymbol{\phi}(\boldsymbol{x})^T \mathbb{V}(\boldsymbol{\beta}) \boldsymbol{\phi}(\boldsymbol{x}')^T = \boldsymbol{\phi}(\boldsymbol{x})^T \Sigma \boldsymbol{\phi}(\boldsymbol{x}')^T = k(\boldsymbol{x}, \boldsymbol{x}')$$

So, function evaluations are **jointly Gaussian** with a covariance defined by a **kernel function**.

Rather than placing a prior on $\boldsymbol{\beta}$, let's place a prior directly on f!

Defining and working with GPs

An outline of what we will cover

- We will quickly recap of our motivation and the switch from weight-space to function space.
- Then we will formally define a Gaussian process and demonstrating its predictive properties.
- Next we discuss the theoretical and practical considerations around fitting a GP.

A change of perspectives

From weight to function space

Previously, we considered an explicit parameterisation of a function $f(\cdot;\boldsymbol{\beta}) : \mathcal{X} \to \mathcal{Y}$ as $f(\cdot;\boldsymbol{\beta}) = \boldsymbol{\phi}(\cdot)^T\boldsymbol{\beta}$.

We observed noisy realisations $y = f(\boldsymbol{x};\boldsymbol{\beta}) + \epsilon$ of our function, and by placing a *prior* on the explicitly stated parameters $\boldsymbol{\beta}$, we were able to derive predictive properties.

We found that function values at a new set of inputs $X_* \in \mathcal{X}^m$ followed a normal distribution:

$$\boldsymbol{f}_* \mid \Phi_*, \Phi, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{y}_*}, K_{\boldsymbol{y}_*}),$$

or, equivalently, that the noisy observations followed a normal distribution:

$$\boldsymbol{y}_* \mid \Phi_*, \Phi, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{y}_*}, K_{\boldsymbol{y}_*} + \sigma_\epsilon^2 I_m).$$

We discovered we could abstract away from $\boldsymbol{\phi}$, so the natural question is:

can we abstract away from $\boldsymbol{\beta}$?

Moving away from $\boldsymbol{\beta}$

Linear combinations of Gaussians are Gaussian!

Having $\boldsymbol{\beta} \sim \mathcal{N}(0, \Sigma)$ gives that $f(\cdot; \boldsymbol{\beta}) = \boldsymbol{\phi}(\cdot)^T \boldsymbol{\beta}$ is univariate normal. We can compute the expectation

$$\mathbb{E}\{f\} = \mathbb{E}\{\boldsymbol{\phi}^T \boldsymbol{\beta}\} = \boldsymbol{\phi}^T \mathbb{E}\{\boldsymbol{\beta}\} = 0,$$

the variance

$$\mathbb{V}\{f\} = \boldsymbol{\phi}^T \mathbb{V}(\boldsymbol{\beta}) \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T (\boldsymbol{x}) \Sigma \boldsymbol{\phi}(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}),$$

and the covariance

$$\text{Cov}\{f(\boldsymbol{x}), f(\boldsymbol{x}')\} = \boldsymbol{\phi}(\boldsymbol{x})^T \mathbb{V}(\boldsymbol{\beta}) \boldsymbol{\phi}(\boldsymbol{x}')^T = \boldsymbol{\phi}(\boldsymbol{x})^T \Sigma \boldsymbol{\phi}(\boldsymbol{x}')^T = k(\boldsymbol{x}, \boldsymbol{x}')$$

So, function evaluations are **jointly Gaussian** with a covariance defined by a **kernel function**.

Rather than placing a prior on $\boldsymbol{\beta}$, let's place a prior directly on f!

Defining of a Gaussian process

Gaussian Process

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Defining of a Gaussian process

Gaussian Process

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

In our case, the random variables will represent the value of our function $f(\boldsymbol{x})$ at location $\boldsymbol{x}$.

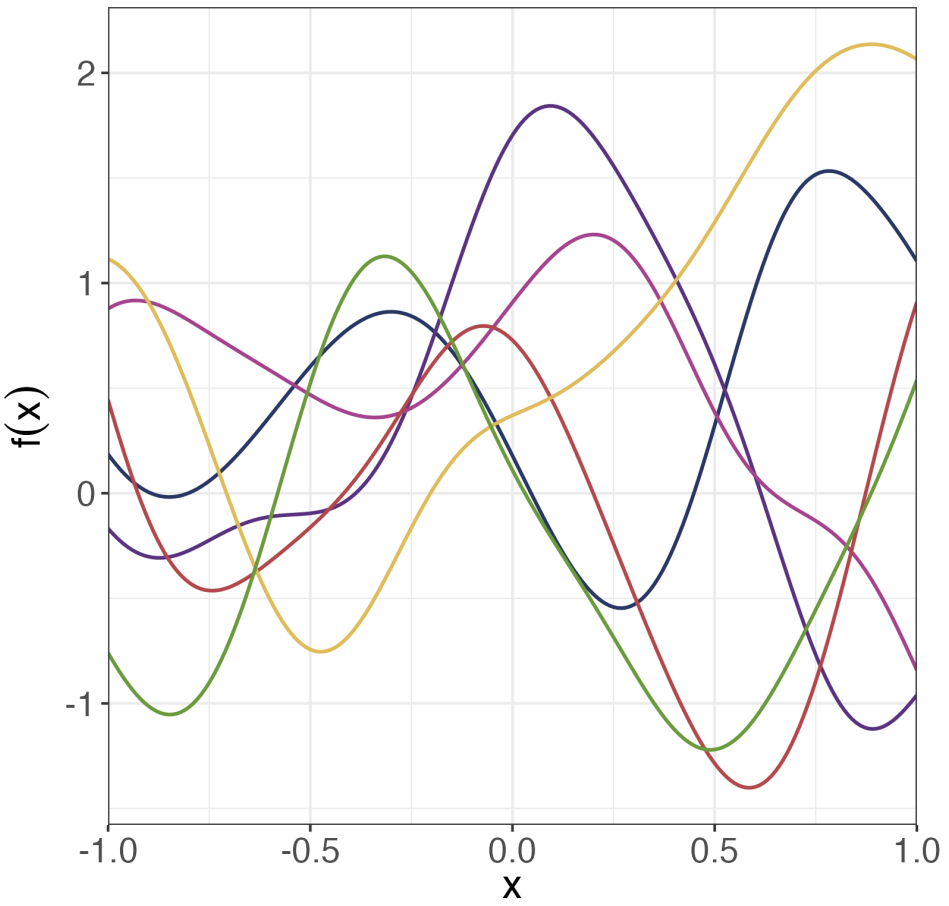
A Gaussian is defined by its first and second moments, so we specify some mean function $m(\cdot) : \mathcal{X} \to \mathbb{R}$ and a kernel function $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. We then write:

$$f \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)).$$

When we don't have any prior knowledge about the mean function set $m \equiv 0$, giving us

$$f \sim \mathcal{GP}(0, k(\cdot, \cdot)).$$

Samples from a GP prior



Defining a Gaussian Process

A simple example

We've already seen a Gaussian process! Look back at our explicitly defined function:

$$f(\boldsymbol{x}; \boldsymbol{\beta}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\beta}, \qquad \boldsymbol{\beta} \sim \mathcal{N}(0, \Sigma).$$

We know for any two inputs $\boldsymbol{x}, \boldsymbol{x}'$ that $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ are jointly Gaussian with zero mean and covariance $\boldsymbol{\phi}(\boldsymbol{x})\Sigma\boldsymbol{\phi}(\boldsymbol{x}')$.

The function values $f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)$ for any number of input points n are also jointly Gaussian - this is just a simple GP!

Defining the covariance of our process

By selecting the covariance function of a Gaussian process, we are *defining* the way that the function values of draws are related.

The covariance of a multivariate Gaussian defines how values at different input points are related.

Defining the covariance of our process

By selecting the covariance function of a Gaussian process, we are *defining* the way that the function values of draws are related.

The covariance of a multivariate Gaussian defines how values at different input points are related.

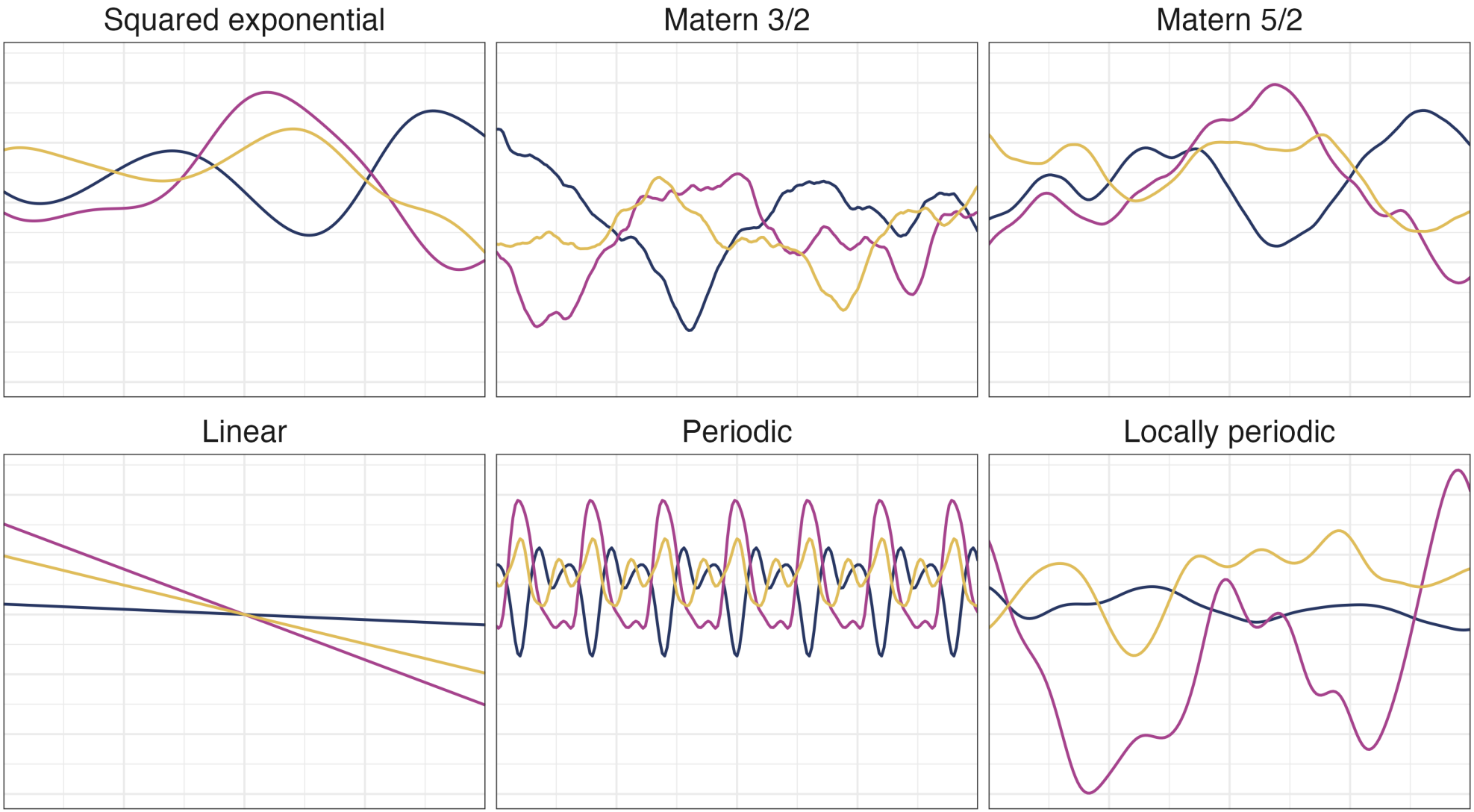
There are lots of options:

