Gaussian Processes — a brief introduction

Carl Edward Rasmussen

Ellis 2023 Cambridge Probabilistic Machine Learning Summer School

July 17-22nd, 2023

Outline

- why?
- defining Gaussian Processes
- learning and inference (1 slide)
- practice: hyperparameters
- Occam's Razor and the marginal likelihood
- covariance functions
- conclusions

Gaussian Processes: what and why?

Gaussian Processes (GPs) marry two of the most ubiqutous and useful concepts in science, engineering and modelling: probability theory and functions.

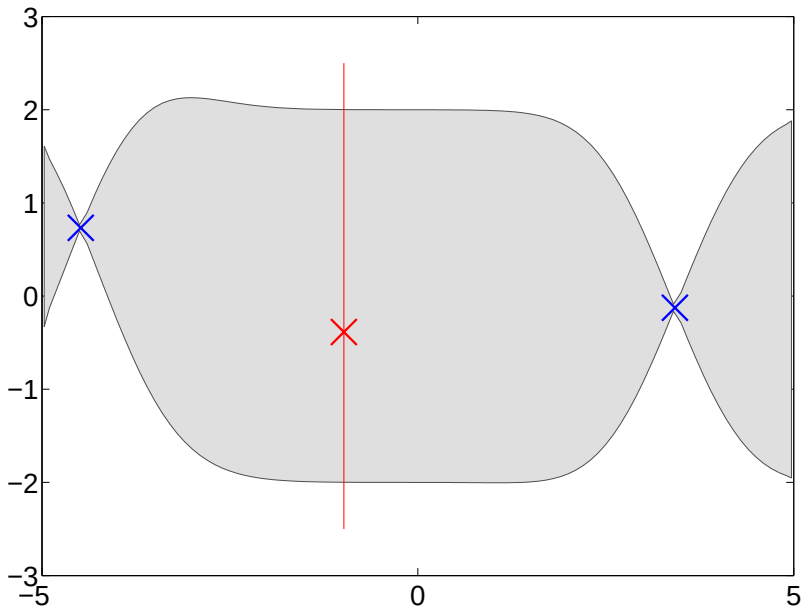
GPs **are** probability distributions over functions.

- GPs are the only practical class of probability distributions over functions
- GPs fit naturally within the Bayesian inference.
- The GP framework is *principled*, *practical* and *powerful*.

Distribution over Functions

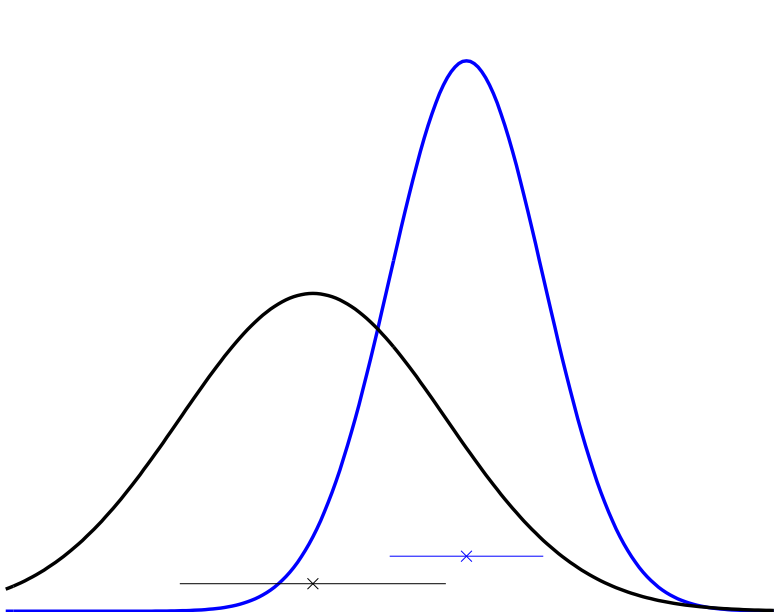
Key idea: use a separate random variable to represent that value of the function $f(x)$ for each possible input x.

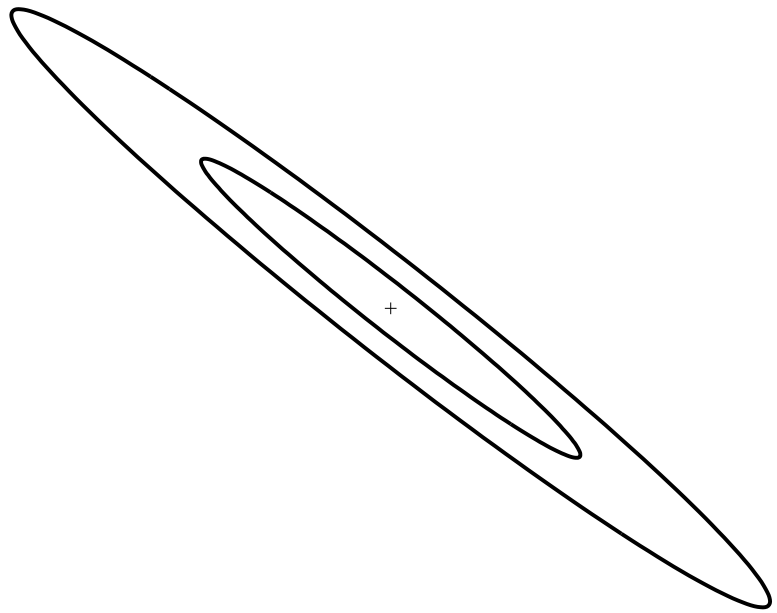
I will use plots like this, to illustrate (marginal) distributions over functions:



The function value at a specific input is characterised by a Gaussian.

The Gaussian Distribution





The univariate Gaussian distribution is given by

$$p(x|\mu,\sigma^2) \;=\; \mathcal{N}(\mu,\sigma^2) \;=\; (2\pi\sigma^2)^{-1/2}\exp\Big(-\frac{1}{2\sigma^2}(x-\mu)^2\Big)$$

The multivariate Gaussian distribution for D-dimensional vectors is given by

$$p(\mathbf{x}|\boldsymbol{\mu},\Sigma) \;=\; \mathcal{N}(\boldsymbol{\mu},\Sigma) \;=\; (2\pi)^{-D/2}|\Sigma|^{-1/2}\exp\Big(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big)$$

