COM4509/6509
Machine Learning and Adaptive Intelligence
Lecture 5: Gaussian Processes

Mike Smith* and Matt Ellis

*m.t.smith@sheffield.ac.uk

Gaussian Processes: The idea

Gaussian Processes: The maths

An alternative understanding: The Kernel Trick

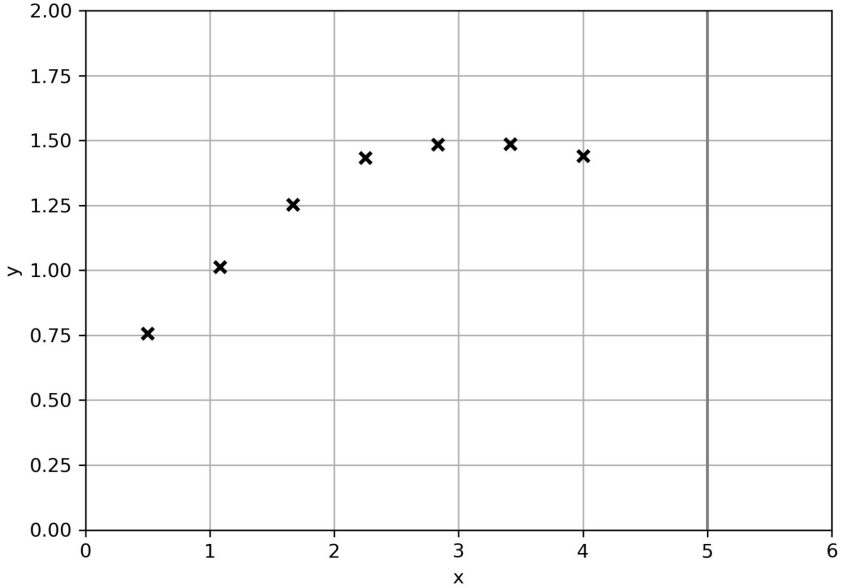
Gaussian Processes for classification and big data.

Our assumptions…

Activity 1

Look at these (noisy) observations of a function.

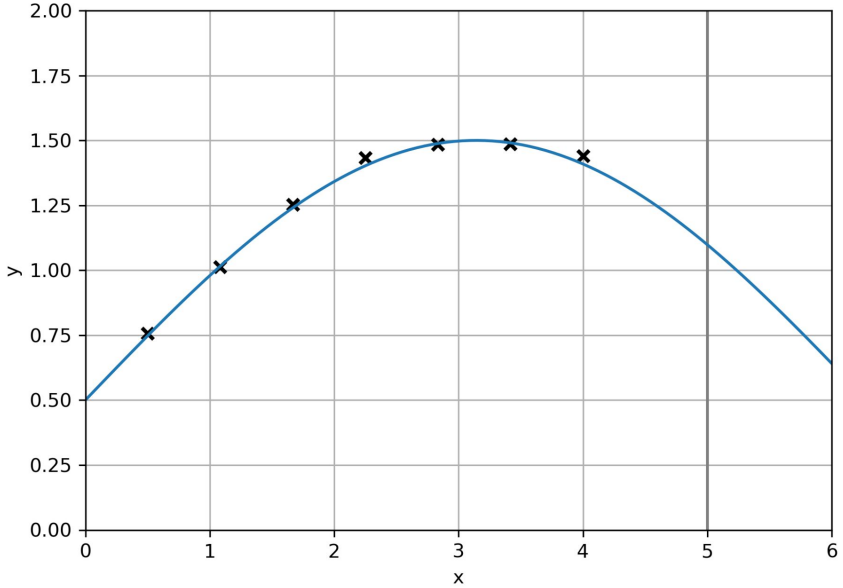
What does the function equal at x=5?



Our assumptions…

What does the hidden (latent) function equal at x=5?

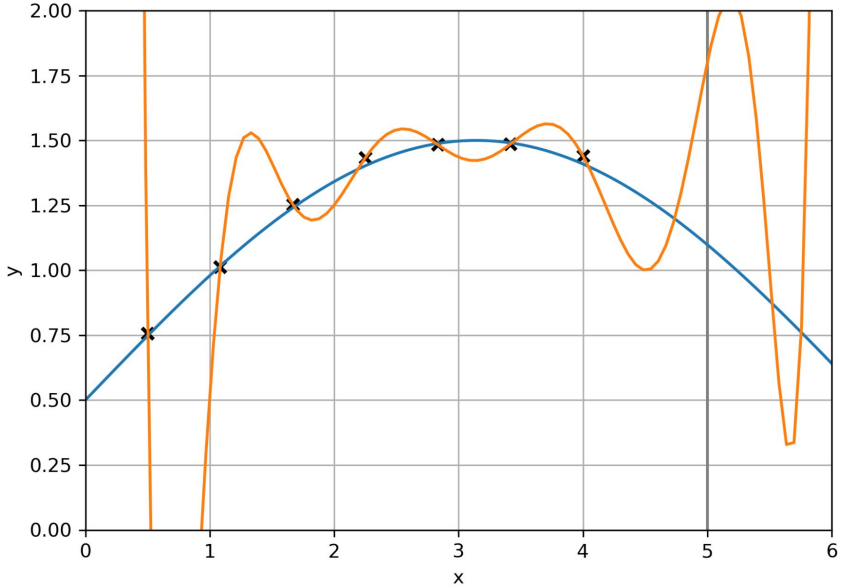
- Maybe around 1.1?



Our assumptions…

What does the hidden (latent) function equal at x=5?

- Maybe around 1.1?
- What about 1.8?



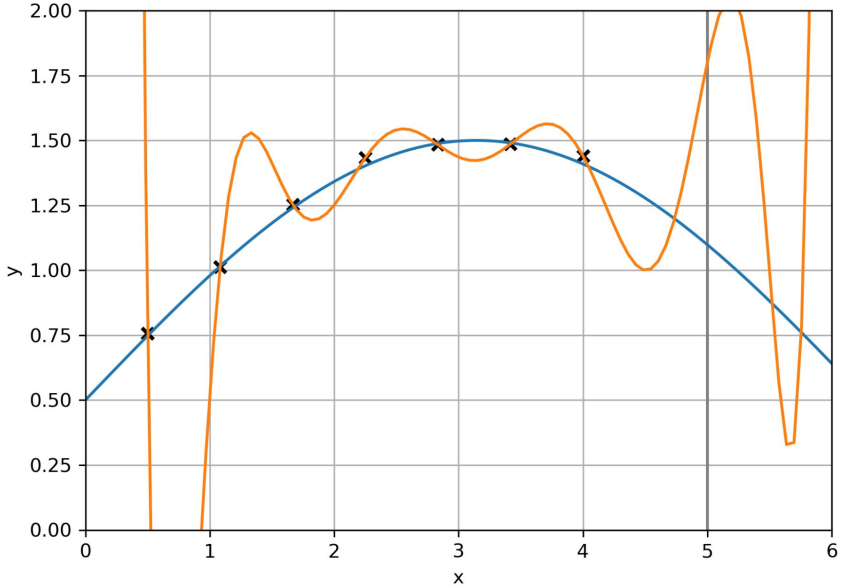
Our assumptions…

What does the hidden (latent) function equal at x=5?

- Maybe around 1.1?
- What about 1.8?

Why do we think this is a “less probable” function than the blue guess?

- The blue one is smoother?
- It’s less ‘complicated’.



Our assumptions…

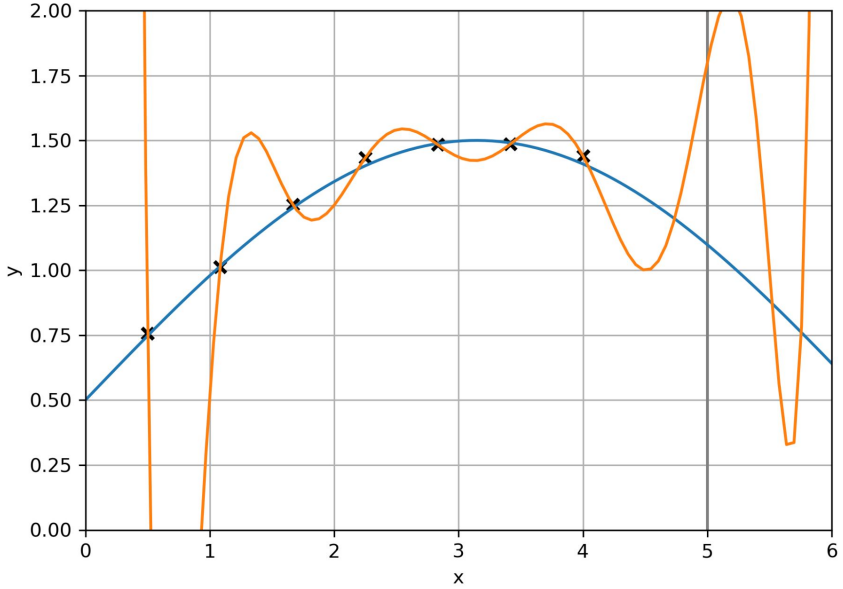
What does the hidden (latent) function equal at x=5?

- Maybe around 1.1?
- What about 1.8?

Why do we think this is a “less probable” function than the blue guess?

- The blue one is smoother?
- It’s less ‘complicated’.

These **assumptions** are our **prior**.



Our assumptions…

If we don’t make any assumptions at all, then we can’t make any predictions.

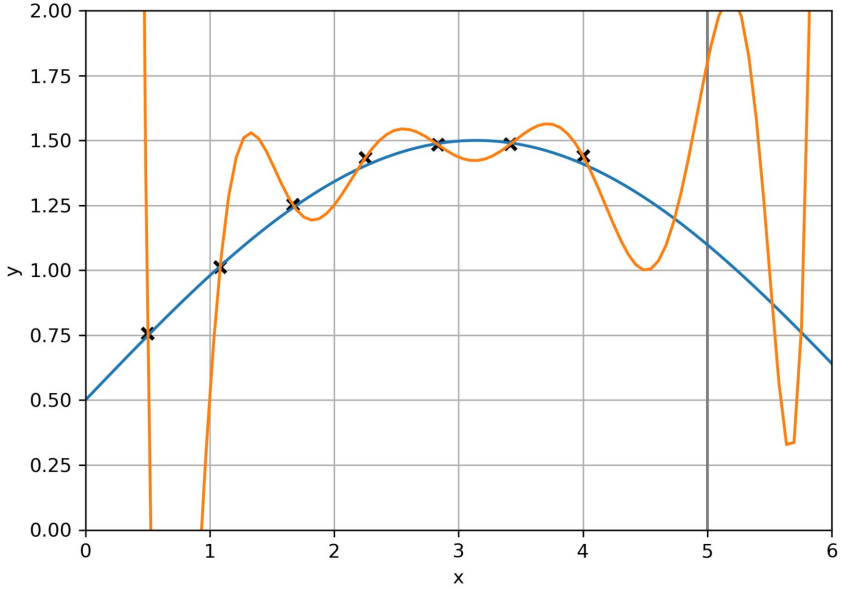
What does the hidden (latent) function equal at x=5?

- Maybe around 1.1?
- What about 1.8?

Why do we think this is a “less probable” function than the blue guess?

- The blue one is smoother?
- It’s less ‘complicated’.

These **assumptions** are our **prior**.



Our assumptions…

If we don’t make any assumptions at all, then we can’t make any predictions.

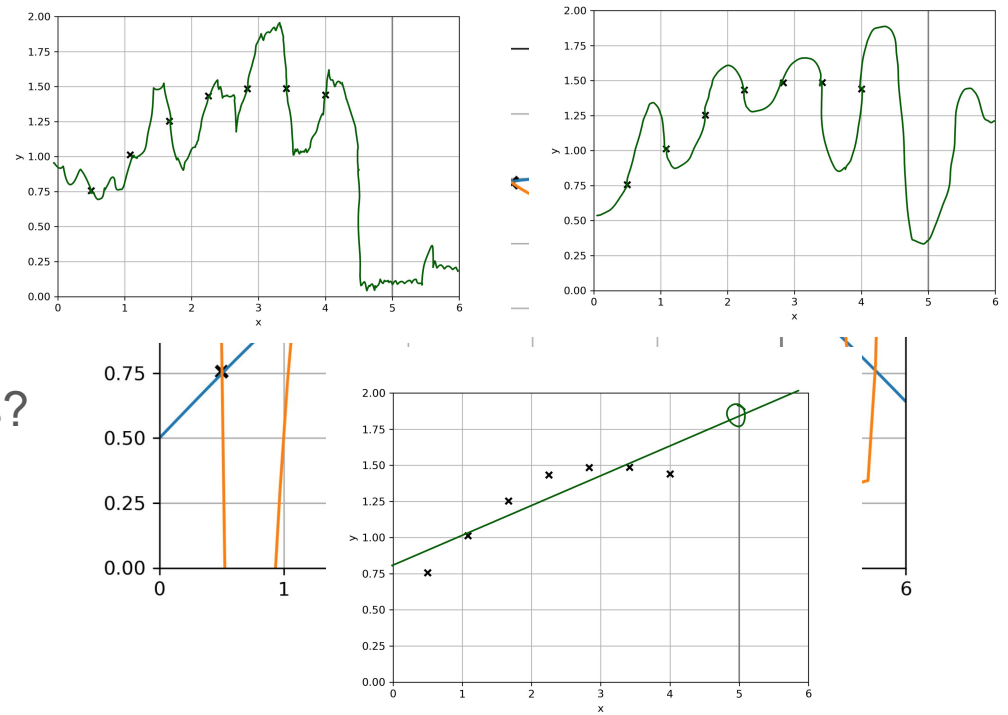
What does the hidden (latent) function equal at x=5?

- Maybe around 1.1?
- What about 1.8?

Why do we think this is a “less probable” function than the blue guess?

- The blue one is smoother?
- It’s less ‘complicated’.

These **assumptions** are our **prior**.



Our assumptions…

What does the hidden (latent) function equal at x=5?

- Maybe around 1.1?
- What about 1.8?

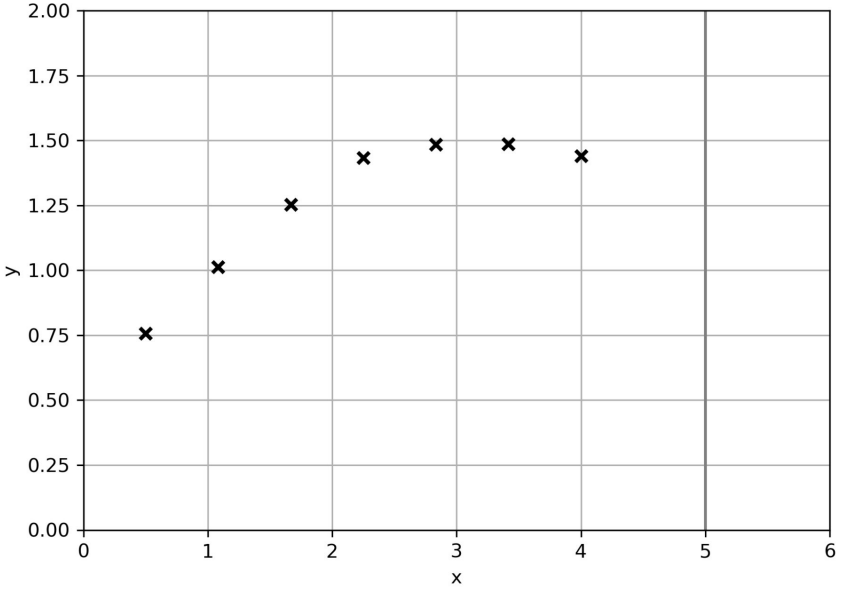
Why do we think this is a “less probable” function than the blue guess?

- The blue one is smoother?
- It’s less ‘complicated’.

These **assumptions** are our **prior**.

If we don’t make any assumptions at all, then we can’t make any predictions.

Why did you assume a smooth function?



Our assumptions…

What does the hidden (latent) function equal at x=5?

- Maybe around 1.1?
- What about 1.8?

Why do we think this is a “less probable” function than the blue guess?

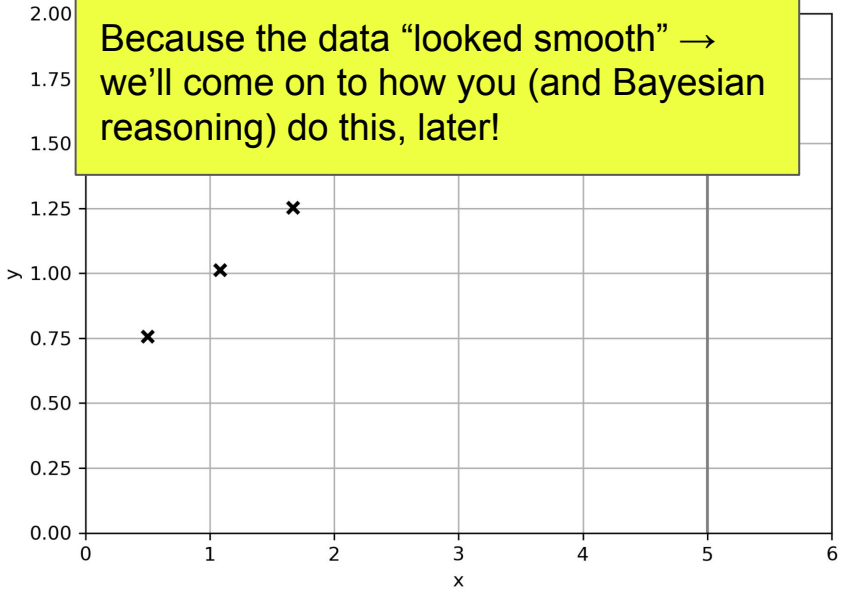
- The blue one is smoother?
- It’s less ‘complicated’.

