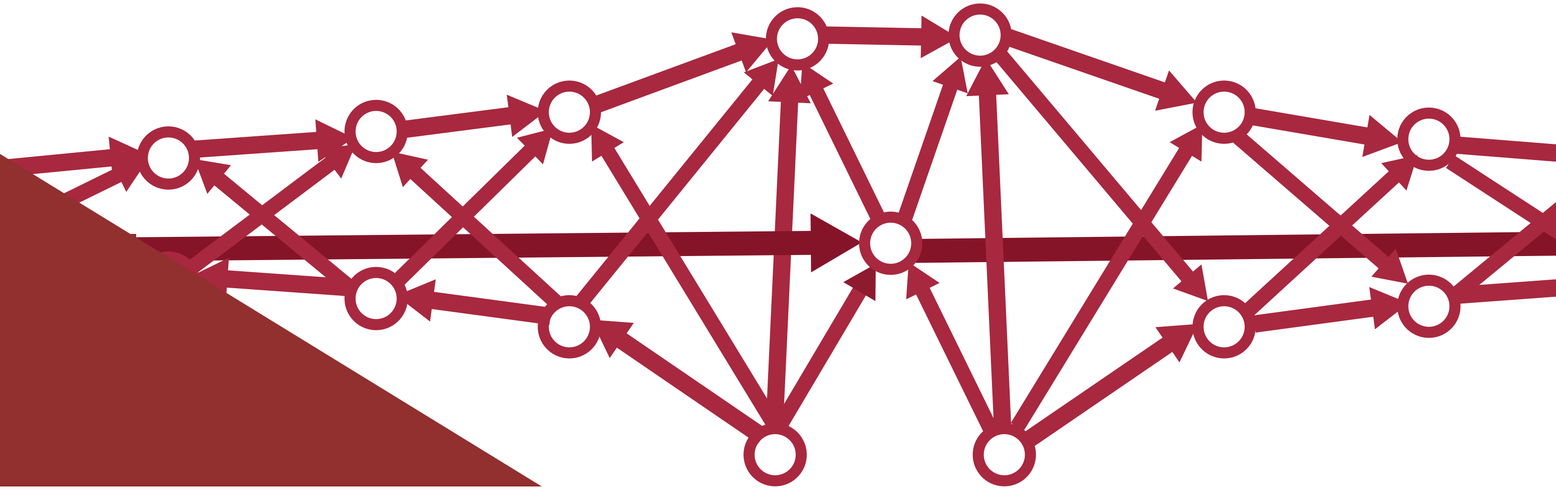
Machine Learning & Data Analysis



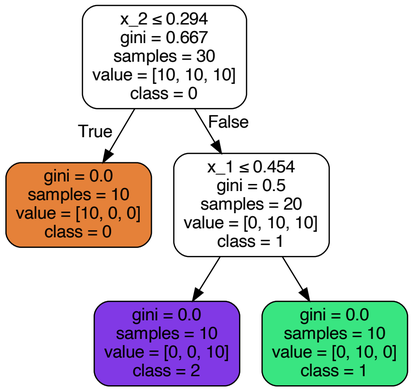
Week 8: Gaussian Processes

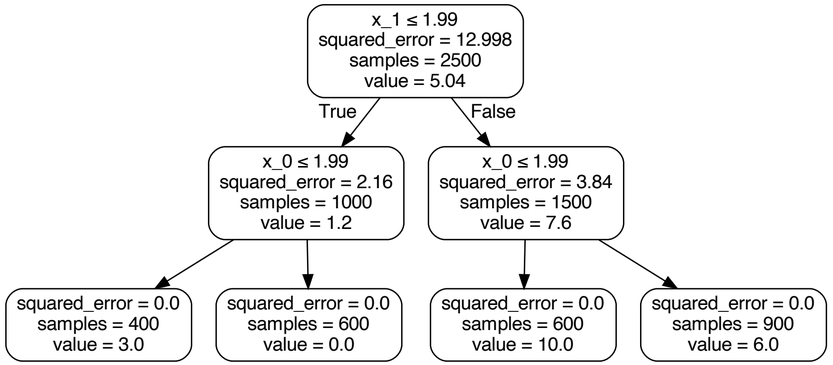
Elliot J. Crowley, 9/11/26

THE UNIVERSITY *of* EDINBURGH

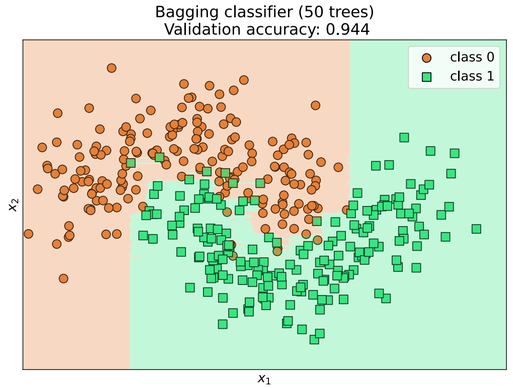
Recap

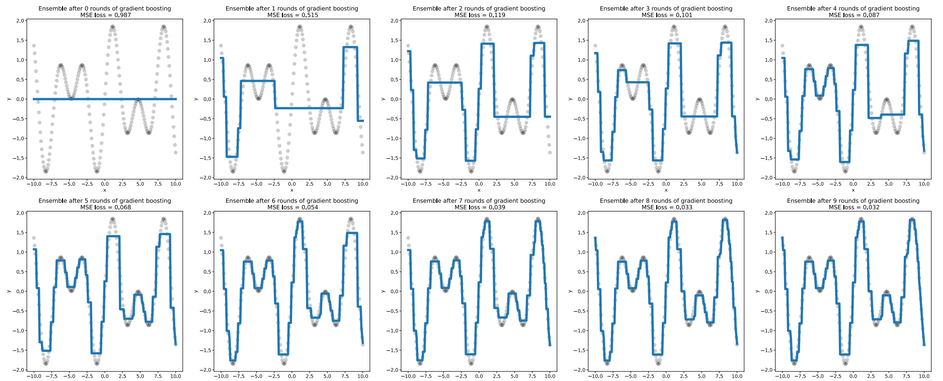
- We looked at classification and regression trees





- We looked at bagging and boosting as techniques for forming ensembles





Gaussian Processes (GPs)

- A non-parametric model that can be used for regression and classification
- **We are only going to consider GPs for regression on this course**
- Let's say there is some unknown function we want to model
- If we model this function with a Gaussian process we assume that the value of the function at any point is a random variable…
- And that any combination of such random variables has a multivariate Gaussian distribution

Random variables

“A mathematical formalisation of a quantity or object which depends on random events”

Wikipedia

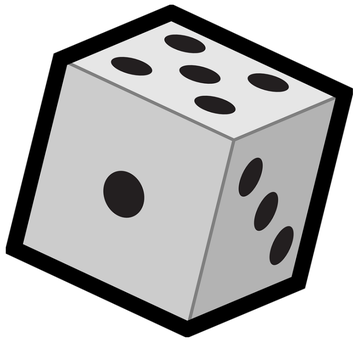
“Something that gives you a different value each time you record it”

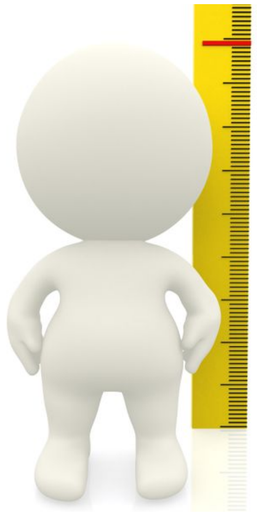
Elliot J. Crowley

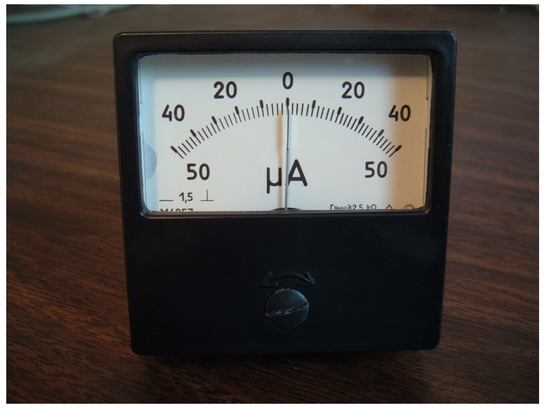
This is not a rigorous definition. A random variable can represent something that hasn’t happened yet for instance

Random variables

- The roll of a die can be treated as a random variable
- Pretty much anything we take measurements of (my height, current flow in a circuit, the mass of a currant bun) can be treated as a random variable
- This is because there will (almost always) be some form of noise giving us fluctuations between measurements







Continuous random variables

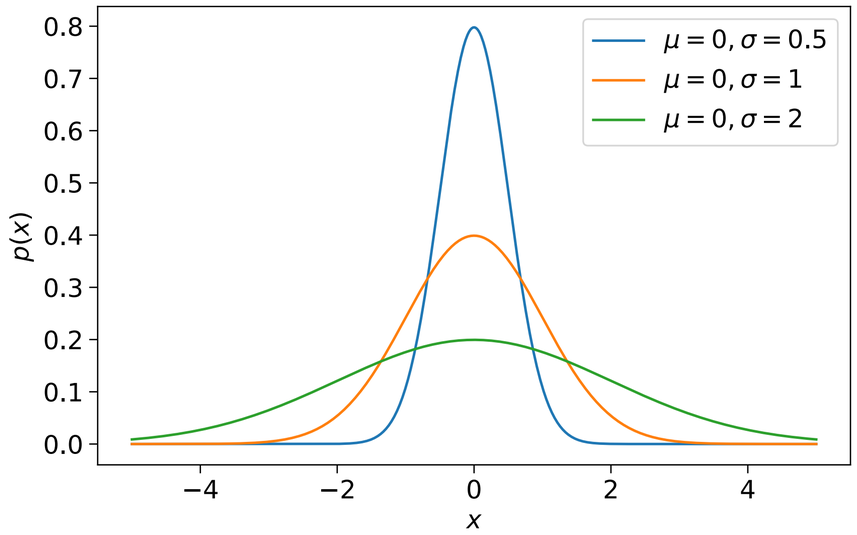
Most of you learnt about these in EM2B

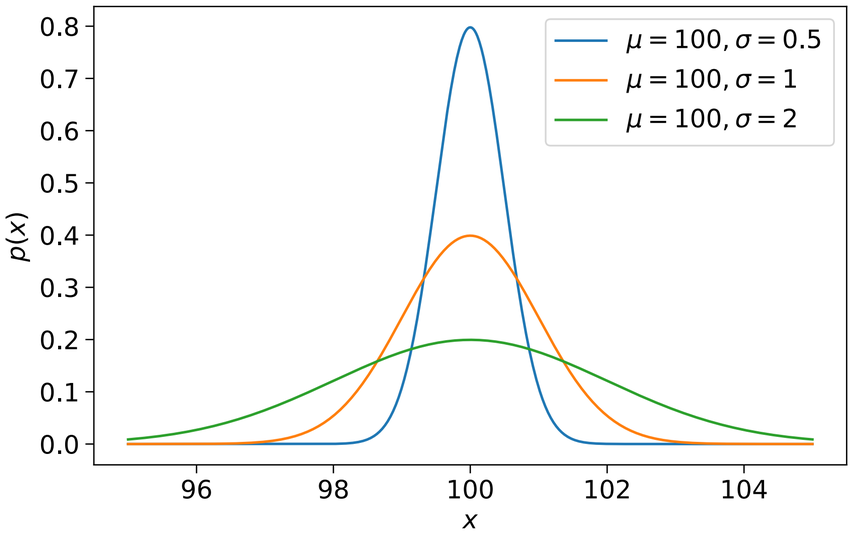
- Let $X \in \mathbb{R}$ be a continuous random variable

