MIT EECS
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MIT STATISTICS + DATA SCIENCE CENTER

LABORATORY FOR INFORMATION & DECISION SYSTEMS
LIDS

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 (GP)?

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference
  - Prediction & uncertainty quantification

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference
  - Prediction & uncertainty quantification
- Using GPs to optimize a function (Bayesian optimization)

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference
  - Prediction & uncertainty quantification
- Using GPs to optimize a function (Bayesian optimization)
- What are the limits? What can go wrong?

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference
  - Prediction & uncertainty quantification
- Using GPs to optimize a function (Bayesian optimization)
- What are the limits? What can go wrong?

- Goals:

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference
  - Prediction & uncertainty quantification
- Using GPs to optimize a function (Bayesian optimization)
- What are the limits? What can go wrong?

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

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference
  - Prediction & uncertainty quantification
- Using GPs to optimize a function (Bayesian optimization)
- What are the limits? What can go wrong?

- 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 artificial intelligence/ machine learning 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 $f(x)$ is a collection indexed by input 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 $f(x)$ is a collection indexed by input 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 $f(x)$ is a collection indexed by input x
- It is specified by its mean function and covariance function:

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

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

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 $f(x)$ is a collection indexed by input x
- It is specified by its mean function and covariance function:

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

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

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

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 $f(x)$ is a collection indexed by input x
- It is specified by its mean function and covariance function:

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

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

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

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

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 $f(x)$ is a collection indexed by input x
- It is specified by its mean function and covariance function:

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

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

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

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

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

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 $f(x)$ is a collection indexed by input x
- It is specified by its mean function and covariance function:

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

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

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

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

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

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 $f(x)$ is a collection indexed by input x
- It is specified by its mean function and covariance function:

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

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

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

- A common default (e.g. in software) is $m(x) = 0$
- 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: $k(x, x') = \sigma^2 \exp(-\frac{1}{2}\|x - x'\|^2), \sigma = 1$

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

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 $f(x)$ is a collection indexed by input x
- It is specified by its mean function and covariance function:

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

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

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

- A common default (e.g. in software) is $m(x) = 0$
- 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: $k(x, x') = \sigma^2 \exp(-\frac{1}{2}\|x - x'\|^2), \sigma = 1$
- For now, assume data is observed without noise [demo1,2]

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

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

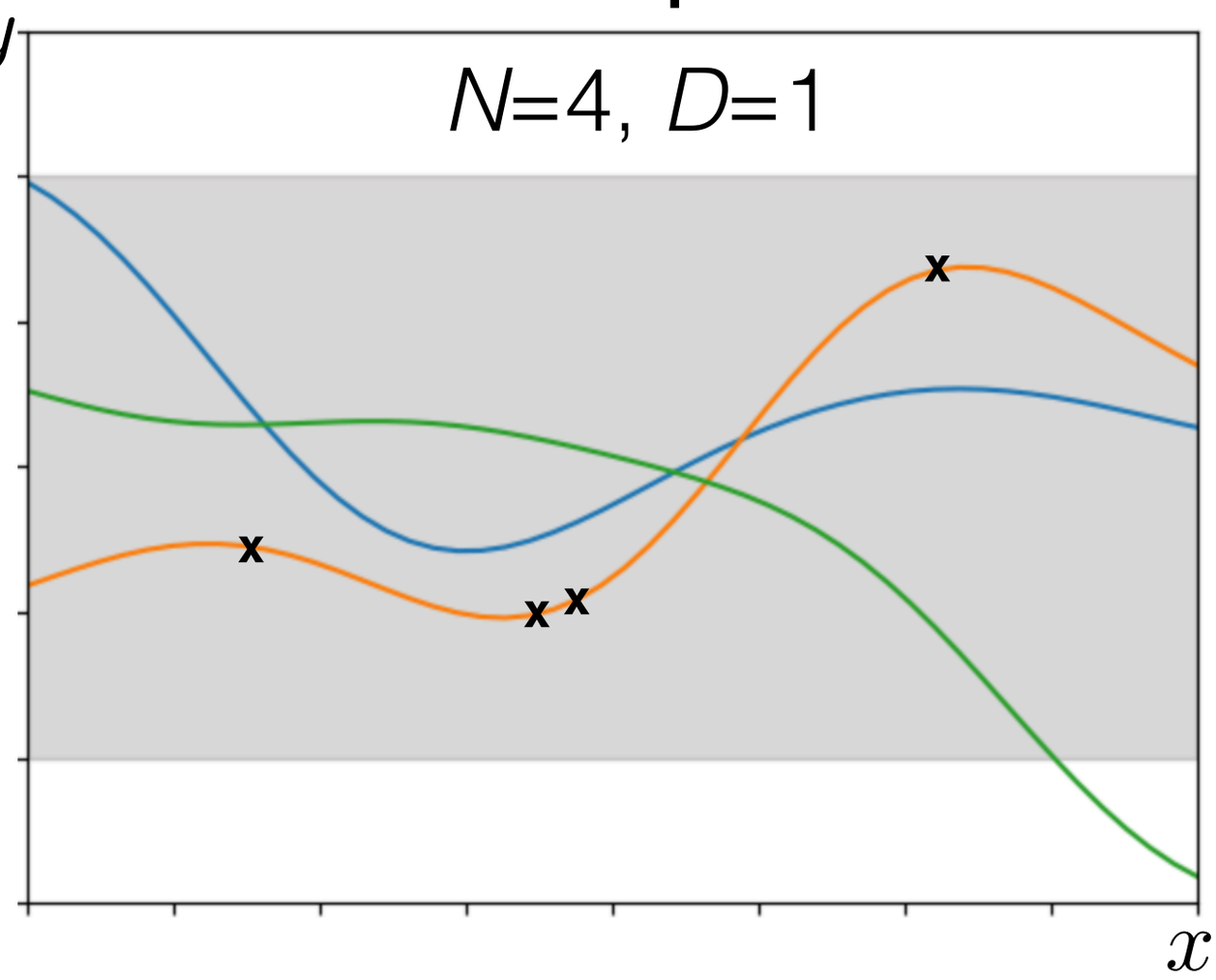
- Let’s be careful to separate two types of “dimension”
  - We’re using a superscript to denote (*M* or *N*) number of points in the space

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

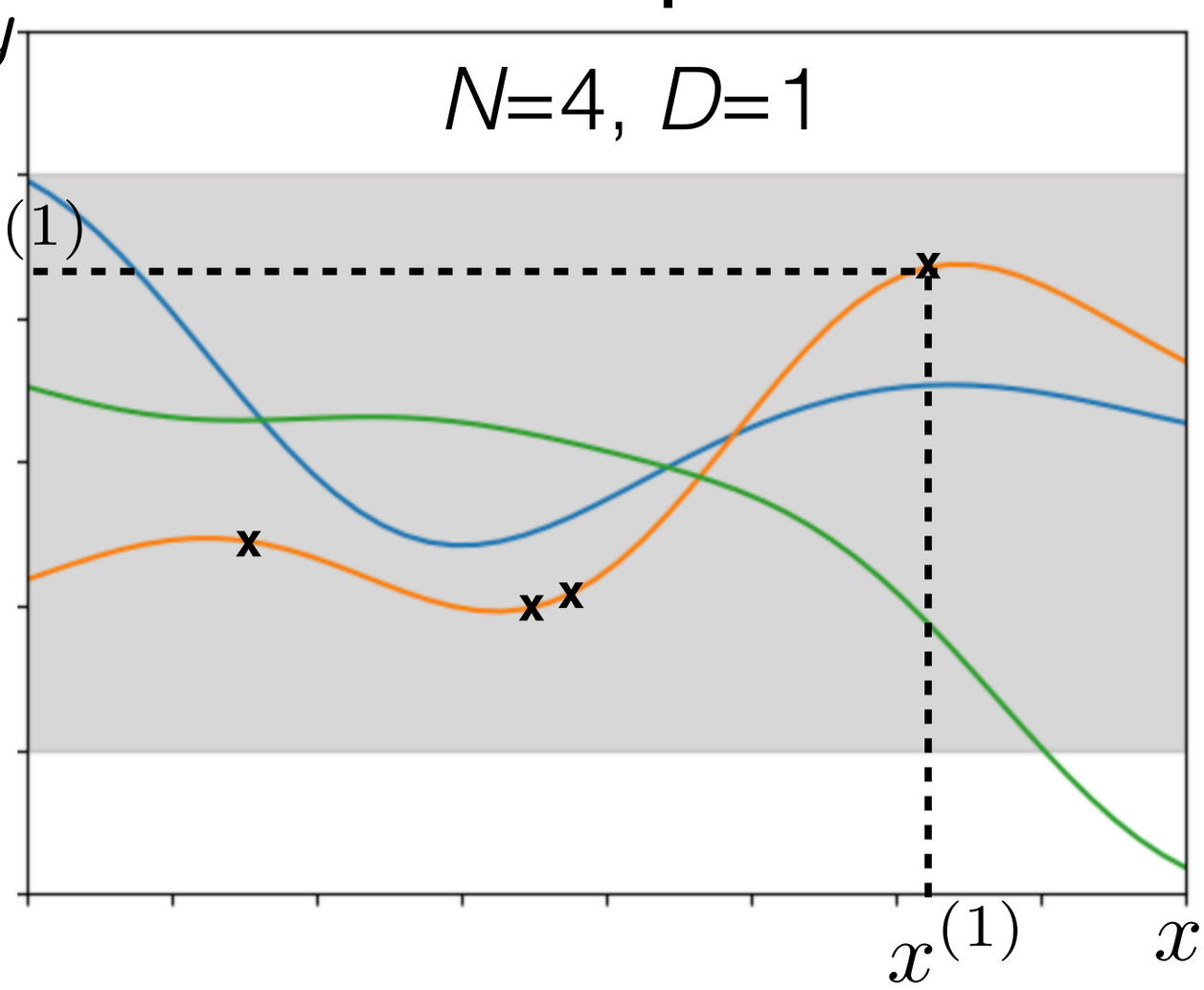
Note on "dimension" of the input

- 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



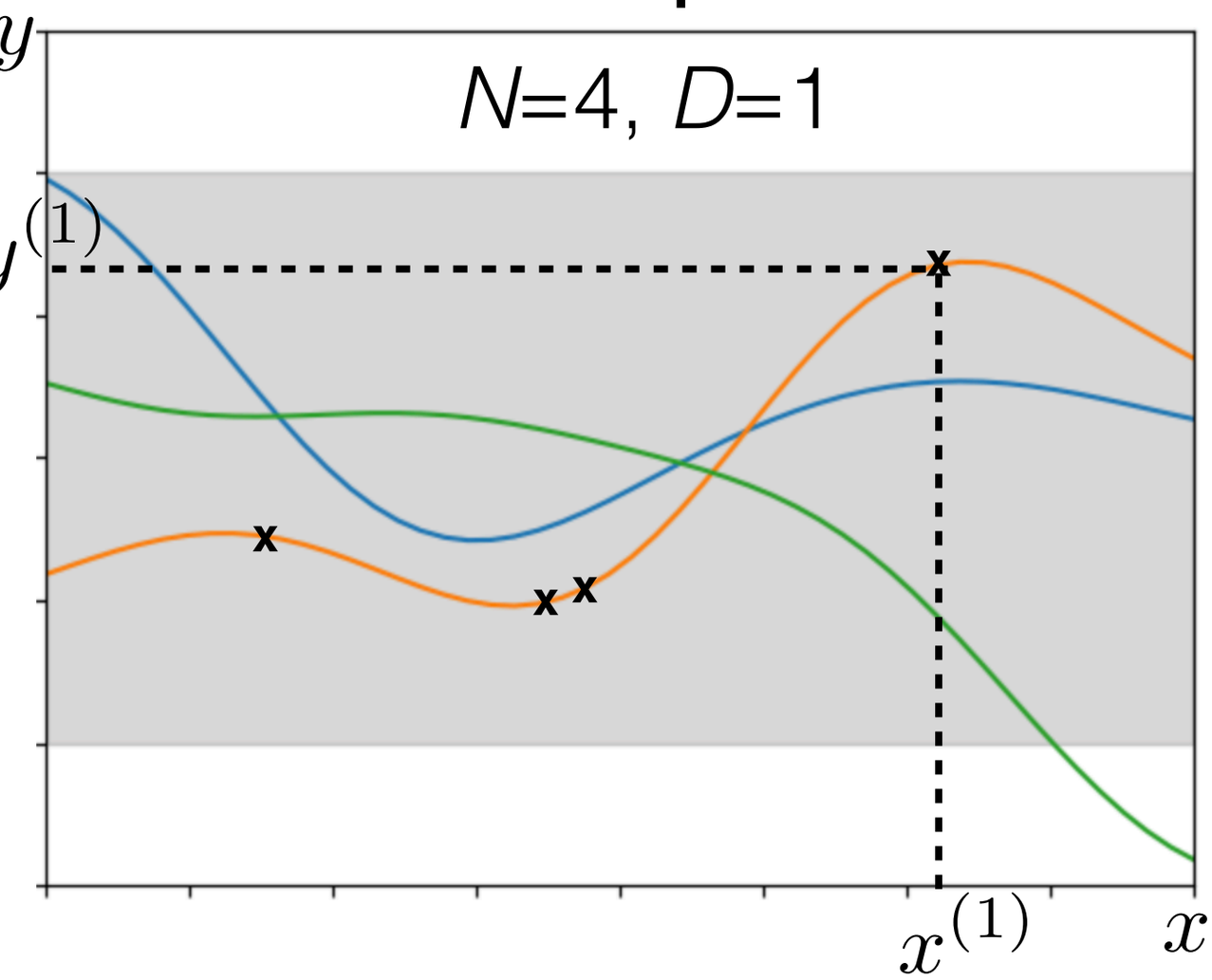
Note on "dimension" of the input

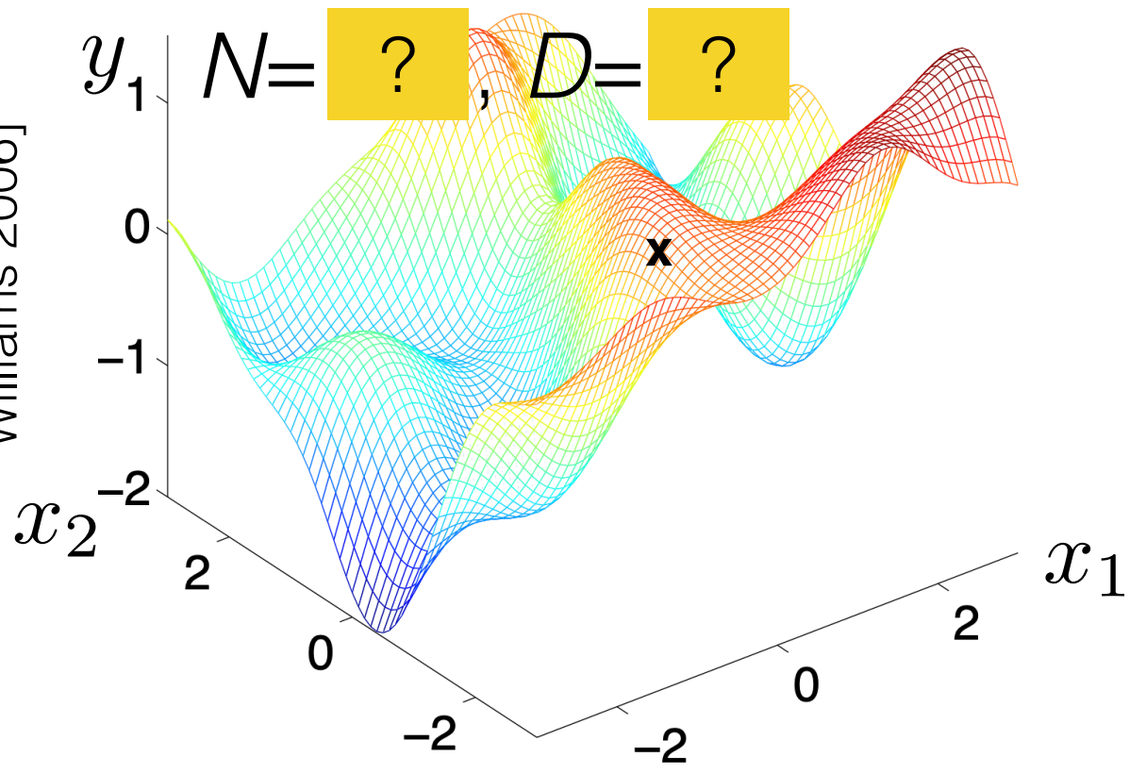
- Let's be careful to separate two types of "dimension"
  - We're using a superscript to denote (*M* or *N*) number of points in the space
  - We'll use a subscript for the (*D*) different elements of a point's vector



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

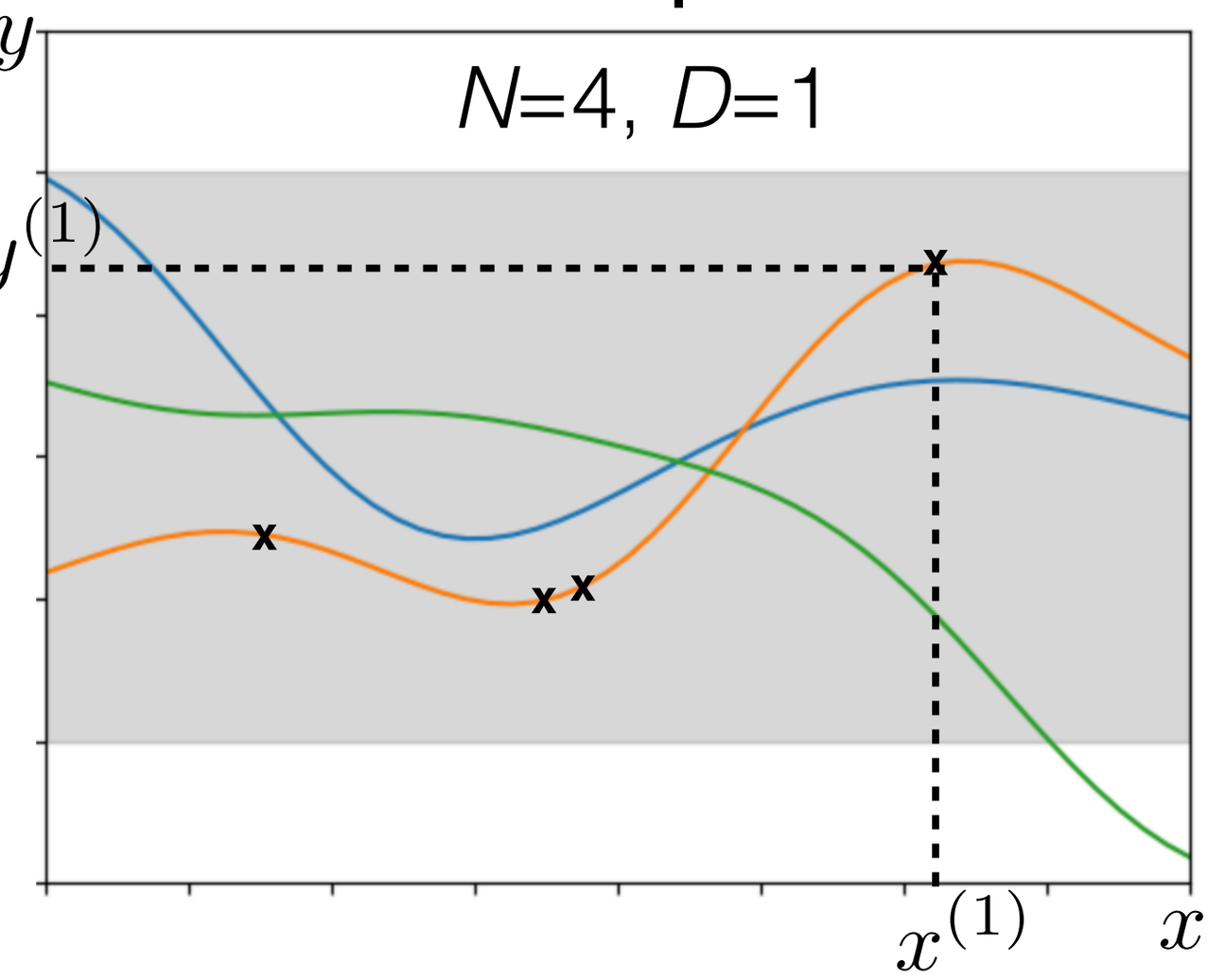


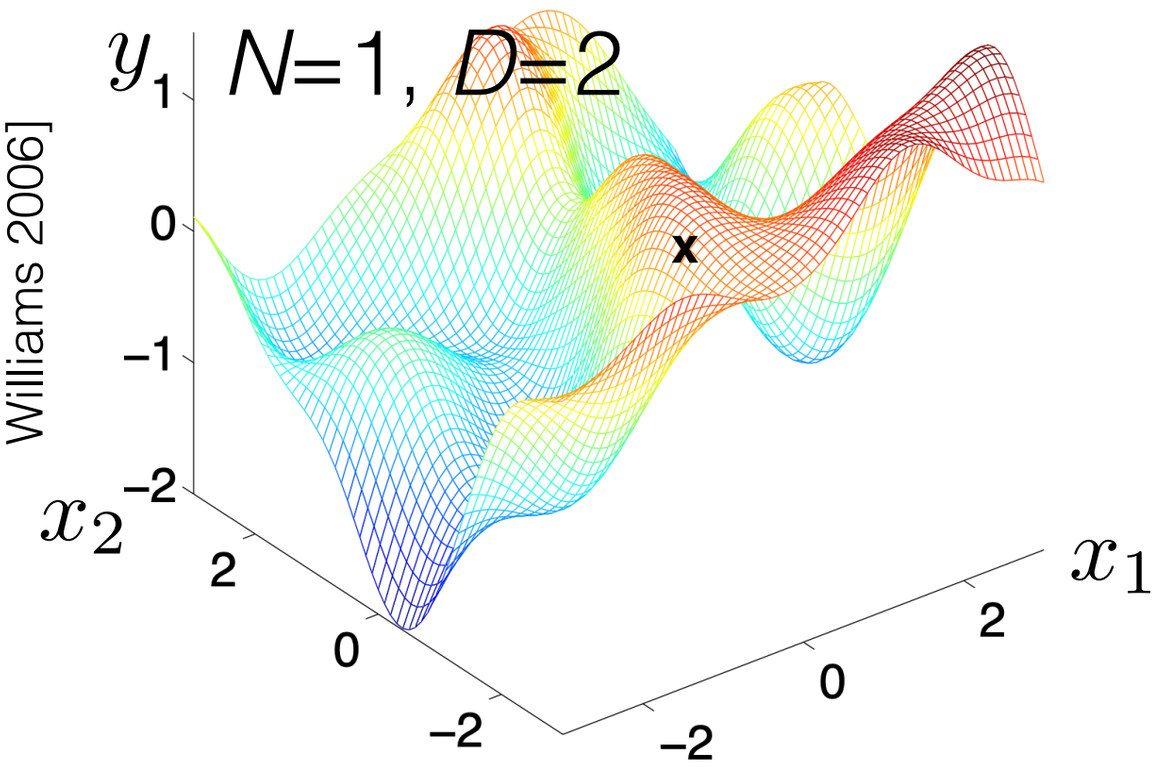


[Rasmussen & Williams 2006]

