Probabilistic methods for Machine Learning

September
27 2022

Filipp Gusev

Topics For Today

- **Probability recap**
- Naïve Bayes
- Gaussian process
- Bayesian hyperparameter optimization

Probability: definitions

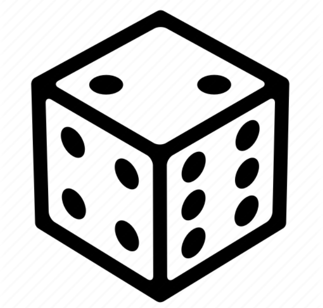
Sample Space Ω:
The set of all outcomes of a random experiment
Event space $\mathcal{F}$ **:**
A set whose elements $A \in \mathcal{F}$ (called events) are subsets of Ω (i.e., $A \subseteq \Omega$ is a collection of possible outcomes of an experiment)
Probability measure:
A function $P : \mathcal{F} \to \mathbb{R}$ that satisfies the following properties:

- $P(A) \geq 0$, for all $A \in \mathcal{F}$
- $P(\Omega)=1$
- If $A_1, A_2, \ldots$ are disjoint events (i.e., $A_i \cap A_j = \emptyset$ for $i \neq j$), then $P(\cup_i A_i) = \sum_i P(A_i)$

Probability: definitions

Sample Space Ω:
The set of all outcomes of a random experiment
Event space $\mathcal{F}$ **:**
A set whose elements $A \in \mathcal{F}$ (called events) are subsets of Ω (i.e., $A \subseteq \Omega$ is a collection of possible outcomes of an experiment)
Probability measure:
A function $P : \mathcal{F} \rightarrow \mathbb{R}$ that satisfies the following properties:

- $P(A) \geq 0$, for all $A \in \mathcal{F}$
- $P(\Omega)=1$
- If $A_1, A_2, \ldots$ are disjoint events (i.e., $A_i \cap A_j = \emptyset$ for $i \neq j$), then $P(\cup_i A_i) = \sum_i P(A_i)$



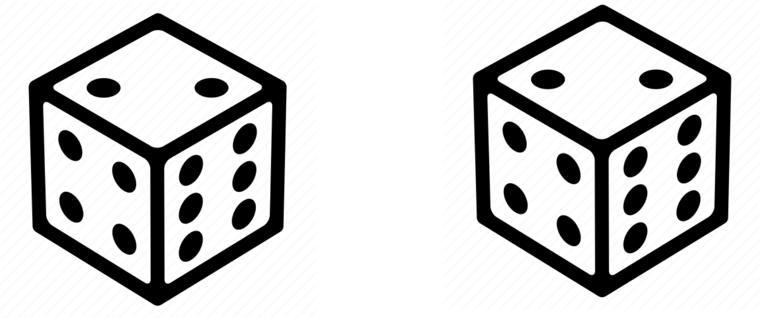
Example: Tossing a six-sided die
$\Omega = \{1, 2, 3, 4, 5, 6\}$
$\mathcal{F} = \{\emptyset, \Omega\} = \{\emptyset, 1, 2, 3, 4, 5, 6\}$
$P(\emptyset) = 0$, $P(\Omega) = 1$
$P(\{1\}) = \frac{1}{6}$
$P(\{1,2,3,4\}) = \frac{4}{6}$
$P(\{2,4,6\}) = \frac{3}{6}$

Probability: Conditional probability and independence

The conditional probability of any event A given B is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Example: tossing two 6-sided die.
A = {We got same digits on first and second die}
B = {Sum of digits on first and second die more than 8}



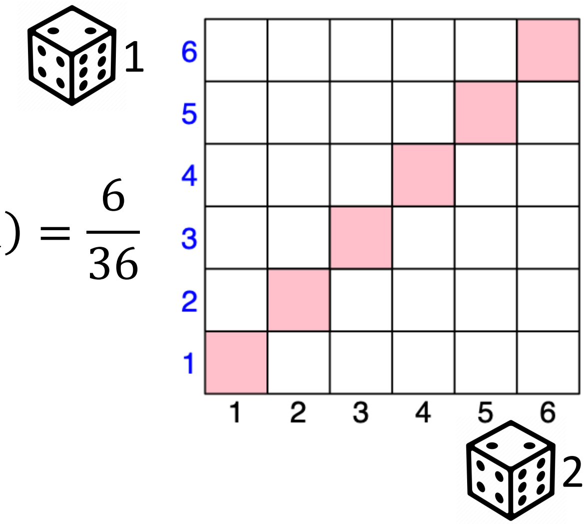
Probability: Conditional probability and independence