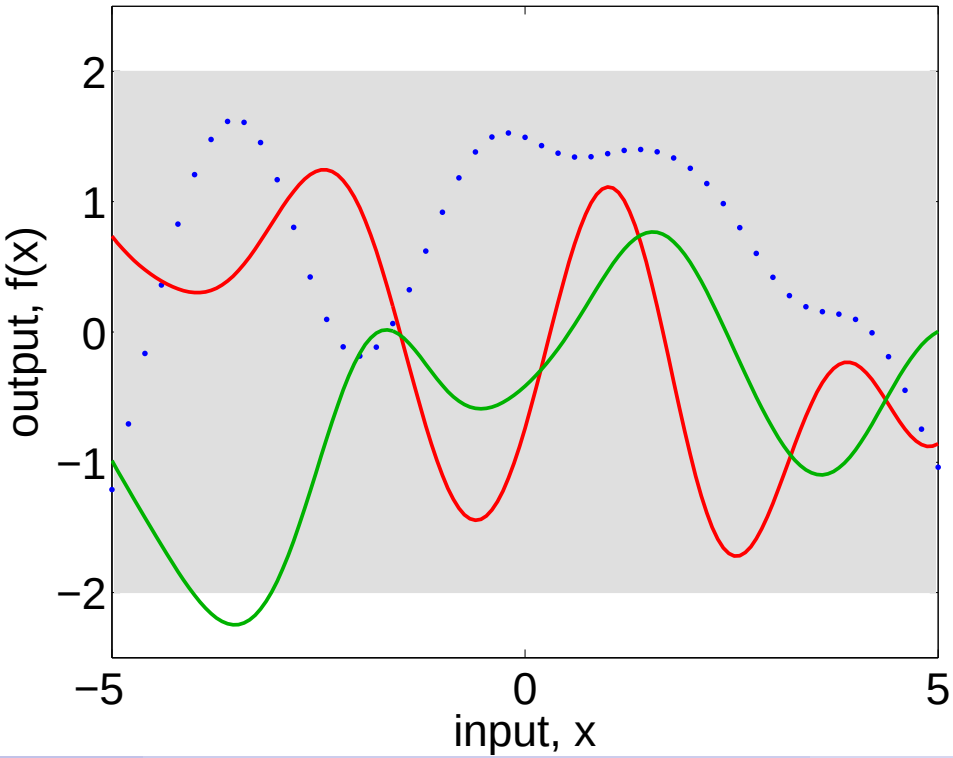
where $\boldsymbol{\mu}$ is the mean vector and Σ the covariance matrix.

From single to multiple function values

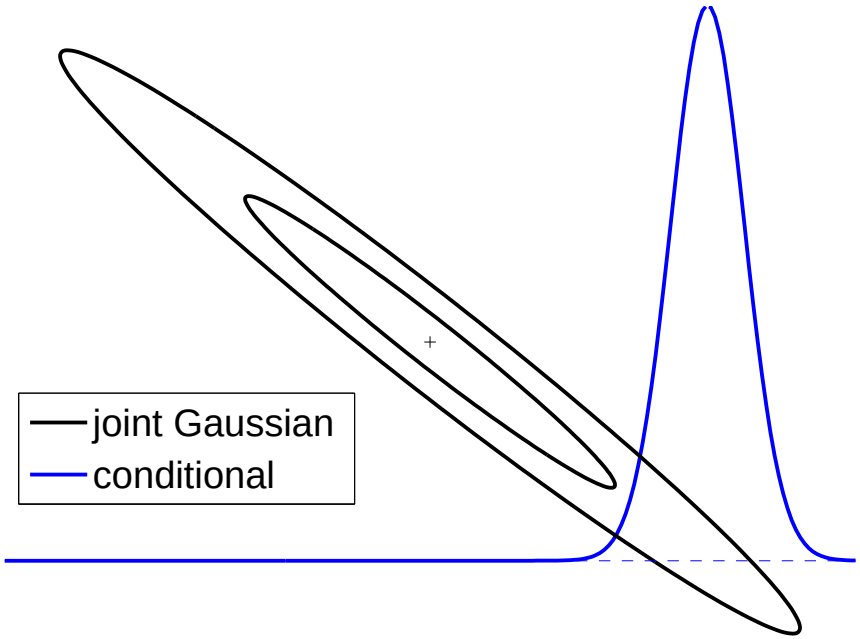
How do we generalize the specification of the value at a single input to multiple function values?

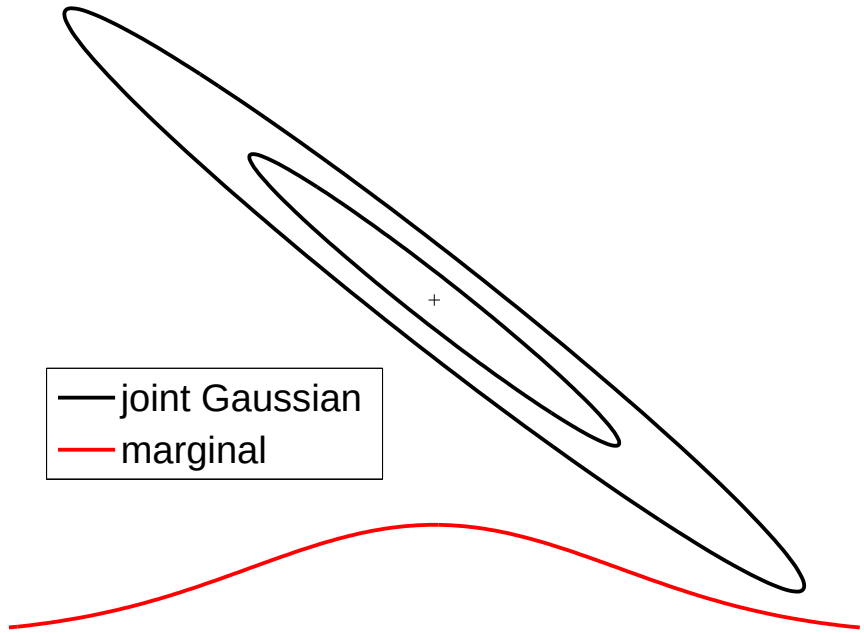
You might think you simply repeat the specification of mean μ and variance σ^2 for each possible input.

That's *almost* right, but not quite; the problem is the distinction between marginals and joints.



Conditionals and Marginals of a Gaussian, pictorial





Both the conditionals $p(x|y)$ and the marginals $p(x)$ of a joint Gaussian $p(x, y)$ are again Gaussian.

Conditionals and Marginals of a Gaussian, algebra

If $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian

$$p(\mathbf{x},\mathbf{y}) \;=\; p\Big(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\Big) \;=\; \mathcal{N}\Big(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix},\; \begin{bmatrix}A & B\\ B^\top & C\end{bmatrix}\Big),$$

we get the marginal distribution of $\mathbf{x}$, $p(\mathbf{x})$ by

$$p(\mathbf{x},\mathbf{y}) \;=\; \mathcal{N}\Big(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix},\; \begin{bmatrix}A & B\\ B^\top & C\end{bmatrix}\Big) \;\Longrightarrow\; p(\mathbf{x}) \;=\; \mathcal{N}(\mathbf{a},\; A),$$

and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ by

$$p(\mathbf{x},\mathbf{y}) = \mathcal{N}\Big(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix},\; \begin{bmatrix}A & B\\ B^\top & C\end{bmatrix}\Big) \;\Longrightarrow\; p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{a}+BC^{-1}(\mathbf{y}-\mathbf{b}),\; A-BC^{-1}B^\top),$$

where $\mathbf{x}$ and $\mathbf{y}$ can be scalars or vectors.