Name	Definition	Type of functions
Squared Exponential	$\alpha \exp\left(-\frac{r^2}{2\ell^2}\right)$	Infinitely differentiable functions
Matérn 3/2	$\alpha\left(1+\frac{\sqrt{3}r}{\ell}\right)\exp\left(-\frac{\sqrt{3}r}{\ell}\right)$	1 time differentiable functions
Matérn 5/2	$\alpha\left(1+\frac{\sqrt{5}r}{\ell}+\frac{5r^2}{3\ell^2}\right)\exp\left(-\frac{\sqrt{5}r}{\ell}\right)$	2 time differentiable functions
Linear Kernel	$\sigma_b^2+\sigma_v^2(x-c)(x'-c)$	Linear function
Periodic Kernel	$\alpha\exp\left(-\frac{2\sin^2(\pi r/p)}{\ell^2}\right)$	Periodic functions
Locally Periodic Kernel	$\alpha\exp\left(-\frac{2\sin^2(\pi r/p)}{\ell^2}\right)\exp\left(-\frac{r}{2\ell^2}\right)$	Functions that are periodic at certain locations

https://www.cs.toronto.edu/~duvenaud/cookbook/

Defining the covariance of our process

Examples of draws from the GP prior



Making predictions

This is all well and good, but...

We want to be able to *learn* from data and make *predictions*.

We observe data $X \in \mathcal{X}^n$ and $\boldsymbol{y} \in \mathcal{Y}^n$ and propose the model:

$$\begin{aligned} y_i \mid f, \boldsymbol{x}_i &\sim \mathcal{N}(f(\boldsymbol{x}_i), \sigma_\epsilon^2), \quad i = 1, \ldots, n, \\ f &\sim \mathcal{GP}(0, k_\theta(\cdot, \cdot)). \end{aligned}$$

(notice the difference between explicitly stating $\boldsymbol{\beta}$).

We see new data $X_* \in \mathcal{X}^m$. How does my knowledge of the $(\boldsymbol{x}_i, y_i)$ pairs influence what I know about the function values f_* at each $\boldsymbol{x}_{*i}$?

We know that $\boldsymbol{y}$ has a joint distribution $\boldsymbol{y} \sim \mathcal{N}(0, K(X, X))$ and we also know that any collection of function values must follow a multivariate Gaussian. It then follows:

$$\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} K(X, X) + \sigma_\epsilon^2 I_n & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right)$$

Making predictions

But how is this useful?

This is where the GP is really powerful! Because we are working with a Gaussian, we can simply condition on past data and still get a Gaussian (a good exercise):

$$\boldsymbol{f}_* \mid X, \boldsymbol{y}, X_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \Sigma_*),$$

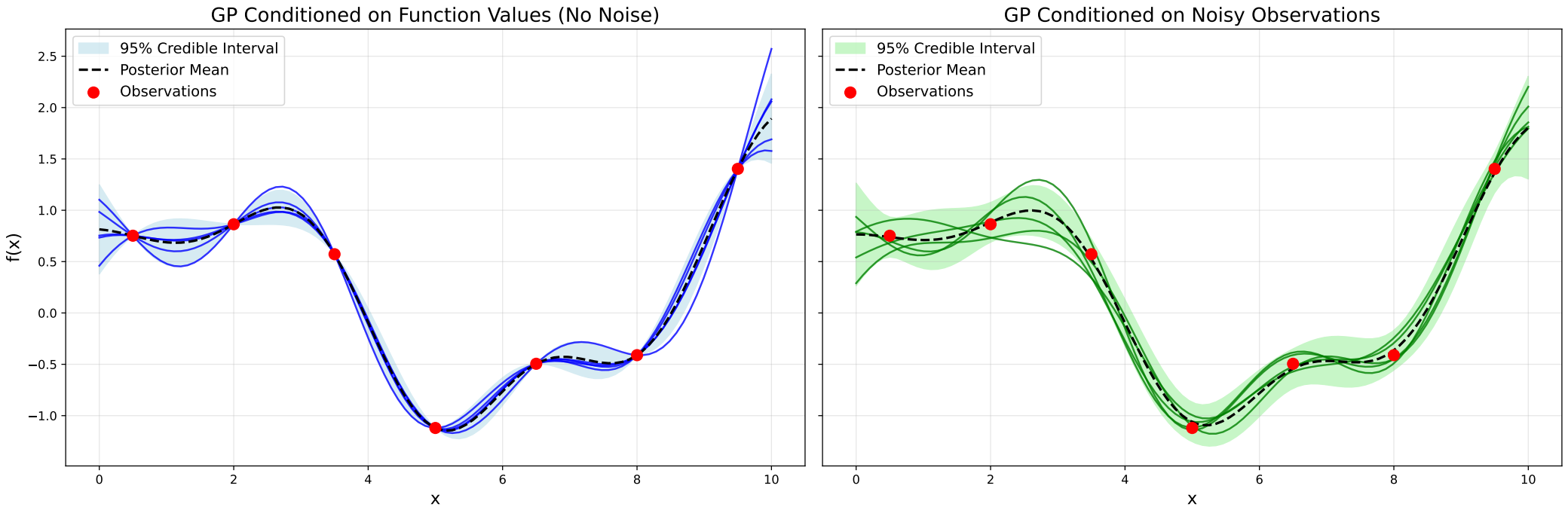
where

$$\begin{aligned}\boldsymbol{\mu}_* &= K(X_*, X)(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}\boldsymbol{y} \\ \Sigma_* &= K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}K(X, X_*).\end{aligned}$$

Note that we are predicting the *function values* $\boldsymbol{f}_*$ and not the observed values $\boldsymbol{y}_*$.

This looks familiar...

Draws from the conditioned distribution



Exercise: a minimal worked example

Let's work with data on $\mathbb{R}$ assuming a model $y_i = f(x_i) + \epsilon$ with training points $\{(x_i, y_i)\}_{i=1}^3$: (1,2), (3,1), (4,3) and look to make predictions for $x_* = 2$.

Let's set the noise variance to $\sigma_\epsilon^2 = 0.1$ and place a GP prior on f with the squared exponential kernel:

$$k(x, x') = \alpha \exp\left\{-\frac{\|x - x'\|_2}{2\ell^2}\right\},$$

with $\alpha = 1$, $\ell = 1$.

Compute the predictive mean and variance.

$$\boldsymbol{\mu}_* = K(X_*, X)(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}\boldsymbol{y}$$
$$\Sigma_* = K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}K(X, X_*).$$

Solution: a minimal worked example

The kernel matrix is:

$$K(x,x) = \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) \\ k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) \\ k(x_3,x_1) & k(x_3,x_2) & k(x_3,x_3) \end{pmatrix} = \begin{pmatrix} 1 & 0.135 & 0.011 \\ 0.135 & 1 & 0.607 \\ 0.011 & 0.607 & 1 \end{pmatrix}$$

Solution: a minimal worked example

The kernel matrix is:

$$K(x,x) = \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) \\ k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) \\ k(x_3,x_1) & k(x_3,x_2) & k(x_3,x_3) \end{pmatrix} = \begin{pmatrix} 1 & 0.135 & 0.011 \\ 0.135 & 1 & 0.607 \\ 0.011 & 0.607 & 1 \end{pmatrix}$$

The cross-covariance vector:

$$K(x_*,X) = \begin{pmatrix} k(x_*,x_1), & k(x_*,x_2), & k(x_*,x_3) \end{pmatrix} = \begin{pmatrix} 0.607, & 0.607, & 0.135 \end{pmatrix}$$

Solution: a minimal worked example

The kernel matrix is:

$$K(x,x) = \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) \\ k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) \\ k(x_3,x_1) & k(x_3,x_2) & k(x_3,x_3) \end{pmatrix} = \begin{pmatrix} 1 & 0.135 & 0.011 \\ 0.135 & 1 & 0.607 \\ 0.011 & 0.607 & 1 \end{pmatrix}$$

The cross-covariance vector:

$$K(x_*,X) = \begin{pmatrix} k(x_*,x_1), & k(x_*,x_2), & k(x_*,x_3) \end{pmatrix} = \begin{pmatrix} 0.607, & 0.607, & 0.135 \end{pmatrix}$$

The test variance is:

$$K(x_*,x_*) = 1$$

Solution: a minimal worked example

The kernel matrix is:

$$K(x,x) = \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) \\ k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) \\ k(x_3,x_1) & k(x_3,x_2) & k(x_3,x_3) \end{pmatrix} = \begin{pmatrix} 1 & 0.135 & 0.011 \\ 0.135 & 1 & 0.607 \\ 0.011 & 0.607 & 1 \end{pmatrix}$$

The cross-covariance vector:

$$K(x_*,X) = \begin{pmatrix} k(x_*,x_1), & k(x_*,x_2), & k(x_*,x_3) \end{pmatrix} = \begin{pmatrix} 0.607, & 0.607, & 0.135 \end{pmatrix}$$

The test variance is:

$$K(x_*,x_*) = 1$$

The posterior mean and variance are then:

$$\mu_* = K(x_*,X)[K(X,X) + \sigma_\epsilon^2 I]^{-1}\mathbf{y} \approx 0.907$$
$$\sigma_*^2 = K(x_*,x_*) - K(x_*,X)[K(X,X) + \sigma_\epsilon^2 I]^{-1}K(X,x_*) \approx 0.366$$

Where are we?