The conditional probability of any event A given B is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Example: tossing two 6-sided die.
A = {We got same digits on first and second die}
B = {Sum of digits on first and second die more than 8}

$$P(A) = \frac{6}{36}$$



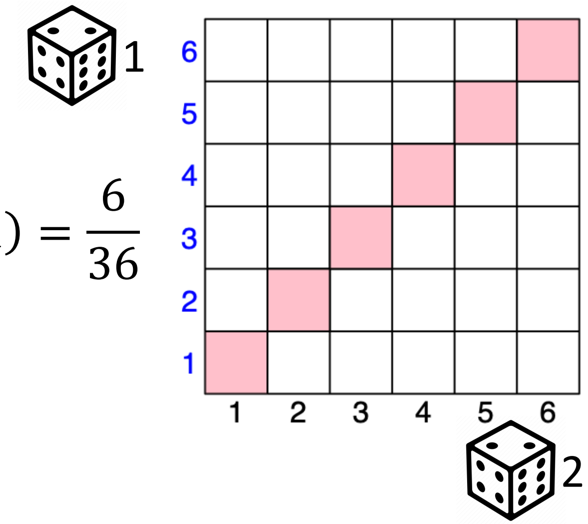
Probability: Conditional probability and independence

The conditional probability of any event A given B is

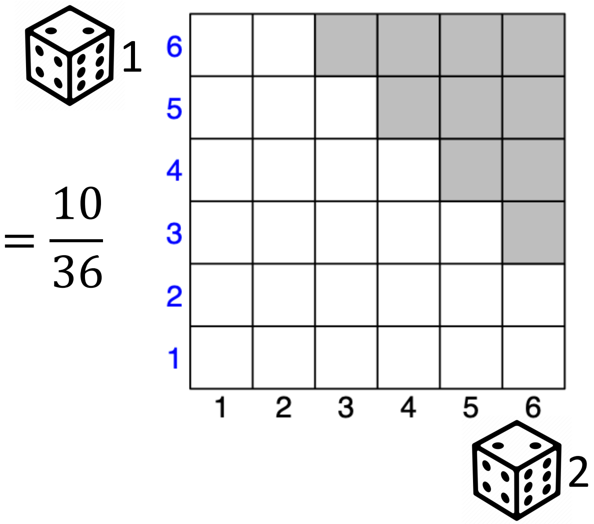
$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Example: tossing two 6-sided die.
A = {We got same digits on first and second die}
B = {Sum of digits on first and second die more than 8}

$$P(A) = \frac{6}{36}$$



$$P(B) = \frac{10}{36}$$



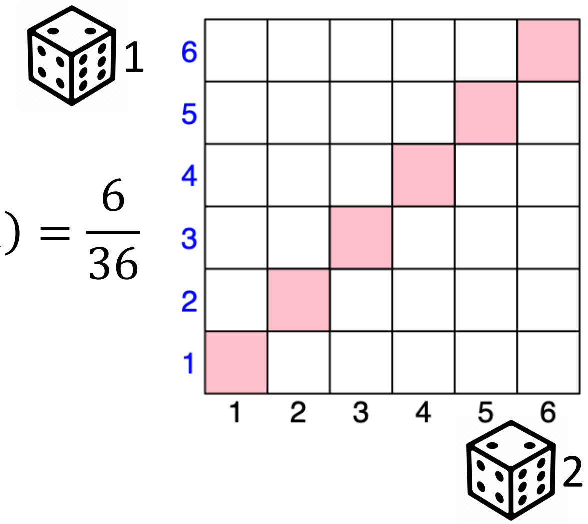
Probability: Conditional probability and independence