- We can describe X using a probability density function (pdf) $p(x)$
- $\mathrm{P}(a < X \leq b) = \int_a^b p(x)\,dx$ so it follows that $\int_{-\infty}^{\infty} p(x)\,dx = 1$
- The mean and variance of X can be computed from $p(x)$
- $\mathbb{E}[X] = \int_{-\infty}^{\infty} xp(x)dx = \mu \qquad \mathbb{E}[(X-\mu)^2] = \mathbb{V}[X] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx = \sigma^2$
- But what should we use for $p(x)$?

The Gaussian (/normal) distribution

- We will default to using Gaussian pdfs
- Gaussians are easy to interpret, mathematically convenient, and can be justified by invoking the Central limit theorem with some handwaving
- $p(x) = \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$ or $X \sim \mathcal{N}(\mu, \sigma^2)$





Bivariate Gaussians

- Suppose we now have two random variables X_1, X_2 that we care about
- We can assume that they are **jointly** Gaussian $p(x_1, x_2) = \mathcal{N}(x_1, x_2; \boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{bmatrix} \right)$$

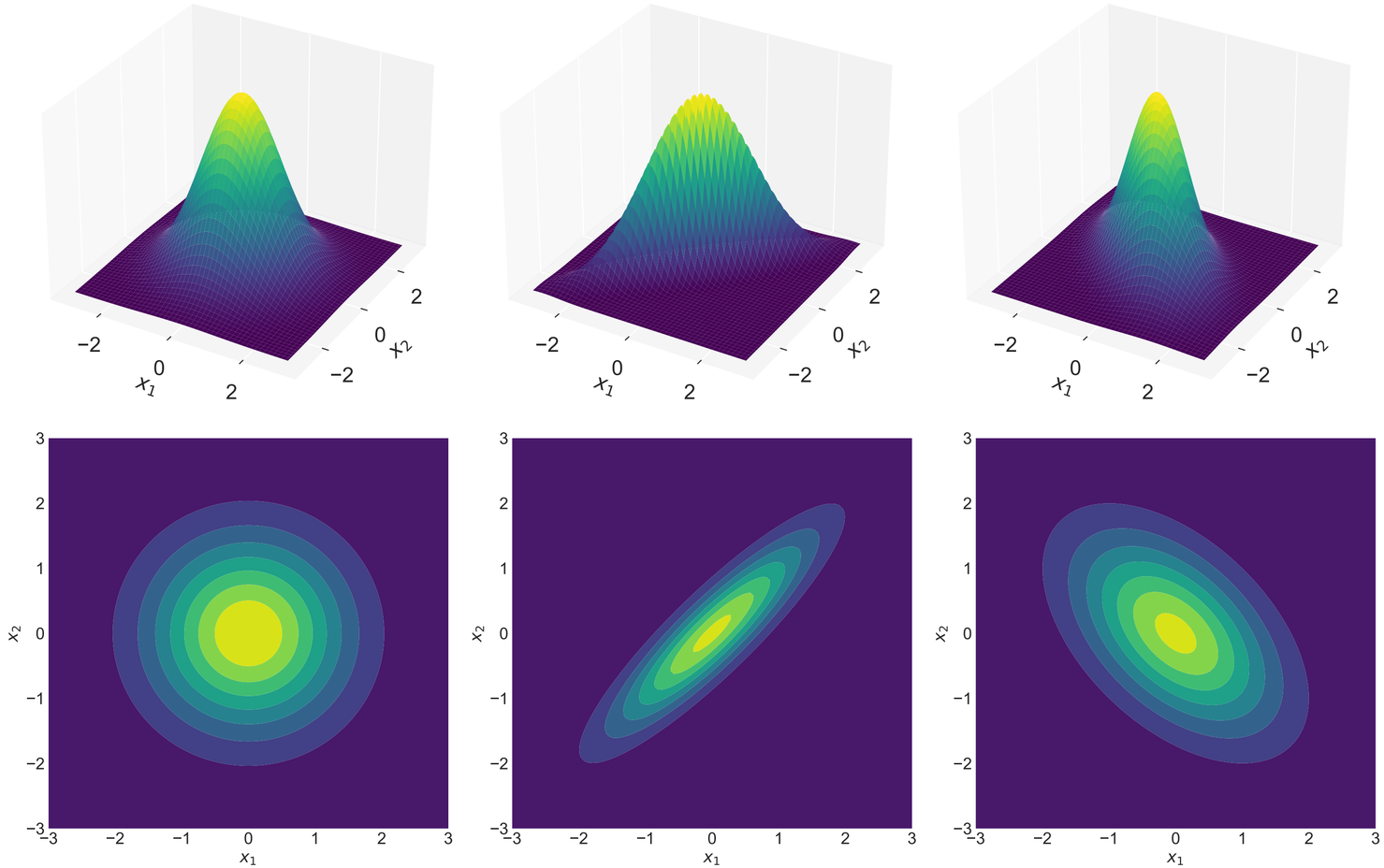
- Here $\Sigma_{i,j} = \mathbb{E}\Big[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])\Big]$. I am using this notation to represent both variances ($i = j$) and covariances ($i \neq j$)

Examples of Bivariate Gaussians

$$p(x_1, x_2) = \mathcal{N}(x_1, x_2; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{bmatrix}$$

$\Sigma_{1,2} = 0$ $\Sigma_{1,2} = 0.9$ $\Sigma_{1,2} = -0.5$



Code for plots adapted from https://www.geeksforgeeks.org/visualizing-the-bivariate-gaussian-distribution-in-python/

Marginalising and conditioning a Bivariate Gaussian

- $$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{bmatrix} \right)$$

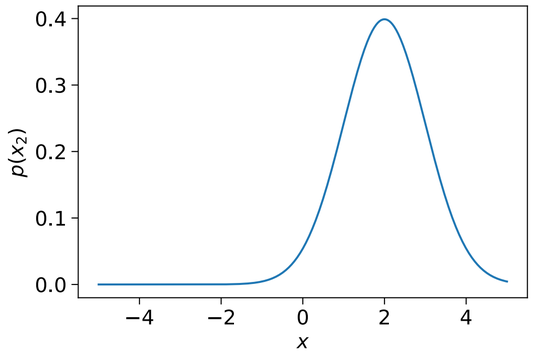
- **Marginalising:**
  $p(x_1) = \mathcal{N}(x_1; \mu_1, \Sigma_{1,1})$ and $p(x_2) = \mathcal{N}(x_2; \mu_2, \Sigma_{2,2})$. **These are Gaussian**

- **Conditioning:**
  What happens if we find out that the exact value for X_1 is x_1?
  We can consider $p(x_2 \mid X_1 = x_1)$. The probability distribution over x_2 conditioned on knowing $X_1 = x_1$. **This is also Gaussian**!

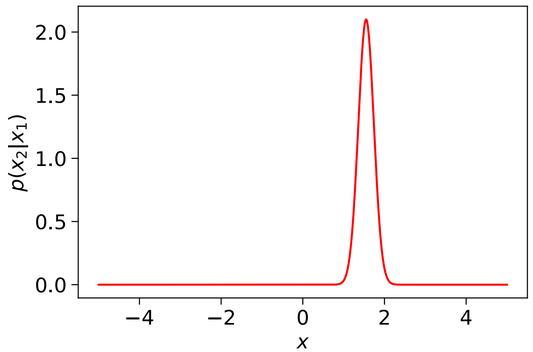
$$p(x_2 \mid X_1 = x_1) = \mathcal{N}(x_2; \mu_2 + \frac{\Sigma_{1,2}}{\Sigma_{1,1}}(x_1 - \mu_1), \Sigma_{2,2} - \frac{\Sigma_{1,2}^2}{\Sigma_{1,1}})$$

Conditioning a Bivariate Gaussian example

- Suppose $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix} \right)$
- We can marginalise to get $p(x_2) = \mathcal{N}(x_2; 2, 1)$



- We then find out that x_1 is definitely -0.5
- $p(x_2 \mid X_1 = -0.5) = \mathcal{N}(x_2; 1.55, 0.19)$



Multivariate Gaussians

- If we have several random variables we care about $X_1, X_2, \ldots, X_D$ then we can combine these into a random vector $\mathbf{x} \in \mathbb{R}^D$
- We can assume these are all jointly Gaussian!
- $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$
- Here $\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_D \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} & \cdots & \Sigma_{1,D} \\ \Sigma_{2,1} & \Sigma_{2,2} & \cdots & \Sigma_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{D,1} & \Sigma_{D,2} & \cdots & \Sigma_{D,D} \end{bmatrix}$

Marginalising a Multivariate Gaussian

If $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_D \end{bmatrix}, \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} & \cdots & \Sigma_{1,D} \\ \Sigma_{2,1} & \Sigma_{2,2} & \cdots & \Sigma_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{D,1} & \Sigma_{D,2} & \cdots & \Sigma_{D,D} \end{bmatrix} \right)$ then…

$$p(x_1) = \mathcal{N}(x_1; \mu_1, \Sigma_{1,1})$$

$$p(x_2) = \mathcal{N}(x_2; \mu_2, \Sigma_{2,2})$$

$$p(x_i) = \mathcal{N}(x_i; \mu_i, \Sigma_{i,i}) \text{ etc.}$$

Conditioning a Multivariate Gaussian

- Let's partition $\mathbf{x}$ as $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]^\top$ where $\mathbf{x}_1 \in \mathbb{R}^{D_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{D_2}$ $D = D_1 + D_2$
- We can write $\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2} \\ \boldsymbol{\Sigma}_{2,1} & \boldsymbol{\Sigma}_{2,2} \end{bmatrix})$
- Let's say we now find out that $\mathbf{x}_1$ is exactly … $\mathbf{x}_1$ (lazy notation :)). We can write $p(\mathbf{x}_2 \,|\, \mathbf{x}_1) = \mathcal{N}(\mathbf{x}_2; \boldsymbol{\mu}_{2|1}, \boldsymbol{\Sigma}_{2|1})$

$$\boldsymbol{\mu}_{2|1} = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{2,1}\boldsymbol{\Sigma}_{1,1}^{-1}(\mathbf{x}_1 - \boldsymbol{\mu}_1)$$

$$\boldsymbol{\Sigma}_{2|1} = \boldsymbol{\Sigma}_{2,2} - \boldsymbol{\Sigma}_{2,1}\boldsymbol{\Sigma}_{1,1}^{-1}\boldsymbol{\Sigma}_{1,2}$$