- We've changed our perspective from weight space to function space. We are now in a position to formally define a Gaussian process.
- We have formally defined a Gaussian process and demonstrated its predictive properties. We saw how this works in a simple example. We now need to decide how to approach modelling with GPs.

  Defining a GP involves selecting:

  1. The covariance kernel, depending on hyperparameters $\boldsymbol{\theta}$.
  2. The hyperparameters $\boldsymbol{\theta}$ of our chosen kernel.

  We will now develop the theoretical and practical approaches to making these choices.
- Next we discuss the theoretical and practical considerations around fitting a GP.

The kernel function

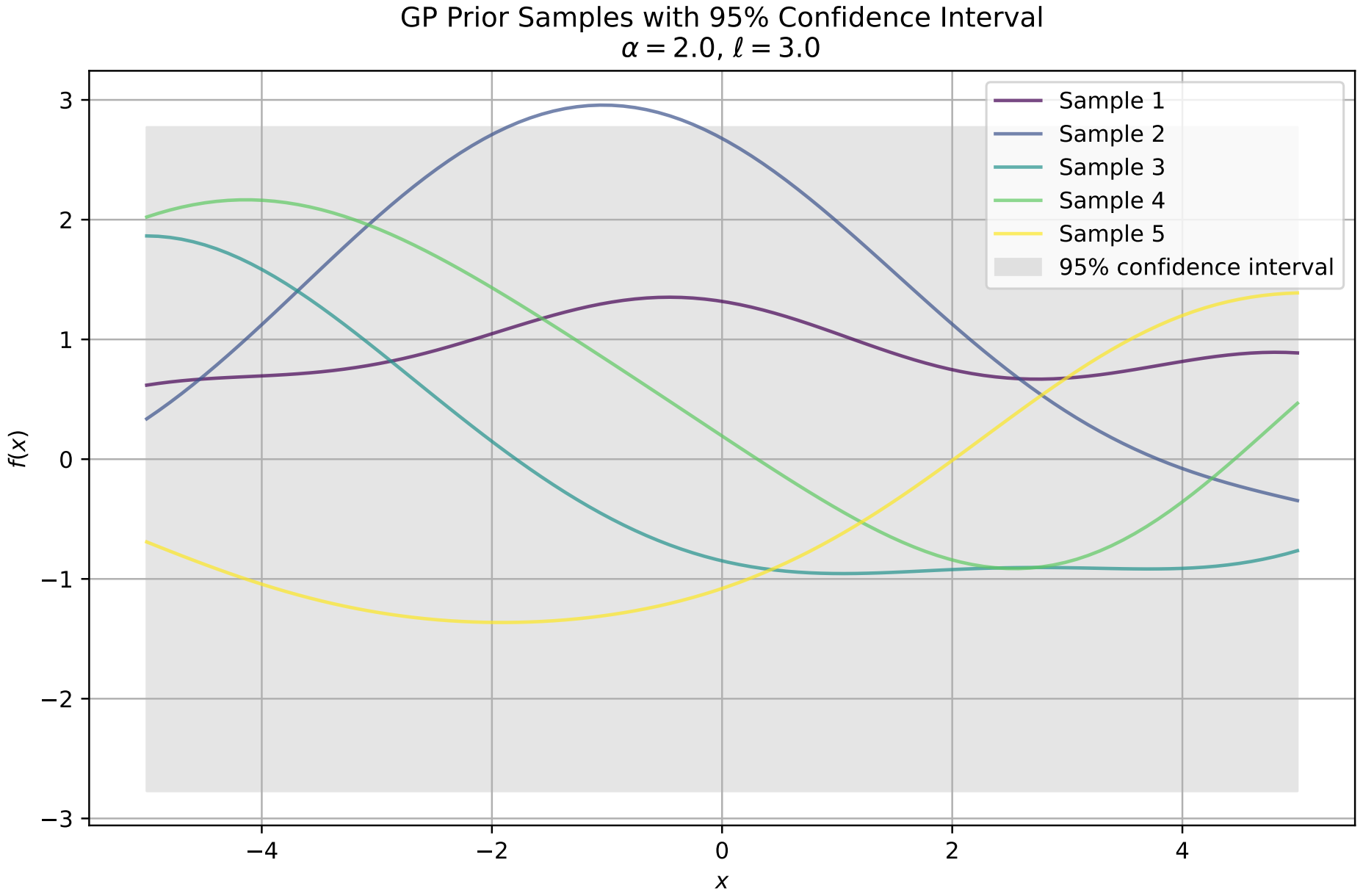
A GP prior depends on a kernel function which itself depends on some hyperparameter parameters. The choice of kernel and its subsequent hyperparameters really affect the behaviour of the GP.

As an example, we will work with the *squared exponential kernel* for the remainder of this lecture:

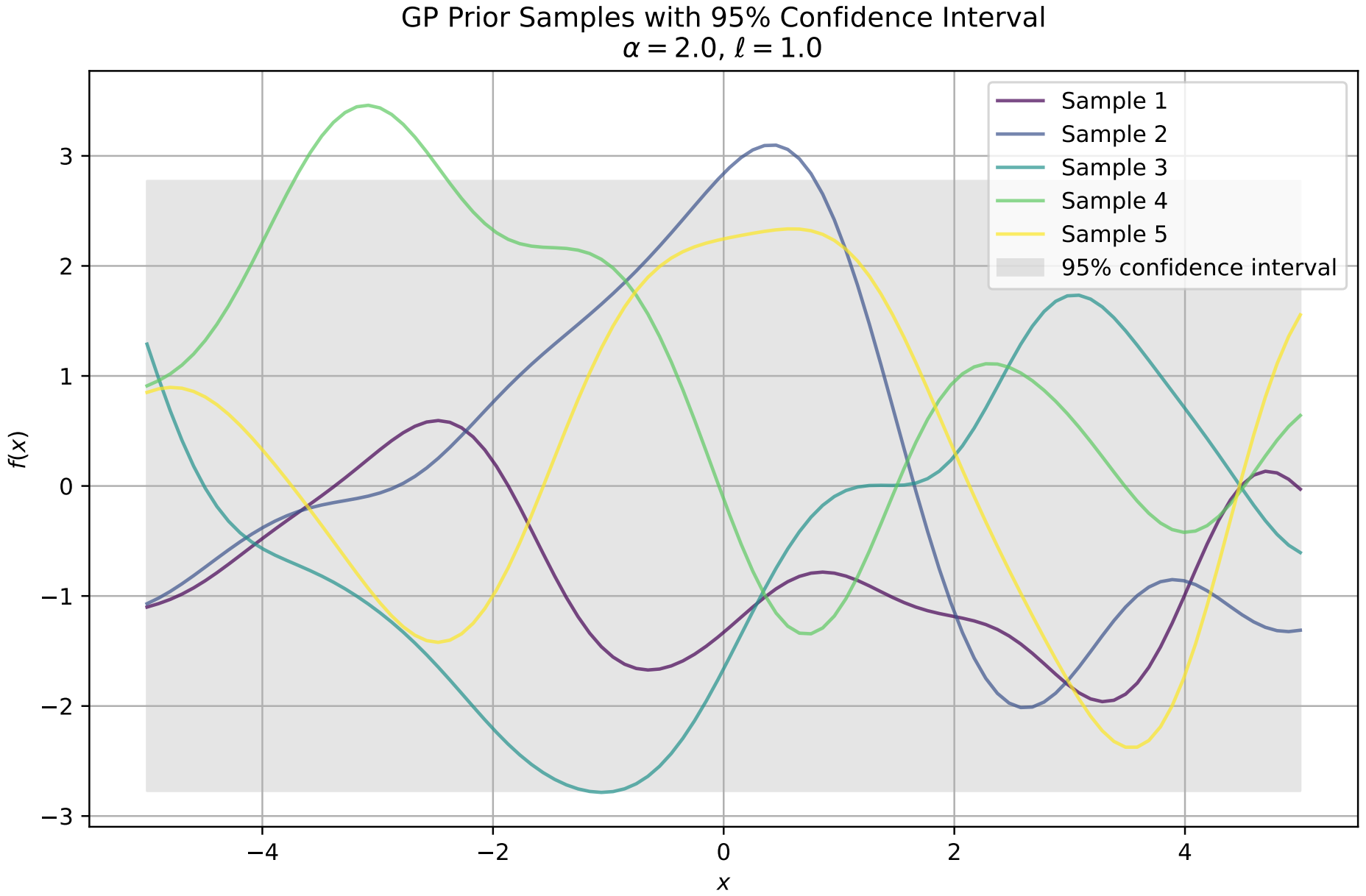
$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|}{2\ell^2}\right)$$

For this kernel, the hyperparameters are $\boldsymbol{\theta} = (\alpha, \ell)$. Varying these hyperparameters changes the output a lot!