The conditional probability of any event A given B is

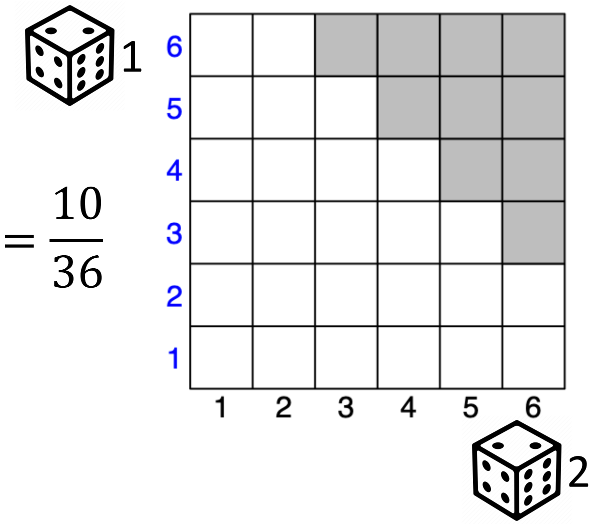
$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Example: tossing two 6-sided die.
A = {We got same digits on first and second die}
B = {Sum of digits on first and second die more than 8}

$$P(A) = \frac{6}{36}$$

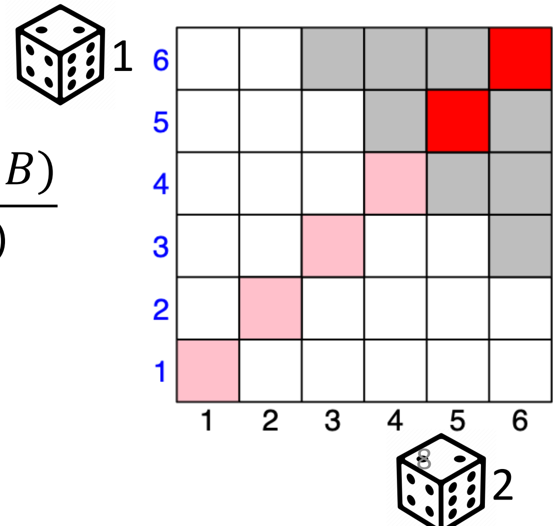


$$P(B) = \frac{10}{36}$$



$$P(A|B) = \frac{P(A \ \cap B)}{P(B)}$$

$$= \frac{2/36}{10/36} = \frac{1}{5}$$



Probability: Conditional probability and independence

The conditional probability of any event A given B is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

In other words, P (A|B) is the probability measure of the event A after observing the occurrence of event B.

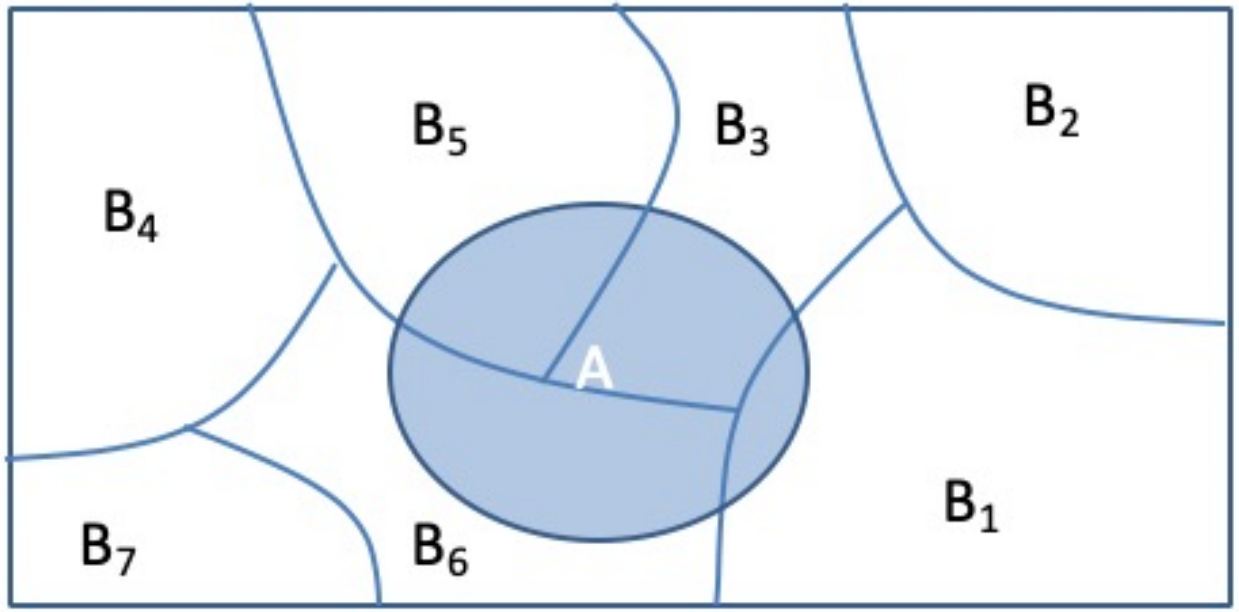
Two events are called **independent** if and only if

