Gaussian Processes for Regression: Models, Algorithms, and Applications, Day 2

Tamara Broderick

Associate Professor

MIT

http://www.tamarabroderick.com/tutorials.html

Roadmap

Roadmap

- A Bayesian approach

Roadmap

- A Bayesian approach
- What is a Gaussian process?

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel
- Gaussian process inference

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel
- Gaussian process inference
  - Prediction & uncertainty quantification

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel
- Gaussian process inference
  - Prediction & uncertainty quantification
- What are the limits? What can go wrong?

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel
- Gaussian process inference
  - Prediction & uncertainty quantification
- What are the limits? What can go wrong?
- Bayesian optimization

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel
- Gaussian process inference
  - Prediction & uncertainty quantification
- What are the limits? What can go wrong?
- Bayesian optimization

- Goals:

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel
- Gaussian process inference
  - Prediction & uncertainty quantification
- What are the limits? What can go wrong?
- Bayesian optimization

- Goals:
  - Learn the mechanism behind standard GPs to identify benefits and pitfalls

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel
- Gaussian process inference
  - Prediction & uncertainty quantification
- What are the limits? What can go wrong?
- Bayesian optimization

- Goals:
  - Learn the mechanism behind standard GPs to identify benefits and pitfalls
  - Learn the skills to be responsible users of standard GPs (transferable to other ML/AI methods)

Gaussian processes

Gaussian processes

- Definition: “A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution.” [Rasmussen and Williams 2006; a much much older idea!]

Gaussian processes

- Definition: "A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution." [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$

Gaussian processes

- Definition: "A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution." [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

Gaussian processes

- Definition: "A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution." [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

  - Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$

Gaussian processes

- Definition: “A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution.” [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

  - Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
  - Covariance function (a.k.a. *kernel*)

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

Gaussian processes

- Definition: “A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution.” [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

x could be just about anything, but in this tutorial, we’ll assume it’s a real vector

  - Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
  - Covariance function (a.k.a. *kernel*)

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

Gaussian processes

- Definition: "A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution." [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

x could be just about anything, but in this tutorial, we'll assume it's a real vector

  - Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
  - Covariance function (a.k.a. *kernel*)

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

- A common default (e.g. in software) is $m(\mathbf{x}) = 0$

Gaussian processes

- Definition: "A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution." [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

x could be just about anything, but in this tutorial, we'll assume it's a real vector

  - Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
  - Covariance function (a.k.a. *kernel*)

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

- A common default (e.g. in software) is $m(\mathbf{x}) = 0$
- One very commonly used covariance function is the *squared exponential* or *radial basis function (RBF)*

Gaussian processes

- Definition: "A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution." [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

x could be just about anything, but in this tutorial, we'll assume it's a real vector

  - Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
  - Covariance function (a.k.a. *kernel*)

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

- A common default (e.g. in software) is $m(\mathbf{x}) = 0$
- One very commonly used covariance function is the *squared exponential* or *radial basis function (RBF)*
  - We'll see a more general form later, but for now we're using: $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2)$

Gaussian processes

- Definition: "A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution." [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

x could be just about anything, but in this tutorial, we'll assume it's a real vector

  - Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
  - Covariance function (a.k.a. *kernel*)

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

- A common default (e.g. in software) is $m(\mathbf{x}) = 0$
- One very commonly used covariance function is the *squared exponential* or *radial basis function (RBF)*
  - We'll see a more general form later, but for now we're using: $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2)$
- For now, assume data is observed without noise

Gaussian processes

- Definition: "A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution." [Rasmussen and Williams 2006; a much much older idea!]
  - E.g. the function $f(\mathbf{x})$ is a collection indexed by input $\mathbf{x}$
- It is specified by its mean function and covariance function:

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

x could be just about anything, but in this tutorial, we'll assume it's a real vector

  - Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
  - Covariance function (a.k.a. *kernel*)

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

- A common default (e.g. in software) is $m(\mathbf{x}) = 0$
- One very commonly used covariance function is the *squared exponential* or *radial basis function (RBF)*
  - We'll see a more general form later, but for now we're using: $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2)$
- For now, assume data is observed without noise [demo1,2]

Note on “dimension” of the input

Note on “dimension” of the input

- Let’s be careful to separate two types of “dimension”

Note on “dimension” of the input

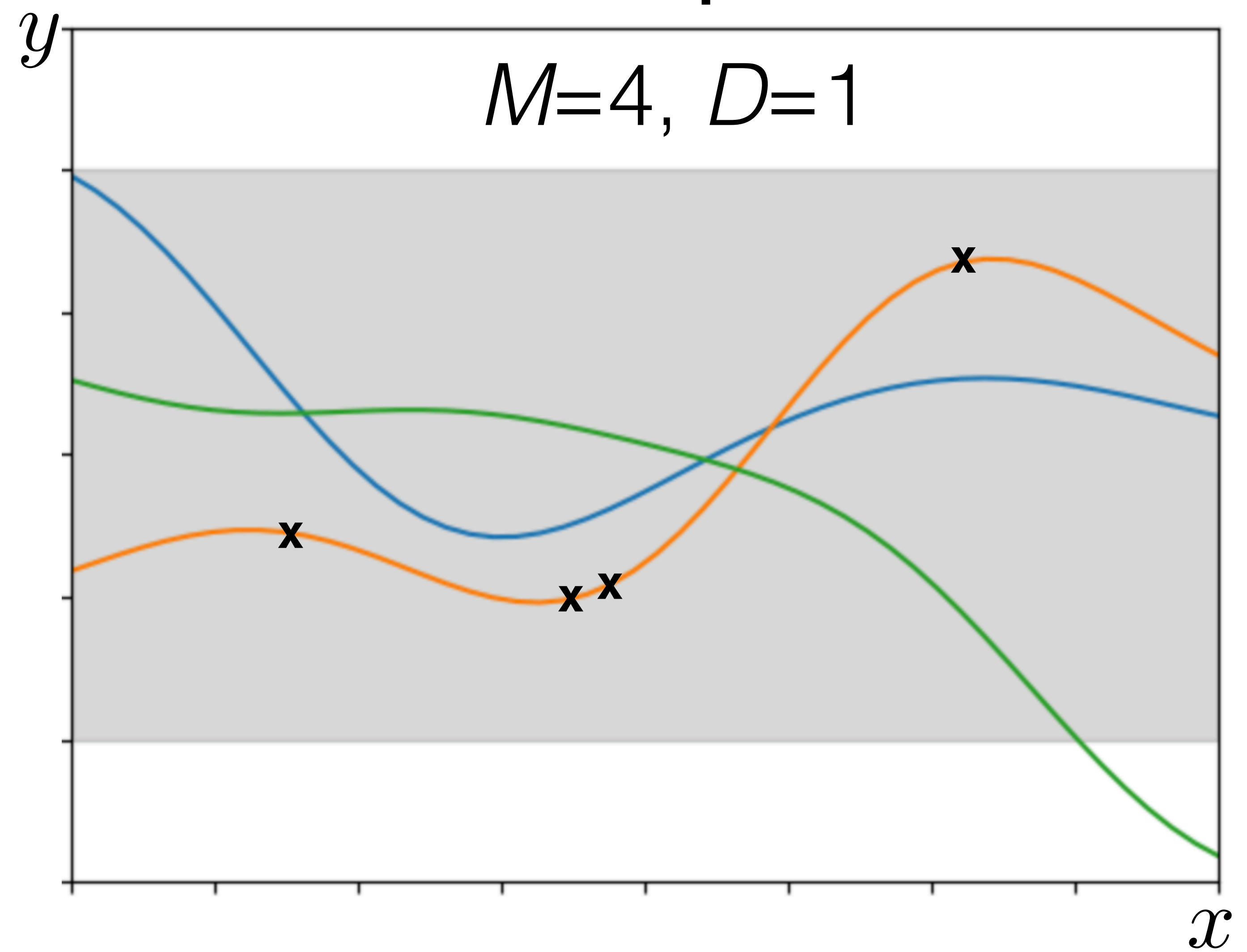
- Let’s be careful to separate two types of “dimension”
  - We’re using a superscript to denote (*M* or *N*) number of points in the space

Note on "dimension" of the input

- Let's be careful to separate two types of "dimension"
  - We're using a superscript to denote (M or N) number of points in the space
  - We'll use a subscript for the (D) different elements of a point's vector

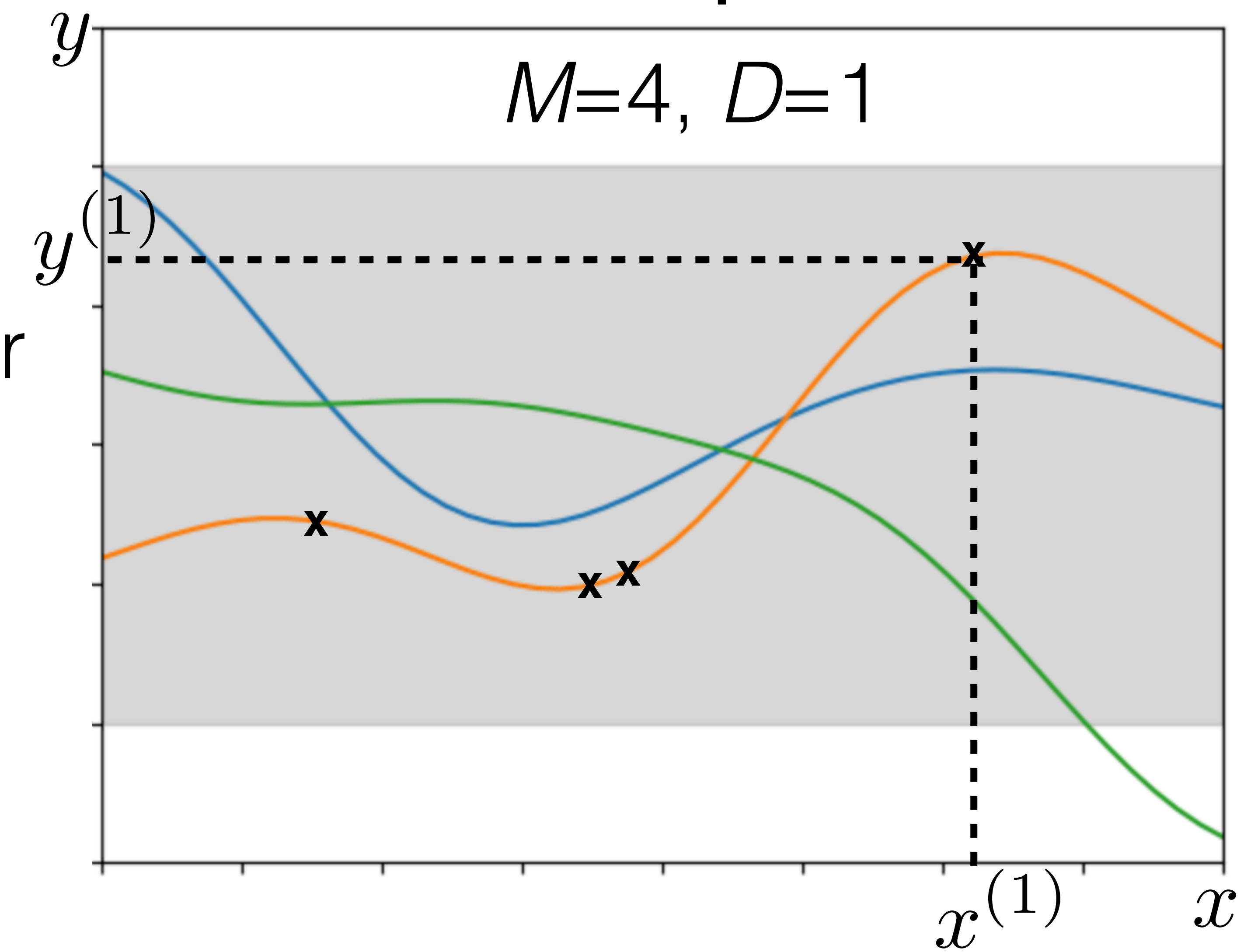
Note on "dimension" of the input

- Let's be careful to separate two types of "dimension"
  - We're using a superscript to denote (M or N) number of points in the space
  - We'll use a subscript for the (D) different elements of a point's vector



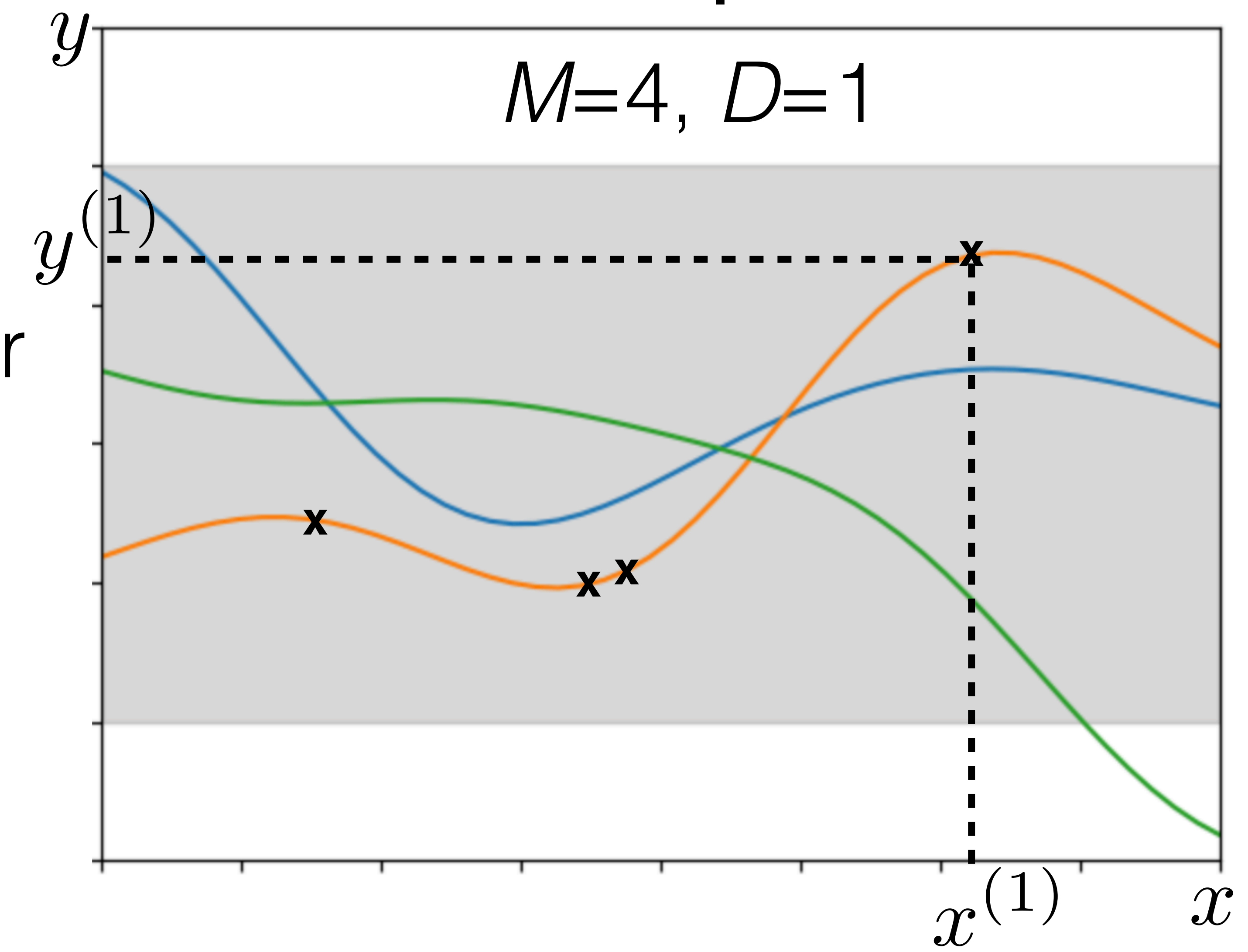
Note on "dimension" of the input

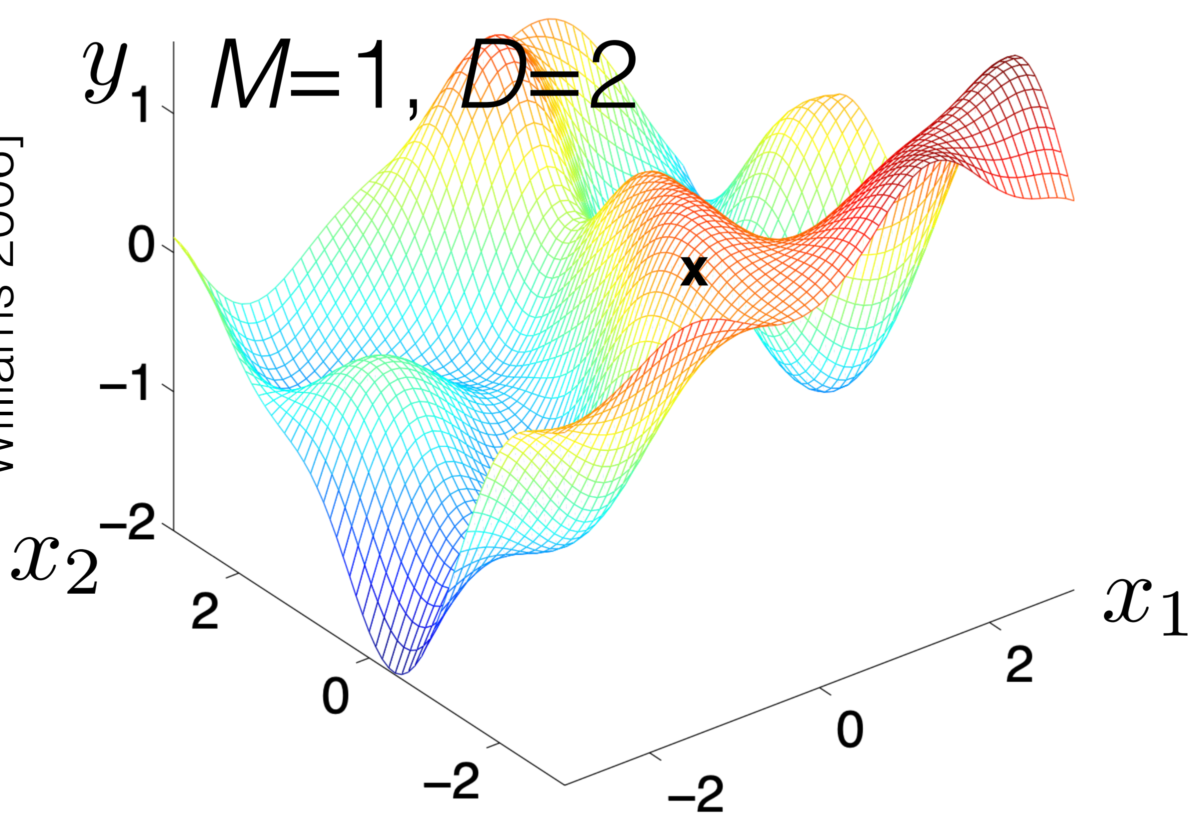
- Let's be careful to separate two types of "dimension"
  - We're using a superscript to denote (*M* or *N*) number of points in the space
  - We'll use a subscript for the (*D*) different elements of a point's vector



Note on "dimension" of the input

- Let's be careful to separate two types of "dimension"
  - We're using a superscript to denote (M or N) number of points in the space
  - We'll use a subscript for the (D) different elements of a point's vector