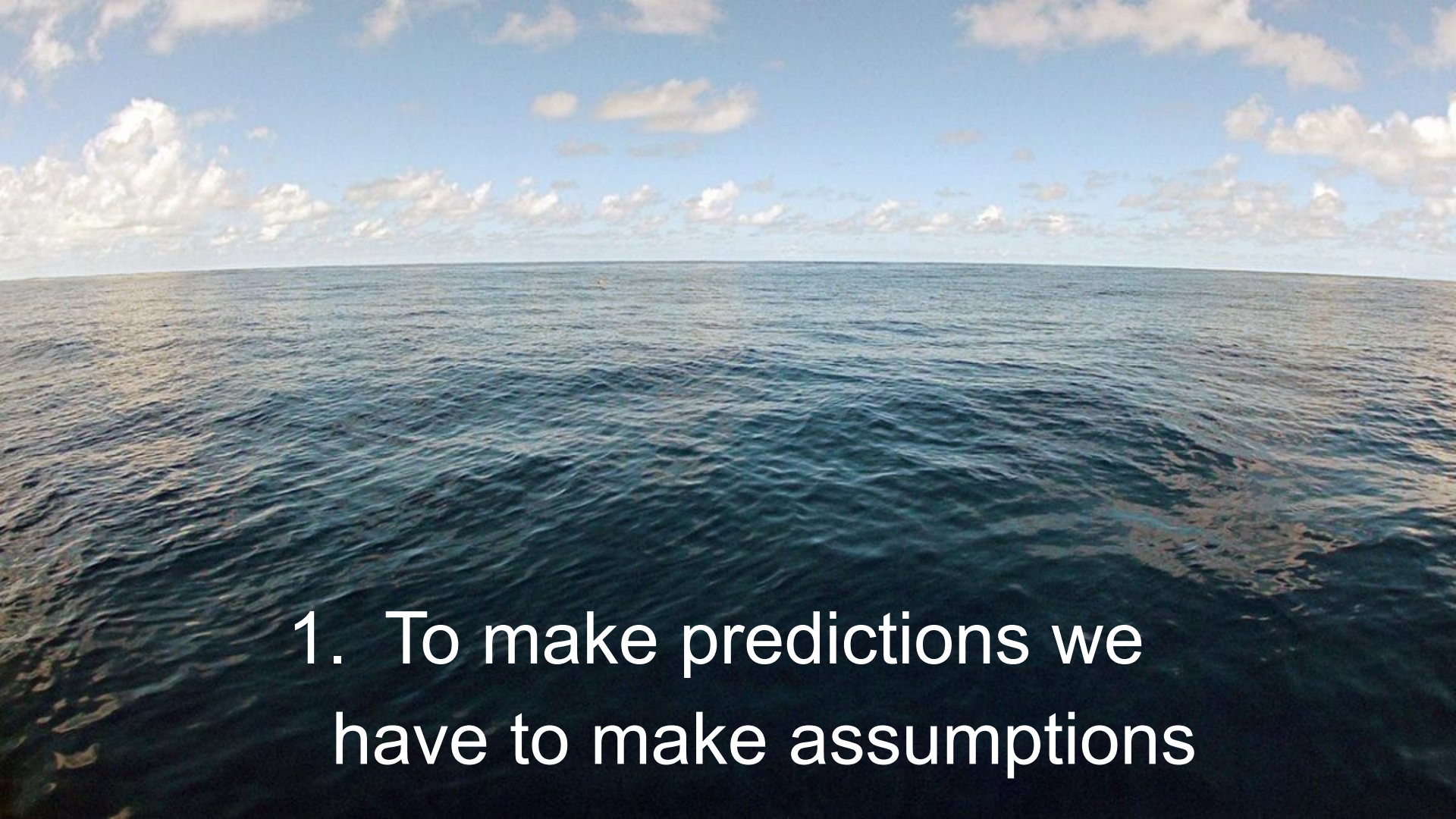
These **assumptions** are our **prior**.

If we don’t make any assumptions at all, then we can’t make any predictions.

Why did you assume a smooth function?

Because the data “looked smooth” → we’ll come on to how you (and Bayesian reasoning) do this, later!



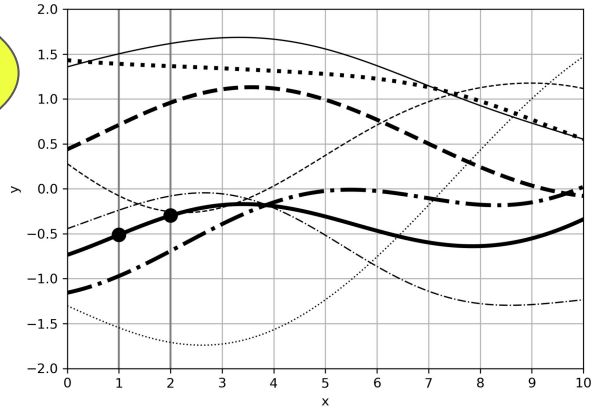

1. To make predictions we
have to make assumptions

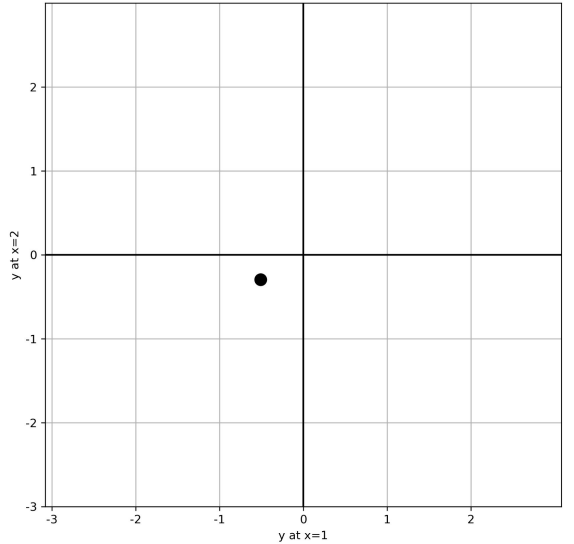
Gaussian processes

Activity 2

For each function, plot its value at x=1, against its value at x=2.

What do you notice?

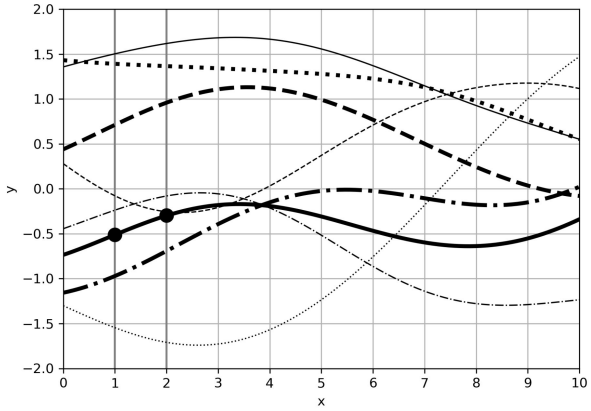


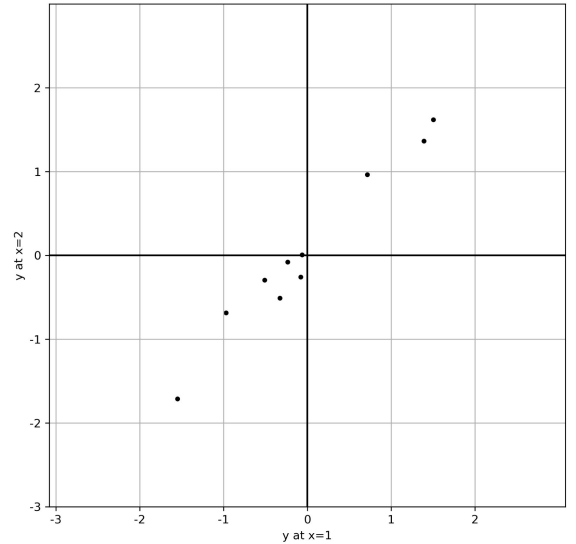


Gaussian processes

For each function, plot its value at x=1, against its value at x=2.

What do you notice?





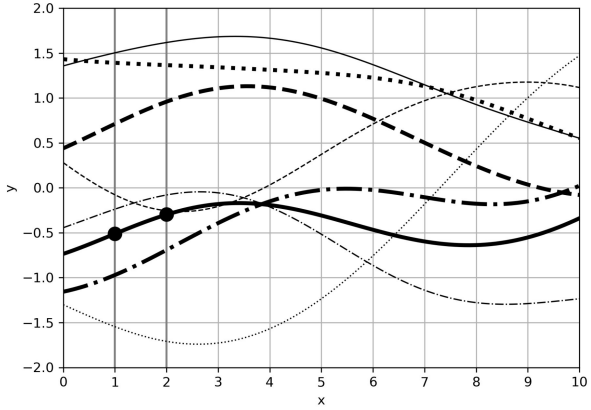
You should have a series of points like this.

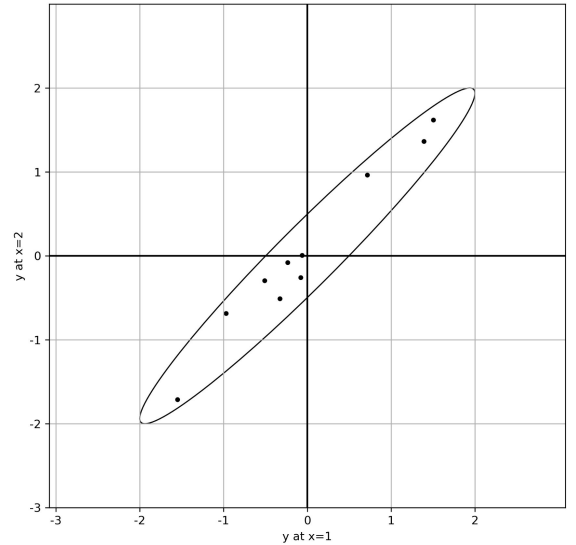
Gaussian processes

For each function, plot its value at x=1, against its value at x=2.

What do you notice?

- There seems to be some correlation between y at x_1 and y at x_2.





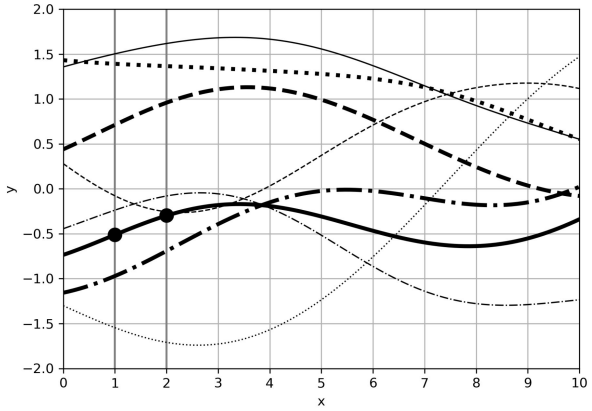
You should have a series of points like this.

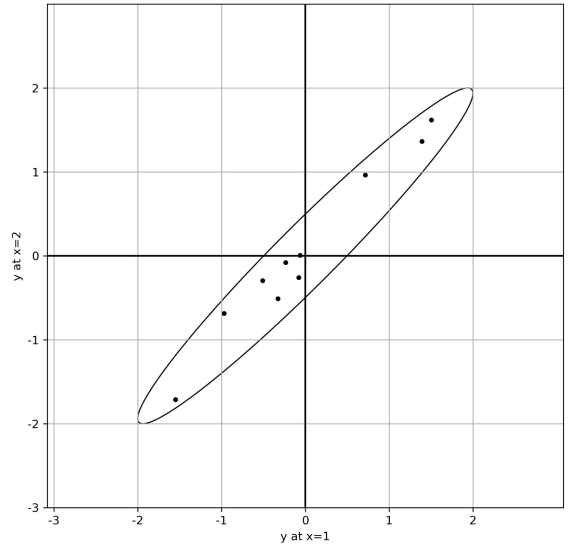
Gaussian processes

For each function, plot its value at x=1, against its value at x=2.

What do you notice?

- There seems to be some correlation between y at x_1 and y at x_2.
- The pattern of points seems to be Gaussian.





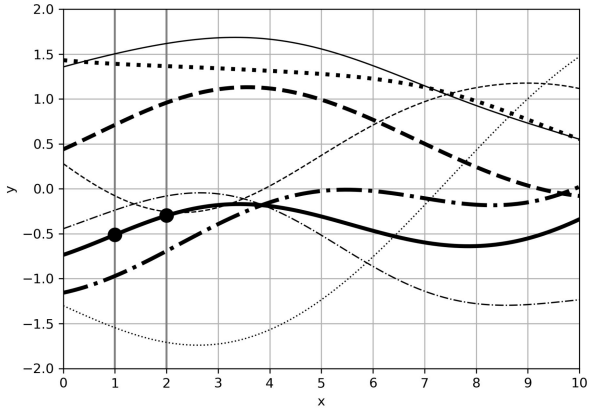
You should have a series of points like this.

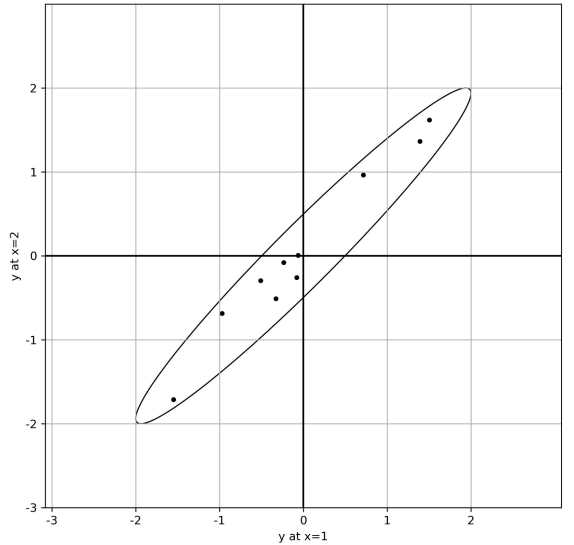
Gaussian processes

For each function, plot its value at x=1, against its value at x=2.

What do you notice?

- There seems to be some correlation between y at x_1 and y at x_2.
- The pattern of points seems to be Gaussian.
- This is because the functions are sampled from a Gaussian process.





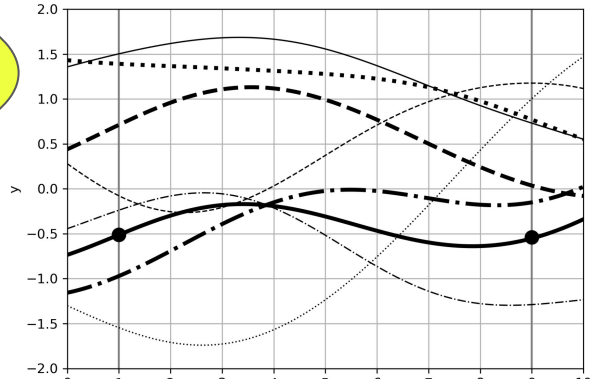
You should have a series of points like this.

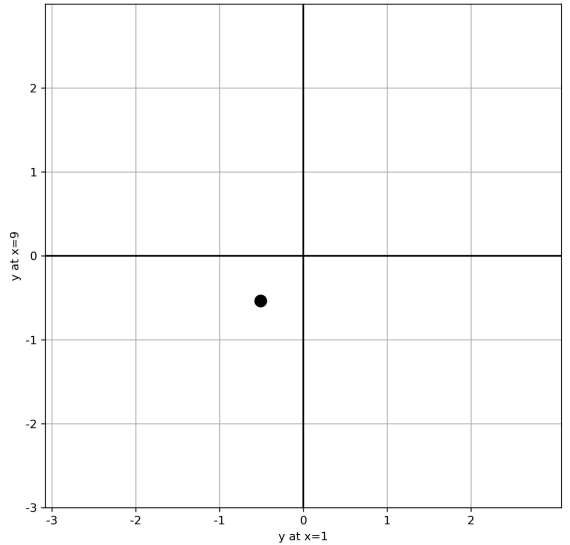
Gaussian processes

Activity 3

For each function, plot its value at x=1, against its value at **x=9**.

What do you notice?



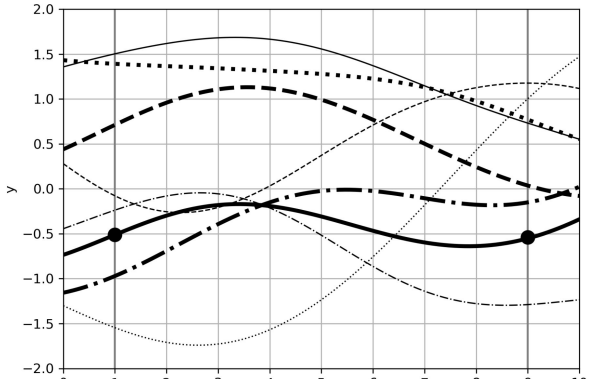


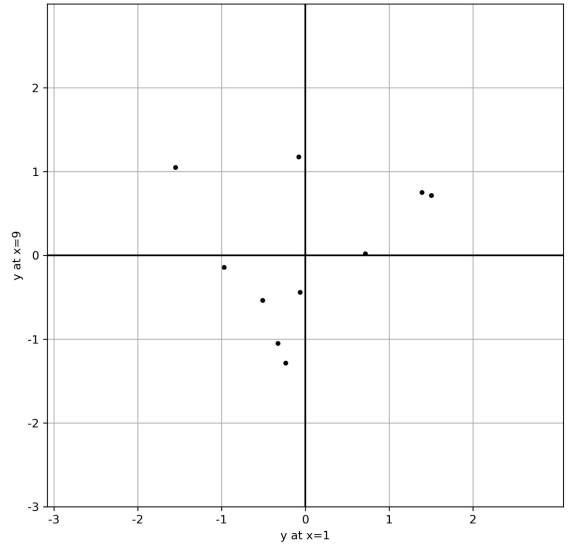
Gaussian processes

For each function, plot its value at x=1, against its value at **x=9**.

What do you notice?

- There seems to be less correlation between the points.



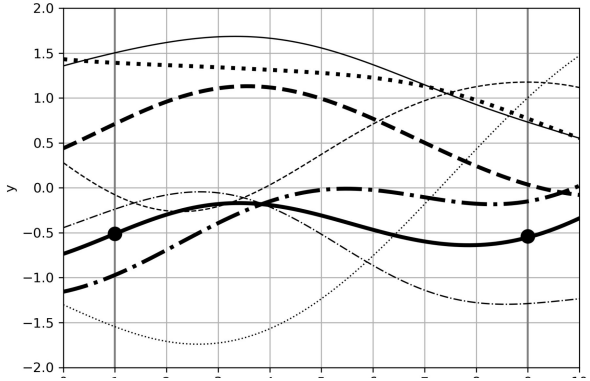


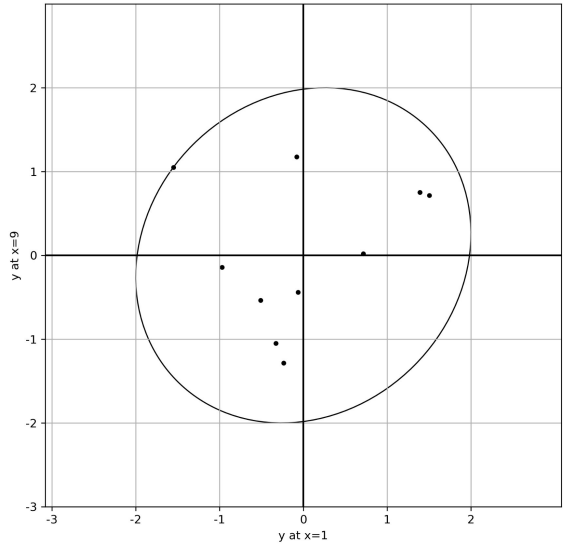
Gaussian processes

For each function, plot its value at x=1, against its value at **x=9**.