$$P(A \ \cap B) = P(A)\text{x}P(B)$$

Independence is equivalent to saying that **observing B does not have any effect on the probability of A**.

Probability: Law of total probability

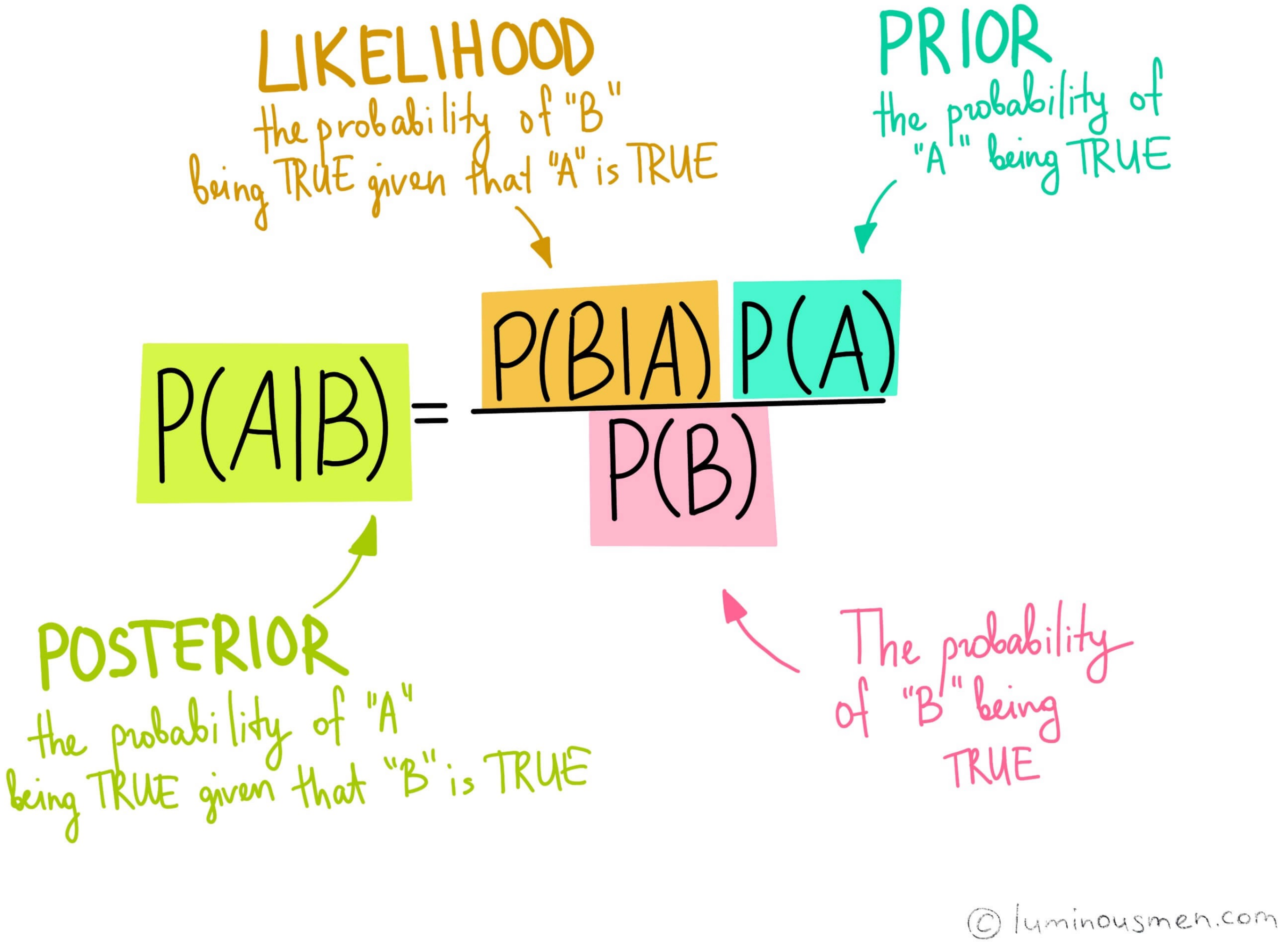
$$P(A) = \sum_{i=1}^{n} P(A \cap B_i) = \sum_{i=1}^{n} P(A|B_i)P(B_i)$$