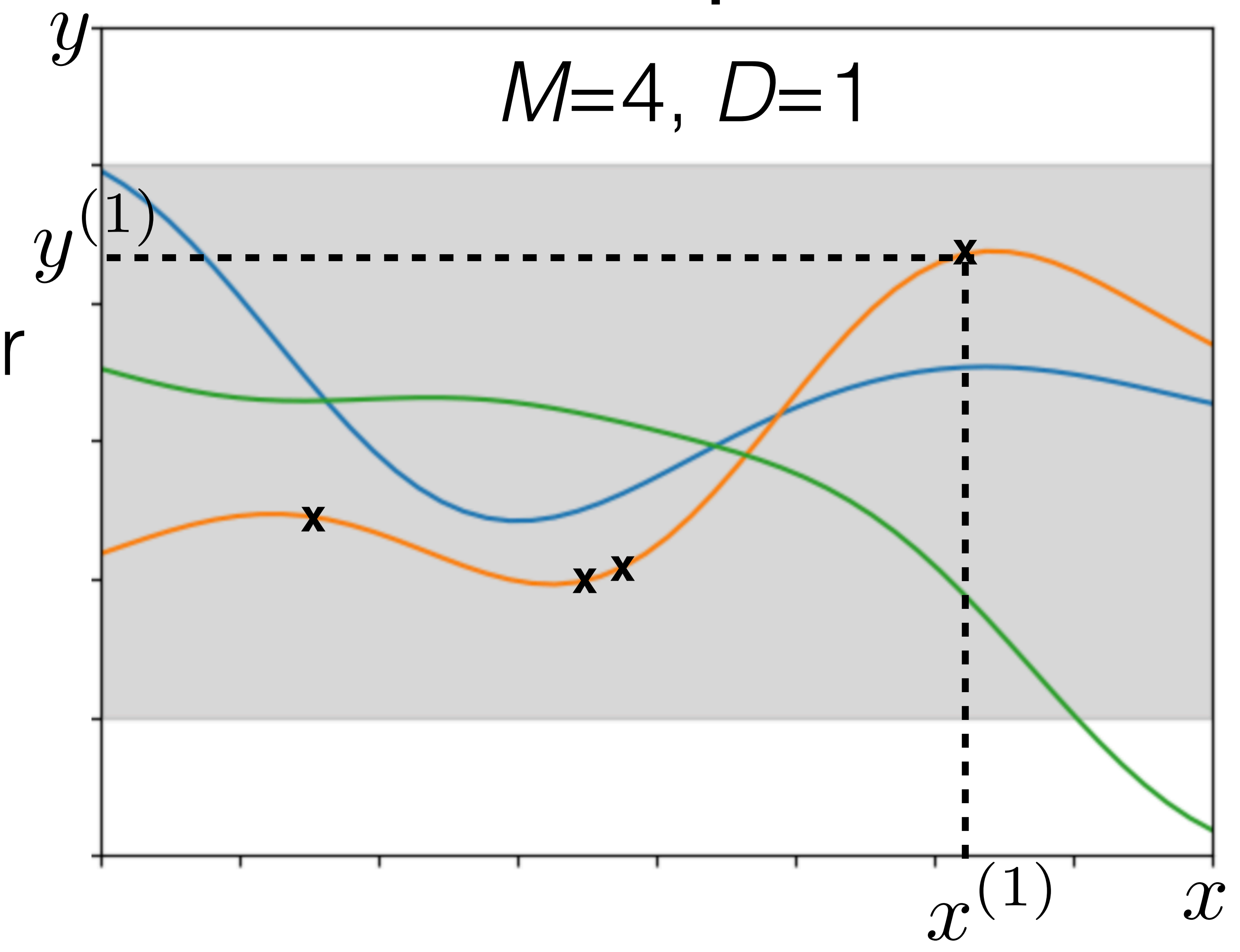


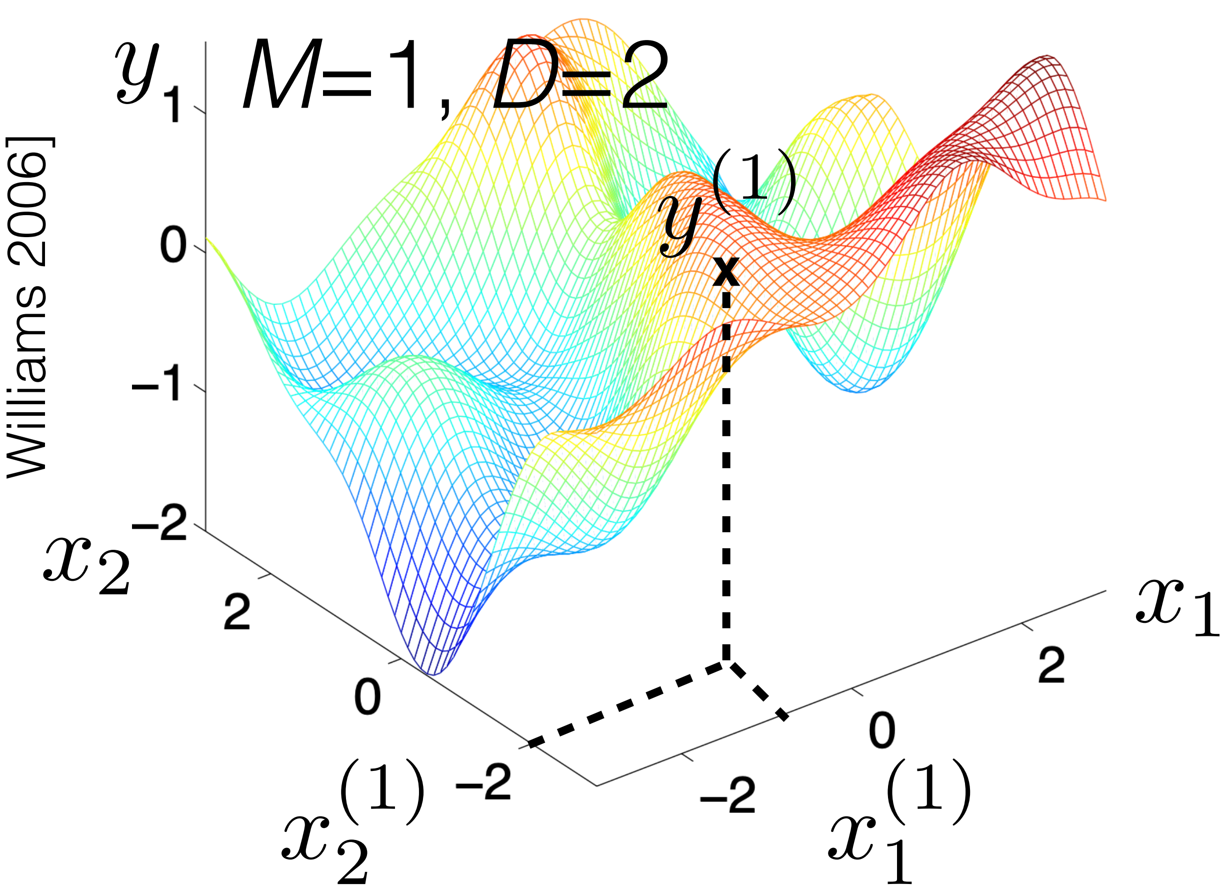


[Rasmussen & Williams 2006]

Note on “dimension” of the input

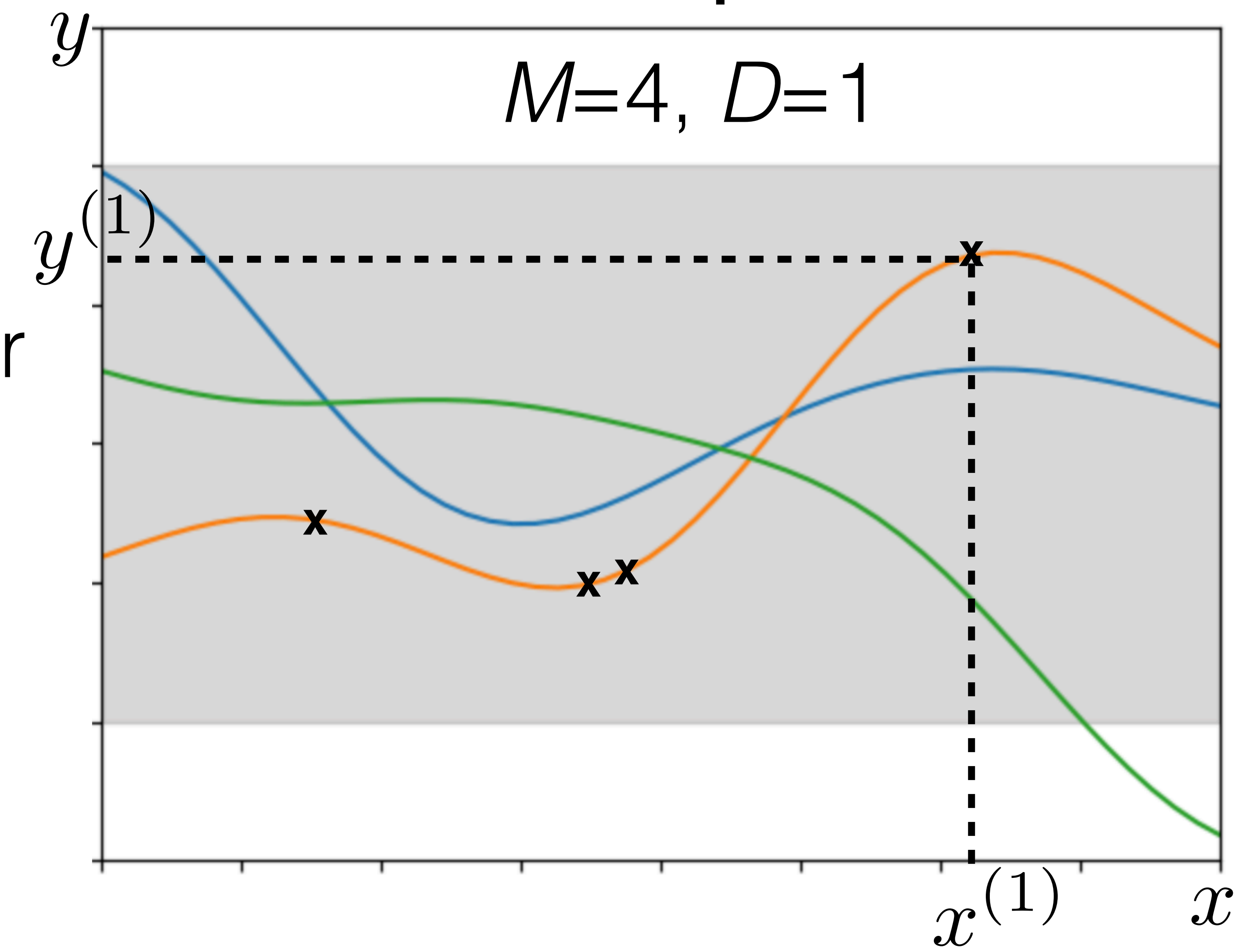
- Let’s be careful to separate two types of “dimension”
  - We’re using a superscript to denote (*M* or *N*) number of points in the space
  - We’ll use a subscript for the (*D*) different elements of a point’s vector

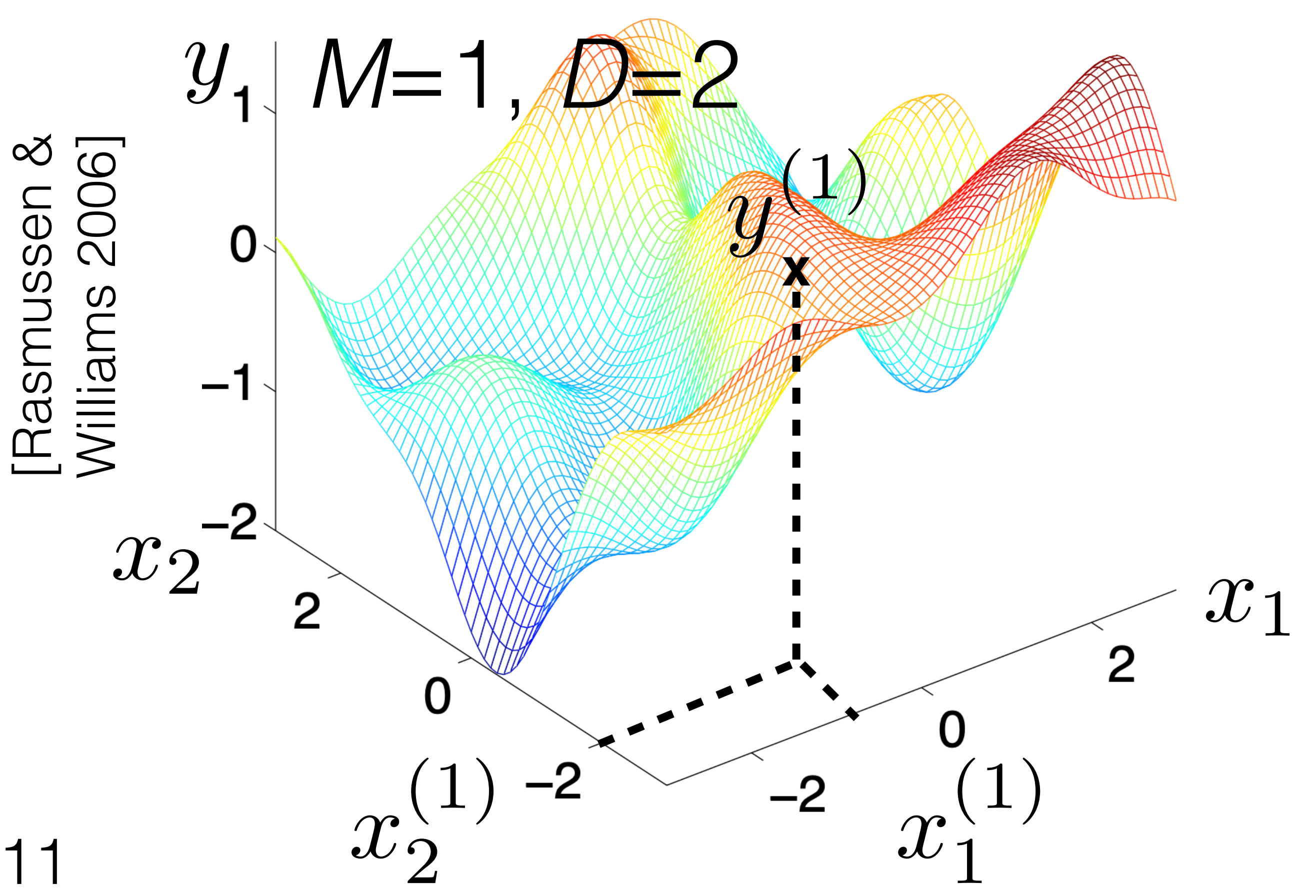




Note on "dimension" of the input

- Let's be careful to separate two types of "dimension"
  - We're using a superscript to denote (*M* or *N*) number of points in the space
  - We'll use a subscript for the (*D*) different elements of a point's vector

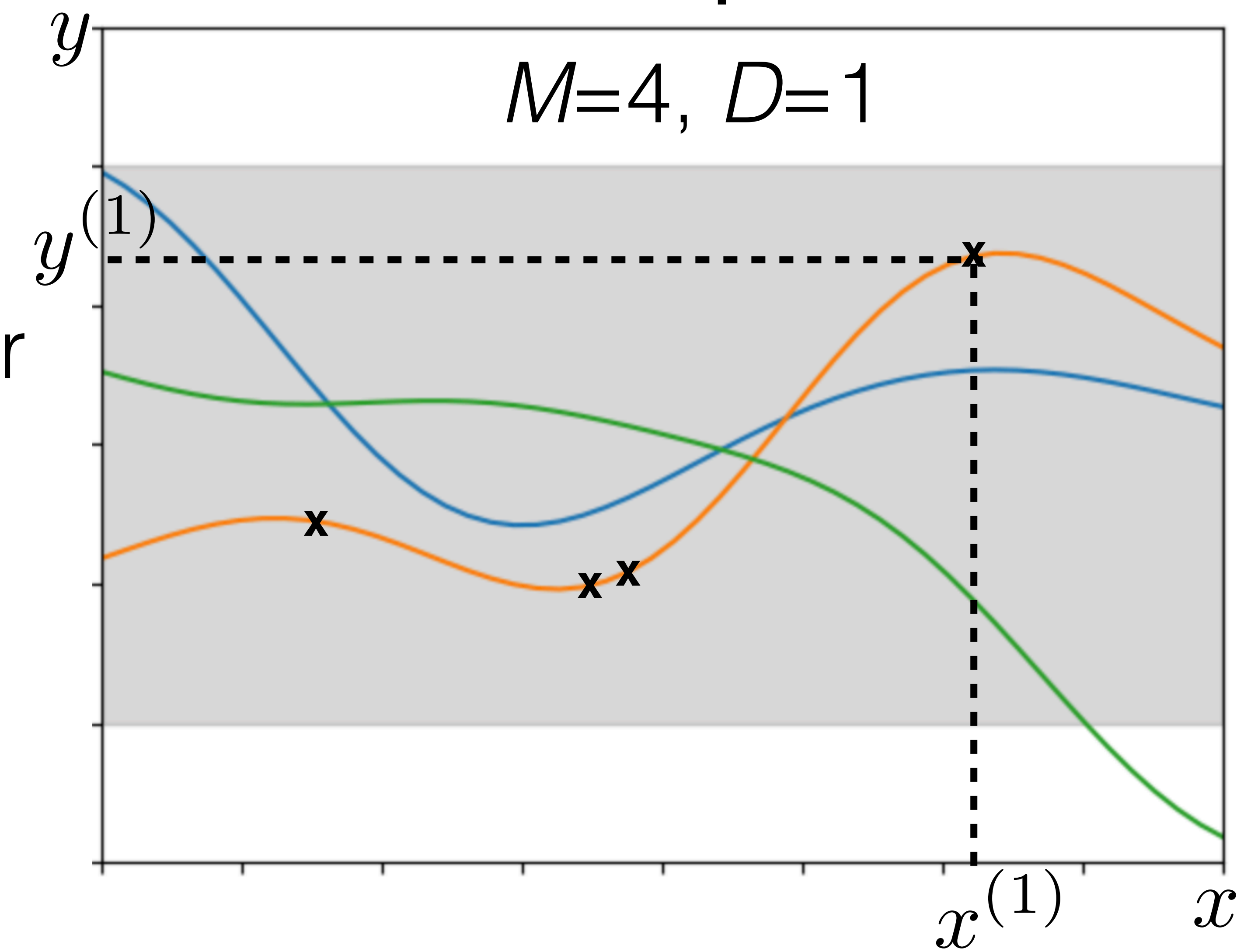


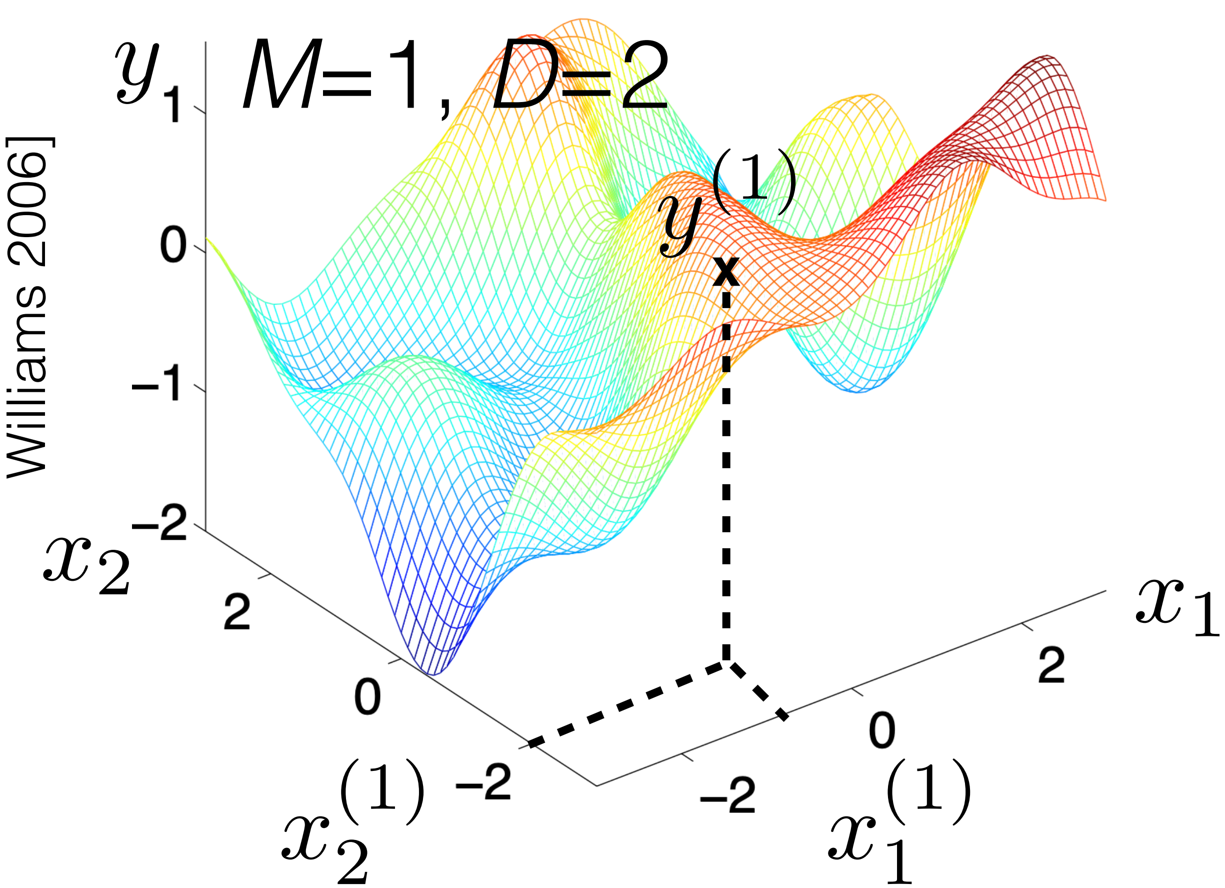


- Note: all of our real-life examples from the start had number of inputs $D > 1$

Note on “dimension” of the input

- Let’s be careful to separate two types of “dimension”
  - We’re using a superscript to denote (*M* or *N*) number of points in the space
  - We’ll use a subscript for the (*D*) different elements of a point’s vector





[Rasmussen & Williams 2006]

- Note: all of our real-life examples from the start had number of inputs $D > 1$
- $D = 1$ is much easier to visualize, but might not be representative

A Bayesian approach

A Bayesian approach

- p(unknowns | data)

A Bayesian approach

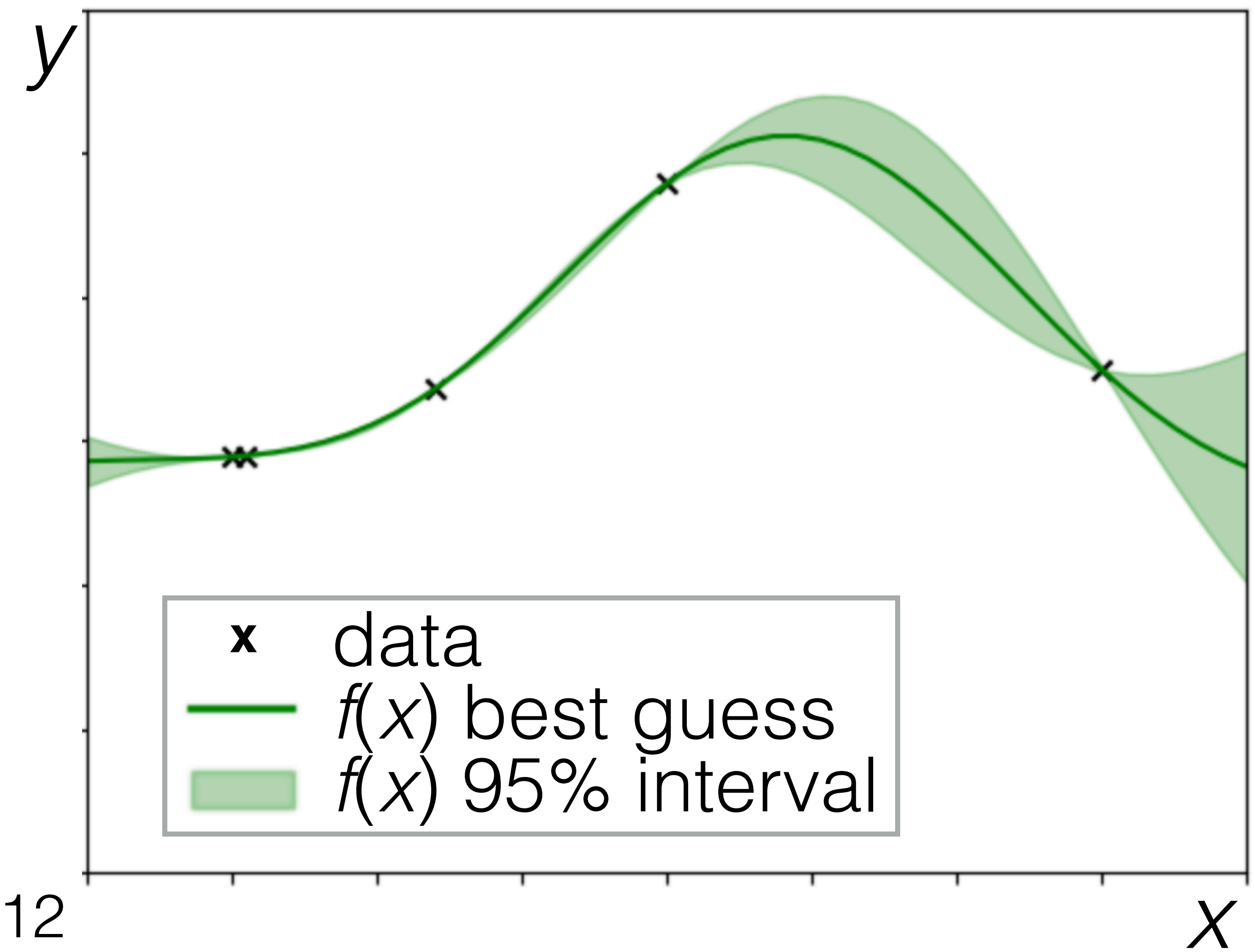
- p(unknowns | data)

Given the data we've seen, what do we know about the underlying function?