Gaussian Processes

A Gaussian Process (GP)

- Consider modelling some unknown function that maps $\mathbf{x} \in \mathbb{R}^D$ to $\mathbb{R}$
- In Week 2, we used models of the form $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$
- A Gaussian process model $f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ is a bit more complicated:

  1. For **any** set of M inputs $\mathbf{X} = \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M)}\right]^\top$ the function values $\mathbf{f} = \left[f(\mathbf{x}^{(1)}), f(\mathbf{x}^{(2)}), \ldots, f(\mathbf{x}^{(M)})\right]^\top$ are **<u>random variables</u>**
  2. These random variables have a **<u>multivariate Gaussian distribution</u>** $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

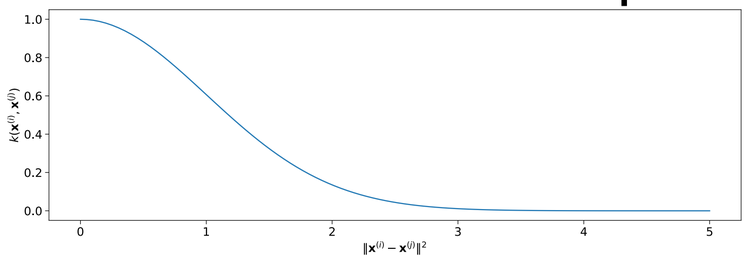
GPs are defined by their mean and kernel functions

- $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- $\boldsymbol{\mu} = [m(\mathbf{x}^{(1)}), m(\mathbf{x}^{(2)}), \ldots, m(\mathbf{x}^{(M)})]^\top$ where m is a user-supplied mean function
- $\boldsymbol{\Sigma}_{i,j} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ where k is a user-supplied kernel function

$$\begin{bmatrix} f(\mathbf{x}^{(1)}) \\ f(\mathbf{x}^{(2)}) \\ \vdots \\ f(\mathbf{x}^{(M)}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{x}^{(1)}) \\ m(\mathbf{x}^{(2)}) \\ \vdots \\ m(\mathbf{x}^{(M)}) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & k(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) & \ldots & k(\mathbf{x}^{(1)}, \mathbf{x}^{(M)}) \\ k(\mathbf{x}^{(2)}, \mathbf{x}^{(1)}) & k(\mathbf{x}^{(2)}, \mathbf{x}^{(2)}) & \ldots & k(\mathbf{x}^{(2)}, \mathbf{x}^{(M)}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}^{(M)}, \mathbf{x}^{(1)}) & k(\mathbf{x}^{(M)}, \mathbf{x}^{(2)}) & \ldots & k(\mathbf{x}^{(M)}, \mathbf{x}^{(M)}) \end{bmatrix} \right)$$

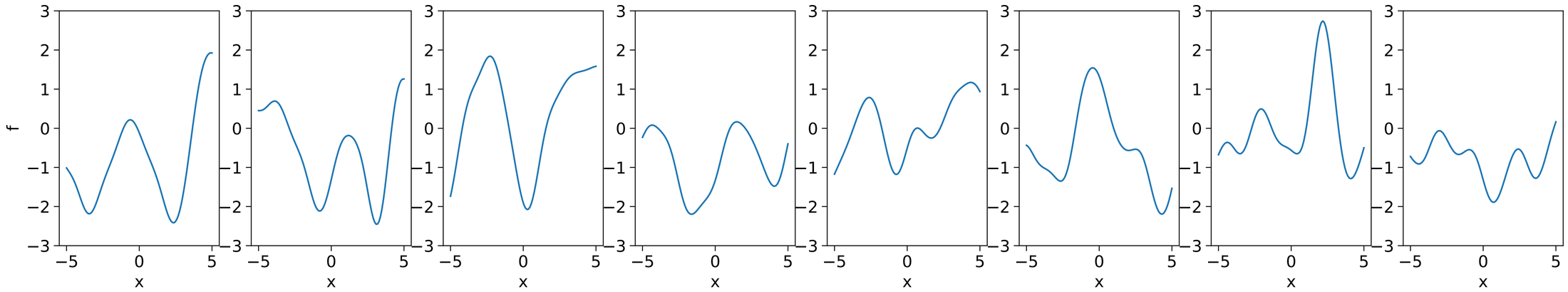
GP prior

- Our choice of mean and kernel function represent our a priori **assumptions** about what the function $f(\mathbf{x})$ should look like before we see any data
- Without any additional information it's reasonable to use $m(\mathbf{x}) = 0$
- The kernel $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ gives the covariance $\Sigma_{i,j}$ between $f(\mathbf{x}^{(i)})$ and $f(\mathbf{x}^{(j)})$
- It is reasonable to assume that the function values of points close together will be correlated and those of points further away will be less correlated
- We can embed this assumption using e.g. $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2}\right)$



Sampling from the GP prior (1D case)

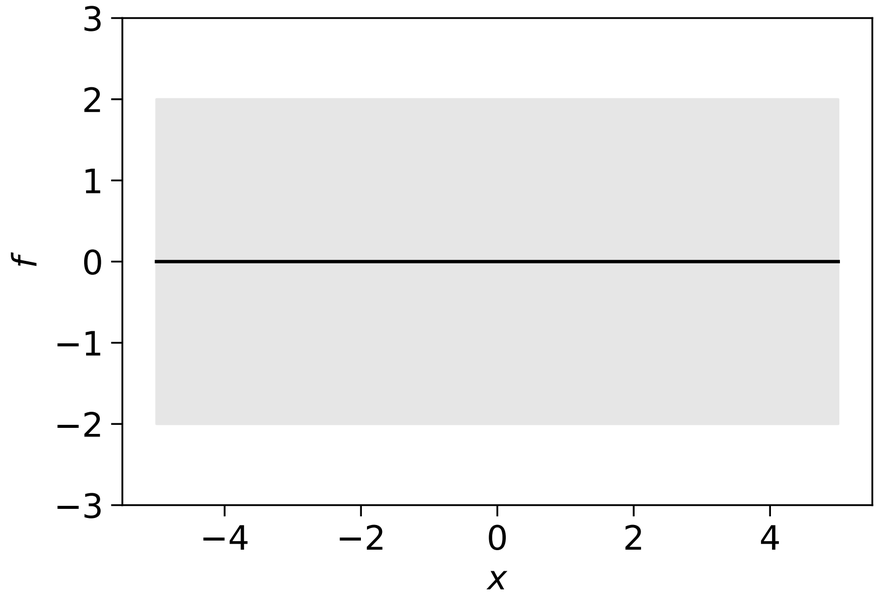
- Consider modelling a $\mathbb{R} \rightarrow \mathbb{R}$ mapping using $f(x) \sim GP(m(x), k(x, x'))$
- $m(x) = 0$ (so $\boldsymbol{\mu} = [0 \quad 0 \quad \ldots]^\top = \mathbf{0}$) and and $\boldsymbol{\Sigma}_{i,j} = k(x^{(i)}, x^{(j)}) = \exp\left(-\frac{(x^{(i)} - x^{(j)})^2}{2}\right)$
- $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ for any $\mathbf{X}$. This is our GP prior
- Let's use $\mathbf{X} = \left[-5, -4.9, -4.8, \ldots, +4.8, +4.9, +5\right]^\top$ and sample $\mathbf{f}$ vectors from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ to get <u>possible</u> functions from our GP prior



A distribution for each function value

- We can marginalise $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ to see what the distribution of function values is for each input x. It is $f(x) \sim \mathcal{N}(0,1) \;\; \forall x$ for our prior

- We can plot the mean μ, and shade $\mu \pm 2\sigma$ (the ~95% confidence interval) of $f(x)$ for each x but this isn't very exciting for $\mu = 0$ and $\sigma^2 = 1$



Conditioning a GP

- Consider some $\mathbf{X}$ and the distribution over function values at these points $p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}_{\mathbf{X}}, \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}})$
- Consider some other points $\mathbf{X}_*$ and the distribution $p(\mathbf{f}_* \mid \mathbf{X}_*) = \mathcal{N}(\mathbf{f}_*; \boldsymbol{\mu}_{\mathbf{X}_*}, \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}_*})$
- By the definition of a GP we can write $\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{\mathbf{X}} \\ \boldsymbol{\mu}_{\mathbf{X}_*} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}} & \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}_*} \\ \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}} & \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}_*} \end{bmatrix} \right)$
- Let's say we now find out the exact values of $\mathbf{f}$. We can condition $\mathbf{f}_*$ on these.
- $p(\mathbf{f}_* \mid \mathbf{f}, \mathbf{X}, \mathbf{X}_*) = \mathcal{N}(\mathbf{f}_*; \boldsymbol{\mu}_{\mathbf{X}_*\mid\mathbf{X}}, \boldsymbol{\Sigma}_{\mathbf{X}_*\mid\mathbf{X}})$ where

$$\boldsymbol{\mu}_{\mathbf{X}_*\mid\mathbf{X}} = \boldsymbol{\mu}_{\mathbf{X}_*} + \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}} \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}}^{-1} (\mathbf{f} - \boldsymbol{\mu}_{\mathbf{X}})$$