Probability: Bayes' theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

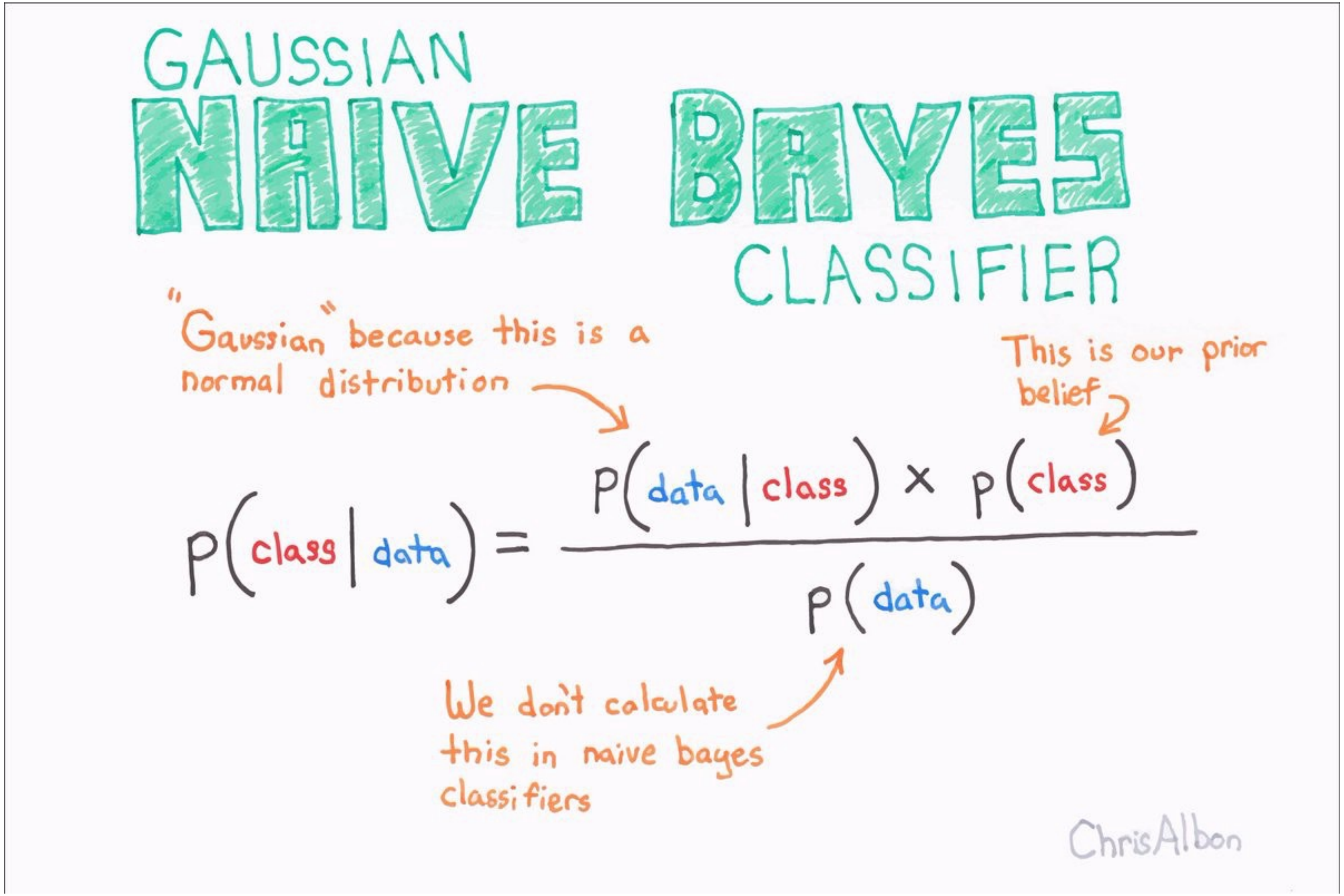
Probability: Bayes' theorem



Topics

- Probability recap
- **Naïve Bayes**
- Gaussian process
- Bayesian hyperparameter optimization