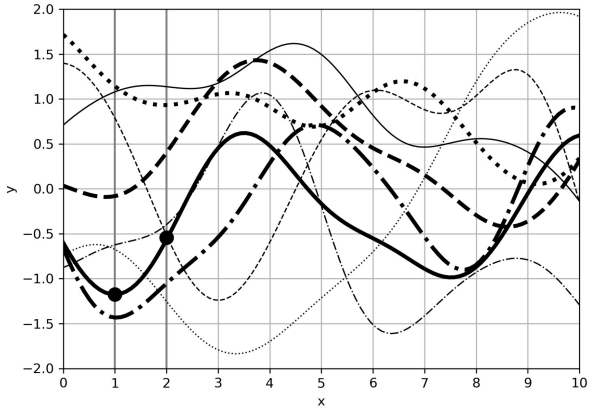
What do you notice?

- There seems to be less correlation between the points.
- The covariance between the values at the two locations is much less.





Gaussian processes



For each function, plot its value at x=1, against its value at **x=9**.

What do you notice?

- There seems to be less correlation between the points.
- The covariance between the values at the two locations is much less.

What would happen to these two covariances we've plotted, if the functions were more 'wiggly'?

Key question

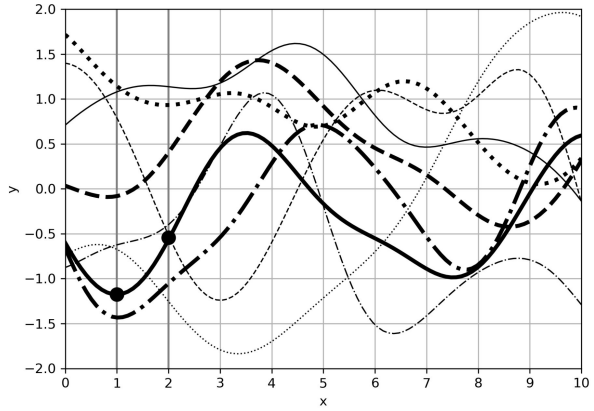
Gaussian processes

For each function, plot its value at x=1, against its value at **x=9**.

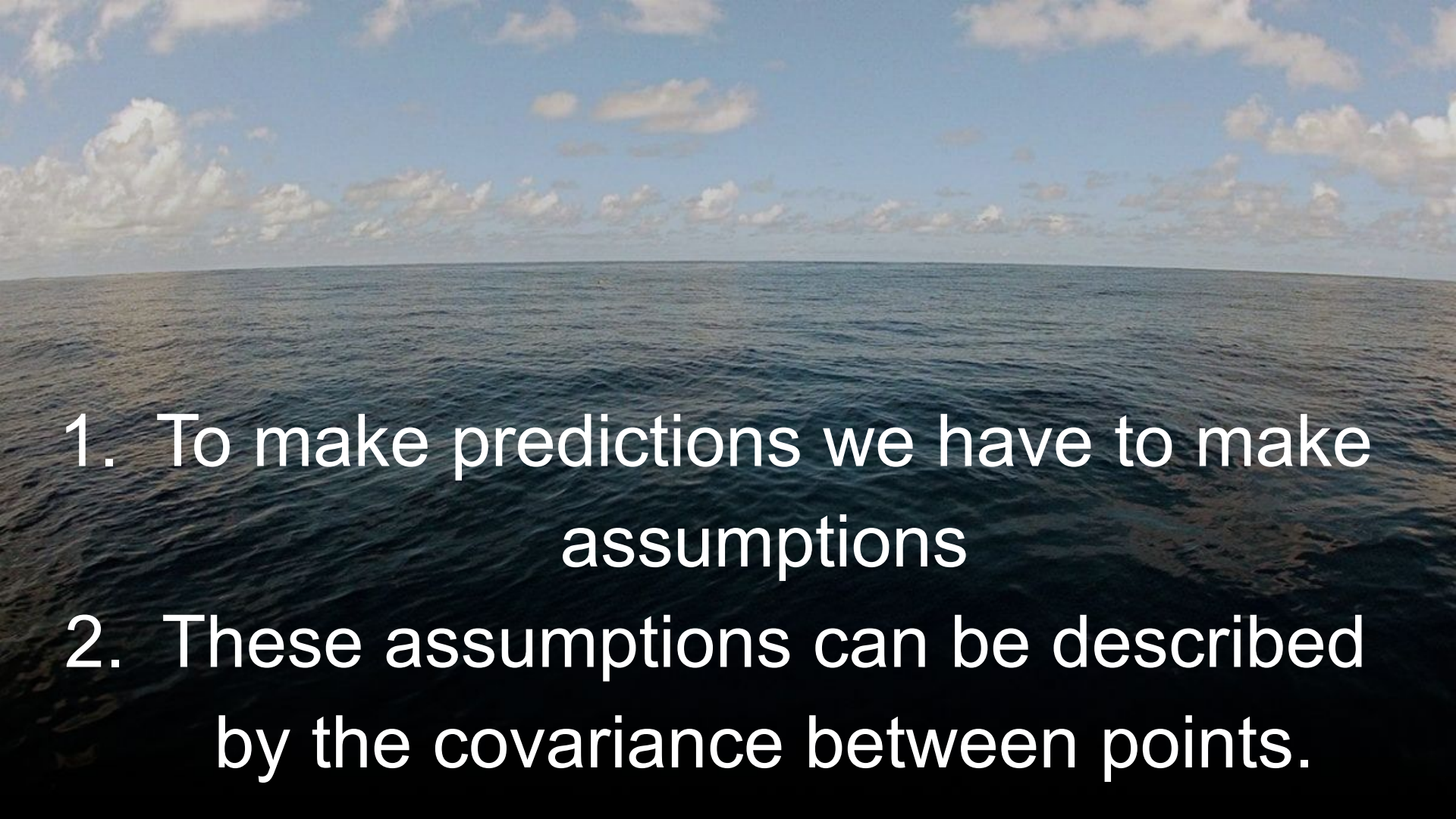
What do you notice?

- There seems to be less correlation between the points.
- The covariance between the values at the two locations is much less.

What would happen to these two covariances we've plotted, if the functions were more 'wiggly'?



There is less correlation (or covariance) between pairs of points, as the function can change more quickly

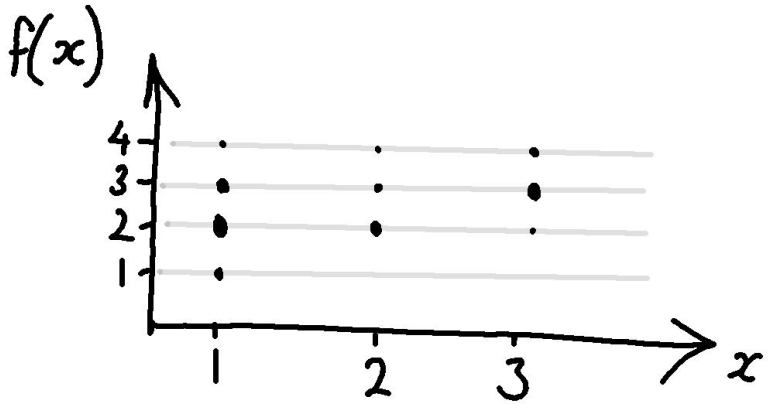

1. To make predictions we have to make assumptions
2. These assumptions can be described by the covariance between points.

What is a stochastic process?

A collection of random variables,

indexed by some variable $x \in X$

For example...

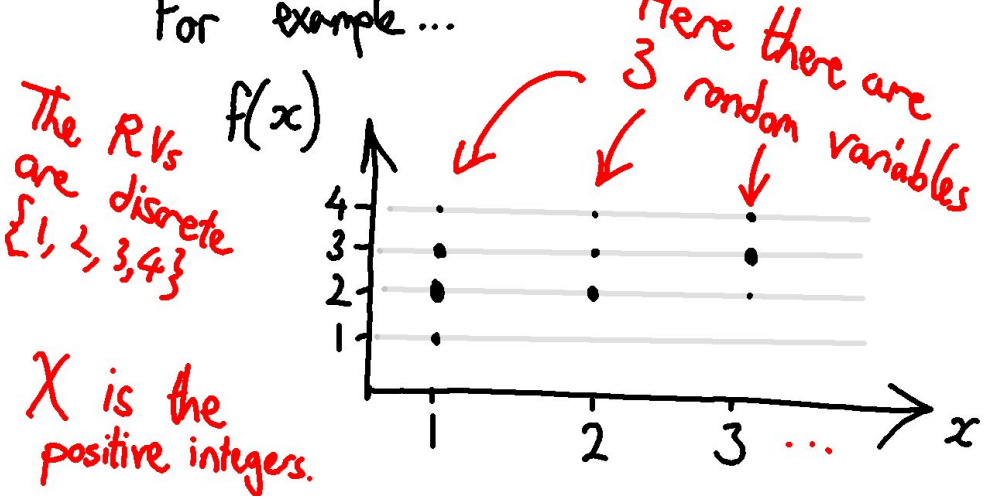


What is a stochastic process?

A collection of random variables,

indexed by some variable $x \in X$

For example...



What is a stochastic process?

A collection of random variables,

indexed by some variable $x \in X$

$f(x) \in \mathbb{R}$

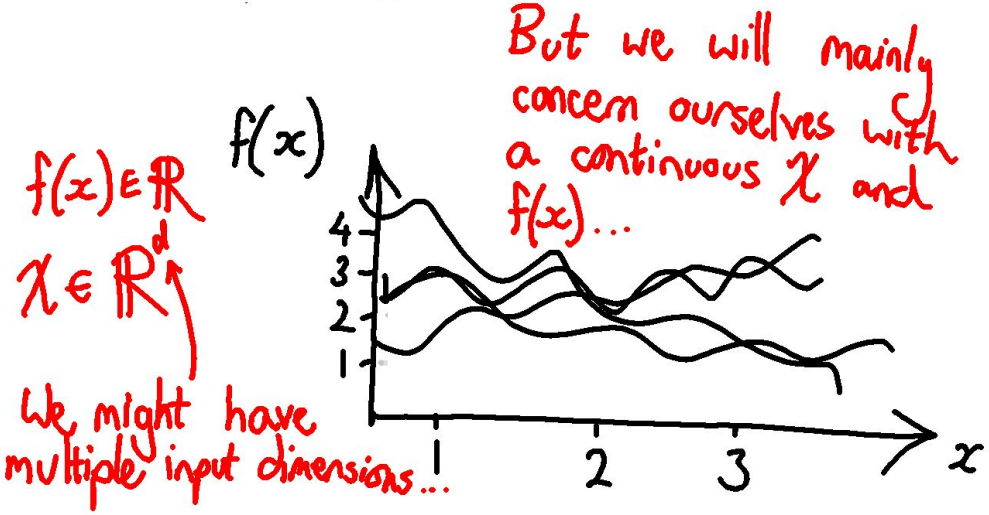
$X \in \mathbb{R}$

But we will mainly concern ourselves with a continuous X and $f(x)$...

What is a stochastic process?

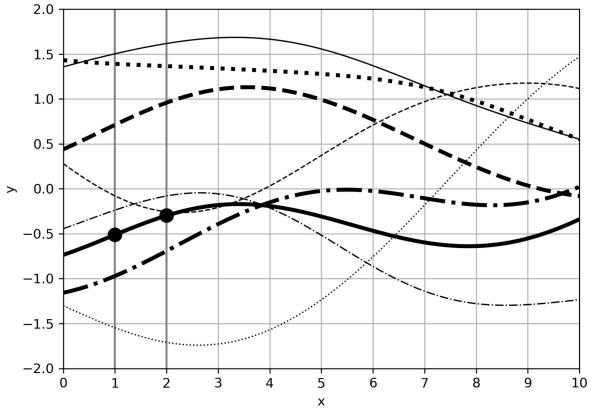
A collection of random variables,

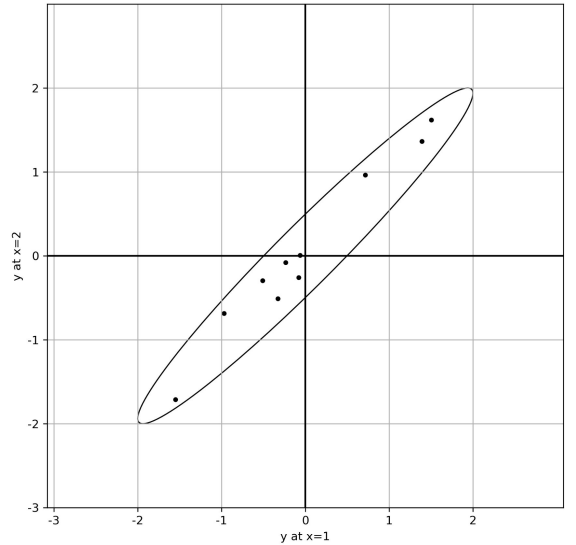
indexed by some variable $x \in \mathcal{X}$



Gaussian Process

A Gaussian process is a stochastic process in which any pair of random variables are Gaussian distributed.

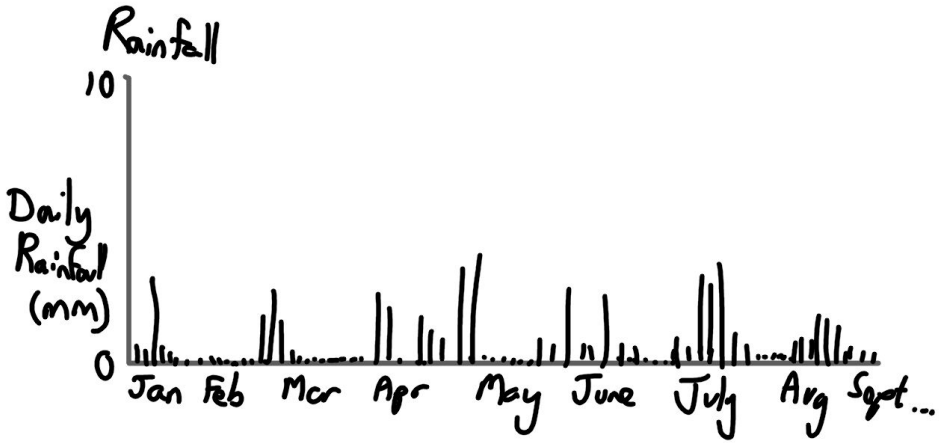




Gaussian Process

A Gaussian process is a stochastic process in which any pair of random variables are Gaussian distributed.

It maybe helps to consider an example of a stochastic process that ISN'T a Gaussian process…



E.g. Daily Rainfall

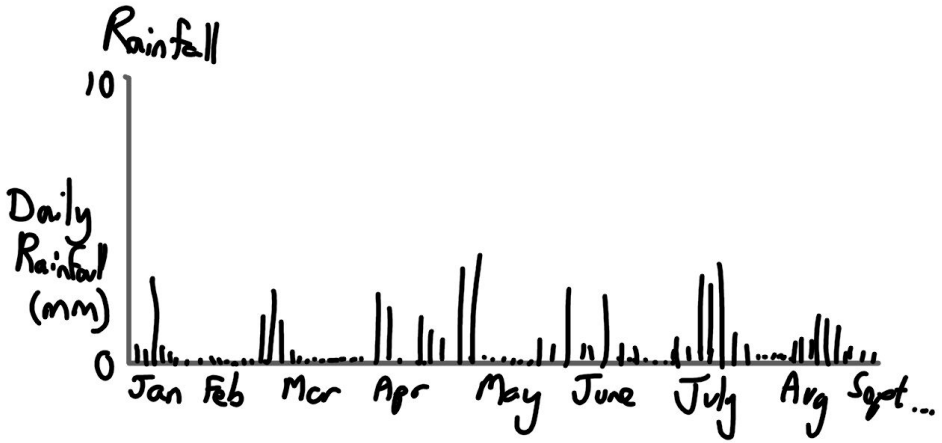
Activity 4

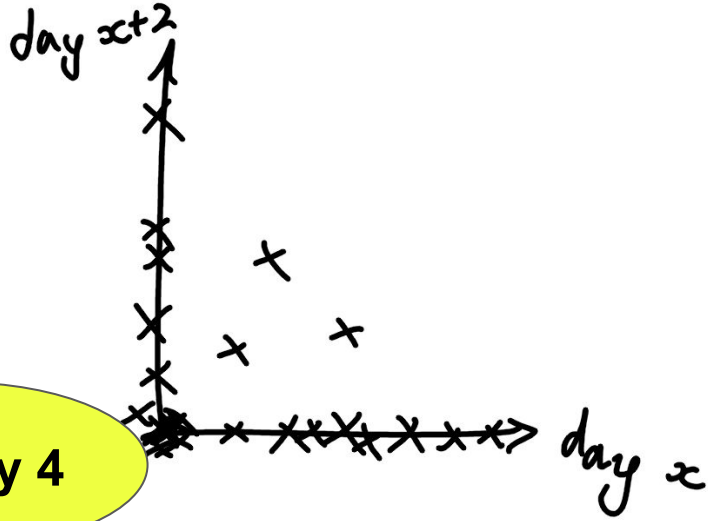
Gaussian Process

A Gaussian process is a stochastic process in which any pair of random variables are Gaussian distributed.

It maybe helps to consider an example of a stochastic process that ISN'T a Gaussian process…

Daily rainfall: If we plot one day against e.g. a day 2 days later, we'll find the distribution is not at all Gaussian.