A Bayesian approach

- p(unknowns | data)

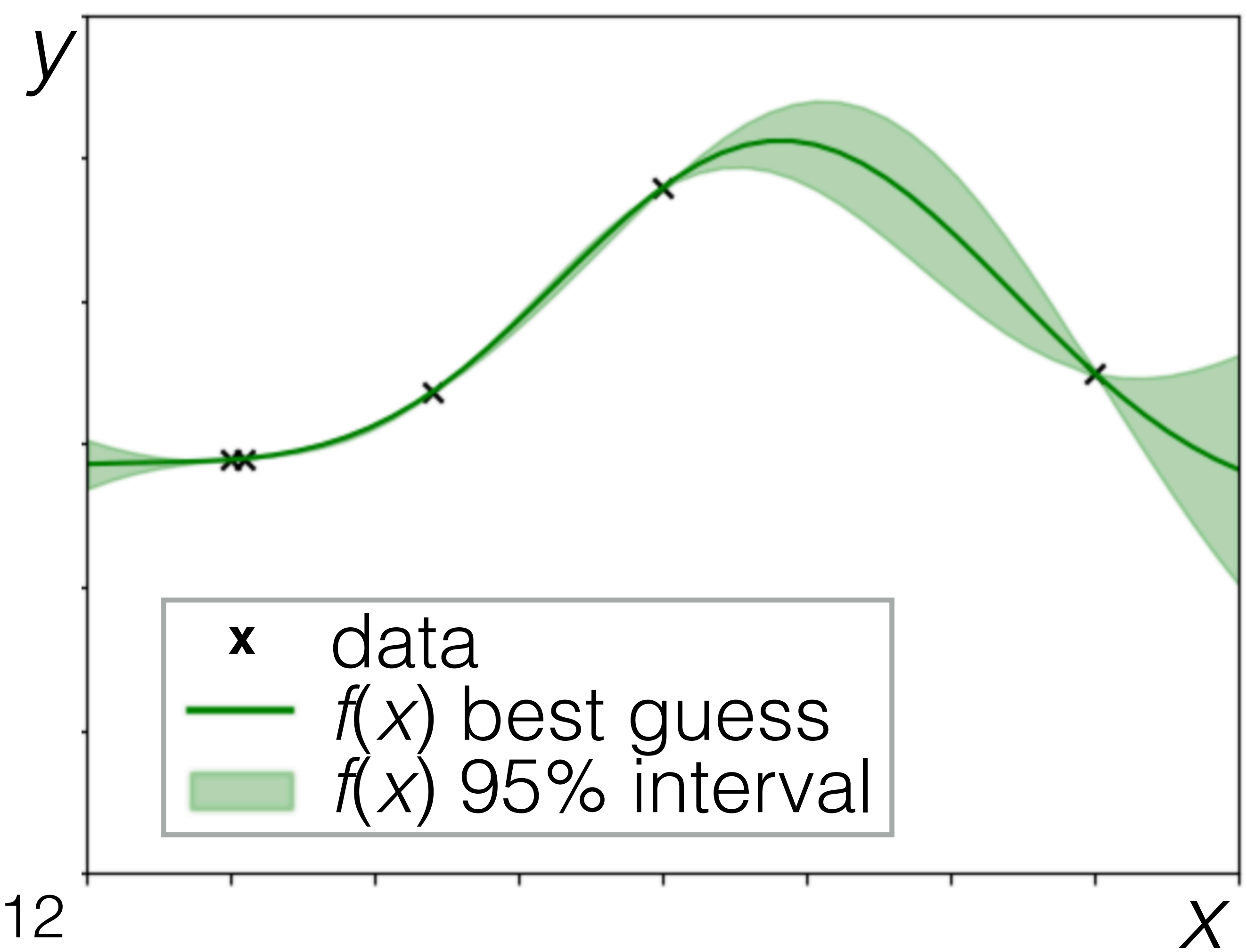
Given the data we've seen, what do we know about the underlying function?



A Bayesian approach

- p(unknowns | data) $\propto$ p(data | unknowns) p(unknowns)

Given the data we've seen, what do we know about the underlying function?

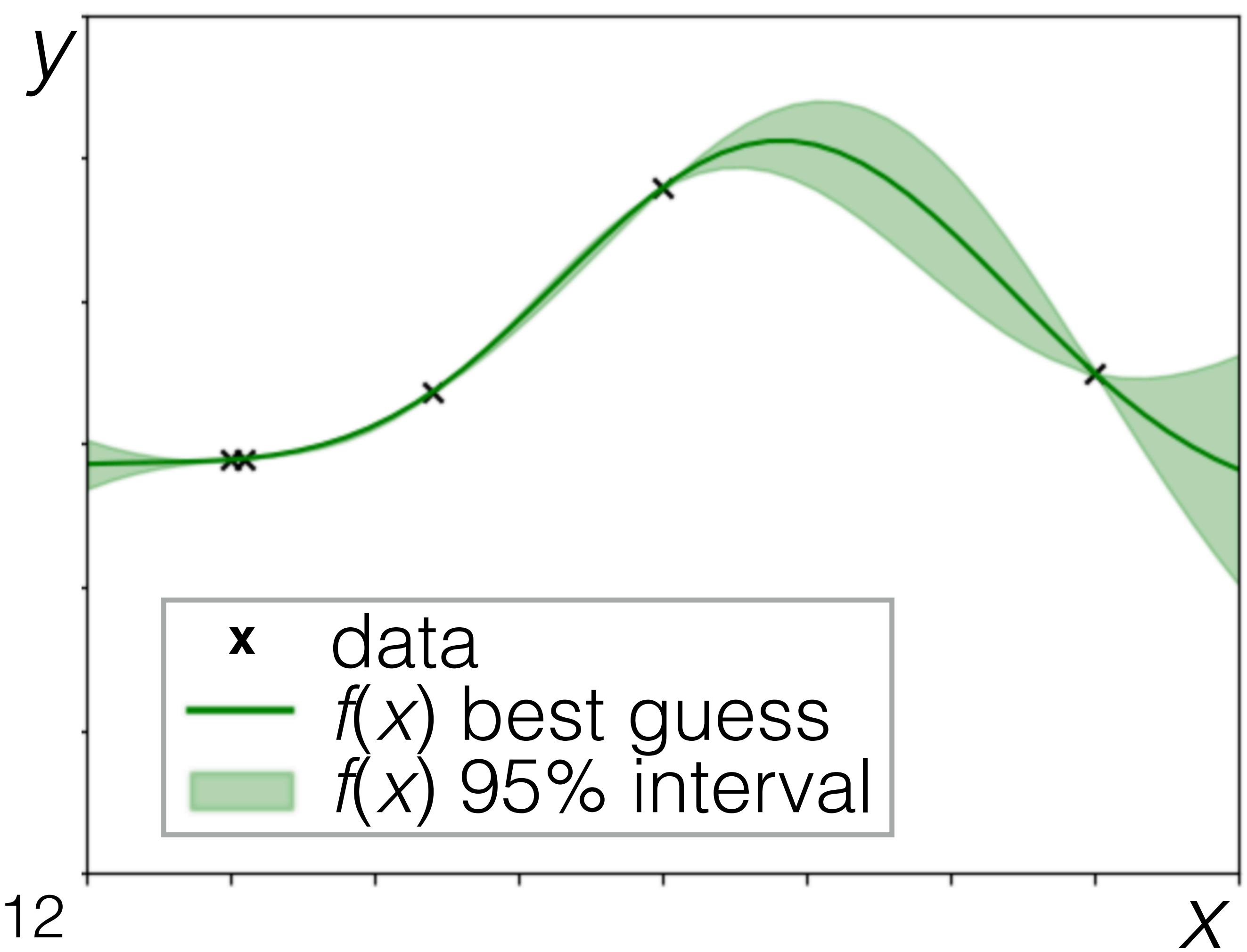


A Bayesian approach

- p(unknowns | data) $\propto$ p(data | unknowns) p(unknowns)

Given the data we've seen, what do we know about the underlying function?

A (statistical) model that can generate functions and data of interest

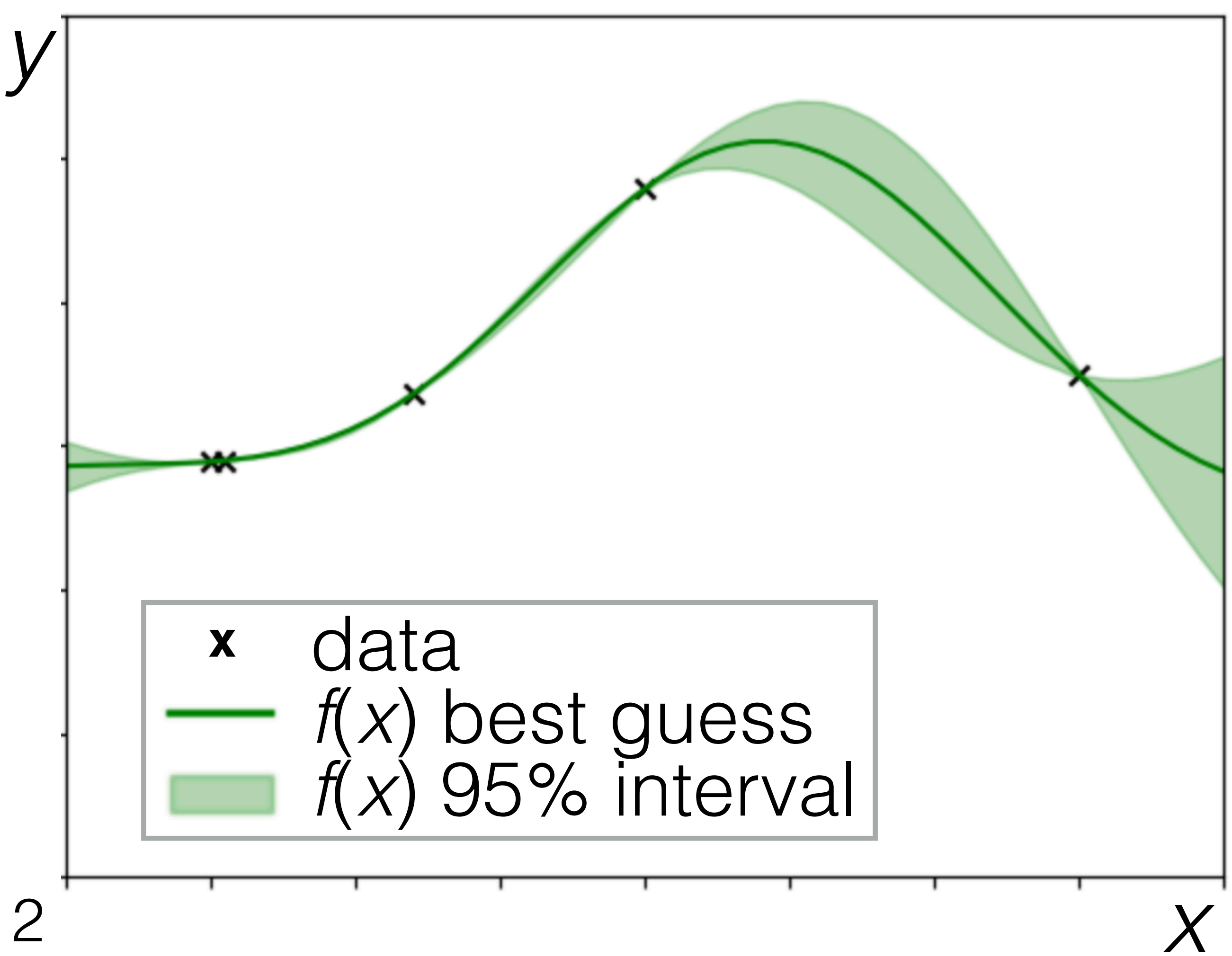


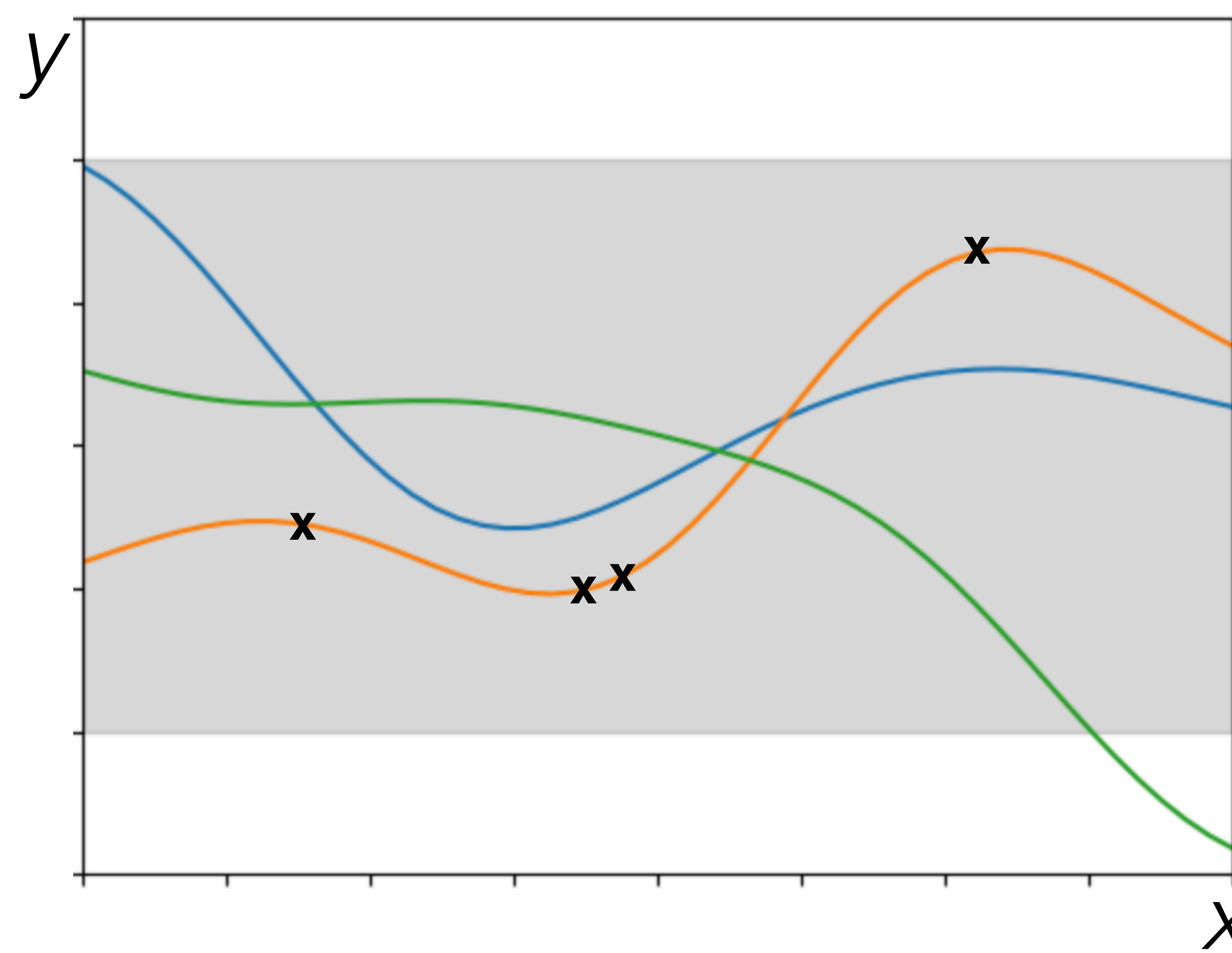
A Bayesian approach

- p(unknowns | data) $\propto$ p(data | unknowns) p(unknowns)

Given the data we've seen, what do we know about the underlying function?

A (statistical) model that can generate functions and data of interest





Inference about unknowns given data

Inference about unknowns given data

- Let X collect the N “training” data points (indexed 1 to N)

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function

Inference about unknowns given data

- Let X collect the N “training” data points (indexed 1 to N)
- Let X' collect the M “test” data points
    - Where we want to evaluate the function
    - Indexed $N+1$ to $N+M$

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*

- Then by our model

$$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \boxed{?}$$

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*

- Then by our model

$$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}$$

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*

- Then by our model

$$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\Bigg(\mathbf{0},$$

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

$$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0},\right.$$

Inference about unknowns given data

- Let *X* collect the *N* "training" data points (indexed 1 to *N*)
- Let *X'* collect the *M* "test" data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X'*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

$$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$$

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X’*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

$$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$$

A good habit to get into: check the dimensions

Inference about unknowns given data

- Let *X* collect the *N* "training" data points (indexed 1 to *N*)
- Let *X'* collect the *M* "test" data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X'*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

$$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$$

X: *N*x*D*
X': *M*x*D*

A good habit to get into: check the dimensions

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X’*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X’: *M*x*D*

*N*x1 $$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$$

A good habit to get into: check the dimensions

Inference about unknowns given data

- Let *X* collect the *N* "training" data points (indexed 1 to *N*)
- Let *X'* collect the *M* "test" data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X'*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*

X': *M*x*D*

$$\begin{array}{c} Nx1 \\ Mx1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$$

A good habit to get into: check the dimensions

Inference about unknowns given data

- Let X collect the N “training” data points (indexed 1 to N)
- Let X' collect the M “test” data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

$$\begin{array}{c c} N\text{x}1 & \multirow{2}{*}{} \\ M\text{x}1 & \end{array}\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$$

X: NxD

X': MxD

A good habit to get into: check the dimensions

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X': *M*x*D*

$$\begin{array}{r} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$$

A good habit to get into: check the dimensions

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X’*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X’: *M*x*D*

$$\begin{array}{r} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}\right)$$

A good habit to get into: check the dimensions

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X’*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X’: *M*x*D*

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

Inference about unknowns given data

- Let X collect the N “training” data points (indexed 1 to N)
- Let X' collect the M “test” data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: NxD

X': MxD

$$\begin{array}{r} N\text{x}1 \\ M\text{x}1 \end{array}\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \boxed{\quad ? \quad}$

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: NxD

X': MxD

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X’*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X’: *M*x*D*

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $K(X',X)K(X,X)^{-1}f(X)$
  - Covariance: $K(X',X') - K(X',X)K(X,X)^{-1}K(X,X')$

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X’*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X’: *M*x*D*

$$\begin{array}{r} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)} \right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $K(X', X)K(X,X)^{-1}f(X)$ Whole mean: ?
  - Covariance: $K(X', X') - K(X', X)K(X,X)^{-1}K(X, X')$

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: NxD
X': MxD

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $K(X',X)K(X,X)^{-1}f(X)$ Whole mean: Mx1
  - Covariance: $K(X',X') - K(X',X)K(X,X)^{-1}K(X,X')$

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X’*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X’: *M*x*D*

$$\begin{array}{c} Nx1 \\ Mx1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)x1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)x(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X') | f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underbrace{K(X',X)}_{MxN} K(X,X)^{-1} f(X)$ Whole mean: *M*x*1*
  - Covariance: $K(X',X') - K(X',X)K(X,X)^{-1}K(X,X')$

Inference about unknowns given data

- Let *X* collect the *N* "training" data points (indexed 1 to *N*)
- Let *X'* collect the *M* "test" data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X'*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*

X': *M*x*D*

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underset{M\text{x}N}{K(X', X)}\,\underset{N\text{x}N}{K(X, X)^{-1}} f(X)$ Whole mean: *M*x*1*
  - Covariance: $K(X', X') - K(X', X)K(X, X)^{-1}K(X, X')$

Inference about unknowns given data

- Let *X* collect the *N* "training" data points (indexed 1 to *N*)
- Let *X'* collect the *M* "test" data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X'*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*

X': *M*x*D*

$$\begin{array}{r} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underset{M\text{x}N}{K(X',X)}\,\underset{N\text{x}N}{K(X,X)^{-1}}\,\underset{N\text{x}1}{f(X)}$ Whole mean: *M*x*1*
  - Covariance: $K(X',X') - K(X',X)K(X,X)^{-1}K(X,X')$

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: NxD
X': MxD

A good habit to get into: check the dimensions

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{f(X)}_{N\text{x}1}$ Whole mean: Mx1
  - Covariance: $K(X',X') - K(X',X)K(X,X)^{-1}K(X,X')$

?