Naïve Bayes: introduction

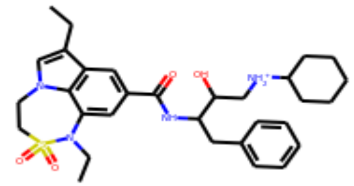
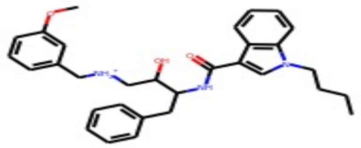


Naïve Bayes: introduction

Problem statement:

- Given features $X_1, X_2, \ldots, X_n$
- Given assumption about feature distribution
- Predict a label Y
- And give level of [un]certainty in prediction

Naïve Bayes: classification example [2 classes]

ROMol	MW	AMW	Sv	Se	Sp	H%	C%	N%		Class
	565.75	6.98	49.03	81.24	51.69	49.38	39.51	3.70	…	1
…										
	521.75	6.60	46.69	78.56	49.93	51.90	37.97	5.06	…	0

$\underbrace{\hspace{5em}}_{X_1,\ldots,X_n \in \mathbb{R} \text{ (features)}}$

$\underbrace{\hspace{2em}}_{Y \in \{1,0\}}$ **(predict whether molecule active or inactive)**

Naïve Bayes: Why "Bayes"?

Use Bayes Rule:

$$P(Y|X_1,\ldots,X_n) = \frac{\overbrace{P(X_1,\ldots,X_n|Y)}^{\text{Likelihood}}\overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(X_1,\ldots,X_n)}_{\text{Normalization Constant}}}$$

- Why did this help?
  Well, we think that we might be able to specify how features are "generated" by the class label

Naïve Bayes: Why "Bayes"? [cont.]

For our example

$$P(Y=0|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=0)P(Y=0)}{P(X_1,\ldots,X_n)} = \frac{P(X_1,\ldots,X_n|Y=0)P(Y=0)}{P(X_1,\ldots,X_n|Y=1)P(Y=1)+P(X_1,\ldots,X_n|Y=0)P(Y=0)}$$

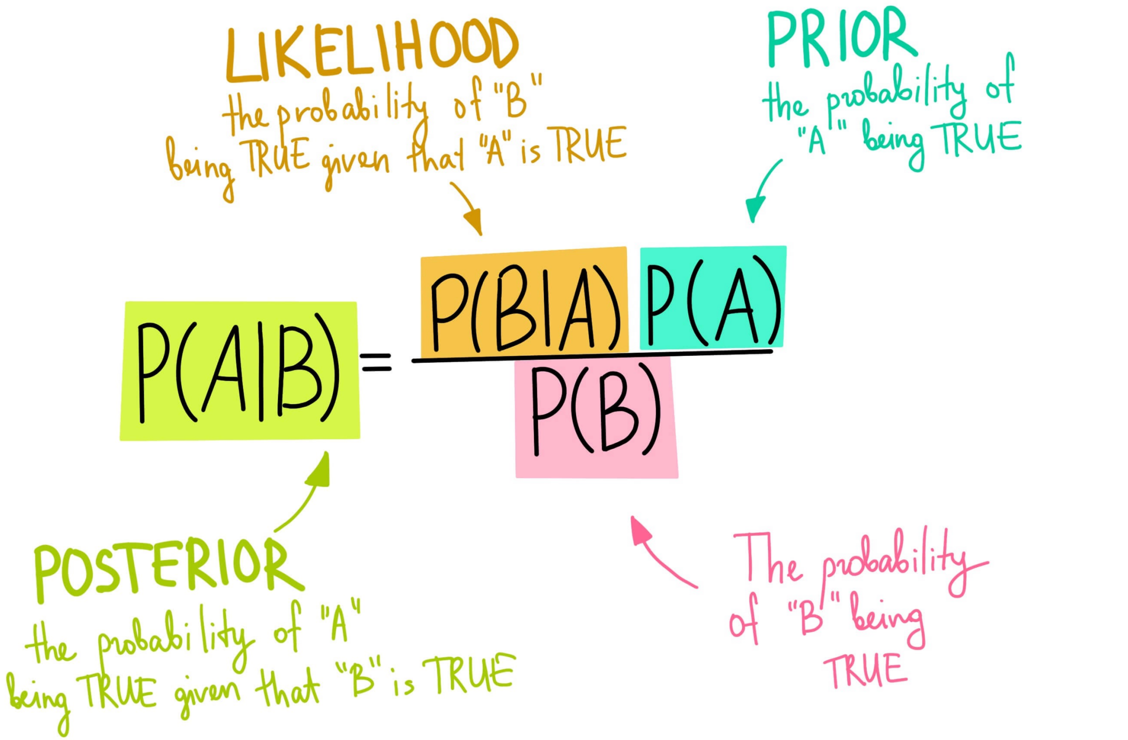
$$P(Y=1|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=1)P(Y=1)}{P(X_1,\ldots,X_n)} = \frac{P(X_1,\ldots,X_n|Y=1)P(Y=1)}{P(X_1,\ldots,X_n|Y=1)P(Y=1)+P(X_1,\ldots,X_n|Y=0)P(Y=0)}$$