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

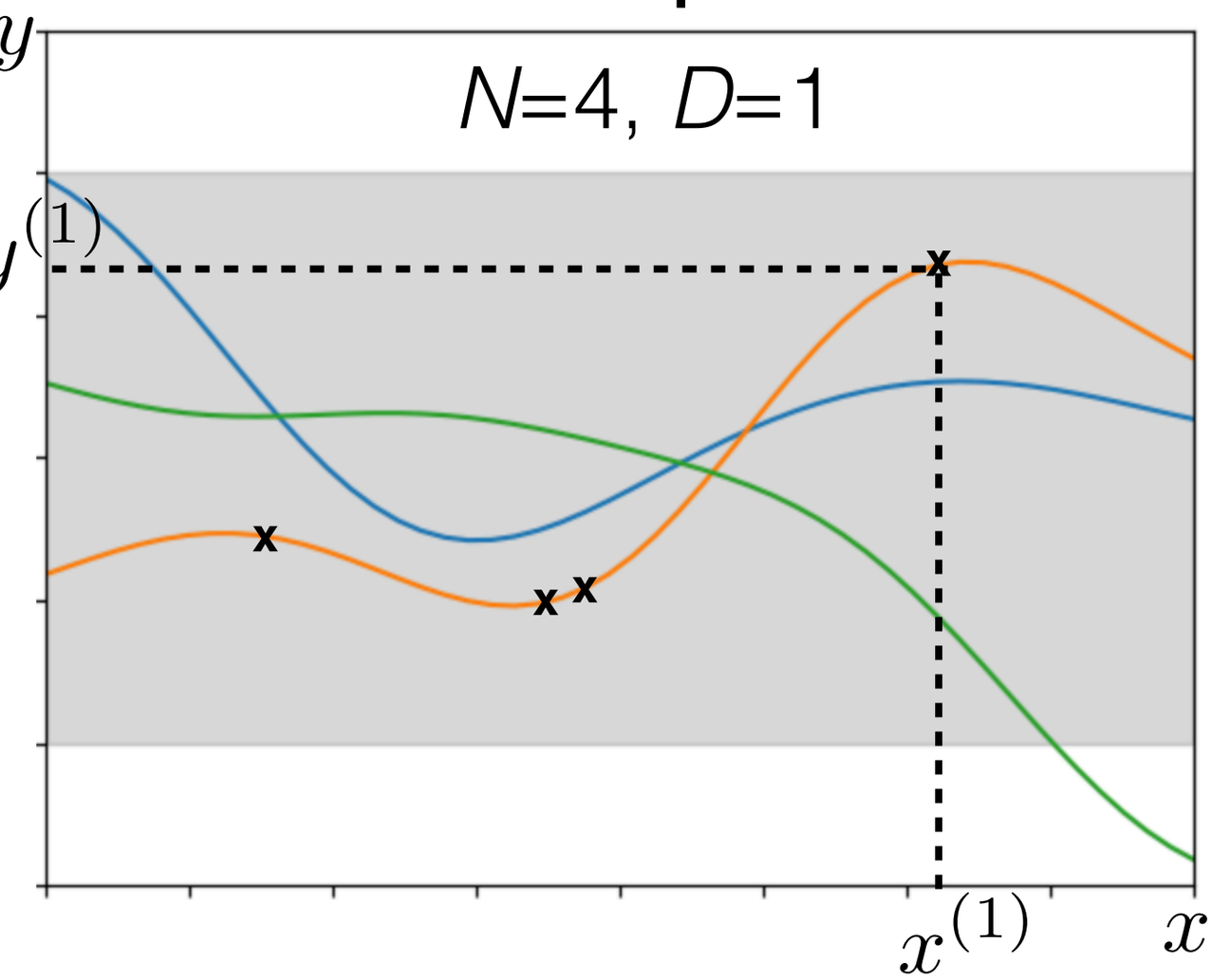


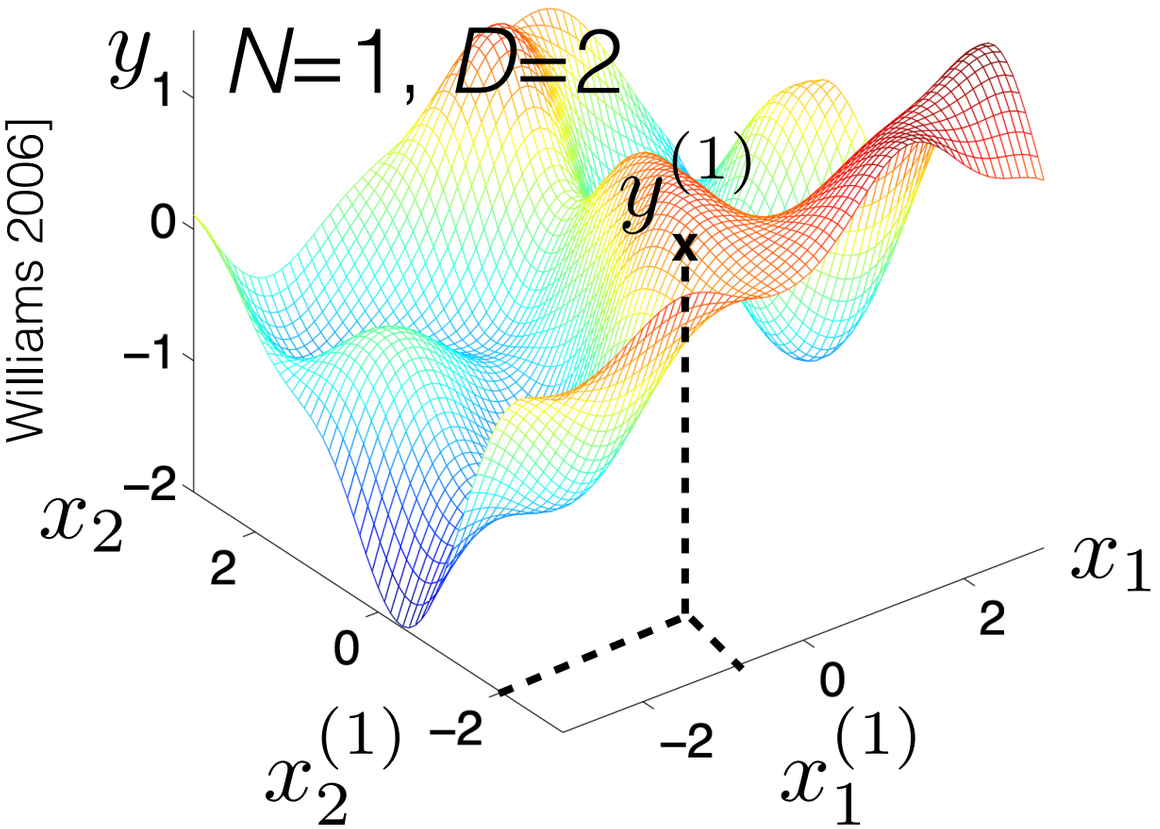


[Rasmussen & Williams 2006]

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

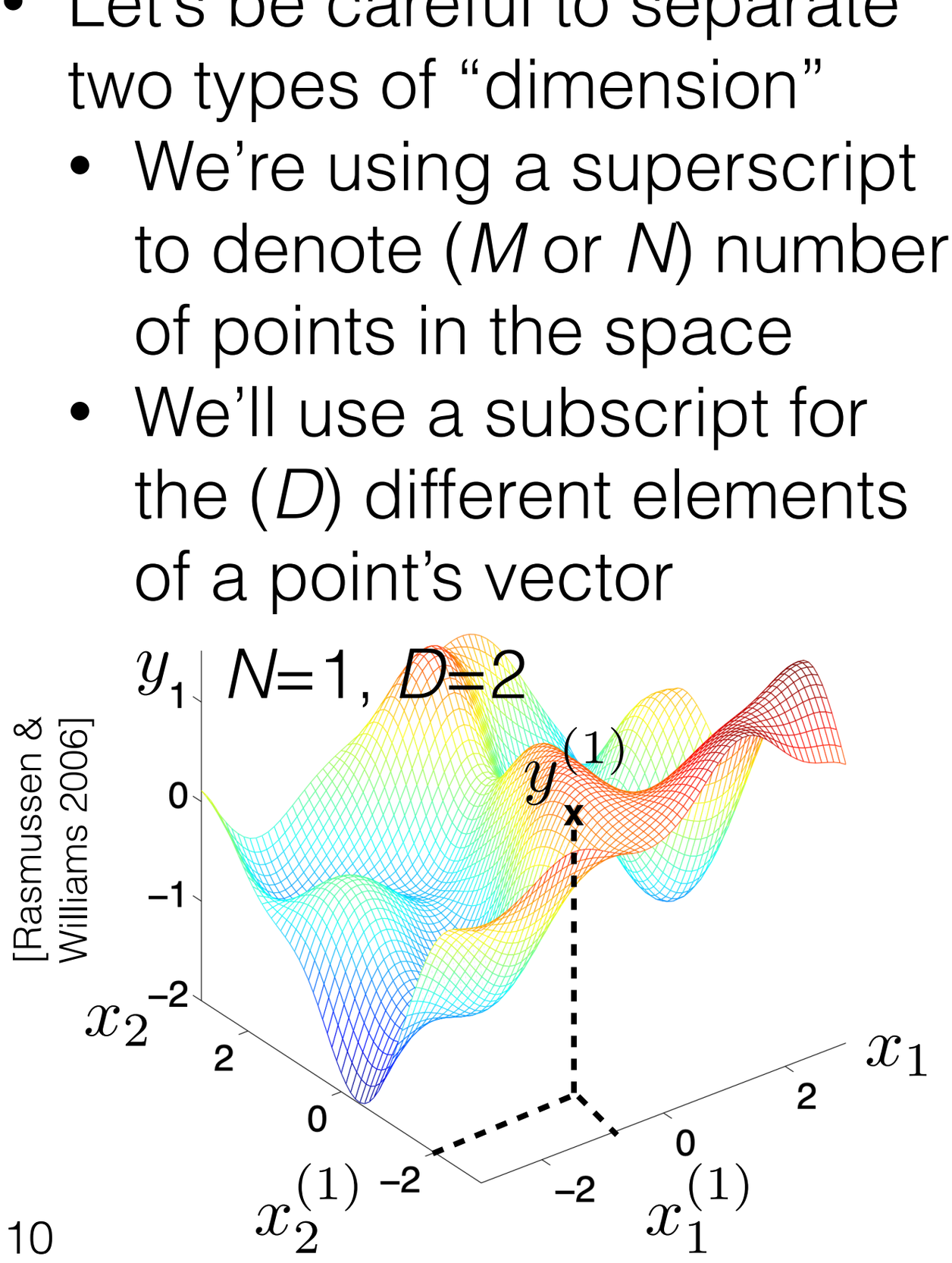




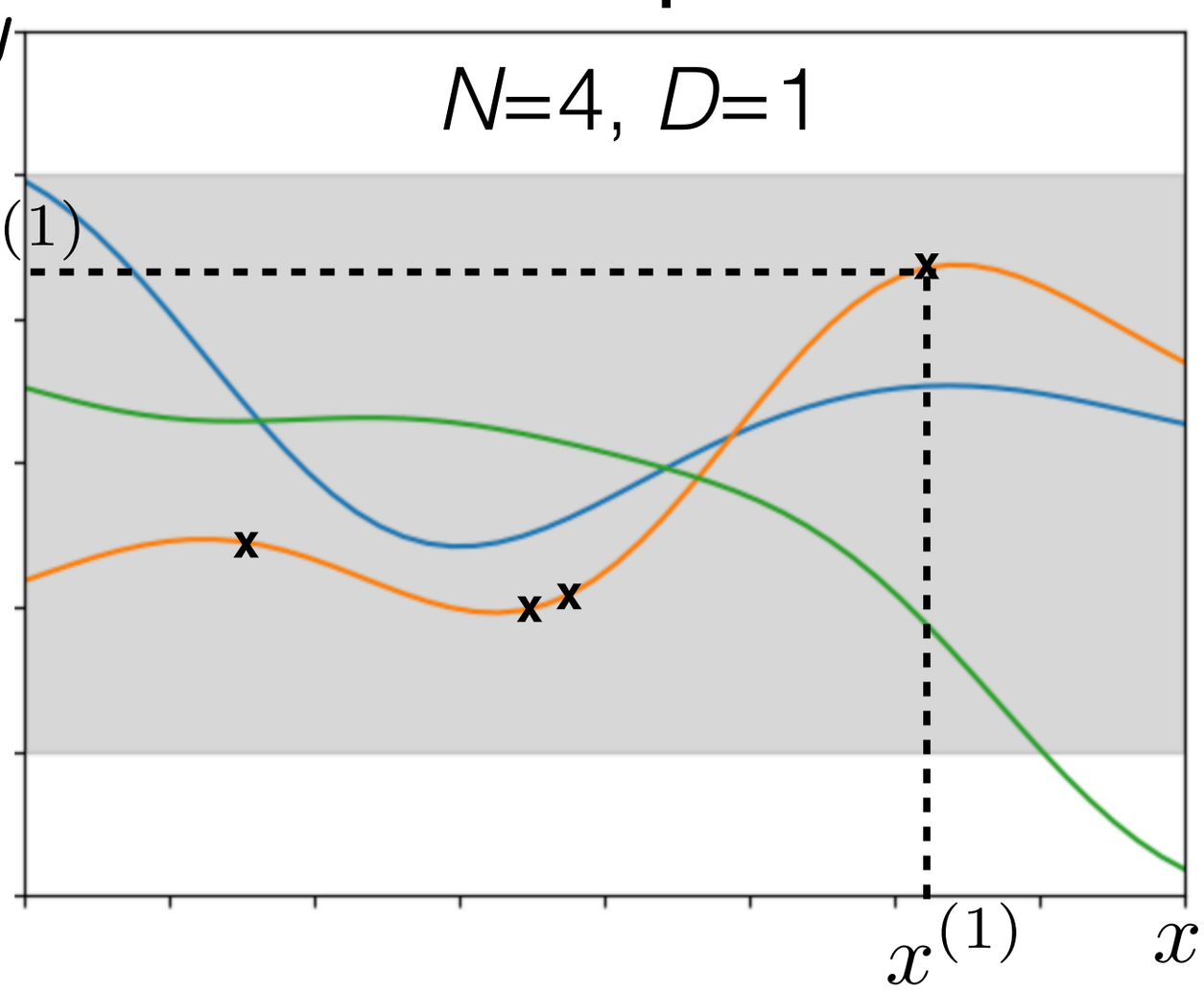
[Rasmussen & Williams 2006]

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



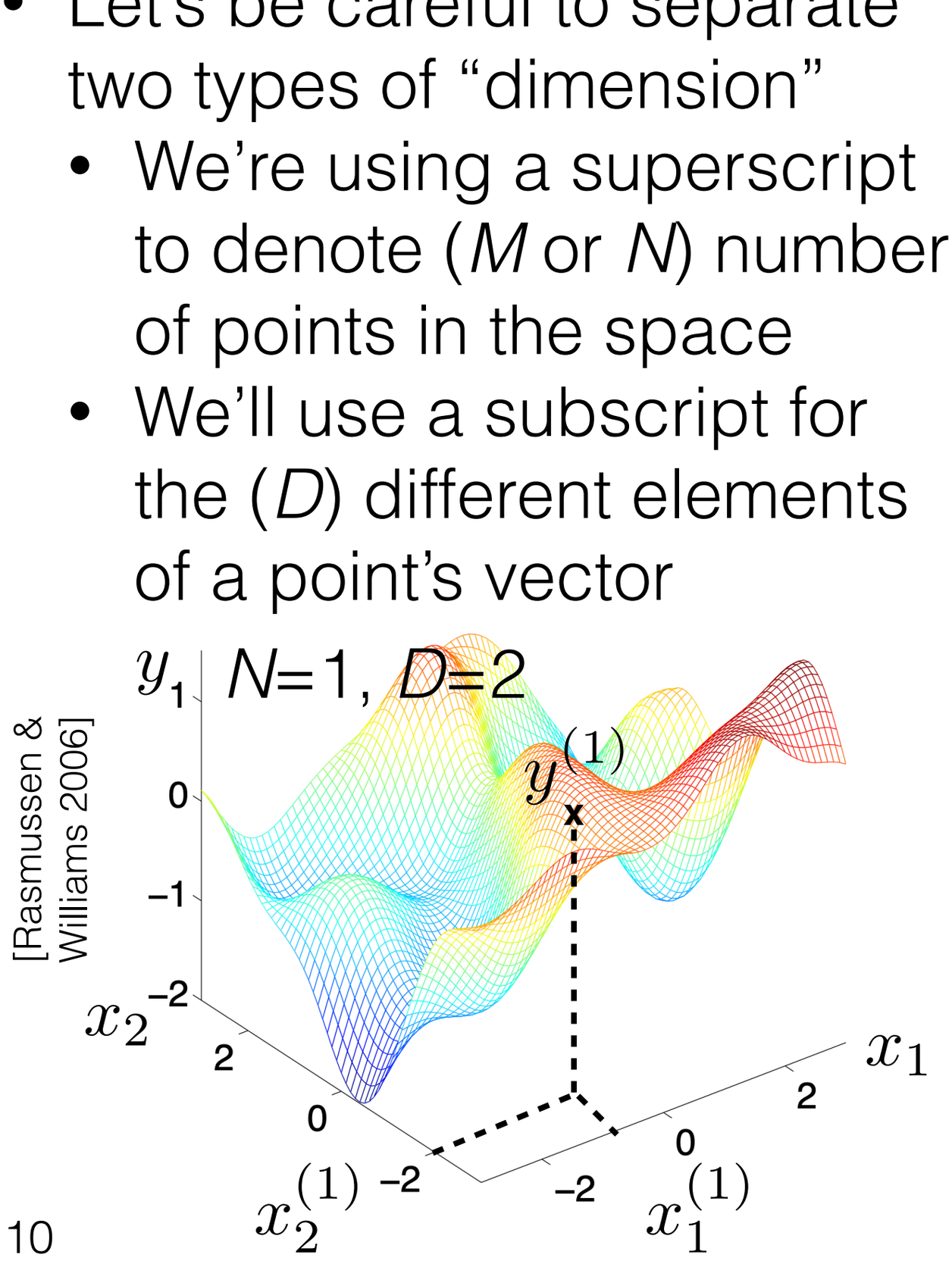
[Rasmussen & Williams 2006]



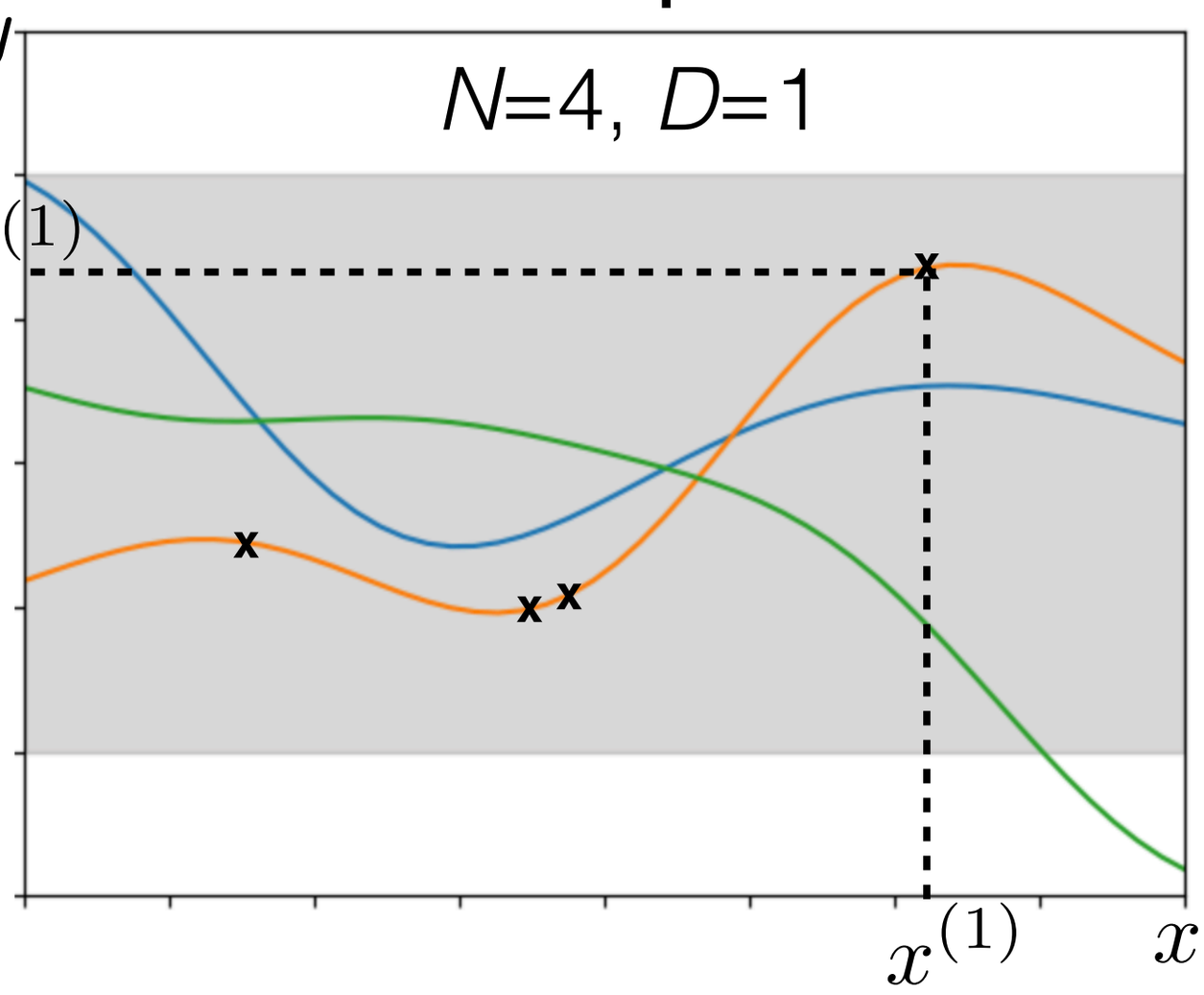
- Note: all of our real-life examples from the start had number of inputs $D > 1$