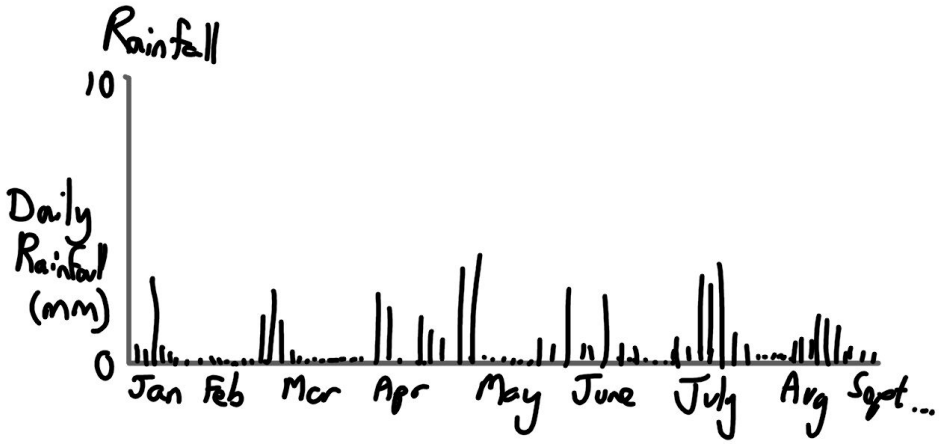
Activity 4

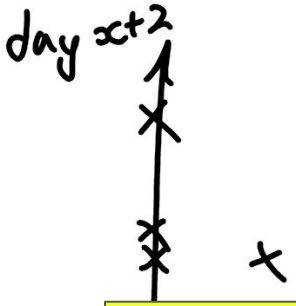
Gaussian Process

A Gaussian process is a stochastic process in which any pair of random variables are Gaussian distributed.

It maybe helps to consider an example of a stochastic process that ISN'T a Gaussian process…

Daily rainfall: If we plot one day against e.g. a day 2 days later, we'll find the distribution is not at all Gaussian.





Activity 4

The rainfall on individual days isn't Gaussian distributed (mostly zero, is non-negative, etc). The distribution between days isn't Gaussian either.

1. To make predictions we have to make assumptions
2. These assumptions can be described by the covariance between points.
3. The “Gaussian” in Gaussian process is the assumption that these different points are Gaussian distributed.

Gaussian Process Regression

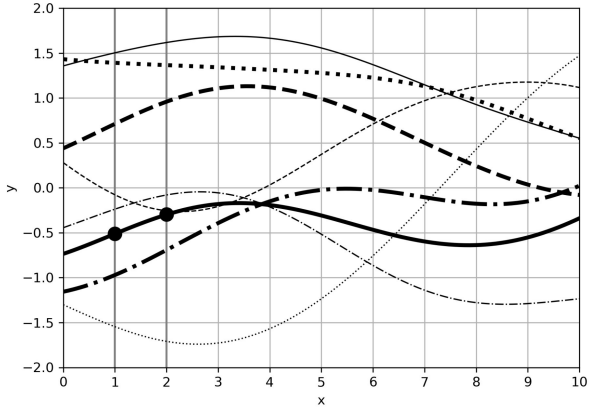
aka: How to make predictions using a Gaussian process

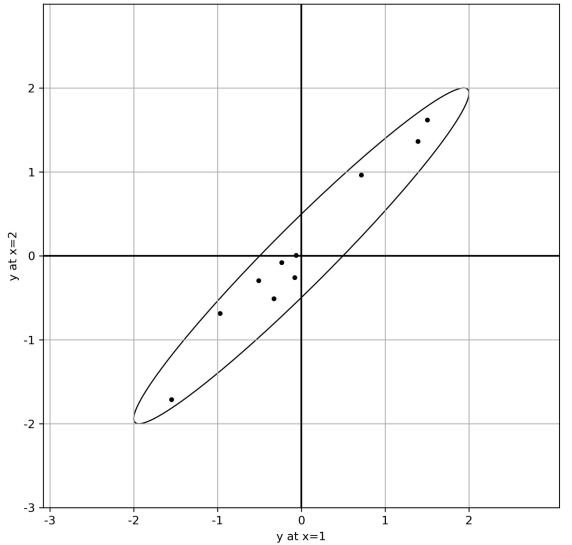
We have seen that we can describe the values at two points using a multivariate Gaussian distribution. Let's start using this to make predictions.

For example: Predict what y is, at x=2.

We'll start with **no observations**.

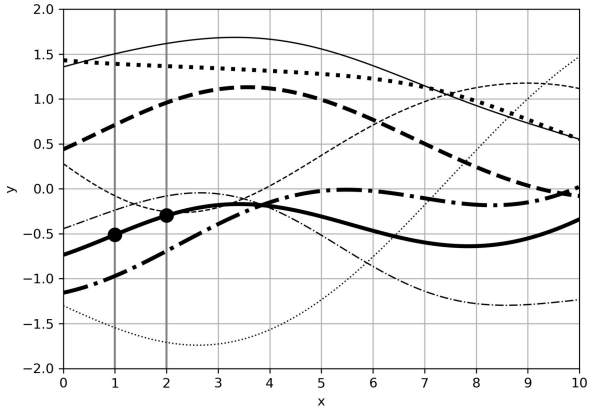
We can still use our assumption to make a prediction though!





Gaussian Process Regression

aka: How to make predictions using a Gaussian process



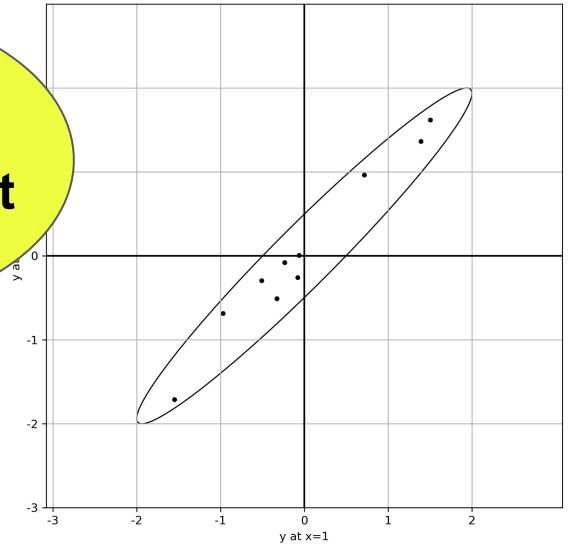
We have seen that we can describe the values at two points using a multivariate Gaussian distribution. Let's start using this to make predictions.

For example: Predict what y

We'll start with **no observations**.

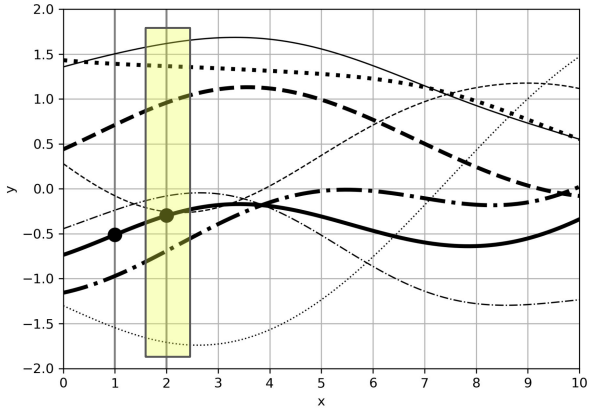
We can still use our assumption to make a prediction though!

Give a rough confidence interval for y at x=2.



Gaussian Process Regression

aka: How to make predictions using a Gaussian process



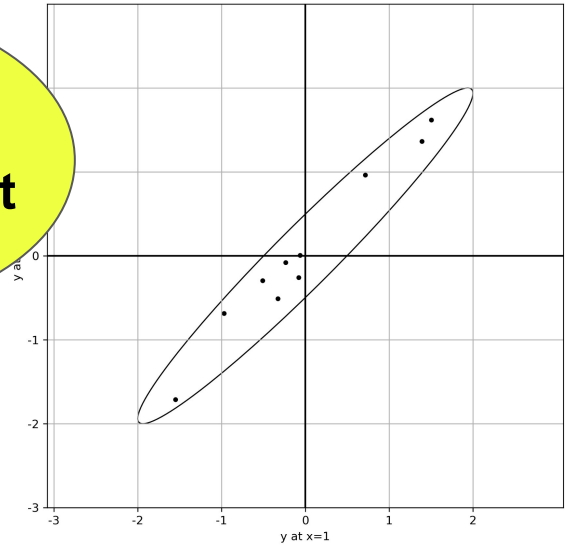
We have seen that we can describe the values at two points using a multivariate Gaussian distribution. Let's start using this to make predictions.

For example: Predict what y

We'll start with **no observations**.

We can still use our assumption to make a prediction though!

Give a rough confidence interval for y at x=2.

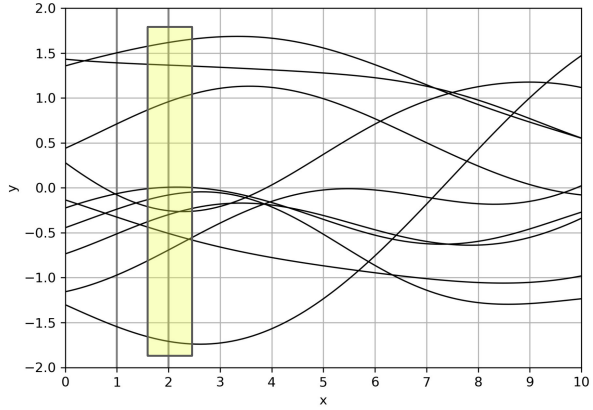


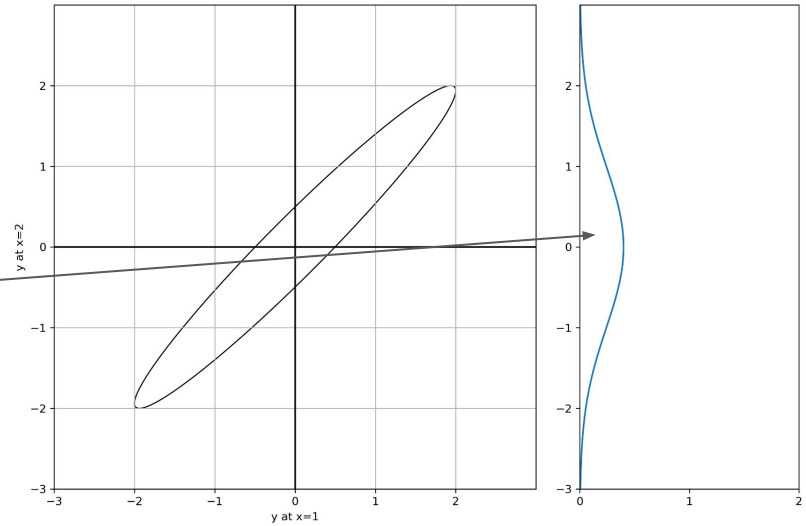
Gaussian Process Regression

We can see that, without any observations, our prediction for y at x=2 is very broad.

Note: If we plot the Gaussian between x=1 and x=2, we can see that the prediction for y at x=2 is the MARGINAL of the distribution.

The marginal of a Gaussian is also a Gaussian!

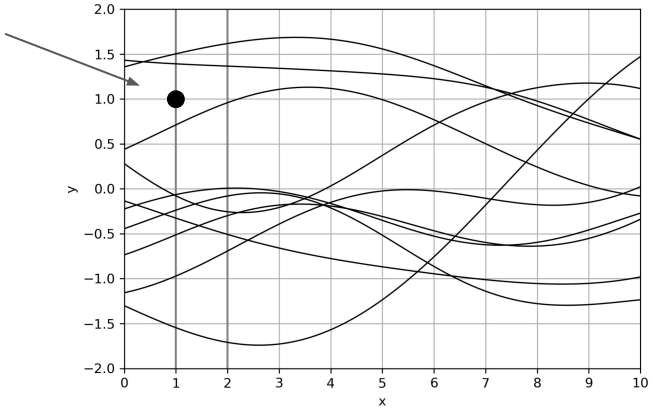


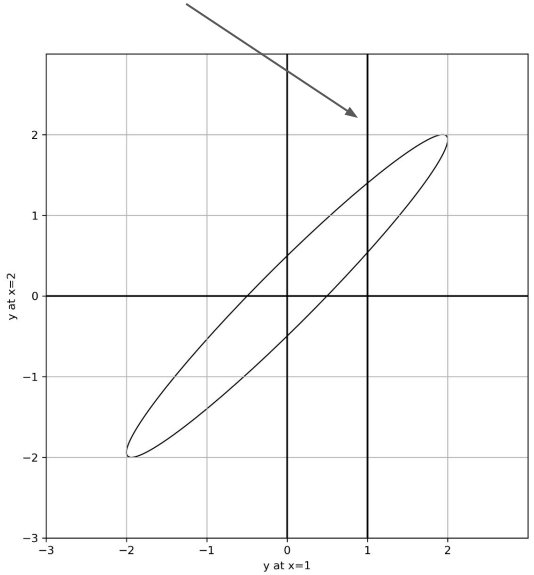


Gaussian Process Regression

What about if we have an observation that, at x=1, y=1…

What will this do to our prediction of y at x=2?



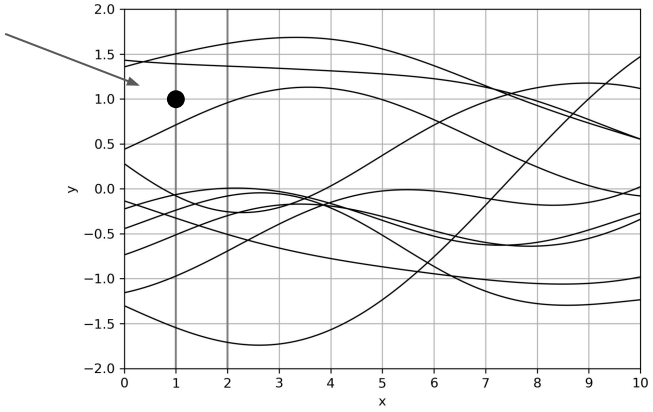


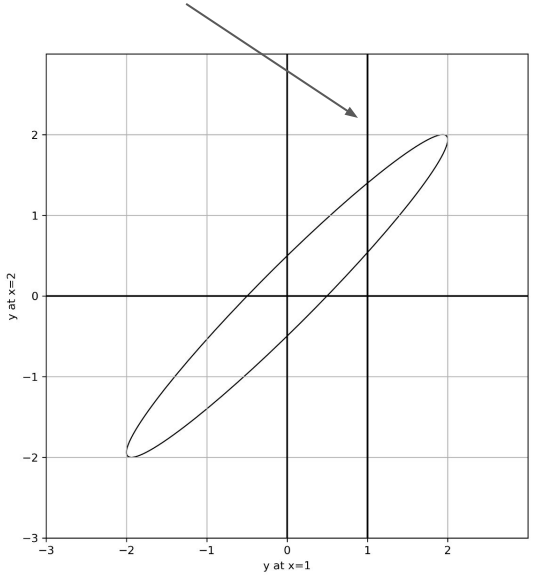
Gaussian Process Regression

What about if we have an observation that, at x=1, y=1…

What will this do to our prediction of y at x=2?

Activity 5





Gaussian Process Regression

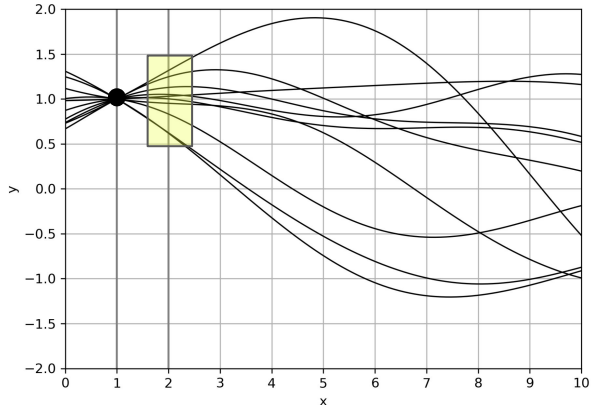
What about if we have an observation that, at x=1, y=1…

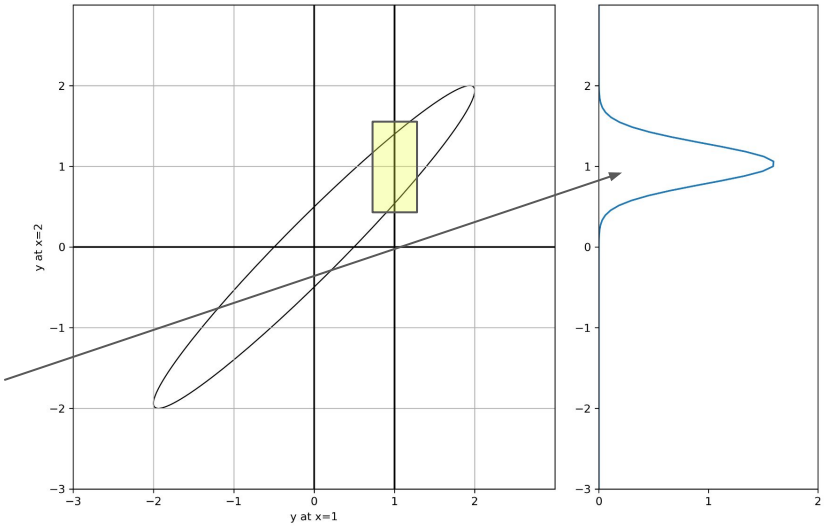
What will this do to our prediction of y at x=2?

- The variance in our prediction is less.
- The mean is now near 1, instead of 0.

We've found the conditional distribution of y, at x=2, given y=1 at x=1. I.e. $p(y_*|x,y,x_*)$.

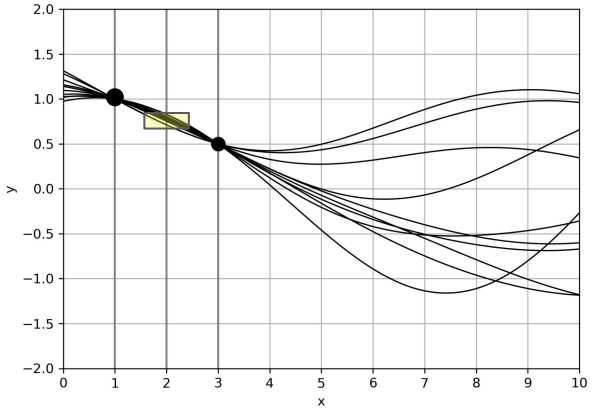
Notice that the conditional distribution is also a Gaussian!