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: NxD
X': MxD

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{f(X)}_{N\text{x}1}$ Whole mean: Mx1
  - Covariance: $\underbrace{K(X',X')}_{M\text{x}M} - K(X',X)K(X,X)^{-1}K(X,X')$

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: NxD
X': MxD

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)} \right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X),X,X' \sim \mathcal{N}$ with
  - Mean: $\underset{M\text{x}N}{K(X',X)}\,\underset{N\text{x}N}{K(X,X)^{-1}}\,\underset{N\text{x}1}{f(X)}$ Whole mean: Mx1
  - Covariance: $\underset{M\text{x}M}{K(X',X')} - \underset{M\text{x}M}{K(X',X)K(X,X)^{-1}K(X,X')}$

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: NxD

X': MxD

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X') | f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underset{M\text{x}N}{K(X',X)}\,\underset{N\text{x}N}{K(X,X)^{-1}}\,\underset{N\text{x}1}{f(X)}$ Whole mean: Mx1
  - Covariance: $\underset{M\text{x}M}{K(X',X')} - \underset{M\text{x}N}{K(X',X)} K(X,X)^{-1} K(X,X')$

MxM

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X': *M*x*D*

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{f(X)}_{N\text{x}1}$ Whole mean: *M*x*1*
  - Covariance: $\underbrace{K(X',X')}_{M\text{x}M} - \underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}K(X,X')$

MxM

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed N+1 to N+M
- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X': *M*x*D*

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{f(X)}_{N\text{x}1}$ Whole mean: *M*x*1*
  - Covariance: $\underbrace{K(X',X')}_{M\text{x}M} - \underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{K(X,X')}_{N\text{x}M}$

*M*x*M*

Inference about unknowns given data

- Let *X* collect the *N* “training” data points (indexed 1 to *N*)
- Let *X’* collect the *M* “test” data points
  - Where we want to evaluate the function
  - Indexed *N*+1 to *N*+*M*
- *K*(*X*,*X’*) is the *N*x*M* matrix with (*n*,*m*) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

X: *N*x*D*
X’: *M*x*D*

$$\begin{array}{c} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{f(X)}_{N\text{x}1}$ Whole mean: *M*x*1*
  - Covariance: $\underbrace{K(X',X')}_{M\text{x}M} - \underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{K(X,X')}_{N\text{x}M}$ (whole: *M*x*M*)
- We’ll infer *f*(*X’*) given our simulated data; recall we’re using
$$k(x,x') = \sigma^2 \exp(-\tfrac{1}{2}(x-x')^2), \sigma = 1$$

Inference about unknowns given data

- Let X collect the N "training" data points (indexed 1 to N)
- Let X' collect the M "test" data points
  - Where we want to evaluate the function
  - Indexed $N+1$ to $N+M$

X: NxD
X': MxD

- $K(X,X')$ is the NxM matrix with (n,m) entry $k(x^{(n)}, x^{(N+m)})$
- Then by our model

$$\begin{array}{r} N\text{x}1 \\ M\text{x}1 \end{array} \begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\mathbf{0}}_{(N+M)\text{x}1}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}}_{(N+M)\text{x}(N+M)}\right)$$

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$ with
  - Mean: $\underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{f(X)}_{N\text{x}1}$ Whole mean: Mx1
  - Covariance: $\underbrace{K(X',X')}_{M\text{x}M} - \underbrace{K(X',X)}_{M\text{x}N}\underbrace{K(X,X)^{-1}}_{N\text{x}N}\underbrace{K(X,X')}_{N\text{x}M}$

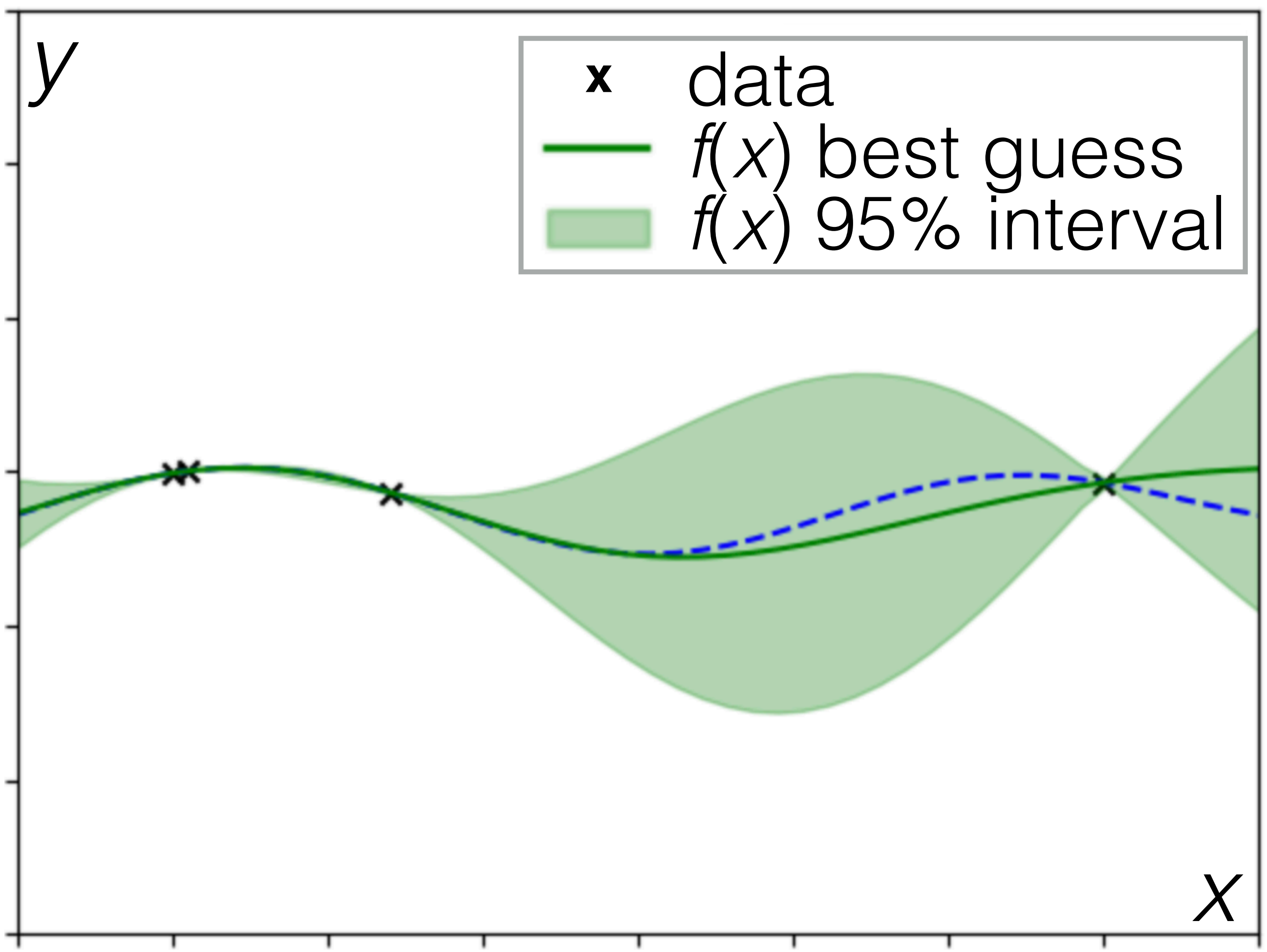
  (overall covariance: MxM)
- We'll infer $f(X')$ given our simulated data; recall we're using

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

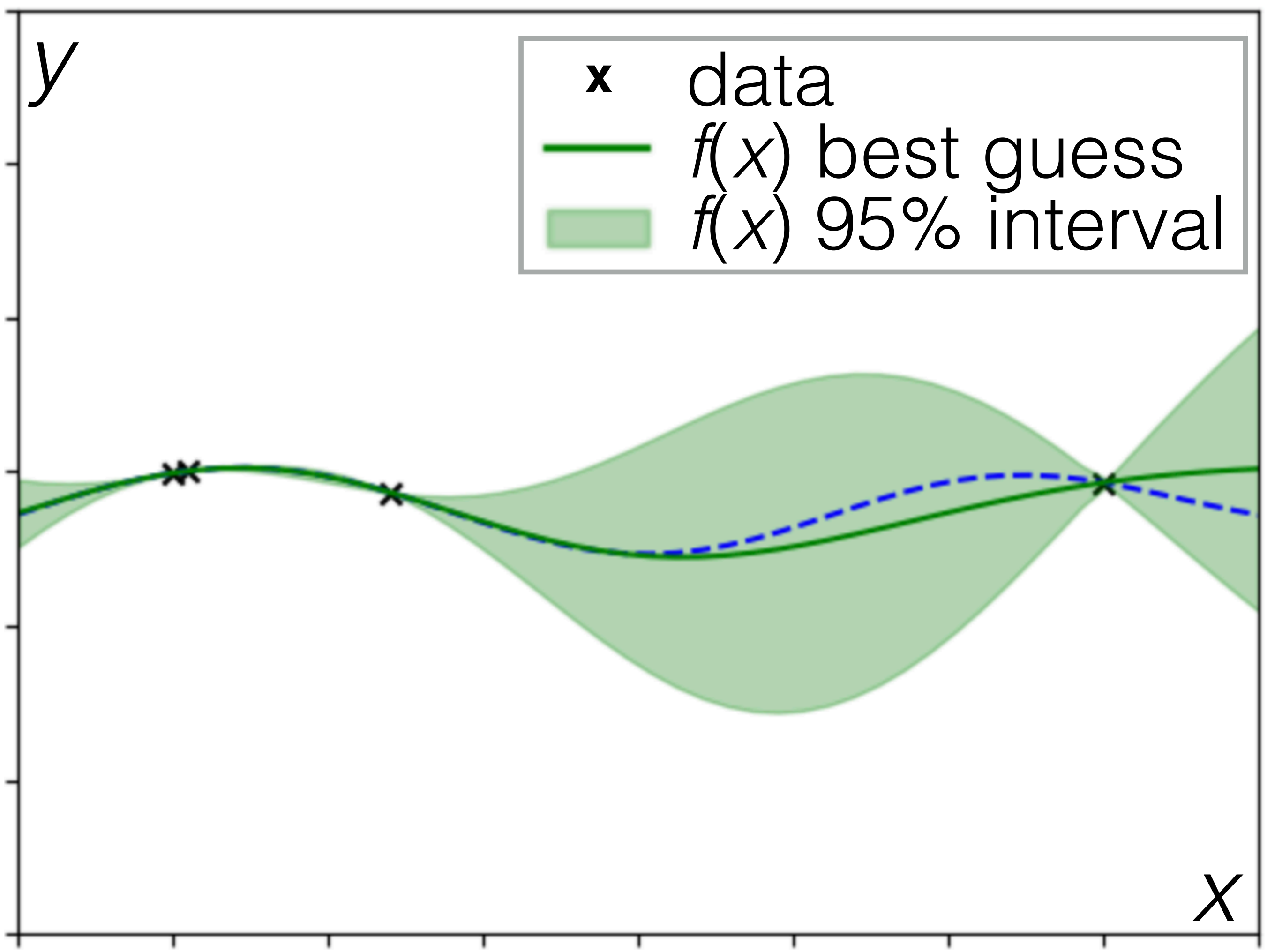
[demo1,2]

Closer look at the uncertainty interval

Closer look at the uncertainty interval

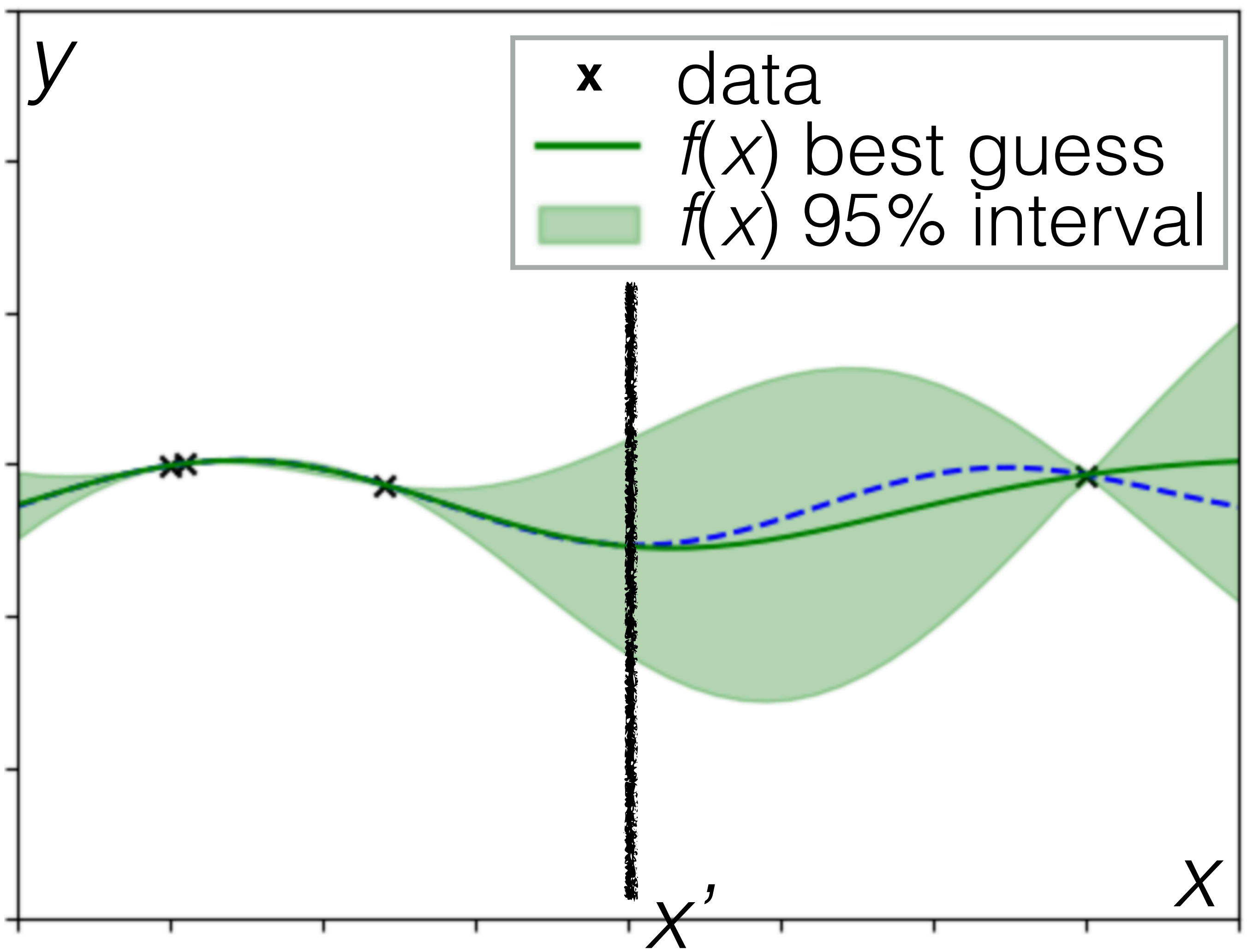


Closer look at the uncertainty interval



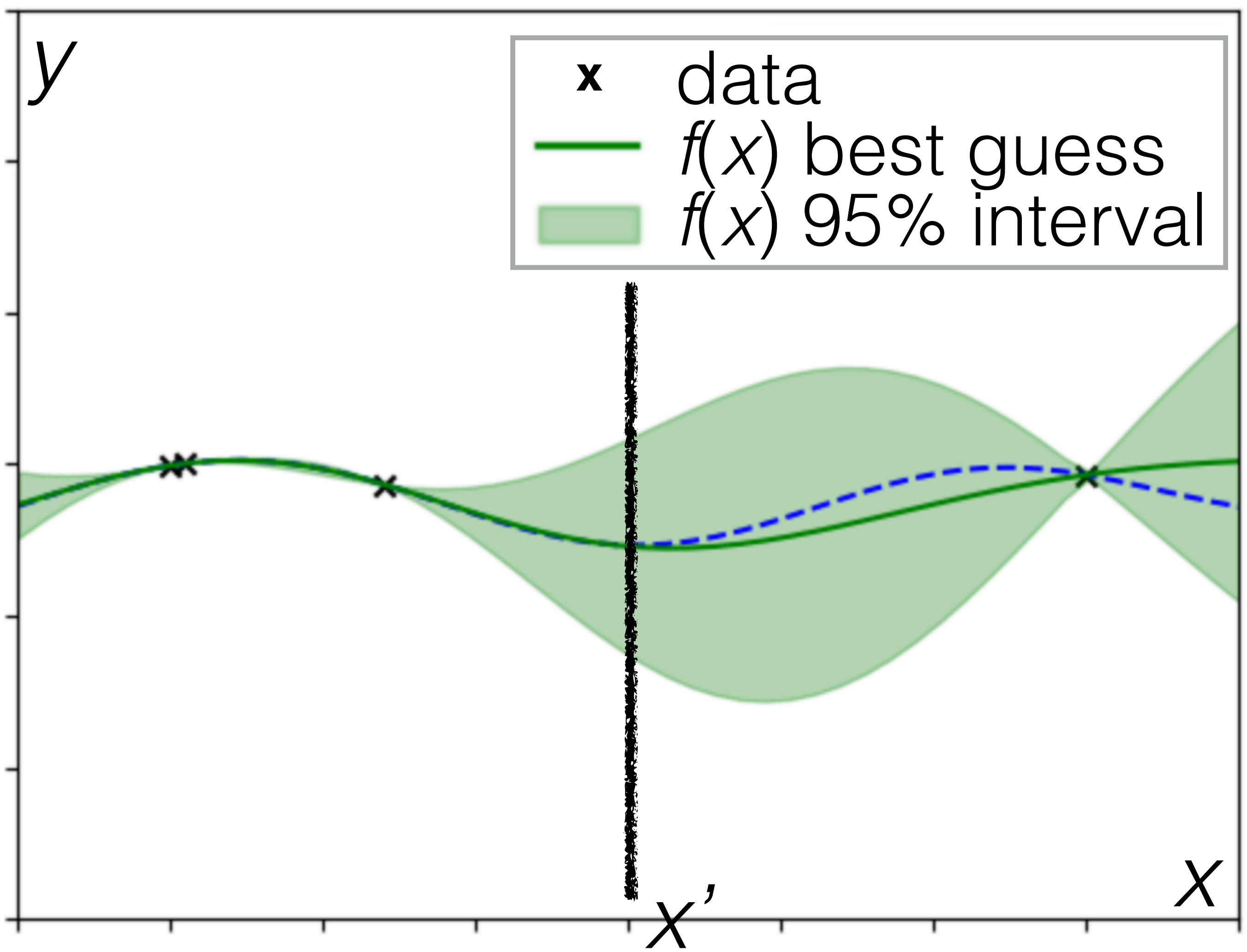
- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian

Closer look at the uncertainty interval



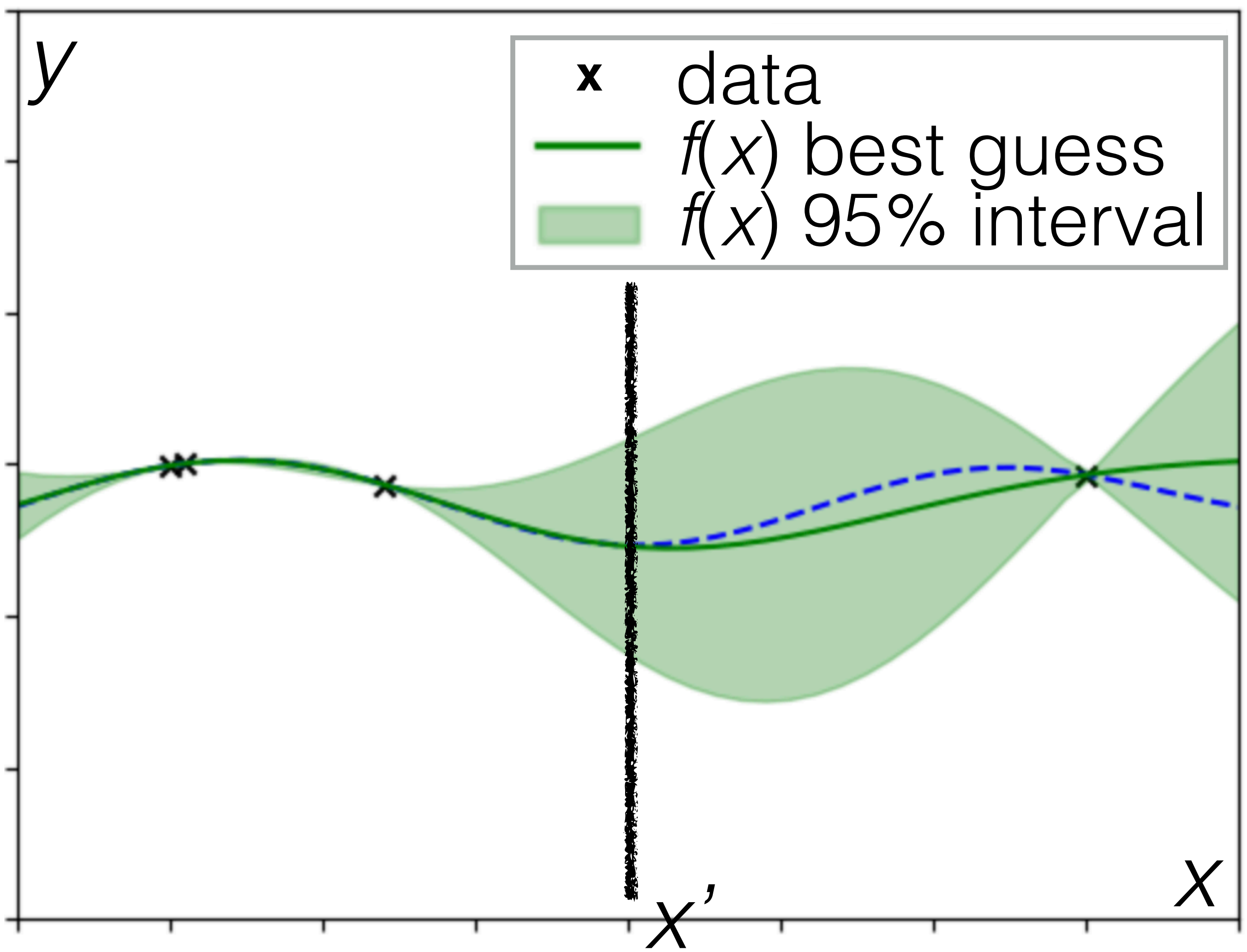
- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian

Closer look at the uncertainty interval



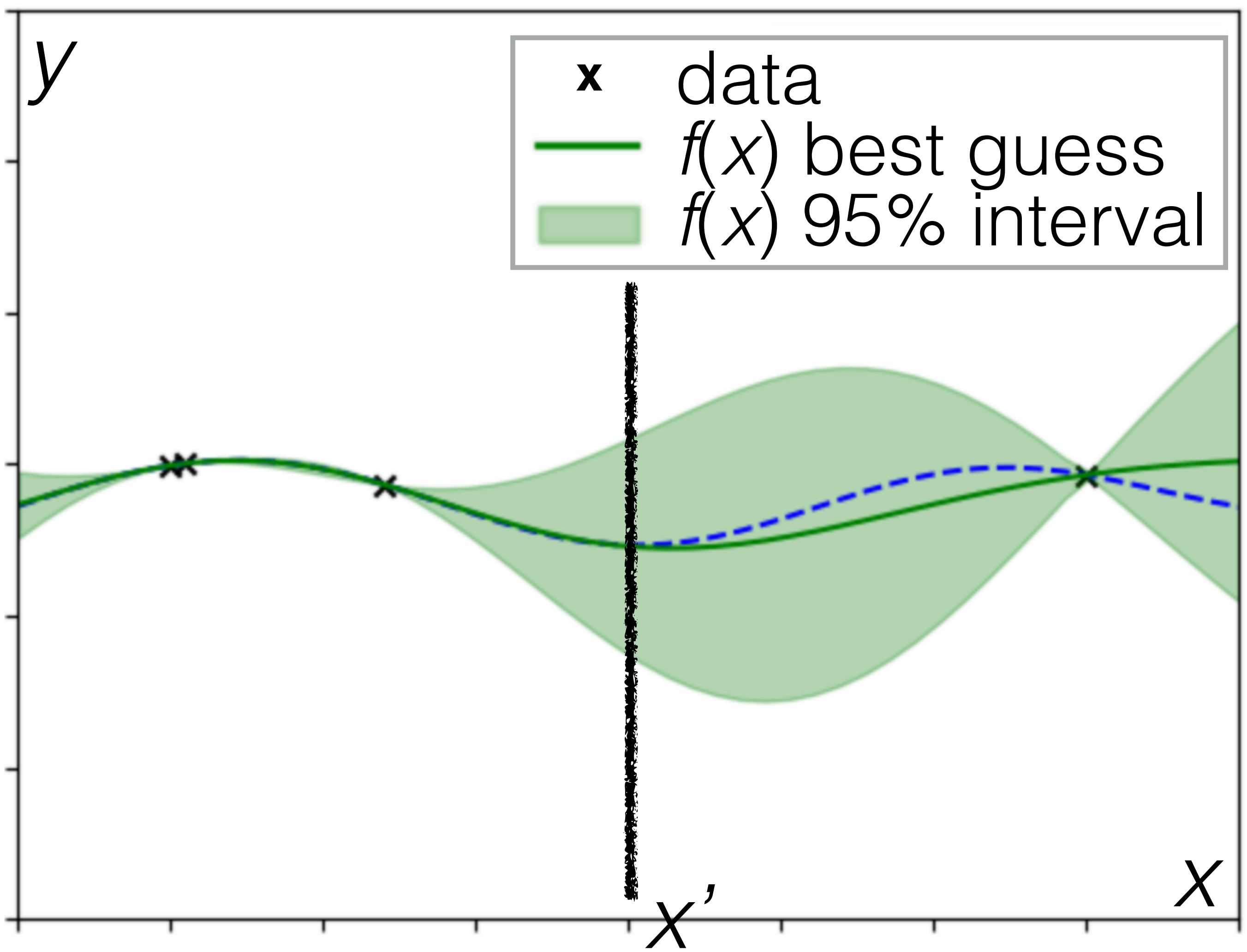
- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian

Closer look at the uncertainty interval



- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

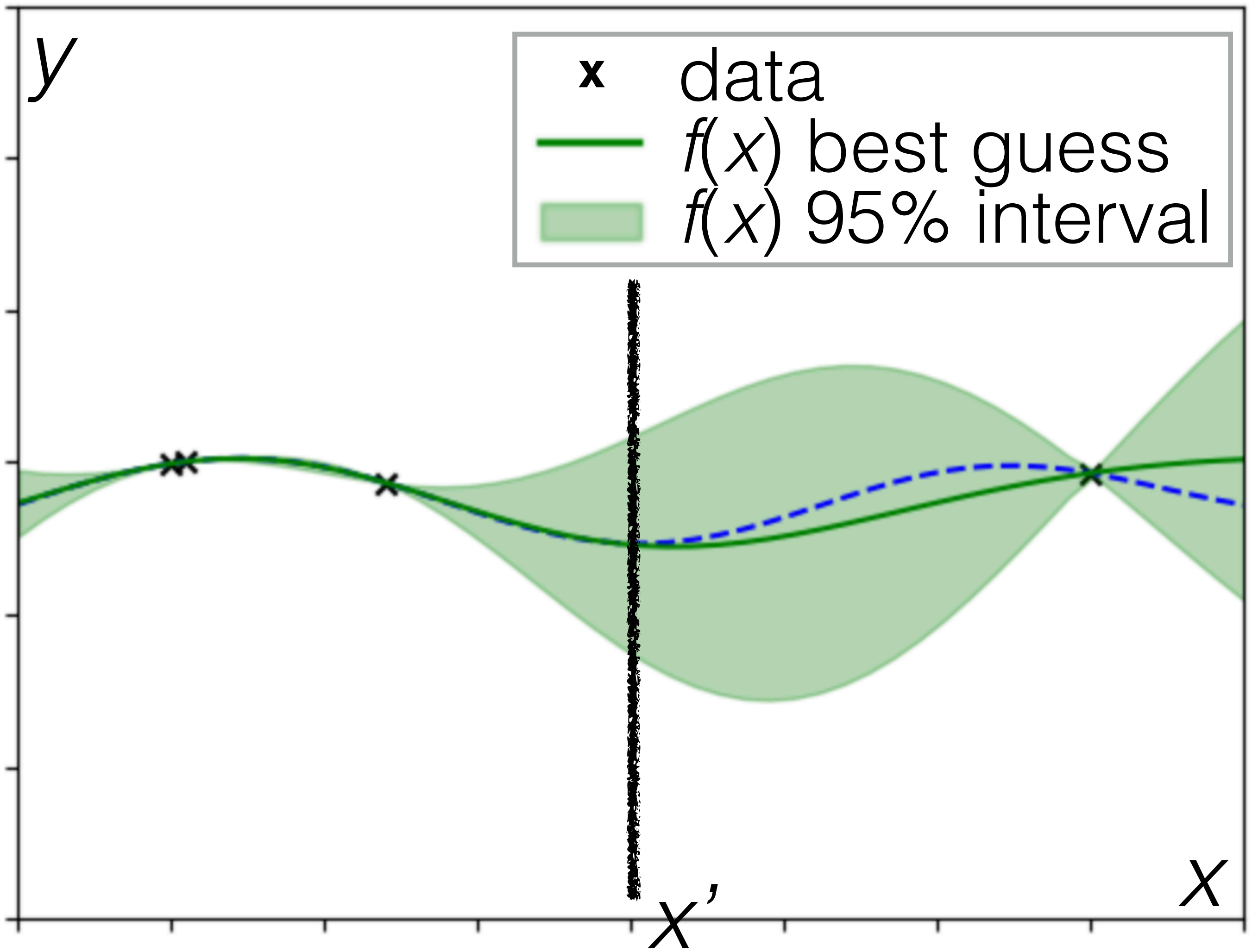
Closer look at the uncertainty interval



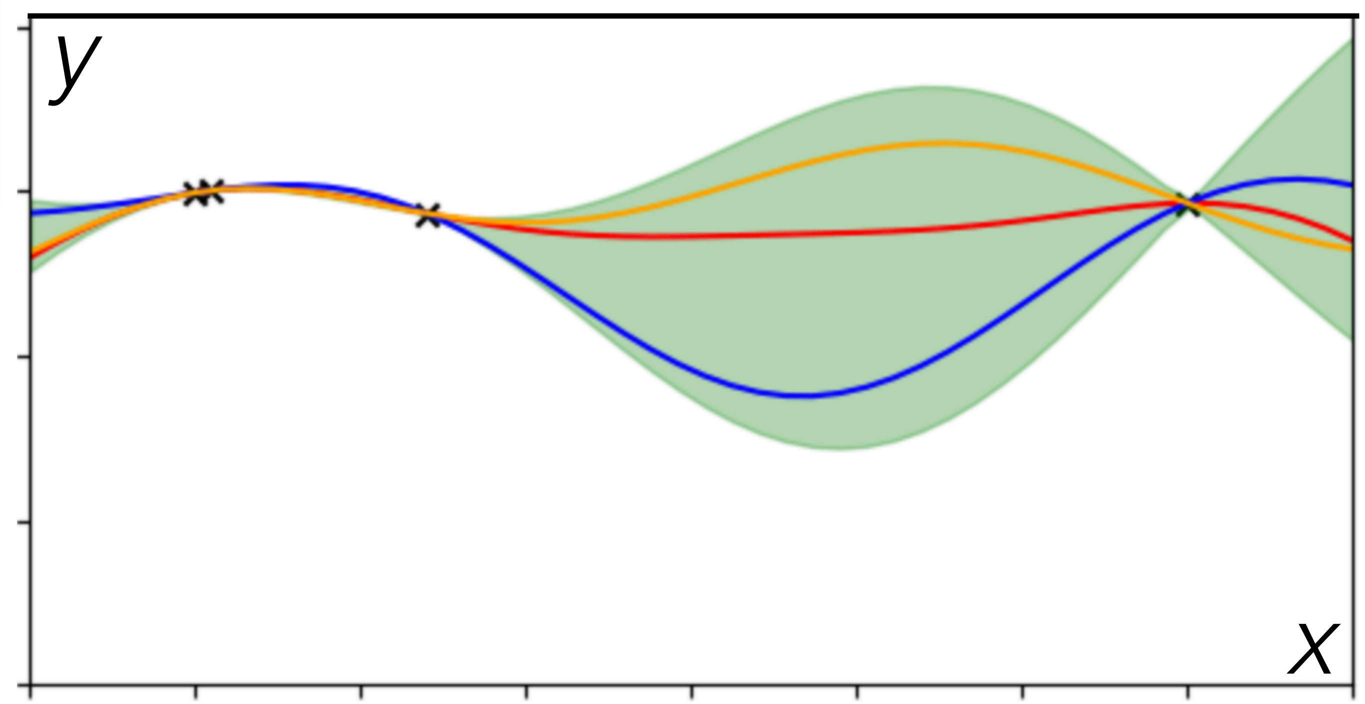
- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