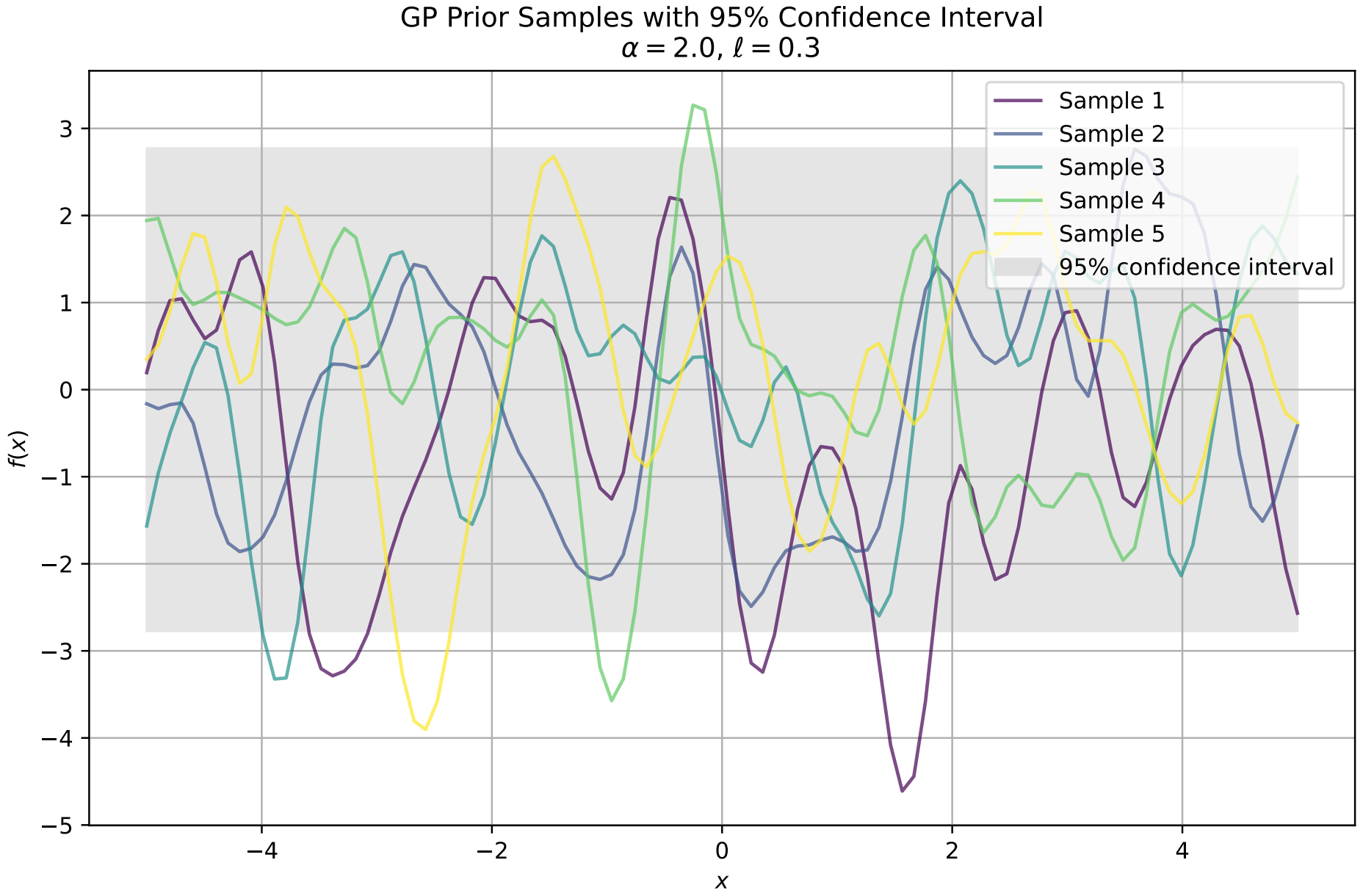
Prior and posterior samples for the squared exponential kernel



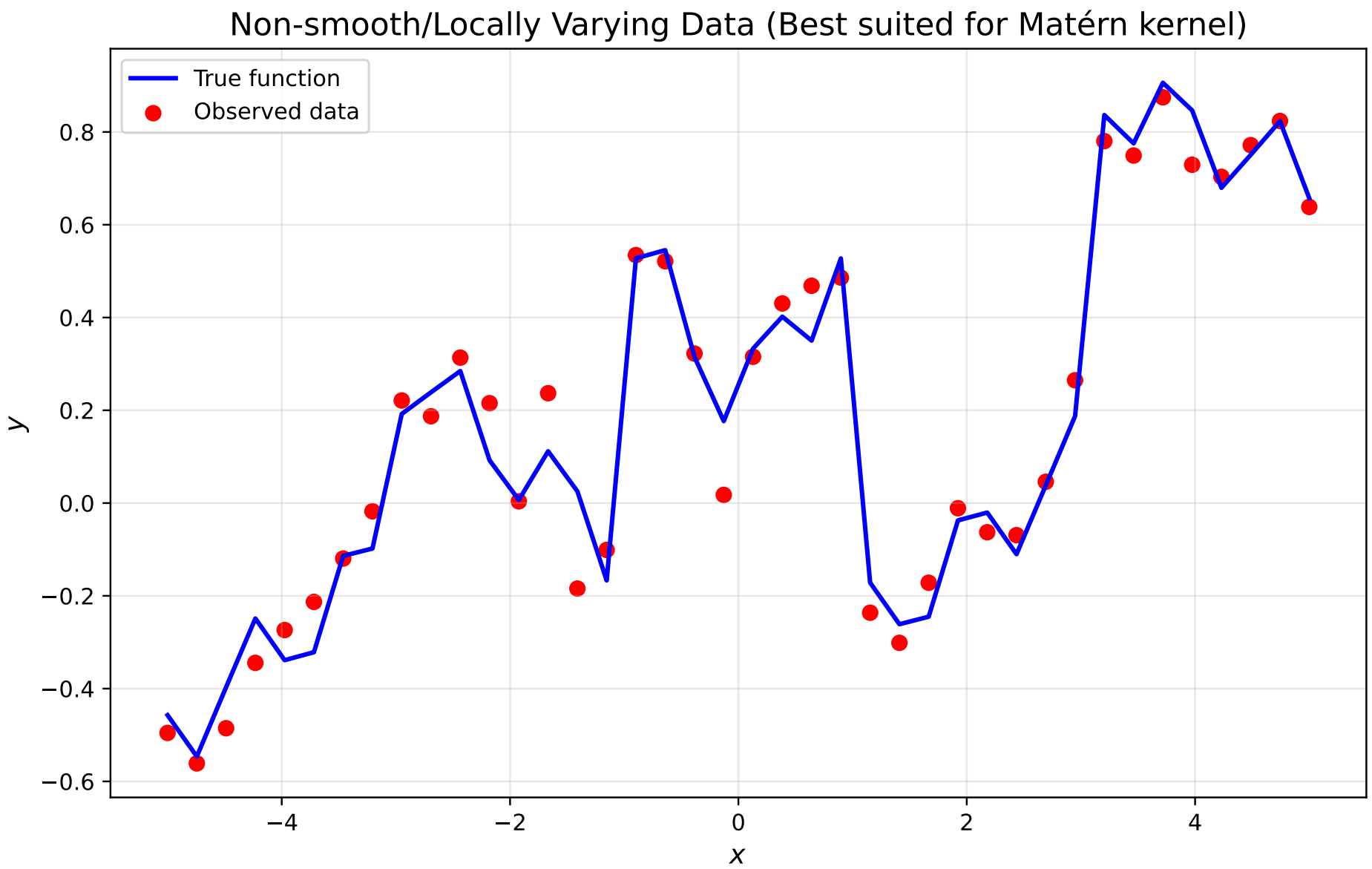
Prior and posterior samples for the squared exponential kernel



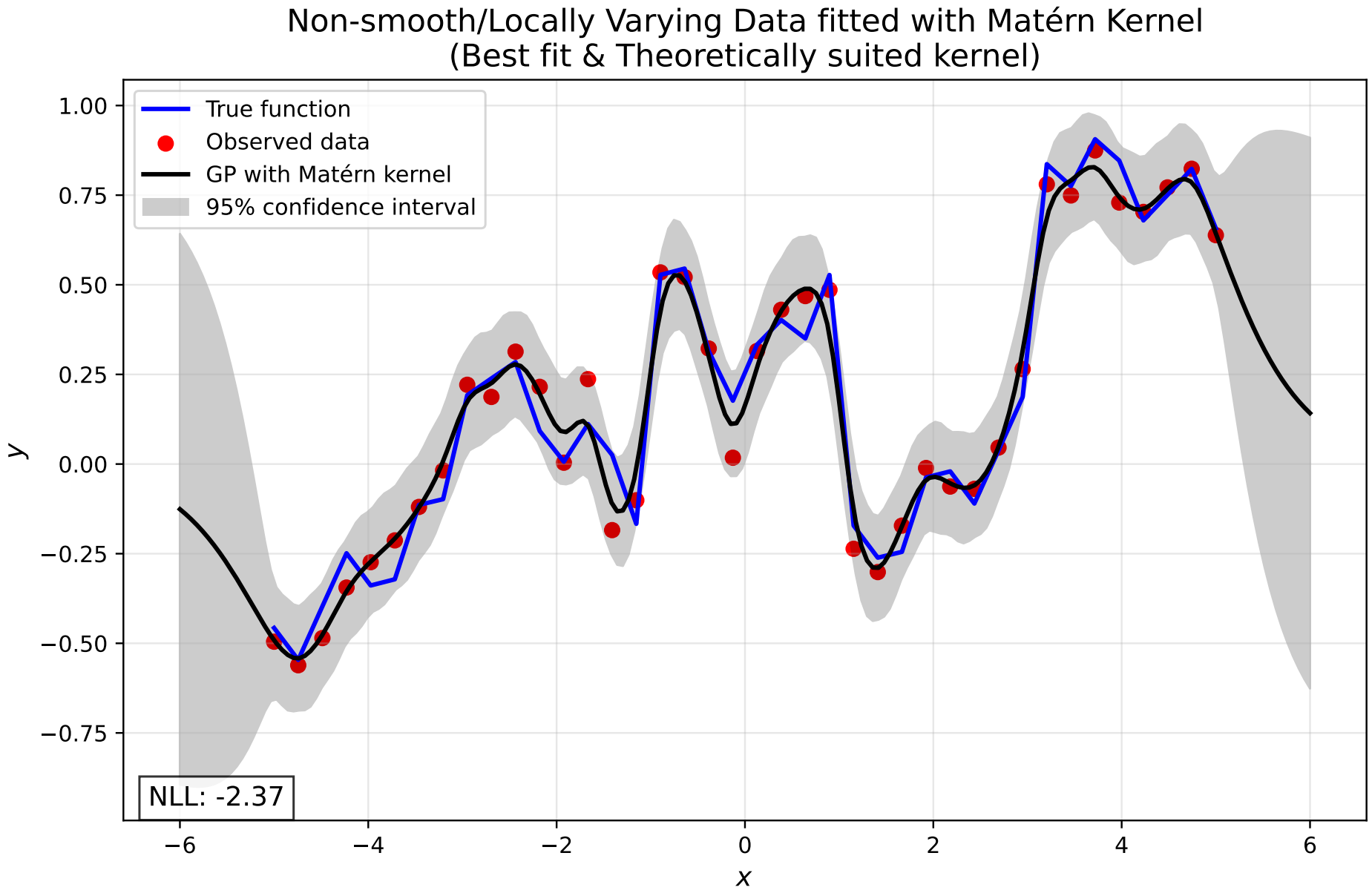
Prior and posterior samples for the squared exponential kernel