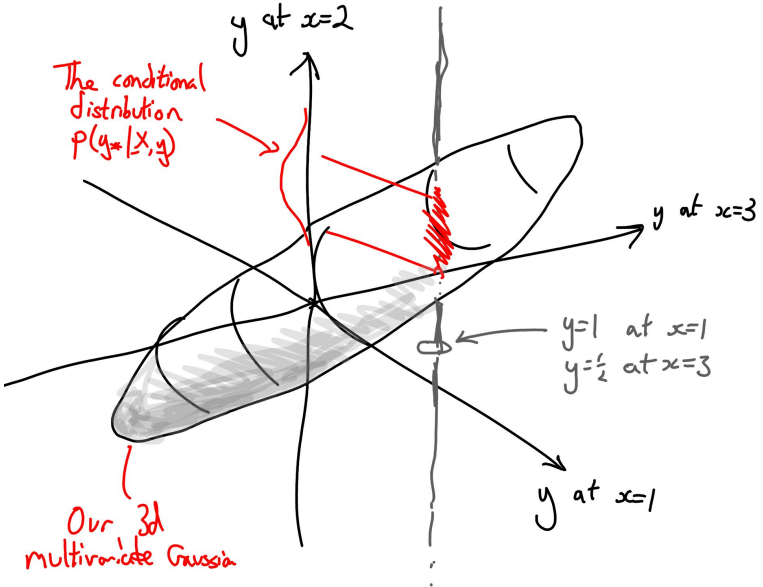




Gaussian Process Regression

If we had another observation (e.g. at x=3, y=½) and we want to predict y at x=2, we need to consider the three dimensional Gaussian between these three locations.





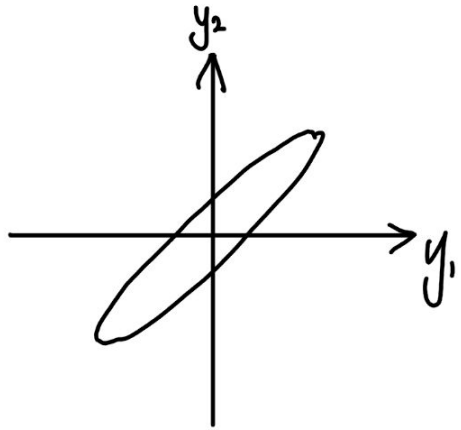
1. To make predictions we have to make assumptions
2. These assumptions can be described by the covariance between points.
3. The “Gaussian” in Gaussian process is the assumption that these different points are Gaussian distributed.
4. We can do prediction (with uncertainty) by CONDITIONING on the Gaussian distribution.

Returning to our simple 1-observation example.

Marginalising a Gaussian

Our prior is that the two locations are normally distributed, with a mean of zero, and a covariance of (for example) ¾ between them:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 3/4 \\ 3/4 & 1 \end{bmatrix} \right)$$

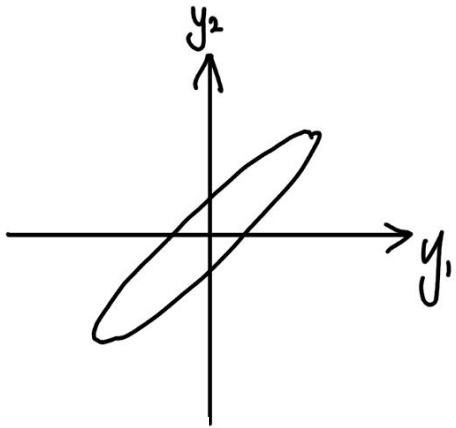


Q1 What is $p(y_2)$?

Marginalising a Gaussian

Our prior is that the two locations are normally distributed, with a mean of zero, and a covariance of (for example) ¾ between them:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \right)$$

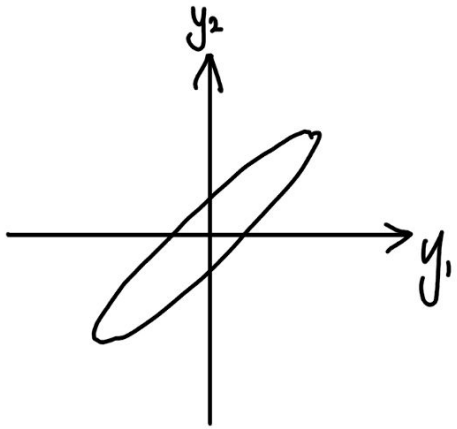


Q1 What is $p(y_2)$?

Marginalising a Gaussian

Our prior is that the two locations are normally distributed, with a mean of zero, and a covariance of (for example) ¾ between them:

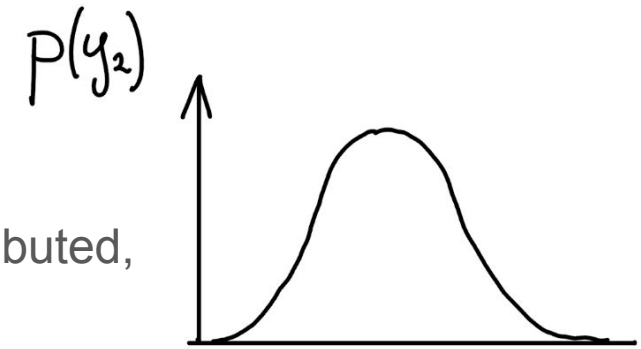
$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \right)$$



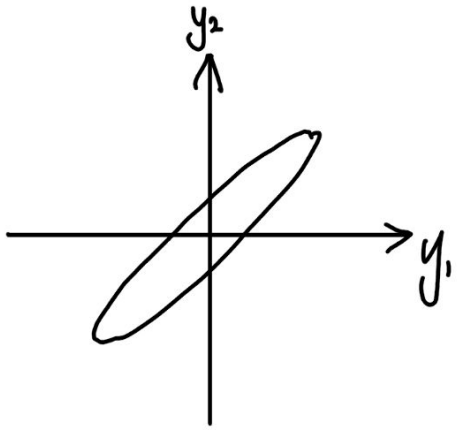
Q1 What is $p(y_2)$? $\longrightarrow$ $p(y_2) \sim N(0, 1)$

Marginalising a Gaussian

Our prior is that the two locations are normally distributed, and a covariance of (for example) ¾ between them:



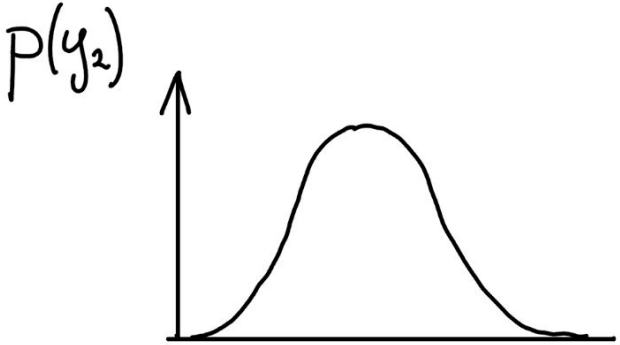
$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \right)$$



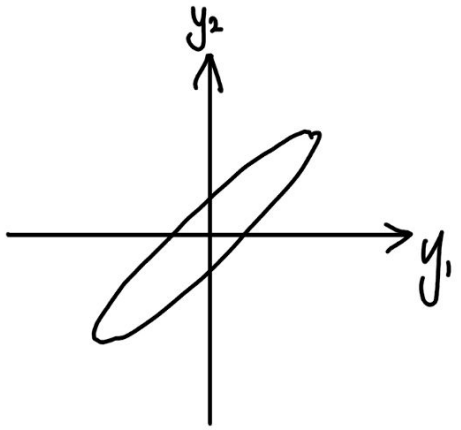
Q1 What is $p(y_2)$? $\longrightarrow$ $p(y_2) \sim N(0, 1)$

Marginalising a Gaussian

Marginalising a Gaussian is really easy, we just ignore the rest of the mean and covariance.



$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \cdot & \cdot \\ \cdot & 1 \end{bmatrix} \right)$$



Q1 What is $p(y_2)$? $\longrightarrow$ $p(y_2) \sim N(0, 1)$

Marginalising a Gaussian

Let $\mathbf{x}$ and $\mathbf{y}$ be jointly Gaussian random vectors

$$\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}A & C\\ C^\top & B\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\tilde{A} & \tilde{C}\\ \tilde{C}^\top & \tilde{B}\end{bmatrix}^{-1}\right)$$

then the *marginal* distribution of $\mathbf{x}$ is

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, A)$$

From C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

Conditioning with a Gaussian

Conditioning on a Gaussian is a little more tricky. But…

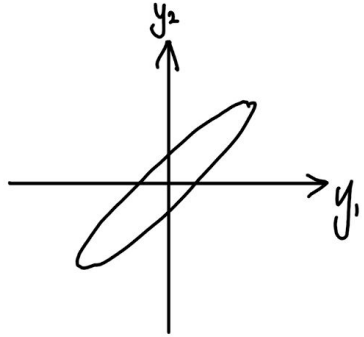
the conditional distribution of a Gaussian is itself a Gaussian!

We still need an expression for the mean and the (co)variance of the new Gaussian.

Q2 What is $p(y_2 | y_1 = 1)$?

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 3/4 \\ 3/4 & 1 \end{bmatrix} \right)$$

$$p(y_2 | y_1 = 1) = N(\mu, \Sigma)$$



$$\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$$

The mean of our prediction at y_2. $\longrightarrow \mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$

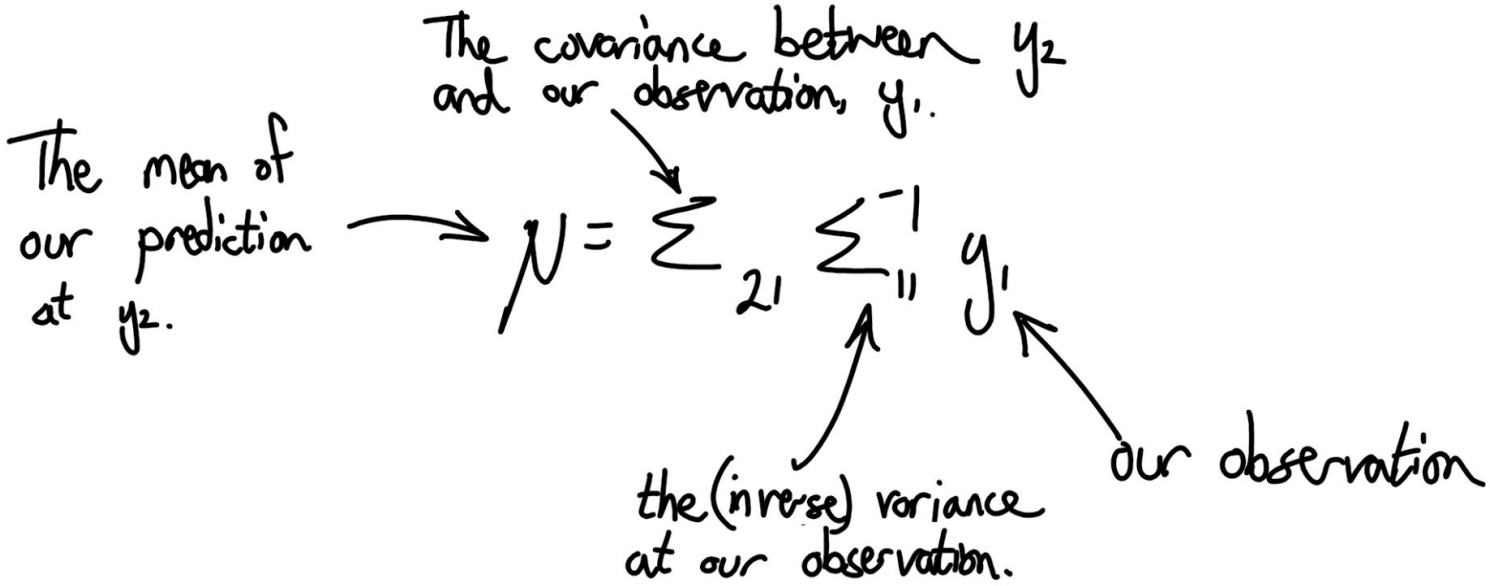
The mean of our prediction at y_2. → $\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$

y_1 ← our observation

The mean of our prediction at y_2. → $\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$

the (inverse) variance at our observation.

our observation


The covariance between y_2 and our observation, y_1.
The mean of our prediction at y_2.
$\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$
the (inverse) variance at our observation.
our observation

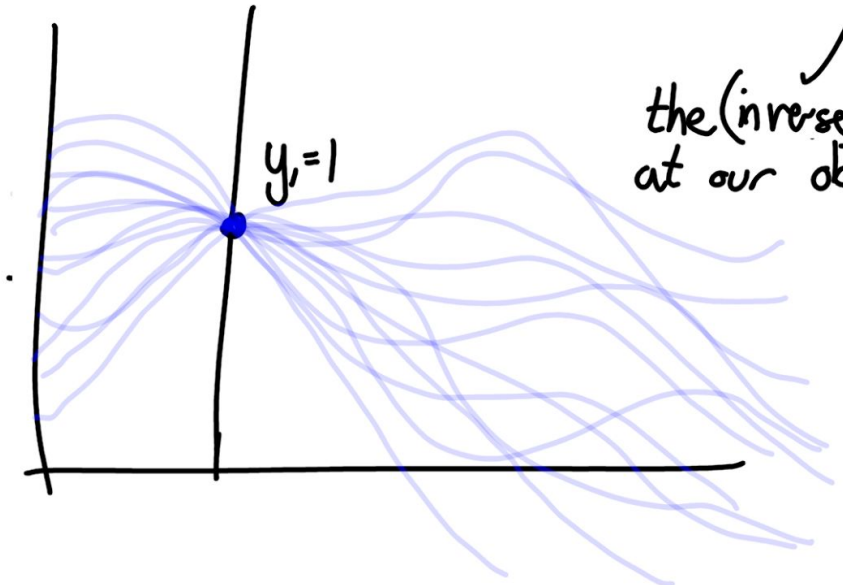
The covariance between y_2 and our observation, y_1.

The mean of our prediction at y_2.

$$\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$$

the (inverse) variance at our observation.

our observation



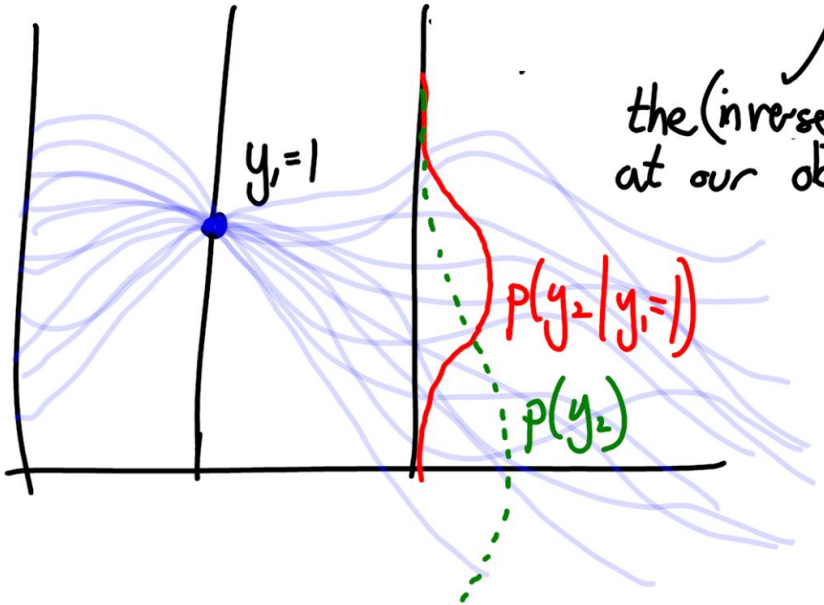
The covariance between y_2 and our observation, y_1.

The mean of our prediction at y_2.

$$\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$$

the (inverse) variance at our observation.

our observation



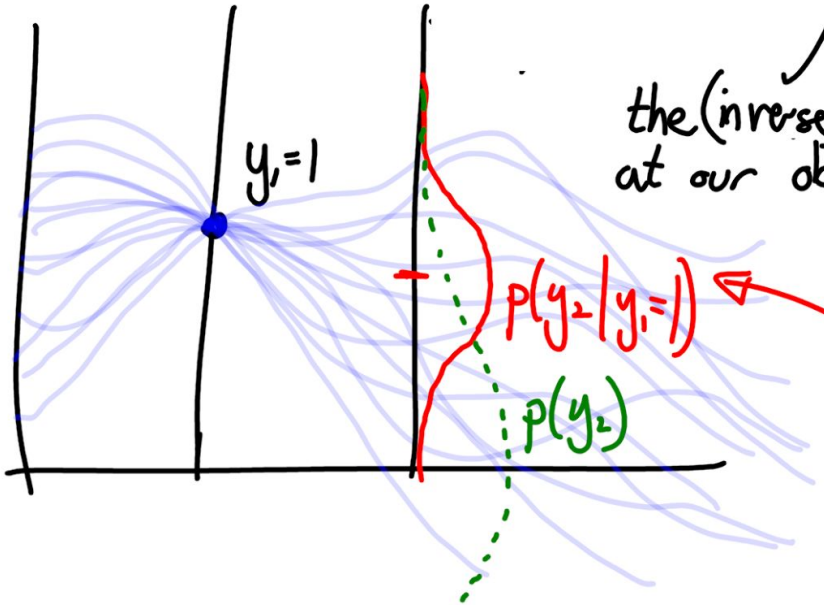
The covariance between y_2 and our observation, y_1.

The mean of our prediction at y_2.

$$\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$$

the (inverse) variance at our observation.

our observation



Notice our conditional distribution mean is a 'mix' between y_1 & the prior mean (0)

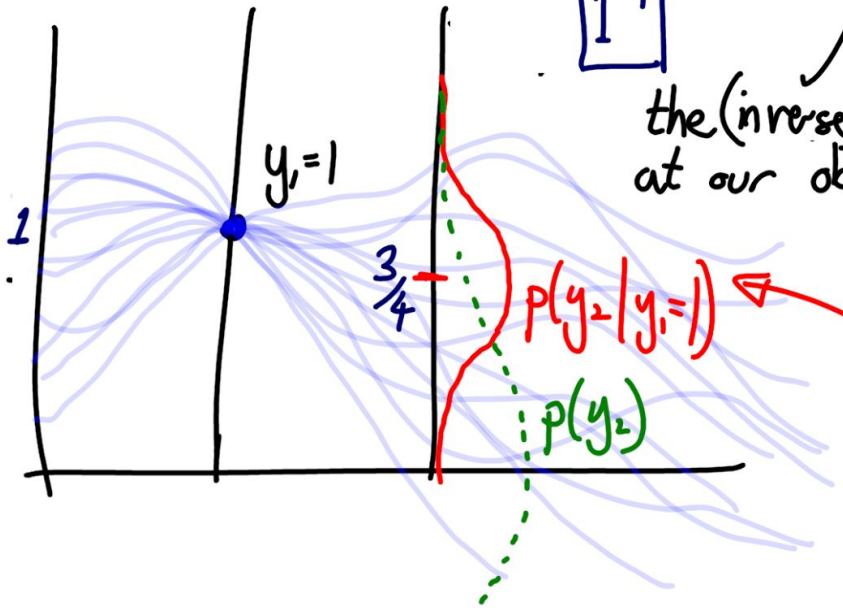
The covariance between y_2 and our observation, y_1. $\boxed{3/4}$

The mean of our prediction at y_2.

$$\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$$

$\boxed{1^{-1}}$ the (inverse) variance at our observation.