From single to multiple and to infinitely many

For the value of the function $f_1 = f(x_1)$ at a single location x_1 we use a scalar Gaussian $f_1 \sim \mathcal{N}(\mu, \sigma^2)$.

For the joint function $\mathbf{f}$ values at two locations x_1, x_2 a multivariate Gaussian $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$

etc

For the joint distribution for the entire function f at all input locations, we use a Gaussian Process $f \sim \mathcal{N}(m, k)$.

Here, f, m and k are *functions*.

A function $\simeq$ infinitely long vector. The *index set* into a vector are $1, 2, \ldots D$, the index set into a function $f(x)$ are the inputs x.

The *mean function* $m(x)$ is a function of a single argument x, whereas the *covariance function* $k(x, x')$ is a function of two arguments.

What is a Gaussian Process?

Definition: *a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.* □

We write

$$f \sim \mathcal{N}(m, k) \tag{1}$$

which is fully specified by it's mean function m and covariance function k.

The only *meaning* we assign to the GP is that for any finite set of inputs $\mathbf{x}$, the corresponding

$$\mathbf{f} = f(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu} = m(\mathbf{x}),\ \Sigma = k(\mathbf{x}, \mathbf{x})). \tag{2}$$

The covariance function must be positive definite.

Random functions from a Gaussian Process

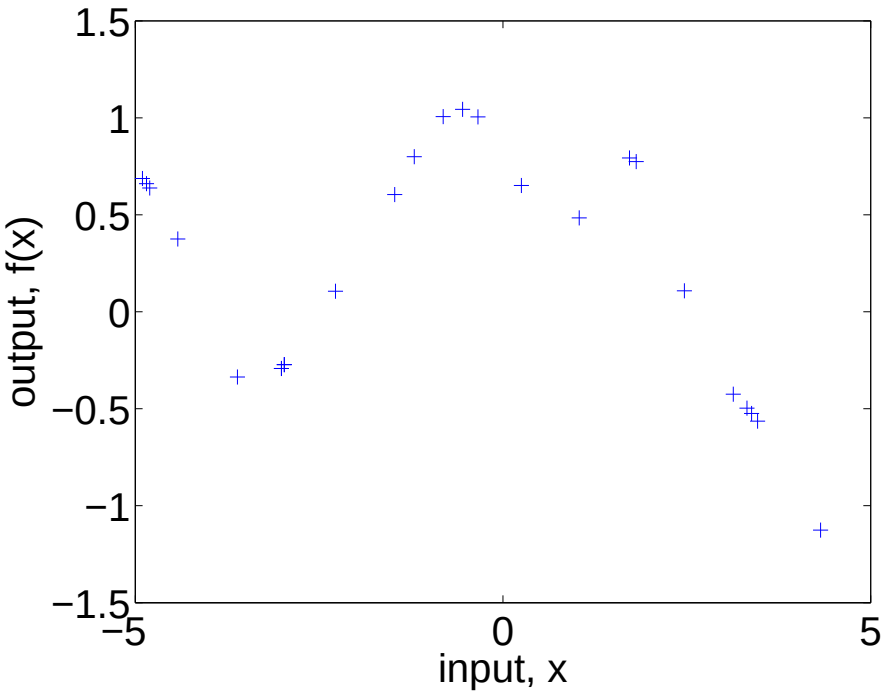
Example one dimensional Gaussian process:

$$f \sim \mathcal{N}(m, k), \quad \text{where } m(x) = 0, \quad \text{and } k(x, x') = \exp(-\tfrac{1}{2}(x - x')^2).$$

To get an indication of what this distribution over functions looks like, focus on a finite subset of function values $\mathbf{f} = (f(x_1), f(x_2), \ldots, f(x_N))^\top$, for which

$$\mathbf{f} \sim \mathcal{N}(0, \Sigma), \quad \text{where } \Sigma_{ij} = k(x_i, x_j).$$

Draw a random value of $\mathbf{f}$ from the distribution as a function of the corresponding x values



Joint Generation

To generate a random sample from a D dimensional joint Gaussian with covariance matrix K and mean vector $\mathbf{m}$: (in octave or matlab)

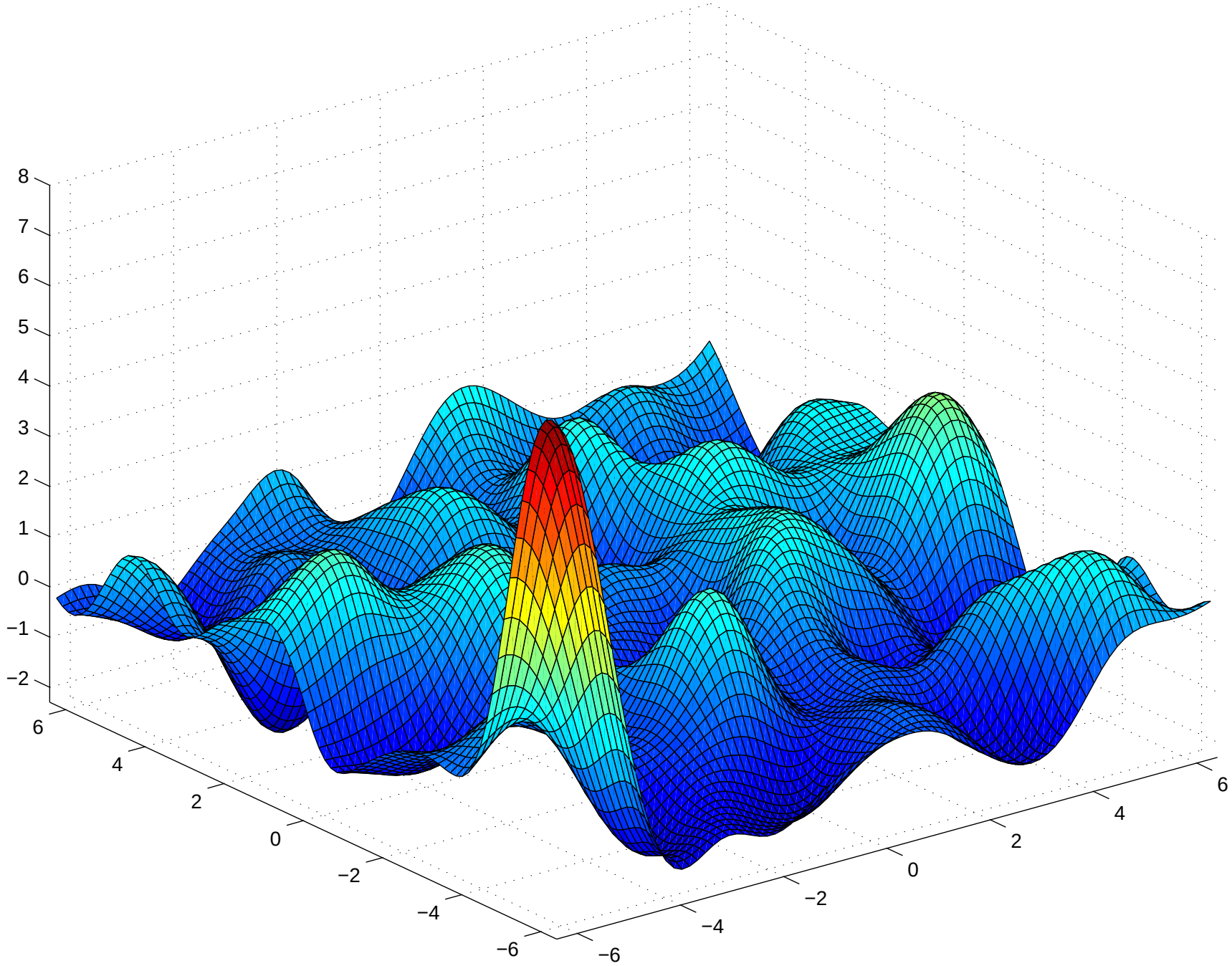
```
z = randn(D,1);
y = chol(K)'*z + m;
```