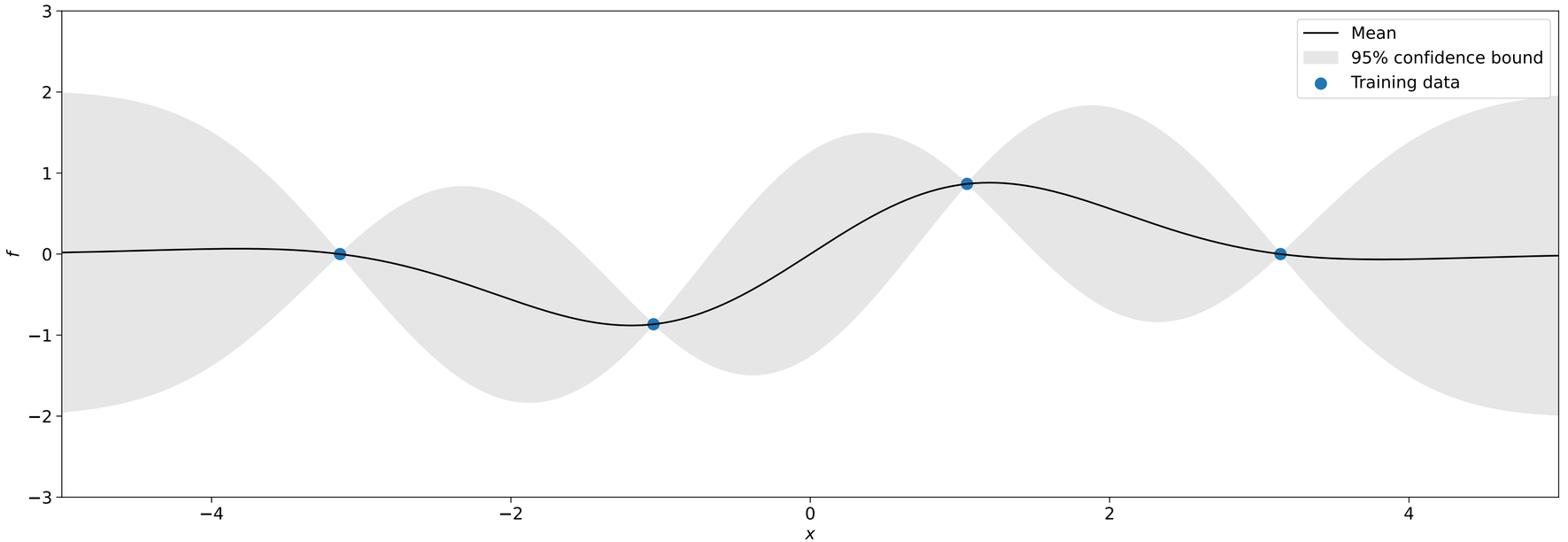
$$\boldsymbol{\Sigma}_{\mathbf{X}_*\mid\mathbf{X}} = \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}_*} - \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}} \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}}^{-1} \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}_*}$$

GPs in the context of ML

- We have $\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \boldsymbol{\mu}_{\mathbf{X}} \\ \boldsymbol{\mu}_{\mathbf{X}_*} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}} & \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}_*} \\ \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}} & \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}_*} \end{bmatrix})$ & $p(\mathbf{f}_* \mid \mathbf{f}, \mathbf{X}, \mathbf{X}_*) = \mathcal{N}(\mathbf{f}_*; \boldsymbol{\mu}_{\mathbf{X}_*|\mathbf{X}}, \boldsymbol{\Sigma}_{\mathbf{X}_*|\mathbf{X}})$
- Now suppose we are doing regression and have training data $\mathfrak{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ where $\mathbf{x} \in \mathbb{R}^D$ and $y \in \mathbb{R}$
- Let $\mathbf{X}$ be our training points $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(N)} \end{bmatrix}^\top$ and $\mathbf{f}$ be our targets $\mathbf{f} = \begin{bmatrix} y^{(1)} & y^{(2)} & \dots & y^{(N)} \end{bmatrix}^\top$
- We can make function predictions at new points $\mathbf{X}_*$ in light of our training data by looking at $p(\mathbf{f}_* \mid \mathbf{f}, \mathbf{X}, \mathbf{X}_*)$. This is our GP posterior

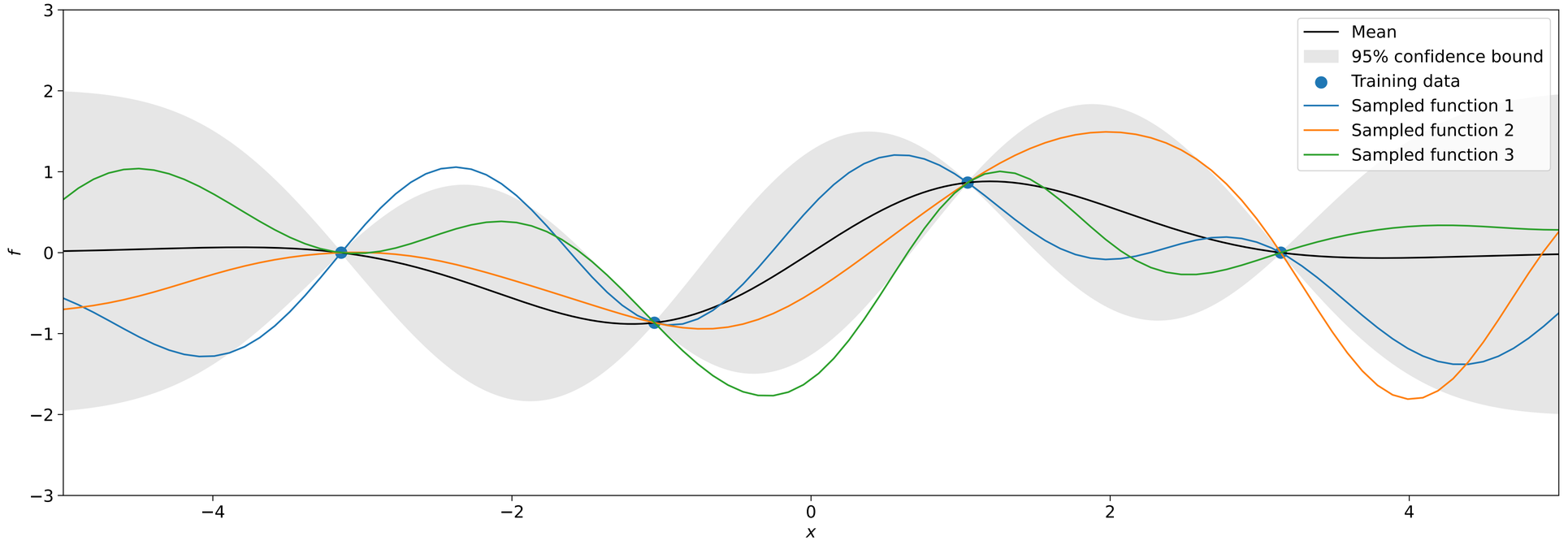
GP regression

- Marginalising $p(\mathbf{f}_* \,|\, \mathbf{f}, \mathbf{X}, \mathbf{X}_*)$ for some function value gives a Gaussian distribution
- The mean of that distribution can be used as a prediction for regression
- The variance quantifies how much uncertainty there is in that prediction



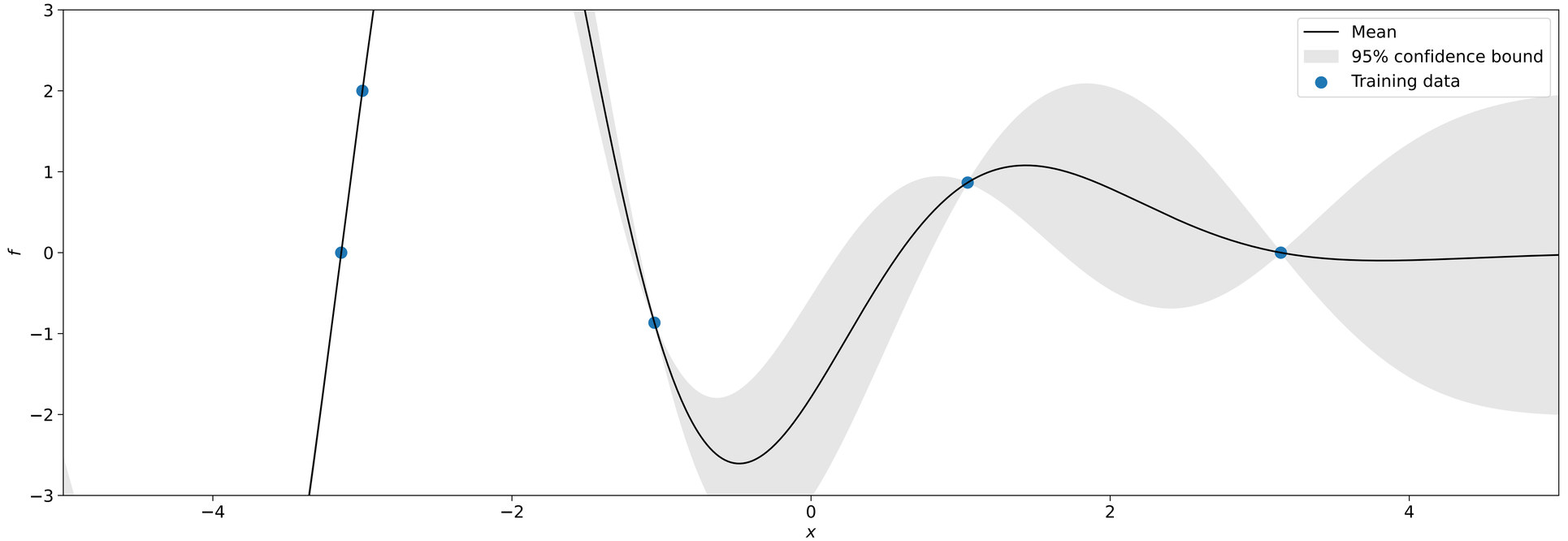
Sampling from the posterior

- $p(\mathbf{f}_* \mid \mathbf{f}, \mathbf{X}, \mathbf{X}_*)$ is just a multivariate Gaussian we can sample from to get possible functions
- Notice that all functions we sample interpolate the training points



Noisy measurements?

- Notice that the mean curve perfectly interpolates the training data (here, and on the previous slide), and that there is zero uncertainty when this happens
- This is a strong requirement, and we might need to accept that our training targets might be noisy :’(



Dealing with noisy measurements

- We can (waves hands) assume that our targets $\mathbf{y} = \begin{bmatrix} y^{(1)} & y^{(2)} & \dots & y^{(N)} \end{bmatrix}^\top$ suffer from additive Gaussian noise i.e. $y = f(\mathbf{x}) + \mathcal{N}(0, \sigma_y^2)$
- For a GP with mean function zero, we have $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{X},\mathbf{X}})$. It follows that $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I})$
- This gives us $\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} \mathbf{\Sigma}_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I} & \mathbf{\Sigma}_{\mathbf{X},\mathbf{X}_*} \\ \mathbf{\Sigma}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{\Sigma}_{\mathbf{X}_*,\mathbf{X}_*} \end{bmatrix})$

From hereon we assume $m(\mathbf{x}) = 0$. This is quite a common assumption. See https://stats.stackexchange.com/questions/63251/what-justifies-the-zero-mean-assumption-for-gaussian-processes for some thoughts on the matter.