$\boxed{1}$ our observation



Notice our conditional distribution mean is a 'mix' between y_1 & the prior mean (0)

The covariance between y_2 and our observation, y_1.

$\frac{3}{4}$

The mean of our prediction at y_2.

$$\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$$

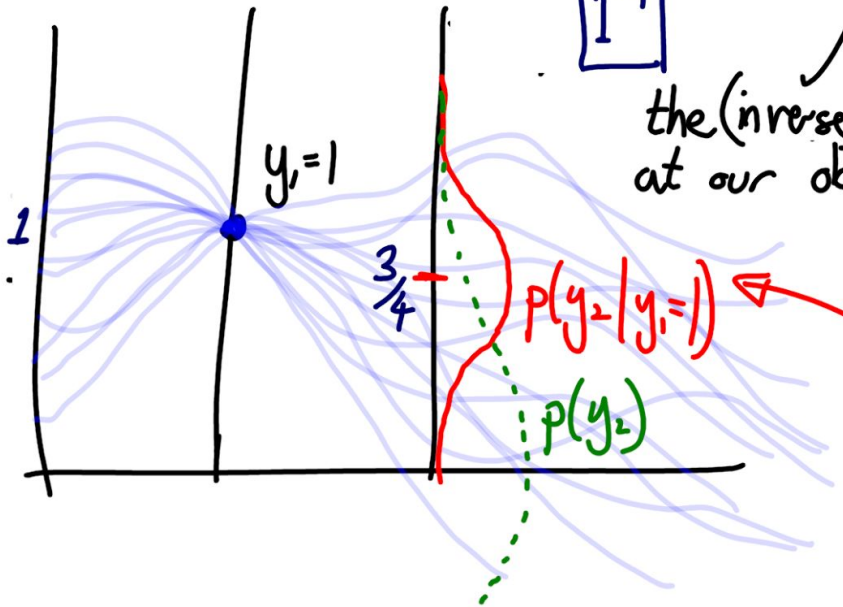
1^{-1}

the (inverse) variance at our observation.

our observation

1

$$\frac{3}{4} \times |1^{-1}| \times \frac{3}{4}$$



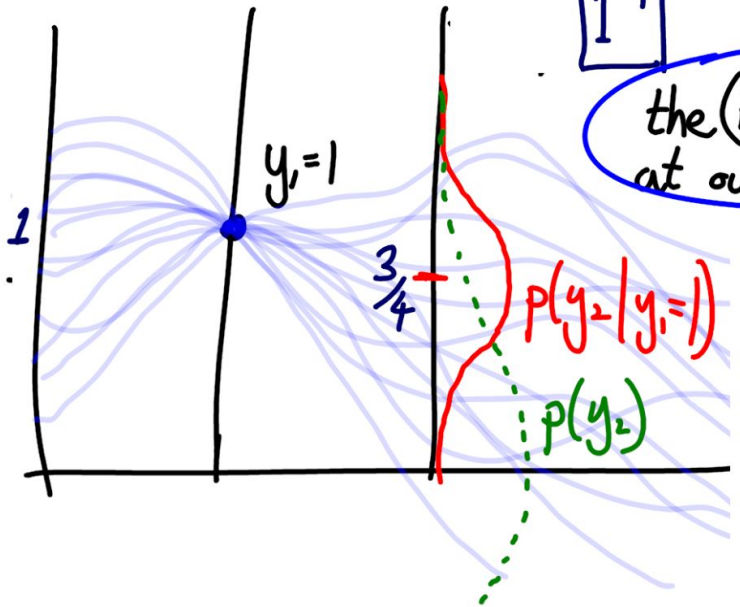
Notice our conditional distribution mean is a 'mix' between y_1 & the prior mean (0)

The covariance between y_2 and our observation, y_1. $\boxed{3/4}$

The mean of our prediction at y_2. $\rightarrow \mu = \Sigma_{21} \boxed{\Sigma_{11}^{-1}} y_1 \rightarrow \frac{3}{4} \times \boxed{1^{-1}} \times \boxed{1} = \frac{3}{4}$

$\boxed{1^{-1}}$ the (inverse) variance at our observation.

$\boxed{1}$ our observation



INTUITION

If there is lots of variance at our observation we should put less weight on the observation.

The covariance between y_2 and our observation, y_1.

$\frac{3}{4}$

The mean of our prediction at y_2.

$$\mu = \Sigma_{21} \Sigma_{11}^{-1} y_1$$

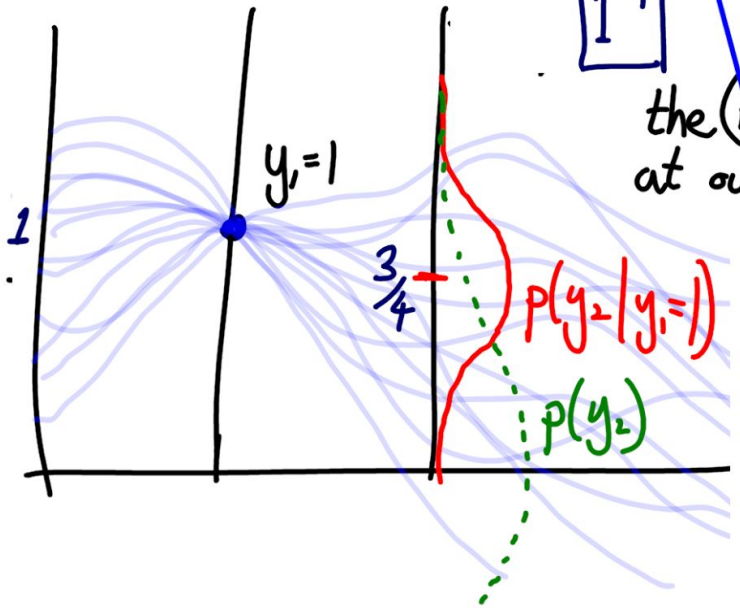
$$\frac{3}{4} \times |1^{-1}| \times \frac{3}{4}$$

1^{-1}

the (inverse) variance at our observation.

1

our observation



INTUITION

If there is more connection between the observation and our test point, we should weight the observation more.

We usually use the following notation.

$$\mu = \underline{K}_{*f} K_{ff}^{-1} \underline{y}$$

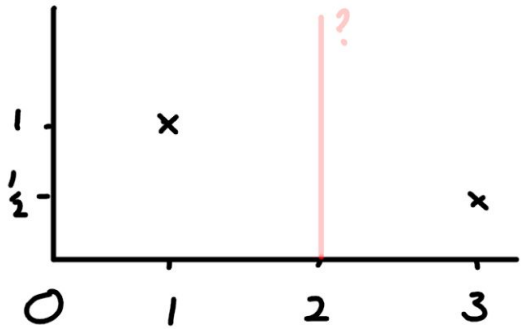
Covariance between test point and training points

Covariance between training points

An example.

Our observations: $X = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ $\underline{y} = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$

Our test location: $\underline{x}_* = [2]$



We want:

$p(\underline{y}_* | X, \underline{y}, \underline{x}_*)$

We need to use this expression to find the posterior mean, but...

$$\mu = \underline{k}_{*f} K_{ff}^{-1} \underline{y}$$

Covariance between test point and training points

Covariance between training points

$$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \underline{y} = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

$$\underline{x}_* = [2]$$

We have seen that we encode our assumptions/priors in the covariance between points. But how do we decide on what the covariance is?

Covariance Function

We need a function to tell us what the covariance is, between any given pair of inputs.

This is called the *covariance function* or *kernel*.

The choice of covariance function depends on our assumptions.

E.g. A common assumption is that nearby points are more similar than distant points. An example covariance function:

$$k(x_1, x_2) = e^{-\frac{|x_1 - x_2|}{l}}$$

Covariance Function

An example covariance function:

$$k(x_1, x_2) = e^{-\frac{|x_1 - x_2|}{l}}$$

This is the **exponential covariance** function.

The ℓ is the lengthscale of the kernel. It controls how quickly the covariance will decay. Imagine if ℓ were a large number, then the exponent would be really small, so the covariance will be nearer to one.

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale of 4.

The kernel $k(x_1, x_2) = e^{\frac{-|x_1 - x_2|}{l}}$

The covariance between the test point and the training points. $\underline{k}_{*f} = \begin{bmatrix} k(x_*, x_1) & k(x_*, x_2) \end{bmatrix}$

The covariance between the training points. $K_{ff} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) \\ k(x_2, x_1) & k(x_2, x_2) \end{bmatrix}$

Our training data

$$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \underline{y} = \begin{bmatrix} 1 \\ 1/2 \end{bmatrix}$$

Test point location

$$\underline{x}_* = \begin{bmatrix} 2 \end{bmatrix}$$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale of 4.

The kernel $k(x_1, x_2) = e^{\frac{-|x_1 - x_2|}{l}}$

The covariance between the test point and the training points. $\underline{k}_{*f} = \begin{bmatrix} e^{\frac{-|2-1|}{4}} & e^{\frac{-|2-3|}{4}} \end{bmatrix}$

The covariance between the training points. $K_{ff} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) \\ k(x_2, x_1) & k(x_2, x_2) \end{bmatrix}$

Our training data

$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \underline{y} = \begin{bmatrix} 1 \\ 1/2 \end{bmatrix}$

Test point location

$\underline{x}_* = [2]$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale of 4.

The kernel $k(x_1, x_2) = e^{-\frac{|x_1 - x_2|}{l}}$

The covariance between the test point and the training points. $\underline{k}_{*f} = \begin{bmatrix} 0.78 & 0.78 \end{bmatrix}$

The covariance between the training points. $K_{ff} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) \\ k(x_2, x_1) & k(x_2, x_2) \end{bmatrix}$

Our training data

$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \underline{y} = \begin{bmatrix} 1 \\ 1/2 \end{bmatrix}$

Test point location

$\underline{x}_* = \begin{bmatrix} 2 \end{bmatrix}$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale of 4.

The kernel $k(x_1, x_2) = e^{-\frac{|x_1 - x_2|}{l}}$

The covariance between the test point and the training points. $\underline{k}_{*f} = \begin{bmatrix} 0.78 & 0.78 \end{bmatrix}$

The covariance between the training points. $K_{ff} = \begin{bmatrix} e^{-\frac{|1-1|}{4}} & e^{-\frac{|1-3|}{4}} \\ e^{-\frac{|3-1|}{4}} & e^{-\frac{|3-3|}{4}} \end{bmatrix}$

Our training data

$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \underline{y} = \begin{bmatrix} 1 \\ 1/2 \end{bmatrix}$

Test point location

$\underline{x}_* = \begin{bmatrix} 2 \end{bmatrix}$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale of 4.

The kernel $k(x_1, x_2) = e^{\frac{-|x_1 - x_2|}{l}}$

The covariance between the test point and the training points. $\underline{k}_{*f} = \begin{bmatrix} 0.78 & 0.78 \end{bmatrix}$

The covariance between the training points. $K_{ff} = \begin{bmatrix} 1 & 0.61 \\ 0.61 & 1 \end{bmatrix}$

Our training data

$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \underline{y} = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$

Test point location

$\underline{x}_* = \begin{bmatrix} 2 \end{bmatrix}$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale

Our training data

$y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$

The kernel

The covariance between the te point and the training points.

The covariance between the training points.

$K_{ff} = \begin{bmatrix} 0.61 & 1 \end{bmatrix}$

```
$ python
>>>
```

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale

Our training data

$$y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

The kernel

The covariance between the te point and the training points.

The covariance between the training points.

$$K_{ff} = \begin{bmatrix} & \\ 0.61 & 1 \end{bmatrix}$$

```
>>> import numpy as np
```

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale

Our training data

$$y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

```
>>> import numpy as np
>>> X = np.array([1,3])
```

The kernel

The covariance between the te point and the training points.

The covariance between the training points.

$$K_{ff} = \begin{bmatrix} & \\ 0.61 & 1 \end{bmatrix}$$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale

```
>>> import numpy as np
>>> X = np.array([1,3])
>>> y = np.array([1,.5])
```

Our training data

$y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$

The kernel

The covariance between the te point and the training points.

The covariance between the training points.

$K_{ff} = \begin{bmatrix} & \\ 0.61 & 1 \end{bmatrix}$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale

```
>>> import numpy as np
>>> X = np.array([1,3])
>>> y = np.array([1,.5])
>>> xstar = np.array([2])
```

Our training data

$y = \begin{bmatrix} 1 \\ 1/2 \end{bmatrix}$

The kernel

The covariance between the te point and the training points.

The covariance between the training points.

$K_{ff} = \begin{bmatrix} & 0.61 & & 1 & \end{bmatrix}$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale

```
>>> import numpy as np
>>> X = np.array([1,3])
>>> y = np.array([1,.5])
>>> xstar = np.array([2])
>>> Kff = np.exp(-np.abs(X[:,None] - X[None,:])/4)
```

Our training data

$y = \begin{bmatrix} 1 \\ 1/2 \end{bmatrix}$

The kernel

The covariance between the te point and the training points.

The covariance between the training points.

$K_{ff} = \begin{bmatrix} & \\ 0.61 & 1 \end{bmatrix}$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale

Our training data

$$y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

```
>>> import numpy as np
>>> X = np.array([1,3])
>>> y = np.array([1,.5])
>>> xstar = np.array([2])
>>> Kff = np.exp(-np.abs(X[:,None] - X[None,:])/4)
>>> Kstarf = np.exp(-np.abs(xstar - X[None,:])/4)
```

The kernel

The covariance between the te point and the training points.

The covariance between the training points.

$$K_{ff} = \begin{bmatrix} & \\ 0.61 & 1 \end{bmatrix}$$

For our example, compute k_{*f} and K_{ff}, using an exponential covariance function with a lengthscale

Our training data

$$y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

```
>>> import numpy as np
>>> X = np.array([1,3])
>>> y = np.array([1,.5])
>>> xstar = np.array([2])
>>> Kff = np.exp(-np.abs(X[:,None] - X[None,:])/4)
>>> Kstarf = np.exp(-np.abs(xstar - X[None,:])/4)
>>> Kstarf@np.linalg.inv(Kff)@y
array([0.72715772])
```

The kernel

The covariance between the te point and the training points.

The covariance between the training points.