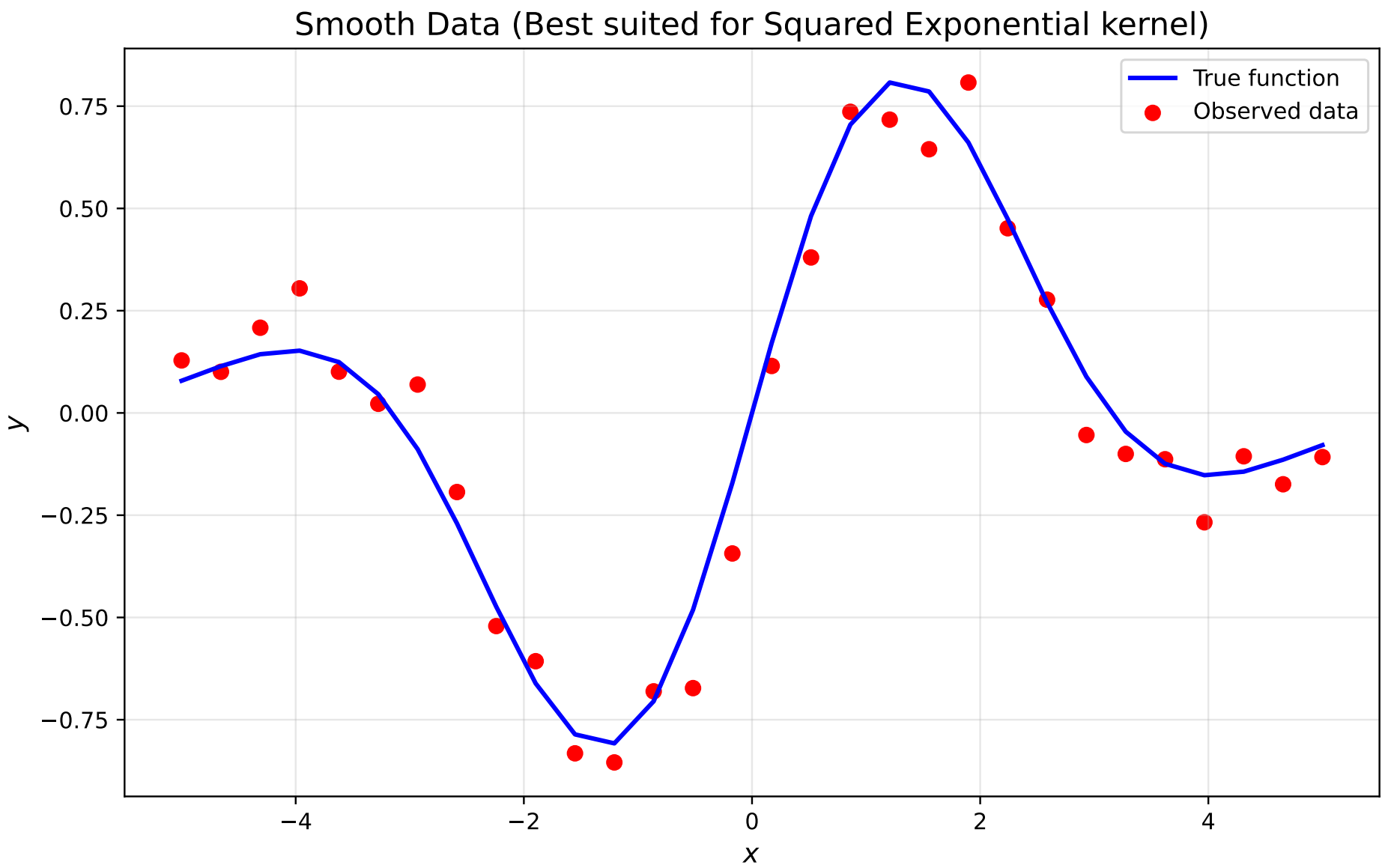
What is the best kernel?



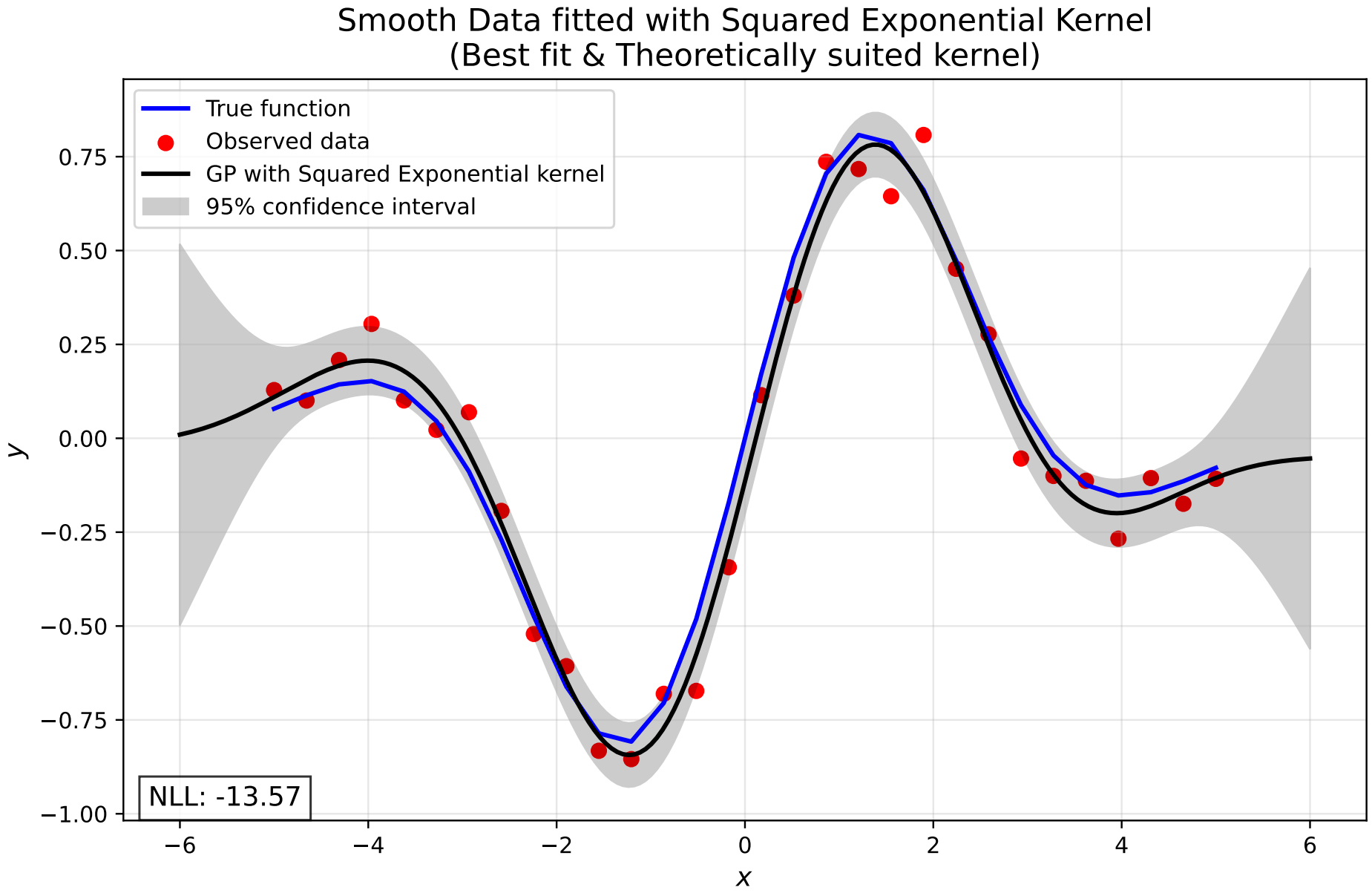
What is the best kernel?



What is the best kernel?



What is the best kernel?



Bayesian model selection

The general task

How do I select the kernel for my GP? How do I then select the hyperparameters for my chosen kernel?

Bayesian model selection

The general task

How do I select the kernel for my GP? How do I then select the hyperparameters for my chosen kernel?