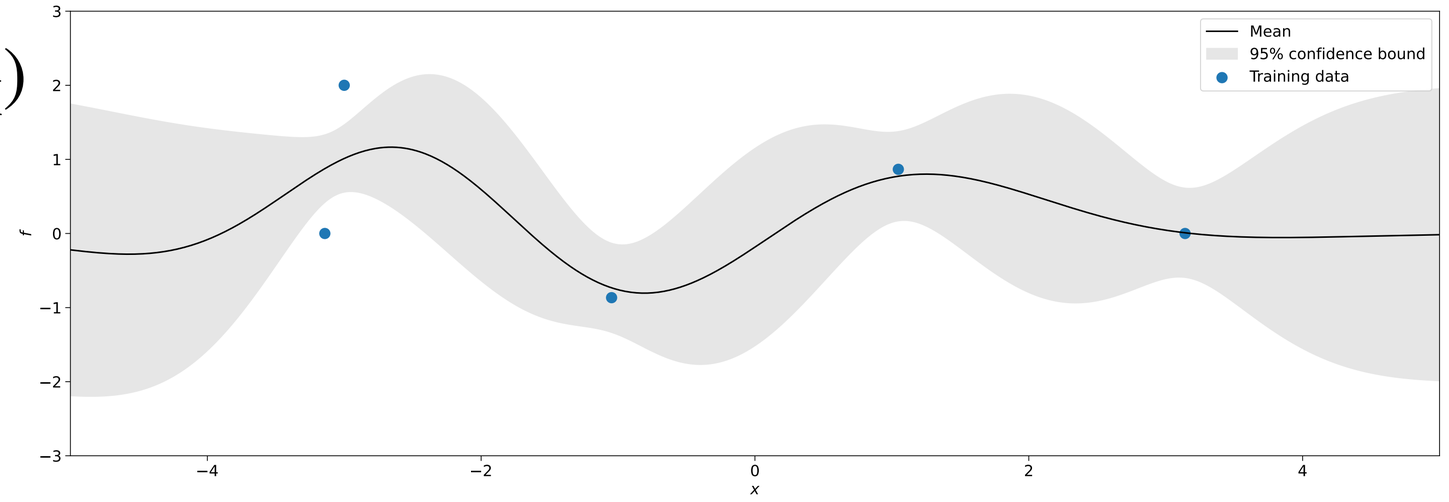
Conditioning a GP subject to noisy measurements

- $$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I} & \boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}_*} \\ \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}} & \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}_*} \end{bmatrix})$$
- We can use the conditioning formula to get a GP posterior subject to noisy measurements

$$p(\mathbf{f}_* \mid \mathbf{y}, \mathbf{X}, \mathbf{X}_*) = \mathcal{N}(\mathbf{f}_*; \boldsymbol{\mu}_{\mathbf{X}_*|\mathbf{X}}, \boldsymbol{\Sigma}_{\mathbf{X}_*|\mathbf{X}})$$

$$\boldsymbol{\mu}_{\mathbf{X}_*|\mathbf{X}} = \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}}(\Sigma_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I})^{-1}\mathbf{y}$$

$$\boldsymbol{\Sigma}_{\mathbf{X}_*|\mathbf{X}} = \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}_*} - \boldsymbol{\Sigma}_{\mathbf{X}_*,\mathbf{X}}(\Sigma_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I})^{-1}\boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}_*}$$



Kernels

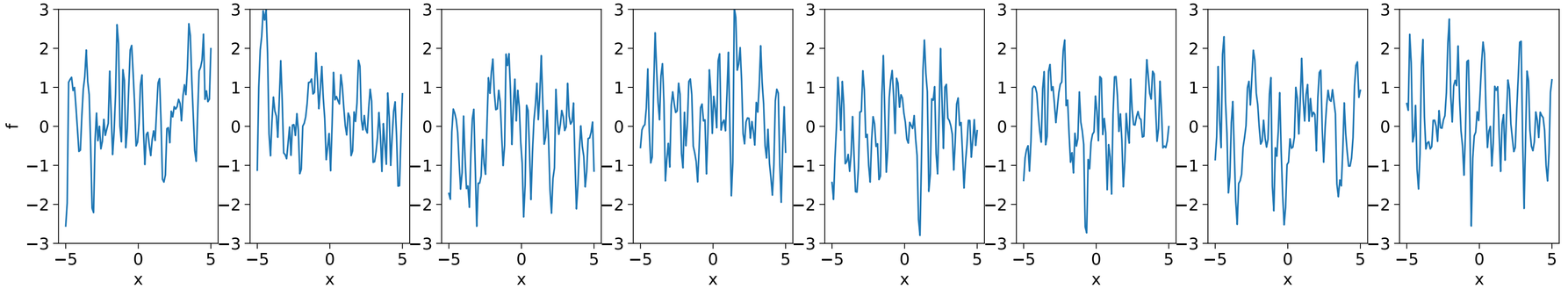
- We have used $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2}\right)$
- This is a simplified version of the RBF or *squared exponential* kernel $k_{SE}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2\ell^2}\right)$
- Notice that it has a hyperparameter ℓ
- The kernel used must lead to a positive semi-definite covariance matrix

The length scale of the RBF kernel

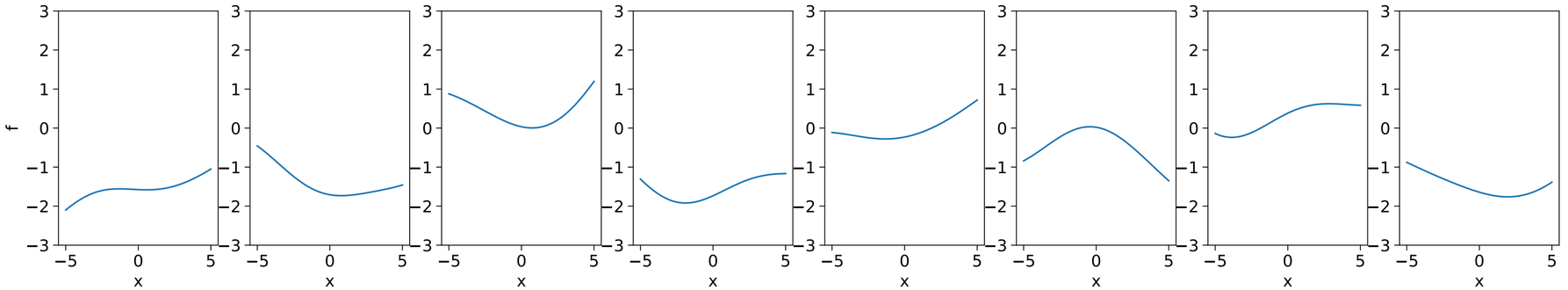
- $k_{SE}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2\ell^2}\right)$

- ℓ is known as the lengthscale and determines how wiggly functions drawn from the GP are

$\ell = 0.1$



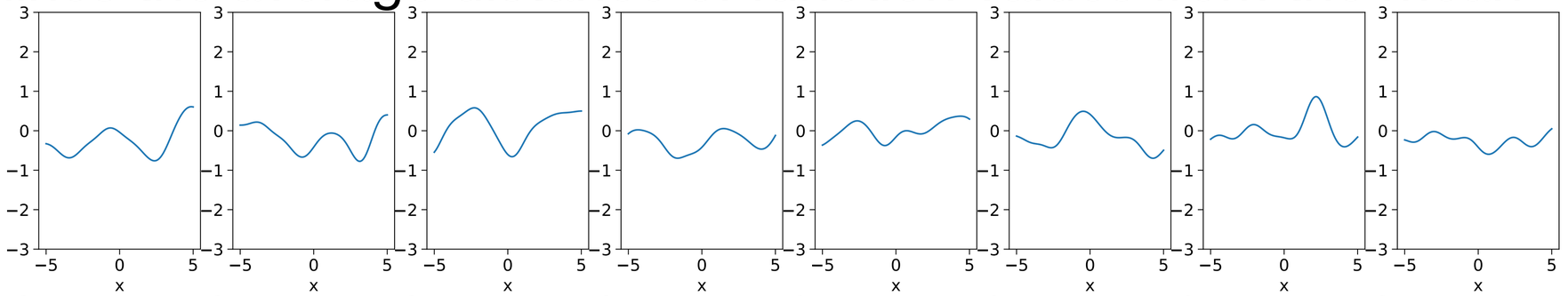
$\ell = 5$



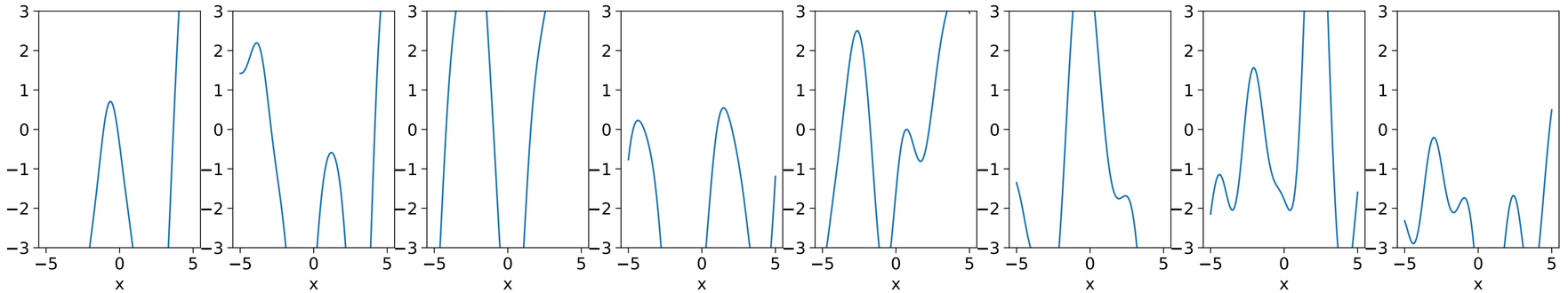
Scaling the RBF kernel

- It is common to multiply kernels by a hyperparameter σ^2 (the output variance)
- $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sigma^2 \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2\ell^2}\right)$
- σ^2 determines the average distance of function values from the mean

$\sigma^2 = 0.1$



$\sigma^2 = 10$



Kernels are important

- The choice of kernel is the most important factor for determining the behaviour of a GP
- It is us providing our assumptions about the problem!

Predicting house price from location:

Prices will correlate for houses that are near to each other.