where `chol` is the Cholesky factor R such that $R^\top R = K$.

Thus, the covariance of $\mathbf{y}$ is:

$$\mathbb{E}[(\mathbf{y}-\mathbf{m})(\mathbf{y}-\mathbf{m})^\top] = \mathbb{E}[R^\top \mathbf{z}\mathbf{z}^\top R] = R^\top \mathbb{E}[\mathbf{z}\mathbf{z}^\top] R = R^\top I R = K.$$

Function drawn at random from a Gaussian Process with Gaussian covariance



Learning or Inference[1] in a Gaussian Process

Let's say you had a number of observations (x_i, y_i), where $i = 1, \ldots, n$, collectively $\mathbf{x}$ and $\mathbf{y}$, and a Gaussian likelihood function with noise variance σ^2_{noise}

$$p(\mathbf{y}|\mathbf{f}) \;=\; \mathcal{N}(\mathbf{y}|\mathbf{f},\; \sigma^2_{\text{noise}} I) \;\propto\; \exp(-\tfrac{1}{2}\sum_{i=1}^{n}(f_i - y_i)^2/\sigma^2_{\text{noise}}),$$

With a GP prior the joint distribution of function f and observations $\mathbf{y}$ is

$$\begin{aligned} p(f, \mathbf{y}) \;&=\; p(f)\, p(\mathbf{y}|\mathbf{f}) \;=\; p(\mathbf{y}) p(f|\mathbf{y}) \\ &=\; \mathcal{N}(f|m,\; k)\, \mathcal{N}(\mathbf{y}|\mathbf{f}) \;=\; Z_{|\mathbf{y}} \mathcal{N}(f|m_{|\mathbf{y}}, k_{|\mathbf{y}}), \end{aligned}$$

with posterior

$$p(f|\mathbf{y}) \;\sim\; \mathcal{N}(f|m_{|\mathbf{y}},\; k_{|\mathbf{y}}),$$

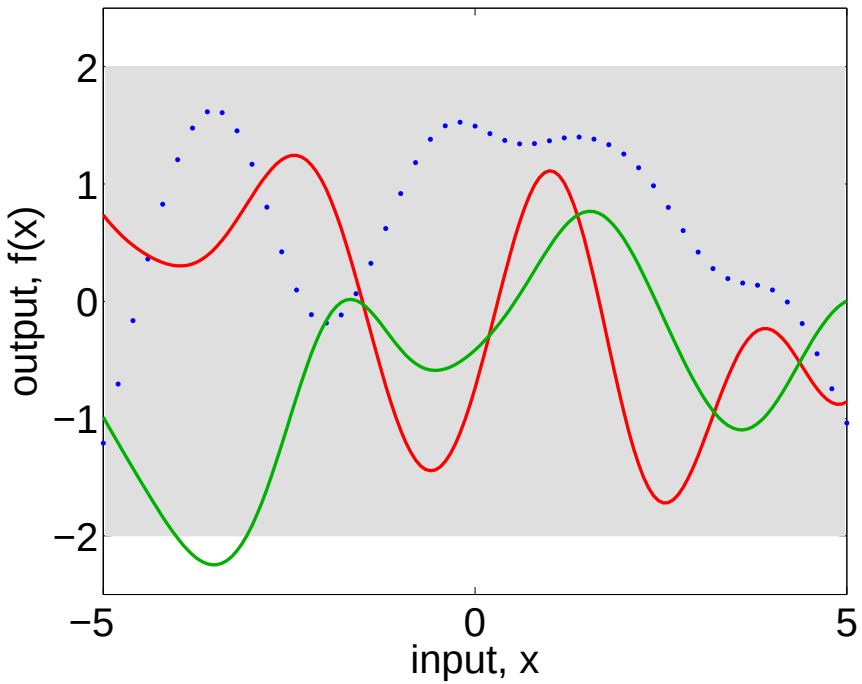
$$\text{where} \begin{cases} m_{|\mathbf{y}}(x) = m(x) + k(x, \mathbf{x})[k(\mathbf{x}, \mathbf{x}) + \sigma^2_{\text{noise}} I]^{-1}(\mathbf{y} - m(\mathbf{x})), \\ k_{|\mathbf{y}}(x, x') = k(x, x') - k(x, \mathbf{x})[k(\mathbf{x}, \mathbf{x}) + \sigma^2_{\text{noise}} I]^{-1} k(\mathbf{x}, x'), \end{cases}$$

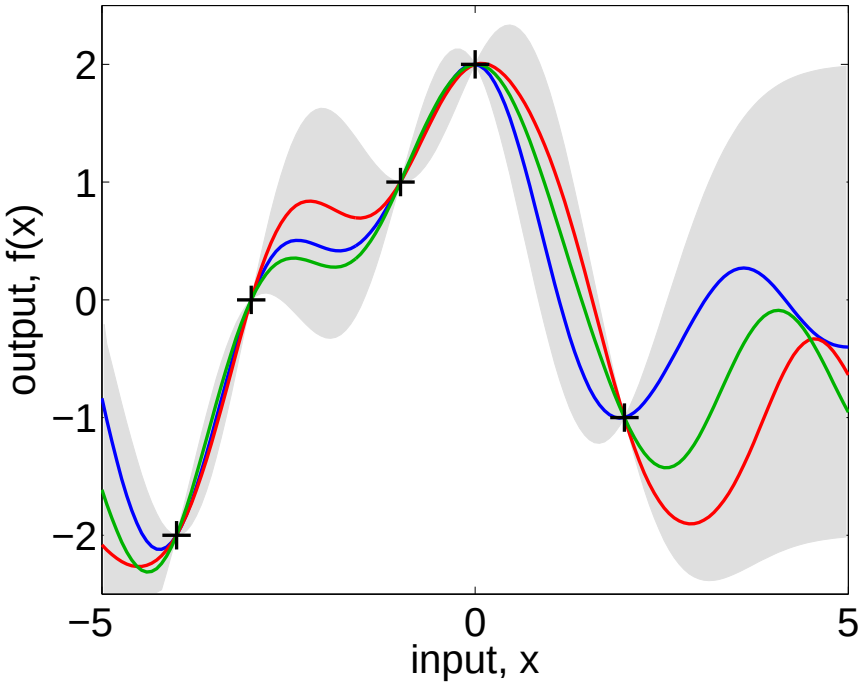
and log marginal likelihood

$$\log Z_{|\mathbf{y}} \;=\; \log \mathcal{N}(\mathbf{y}|m(\mathbf{x}),\; K(\mathbf{x}, \mathbf{x}) + \sigma^2_{\text{noise}} I).$$

[1] throughout, we use the statistical meaning of the word *inference*, not the neural network one

Prior and Posterior





Marginal distributions and samples of the joint, from the prior and the posterior given 5 close to noise free observations.