- Draw random *f* conditional on the training data

Closer look at the uncertainty interval

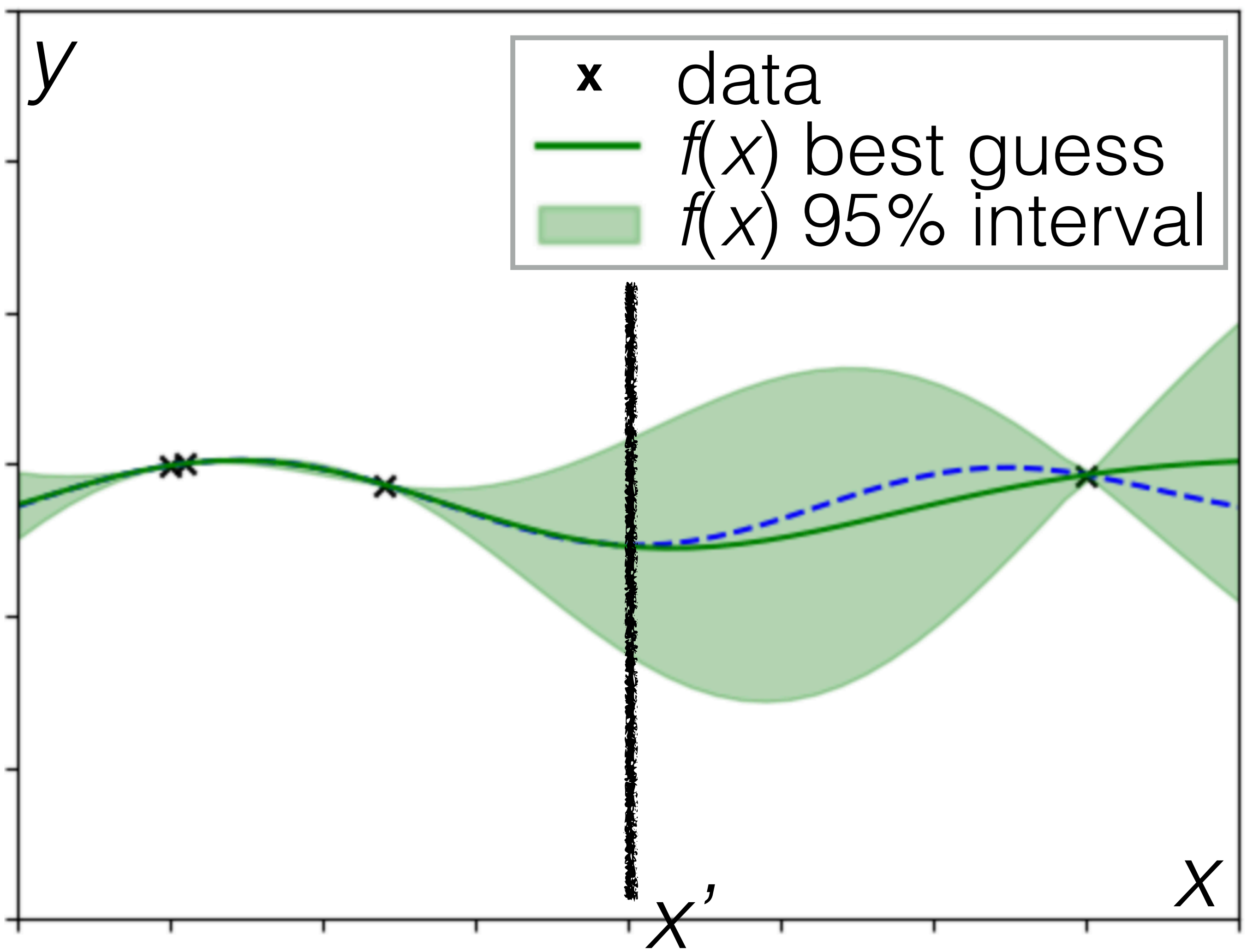


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

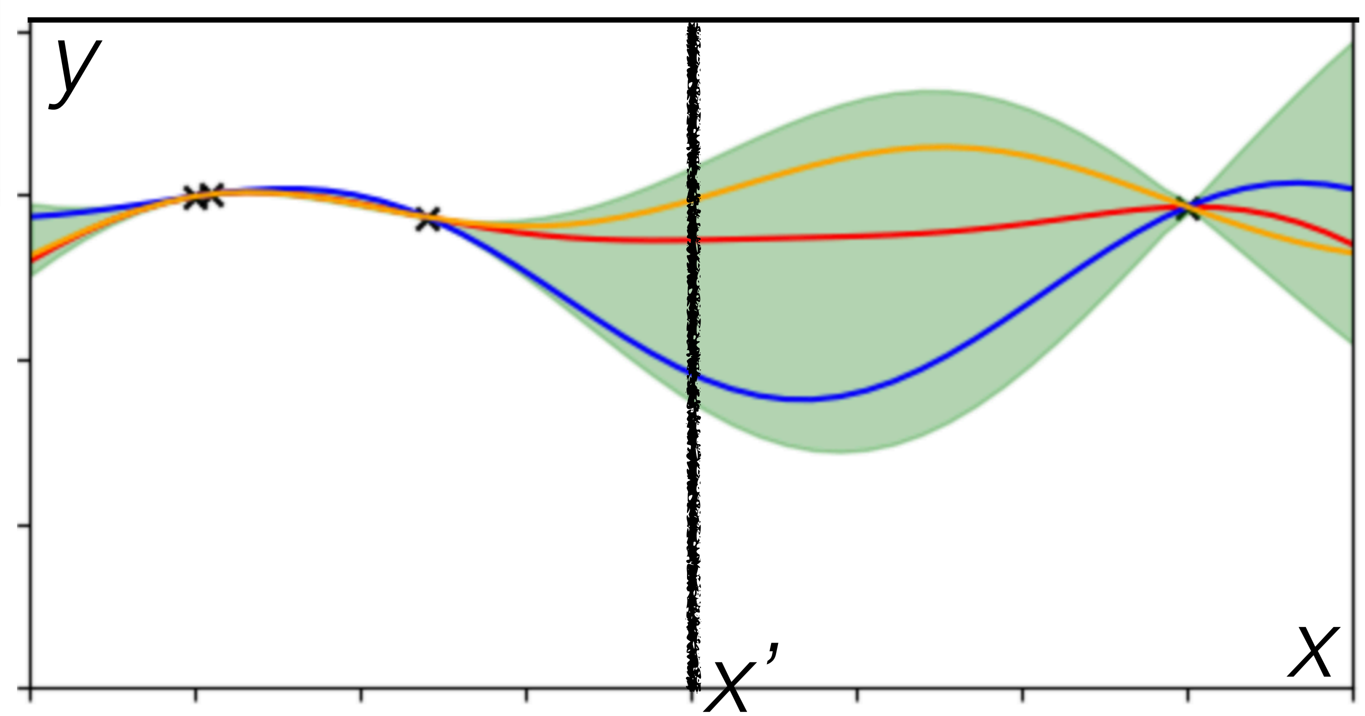


- Draw random *f* conditional on the training data

Closer look at the uncertainty interval

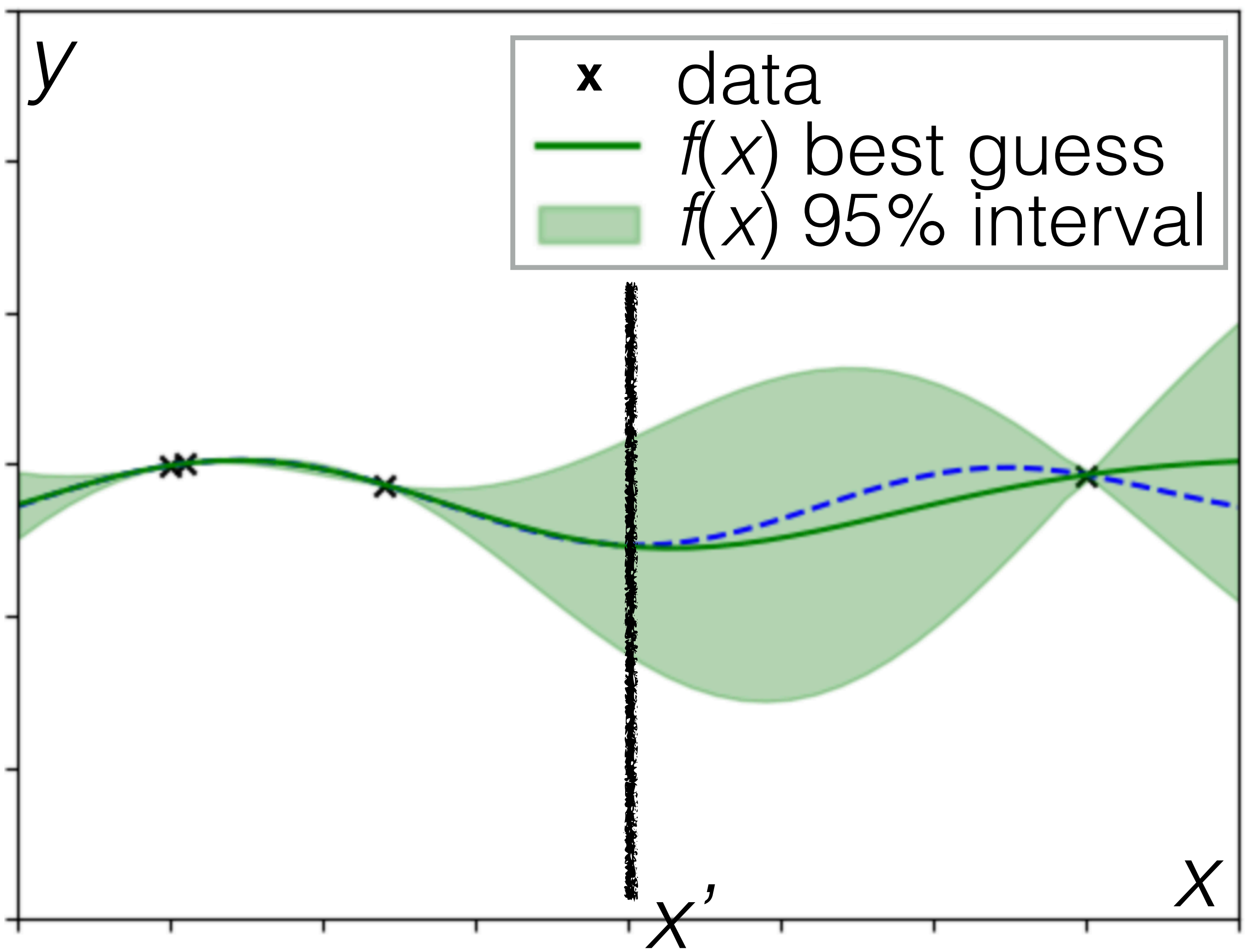


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

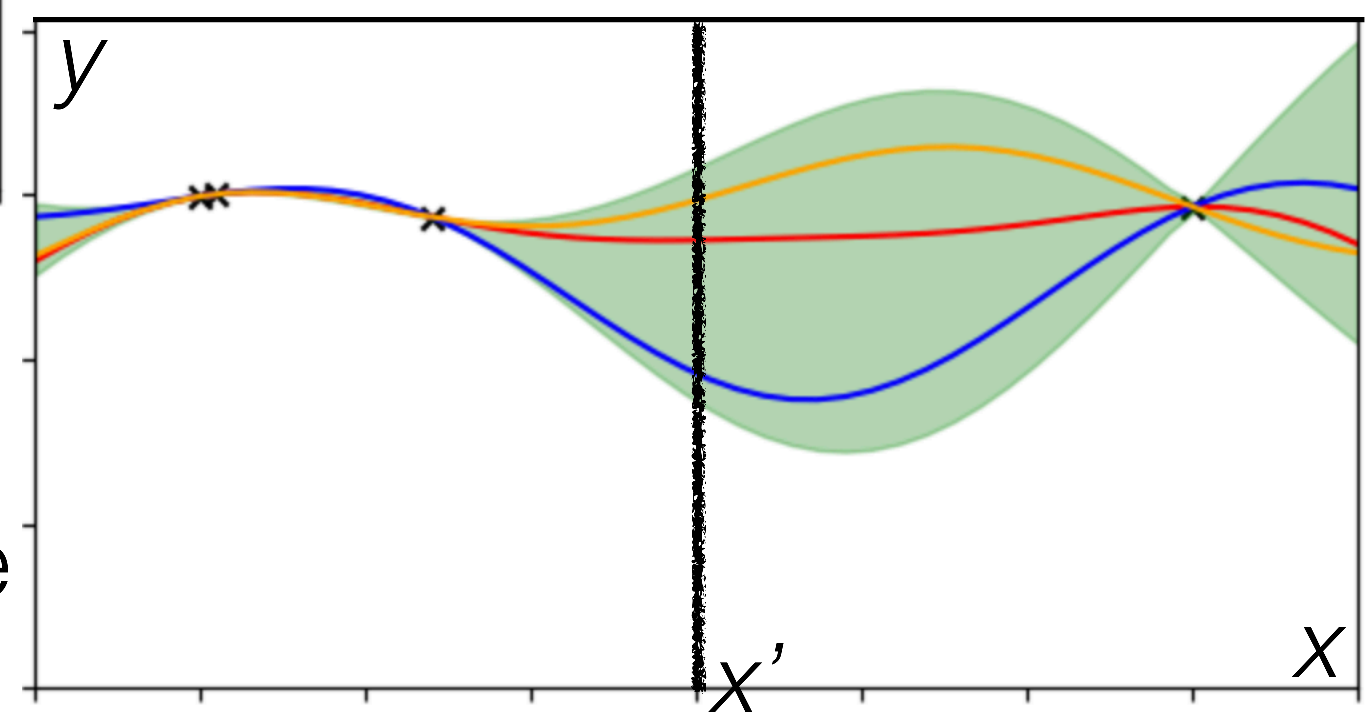


- Draw random *f* conditional on the training data

Closer look at the uncertainty interval

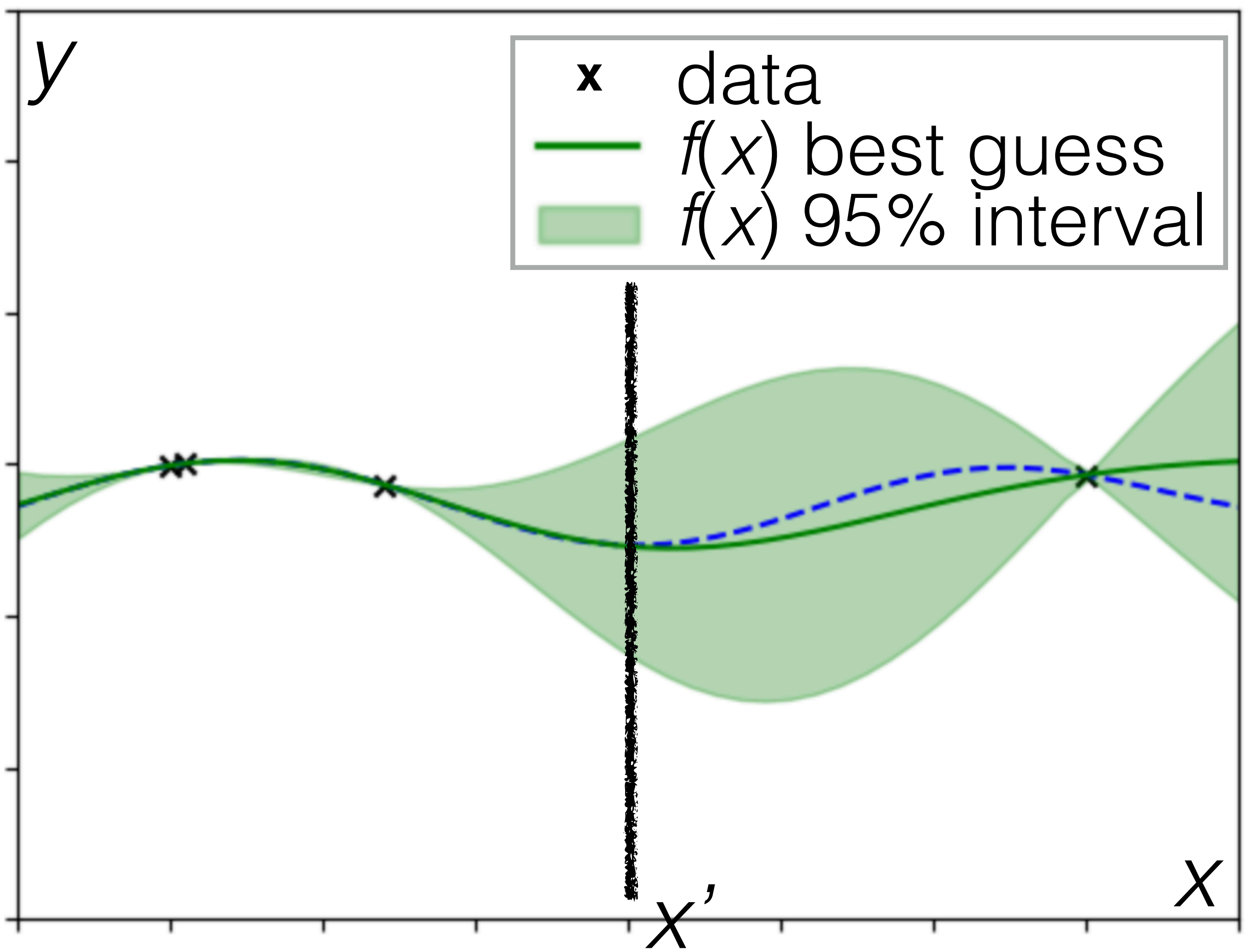


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

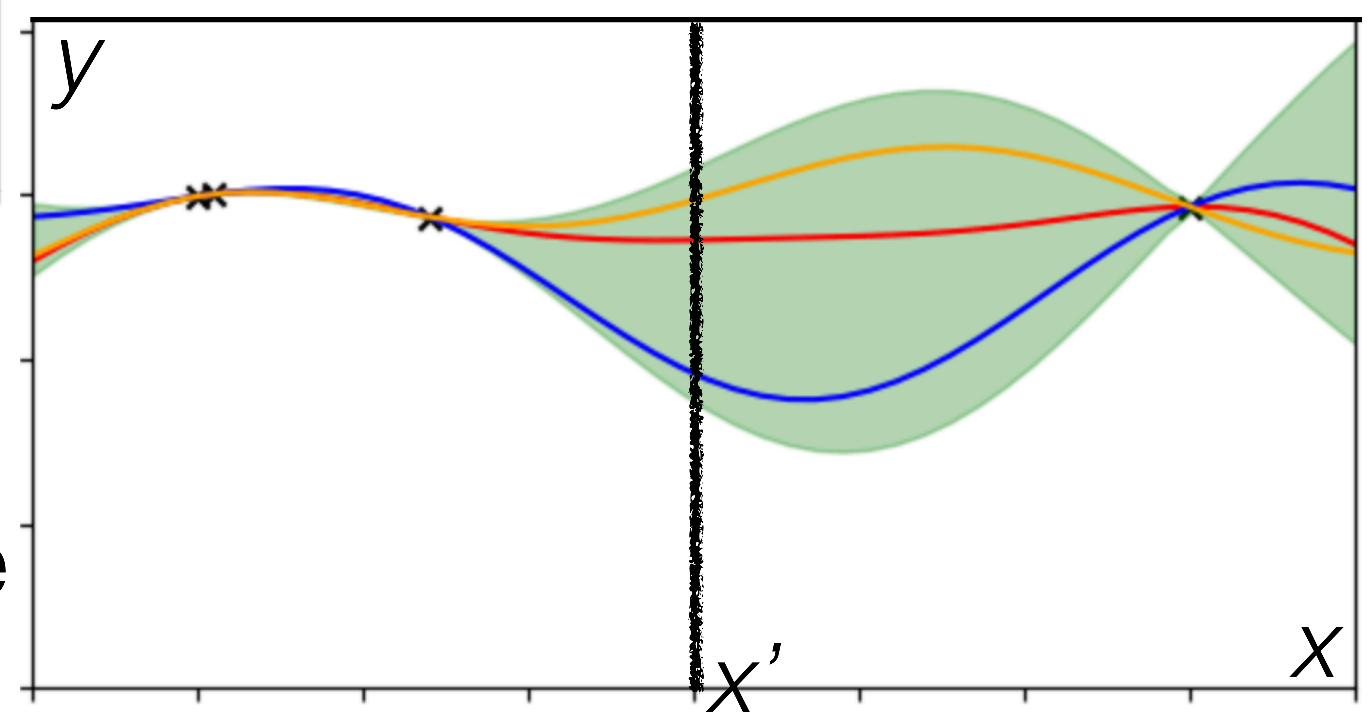


- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ?

Closer look at the uncertainty interval

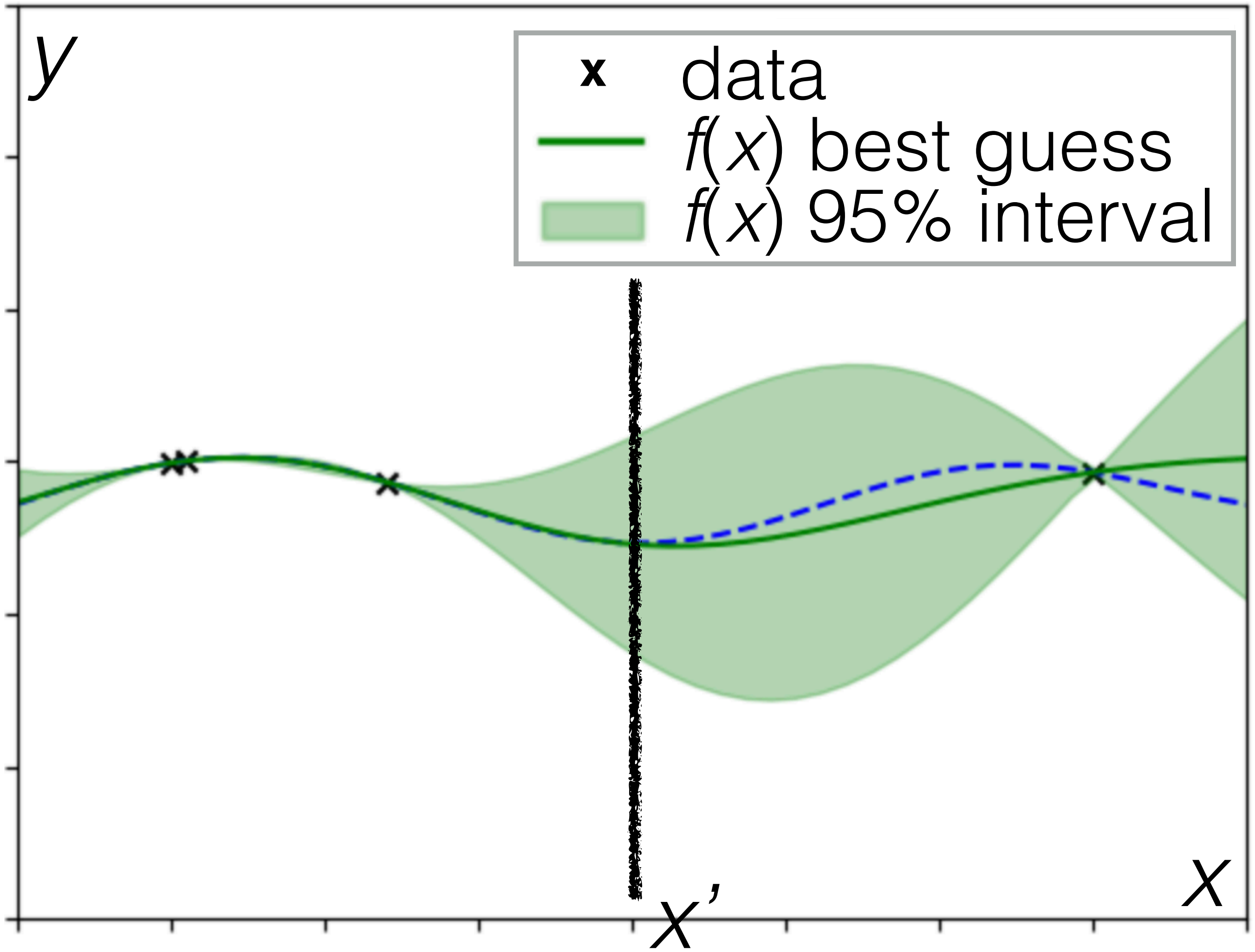


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

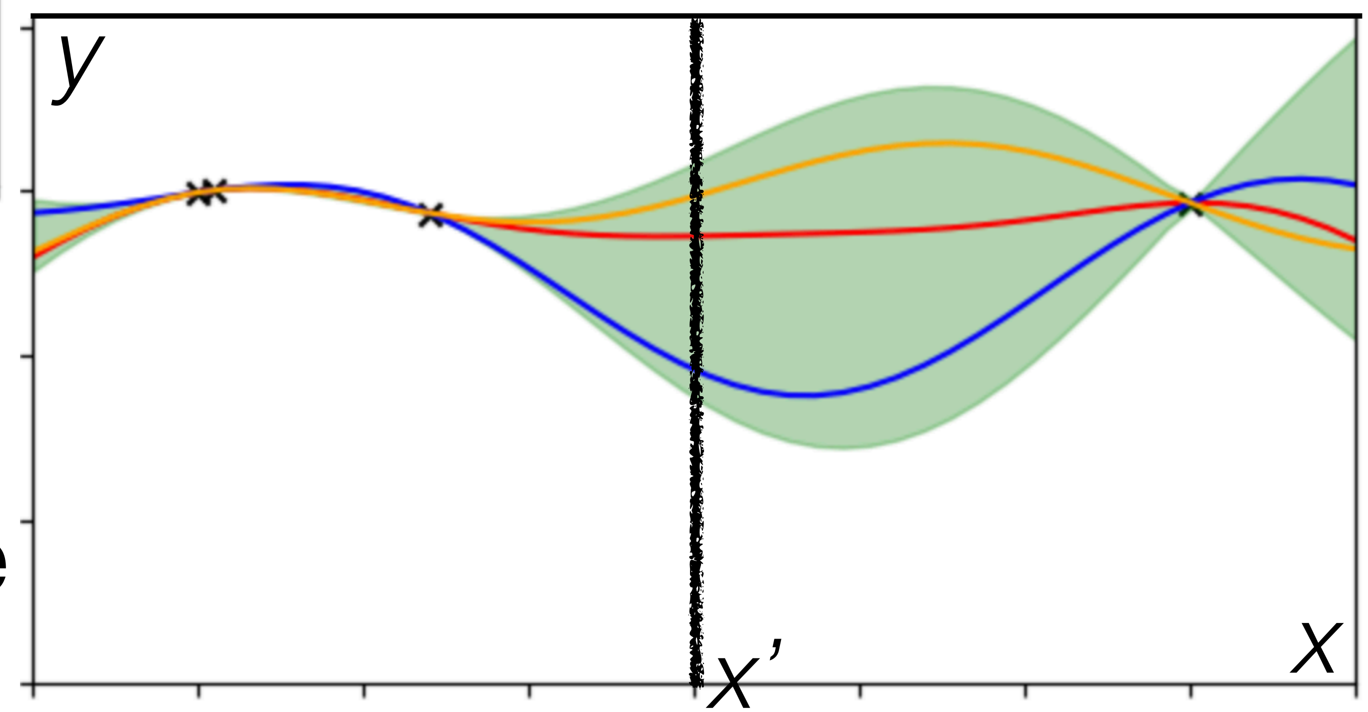


- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ~95%

Closer look at the uncertainty interval

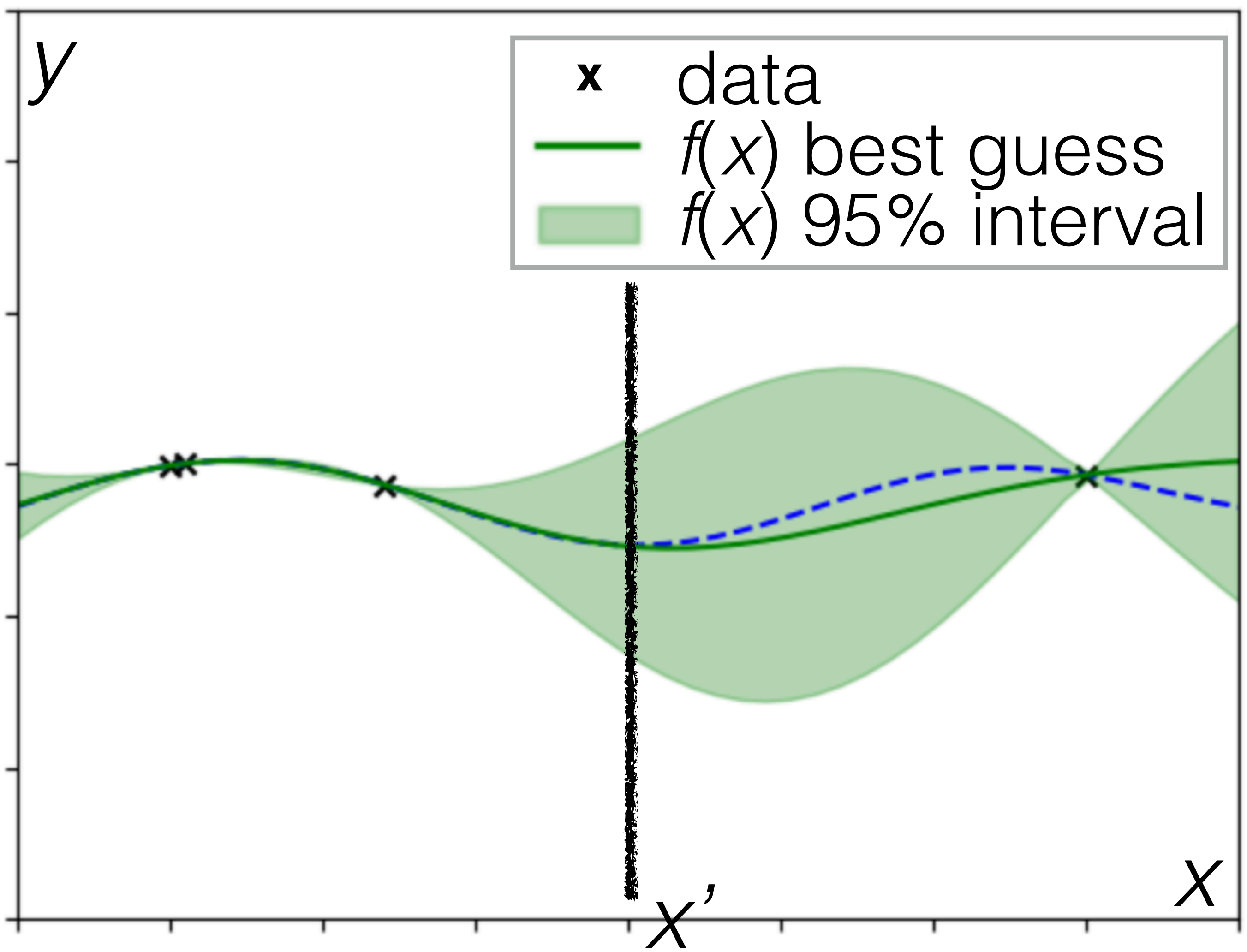


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

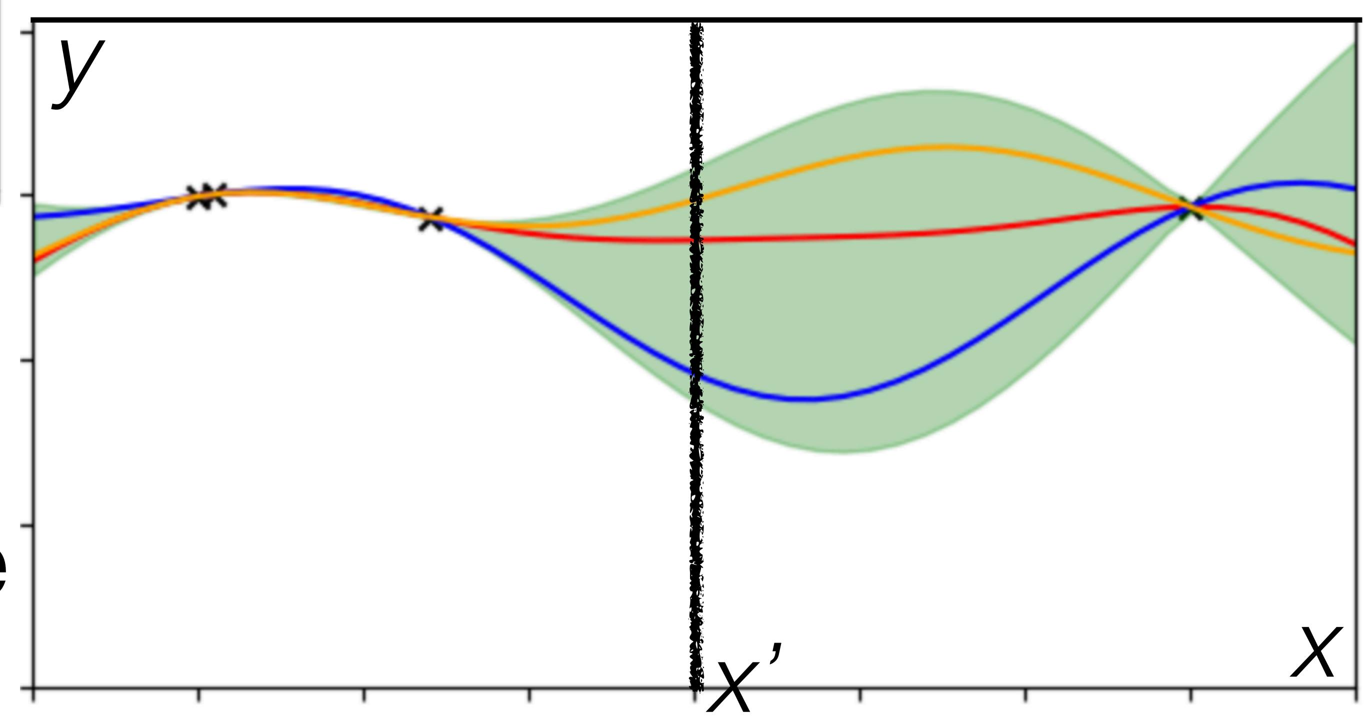


- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ~95%
- Probability that all points on *f* fall within the green interval across the whole plot ? ~95%