$$K_{ff} = \begin{bmatrix} & \\ 0.61 & 1 \end{bmatrix}$$

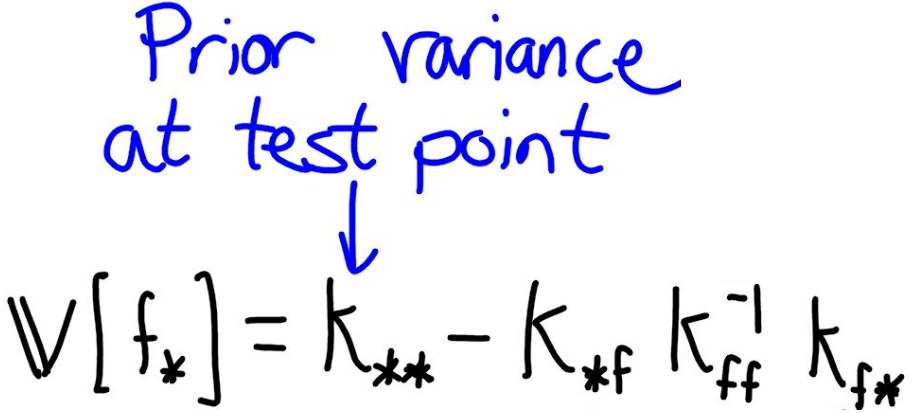
Posterior Covariance

The variance of our prediction is,

$$\mathbb{V}[f_*] = K_{**} - K_{*f} K_{ff}^{-1} K_{f*}$$

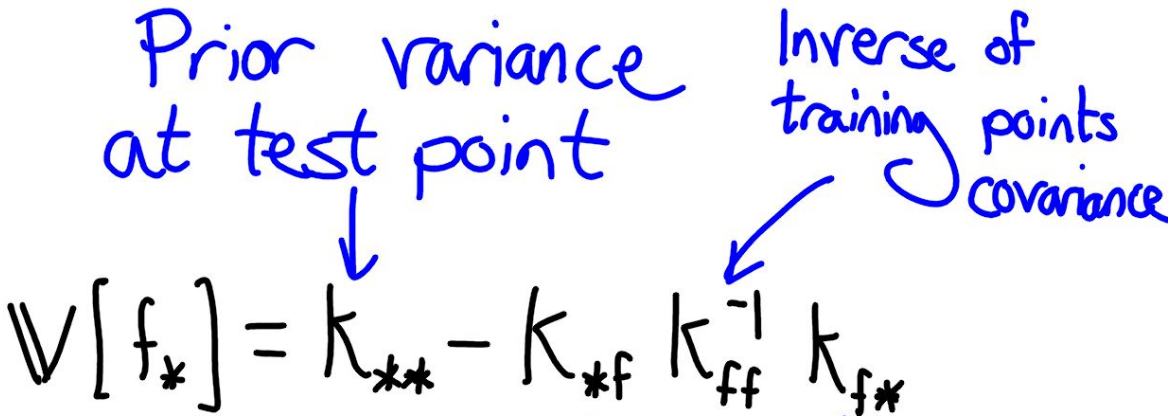
Posterior Covariance

The variance of our prediction is,



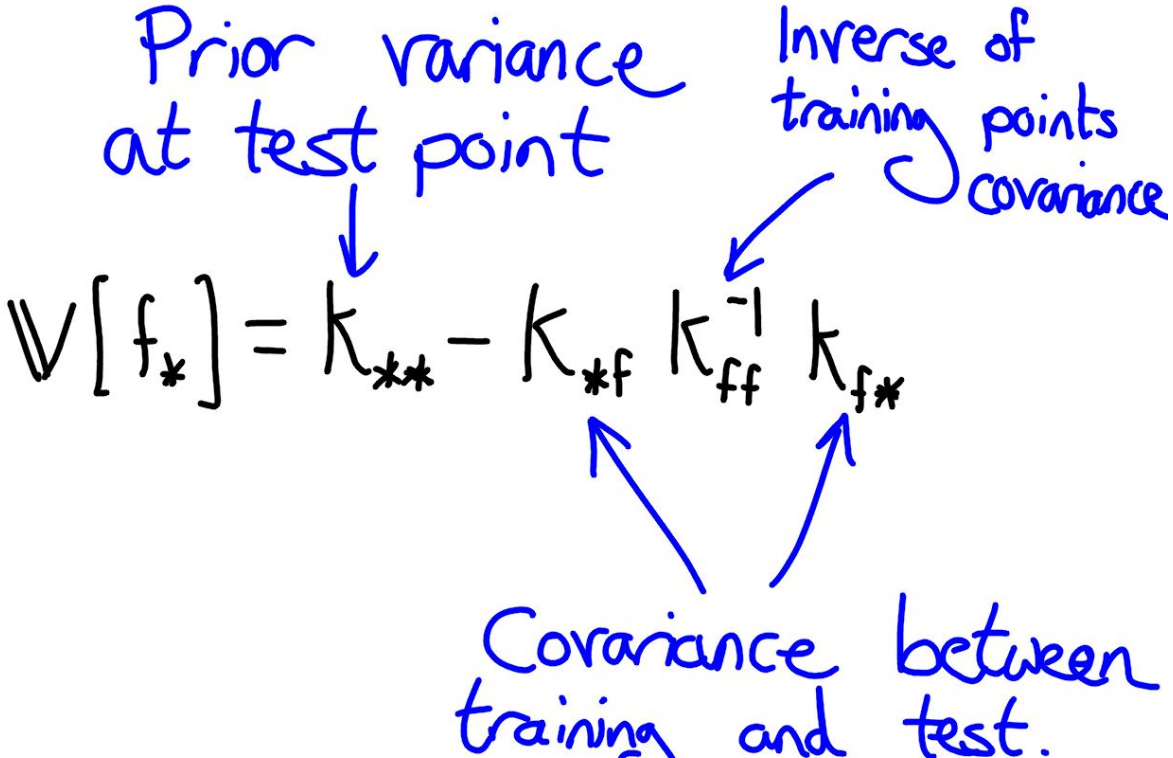
Posterior Covariance

The variance of our prediction is,

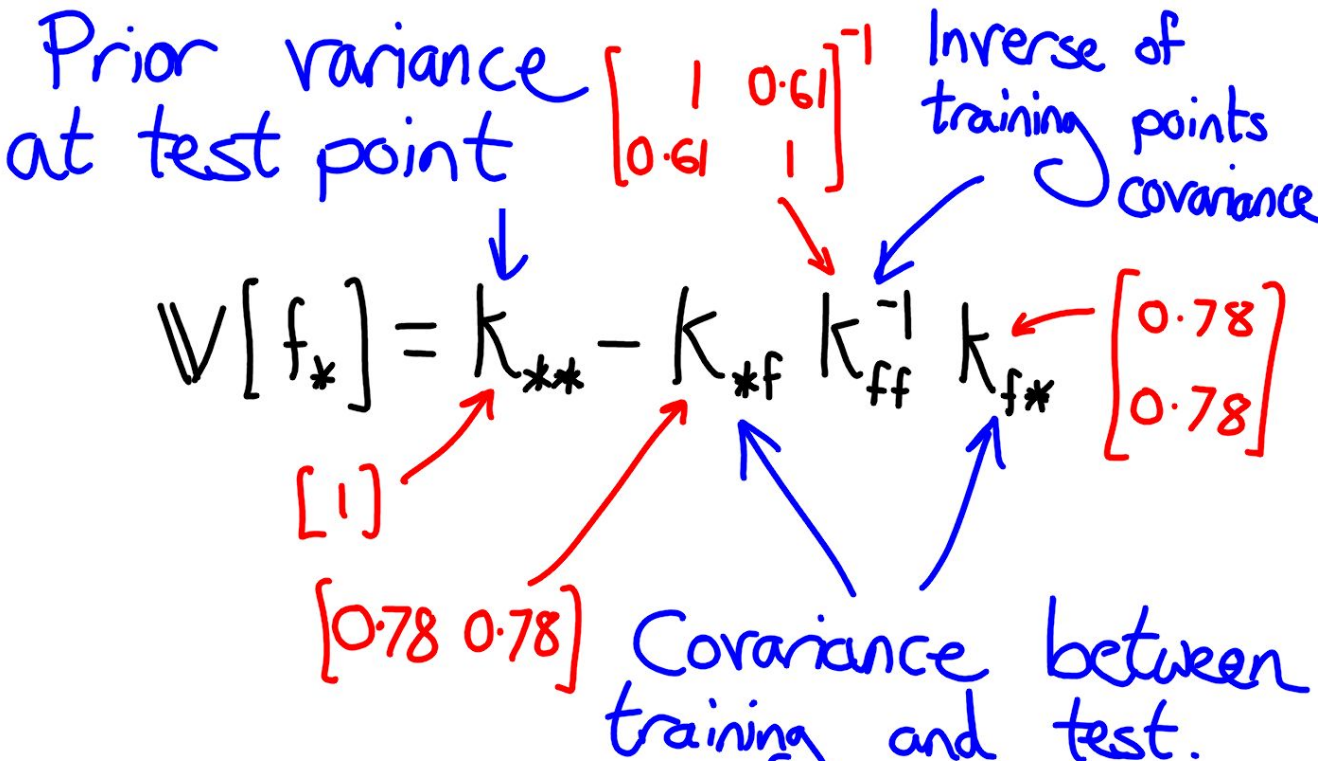


Posterior Covariance

The variance of our prediction is,

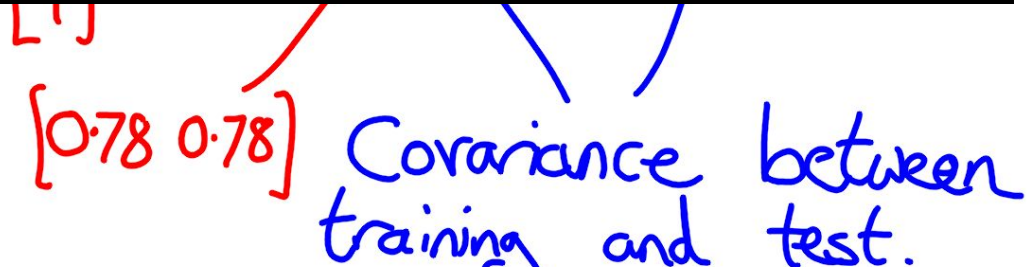


So we know k_{*f} and k_{ff}. So we can now compute the mean of our posterior.



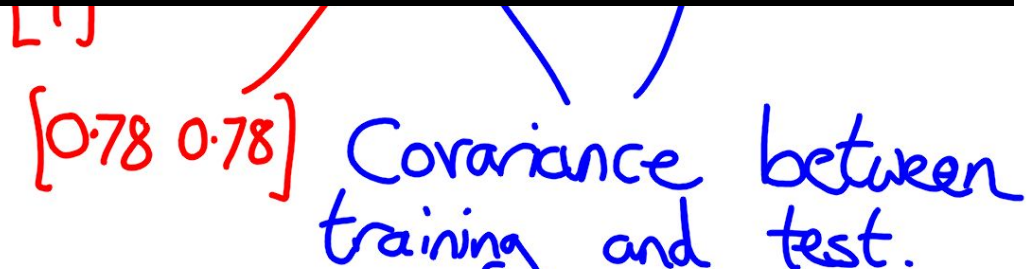
So we know k_{*f} and k_{ff}. So we can now compute the mean of our posterior.

```
>>> import numpy as np
>>> X = np.array([1,3])
>>> y = np.array([1,.5])
>>> xstar = np.array([2])
>>> Kff = np.exp(-np.abs(X[:,None] - X[None,:])/4)
>>> Kstarf = np.exp(-np.abs(xstar - X[None,:])/4)
>>> Kstar@np.linalg.inv(Kff)@y
array([0.72715772])
>>> Kstarstar = np.array([1])
```



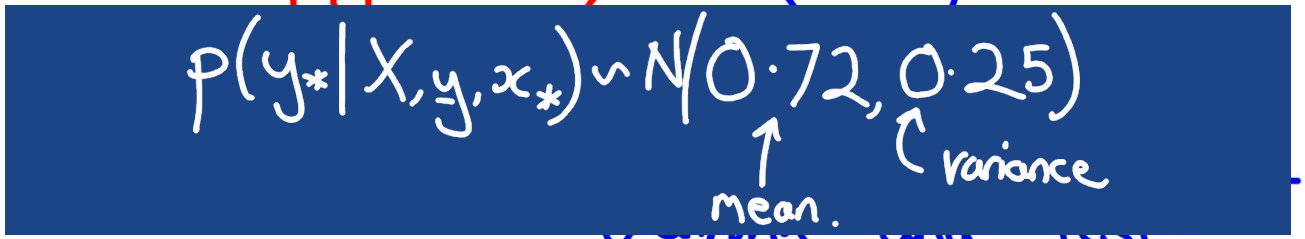
So we know k_{*f} and k_{ff}. So we can now compute the mean of our posterior.

```
>>> import numpy as np
>>> X = np.array([1,3])
>>> y = np.array([1,.5])
>>> xstar = np.array([2])
>>> Kff = np.exp(-np.abs(X[:,None] - X[None,:])/4)
>>> Kstarf = np.exp(-np.abs(xstar - X[None,:])/4)
>>> Kstar@np.linalg.inv(Kff)@y
array([0.72715772])
>>> Kstarstar = np.array([1])
>>> Kstarstar - Kstarf @ np.linalg.inv(Kff) @ Kstarf.T
array([[0.24491866]])
```

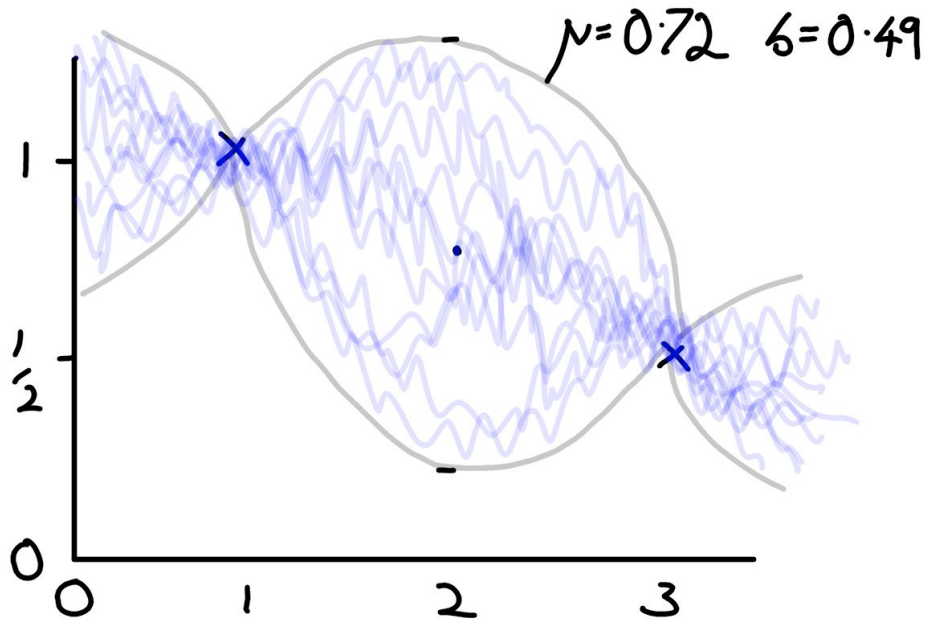


So we know k_{*f} and k_{ff}. So we can now compute the mean of our posterior.

```
>>> import numpy as np
>>> X = np.array([1,3])
>>> y = np.array([1,.5])
>>> xstar = np.array([2])
>>> Kff = np.exp(-np.abs(X[:,None] - X[None,:])/4)
>>> Kstarf = np.exp(-np.abs(xstar - X[None,:])/4)
>>> Kstar@np.linalg.inv(Kff)@y
array([0.72715772])
>>> Kstarstar = np.array([1])
>>> Kstarstar - Kstarf @ np.linalg.inv(Kff) @ Kstarf.T
array([[0.24491866]])
```



So we know k_{*f} and k_{ff}. So we can now compute the mean of our posterior.

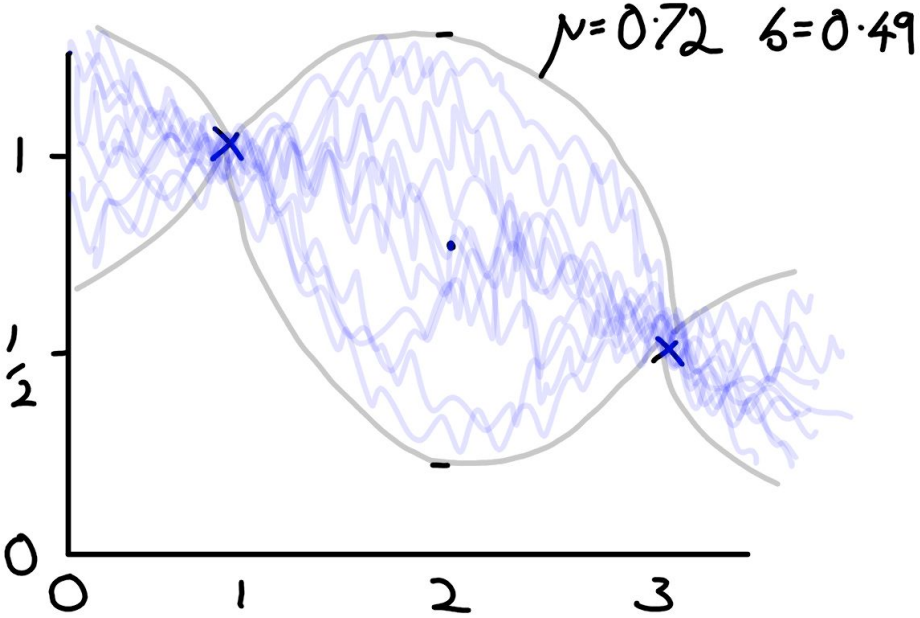


$$p(y_* | X, y, x_*) \sim N(0.72, 0.25)$$

mean. variance

So we know k_{*f} and k_{ff}. So we can now compute the mean of our posterior.

The prediction is very broad, with quite a large variance. This is because we chose the exponential kernel, which is not at all smooth.



$$p(y_* | X, y, x_*) \sim N(0.72, 0.25)$$

mean. variance

1. To make predictions we have to make assumptions
2. These assumptions can be described by the covariance between points.
3. A Gaussian process assumes that different location values are Gaussian distributed.
4. We can do prediction by conditioning on the Gaussian distribution.
5. We need to specify a covariance function so we can determine the covariance between pairs of locations.

Activity 6

Let's compute the prediction for the same problem, but this time using the exponentiated quadratic kernel, with lengthscale = 4.

Activity 6

$$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

$$x_* = [2]$$

$$k(x_1, x_2) = e^{-\frac{(x_1 - x_2)^2}{2l^2}}$$

The exponentiated quadratic kernel.
(also called the squared exponential, or Gaussian kernel)

Use lengthscale = 4

What is the mean and covariance of the prediction?

Reminder...

$$\mu = K_{*f} K_{ff}^{-1} y$$

$$\mathbb{V}[f_*] = K_{**} - K_{*f} K_{ff}^{-1} K_{f*}$$

Activity 6

$$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

$$x_* = [2]$$

$$k(x_1, x_2) = e^{-\frac{(x_1 - x_2)^2}{2\ell^2}}$$

As you don't all have a laptop, here are the values of the matrices you'll need….

$$k_{*f} = [0.97 \quad 0.97]$$

$$k_{ff} = \begin{bmatrix} 1 & 0.88 \\ 0.88 & 1 \end{bmatrix}$$

$$k_{**} = [1]$$

Reminder…

$$\mu = k_{*f} k_{ff}^{-1} y$$

$$\mathbb{V}[f_*] = k_{**} - k_{*f} k_{ff}^{-1} k_{f*}$$

Activity 6

$$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

$$x_* = [2]$$

$$k(x_1, x_2) = e^{-\frac{(x_1 - x_2)^2}{2\ell^2}}$$

I'll invert the covariance matrix for you, too.

$$k_{*f} = [0.97 \quad 0.97]$$

$$k_{ff}^{-1} = \begin{bmatrix} 4.52 & -3.99 \\ -3.99 & 4.52 \end{bmatrix}$$

$$k_{**} = [1]$$

Reminder...

$$\mu = k_{*f} k_{ff}^{-1} y$$

$$\mathbb{V}[f_*] = k_{**} - k_{*f} k_{ff}^{-1} k_{f*}$$

Activity 6

$$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

$$x_* = [2]$$

$$k(x_1, x_2) = e^{-\frac{(x_1 - x_2)^2}{2\ell^2}}$$

I'll invert the covariance matrix for you, too.

$$k_{*f} = [0.97 \quad 0.97]$$

$$k_{ff}^{-1} = \begin{bmatrix} 4.52 & -3.99 \\ -3.99 & 4.52 \end{bmatrix}$$

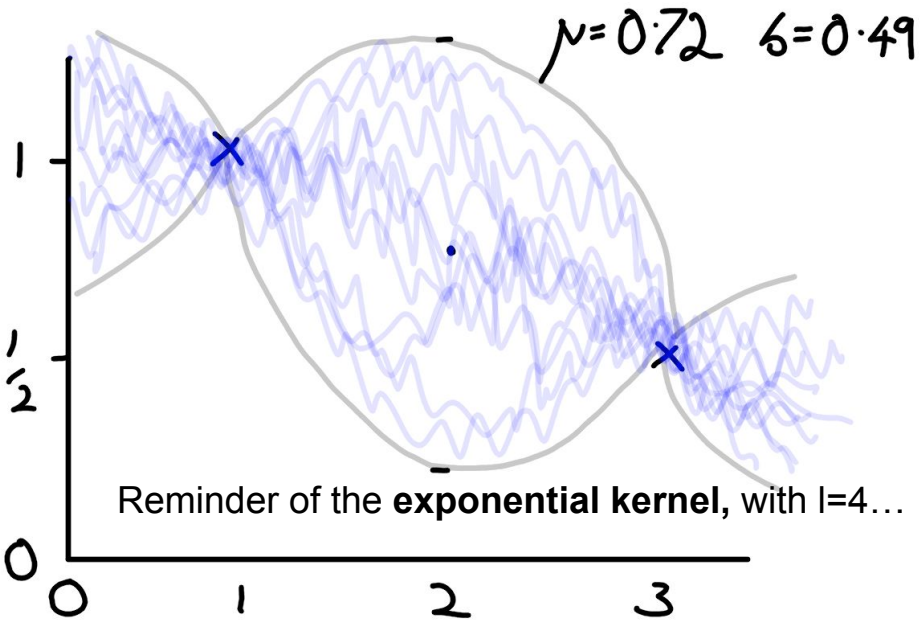
$$k_{**} = [1]$$

Reminder...

$$\mu = (0.97 \; 0.97) \begin{pmatrix} 1 & 0.88 \\ 0.88 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix} = 0.77$$

$$\mathbb{V}[f_*] = (1) - (0.97 \; 0.97) \begin{pmatrix} 1 & 0.88 \\ 0.88 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0.97 \\ 0.97 \end{pmatrix} = 0.0019$$

We assumed, by using the SE kernel, that our function is very smooth, which means there is little uncertainty in the posterior.