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

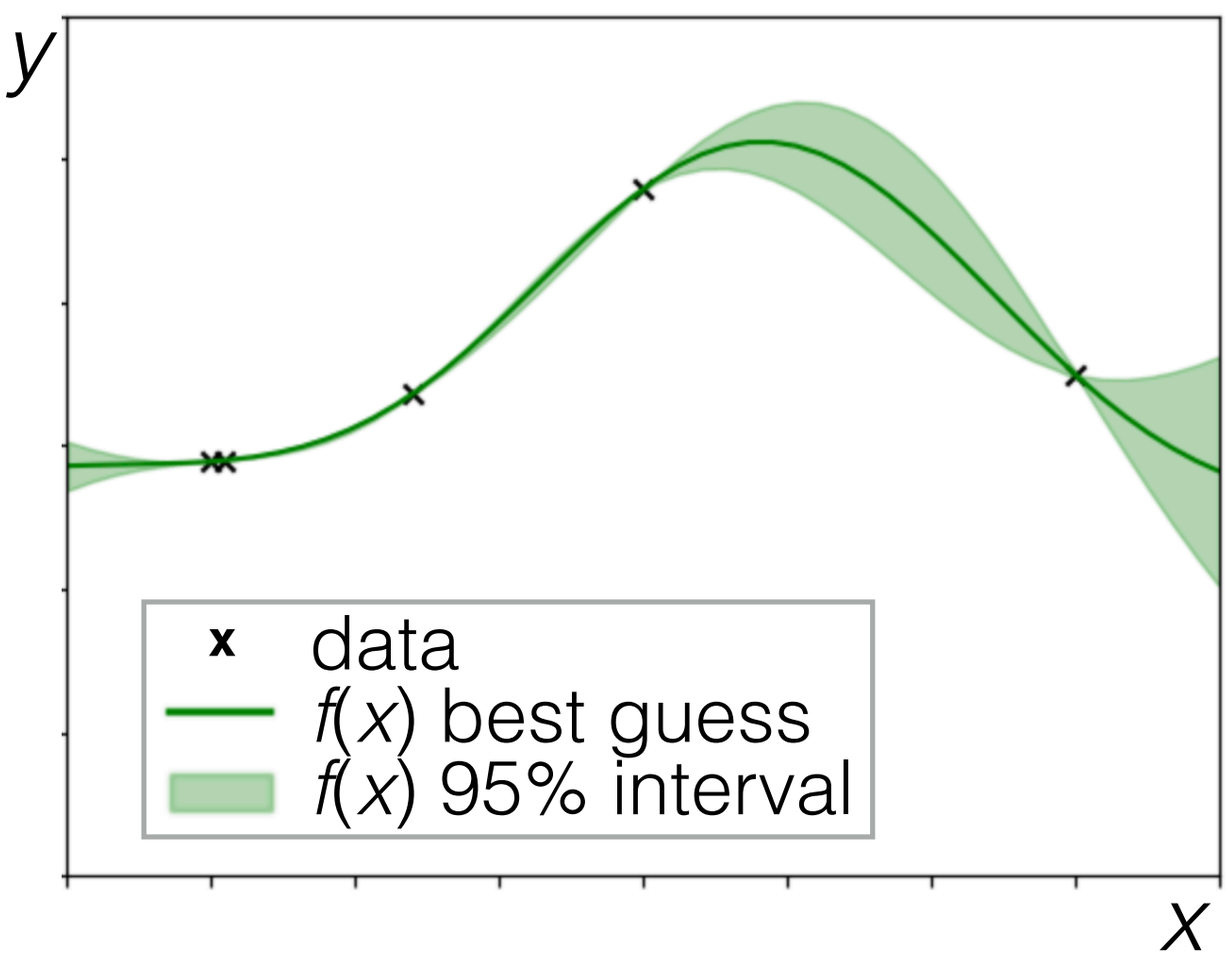
- p(unknowns | data)

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

A Bayesian approach

- p(unknowns | data)

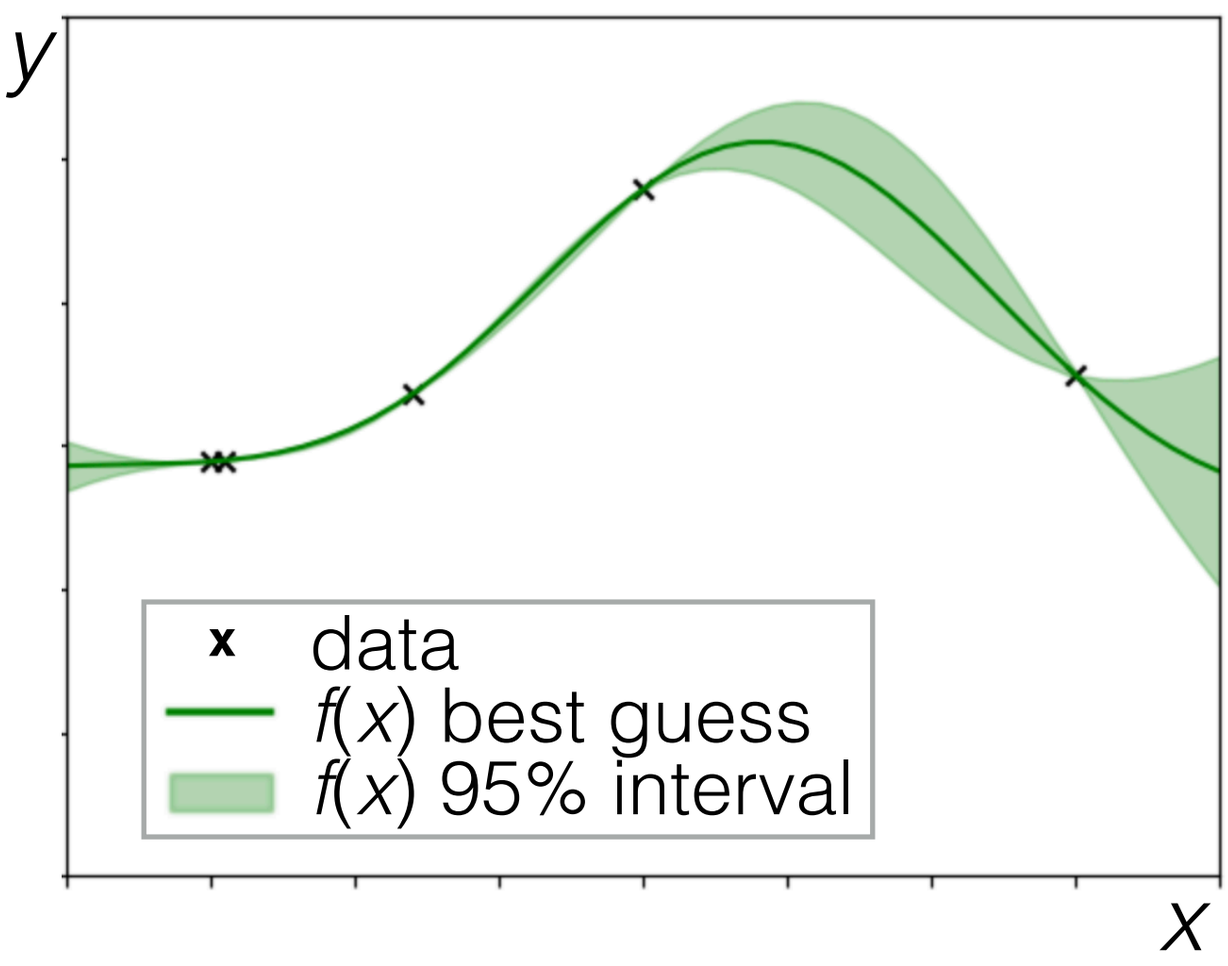
Given the data we've seen, what do we know about the underlying function?



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?

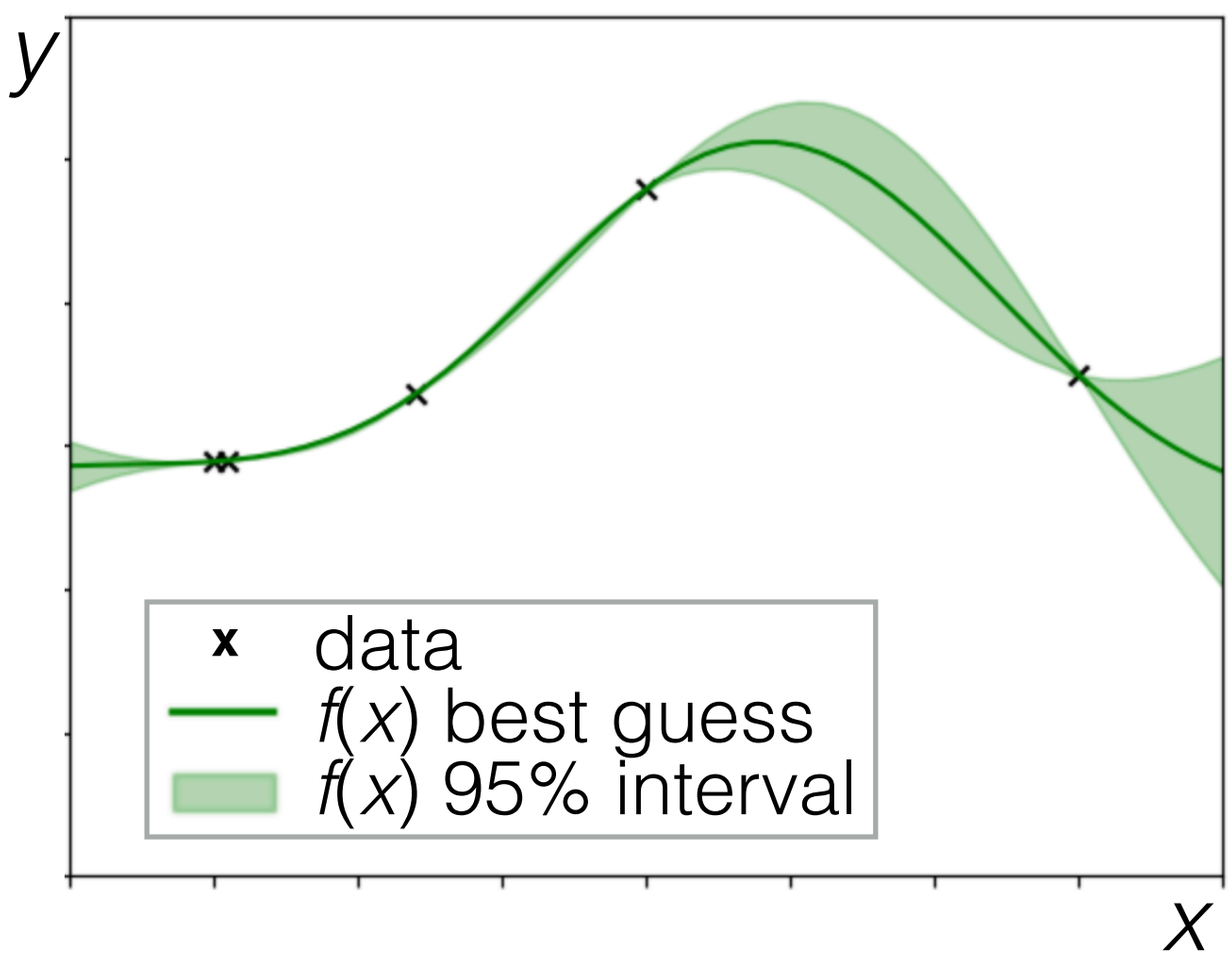


A Bayesian approach

- 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

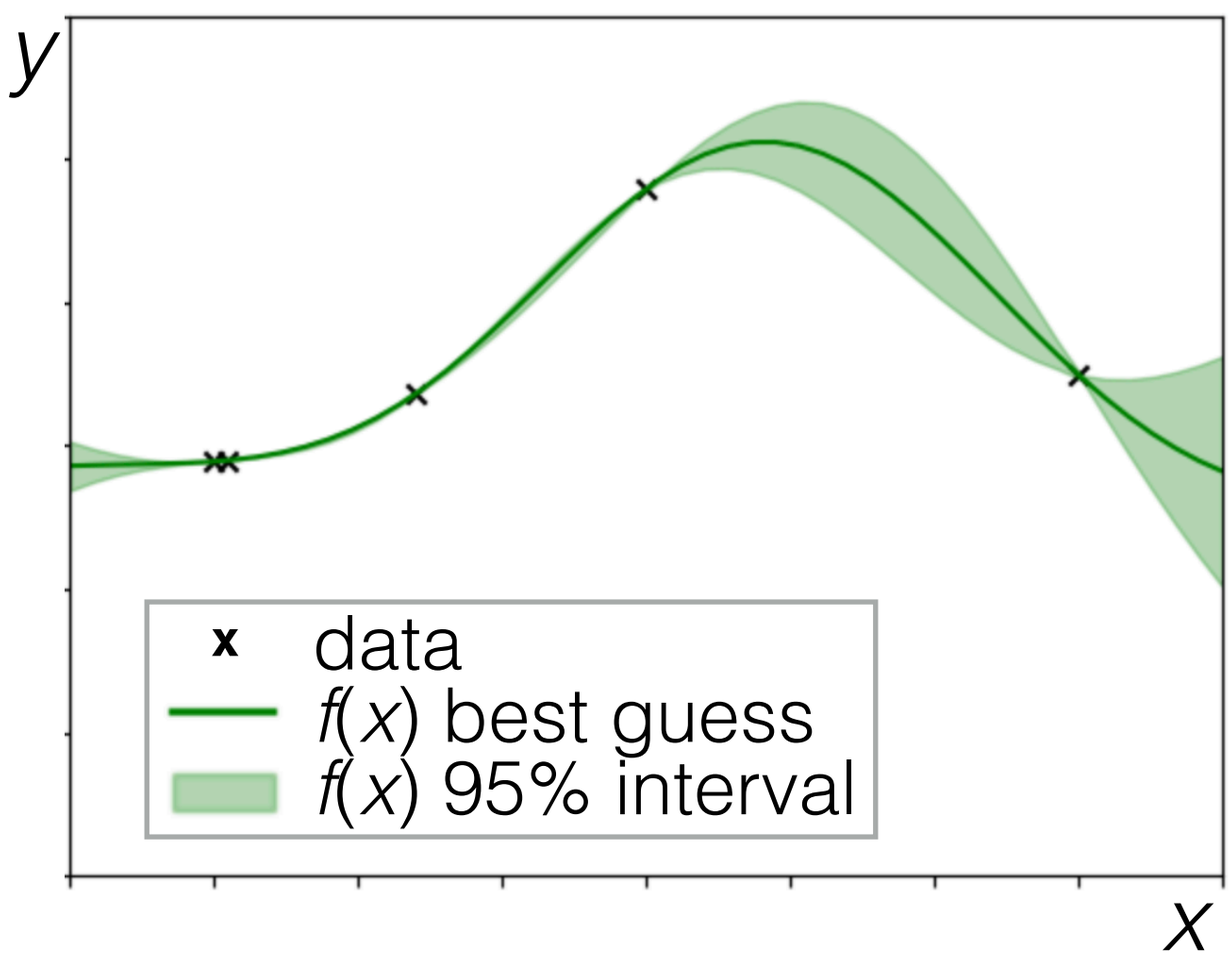


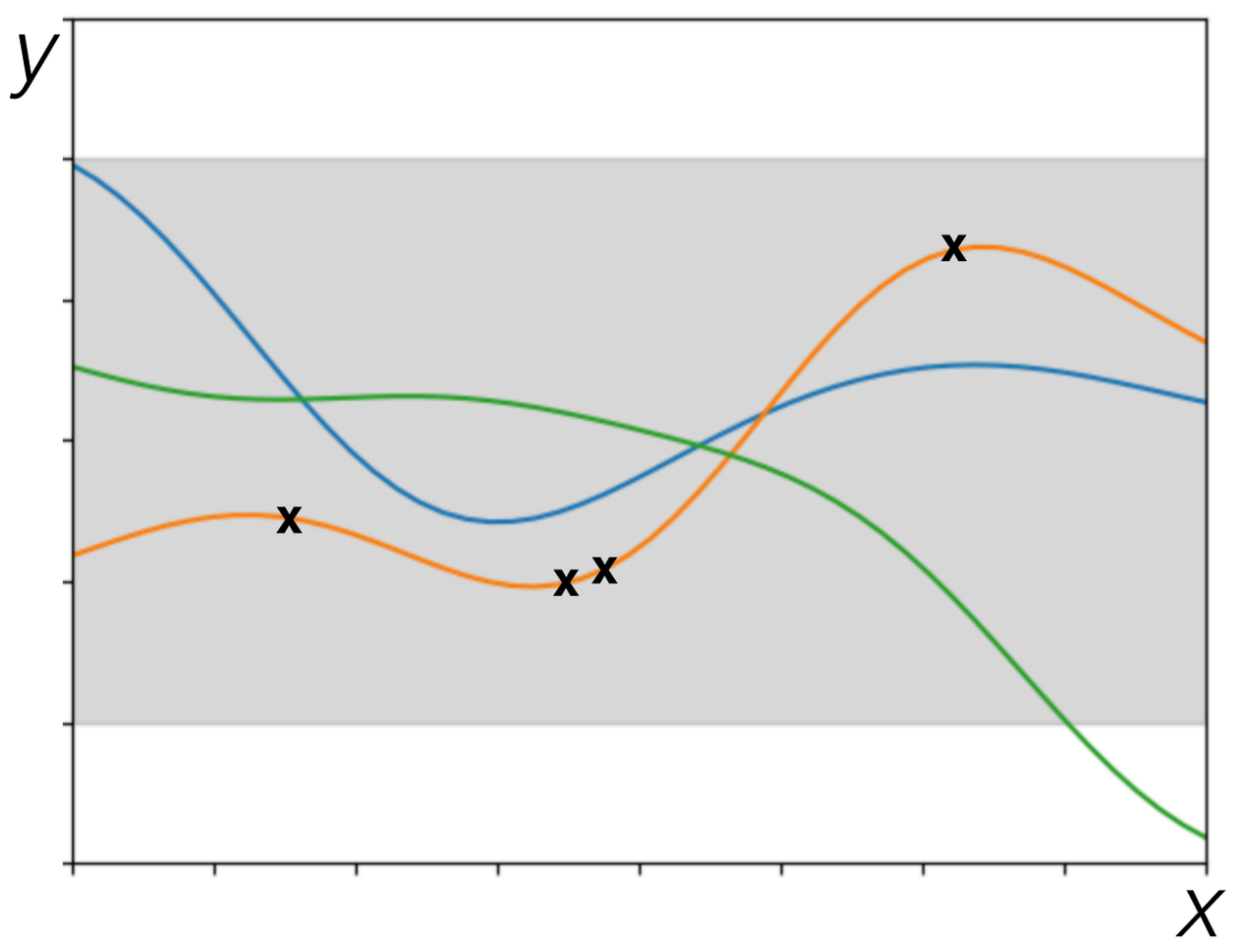
A Bayesian approach

- 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

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

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

$$\begin{bmatrix} f(X) \\ f(X') \end{bmatrix} \sim \mathcal{N}\left(\text{?} \right.$$

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*

- 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*

- 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

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

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 ?

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*

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': ?

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*

A good habit to get into: check the dimensions

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

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

X: *N*x*D*

X': *M*x*D*

A good habit to get into: check the dimensions

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

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*

A good habit to get into: check the dimensions

*N*x1

?

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

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*

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

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*

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

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*

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

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*

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

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*

A good habit to get into: check the dimensions

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

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*

A good habit to get into: check the dimensions

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

- The conditional satisfies $f(X')|f(X), X, X' \sim$?

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: *N*x*D*

X': *M*x*D*

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

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

A good habit to get into: check the dimensions

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

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: *N*x*D*

X': *M*x*D*

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

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

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$
  - Can compute mean & covariance in closed form

via Gaussian facts

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: *N*x*D*

X': *M*x*D*

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

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

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$
  - Can compute mean & covariance in closed form

?

via Gaussian facts

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: *N*x*D*

X': *M*x*D*

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

*N*x1 / *M*x1

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

(*N*+*M*)x1 (*N*+*M*)x(*N*+*M*)

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$
  - Can compute mean & covariance in closed form

*M*x1

via Gaussian facts

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: *N*x*D*

X': *M*x*D*

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

A good habit to get into: check the dimensions

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

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$
  - Can compute mean & covariance in closed form

via Gaussian facts

Mx1 ?

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: *N*x*D*

X': *M*x*D*

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

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

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$
  - Can compute mean & covariance in closed form

Mx1 *MxM*

via Gaussian facts

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

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

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

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$
  - Can compute mean & covariance in closed form

*M*x1 *M*x*M*

via Gaussian facts

- 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: *N*x*D*

X': *M*x*D*

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

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

A good habit to get into: check the dimensions

- The conditional satisfies $f(X')|f(X), X, X' \sim \mathcal{N}$
  - Can compute mean & covariance in closed form

*M*x1 *M*x*M*

via Gaussian facts

- 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]

Squared exponential kernel revisited

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo1]

Squared exponential kernel revisited

- What if we happened to measure our data on a different scale? [demo1]
- 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? [demo1]
- 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? [demo1]
- 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? [demo1]
- 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? [demo1]
- 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*?

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

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*?

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

- 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*?

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

- 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

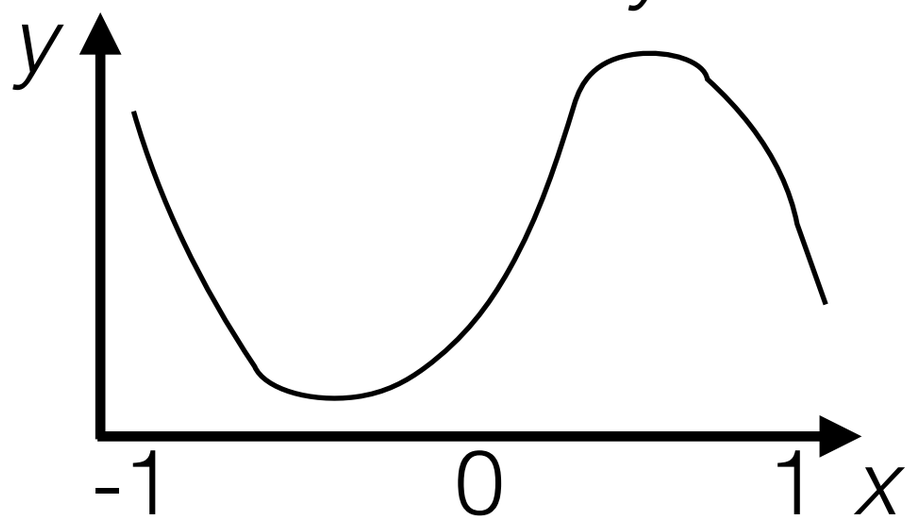
- What if we happened to measure our data on a different scale? [demo1, demo2]
- We've been using this particular kernel:

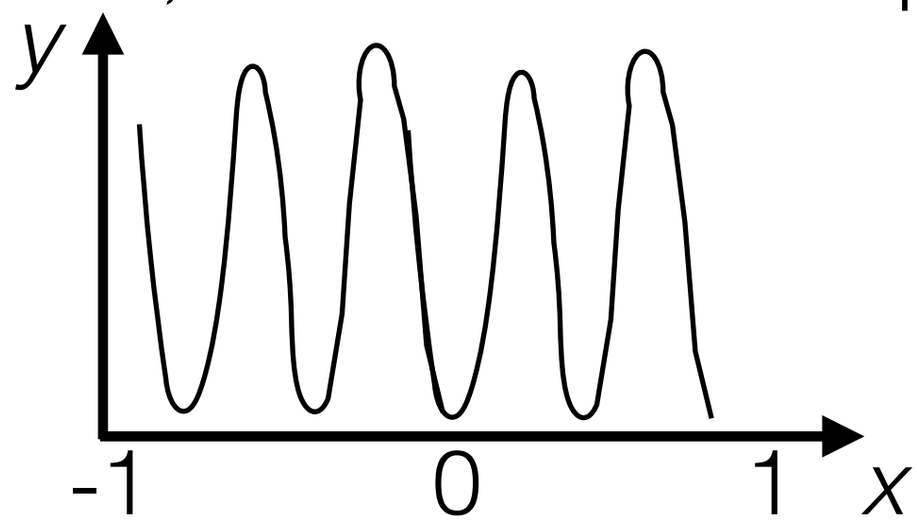
$$k(x, x') = \sigma^2 \exp(-\tfrac{1}{2}(x - x')^2), \sigma = 1$$

  - 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*?