Closer look at the uncertainty interval

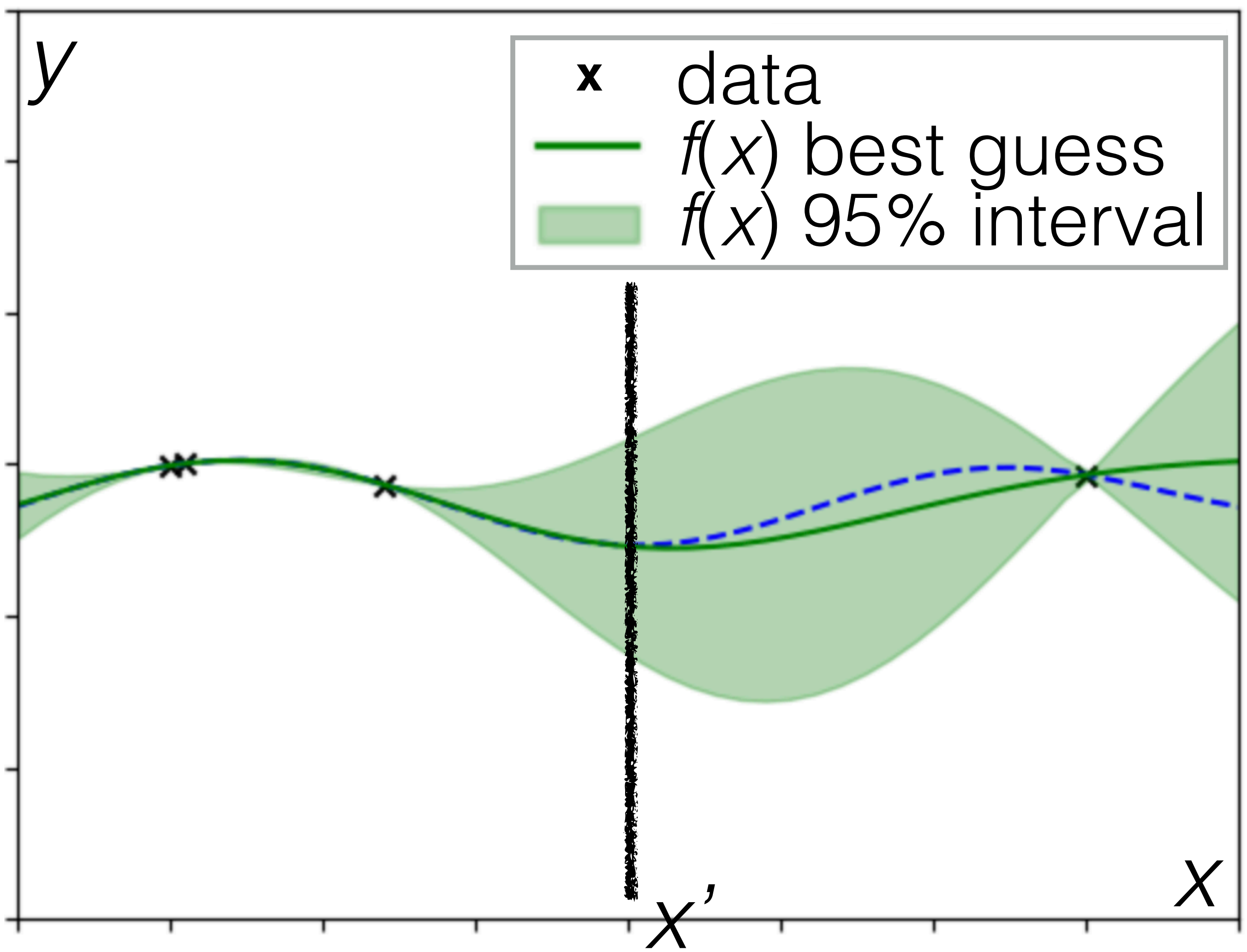


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

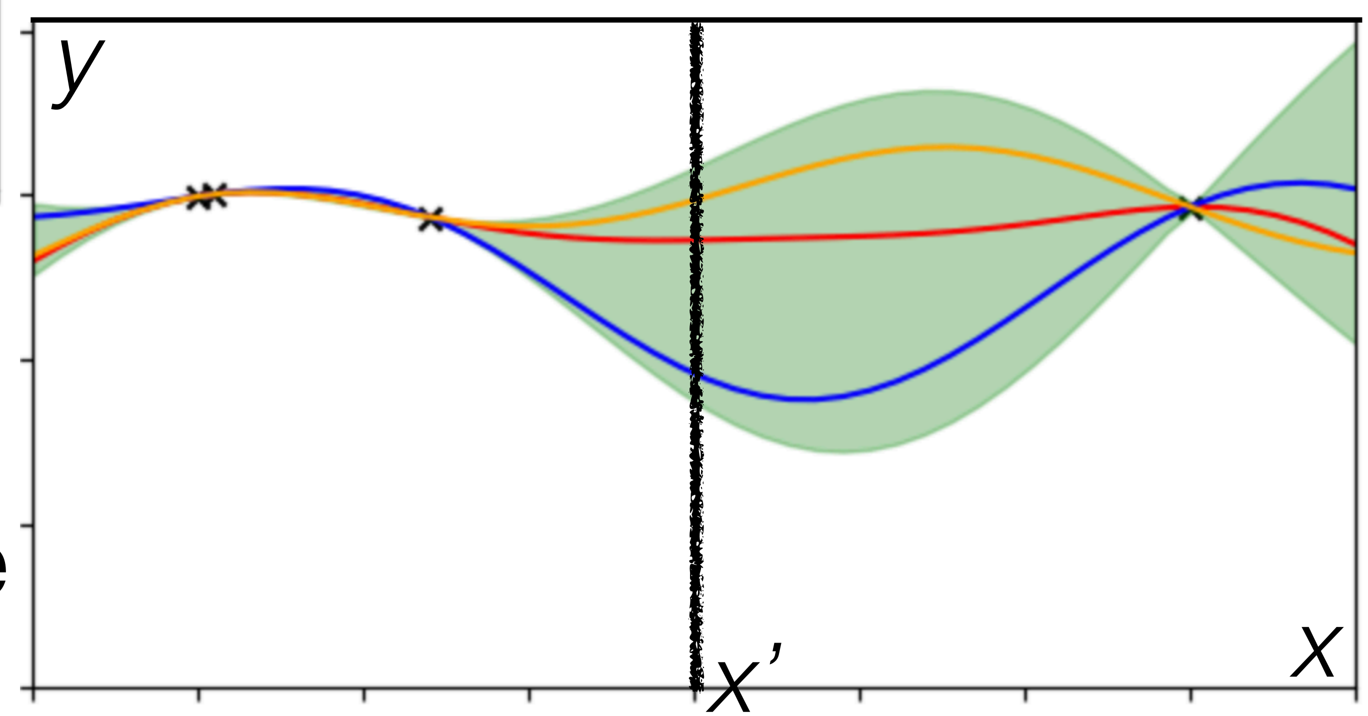


- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ~95%
- Probability that all points on *f* fall within the green interval across the whole plot will generally *not* be ~95%

Closer look at the uncertainty interval



- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs



- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ~95%
- Probability that all points on *f* fall within the green interval across the whole plot will generally *not* be ~95%

Squared exponential kernel revisited

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo]

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo]
- We've been using this particular kernel:

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

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo]
- We've been using this particular kernel:

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

  - What do we expect from the scale of $f(x)$ a priori?

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo]
- We've been using this particular kernel:
$$k(x, x') = \sigma^2 \exp(-\tfrac{1}{2}(x - x')^2), \sigma = 1$$
  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in$?

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo]
- We've been using this particular kernel:

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

  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo]
- We've been using this particular kernel:

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

  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$
    - Marginal variance cannot increase with data

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo1, demo2]
- We've been using this particular kernel:

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

  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$
    - Marginal variance cannot increase with data

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo1, demo2]
- We've been using this particular kernel:

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

    - What do we expect from the scale of *f*(*x*) a priori?
        - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$
        - Marginal variance cannot increase with data
    - What counts as "close" in *x*?

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo1, demo2]
- We've been using this particular kernel:

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

  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$
    - Marginal variance cannot increase with data
  - What counts as "close" in *x*?

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo1, demo2]
- We've been using this particular kernel:

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

  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$
    - Marginal variance cannot increase with data
  - What counts as "close" in *x*?

$$\exp(-\tfrac{1}{2}2^2) \approx 0.14 \quad \exp(-\tfrac{1}{2}3^2) \approx 0.011 \quad \exp(-\tfrac{1}{2}4^2) \approx 0.00034$$

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo1, demo2]
- We've been using this particular kernel:

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

  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$
    - Marginal variance cannot increase with data
  - What counts as "close" in *x*?

$$\exp(-\tfrac{1}{2}2^2) \approx 0.14 \quad \exp(-\tfrac{1}{2}3^2) \approx 0.011 \quad \exp(-\tfrac{1}{2}4^2) \approx 0.00034$$

- What can we do to handle different *x* and *f*(*x*) scales?

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo1, demo2]
- We've been using this particular kernel:

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

  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$
    - Marginal variance cannot increase with data
  - What counts as "close" in *x*?

$$\exp(-\tfrac{1}{2}2^2) \approx 0.14 \quad \exp(-\tfrac{1}{2}3^2) \approx 0.011 \quad \exp(-\tfrac{1}{2}4^2) \approx 0.00034$$

- What can we do to handle different *x* and *f*(*x*) scales?
  - Normalization in *y* can help; in *x*, can still be hiccups

Squared exponential kernel revisited

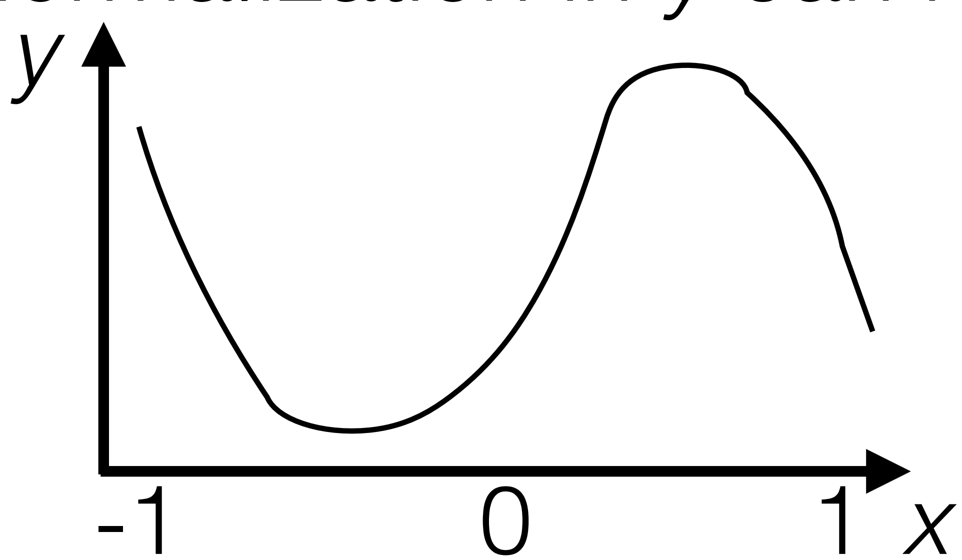
- What if we happened to measure our data on a different scale? [demo1, demo2]
- We've been using this particular kernel:

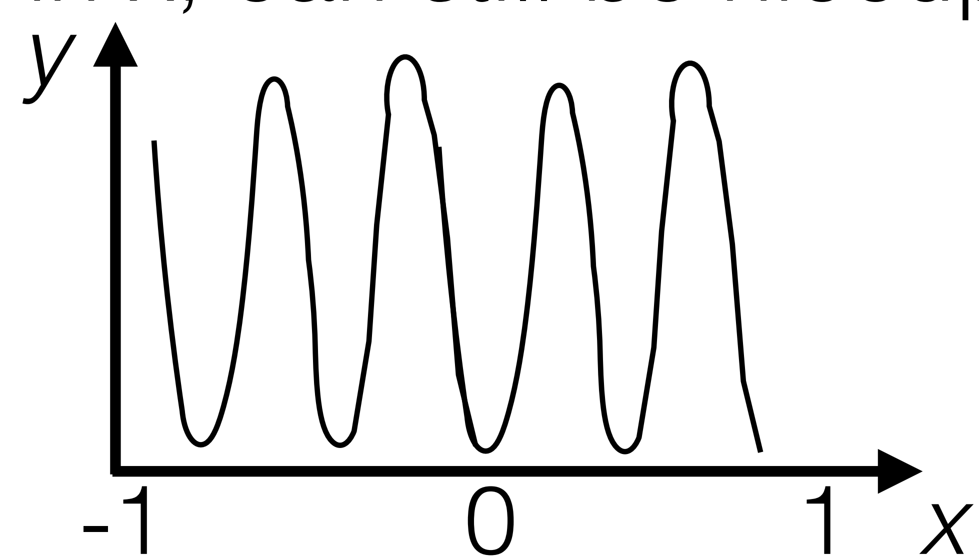
$$k(x, x') = \sigma^2 \exp(-\tfrac{1}{2}(x - x')^2), \sigma = 1$$

  - What do we expect from the scale of *f*(*x*) a priori?
    - At one *x*, with ~95% probability a priori, $f(x) \in (-2, 2)$
    - Marginal variance cannot increase with data
  - What counts as "close" in *x*?

$$\exp(-\tfrac{1}{2}2^2) \approx 0.14 \quad \exp(-\tfrac{1}{2}3^2) \approx 0.011 \quad \exp(-\tfrac{1}{2}4^2) \approx 0.00034$$

- What can we do to handle different *x* and *f*(*x*) scales?
  - Normalization in *y* can help; in *x*, can still be hiccups





Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\ell_d^2})$$

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \underbrace{\boxed{\sigma^2}}_{\text{signal variance}} \exp\left(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\underbrace{\boxed{\ell_d^2}}_{\text{lengthscales}}}\right)$$

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \underbrace{\sigma^2}_{\text{signal variance}} \exp(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\underbrace{\ell_d^2}_{\text{lengthscales}}})$$

  - *Parameters* (here, *f*) parametrize the distribution of the data. If we knew them, we could generate the data.

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \underbrace{\sigma^2}_{\text{signal variance}} \exp(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\underbrace{\ell_d^2}_{\text{lengthscales}}})$$

  - *Parameters* (here, *f*) parametrize the distribution of the data. If we knew them, we could generate the data.
    - GPs: *nonparametric* model: infinite # of latent params

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \underbrace{\sigma^2}_{\text{signal variance}} \exp(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{\underbrace{\ell_d^2}_{\text{lengthscales}}})$$

  - *Parameters* (here, *f*) parametrize the distribution of the data. If we knew them, we could generate the data.
    - GPs: *nonparametric* model: infinite # of latent params
  - *Hyperparameters* parametrize the distribution of the parameters. If known, we could generate the parameters.

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\ell_d^2})$$

signal variance   lengthscales

  - *Parameters* (here, *f*) parametrize the distribution of the data. If we knew them, we could generate the data.
    - GPs: *nonparametric* model: infinite # of latent params
  - *Hyperparameters* parametrize the distribution of the parameters. If known, we could generate the parameters.
- Algorithm:

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \underbrace{\sigma^2}_{\text{signal variance}} \exp(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\underbrace{\ell_d^2}_{\text{lengthscales}}})$$

  - *Parameters* (here, *f*) parametrize the distribution of the data. If we knew them, we could generate the data.
    - GPs: *nonparametric* model: infinite # of latent params
  - *Hyperparameters* parametrize the distribution of the parameters. If known, we could generate the parameters.
- Algorithm:
  - Fit a value for the hyperparameters using the data.

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \underbrace{\sigma^2}_{\text{signal variance}} \exp\left(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\underbrace{\ell_d^2}_{\text{lengthscales}}}\right)$$

  - *Parameters* (here, *f*) parametrize the distribution of the data. If we knew them, we could generate the data.
    - GPs: *nonparametric* model: infinite # of latent params
  - *Hyperparameters* parametrize the distribution of the parameters. If known, we could generate the parameters.
- Algorithm:
  - Fit a value for the hyperparameters using the data.
  - Given those values, now compute and report the mean and uncertainty intervals.

Squared exponential kernel revisited

- A common option in practice and in software is to fit the *hyperparameters* of a more general *squared exponential kernel* from data
  - More general form of the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\ell_d^2})$$

signal variance lengthscales

  - *Parameters* (here, *f*) parametrize the distribution of the data. If we knew them, we could generate the data.
    - GPs: *nonparametric* model: infinite # of latent params
  - *Hyperparameters* parametrize the distribution of the parameters. If known, we could generate the parameters.
- Algorithm:
  - Fit a value for the hyperparameters using the data.
  - Given those values, now compute and report the mean and uncertainty intervals. [demo1,2,3]

Observation noise

Observation noise

- So far we've been assuming that we observed $f(x)$ directly

Observation noise

- So far we've been assuming that we observed $f(x)$ directly
- But often the actual observation y has additional noise:

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*

[demo1]

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed

[demo1]

Why?

Observation noise

- So far we've been assuming that we observed $f(x)$ directly
- But often the actual observation y has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent f
  - The y's are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is ?

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$\mathrm{Cov}(y^{(n)}, y^{(n')}) =$?

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$ and want to learn the latent *f*

[demo1]

  - The *y*'s are multivariate-Gaussian-distributed
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m,k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0,\tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$ and want to learn the latent *f*

[demo1]

  - The *y*'s are multivariate-Gaussian-distributed
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$
- Now: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix}$

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m,k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$
- Now: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X)+\tau^2 I & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$
- Now: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \tau^2 I & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m,k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$
- Now: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \tau^2 I & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X') \\ K(X', X) & K(X', X') \end{bmatrix}\right)$
- Now: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \tau^2 I & K(X, X') \\ K(X', X) & K(X', X') \end{bmatrix}\right)$

What if we put *y* here instead?

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*

[demo1]

  - The *y*'s are multivariate-Gaussian-distributed
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$
- Now: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \tau^2 I & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$

[demo2, demo3]

Observation noise

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed [demo1]
    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$
- Now: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \tau^2 I & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$

Can you state a non-trivial lower bound on the marginal variance of a test $y^{(m)}$?

[demo2, demo3]

Observation noise

Even when observations are "perfect," use a (very small) *nugget* for numerical reasons

- So far we've been assuming that we observed *f*(*x*) directly
- But often the actual observation *y* has additional noise:

$$f \sim \mathcal{GP}(m, k), y^{(n)} \sim f(\mathbf{x}^{(n)}) + \epsilon^{(n)}, \epsilon^{(n)} \overset{iid}{\sim} \mathcal{N}(0, \tau^2)$$

  - We observe $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ and want to learn the latent *f*
  - The *y*'s are multivariate-Gaussian-distributed

[demo1]

    - Note: the sum of independent Gaussians is a Gaussian with means summed and covariances summed
    - So the mean of $y^{(n)}$ is $m(\mathbf{x}^{(n)})$ and

$$\mathrm{Cov}(y^{(n)}, y^{(n')}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \tau^2 \mathbf{1}\{n = n'\}$$

Why compare indices, not *x*'s?

- Before: $\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$
- Now: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \tau^2 I & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$

Can you state a non-trivial lower bound on the marginal variance of a test $y^{(m)}$?

[demo2, demo3]

What uncertainty are we quantifying?

What uncertainty are we quantifying?

- It’s worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art

What uncertainty are we quantifying?

- It’s worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art
  - E.g. “significance”, “bias”, “generalization”

What uncertainty are we quantifying?

- It’s worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art
  - E.g. “significance”, “bias”, “generalization”
  - Every precise use of “uncertainty” has this issue
    - E.g. frequentist sampling, Bayesian, etc.

What uncertainty are we quantifying?

- It’s worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art
  - E.g. “significance”, “bias”, “generalization”
  - Every precise use of “uncertainty” has this issue
    - E.g. frequentist sampling, Bayesian, etc.
  - We should always make sure we can distinguish what is, and what is not, covered by the term of art

What uncertainty are we quantifying?

- It's worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art
  - E.g. "significance", "bias", "generalization"
  - Every precise use of "uncertainty" has this issue
    - E.g. frequentist sampling, Bayesian, etc.
  - We should always make sure we can distinguish what is, and what is not, covered by the term of art
- A standard setup (our setup so far):

What uncertainty are we quantifying?

- It's worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art
  - E.g. "significance", "bias", "generalization"
  - Every precise use of "uncertainty" has this issue
    - E.g. frequentist sampling, Bayesian, etc.
  - We should always make sure we can distinguish what is, and what is not, covered by the term of art
- A standard setup (our setup so far):
  - We model the data as generated according to a GP with squared exponential kernel and observation noise

What uncertainty are we quantifying?

- It’s worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art
  - E.g. “significance”, “bias”, “generalization”
  - Every precise use of “uncertainty” has this issue
    - E.g. frequentist sampling, Bayesian, etc.
  - We should always make sure we can distinguish what is, and what is not, covered by the term of art
- A standard setup (our setup so far):
  - We model the data as generated according to a GP with squared exponential kernel and observation noise
  - We fit the hyperparameters (the signal variance, the length scale(s), and the noise variance) to single values

What uncertainty are we quantifying?