To classify, we'll simply compute these two probabilities and predict based on which one is greater

Naïve Bayes: Why “Naïve”?

For the Bayes classifier, we need to “learn” two functions, the **likelihood** and the **prior**

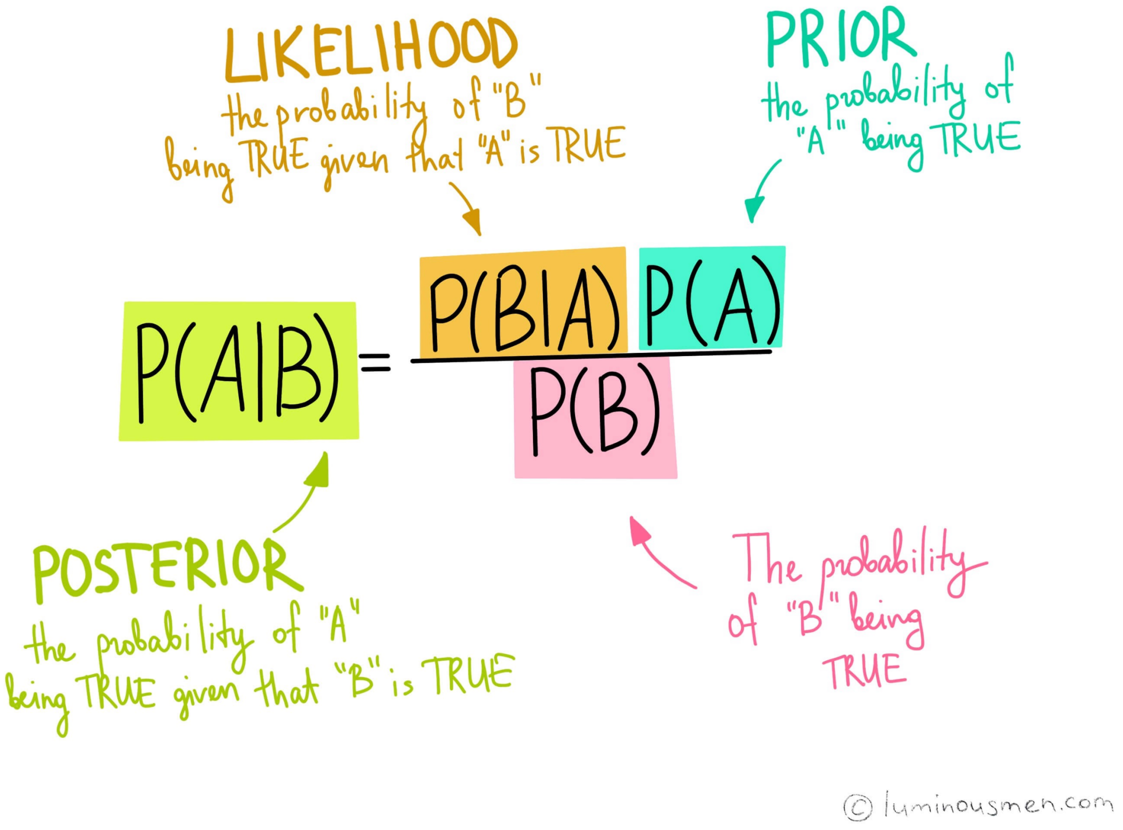
But:



Naïve Bayes: Why “Naïve”?