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

- 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:

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

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:

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

signal variance — lengthscales

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:

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

signal variance — lengthscales

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

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:

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

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

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:

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

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.

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:

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

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:

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

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:
  1. Fit values 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:

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

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:
  1. Fit values for the hyperparameters using the data.
  2. 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:

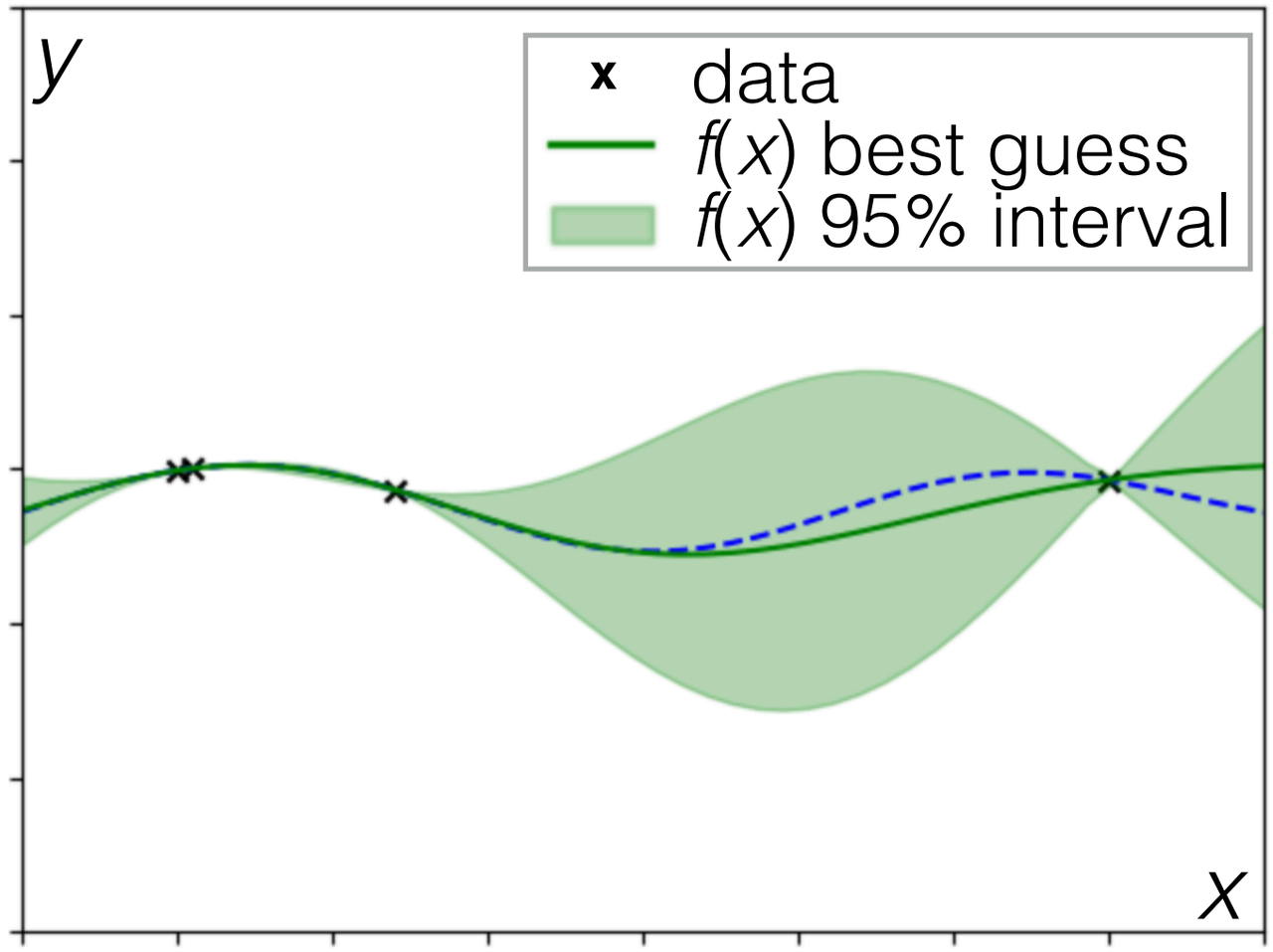
$$k(x, x') = \sigma^2 \exp(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{\ell_d^2})$$

signal variance — lengthscales

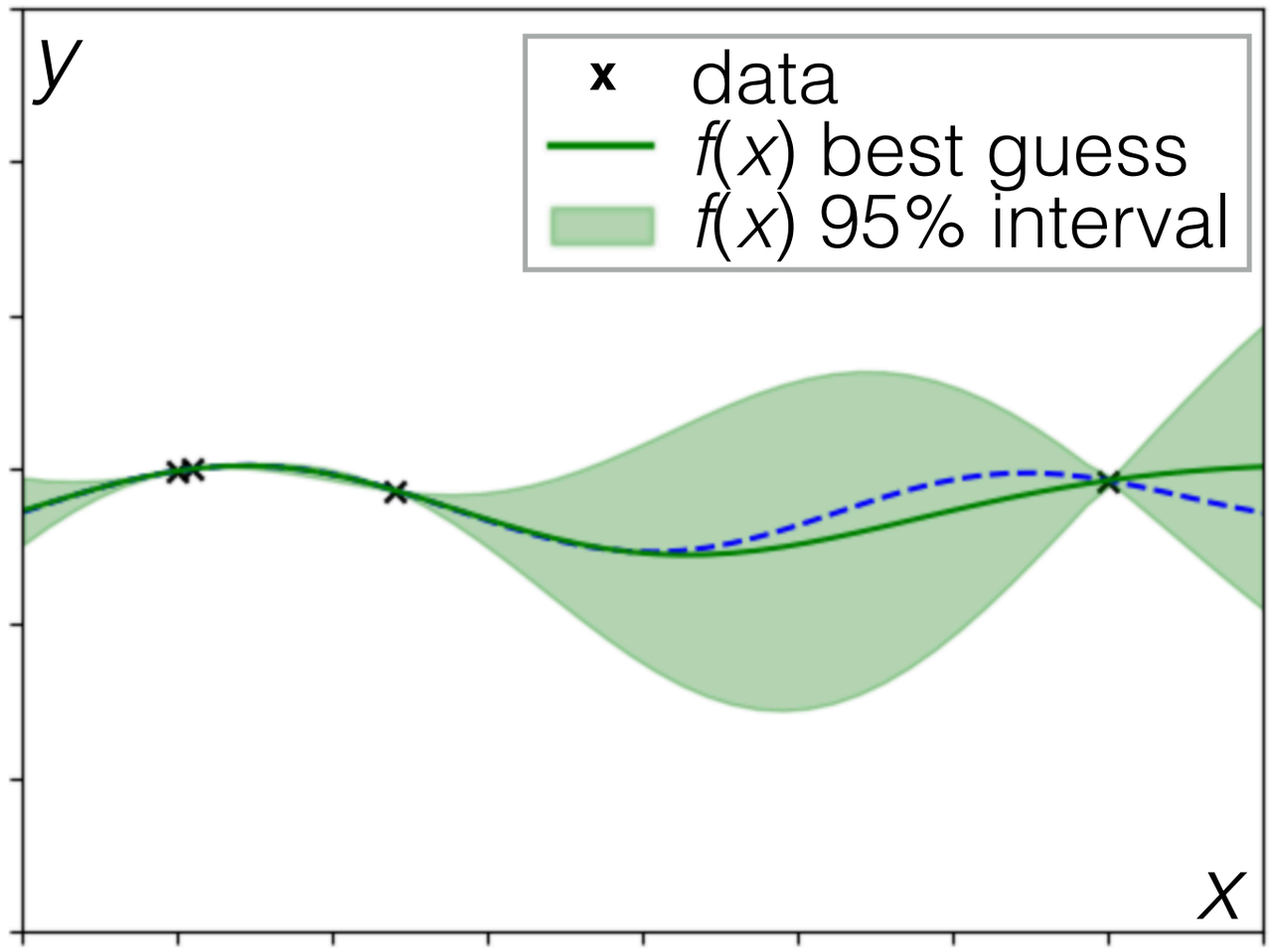
  - *Parameters* (here, *f*) parametrize the distribution of the data. If we knew them, we could generate the data.
    - GPs: *nonparametric* model: infinite # of latent params
  - *Hyperparameters* parametrize the distribution of the parameters. If known, we could generate the parameters.
- Algorithm: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

Closer look at the uncertainty interval

Closer look at the uncertainty interval

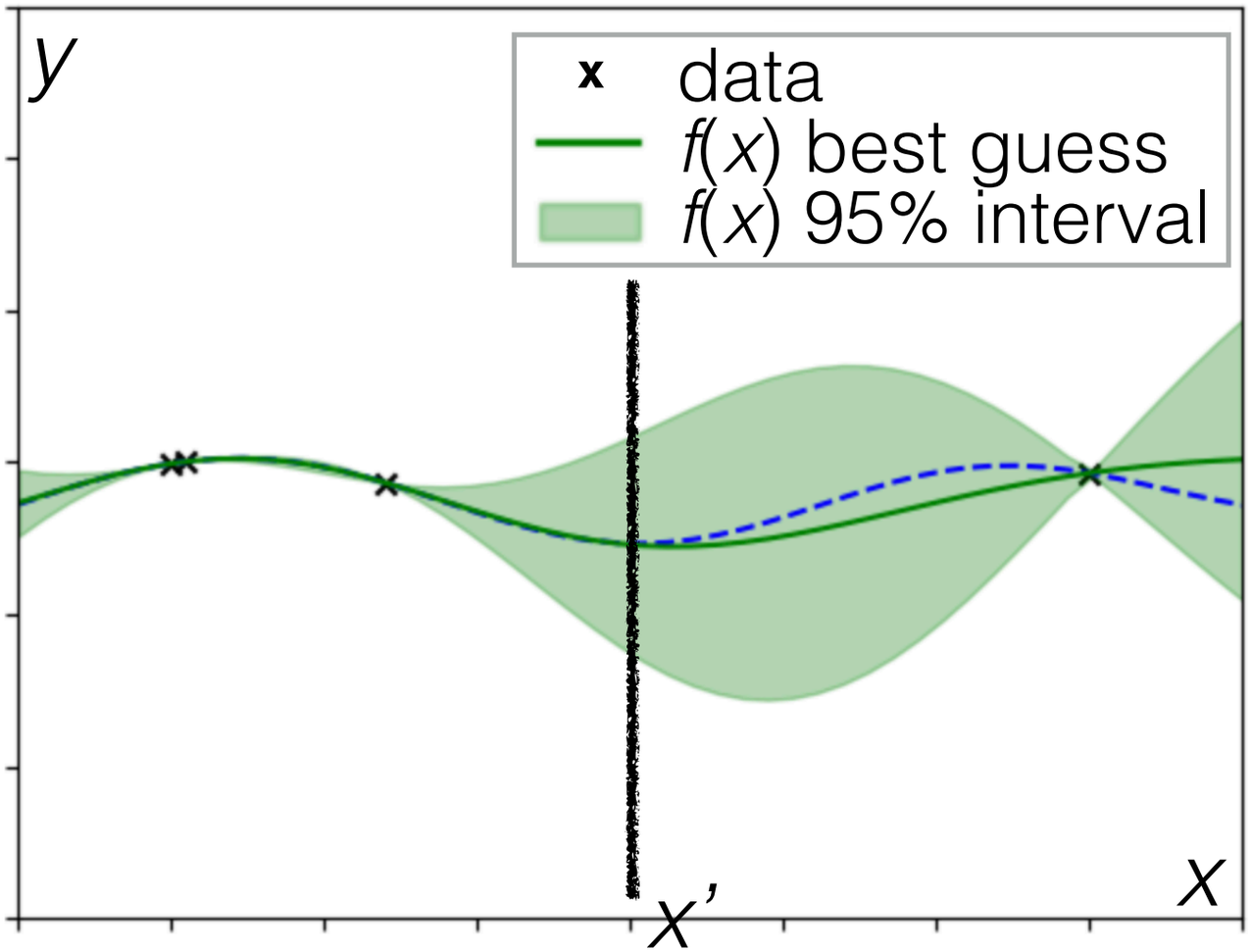


Closer look at the uncertainty interval



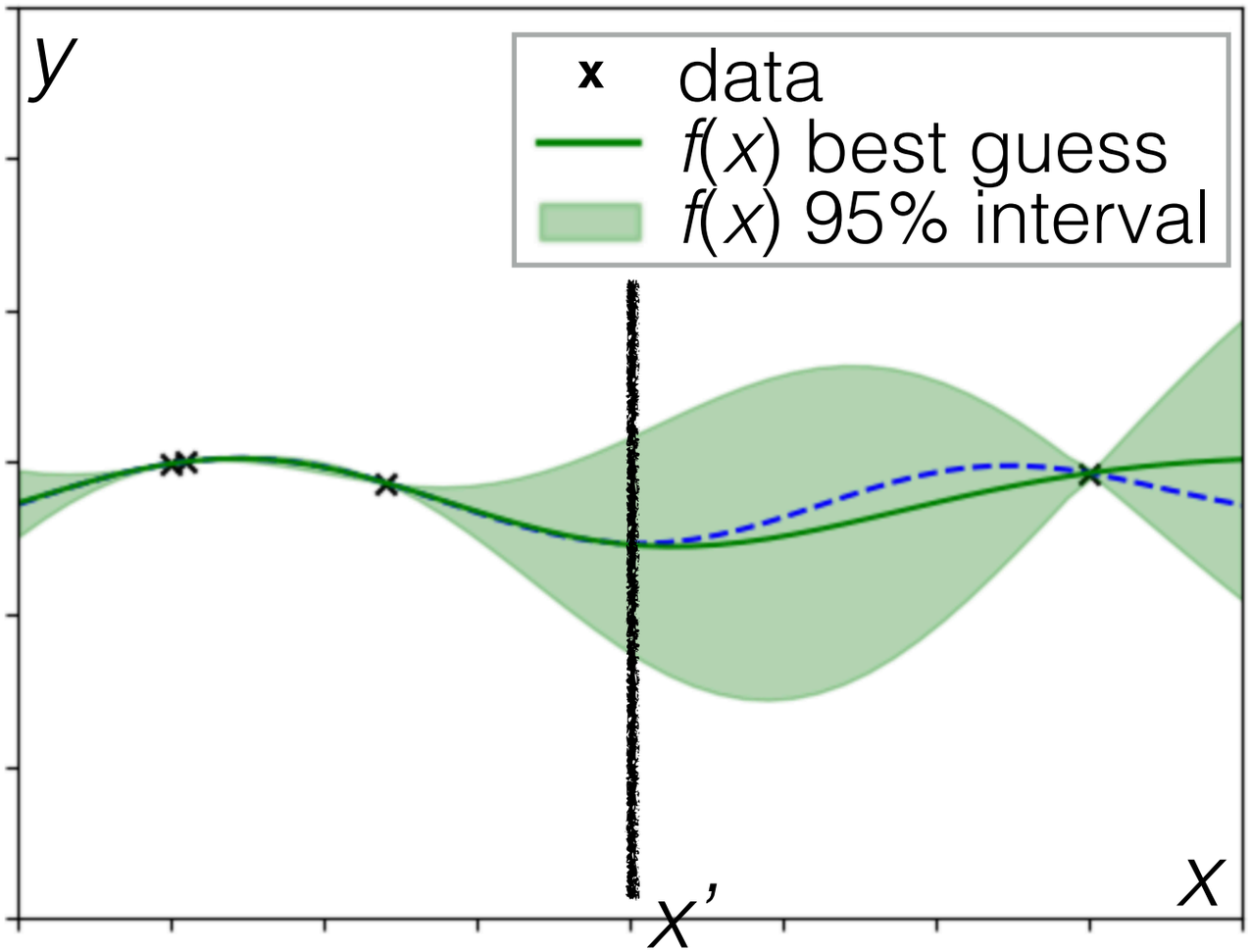
- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian

Closer look at the uncertainty interval



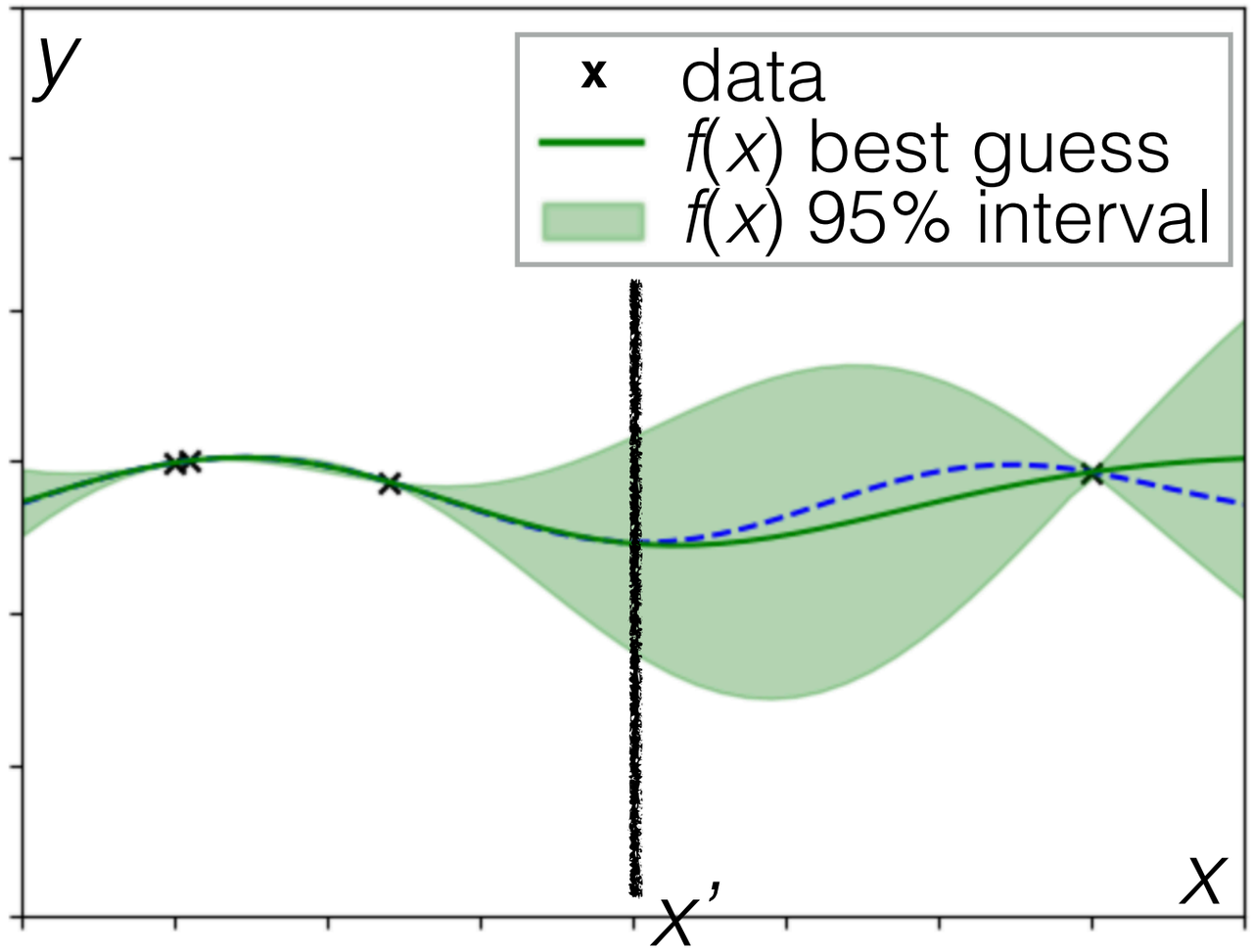
- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian

Closer look at the uncertainty interval



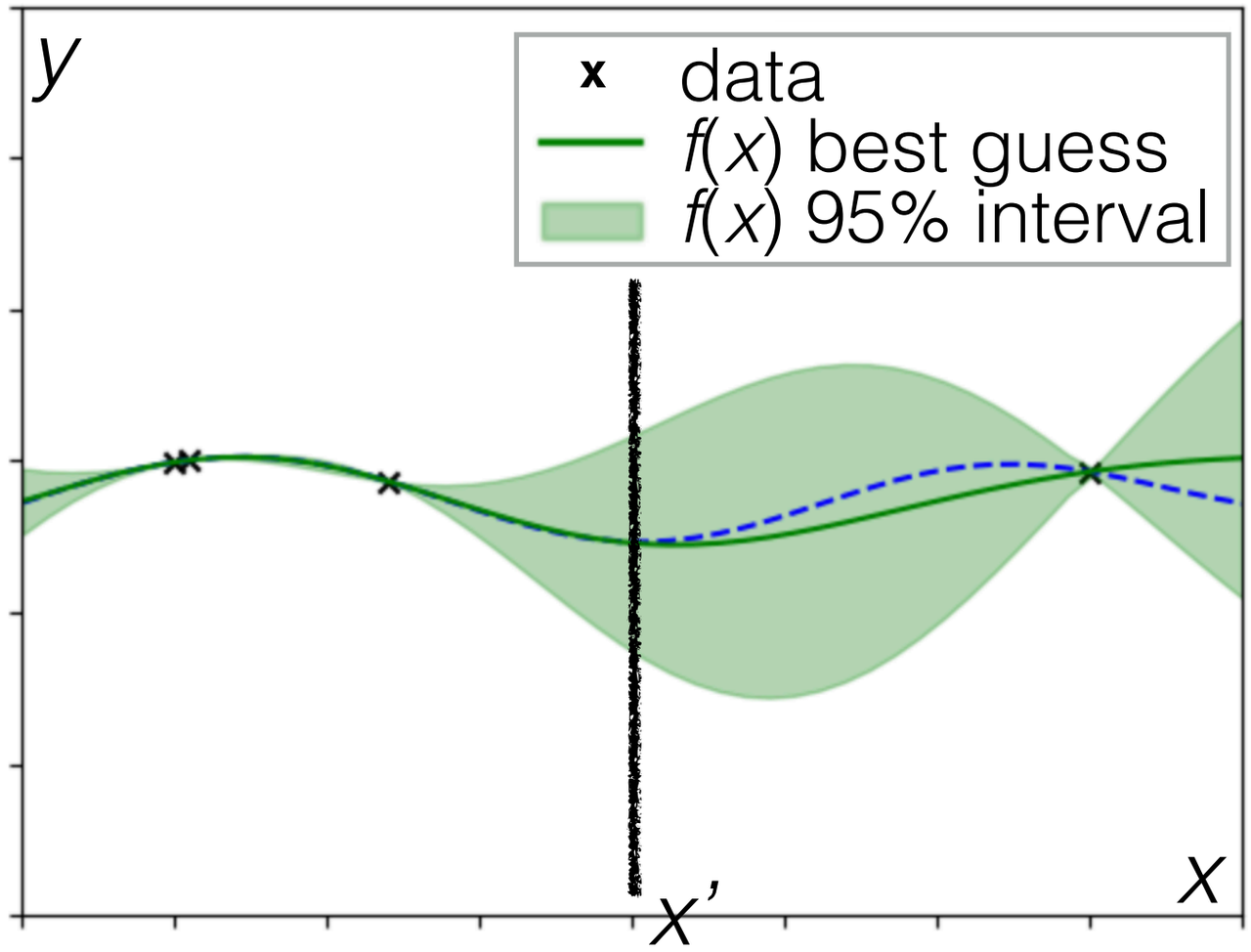
- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian

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

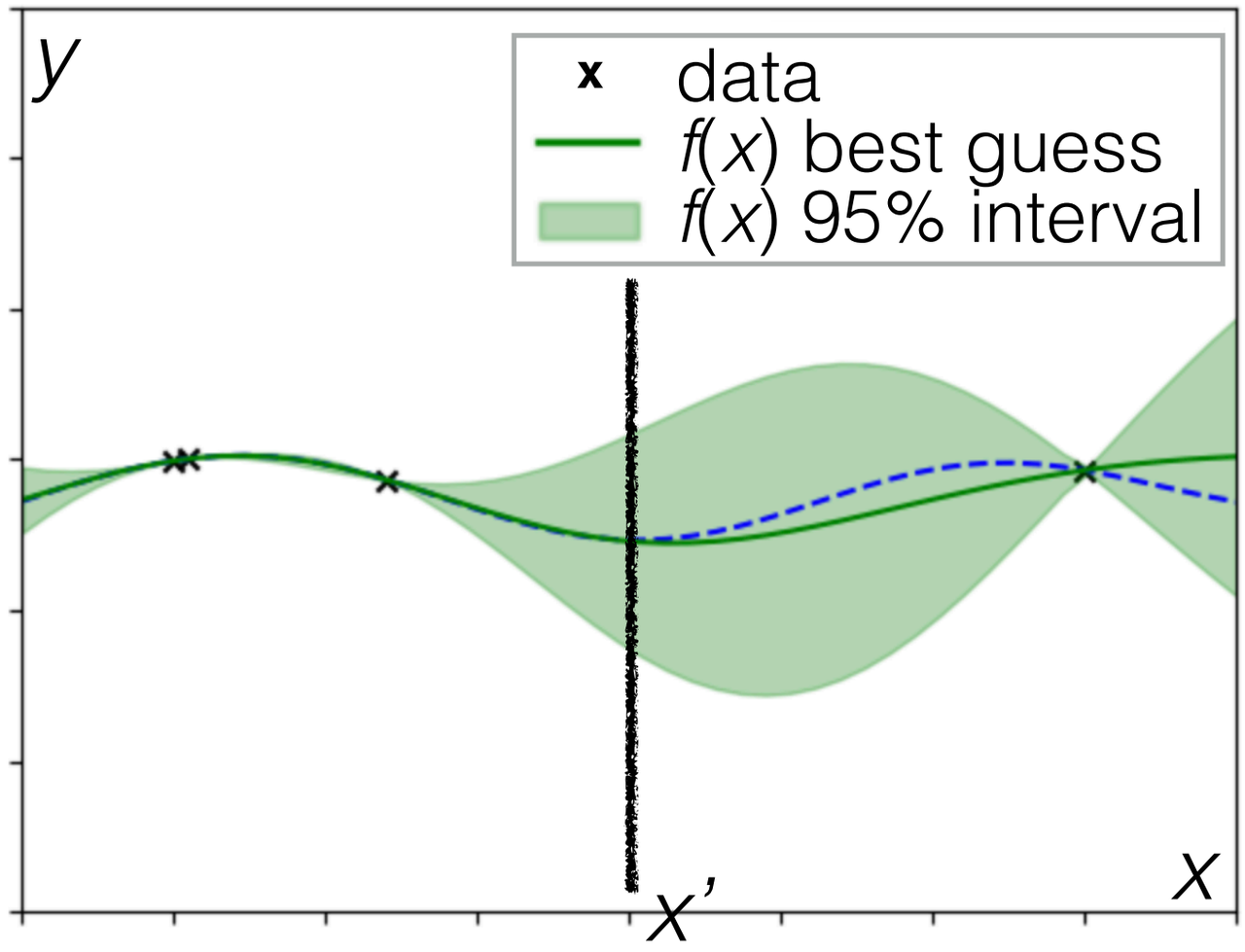
Closer look at the uncertainty interval



- 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

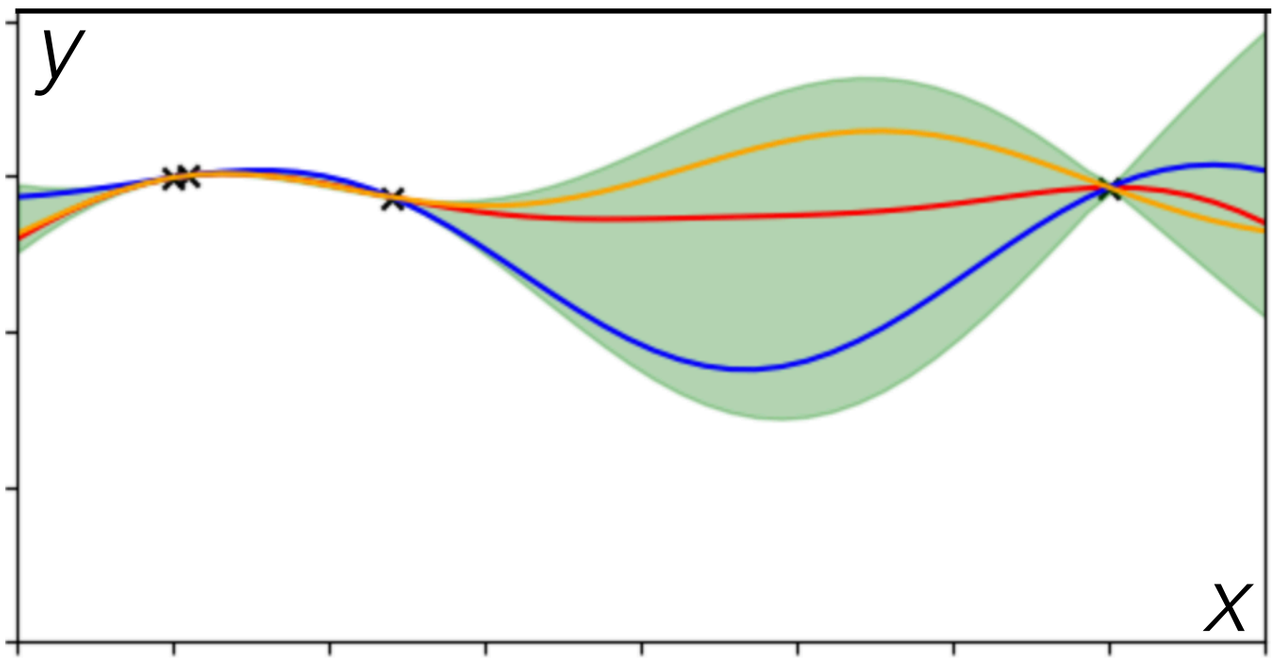
- Draw random *f* conditional on the training data

Closer look at the uncertainty interval

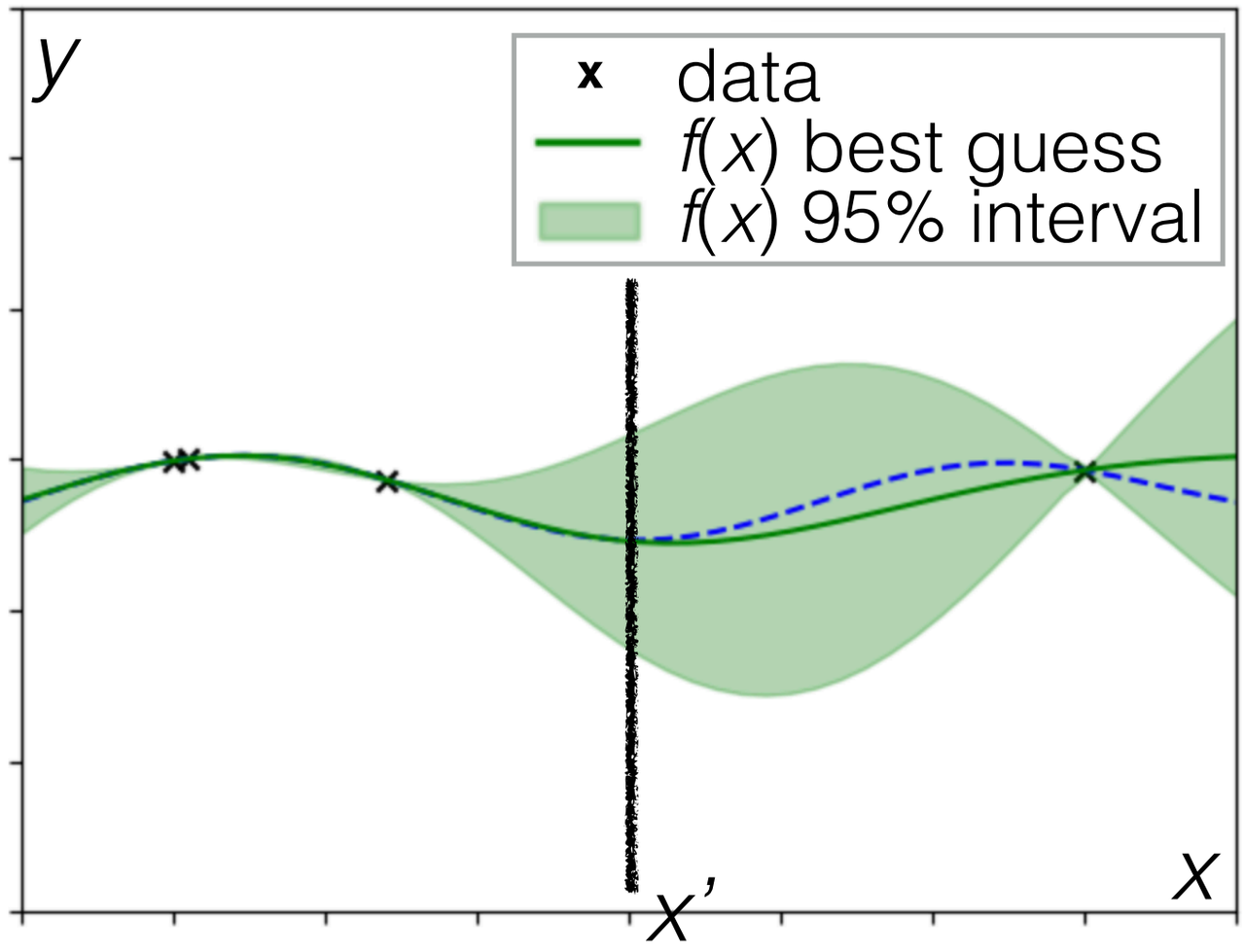


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

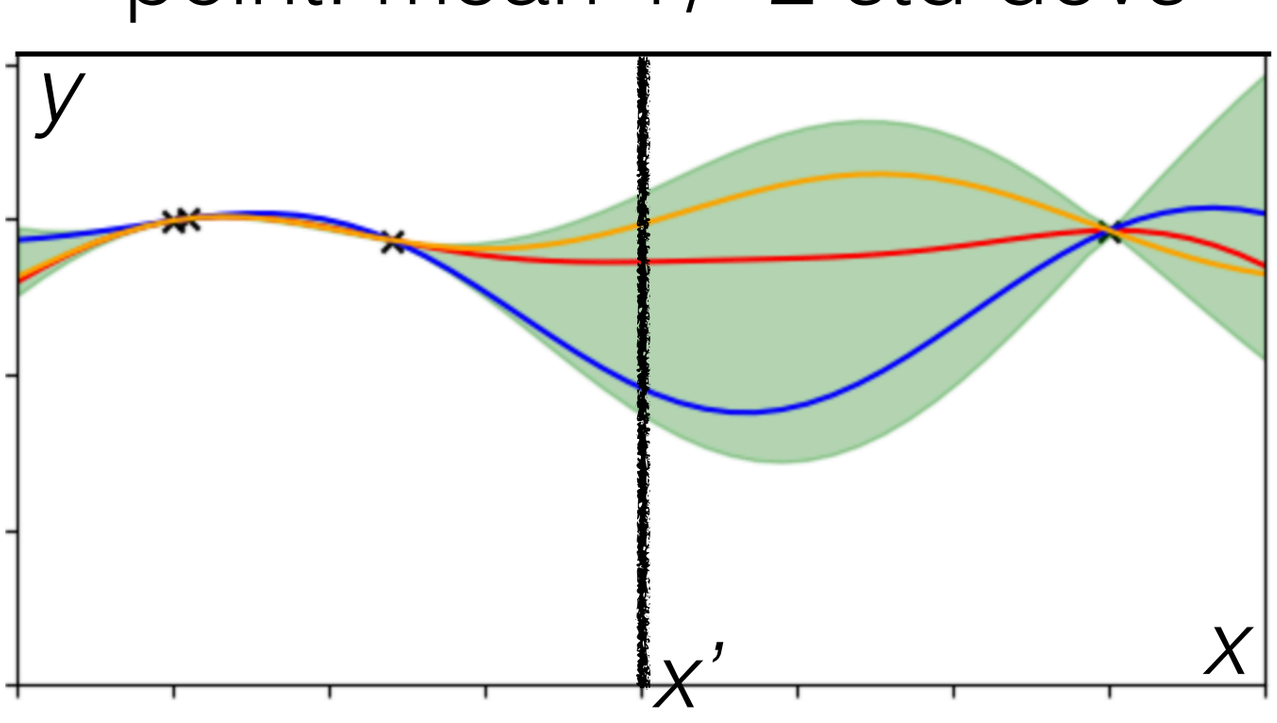
- Draw random *f* conditional on the training data