For a collections of models $\{\mathcal{H}_m\}_{m \in \mathcal{M}}$, Bayesian model selection functions through the following hierarchy:

$$\begin{aligned} \boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m &\sim p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m), \\ \boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m &\sim p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m), \\ \boldsymbol{\theta} \mid \mathcal{H}_m &\sim p(\boldsymbol{\theta} \mid \mathcal{H}_m), \\ \mathcal{H}_m &\sim p(\mathcal{H}_m). \end{aligned}$$

Linking back to GPs

Grounding the abstract

We will work in the abstract for now:

$$\begin{aligned}
\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m &\sim p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m), \\
\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m &\sim p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m), \\
\boldsymbol{\theta} \mid \mathcal{H}_m &\sim p(\boldsymbol{\theta} \mid \mathcal{H}_m), \\
\mathcal{H}_m &\sim p(\mathcal{H}_m).
\end{aligned}$$

But note how this relates to GPs:

Abstract		**GP Specific**
$\mathcal{H}_m$	$\rightarrow$	Kernel family (RBF, Matérn, etc.)
$\boldsymbol{\theta}$	$\rightarrow$	Kernel hyperparameters (length scales, signal variance)
$\boldsymbol{w}$	$\rightarrow$	Function values $\boldsymbol{f}$ at observed points
$\boldsymbol{y}$	$\rightarrow$	Noisy observations

Bayesian model selection

We proceed iteratively to compute the posteriors. First, compute the posterior of the parameters:

$$p(\boldsymbol{w} \mid \boldsymbol{y}, X, \boldsymbol{\theta}, \mathcal{H}_m) = \frac{p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m) p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m)}{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m)}.$$

Unnecessarily notationally heavy...

$$\begin{aligned}
p(\boldsymbol{w} \mid \boldsymbol{y}, X, \boldsymbol{\theta}, \mathcal{H}_m) &= \frac{p(\boldsymbol{w}, \boldsymbol{y}, X, \boldsymbol{\theta}, \mathcal{H}_m)}{\int p(\boldsymbol{w}, \boldsymbol{y}, X, \boldsymbol{\theta}, \mathcal{H}_m) \mathrm{d}\boldsymbol{w}} \\
&= \frac{p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m) p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m) p(\mathcal{H}_m)}{p(\boldsymbol{\theta} \mid \mathcal{H}_m) p(\mathcal{H}_m) \int p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m) p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m) \mathrm{d}\boldsymbol{w}} \\
&= \frac{p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m) p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m)}{\int p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m) p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m) \mathrm{d}\boldsymbol{w}} \\
&= \frac{p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m) p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m)}{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m)}
\end{aligned}$$

Bayesian model selection

Then we compute the (marginal) posterior over the hyperparameters:

$$p(\boldsymbol{\theta} \mid X, \boldsymbol{y}, \mathcal{H}_m) = \frac{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m)}{p(\boldsymbol{y} \mid X, \mathcal{H}_m)}.$$

A little less notationally heavy...

$$\begin{aligned} p(\boldsymbol{\theta} \mid \boldsymbol{y}, X, \mathcal{H}_m) &= \frac{p(\boldsymbol{\theta}, \boldsymbol{y}, X, \mathcal{H}_m)}{\int p(\boldsymbol{y}, X, \boldsymbol{\theta}, \mathcal{H}_m) \mathrm{d}\boldsymbol{\theta}} \\ &= \frac{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m) p(\mathcal{H}_m)}{p(\mathcal{H}_m) \int p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m) \mathrm{d}\boldsymbol{\theta}} \\ &= \frac{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m)}{\int p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m) \mathrm{d}\boldsymbol{\theta}} \\ &= \frac{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m)}{p(\boldsymbol{y} \mid X, \mathcal{H}_m)} \end{aligned}$$

Bayesian model selection

Finally, we compute the (marginal) posterior for the model:

$$p(\mathcal{H}_m \mid \boldsymbol{y}, X) = \frac{p(\boldsymbol{y} \mid X, \mathcal{H}_m)p(\mathcal{H}_m)}{p(\boldsymbol{y} \mid X)}.$$

Not too notationally heavy...

$$\begin{aligned} p(\mathcal{H}_m \mid \boldsymbol{y}, X) &= \frac{p(\mathcal{H}_m, X, \boldsymbol{y})}{\sum_m p(\mathcal{H}_m, X, \boldsymbol{y})} \\ &= \frac{p(\boldsymbol{y} \mid X, \mathcal{H}_m)p(\mathcal{H}_m)}{\sum_m p(\boldsymbol{y} \mid X, \mathcal{H}_m)p(\mathcal{H}_m)} \\ &= \frac{p(\boldsymbol{y} \mid X, \mathcal{H}_m)p(\mathcal{H}_m)}{p(\boldsymbol{y} \mid X)} \end{aligned}$$

Bayesian model selection

In summary, we have the three marginal posteriors which we would like to maximise and use for model selection:

$$p(\boldsymbol{w} \mid \boldsymbol{y}, X, \boldsymbol{\theta}, \mathcal{H}_m) = \frac{p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m) p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m)}{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m)},$$

$$p(\boldsymbol{\theta} \mid X, \boldsymbol{y}, \mathcal{H}_m) = \frac{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m)}{p(\boldsymbol{y} \mid X, \mathcal{H}_m)},$$

$$p(\mathcal{H}_m \mid \boldsymbol{y}, X) = \frac{p(\boldsymbol{y} \mid X, \mathcal{H}_m) p(\mathcal{H}_m)}{p(\boldsymbol{y} \mid X)}$$

These depend on the following normalising constants:

$$p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) = \int p(\boldsymbol{y} \mid X, \boldsymbol{w}, \mathcal{H}_m) p(\boldsymbol{w} \mid \boldsymbol{\theta}, \mathcal{H}_m) \mathrm{d}\boldsymbol{w}$$

$$p(\boldsymbol{y} \mid X, \mathcal{H}_m) = \int p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m) \mathrm{d}\boldsymbol{\theta}$$

$$p(\boldsymbol{y} \mid X) = \sum_m p(\boldsymbol{y} \mid X, \mathcal{H}_m) p(\mathcal{H}_m).$$

Selecting the hyperparameters in practice

For hyperparameter selection, we are interested in:

$$p(\boldsymbol{\theta} \mid X, \boldsymbol{y}, \mathcal{H}_m) = \frac{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m)p(\boldsymbol{\theta} \mid \mathcal{H}_m)}{p(\boldsymbol{y} \mid X, \mathcal{H}_m)},$$

We have two approaches:

Exact posterior:

$$p(\boldsymbol{\theta} \mid X, \boldsymbol{y}, \mathcal{H}_m)$$

Pros:

- We get a full distribution (truly Bayesian)

Cons:

- Computationally intensive to sample from.
- Need to specify hyperpriors.

Marginal likelihood:

$$p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m)$$

Pros:

- Computationally easier than the exact posterior.
- Don't need to specify hyperpriors.

Cons:

- Only get a point estimate.
- Not exactly targetting what we want.

Exact posterior

We have an expression for the true posterior: $p(\boldsymbol{\theta} \mid X, \boldsymbol{y}, \mathcal{H}_m)$. We can target this using MCMC!

Consider again the model:

$$\begin{aligned} y_i \mid f, \boldsymbol{x}_i &\sim \mathcal{N}(f(\boldsymbol{x}_i), \sigma_\epsilon^2), \quad i = 1, \ldots, n, \\ f &\sim \mathcal{GP}(0, k_\theta(\cdot, \cdot)). \end{aligned}$$

With the squared exponential kernel:

$$k(x, x') = \alpha \exp\left\{ -\frac{\|x - x'\|_2}{2\ell^2} \right\},$$

the model has hyperparameter vector $\boldsymbol{\theta} = (\alpha, \ell)$, we place hyperiors on them:

$$\begin{aligned} \alpha &\sim \text{InvGamma}(5, 1) \\ \ell &\sim \text{InvGamma}(5, 1). \end{aligned}$$

We can now write down the log density of the hyperpriors $\log p(\ell), \log p(\alpha)$ and the loglikelihood $\log p(\boldsymbol{y} \mid X, \alpha, \ell)$ and the MCMC does the rest!

Good new for my Bayesian brethren

This is a *fully* Bayesian approach (yay).

Maximal marginal likelihood

In the expression for the hyperparameter posterior:

$$p(\boldsymbol{\theta} \mid X, \boldsymbol{y}, \mathcal{H}_m) = \frac{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) p(\boldsymbol{\theta} \mid \mathcal{H}_m)}{p(\boldsymbol{y} \mid X, \mathcal{H}_m)},$$

the normalising constant is hard to obtain. Instead, we can maximise the *marginal likelihood* in the numerator:

$$p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m).$$

This is OK:

- with uninformative priors, the maxima coincide,
- with informative priors, with sufficient data the likelihood dominates.

This is called "Empirical Bayes" or "Type II Maximum Likelihood".

Warning for my Bayesian brethren

This isn't a *truly* Bayesian approach as we aren't utilising the posteriors.

A warning about maximisation

For a GP, the marginal loglikelihood takes the form:

$$\log p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m) = -\frac{1}{2}\boldsymbol{y}^T \left(K_{\boldsymbol{\theta}}(X, X) + \sigma_\epsilon^2 I_n\right)^{-1} \boldsymbol{y} - \frac{1}{2}\log\det\left\{K_{\boldsymbol{\theta}}(X, X) + \sigma_\epsilon^2 I_n\right\} - \frac{n}{2}\log(2\pi)$$

This is usually (very, very) far from convex and will contain many local maxima, making am optimisation scheme very sensitive to initialisation.

Practically, this means we need to do multiple runs with different initialisations (construct a grid based on heuristics) and select the solution that leads to the largest marginal maximum likelihood.

Selecting the kernel in practice

But how do we actually select the model (kernel) in practice?