RBF kernel makes sense

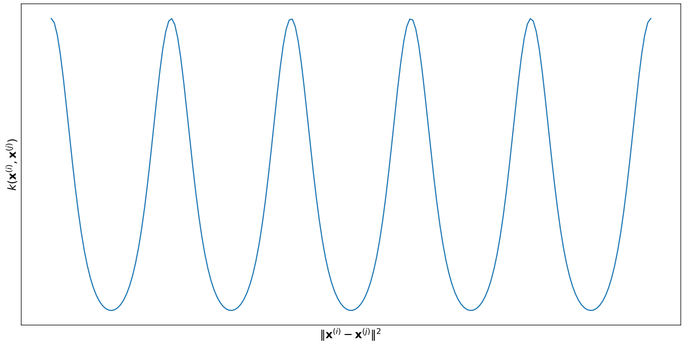
Predicting the temperature given the date:

There is seasonality. December one year correlates to December the next even though they are far away!

We need a periodic kernel

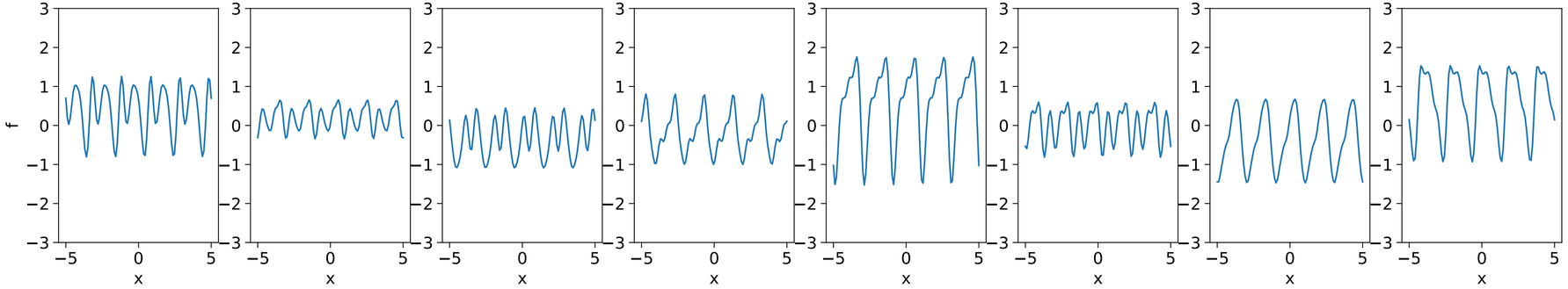
The period kernel (Exp-Sine-Squared)

- $k_P(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{2\sin^2(\frac{\pi\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|}{p})}{\ell^2}\right)$

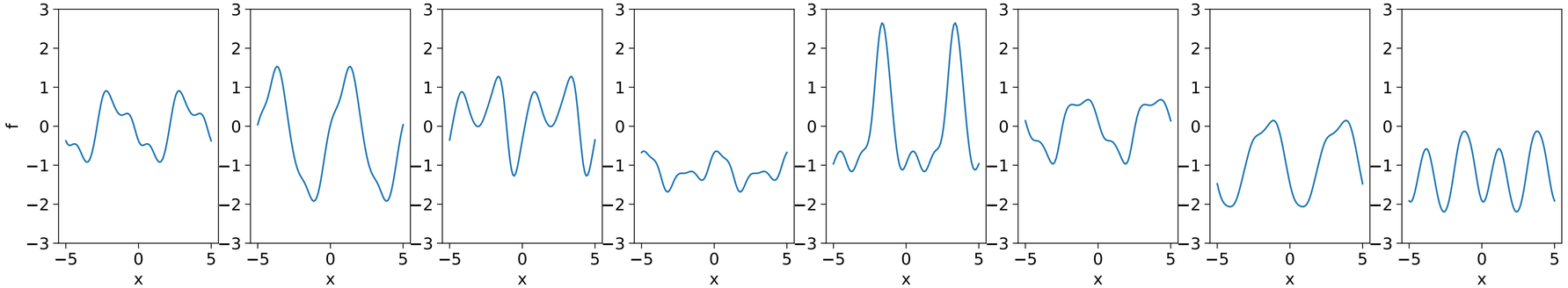


- ℓ behaves the same as for the RBF kernel. p is the *periodicity*

$p = 2$



$p = 5$



There are lots of other kernels!

- See https://www.cs.toronto.edu/~duvenaud/cookbook/ for all the kernels you could ever care to know about
- In the above, all kernels have a scaling factor
- Sklearn kernels **don't** by default, just be aware of this!

Hyperparameters again!

- Kernels have hyperparameters
- We could do a grid search to find the best hyperparameters to minimise e.g. MSE on a validation set, but this would be slow. Is there a faster way? (Yes)
- Let's find the hyperparameters that maximise the likelihood of our targets, given our data: $p(\mathbf{y} \mid \mathbf{X})$
- With noisy measurements we have defined $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I})$
- In the expression above, the dependence on $\mathbf{X}$ was implicit. Let's write the whole thing explicitly: $p(\mathbf{y} \mid \mathbf{X}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{\Sigma}_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I})$

Optimisation

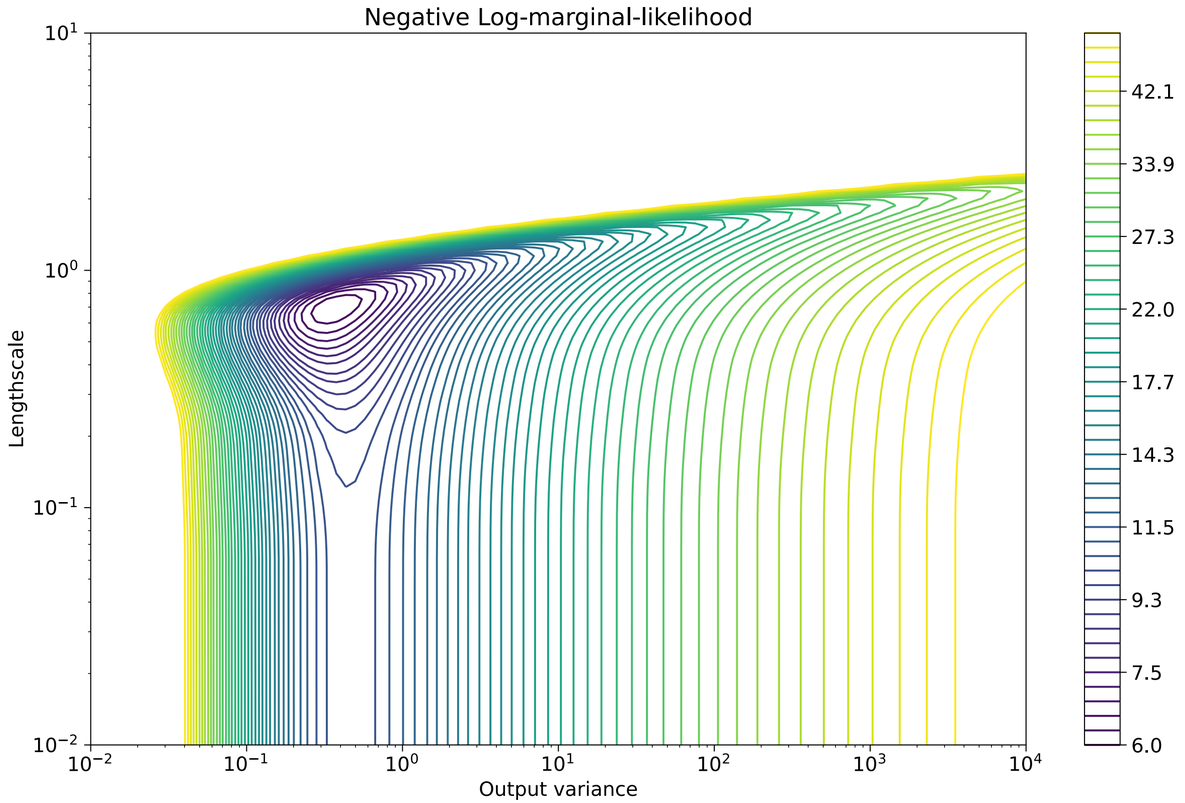
- We have $p(\mathbf{y} \mid \mathbf{X}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{\Sigma}_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I})$
- Collecting our hyperparameters into a vector $\boldsymbol{\theta}$, we want to solve $\underset{\boldsymbol{\theta}}{\text{maximise}} \;\; p(\mathbf{y} \mid \mathbf{X})$
- This is equivalent to solving $\underset{\boldsymbol{\theta}}{\text{minimise}} \;\; -\log p(\mathbf{y} \mid \mathbf{X})$
- This can be achieved using a gradient-based optimiser for differentiable kernels

$$\log p(\mathbf{y} \mid \mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{\Sigma}_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log |\, \mathbf{\Sigma}_{\mathbf{X},\mathbf{X}} + \sigma_y^2 \mathbf{I} \,| - \frac{N}{2}\log 2\pi$$

This quantity is known as the log **marginal** likelihood because we've integrated over $\mathbf{f}$

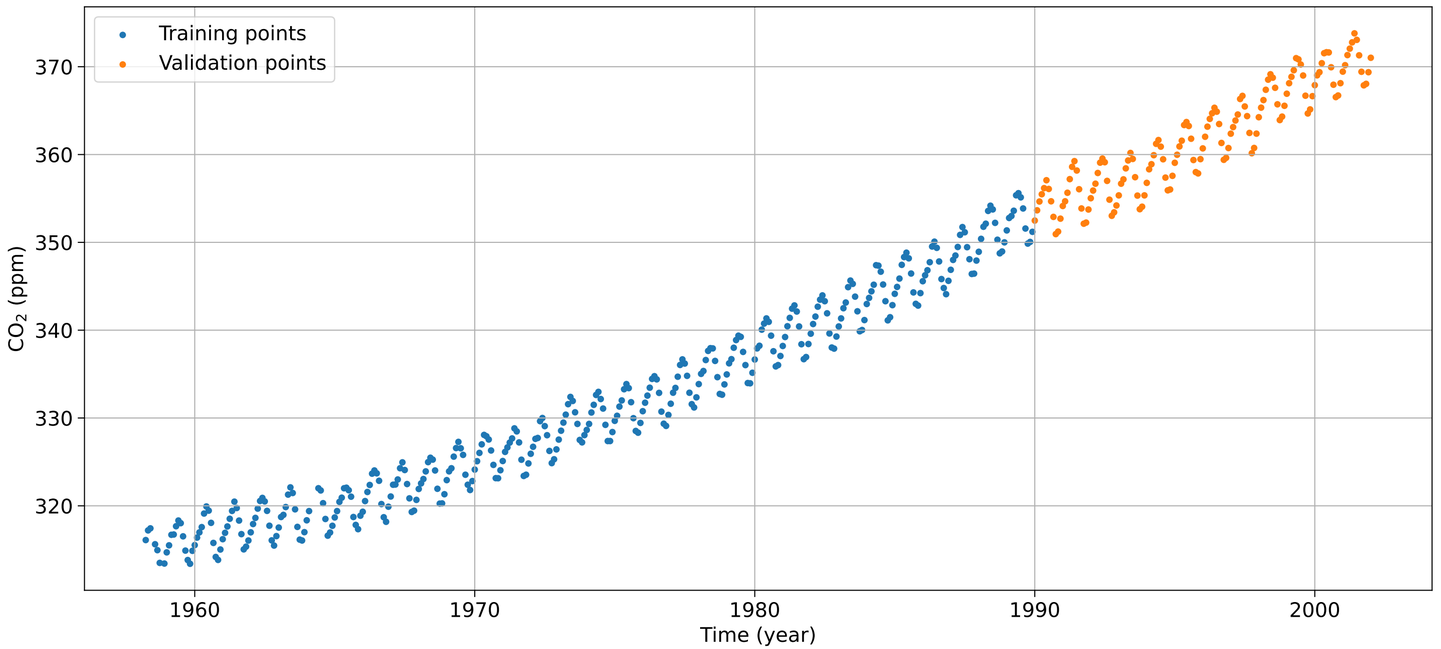
Minimising the negative log-marginal-likelihood