Closer look at the uncertainty interval

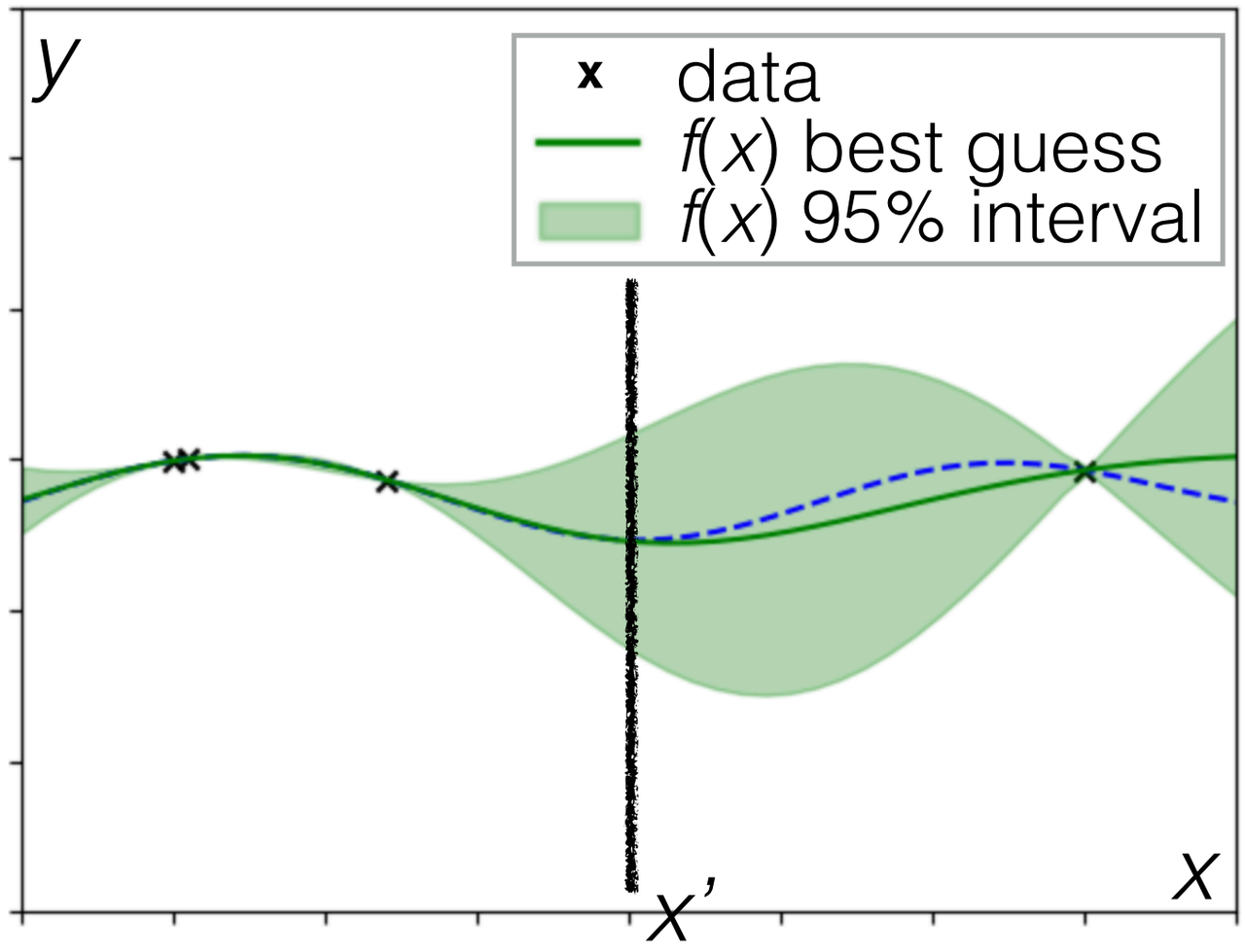


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

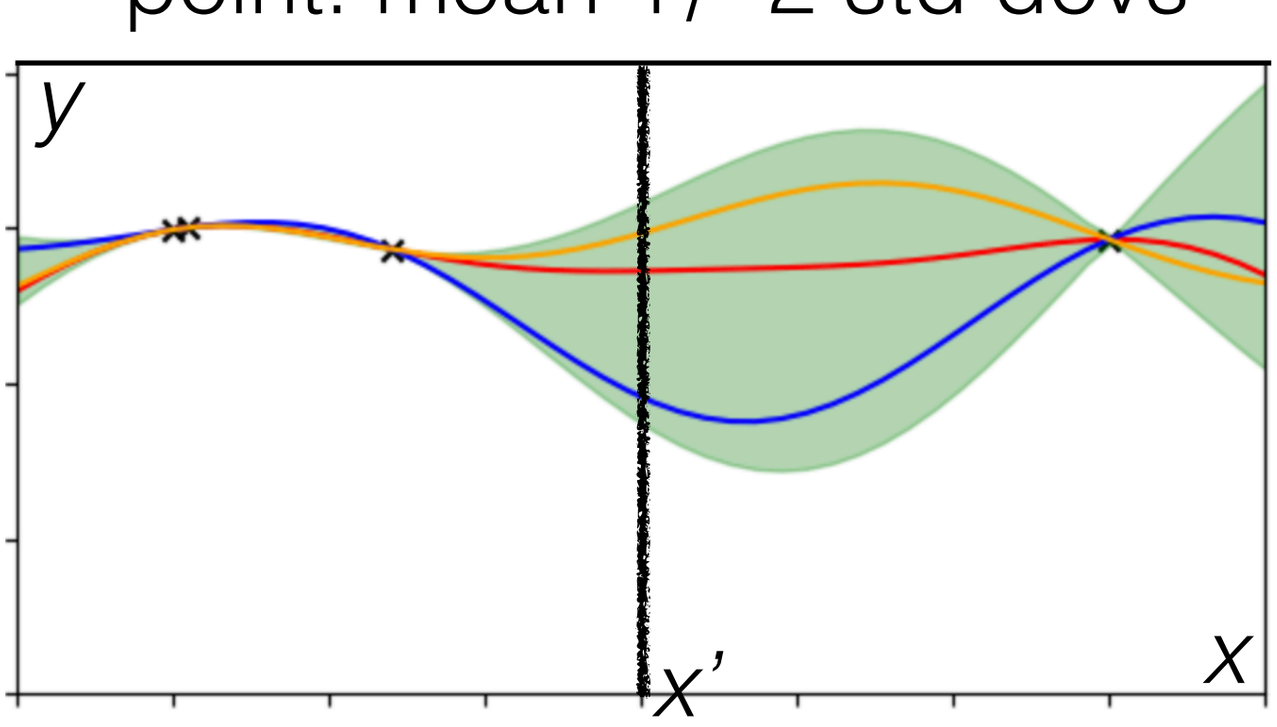


- Draw random *f* conditional on the training data

Closer look at the uncertainty interval

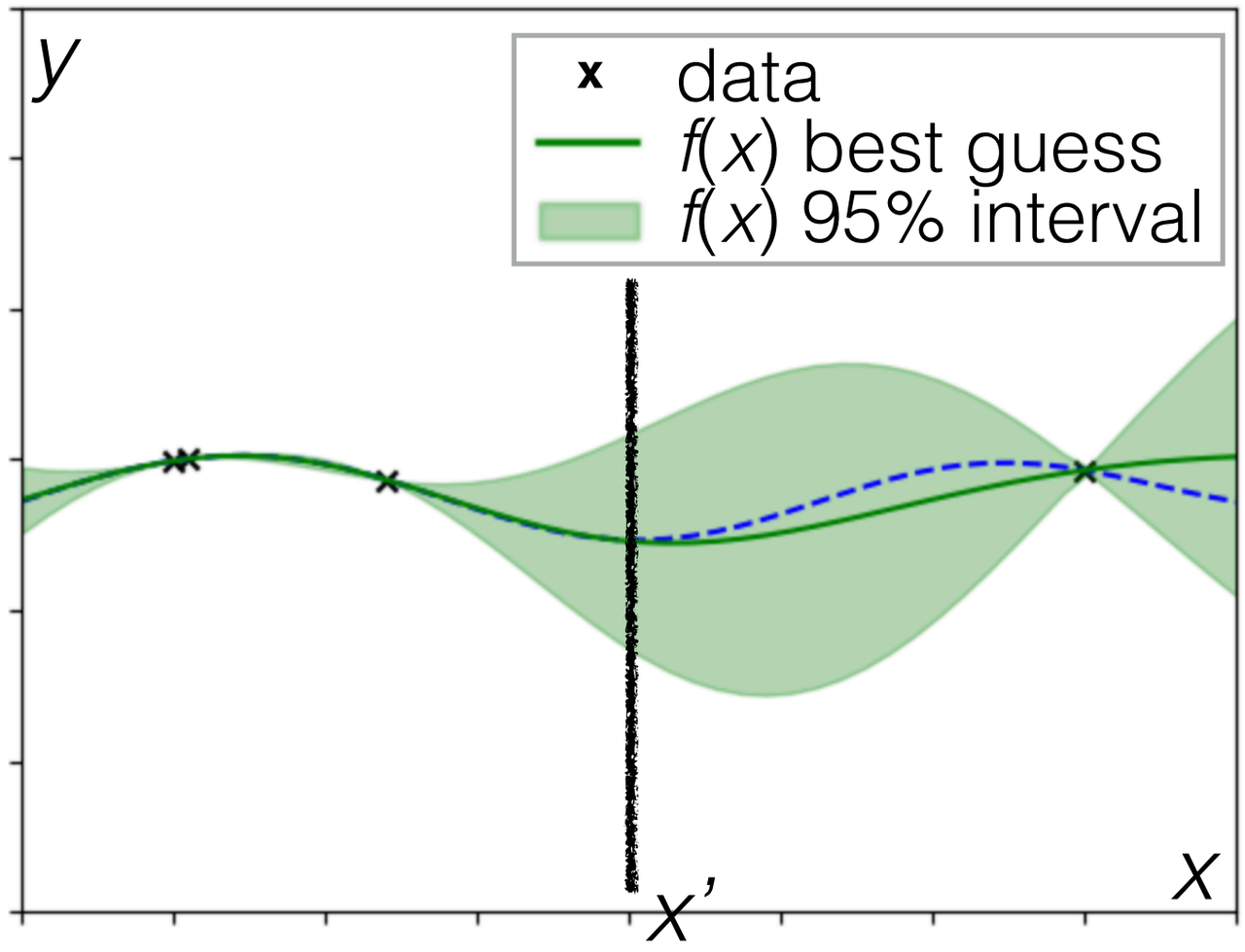


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

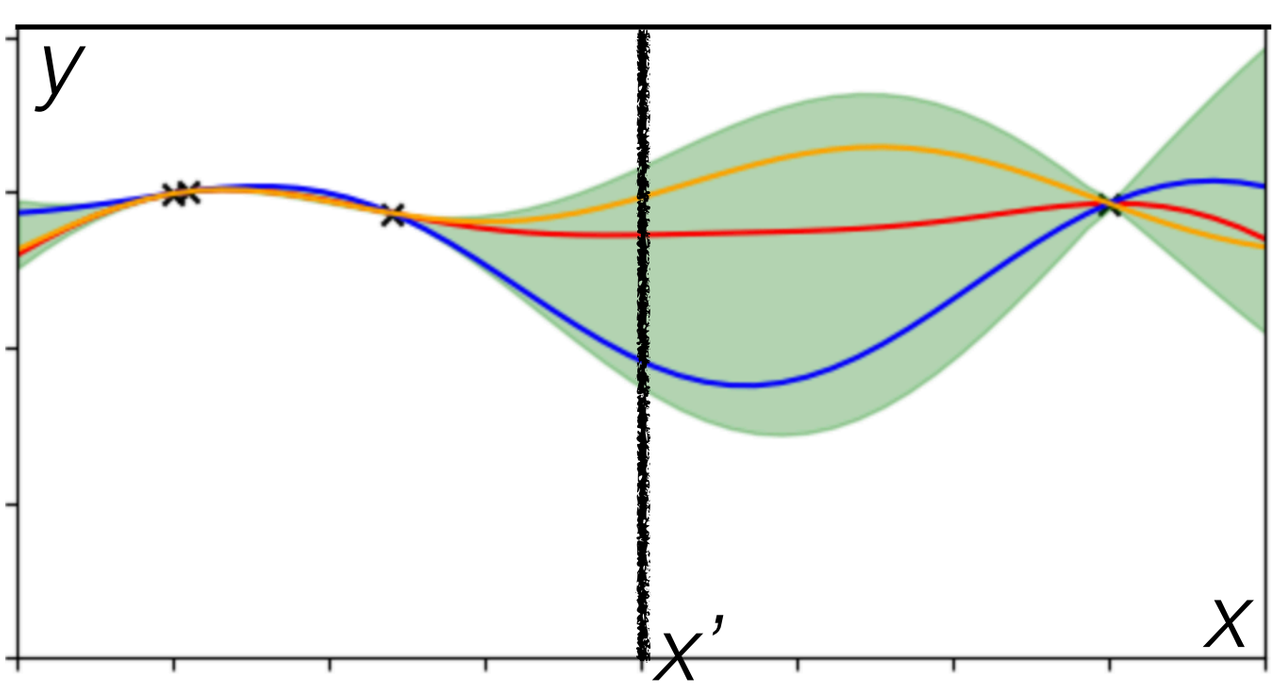


- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ?

Closer look at the uncertainty interval

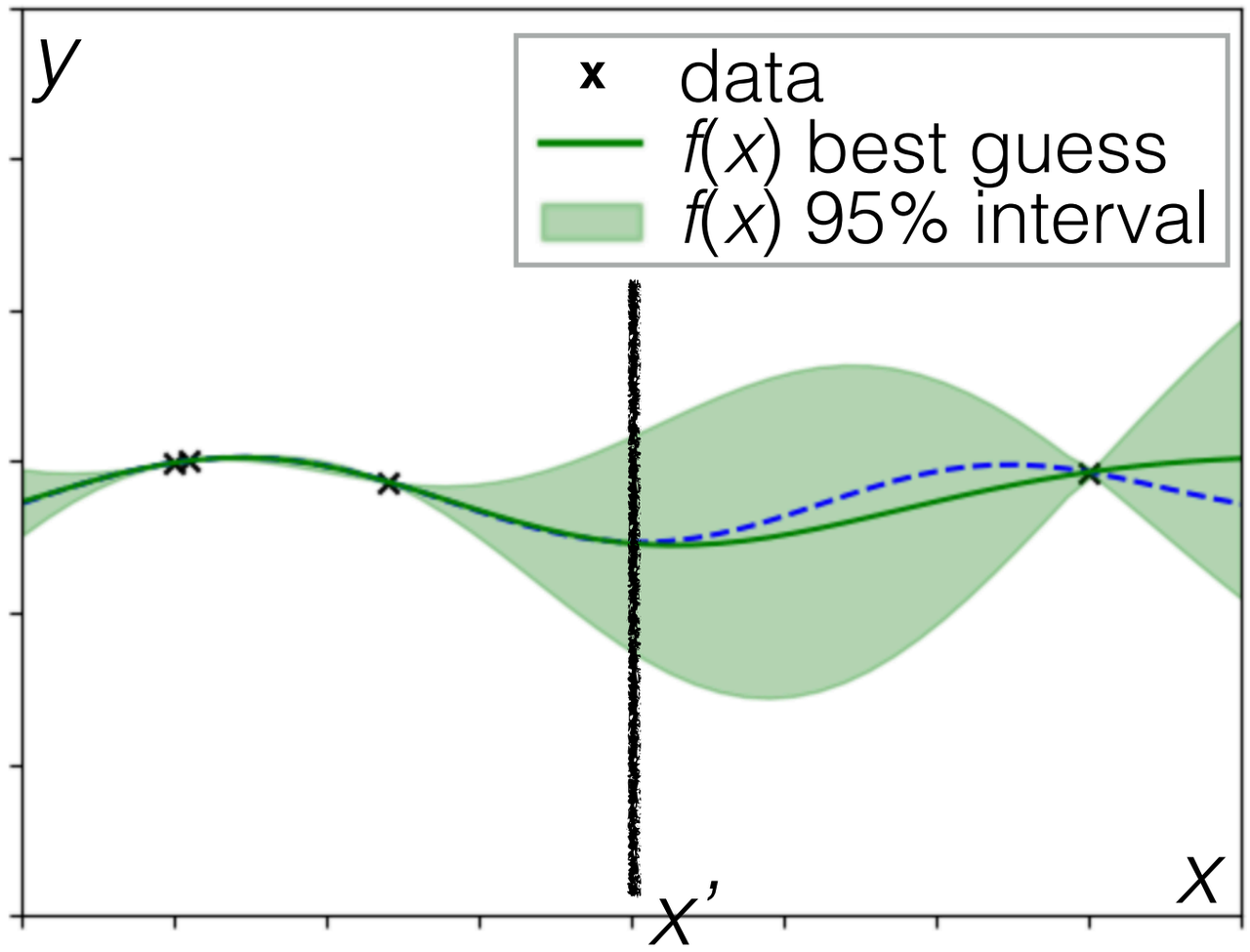


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

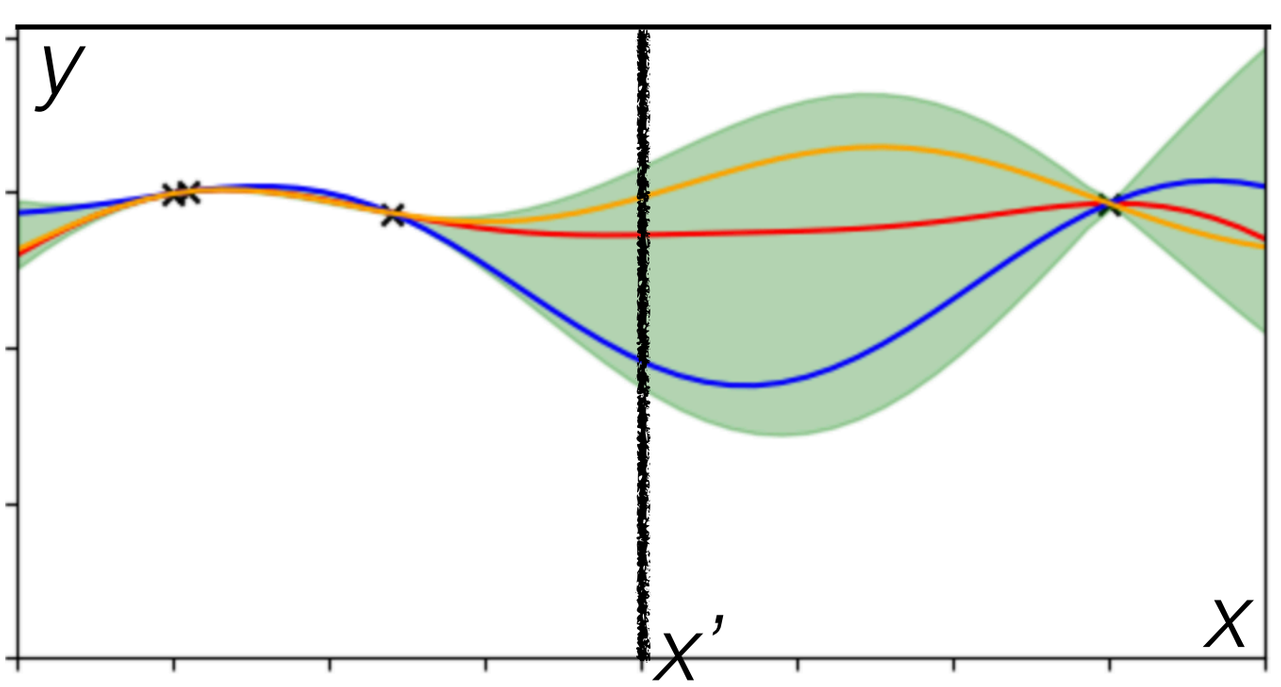


- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ~95%

Closer look at the uncertainty interval

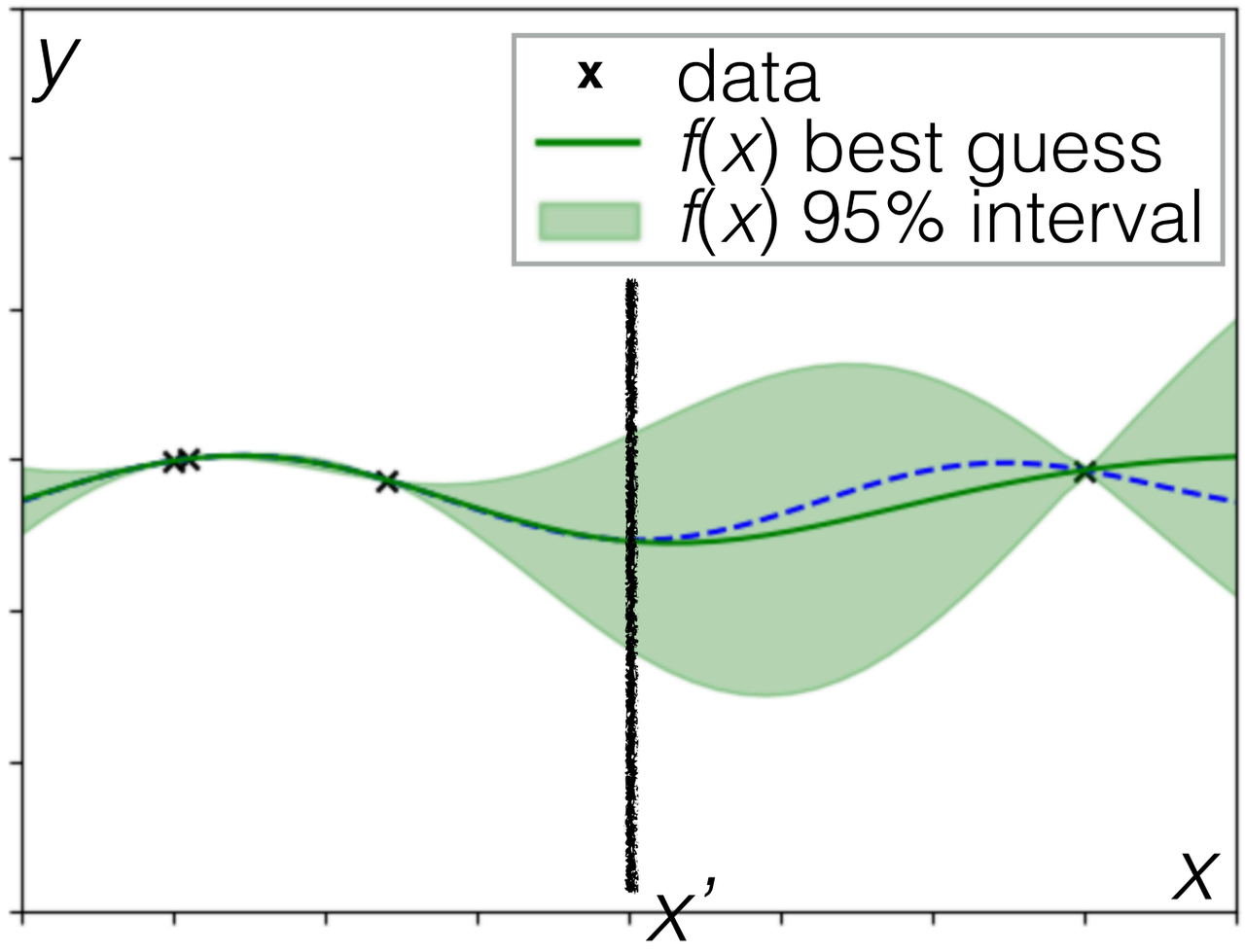


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

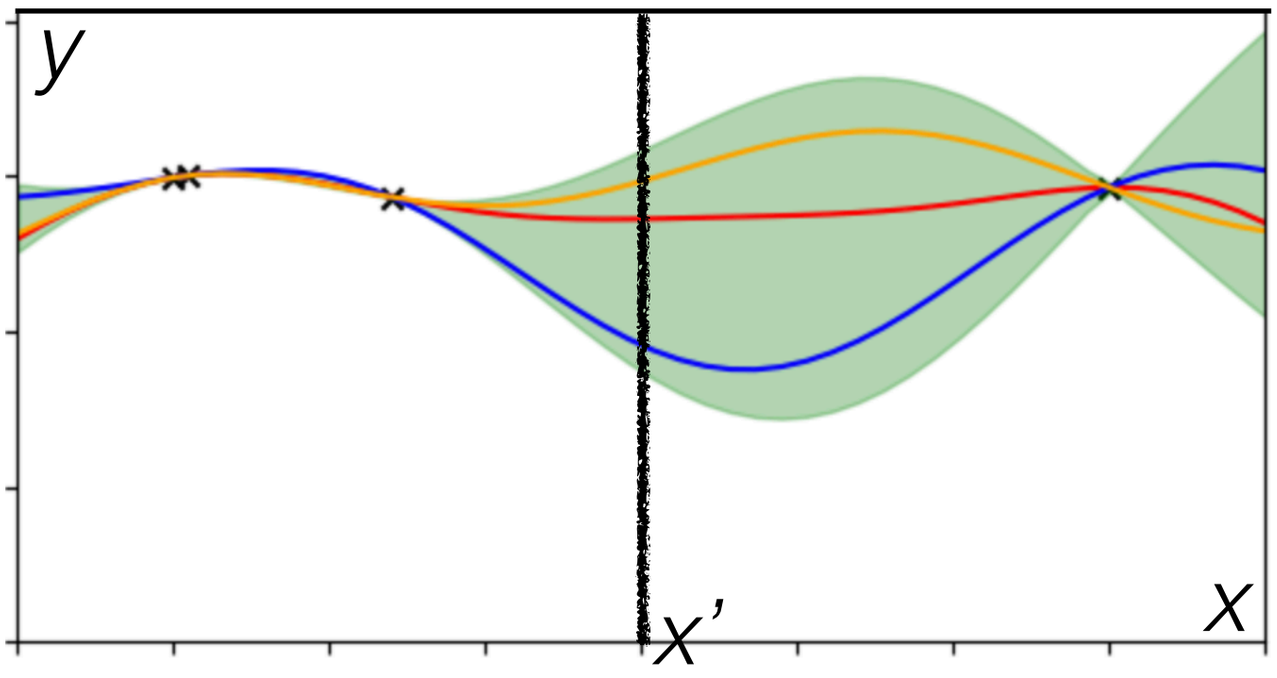


- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ~95%
- Probability that all points on *f* fall within the green interval across the whole plot ? ~95%

Closer look at the uncertainty interval

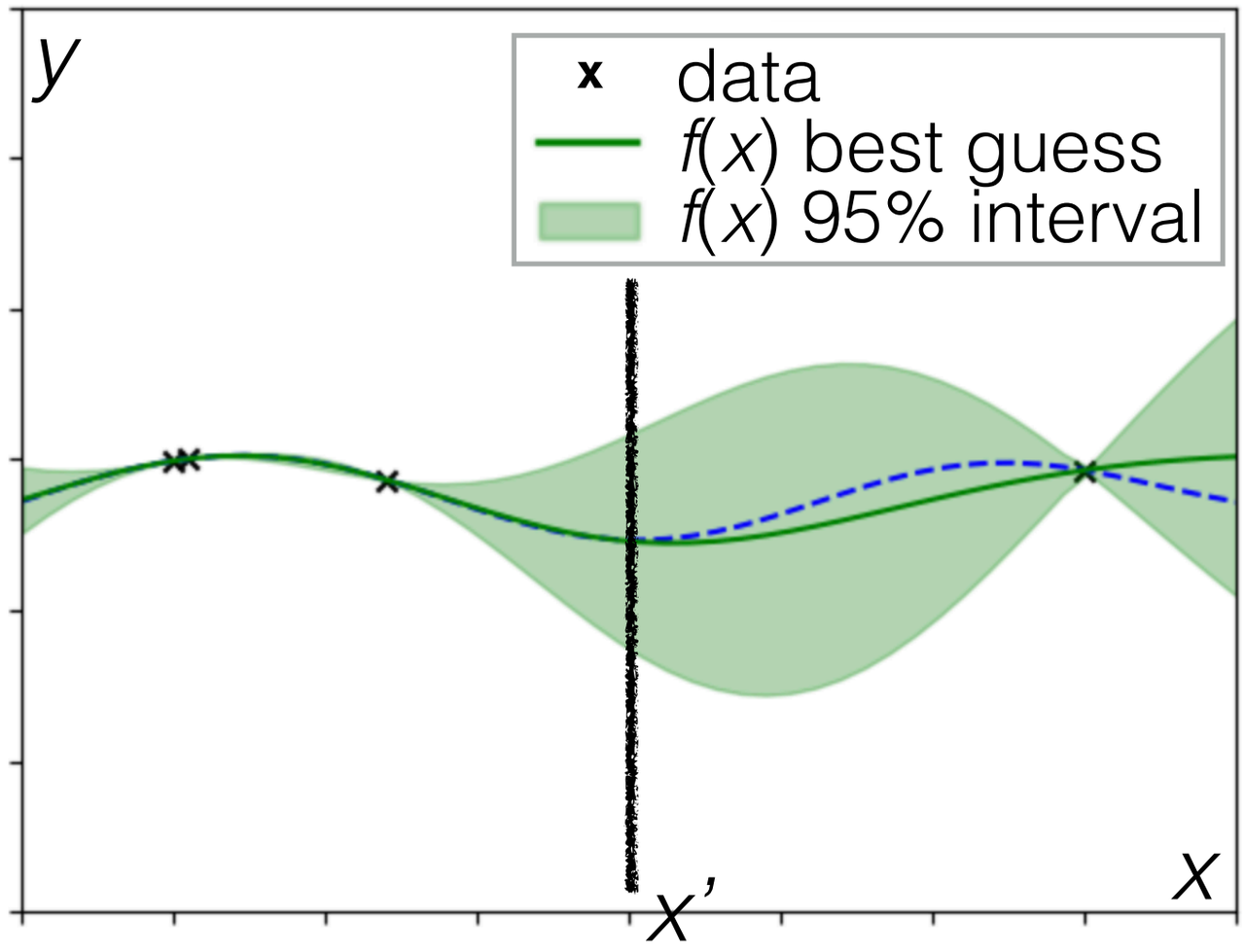


- Under GP, $f(x')|f(X), X, x'$ at a point *x'* is marginally Gaussian
- The green line at point *x'* is the mean of that Gaussian
- The green interval at that point: mean +/- 2 std devs

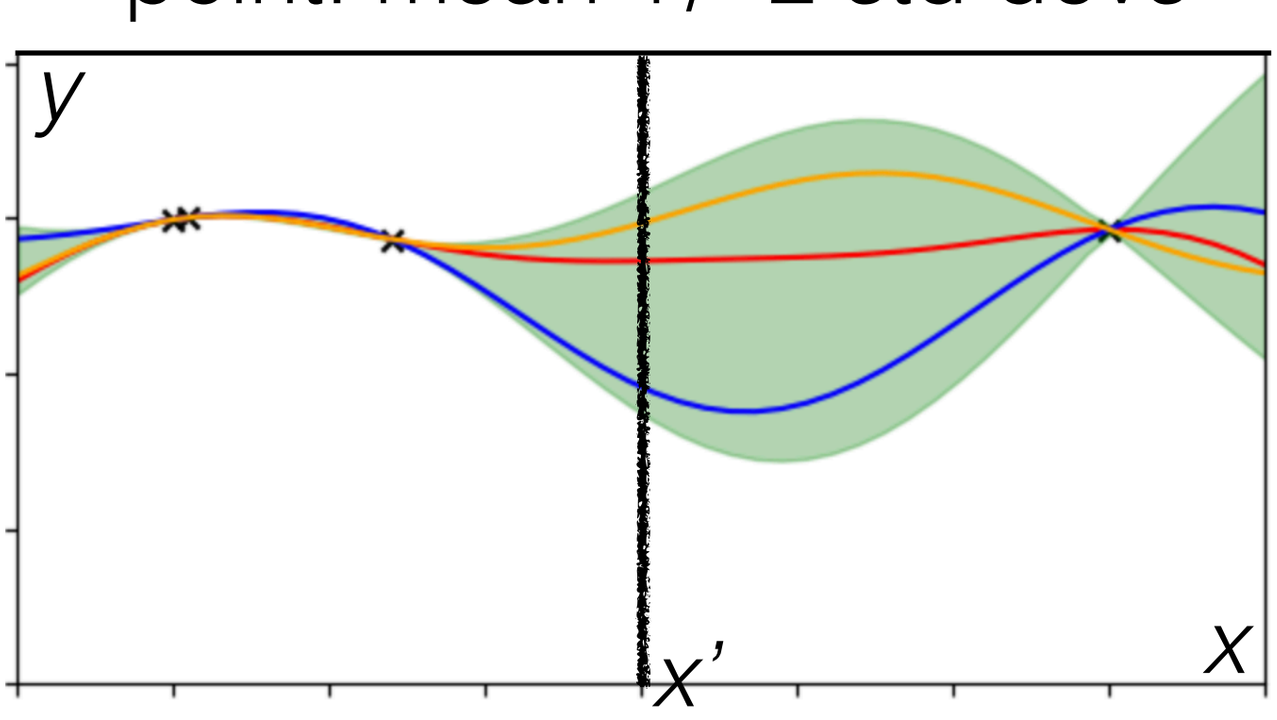


- Draw random *f* conditional on the training data
- Probability the draw is in the interval at *x'* is ~95%
- Probability that all points on *f* fall within the green interval across the whole plot will generally *not* be ~95%