- It's worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art
  - E.g. "significance", "bias", "generalization"
  - Every precise use of "uncertainty" has this issue
    - E.g. frequentist sampling, Bayesian, etc.
  - We should always make sure we can distinguish what is, and what is not, covered by the term of art
- A standard setup (our setup so far):
  - We model the data as generated according to a GP with squared exponential kernel and observation noise
  - We fit the hyperparameters (the signal variance, the length scale(s), and the noise variance) to single values
  - The reported uncertainties are what result when the GP model and fitted hyperparameters are exactly correct

What uncertainty are we quantifying?

- It’s worth being aware that data science (ML/stats/AI) often overloads common colloquial terms with terms of art
  - E.g. “significance”, “bias”, “generalization”
  - Every precise use of “uncertainty” has this issue
    - E.g. frequentist sampling, Bayesian, etc.
  - We should always make sure we can distinguish what is, and what is not, covered by the term of art
- A standard setup (our setup so far):
  - We model the data as generated according to a GP with squared exponential kernel and observation noise
  - We fit the hyperparameters (the signal variance, the length scale(s), and the noise variance) to single values
  - The reported uncertainties are what result when the GP model and fitted hyperparameters are exactly correct

Are there other uncertainties that aren’t being quantified here?

Some other sources of uncertainty

[demo1,2,3]

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do?

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

[demo1,2]

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

[demo1,2]

- A GP with your mean & kernel may be meaningfully misspecified for the data

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

[demo1,2]

- A GP with your mean & kernel may be meaningfully misspecified for the data (is your model what you think it is? check defaults!)

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

[demo1,2]

- A GP with your mean & kernel may be meaningfully misspecified for the data (is your model what you think it is? check defaults!)
  - Box: “All models are wrong, but some are useful”

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

[demo1,2]

- A GP with your mean & kernel may be meaningfully misspecified for the data (is your model what you think it is? check defaults!)
  - Box: "All models are wrong, but some are useful"
  - What can we do?

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

[demo1,2]

- A GP with your mean & kernel may be meaningfully misspecified for the data (is your model what you think it is? check defaults!)
  - Box: “All models are wrong, but some are useful”
  - What can we do? First: unit test, plot, sense check!

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

[demo1,2]

- A GP with your mean & kernel may be meaningfully misspecified for the data (is your model what you think it is? check defaults!)
  - Box: “All models are wrong, but some are useful”
  - What can we do? First: unit test, plot, sense check!
    - Can change the mean and/or kernel
      - E.g. local/heteroskedastic models, periodic kernels, linear mean function, many many more

Some other sources of uncertainty

[demo1,2,3]

- There may be multiple sets of substantively different hyperparameter values that are both plausible and consistent with the observed data
  - What can we do? First: unit test, plot, sense check!
    - Ask what is possible to learn with the data available
    - Multiple random restarts: plot the results
    - Bayesian model of the hyperparameters → be careful: expense & interpretation

[demo1,2]

- A GP with your mean & kernel may be meaningfully misspecified for the data (is your model what you think it is? check defaults!)
  - Box: “All models are wrong, but some are useful”
  - What can we do? First: unit test, plot, sense check!
    - Can change the mean and/or kernel
      - E.g. local/heteroskedastic models, periodic kernels, linear mean function, many many more

Extrapolation

Extrapolation

- *Extrapolation*: Estimation/prediction beyond the observed data

Extrapolation

- *Extrapolation*: Estimation/prediction beyond the observed data
  - Compare to *interpolation*: estimation/prediction within the observed data

Extrapolation

- *Extrapolation*: Estimation/prediction beyond the observed data
    - Compare to *interpolation*: estimation/prediction within the observed data [demo1,2]

Extrapolation

- *Extrapolation*: Estimation/prediction beyond the observed data
  - Compare to *interpolation*: estimation/prediction within the observed data [demo1,2]
- When using GPs with a squared exponential kernel:
  - Data points that are more than a handful of length scales from other data points will revert to prior behavior

Extrapolation

- *Extrapolation*: Estimation/prediction beyond the observed data
    - Compare to *interpolation*: estimation/prediction within the observed data [demo1,2]
- When using GPs with a squared exponential kernel:
    - Data points that are more than a handful of length scales from other data points will revert to prior behavior
- Note: extrapolation isn't a special issue unique to GPs. It's a fundamentally hard problem for all data analysis methods

Extrapolation

- *Extrapolation*: Estimation/prediction beyond the observed data
  - Compare to *interpolation*: estimation/prediction within the observed data [demo1,2]
- When using GPs with a squared exponential kernel:
  - Data points that are more than a handful of length scales from other data points will revert to prior behavior
- Note: extrapolation isn't a special issue unique to GPs. It's a fundamentally hard problem for all data analysis methods
  - To extrapolate, you need to make assumptions

Extrapolation

- *Extrapolation*: Estimation/prediction beyond the observed data
  - Compare to *interpolation*: estimation/prediction within the observed data [demo1,2]
- When using GPs with a squared exponential kernel:
  - Data points that are more than a handful of length scales from other data points will revert to prior behavior
- Note: extrapolation isn't a special issue unique to GPs. It's a fundamentally hard problem for all data analysis methods
  - To extrapolate, you need to make assumptions
    - When you have domain knowledge of a system, you might be able to use it to extrapolate

Extrapolation

- *Extrapolation*: Estimation/prediction beyond the observed data
  - Compare to *interpolation*: estimation/prediction within the observed data [demo1,2]
- When using GPs with a squared exponential kernel:
  - Data points that are more than a handful of length scales from other data points will revert to prior behavior
- Note: extrapolation isn’t a special issue unique to GPs. It’s a fundamentally hard problem for all data analysis methods
  - To extrapolate, you need to make assumptions
    - When you have domain knowledge of a system, you might be able to use it to extrapolate
    - When you’re letting a machine learning method use its defaults, it’s making assumptions. Do you know what those assumptions are?

More than one input

More than one input

- Our illustrations have almost all been for one input so far

More than one input

- Our illustrations have almost all been for one input so far
- But in real life, it's typical to have more than one input

More than one input

- Our illustrations have almost all been for one input so far
- But in real life, it's typical to have more than one input
- What could go wrong?

More than one input

- Our illustrations have almost all been for one input so far
- But in real life, it’s typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
    - Possibly different lengthscales. Check defaults.

More than one input

- Our illustrations have almost all been for one input so far
- But in real life, it’s typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
  - Possibly different lengthscales. Check defaults.
  - Regression in high dimensions is a fundamentally hard problem (without additional assumptions)

More than one input

- Our illustrations have almost all been for one input so far
- But in real life, it's typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
  - Possibly different lengthscales. Check defaults.
  - Regression in high dimensions is a fundamentally hard problem (without additional assumptions)
- All points are "far away" in high dimensions. Illustration:

More than one input

- Our illustrations have almost all been for one input so far
- But in real life, it’s typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
    - Possibly different lengthscales. Check defaults.
    - Regression in high dimensions is a fundamentally hard problem (without additional assumptions)
- All points are “far away” in high dimensions. Illustration:
    - Uniformly randomly sample 10,000 points on $[0,1]^D$

[Binois, Wycoff 2022]

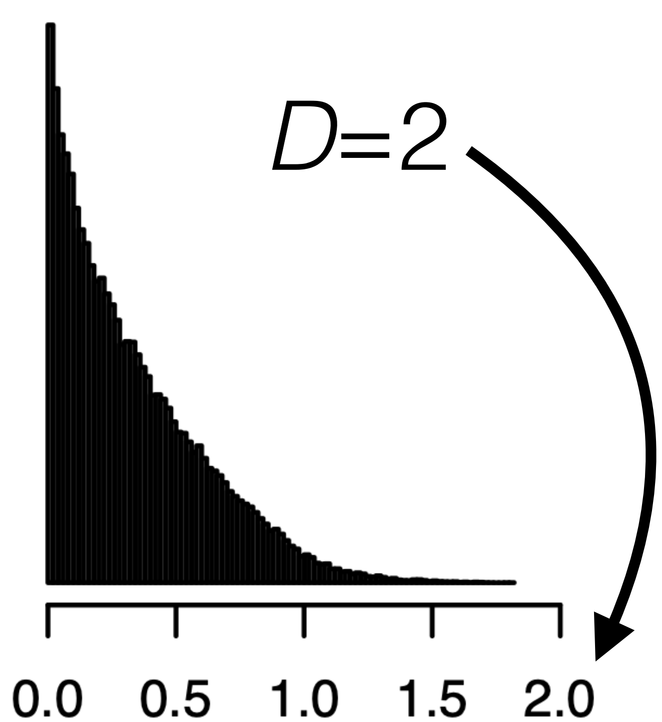
More than one input

- Our illustrations have almost all been for one input so far
- But in real life, it's typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
  - Possibly different lengthscales. Check defaults.
  - Regression in high dimensions is a fundamentally hard problem (without additional assumptions)
- All points are "far away" in high dimensions. Illustration:
  - Uniformly randomly sample 10,000 points on $[0,1]^D$
  - Make a histogram of squared inter-point distances

[Binois, Wycoff 2022]

More than one input

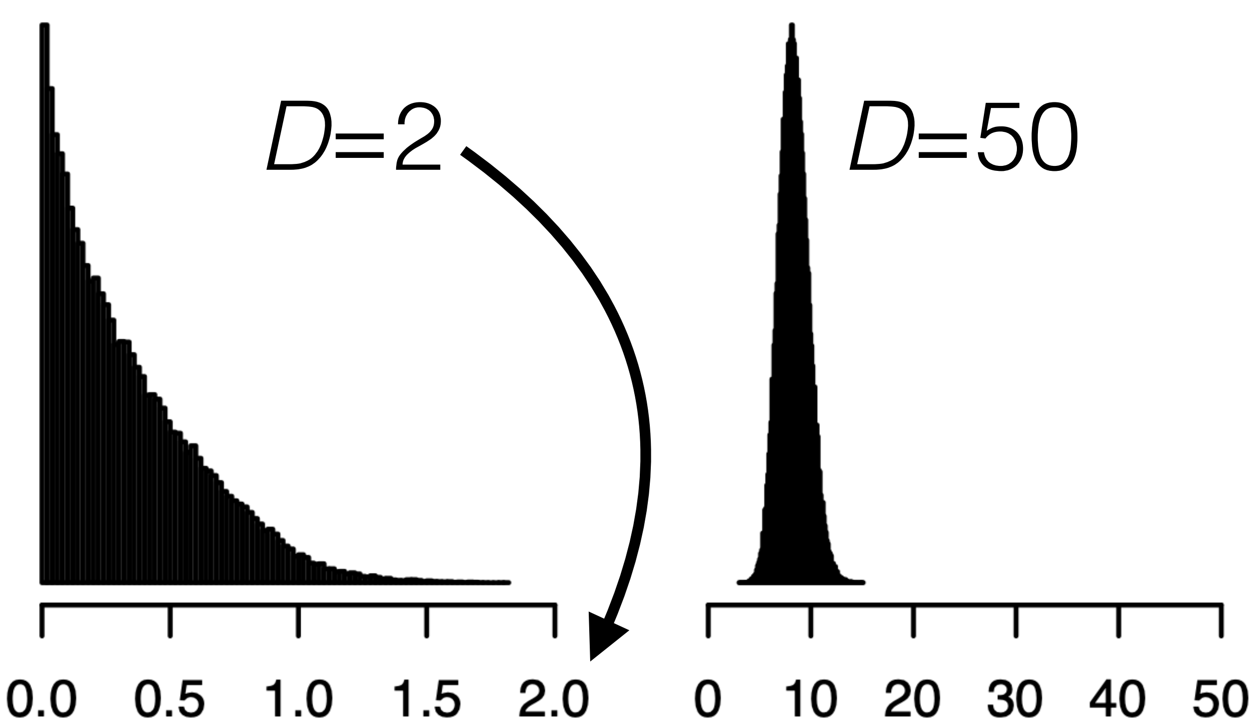
- Our illustrations have almost all been for one input so far
- But in real life, it's typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
    - Possibly different lengthscales. Check defaults.
    - Regression in high dimensions is a fundamentally hard problem (without additional assumptions)
- All points are "far away" in high dimensions. Illustration:
    - Uniformly randomly sample 10,000 points on $[0,1]^D$
    - Make a histogram of squared inter-point distances



[Binois, Wycoff 2022]

More than one input

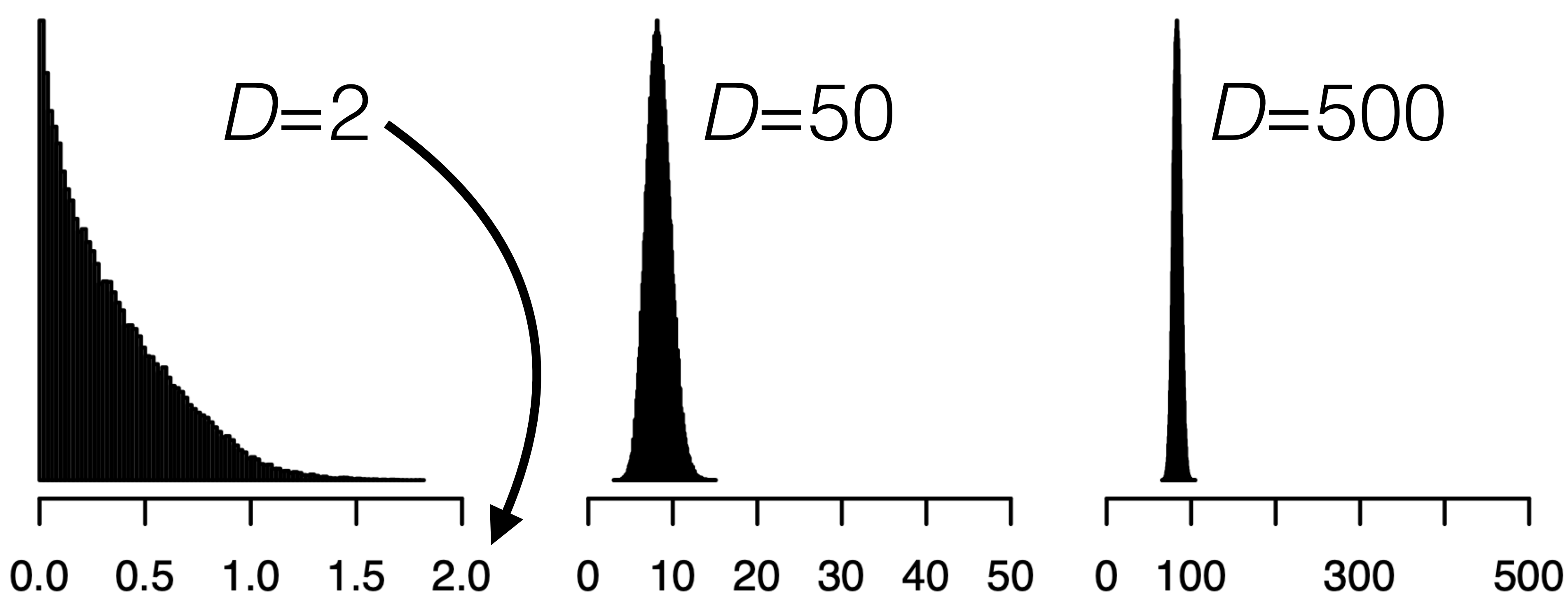
- Our illustrations have almost all been for one input so far
- But in real life, it's typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
  - Possibly different lengthscales. Check defaults.
  - Regression in high dimensions is a fundamentally hard problem (without additional assumptions)
- All points are "far away" in high dimensions. Illustration:
  - Uniformly randomly sample 10,000 points on $[0,1]^D$
  - Make a histogram of squared inter-point distances



[Binois, Wycoff 2022]

More than one input

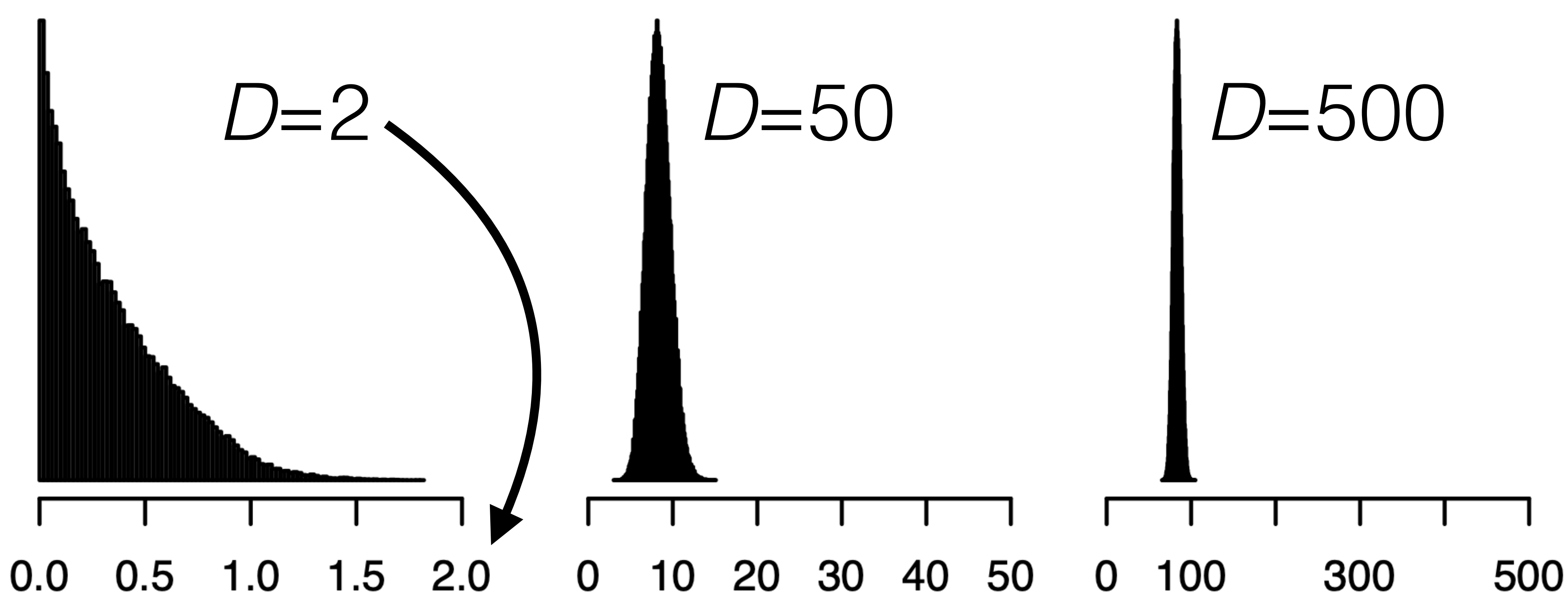
- Our illustrations have almost all been for one input so far
- But in real life, it's typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
  - Possibly different lengthscales. Check defaults.
  - Regression in high dimensions is a fundamentally hard problem (without additional assumptions)
- All points are "far away" in high dimensions. Illustration:
  - Uniformly randomly sample 10,000 points on $[0,1]^D$
  - Make a histogram of squared inter-point distances



[Binois, Wycoff 2022]

More than one input

- Our illustrations have almost all been for one input so far
- But in real life, it's typical to have more than one input
- What could go wrong? Previous lessons apply, but also:
  - Possibly different lengthscales. Check defaults.
  - Regression in high dimensions is a fundamentally hard problem (without additional assumptions)
- All points are "far away" in high dimensions. Illustration:
  - Uniformly randomly sample 10,000 points on $[0,1]^D$
  - Make a histogram of squared inter-point distances



[Binois, Wycoff 2022]

- Recall: points "far" from data default to the prior mean and variance

Some high points of what got cut for time

- We ran out of time! Here are some high-level summary points beyond what we discussed together:
  - Running time for GP regression can be an issue with a large number of training data points
    - In particular, the matrix inverse can be expensive
    - There are incredibly many papers about fast approximations to the exact Gaussian process
      - Each approximation has pros and cons
- Bayesian optimization inherits many of the pros and cons of Gaussian processes for regression
  - Exercise: once you learn about Bayesian optimization, think about how the pros and cons we discussed together might translate there

Roadmap

- A Bayesian approach
- What is a Gaussian process?
  - Popular version using a squared exponential kernel
- Gaussian process inference
  - Prediction & uncertainty quantification
- What are the limits? What can go wrong?
- Bayesian optimization

- Goals:
  - Learn the mechanism behind standard GPs to identify benefits and pitfalls
  - Learn the skills to be responsible users of standard GPs (transferable to other ML/AI methods)

Some of our recent related work

- Can use arbitrary models in ML/AI/Stats if you can evaluate.
  - But popular validation methods assume iid data. A spatial solution: Burt, Shen, and Broderick. Consistent Validation for Predictive Methods in Spatial Settings. *AISTATS* 2025.
- Calibrated uncertainties in certain spatial settings: Burt*, Berlinghieri*, Bates, and Broderick. Smooth Sailing: Lipschitz-Driven Uncertainty Quantification for Spatial Association. arXiv:2502.06067.
- GPs + fluid dynamics: Berlinghieri, Trippe, Burt, Giordano, Srinivasan, Özgökmen, Xia, and Broderick. Gaussian processes at the Helm(holtz): A more fluid model for ocean currents. *ICML* 2023.
- Some checks for meaningful science: Broderick, Gelman, Meager, Smith, Zheng. Toward a taxonomy of trust for probabilistic machine learning, *Science Advances* 2023.

Resources

http://www.tamarabroderick.com/tutorials.html

- Rasmussen and Williams 2006. *Gaussian Processes for Machine Learning*. gaussianprocess.org/gpml/ Chs 1,2,4,5
- Gramacy 2020. *Surrogates: Gaussian process modeling, design and optimization for the applied sciences*. bookdown.org/rbg/surrogates/
- Frazier 2018. A Tutorial on Bayesian Optimization. arxiv.org/abs/1807.02811
- Garnett 2023. *Bayesian Optimization*. bayesoptbook.com/
- Software options include:
  - scikit-learn, GPy, GPflow, GPyTorch
- My setup for this tutorial: pip install X
  - X = jupyterlab, notebook, numpy, matplotlib, scikit-learn