The outlined methodology gives a principled framework for selecting a kernel and thus model $\mathcal{H}_m$ through $p(\mathcal{H}_m \mid \boldsymbol{y}, X)$.

However, for tractability, we don't often work with $p(\mathcal{H}_m \mid \boldsymbol{y}, X)$.

In practice:

1. Propose a number of kernels $k_{1,\boldsymbol{\theta}_1}$, $k_{2,\boldsymbol{\theta}_2}$, $\ldots$, $k_{K,\boldsymbol{\theta}_K}$.
2. For each choice, use the above maximum marginal likelihood approach to select the best parameter for that kernel.
3. Select the kernel whose best parameters lead to the largest maximum marginal likelihood.

Summary of the practical approach

How does this all tie together?

1. Observe data pairs $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ which we hypothesise follows $y_i = f(\boldsymbol{x}_i) + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$.
2. Select a number of kernels $\{k_{m,\boldsymbol{\theta}_m}\}_{m=1}^M$ each depending on hyperparameters $\boldsymbol{\theta}_m$ (this is selecting a model $\mathcal{H}_m$) and specify a prior on $f \sim \mathcal{GP}(0, k_{m,\boldsymbol{\theta}_m}(\cdot,\cdot))$.
3. For each model $\mathcal{H}_m$, maximise the marginal likelihood $p(\boldsymbol{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_m)$ over $\boldsymbol{\theta}_m$ to get $\hat{\boldsymbol{\theta}}_m$.

   To find each $\hat{\boldsymbol{\theta}}_m$, we will select an optimiser and start from different values of $\boldsymbol{\theta}_m$ as we know the function is non-convex.

   Select the model $\mathcal{H}_m$ with the largest maximum marginal likelihood. Write the optimal model index as $\hat{m} \in [M]$.
4. For prediction, work with a GP prior with kernel $k_{\hat{m},\hat{\boldsymbol{\theta}}_m}(\cdot,\cdot)$, depending on the optimised parameter $\hat{\boldsymbol{\theta}}_m$.

Where are we?

- We've changed our perspective from weight space to function space. We are now in a position to formally define a Gaussian process.
- We formally defined a Gaussian process and demonstrated its predictive properties. We saw how this works in a simple example.
- We discussed the theoretical and practical considerations around fitting a GP. We now have a practical strategy for selecting the kernel and its subsequent hyperparameters.

What do we mean by a draw?

Just as an aside

We have worked in the abstract/theoretical, let's try and ground ourselves!

What do we mean to say that we "draw" from the GP?

We can't draw infinitely many points. Practically, when we "draw" from a GP, we choose a grid of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and evaluate the kernel on that grid:

$$K(X, X) = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}$$

We then have a multivariate Gaussian of dimension n, and have $\boldsymbol{f} \sim \mathcal{N}(0, K(X, X))$, where here we are thinking of $\boldsymbol{f}$ as a vector representing our function.

A draw from this normal distribution comprises the draws we saw before:

$$\boldsymbol{f} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^T.$$

BUT... how do we even draw from a multivariate normal?

Drawing from a multivariate normal

Decomposing a covariance matrix

Positive definite matrix

When a symmetric $n \times n$ matrix A satisfy the following conditions, A is called a **positive definite matrix**.

$$\boldsymbol{z}^\top A \boldsymbol{z} = \sum_{i=1}^{n} \sum_{j=1}^{n} z_i z_j A_{ij} > 0$$

here $\boldsymbol{z} \in \mathbb{R}^n$ is any non zero vector.

Cholesky decomposition

A positive definite matrix can be decomposed in to a lower triangular matrix and its transpose.

$$LL^\top = A$$

Here, L is called the **Cholesky factor**.

Cholesky decomposition

Cholesky decompositions are very numerically stable to compute.

As an example, we can decompose this positive definite matrix as:

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

Our covariance matrices are positive definite, so we can decompose them.

Using Cholesky to draw from a multivariate normal

OK, so we want to draw $\boldsymbol{f} \sim \mathcal{N}(0, K(X, X))$.

$K(X, X)$ is positive definite, so let's decompose it as $K(X, X) = LL^T$, so

$$\boldsymbol{f} \sim \mathcal{N}(0, LL^T).$$

Recall basic linearity of Gaussians:

$$\boldsymbol{z} \sim \mathcal{N}(0, I) \quad \Rightarrow \quad B\boldsymbol{z} \sim \mathcal{N}(0, BB^T).$$

So, to draw $\boldsymbol{f}$, we can do:

- Draw $\boldsymbol{z} \sim \mathcal{N}(0, I)$
- Transform $\boldsymbol{f} = L\boldsymbol{z}$!

This is cheap and numerically stable (and it's what we do in Stan!)

Things that we need to consider next

We have worked almost exclusively with theory, and we need to ground this in practical examples.

Computing an $n \times n$ covariance matrix has an $\mathcal{O}(n^2)$ cost. Computing the inverse of this covariance matrix has an $\mathcal{O}(n^3)$ cost. We will be very limited in the size of the datasets we can consider!

Next, we will discuss numerical issues, solutions and strategies for approximation.

Gaussian process approximation

The need to approximate

The posterior predictive for function values $\boldsymbol{f}_*$ at input points $X_* \in \mathcal{X}^m$ takes the form:

$$\boldsymbol{f}_* \mid X, \boldsymbol{y}, X_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \Sigma_*),$$

where

$$\begin{aligned}\boldsymbol{\mu}_* &= K(X_*, X)(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}\boldsymbol{y} \\ \Sigma_* &= K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}K(X, X_*).\end{aligned}$$

Computing the inverse: $(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}$ is an $\mathcal{O}(n^3)$ operation, and becomes very slow for $n > 1000$!

Key point

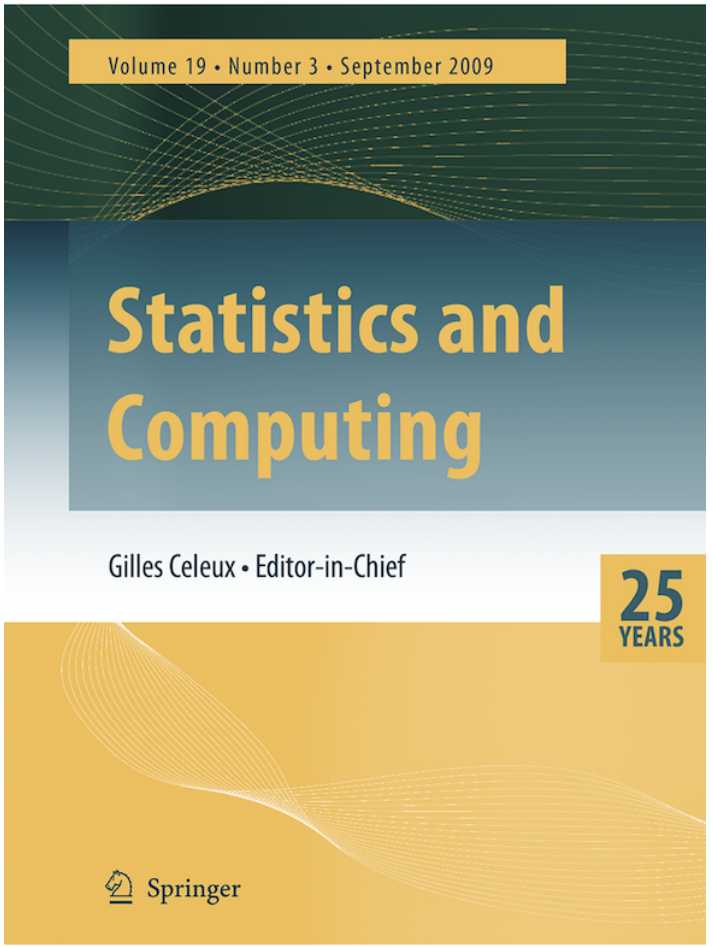
We need to develop methods for approximation...

Gaussian process approximations

How do we reduce the computational cost of GPs

There are many methods for reducing the computational cost of GPs.

I will introduce a method called **Hilbert space approximate Gaussian process** (HSGP) proposed by Solin & Sarkka in 2019 for which Riutort-Mayol et al. wrote a tutorial paper for implementation in probabilistic programming languages.



1. https://link.springer.com/article/10.1007/s11222-019-09886-w
2. https://link.springer.com/article/10.1007/s11222-022-10167-2

Stationary kernels

Invariant to translations

Stationary covariance kernels

A kernel function $k(\cdot, \cdot)$ is **stationary** if it is function of displacement only: $\boldsymbol{\tau} = \boldsymbol{x} - \boldsymbol{x}' \in \mathbb{R}^D$ *i.e.*, and so can be written as,

$$k(\boldsymbol{x}, \boldsymbol{x}') \equiv k(\boldsymbol{\tau}).$$

Stationary covariance kernels are invariant to translations in the input space.

For example,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha \exp\left\{-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\ell^2}\right\} = \alpha \exp\left\{-\frac{\|\boldsymbol{\tau}\|^2}{2\ell^2}\right\} = k(\boldsymbol{\tau}).$$

Stationary kernels

Bochner's Theorem

Bochner's theorem states that a bounded, continuous and positive definite kernel function $k(\boldsymbol{\tau})$ can be represented as

$$k(\boldsymbol{\tau}) = \frac{1}{(2\pi)^d} \int \exp\left(i\boldsymbol{\omega}^T\boldsymbol{\tau}\right) \mu(\mathrm{d}\boldsymbol{\omega}),$$

where μ is a positive measure. If $\mu(\boldsymbol{\omega})$ has a density, it is called the **spectral density** $S(\boldsymbol{\omega})$.