Hyperparameters: properties of covariance functions

The covariance function which we have seen before

$$k(x, x') = \exp(-\tfrac{1}{2}(x - x')^2),$$

encodes that $f(x)$ and $f(x')$ have large covariance if x is close to x', but it doesn't really quantify what is meant by close to?

We can parameterize the covariance function using hyperparameters such as ℓ, in

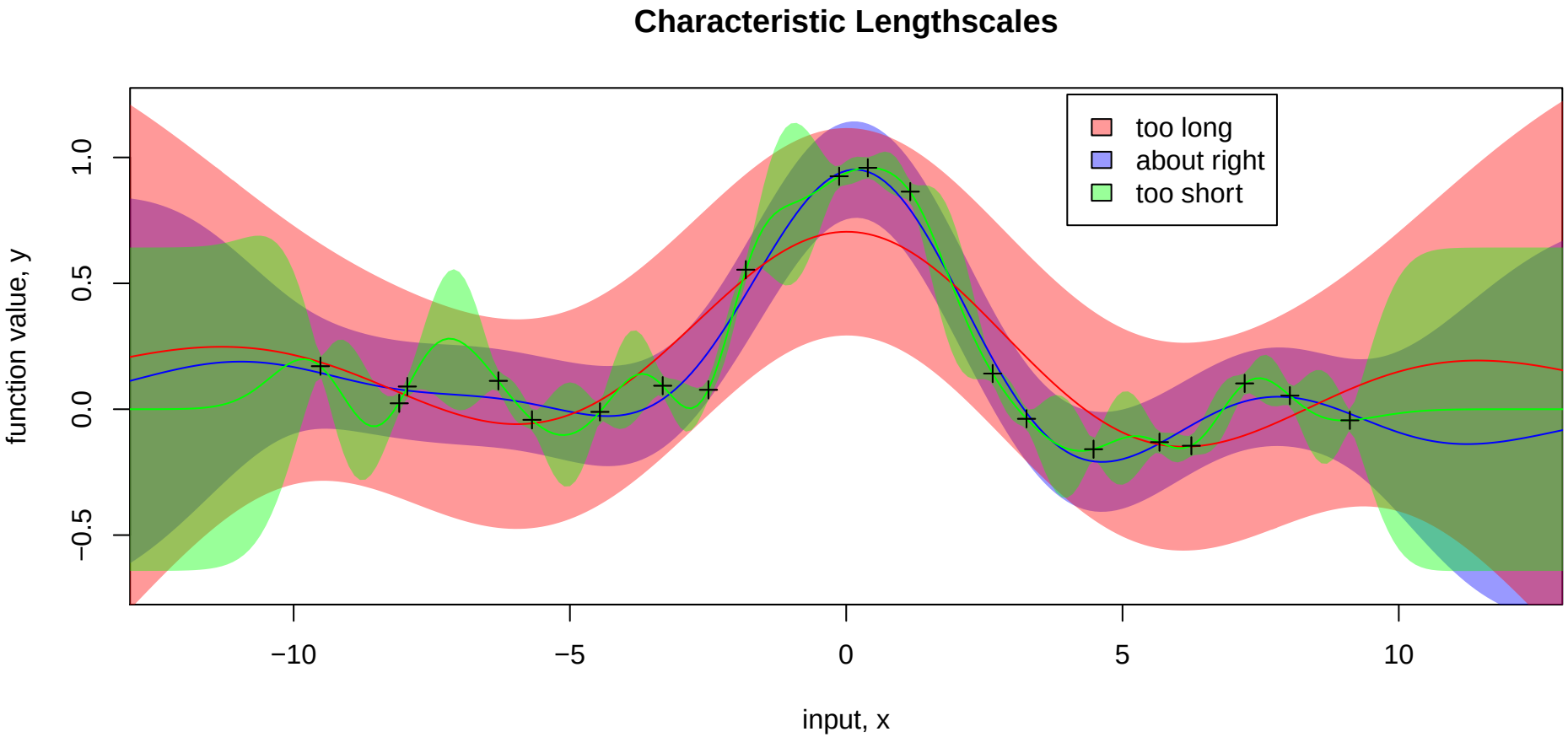
$$k(x, x') = \exp\left(-\frac{(x - x')^2}{2\ell^2}\right).$$

Learning in Gaussian process models involves finding

- the form of the covariance function, and
- any unknown (hyper-) parameters θ.

Example: Fitting the length scale parameter

Parameterized covariance function: $k(x, x') = v^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$.



The posterior GP is plotted for 3 different length scales (the blue curve corresponds to optimizing the marginal likelihood). Notice, that an almost exact fit to the data can be achieved by reducing the length scale – but the marginal likelihood does not favour this!