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%

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

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*
  - 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: $\begin{bmatrix} y^{(1:N)} \\ f(X') \end{bmatrix}$

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

[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]

Bayesian optimization

[adapted slightly from Adams 2014 slides]

Bayesian optimization

- We have some sparsely observed points on a function

[adapted slightly from Adams 2014 slides]

Bayesian optimization

- We have some sparsely observed points on a function
- We would like to *minimize* that function

[adapted slightly from Adams 2014 slides]

Bayesian optimization

to maximize, just choose the negative of the original function

- We have some sparsely observed points on a function
- We would like to *minimize* that function

[adapted slightly from Adams 2014 slides]

Bayesian optimization

to maximize, just choose the negative of the original function

- We have some sparsely observed points on a function
- We would like to *minimize* that function

  E.g. max deep learning performance; max lift for booster

[adapted slightly from Adams 2014 slides]

Bayesian optimization

to maximize, just choose the negative of the original function

- We have some sparsely observed points on a function
- We would like to *minimize* that function

  E.g. max deep learning performance; max lift for booster
- We are willing to gather more data, but evaluating the function is expensive

[adapted slightly from Adams 2014 slides]

Bayesian optimization

to maximize, just choose the negative of the original function

- We have some sparsely observed points on a function
- We would like to *minimize* that function

  E.g. max deep learning performance; max lift for booster
- We are willing to gather more data, but evaluating the function is expensive

BayesOpt idea: Model the unknown function with a GP

- Use the means and uncertainties to decide where to evaluate the function next

[adapted slightly from Adams 2014 slides]

Bayesian optimization

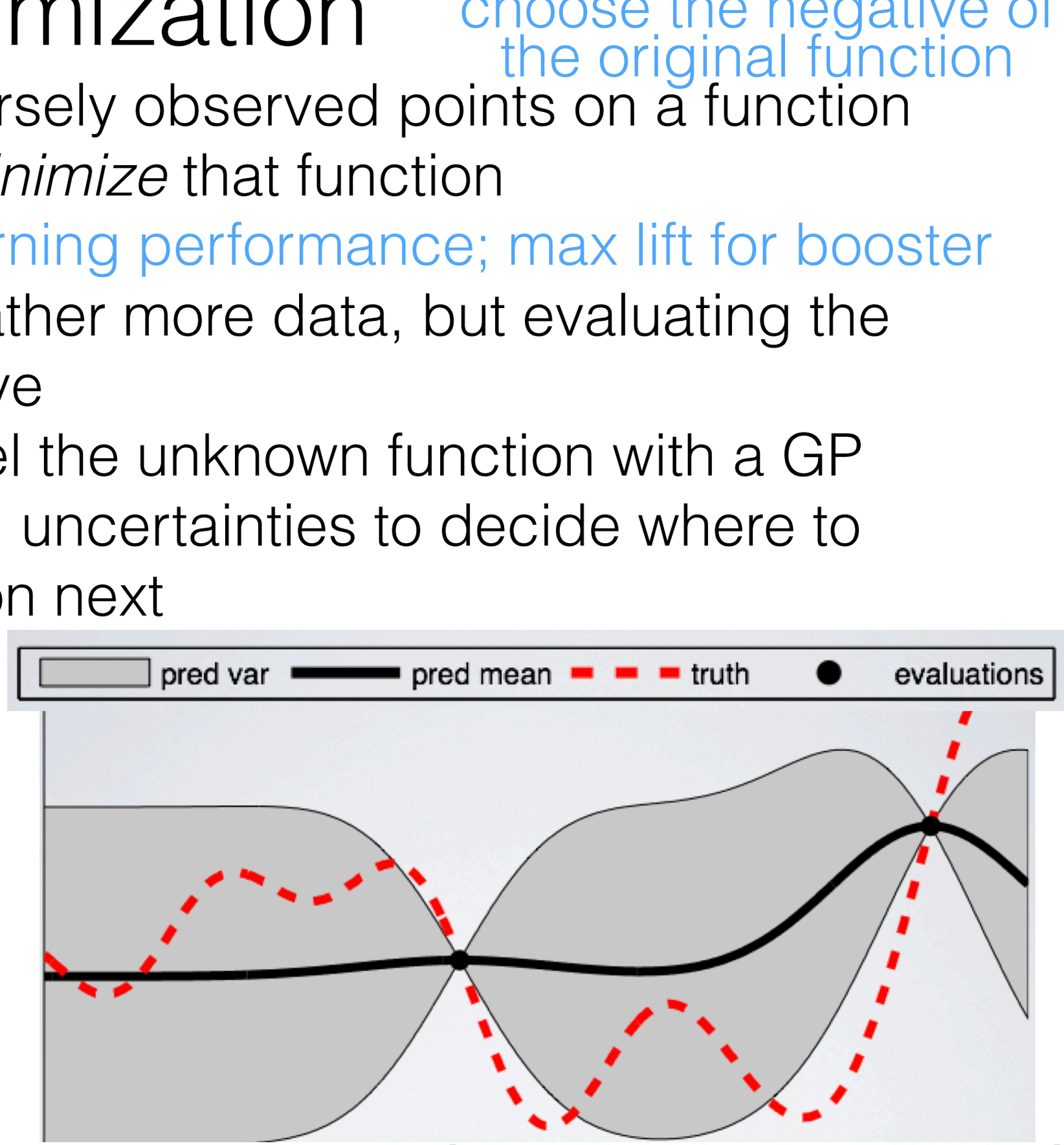
to maximize, just choose the negative of the original function

- We have some sparsely observed points on a function
- We would like to *minimize* that function

  E.g. max deep learning performance; max lift for booster
- We are willing to gather more data, but evaluating the function is expensive

BayesOpt idea: Model the unknown function with a GP

- Use the means and uncertainties to decide where to evaluate the function next



[adapted slightly from Adams 2014 slides]

Bayesian optimization

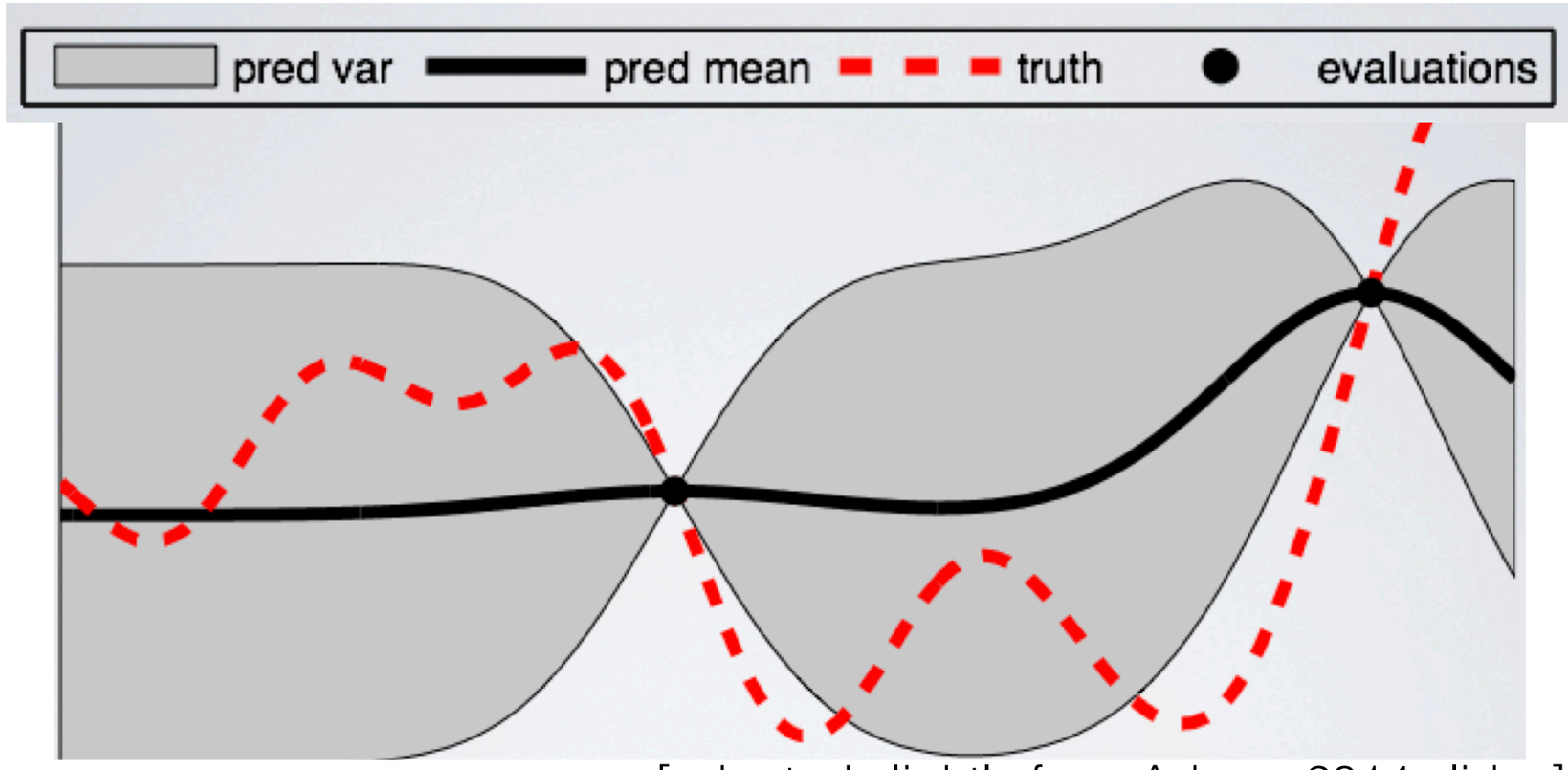
to maximize, just choose the negative of the original function

- We have some sparsely observed points on a function
- We would like to *minimize* that function

  E.g. max deep learning performance; max lift for booster
- We are willing to gather more data, but evaluating the function is expensive

BayesOpt idea: Model the unknown function with a GP

- Use the means and uncertainties to decide where to evaluate the function next
- First have to decide how to trade off exploration and exploitation



[adapted slightly from Adams 2014 slides]

Bayesian optimization

to maximize, just choose the negative of the original function

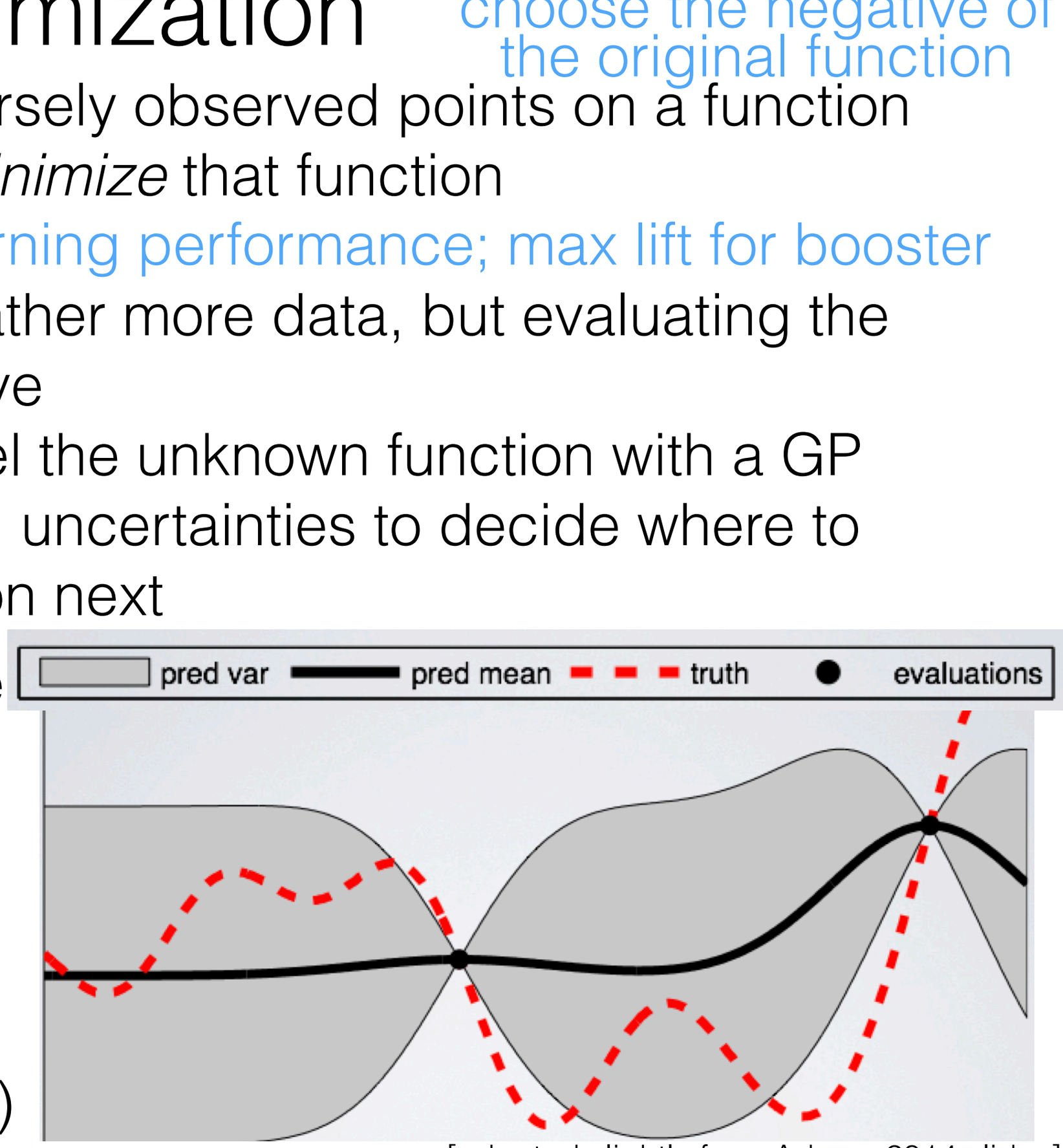
- We have some sparsely observed points on a function
- We would like to *minimize* that function

  E.g. max deep learning performance; max lift for booster
- We are willing to gather more data, but evaluating the function is expensive

BayesOpt idea: Model the unknown function with a GP

- Use the means and uncertainties to decide where to evaluate the function next

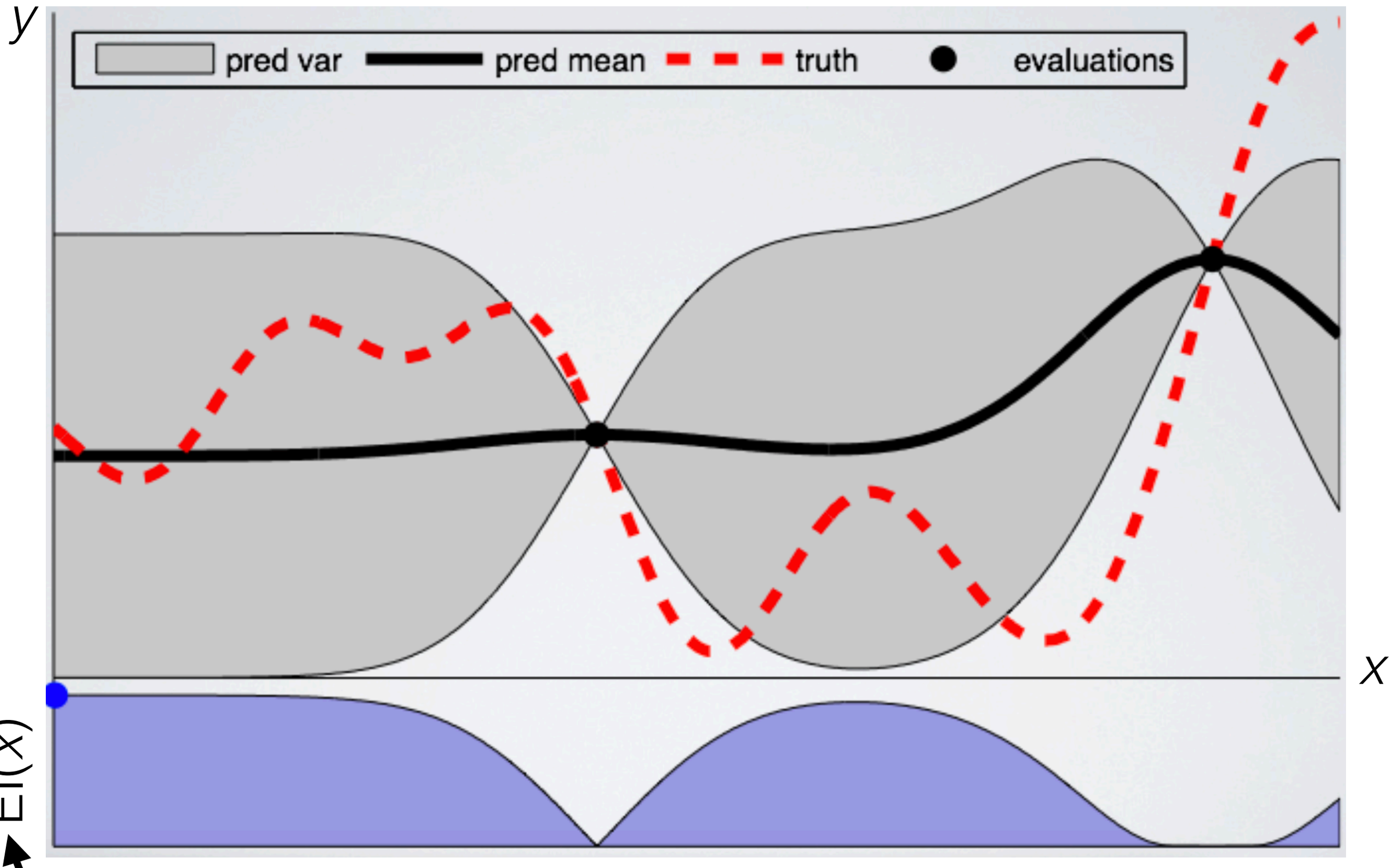
- First have to decide how to trade off exploration and exploitation
  - One option is "expected improvement" (EI)



[adapted slightly from Adams 2014 slides]

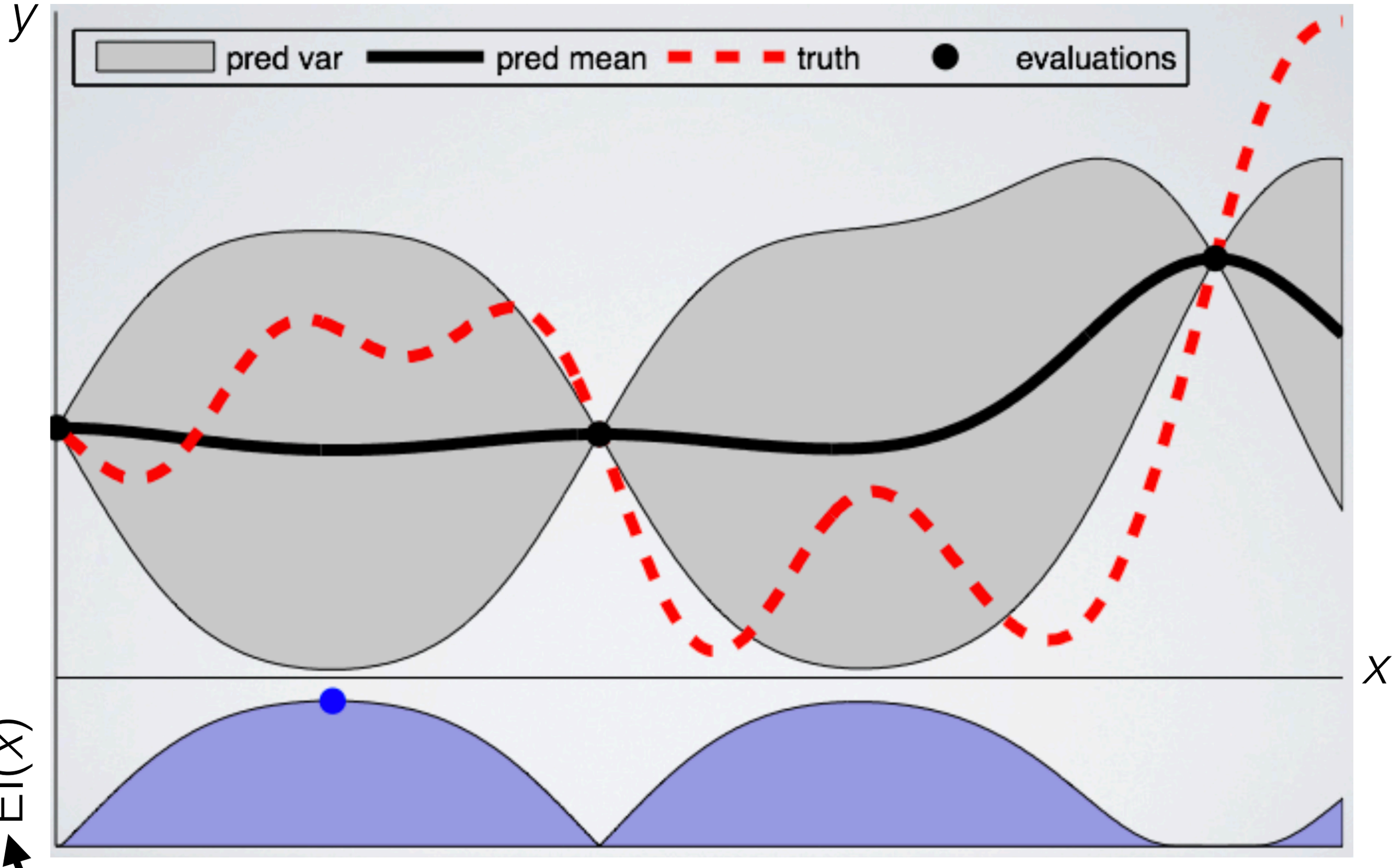
Bayesian optimization

[adapted slightly from Adams 2014 slides]