Spectral density functions

For every stationary covariance kernel, there is a spectral density

Every stationary covariance kernel has a corresponding spectral density function. For instance, the D-dimensional Matérn class covariance kernel has the following spectral density function

$$S_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \alpha \frac{2^D \pi^{D/2} \Gamma(\nu + D/2)(2\nu)^\nu}{\Gamma(\nu)\ell^{2\nu}} \left(\frac{2\nu}{\ell^2} + 4\pi^2 \boldsymbol{\omega}^\top \boldsymbol{\omega} \right)^{-(\nu + D/2)}$$

Here, $\boldsymbol{\omega} \in \mathbb{R}^D$ is a vector in the frequency domain and $\boldsymbol{\theta} = (\nu, \ell, \alpha)$.

Matérn class covariance kernels and respective spectral densities for $D = 1$

Name	Expression	Spectral density
Squared exponential	$\alpha^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$	$\alpha\sqrt{2\pi}\ell \exp\left(-\frac{1}{2}\ell^2\omega^2\right)$
Matérn 3/2	$\alpha\left(1 + \frac{\sqrt{3}r}{\ell}\right)\exp\left(-\frac{\sqrt{3}r}{\ell}\right)$	$4\alpha\frac{3^{3/2}}{\ell^2}\left(\frac{3}{\ell^2} + \omega^2\right)^{-2}$
Matérn 5/2	$\alpha\left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right)\exp\left(-\frac{\sqrt{5}r}{\ell}\right)$	$32\alpha\frac{5^{5/2}}{3\ell^5}\left(\frac{5}{\ell^2} + \omega^2\right)^{-3}$

Representing kernels with spectral density functions

Spectral density + eigenvalues + eigenvectors

Here we have a key result for how we can express a kernel as an infinite sum.

Expressing stationary kernels using spectral density functions

On a compact domain $\Omega = [-L, L] \subset \mathbb{R}$, stationary kernels can be written as the following infinite sum:

$$k(x, x') = \sum_{s=1}^{\infty} S_\theta(\sqrt{\lambda_s})\phi_s(x)\phi_m(x')$$

where, S_θ is the spectral density, and λ_m, $\phi_m(x)$ are given as,

$$\lambda_s = \left(\frac{s\pi}{2L}\right)^2, \quad \phi_s(x) = \sqrt{\frac{1}{L}} \sin\left(\sqrt{\lambda_s}(x + L)\right)$$

respectively. Note that the eigenvalues and eigenfunctions do not depend on the spectral density.

Approximating the kernel

Removing the high finer details

Notice that the eigenfunction $\phi_m(x)$ is a periodic function which increases its frequency with m. Most information about the kernel is contained within the low frequency components:

$$S_\theta(\omega) \propto \left(\frac{2\nu}{\ell^2} + 4\pi^2\omega^2\right)^{-(\nu+1/2)} \to 0 \quad \text{as } |\omega| \to \infty.$$

Because of this, we can truncate the infinite sum

$$k(x, x') = \sum_{s=1}^{\infty} S_\theta(\sqrt{\lambda_s})\phi_s(x)\phi_s(x'),$$

to the first p terms, and approximate the kernel as

$$k(x, x') \approx \sum_{s=1}^{p} S_\theta(\sqrt{\lambda_s})\phi_s(x)\phi_s(x').$$

Key point

Covariance kernels can be approximated using the spectral density and the first m terms of the infinite sum.

Gaussian process approximations

Rewriting in matrix notation

Let's rewrite this in matrix notation:

Approximation of the covariance kernel

$$k(x, x') \approx \sum_{s=1}^{p} S_\theta(\sqrt{\lambda_s})\phi_s(x)\phi_s(x') = \boldsymbol{\phi}(x)^\top \Delta \boldsymbol{\phi}(x')$$

where $\boldsymbol{\phi}(x) = \{\phi_s(x)\}_{s=1}^{p} \in \mathbb{R}^p$ is a column vector of eigenfunction values and $\Delta \in \mathbb{R}^{p \times p}$ is a diagonal matrix consisting of spectral densities evaluated at the square root of the eigenvalues.

$$\Delta = \begin{pmatrix} S_\theta(\sqrt{\lambda_1}) & & \\ & \ddots & \\ & & S_\theta(\sqrt{\lambda_p}) \end{pmatrix}$$

Gaussian process approximation

The covariance matrix

When using this approximation, the covariance matrix becomes

$$K(X, X) \approx \Phi(X)\Delta\Phi(X)^\top.$$

Here, $\Phi \in \mathbb{R}^{n \times p}$ is a matrix of eigenfunctions.

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \dots & \phi_p(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_n) & \dots & \phi_p(x_n) \end{pmatrix}$$

We want to sample

$$\boldsymbol{f} \sim \mathcal{N}(0, K(X, X)),$$

which we approximate with a draw,

$$\tilde{\boldsymbol{f}} \sim \mathcal{N}(0, \Phi(X)\Delta\Phi(X)^T) \quad \iff \quad \boldsymbol{f} = \Phi(X)\Delta^{1/2}\boldsymbol{z}, \qquad \boldsymbol{z} \sim \mathcal{N}(0, I).$$

Thinking of a function $f(x)$ as a very long vector $(f_1(x), f_2(x), \cdots)$, you can convince yourself that a draw f from a GP prior evaluated at x is:

$$f(x) \approx \sum_{m=1}^{M} (S_\theta(\sqrt{\lambda_m}))^{1/2} \phi_m(x) \beta_m$$

where $\beta_m \sim \mathcal{N}(0, 1)$.

Approximating the predictive

Our predictive for new data $X_* \in \mathcal{X}^m$ is exactly:

$$\boldsymbol{f}_* \mid X, \boldsymbol{y}, X_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \Sigma_*),$$

where

$$\begin{aligned}
\boldsymbol{\mu}_* &= K(X_*, X)(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}\boldsymbol{y} \\
\Sigma_* &= K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_\epsilon^2 I_n)^{-1}K(X, X_*).
\end{aligned}$$

Now we can *approximate* this predictive distribution using the covariance approximation:

$$\begin{aligned}
\boldsymbol{\mu}_* &\approx \Phi(X_*)\Delta\Phi(X)^T \left(\underbrace{\Phi(X)\Delta\Phi(X)^T}_{n\times n} + \sigma_\epsilon^2 I_n \right)^{-1} \boldsymbol{y} \\
&= \Phi(X_*) \left(\underbrace{\Phi(X)^T\Phi(X)}_{p\times p} + \sigma_\epsilon^2 \Delta^{-1} \right)^{-1} \Phi(X)^T\boldsymbol{y} \\
\Sigma_* &\approx \Phi(X_*)\Delta\Phi(X_*)^T - \Phi(X_*)\Delta\Phi(X)^T \left(\Phi(X)\Delta\Phi(X)^T + \sigma_\epsilon^2 I_n\right)^{-1} \Phi(X)\Delta\Phi(X_*)^T \\
&= \sigma_\epsilon^2 \Phi(X_*) \left(\Phi(X)^T\Phi(X) + \sigma_\epsilon^2\Delta^{-1}\right)^{-1} \Phi(X_*).
\end{aligned}$$

Reduction in the computational cost

How much did we gain

Upon examination of the approximation

$$f(x) \approx \sum_{s=1}^{p} (S_\theta(\sqrt{\lambda_s}))^{1/2} \phi_s(x) \beta_s$$

we notice the following:

1. λ_s and $\phi_s(x)$ does not depend on the parameters of the GP. Thus we only need to compute them once beforehand and reuse them
2. Only the p spectral density terms $\{S_\theta(\sqrt{\lambda_s})\}_{s=1}^{p}$ is dependent on the GP parameters

Key point

For each MCMC iteration we need to calculate

1. The value of p spectral densities $S_\theta(\sqrt{\lambda_s})$ and $(\mathcal{O}(p))$,
2. the p term sum of n data points $(\mathcal{O}(np))$

Hence, the total computational cost works out to be $\mathcal{O}(np + p)$.

In general $p \ll n$ thus compared to $\mathcal{O}(n^3)$, we significantly reduce the necessary computations.

Boundary condition L and number of eigenfunctions

p

Some hyperparameters

To use HSGP, we need to specify L **which determines the range** $\Omega = [-L, L]$ **for which the approximation is valid and the number of eigenfunctions** p.

Let $\{x_i\}_{i=1}^n$ be some univariate input data with mean 0. If we let $S = \max_i |x_i|$, than for any given i, $x_i \in [-S, S]$. Hence, we can define the boundary condition L as follows:

Boundary condition

$$L = c \times S$$

where, $c \geq 1$ is a proportional scaling factor.

The relationship between c, M, and ℓ

How to set the hyper parameters

Riutort-Mayol et al. (2022) studied the accuracy of the approximation for different values of the proportional scaling factor c, the number of eigenfunctions p, and the lengthscale ℓ for the Squared Exponential, Matérn 3/2, Matérn 5/2 kernels. They provide recommendations for the value of c and p based on empirical results.

Covariance kernel	Conditions for the hyperparameters	
Squared exponential	$p = 1.75\frac{c}{\ell/S}$,	$c \geq 3.2\ell/S$ & $c \geq 1.2$
Matérn 5/2	$p = 2.65\frac{c}{\ell/S}$,	$c \geq 4.1\ell/S$ & $c \geq 1.2$
Matérn 3/2	$p = 3.42\frac{c}{\ell/S}$,	$c \geq 4.5\ell/S$ & $c \geq 1.2$

Key point

Riutort-Mayol et al. (2022) acts as a initial guideline, but because ℓ is unknown, the hyper-parameters may need to be tuned. In general, regardless of the kernel $c \geq 1.2$ in order for the approximation to be accurate.

References and Further reading

Some great resources on GPs

- Carl Edward Rasmussen and Christopher K. I. Williams. **Gaussian Process for Machine Learning**. The MIT Press, 2006.
- Mark van der Wilk's course on **Probilistic Inference**. GitHub repository.
- Arno Solin and Simo Särkkä. **Hilbert space methods for reduced-rank Gaussian process regression**. link.
- Riutort-Mayol et al. **Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming**. link.

Imperial College London

MACHINE LEARNING & GLOBAL HEALTH NETWORK

Thank you.

Joshua Corneck
Imperial College London

March 24, 2025