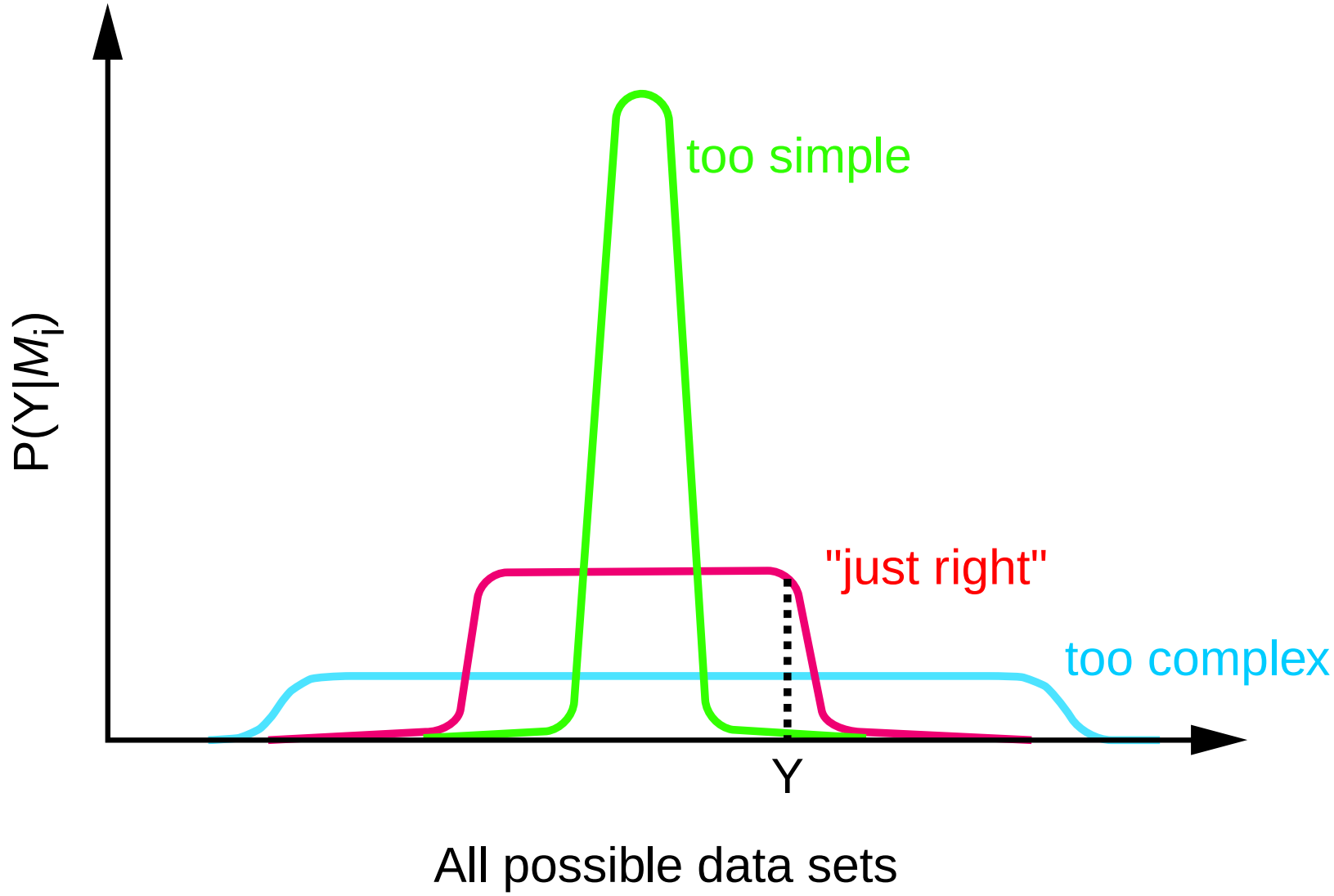
The Gaussian process marginal likelihood

Log marginal likelihood has a closed form

$$\begin{aligned}\log Z_{|\mathbf{y}} &= \log p(\mathbf{y}|\mathbf{x}) \\ &= -\frac{1}{2}(\mathbf{y}-\mathbf{m})^\top [K+\sigma_n^2 I]^{-1}(\mathbf{y}-\mathbf{m}) - \frac{1}{2}\log|K+\sigma_n^2 I| - \frac{n}{2}\log(2\pi)\end{aligned}$$

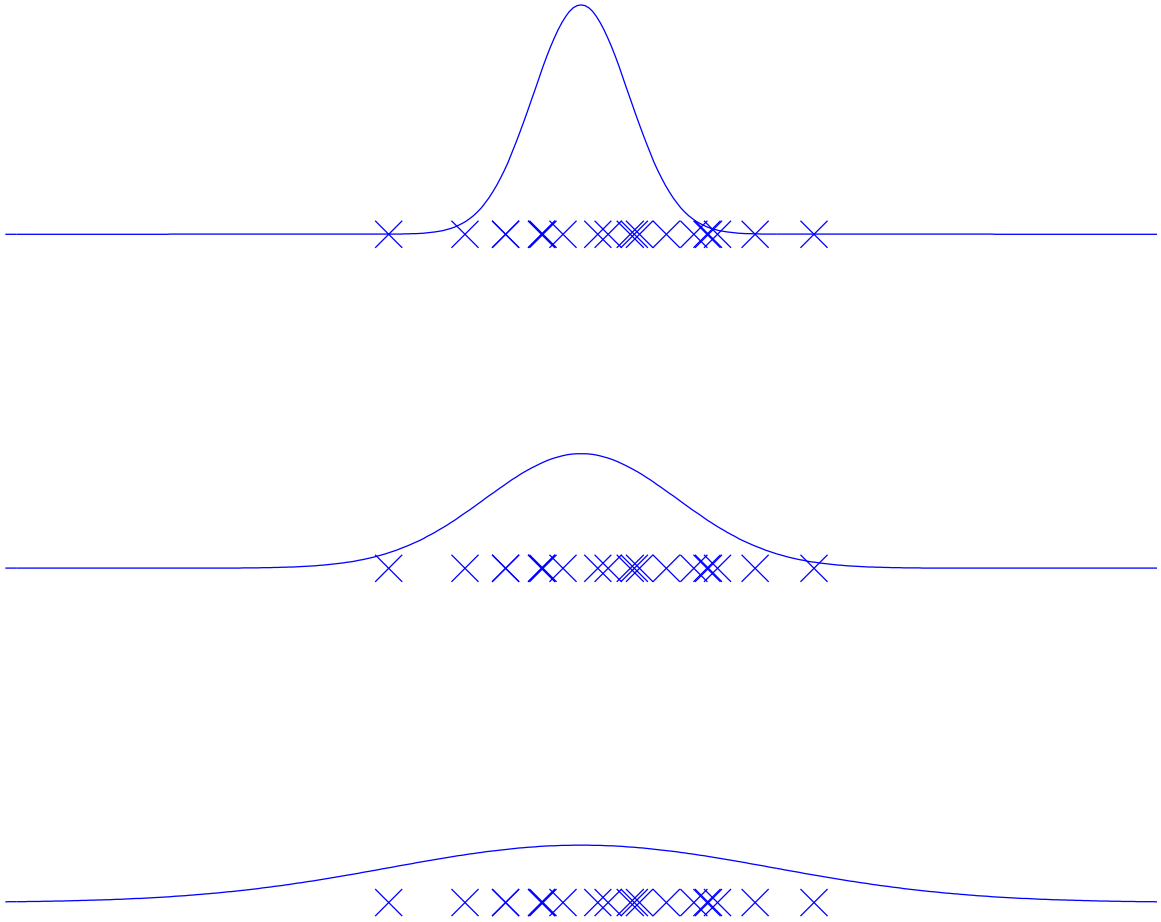
and is the combination of a data fit term and complexity penalty. Occam's Razor is automatic.

How can Bayes rule help find the right model complexity? Marginal likelihoods and Occam's Razor



An illustrative analogous example

Imagine the simple task of fitting the variance, σ^2, of a zero-mean Gaussian to a set of n scalar observations.