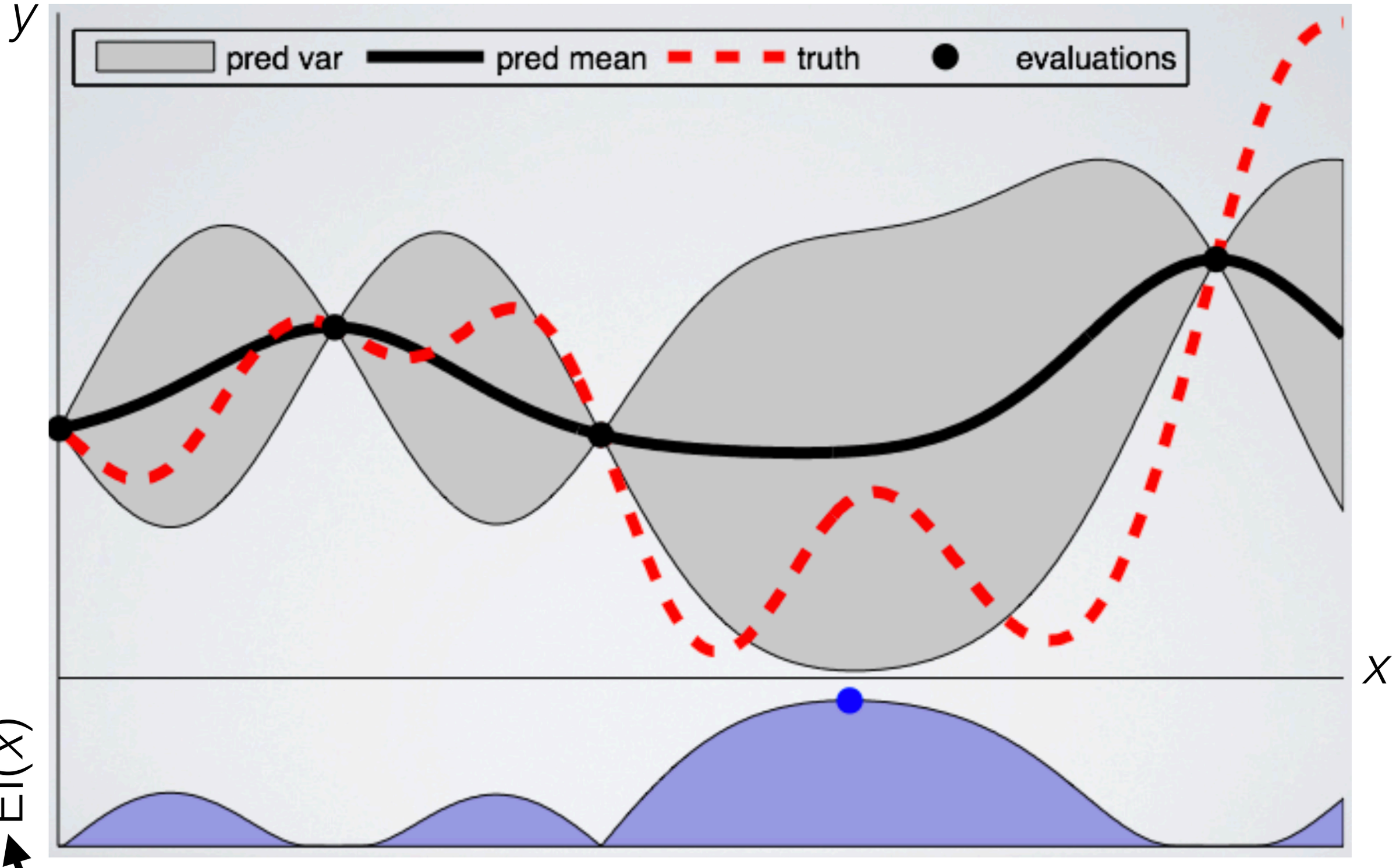
Bayesian optimization

[adapted slightly from Adams 2014 slides]



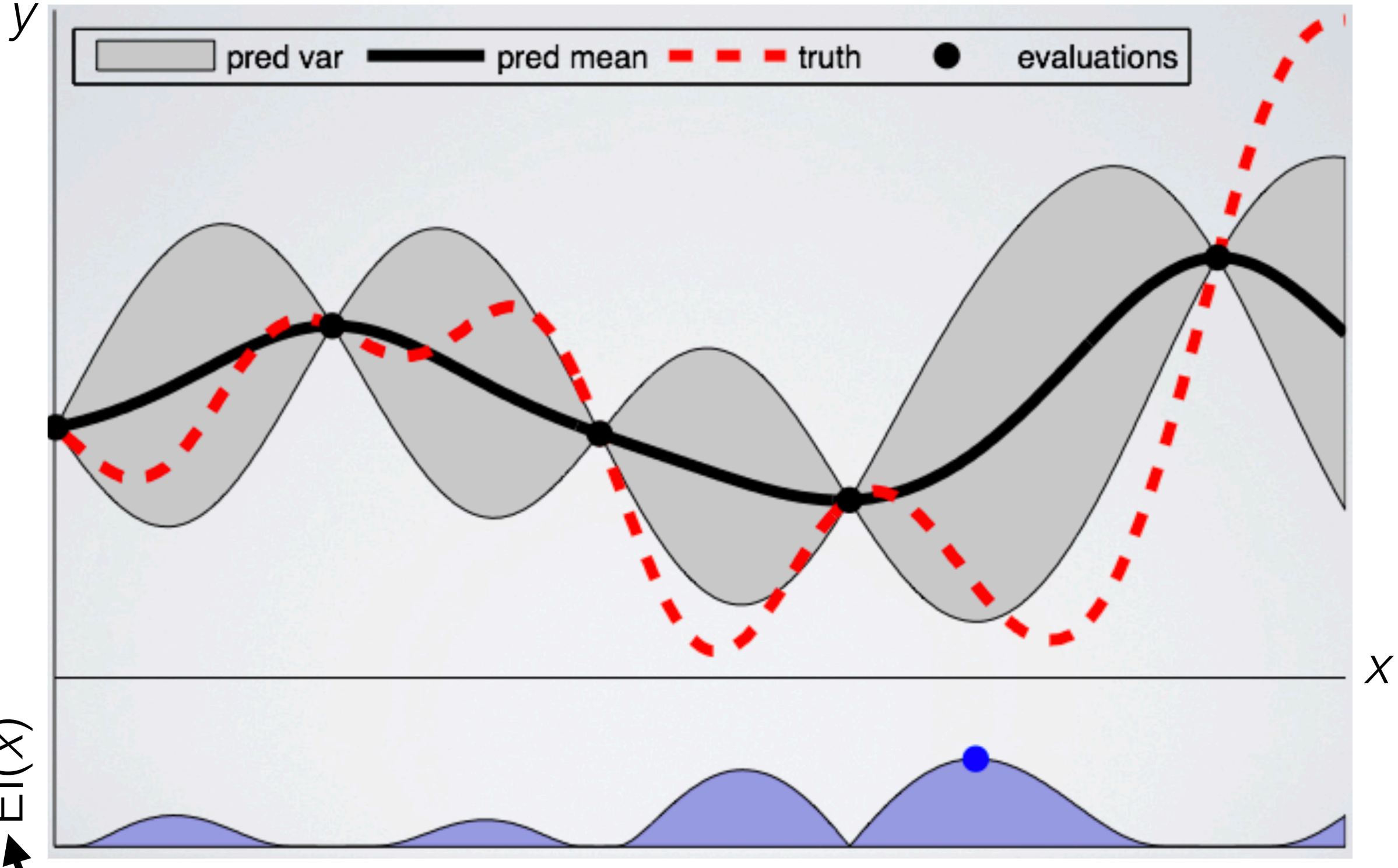
Bayesian optimization

[adapted slightly from Adams 2014 slides]



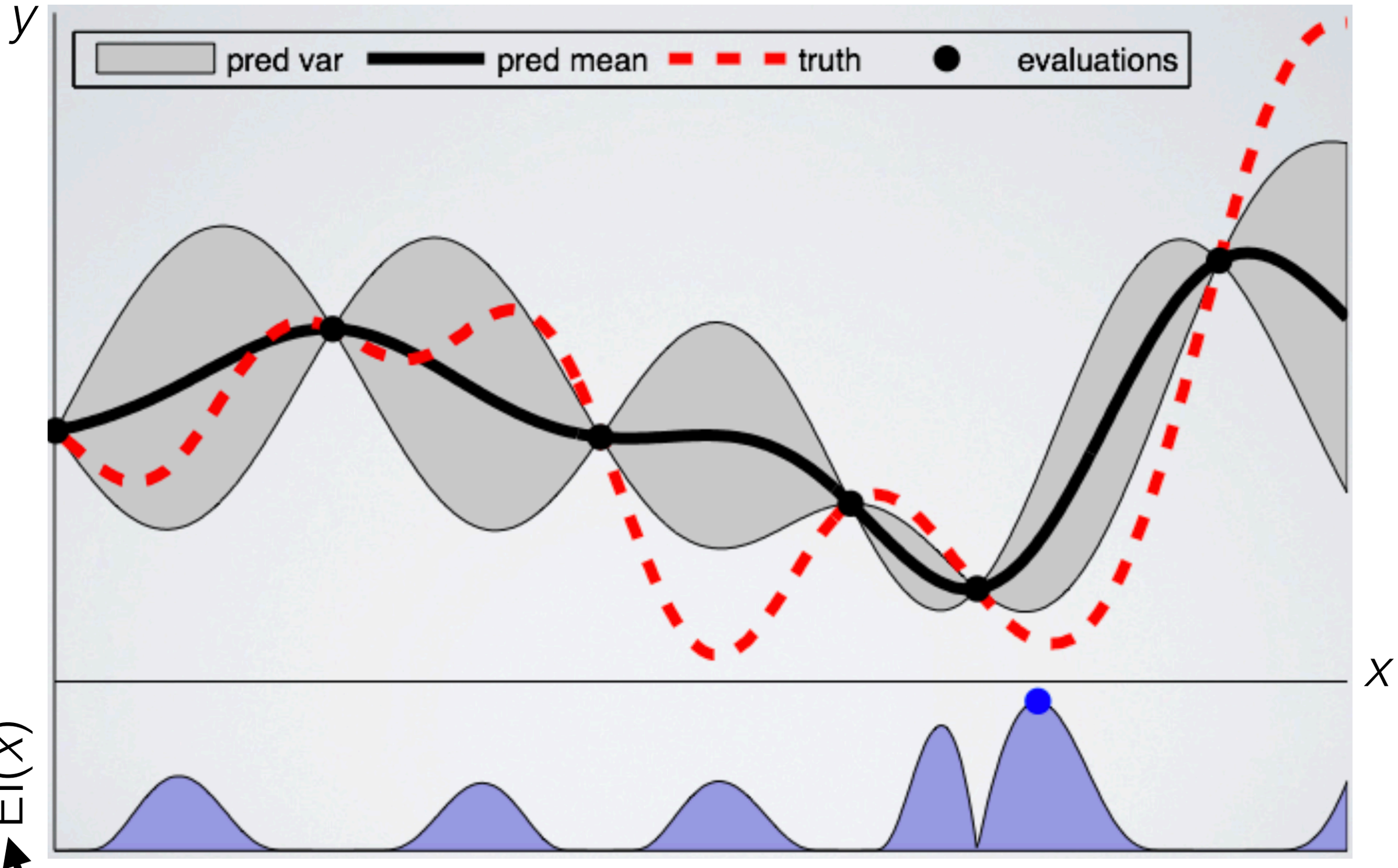
Bayesian optimization

[adapted slightly from Adams 2014 slides]



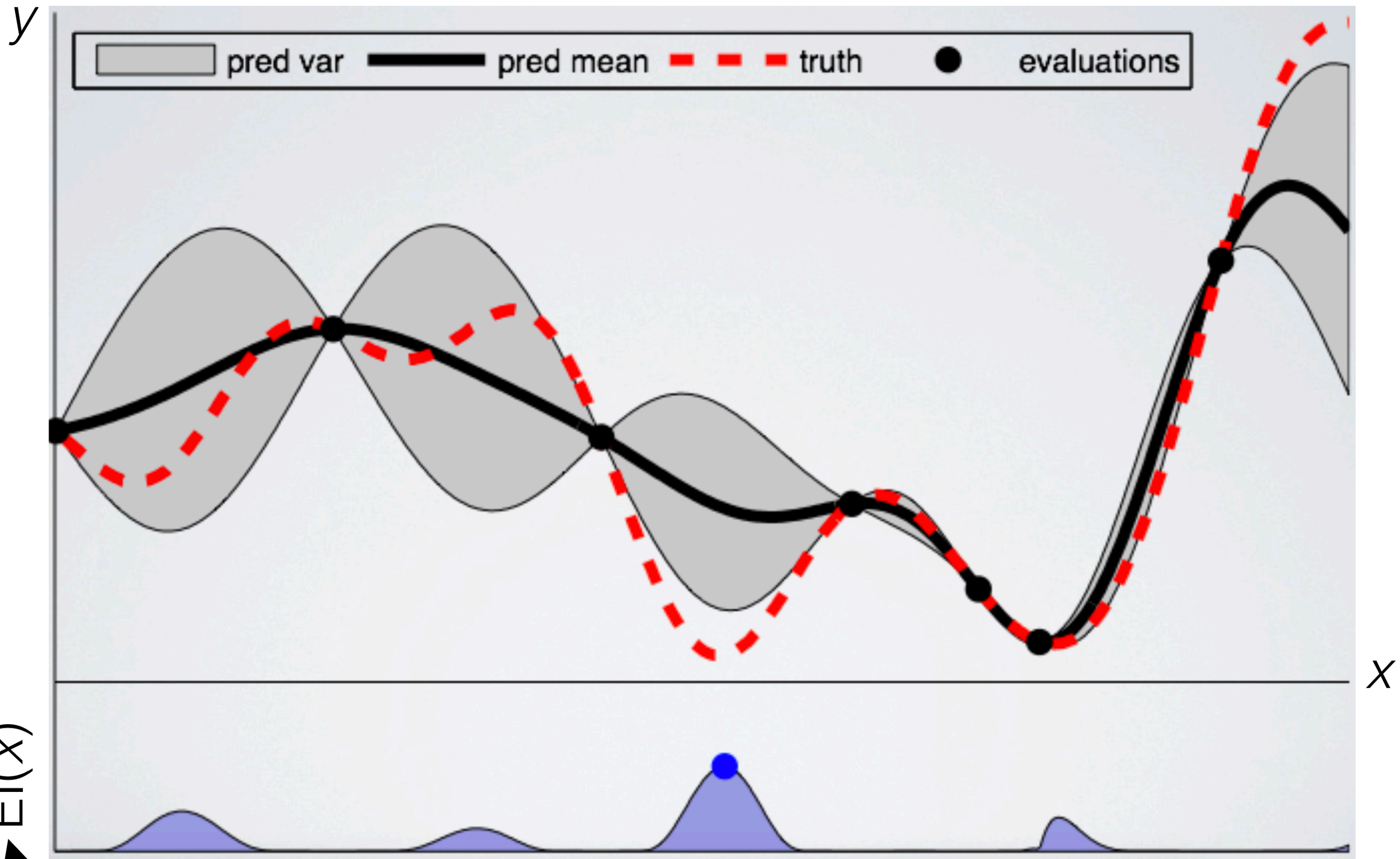
Bayesian optimization

[adapted slightly from Adams 2014 slides]



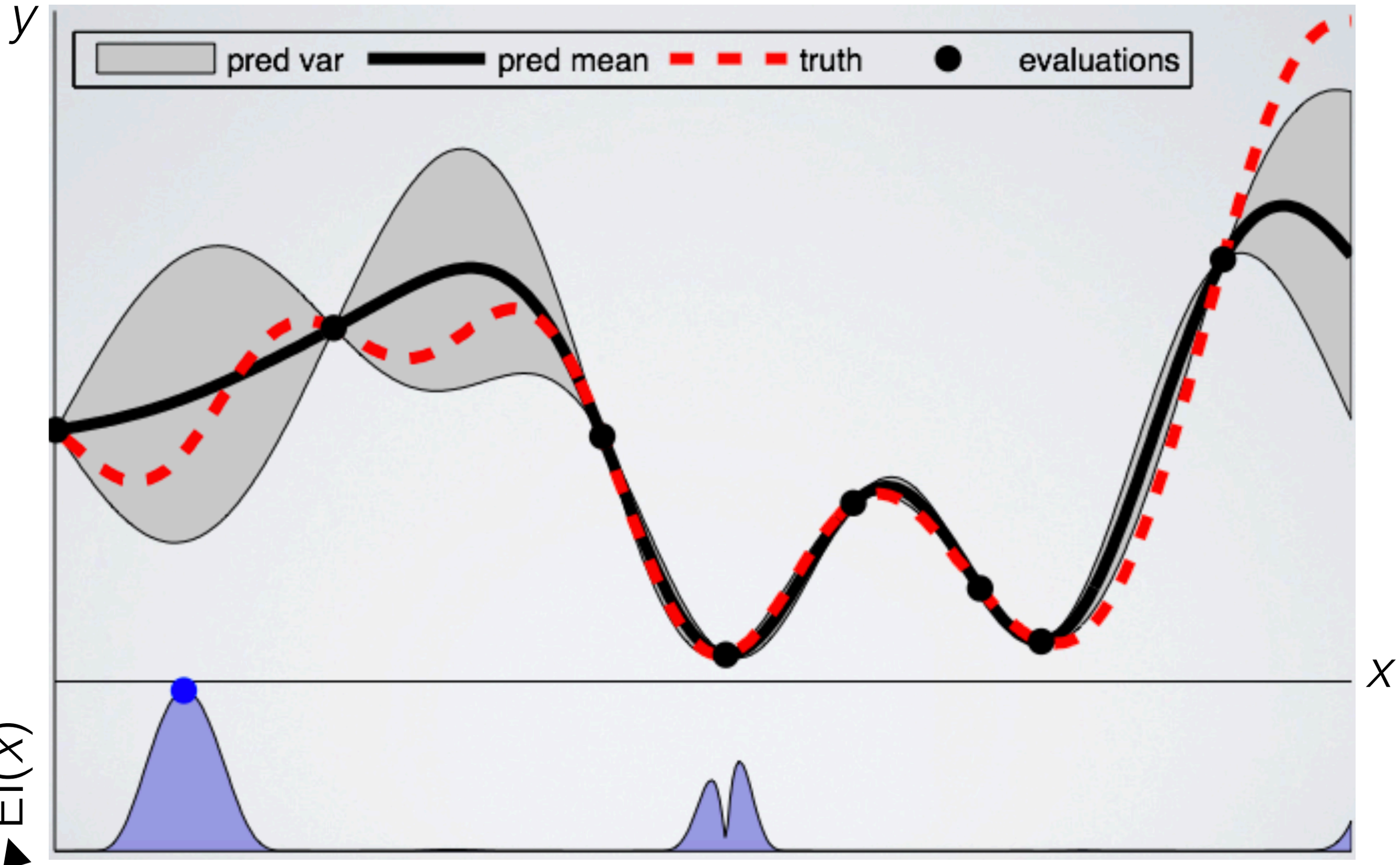
Bayesian optimization

[adapted slightly from Adams 2014 slides]



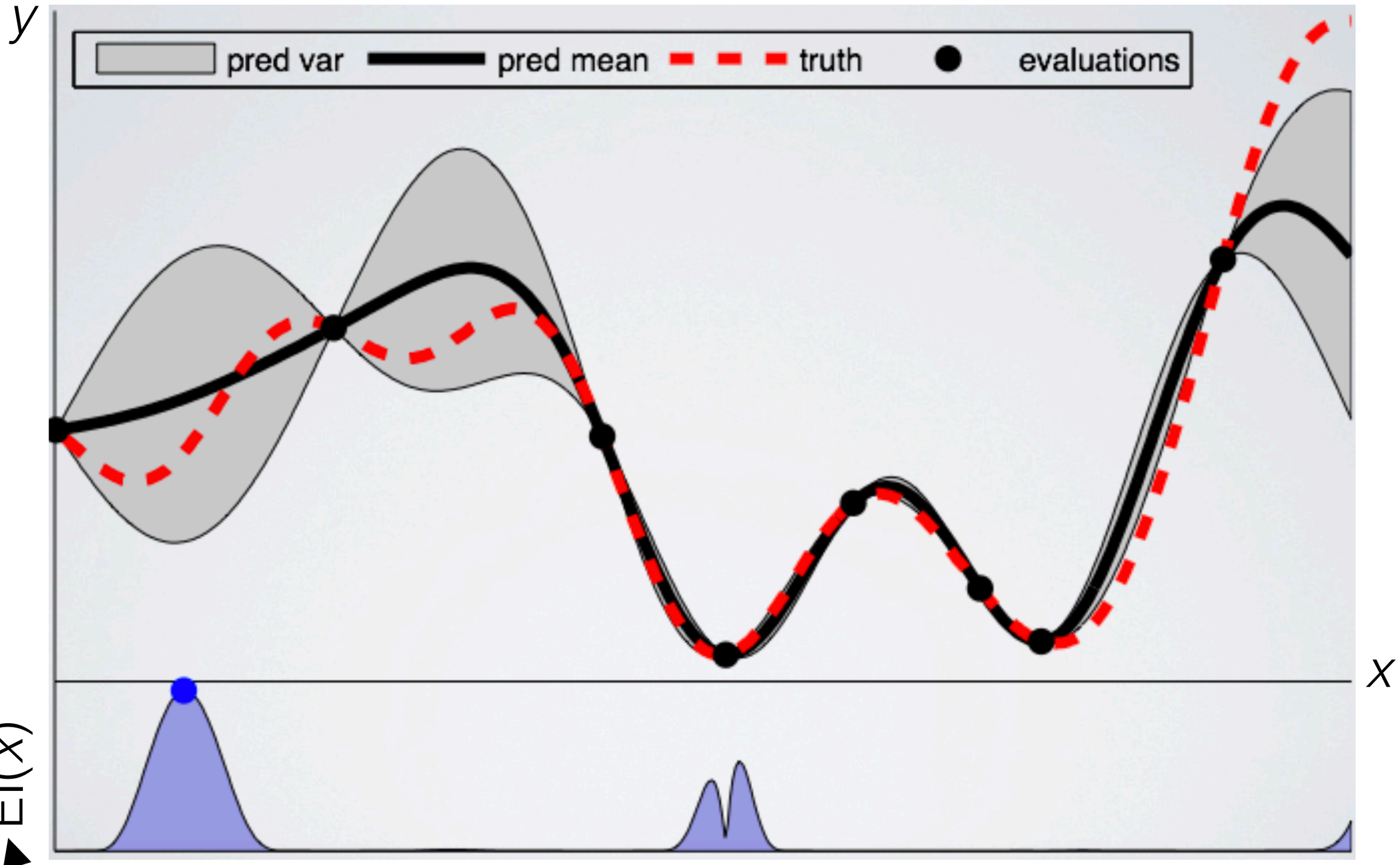
Bayesian optimization

[adapted slightly from Adams 2014 slides]



Bayesian optimization

[adapted slightly from Adams 2014 slides]



Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference
  - Prediction & uncertainty quantification
- Using GPs to optimize a function (Bayesian optimization)
- What are the limits? What can go wrong?

- 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 artificial intelligence/ machine learning methods)

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?

Some highlights of what got cut for time

- There's no way we could possibly cover every topic related to Gaussian processes, Bayesian optimization, or regression!
- Here are some high-level summary points beyond what we discussed together:
  - In high dimensions, all points are "far" from all other points. And recall that the GP reverts to the prior when inputs are "far" from other inputs.
  - 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
- A lot of the challenges we discussed are general for regression, not specific to Gaussian processes

Roadmap

- A Bayesian approach
- What is a Gaussian process (GP)?
  - A particularly popular version in practice
- Gaussian process inference
  - Prediction & uncertainty quantification
- Using GPs to optimize a function (Bayesian optimization)
- What are the limits? What can go wrong?

- 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 artificial intelligence/ machine learning methods)

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