- When you fit a GP to data in sklearn, an optimiser is used to find the kernel hyperparameters that minimise the negative log-marginal-likelihood on **train**
- There may be local minima…
- We can run it multiple times from different initial positions
- Minimising this objective is fast but not necessarily best



GP example

- Kernels can be combined to form new kernels
- We'll look at a famous GP example: Predicting monthly CO_2 concentrations (in ppm) from the Mauna Loa Observatory in Hawaii



https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_co2.html#sphx-glr-auto-examples-gaussian-process-plot-gpr-co2-py

GP example

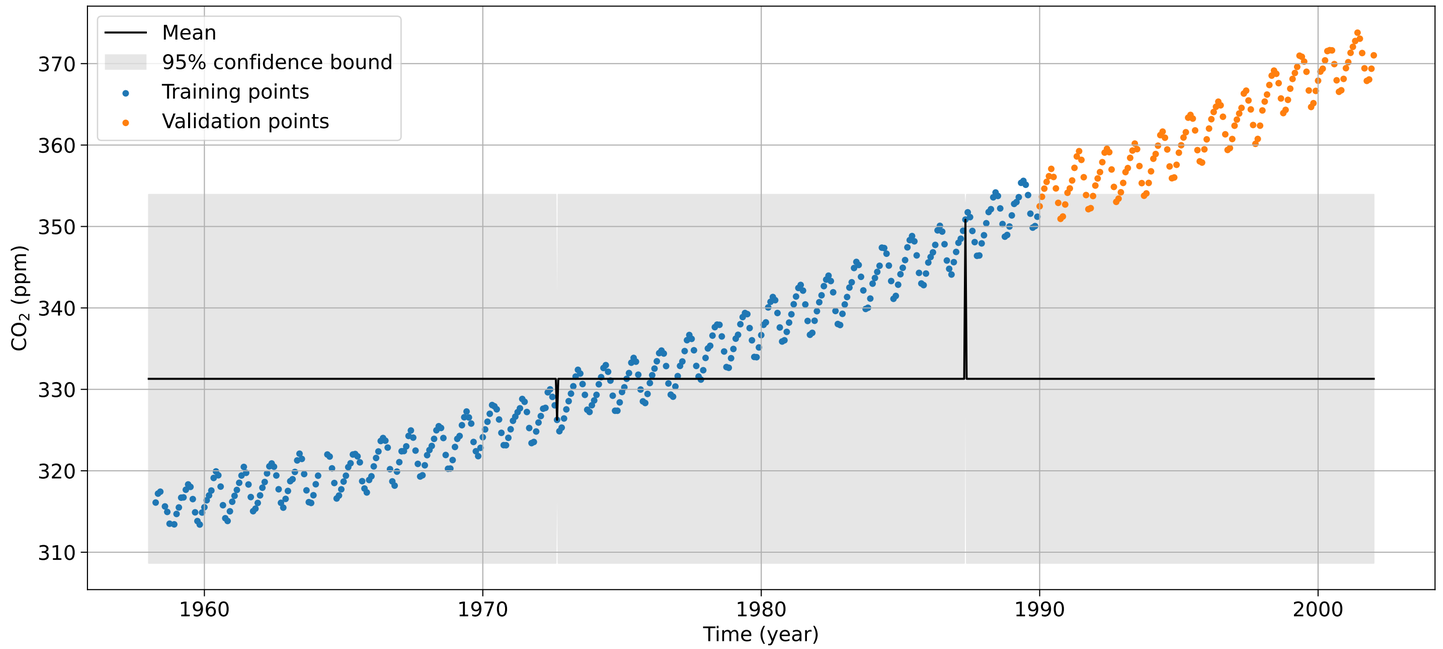
- First, we'll make our data zero mean (but show the true value in the plot)
- Let's naively assume there is minimal measurement noise ($\sigma_y = 10^{-3}$)
- Points close to each other are correlated so let's use an RBF kernel

$$k = \sigma^2 k_{RBF}(\ell)$$

$$\sigma = 0.999$$

$$\ell = 10^{-5}$$

I think it broke.



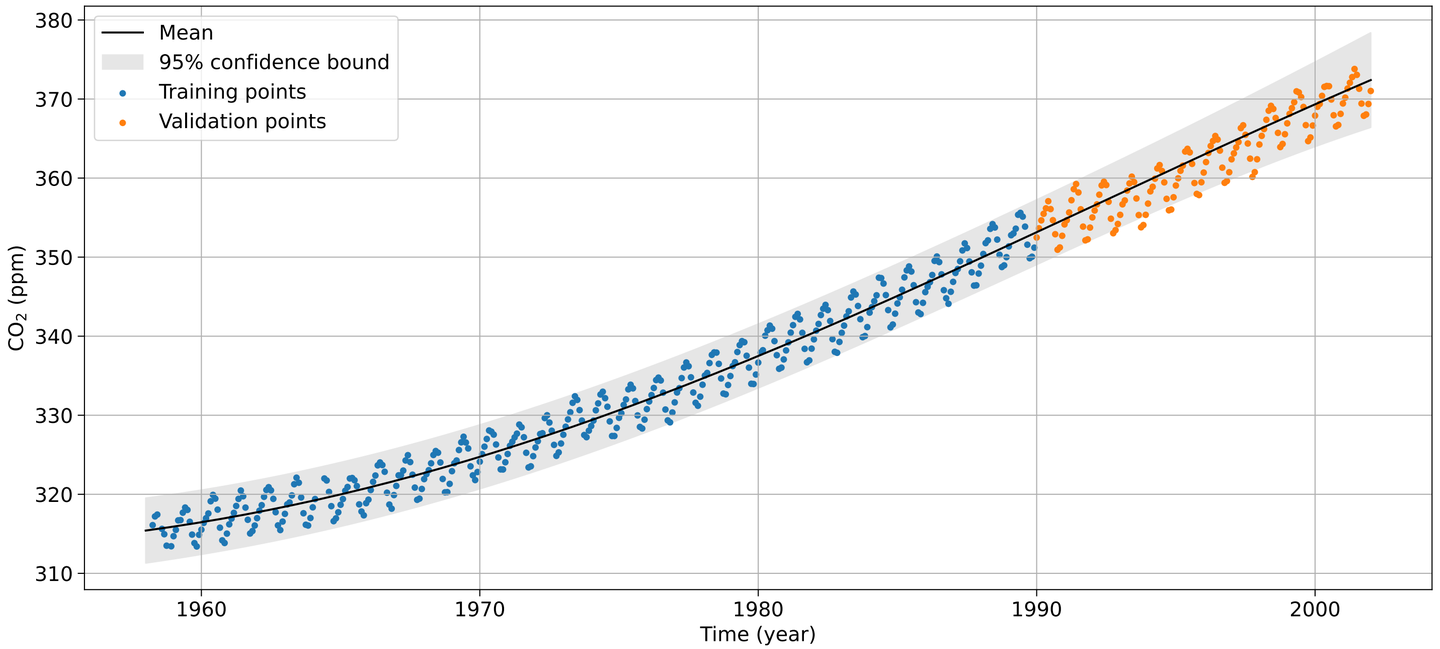
GP example

- There is going to be measurement noise, but instead of guessing we can add a white noise kernel $k_{noise}(x, x') = \sigma_n^2 \mathbb{I}(x = x')$ and fit σ_n as a hyperparameter

$$k = \sigma^2 k_{RBF}(\ell) + k_{noise}(\sigma_n)$$

$$\sigma = 3.92$$
$$\ell = 46$$
$$\sigma_n = 0.0319$$

Better. But it doesn't wiggle.



GP example

- Our model can deal with the rising trend but not the seasonal variation
- Let's add a periodic kernel and keep the period fixed as 1 year because we know that this is the frequency!
- I also fixed the noise to stop it overfitting (GPs are fairly hacky)

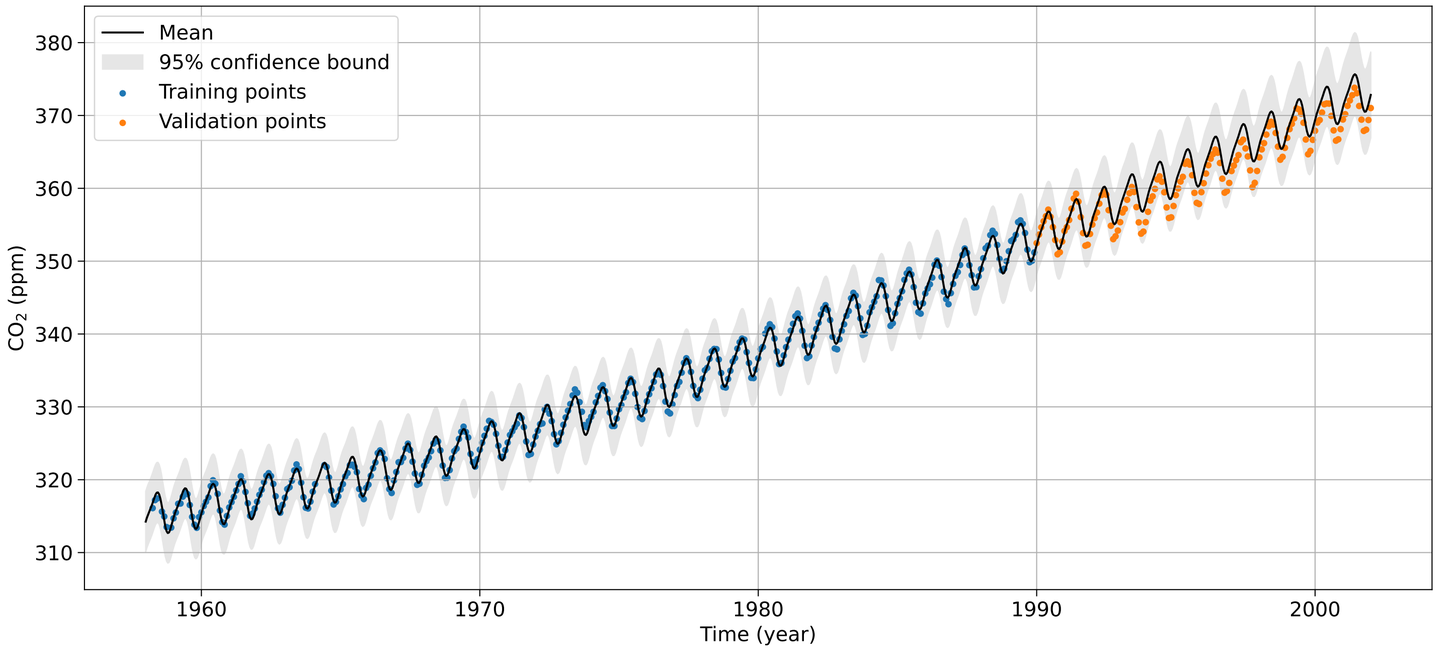
$$k = \sigma^2_{RBF} k_{RBF}(\ell_{RBF}) + k_{noise}(\sigma_n = 3.92) + \sigma^2_P k_P(\ell_P = 1)$$