The log likelihood is $\log p(\mathbf{y}|\mu, \sigma^2) = -\frac{1}{2}\mathbf{y}^\top I\mathbf{y}/\sigma^2 - \frac{1}{2}\log|I\sigma^2| - \frac{n}{2}\log(2\pi)$

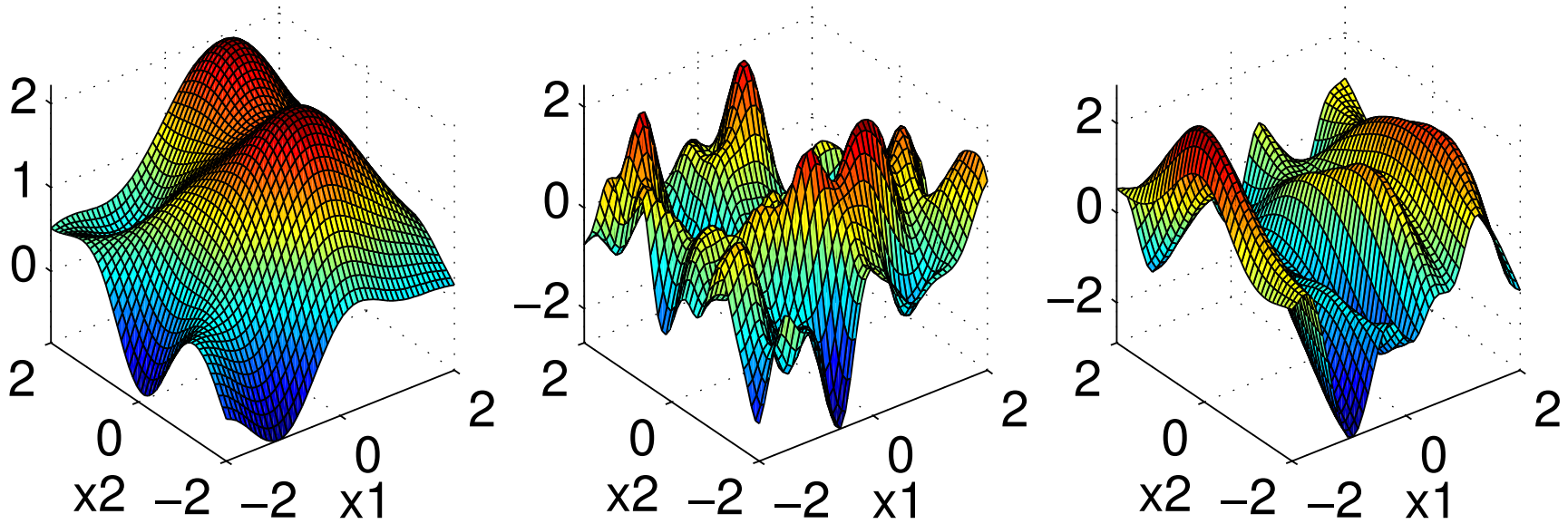
Model Selection, Hyperparameters, and ARD

We need to determine both the *form* and *parameters* of the covariance function. We typically use a hierarchical model, where the parameters of the covariance are called hyperparameters.

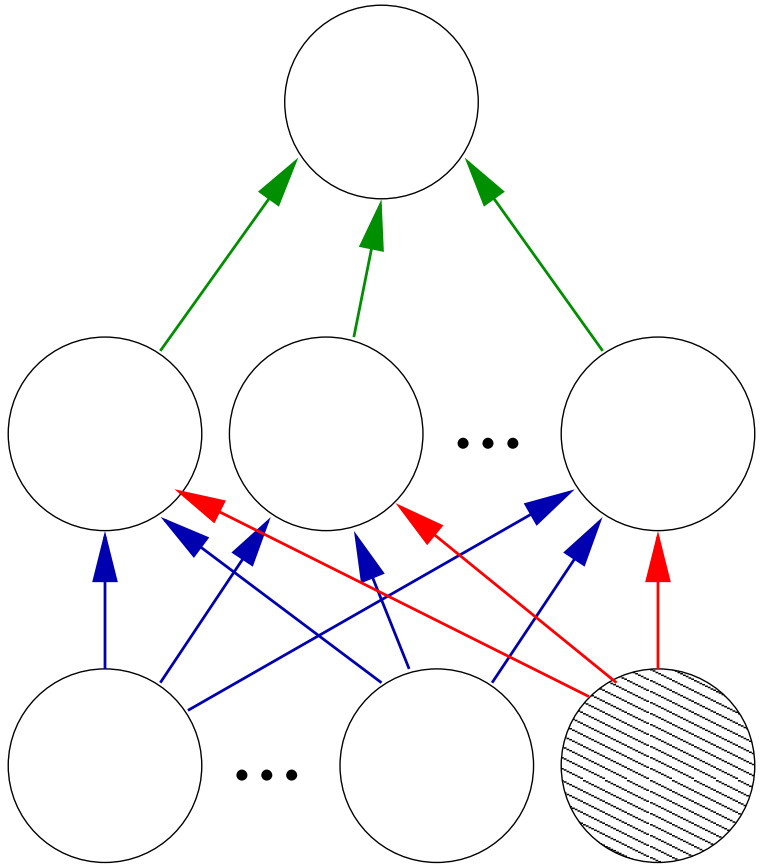
A very useful idea is to use automatic relevance determination (ARD) covariance functions for feature/variable selection, e.g.:

$$k(\mathbf{x}, \mathbf{x}') = v_0^2 \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{2v_d^2}\right), \qquad \text{hyperparameters } \theta = (v_0, v_1, \ldots, v_d, \sigma_n^2).$$

v1=v2=1 v1=v2=0.32 v1=0.32 and v2=1



Feed Forward Neural Networks



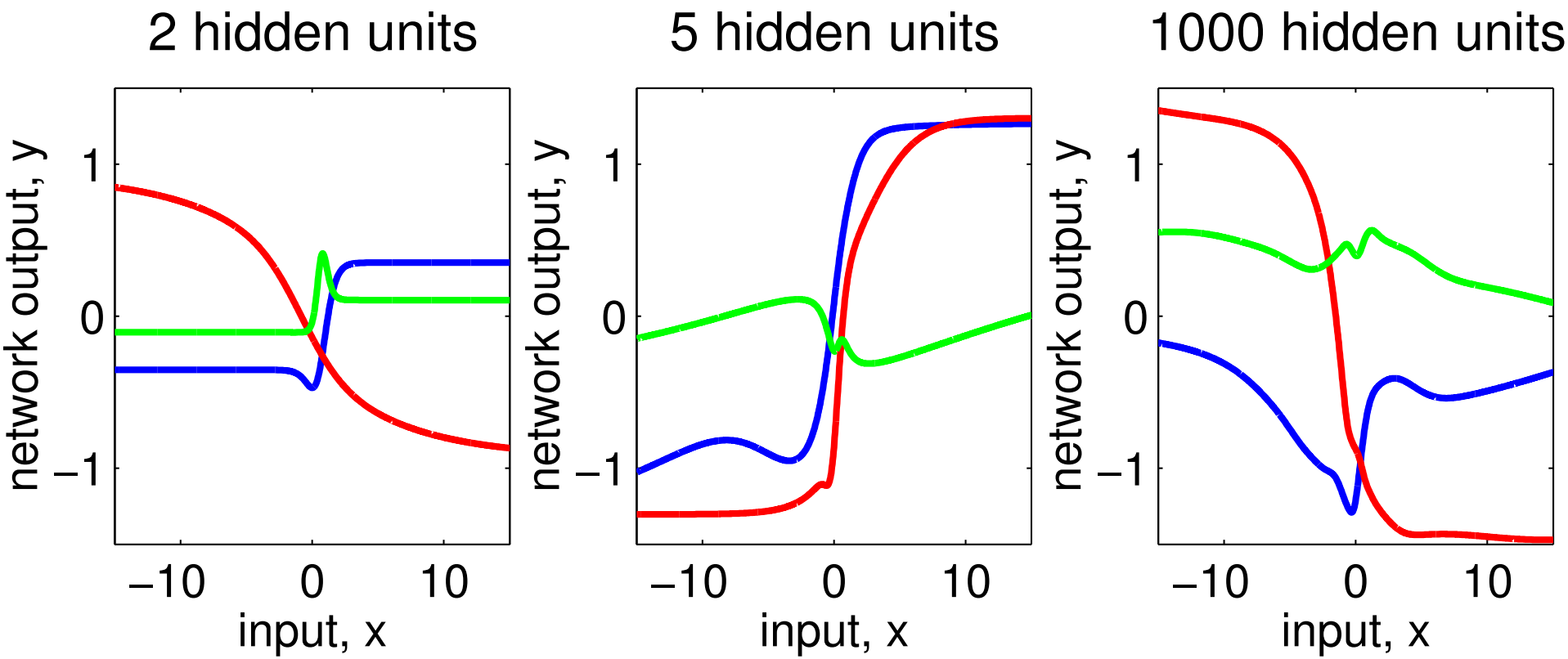
Weight groups:
output weights
input–hidden
bias–hidden