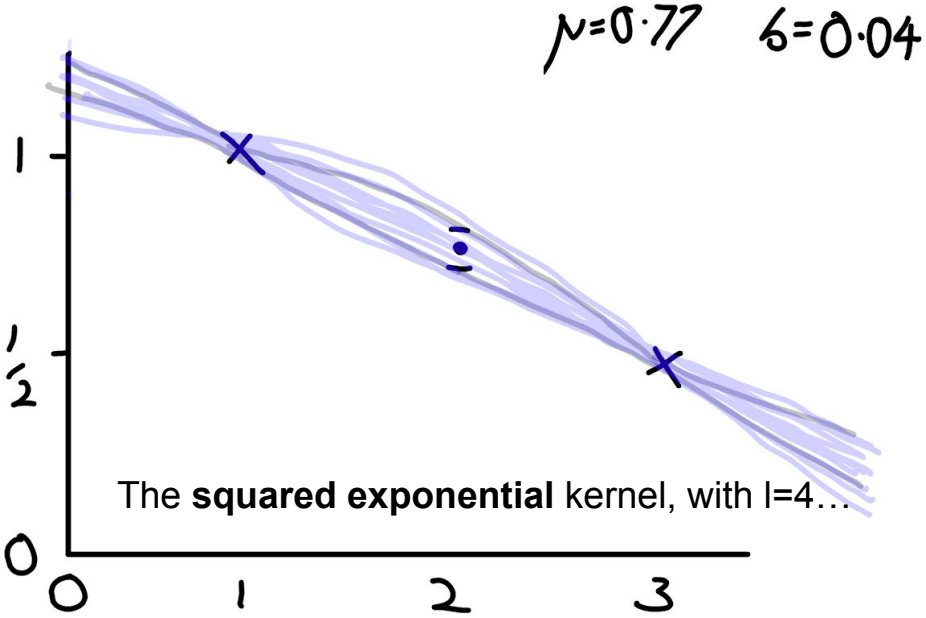


Reminder of the **exponential kernel,** with l=4…

$$p(y_* \mid X, y, x_*) \sim N(0.72, 0.25)$$

mean. variance

We assumed, by using the SE kernel, that our function is very smooth, which means there is little uncertainty in the posterior.



$$p(y_* \mid X, y, x_*) \sim N(0.77, 0.0019)$$

mean. variance

(skip slide! included for reference)

Summary / for Reference: Rules of Marginalising and Conditioning a Gaussian

Let $\mathbf{x}$ and $\mathbf{y}$ be jointly Gaussian random vectors

$$\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}A & C\\ C^\top & B\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\tilde{A} & \tilde{C}\\ \tilde{C}^\top & \tilde{B}\end{bmatrix}^{-1}\right)$$

then the *marginal* distribution of $\mathbf{x}$ and the *conditional* distribution of $\mathbf{x}$ given $\mathbf{y}$ are

$$\begin{aligned}\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, A), \quad \text{and} \quad \mathbf{x}|\mathbf{y} &\sim \mathcal{N}\big(\boldsymbol{\mu}_x + CB^{-1}(\mathbf{y}-\boldsymbol{\mu}_y),\ A - CB^{-1}C^\top\big)\\ \text{or} \quad \mathbf{x}|\mathbf{y} &\sim \mathcal{N}\big(\boldsymbol{\mu}_x - \tilde{A}^{-1}\tilde{C}(\mathbf{y}-\boldsymbol{\mu}_y),\ \tilde{A}^{-1}\big). \qquad \text{(A.6)}\end{aligned}$$

From C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

Gaussian Process Regression: Summary

The “conditioning on an input” rule of a Gaussian leads to these two expressions for the mean and variance of a prediction, if we assume our data is generated by a Gaussian process.

$$\bar{f}_* = \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y},$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}_*.$$

K is K_{ff} (covariance between training points)

This term is because our observations might be noisy (i.e. not of the actual underlying function). σ_n^2 is the variance of the noise in the observations.

The Kernel Trick

Until now I've tried to explain GPs in a fairly 'practical' way, but it's important to understand their relationship to linear regression, features and **the kernel trick**.

Remember in the last lecture,
We computed the weights with:

$$w = (X^T X)^{-1} X y$$

We could make predictions by computing X_* for the test point, so our prediction is:

$$y_* = X_* w$$

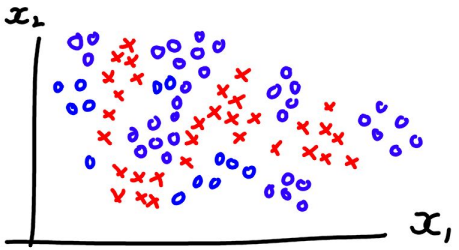
Remember in the last lecture,
We computed the weights with:

$$w = (X^T X)^{-1} X y$$

We could make predictions by computing X_* for the test point, so our prediction is:

$$y_* = X_* (X^T X)^{-1} X^T y$$

Imagine we have some data:



We could use a Gaussian basis to help,

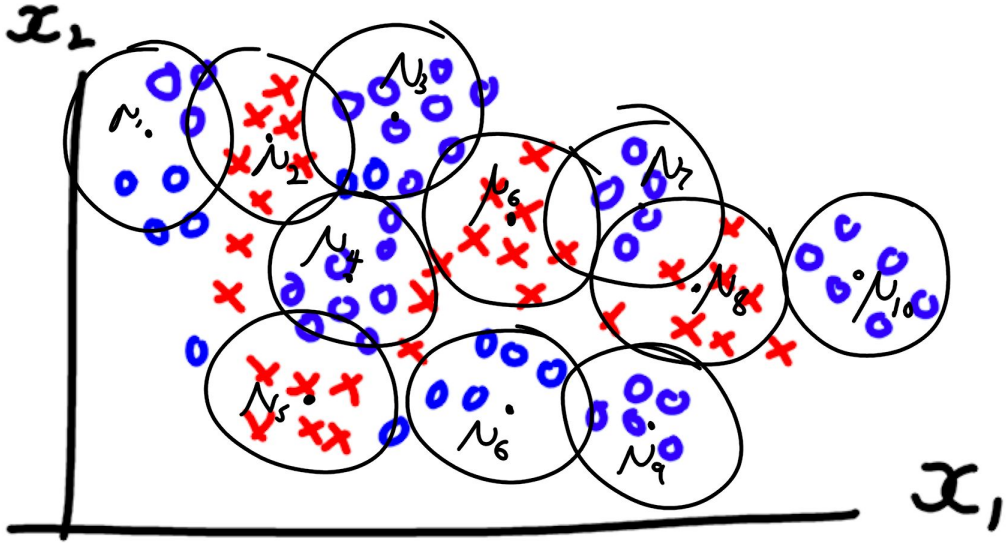
$$\phi(\underline{x}) = \left[e^{-(\underline{x}-\underline{\mu}_1)^2} \quad e^{-(x-\mu_2)^2} \quad e^{-(x-\mu_3)^2} \quad \dots \quad e^{-(x-\mu_d)^2} \right]$$

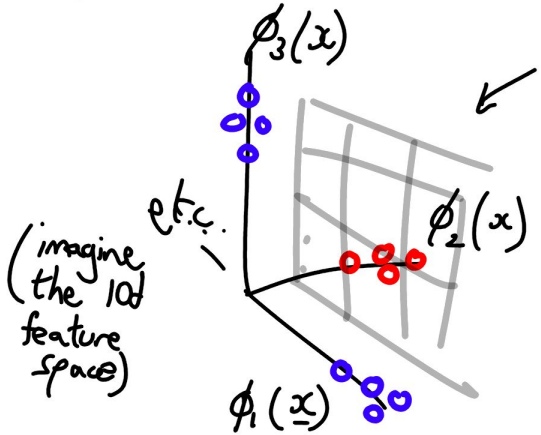
This gives us a d-dimensional space, in which the two classes are separable.

But, remember in the linear regression lecture, we needed to invert

$$\underset{d\times n}{X^T} \underset{n\times d}{X}^{-1}$$

If d is really big it becomes very expensive to compute...





Linearly separable in this 10 dimensional feature space

We can rewrite our expression for y_*...

$$y_* = X_*(X^TX)^{-1}X^Ty$$

$$= X_*X^T(XX^T)^{-1}y$$

First, let's check this is true...

$$X_*(X^TX)^{-1}X^Ty = X_*X^T(XX^T)^{-1}y$$

$$(X^TX)(X^TX)^{-1}X^T = (X^TX)X^T(XX^T)^{-1}$$

$$X^T = (X^TX)X^T(XX^T)^{-1}$$

$$X^T = X^TXX^T(XX^T)^{-1}$$

$$X^T = X^T$$

We can rewrite our expression for y_*...

$$y_* = X_* \left(X^T X\right)^{-1} X^T y$$

$$= X_* X^T \left(X X^T\right)^{-1} y$$

Second, why is this useful?

Each term in XX^T is:

$$\phi(x_i)^T \phi(x_j) = \sum_{k=1}^{d} \phi_k(x_i)\phi_k(x_j)$$

We replace this with a single function:

$$\phi(x_i)^T \phi(x_j) = k(x_i, x_j)$$ ← This is much quicker to calculate

The Kernel Trick and Gaussian Processes

The kernel replaces/computes the inner product between pairs of points.

Because we now don't need to explicitly compute the inner product…

(i.e. the sum: $\sum_{k=1}^{d} \phi_k(x_i)\phi_k(x_j)$)

We can effectively include an infinite number of bases!

I won't go into any more detail, but hopefully this gives a sense of the kernel trick, and how it's useful to both Gaussian processes, Support Vector Machines, and other kernel-methods.

Hyperparameters

We mentioned at the start, that we intuitively chose the ‘smoothness’ of our imagined function based on the data we already could see.

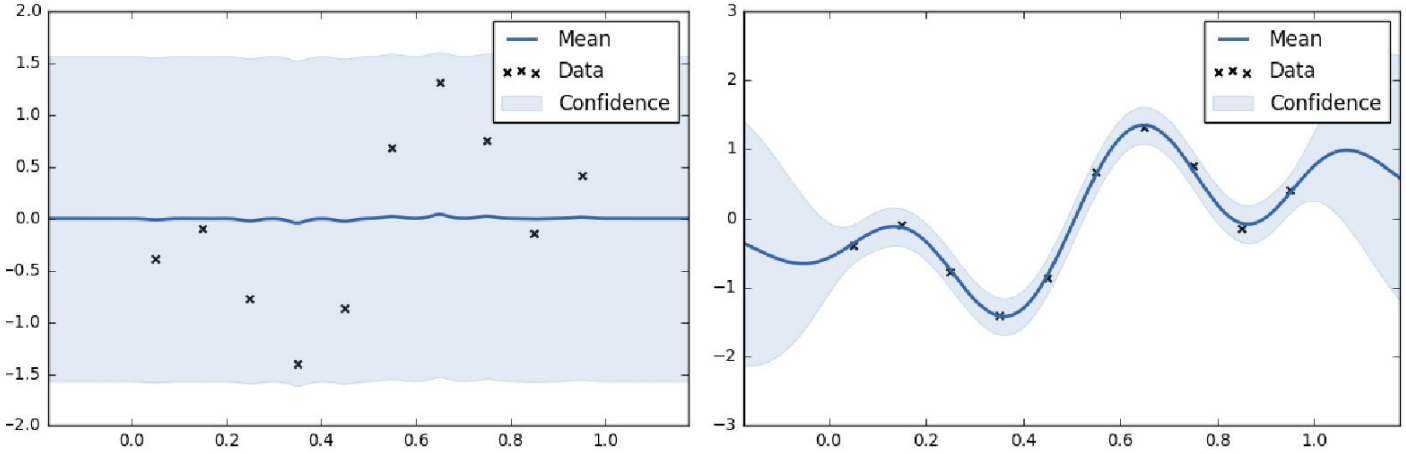
This is equivalent to choosing the lengthscale (for example).

The lengthscale is a **hyperparameter** (there are others, for example, the likelihood “noise” variance).

Usually people pick them by maximising the likelihood (over the training data).

Hyperparameters

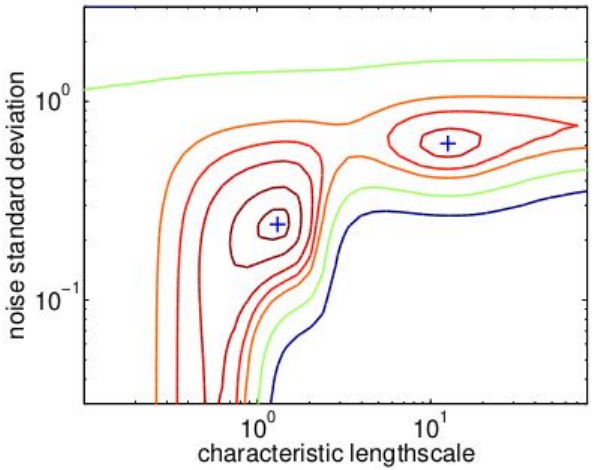
Often two very different choices of hyperparameters can both explain the data quite well (similar likelihood):



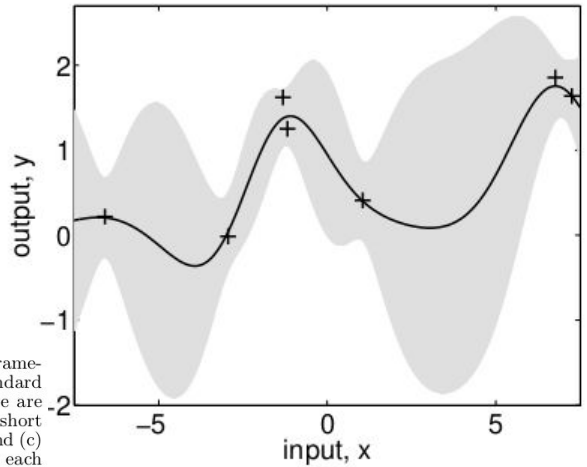
From Richard Wilkinson's GPSS talk
http://gpss.cc/gpss18/slides/Wilkinson2018.pdf

Hyperparameters

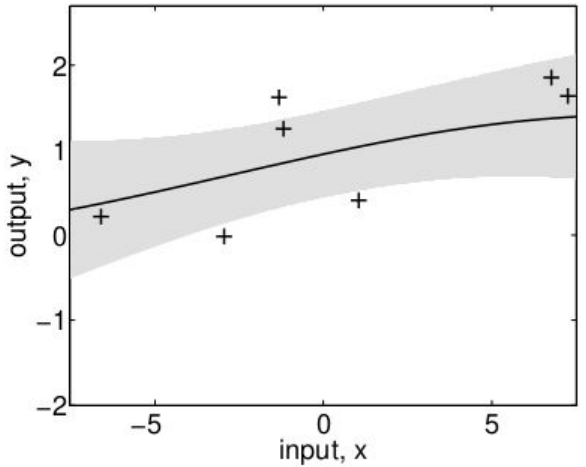
Often two very different choices of hyperparameters can both explain the data quite well (similar likelihood):



(a)



(b)



(c)

From C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

Figure 5.5: Panel (a) shows the marginal likelihood as a function of the hyperparameters ℓ (length-scale) and σ_n^2 (noise standard deviation), where $\sigma_f^2 = 1$ (signal standard deviation) for a data set of 7 observations (seen in panels (b) and (c)). There are two local optima, indicated with '+': the global optimum has low noise and a short length-scale; the local optimum has a high noise and a long length scale. In (b) and (c) the inferred underlying functions (and 95% confidence intervals) are shown for each of the two solutions. In fact, the data points were generated by a Gaussian process with $(\ell, \sigma_f^2, \sigma_n^2) = (1, 1, 0.1)$ in eq. (5.1).

Gaussian processes: For other problems

- Until now we've just looked at regression.
- Can be used with other types of data/problems,
  - E.g. count data, classification, etc…
- But the outputs won't be Gaussian distributed!
  - So not from a Gaussian process!
- We transform them using link functions, and model a latent function as being a Gaussian process, which then (via the link function) produces the observations.

Gaussian processes: For big data

- One challenge is that we solved the problem of lots of features (i.e. big D), but replaced it with the problem of what to do if we have big N?
- X^TX was size DxD.
- XX^T is of size NxN.

$$y_* = X_*\left(X^TX\right)^{-1}X^Ty$$
$$= X_*X^T\left(XX^T\right)^{-1}y$$

Gaussian processes: For big data

- One challenge is that we solved the problem of lots of features (i.e. big D), but replaced it with the problem of what to do if we have big N?
- X^TX was size DxD.
- XX^T is of size NxN.
- Various approaches try to manage this.
- The two most popular are:
  - Using inducing points
  - (combined with) using stochastic variational inference

$$y_* = X_*\left(X^TX\right)^{-1}X^Ty$$
$$= X_*X^T\left(XX^T\right)^{-1}y$$

Gaussian processes: For big data

- **Inducing Points:** replace our full N data points with just a small number (M) of ‘synthetic’ data points.
- If we are clever about where we place them we can approximate our true data well.

Gaussian processes: For big data

- **Variational Inference:** We can improve on just using some inducing points as if they are data, by also finding an alternative Gaussian (or other structure) to explain how the inducing points are related.
- We can iteratively improve this alternative approximation,
- and because it’s iterative, we can use the tools of stochastic gradient descent to allow us to work with arbitrarily large N.

Summary

1. To make predictions we have to make assumptions
2. We define these with the covariance between points.
3. A GP assumes different points Gaussian distributed.
4. Predict by conditioning on some dimensions of our Gaussian distribution.
5. We need to specify a covariance function to determine the covariance between pairs of locations.
6. GPs can be used with non-GP data, and large N.