$$\sigma_{RBF} = 4.6$$

$$\ell_{RBF} = 50.5$$

$$\sigma_P = 0.485$$

That'll do



The pros and cons of Gaussian processes

- **Pro:** They give you confidence intervals
- **Pro:** They work well in the low-data setting
- **Pro:** Providing a kernel can be more intuitive for baking in assumptions about a problem than specifying a functional form from inputs to outputs
- **Con:** Fitting a GP involves inverting a matrix. This is very expensive when there is lots of data
- **Con:** They assume Gaussian noise on the measurements, which might not be true
- **Con:** Finding the right kernel can be tricky

Summary

- You have revised random variables and Gaussian pdfs
- You have learnt that Gaussian processes (GPs) model function values as random variables that have a Multivariate Gaussian distribution
- You have seen how this distribution is defined in terms of a mean function (which is usually zero) and a kernel function
- You have learnt how to condition a GP on training data for prediction
- You have seen that kernels have hyperparameters which affect their behaviour
- You have seen that these can be optimised by maximising the marginal likelihood

Questions

Q1

Frank wants to model an unknown function $g : \mathbb{R} \to \mathbb{R}$. He uses a Gaussian process prior:

$$f(x) \sim GP(0, 4 \exp\left(-\frac{(x - x')^2}{2}\right))$$

State the distributions for $f(5)$ and $f(4)$. What is the Pearson correlation coefficient between these random variables?

Q2

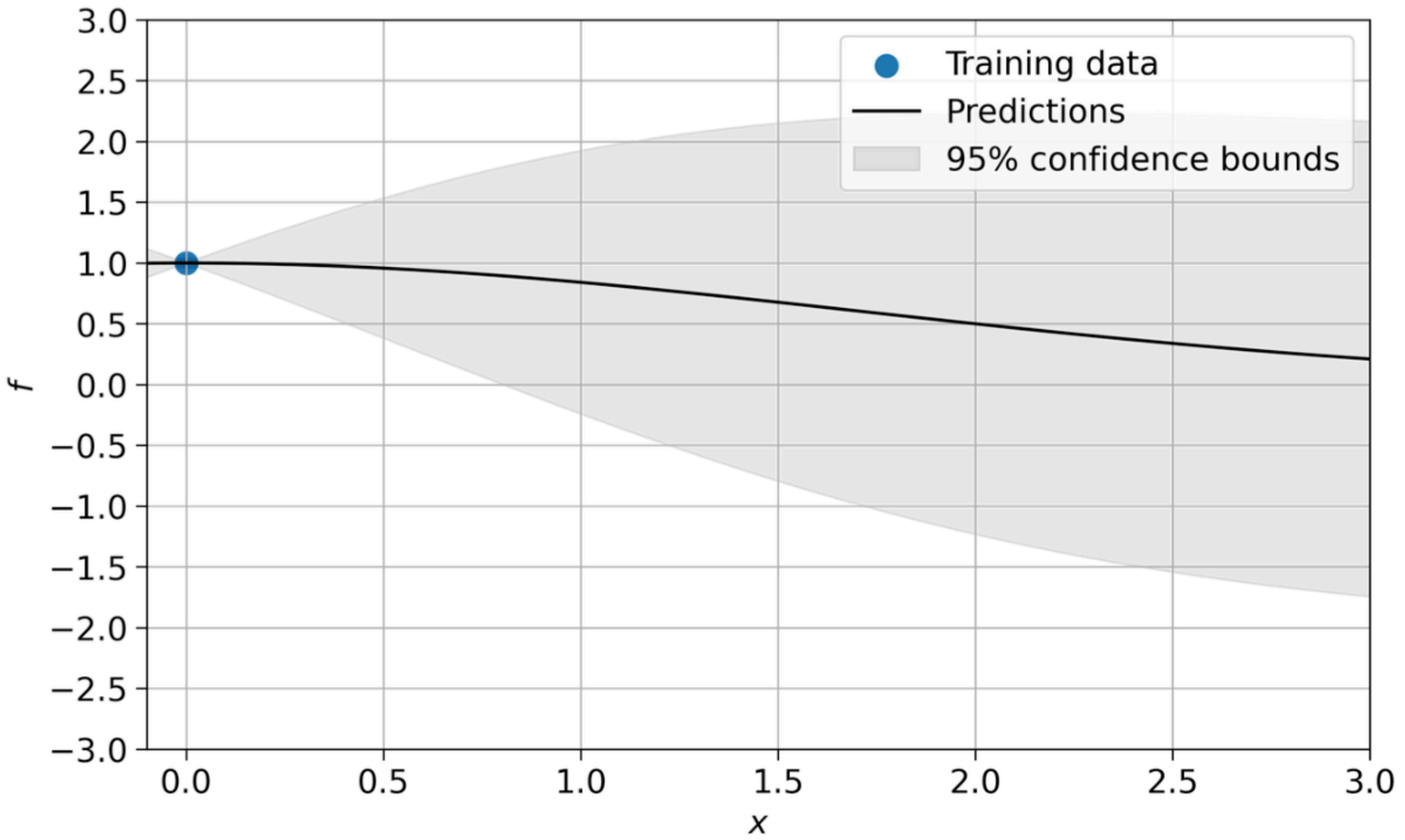
We have a Gaussian process (GP) prior

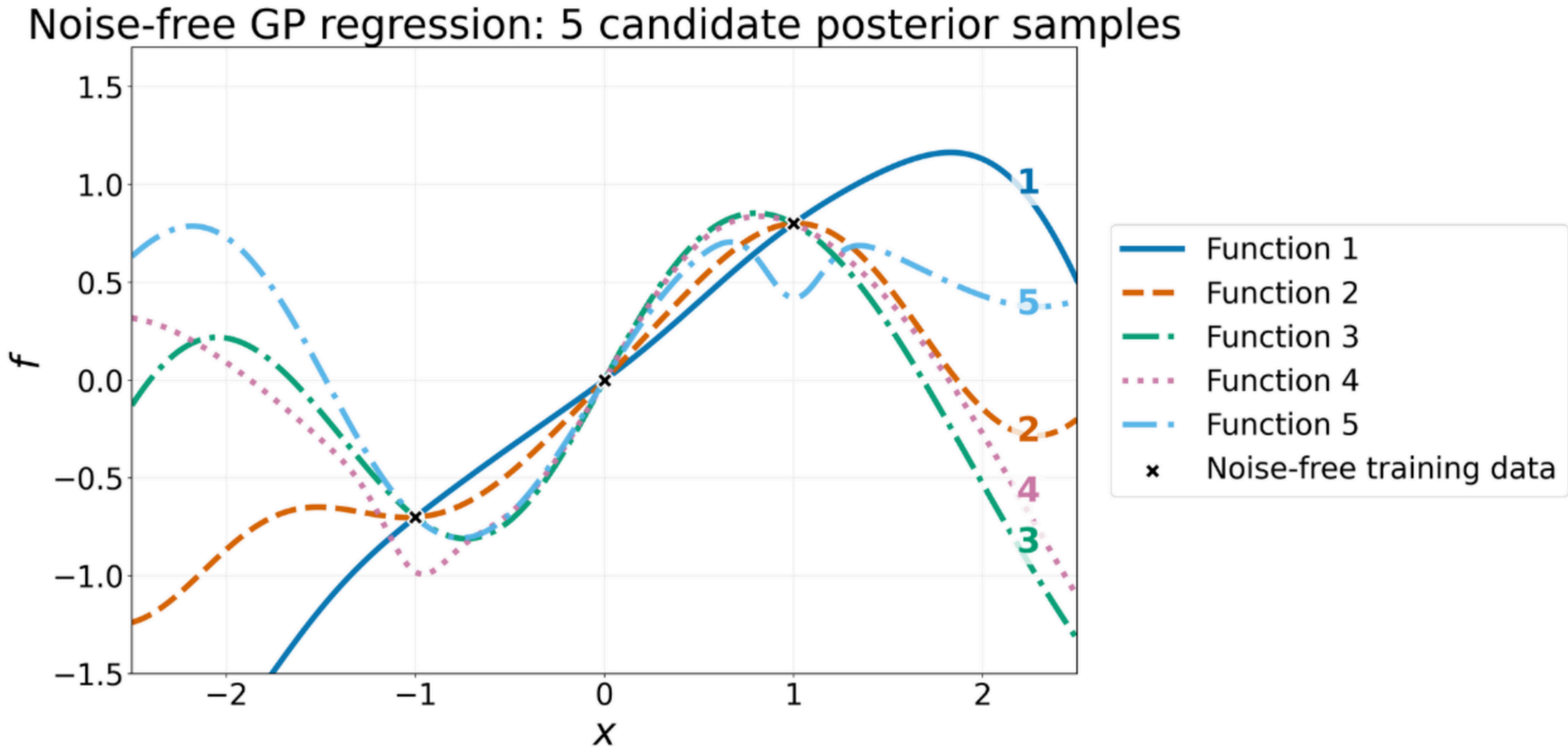
$$(x) \sim GP(0, \exp(-\frac{(x-x')^2}{2\ell^2}))$$

where ℓ is fixed.

The figure shows the corresponding GP posterior if we condition on a single data point.

What is the value of ℓ? Assume there is no measurement noise.



Q3

Which of the five functions in the plot cannot be correct posterior samples for a noise-free GP regression model?