A feed forward neural network implements the function:

$$f(x) = \sum_{i=1}^{H} v_i \tanh\left(\sum_j u_{ij} x_j + b_j\right)$$

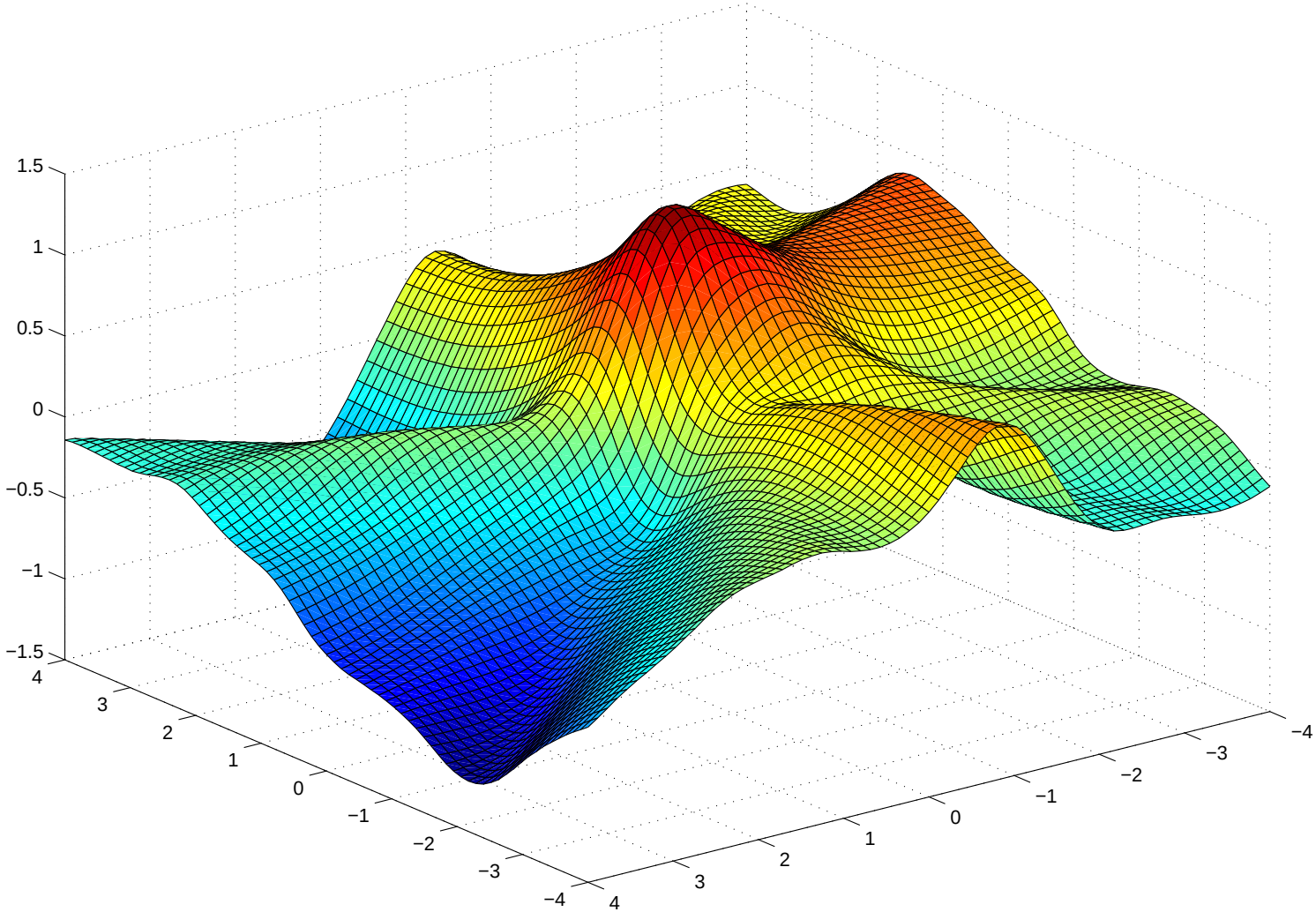
Limits of Large Neural Networks

Sample random neural network weights from a appropriately scaled Gaussian prior.



Note: The prior on the neural network weights *induces* a prior over functions.

Function drawn at random from a Neural Network covariance function



$$k(x, x') = \frac{2}{\pi} \arcsin\left(\frac{2x^\top \Sigma x'}{\sqrt{(1 + x^\top \Sigma x)(1 + 2x'^\top \Sigma x')}}\right).$$

Composite covariance functions

We've seen examples of covariance functions.

Covariance functions have to be possitive definite.

One way of building covariance functions is by composing simpler ones in various ways

- sums of covariance functions $k(x, x') = k_1(x, x') + k_2(x, x')$
- products $k(x, x') = k_1(x, x') \times k_2(x, x')$
- other combinations: $g(x)k(x, x')g(x')$
- etc.

Conclusions

GPs are a small but powerful generalisation of the Gaussian to *functions*; we can

- calculate marginals
- sample from the joint marginals
- update when data is observed

GPs are the powerful, principled and practical way to do inference about functions

Important things that I haven't spoken about

- library of covariance functions
- non-Gaussian likelihoods
- computational constraints: sparse approximations

Want to know more:
Rasmussen and Williams (2006): Gaussian Processes for Machine Learning