For the Bayes classifier, we need to “learn” two functions, the **likelihood** and the **prior**

But: How many parameters are required to specify the prior for our example?



Naïve Bayes: Why "Naïve"? [cont.]

For the Bayes classifier, we need to "learn" two functions, the **likelihood** and the **prior**

Likelihood

Prior

$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

Normalization Constant

But: How many parameters are required to specify the prior for our example?

# of parameters for modeling $P(X_1,\ldots,X_n|Y)$: $\mathbf{2(2^n-1)}$

Naïve Bayes: Why "Naïve"? [cont. 2]

The *Naïve Bayes Assumption*:

Assume that all features are *conditionally independent* **given the class label Y**

$$P(X_1, \ldots, X_n | Y) = \prod_{i=1}^{n} P(X_i | Y)$$

Likelihood

Prior

$$P(Y | X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n | Y) P(Y)}{P(X_1, \ldots, X_n)}$$

Normalization Constant

Naïve Bayes: Why "Naïve"? [cont. 2]

The *Naïve Bayes Assumption*:

Assume that all features are *conditionally independent* **given the class label Y**

$$P(X_1, \ldots, X_n | Y) = \prod_{i=1}^{n} P(X_i | Y)$$

$$P(Y|X_1, \ldots, X_n) = \frac{\overbrace{P(X_1, \ldots, X_n|Y)}^{\text{Likelihood}}\overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(X_1, \ldots, X_n)}_{\text{Normalization Constant}}}$$

But: How many parameters are required to specify the prior for our example with this assumption?

Naïve Bayes: Why "Naïve"? [cont. 2]

The *Naïve Bayes Assumption*:

Assume that all features are *conditionally independent* **given the class label Y**

$$P(X_1, \ldots, X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

$$P(Y|X_1, \ldots, X_n) = \frac{\overbrace{P(X_1, \ldots, X_n|Y)}^{\text{Likelihood}}\,\overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(X_1, \ldots, X_n)}_{\text{Normalization Constant}}}$$

But: How many parameters are required to specify the prior for our example with this assumption?

# of parameters for modeling $P(X_1|Y),\ldots,P(X_n|Y)$:**2n**

Naïve Bayes: Why "Naïve"? [cont. 2]

The *Naïve Bayes Assumption*:

Assume that all features are *conditionally independent* **given the class label Y**

$$P(X_1, \ldots, X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

$$P(Y|X_1, \ldots, X_n) = \frac{\overbrace{P(X_1, \ldots, X_n|Y)}^{\text{Likelihood}}\overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(X_1, \ldots, X_n)}_{\text{Normalization Constant}}}$$

But: How many parameters are required to specify the prior for our example with this assumption?

\# of parameters for modeling $P(X_1|Y), \ldots, P(X_n|Y)$: **$2n \lll 2(2^n-1)$**

Naïve Bayes: Why "Naïve"? [And why **2n**]

The *Naïve Bayes Assumption*:

Assume that all features are *conditionally independent* **given the class label Y**

$$P(X_1, \ldots, X_n | Y) = \prod_{i=1}^{n} P(X_i | Y)$$

Gaussian naive Bayes

assumption: features distributed normally :

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

But: How many parameters are required to specify the prior for our example with this assumption?

# of parameters for modeling $P(X_1|Y),\ldots,P(X_n|Y)$:**2n**

Naïve Bayes: Training

Training in Naïve Bayes is **easy :**

we have to select **feature distribution** (aka likelihood function)
[usually Gaussian distribution]

and **estimate parameters** of model of feature distribution

This corresponds to Maximum Likelihood estimation of model parameters.

Naïve Bayes: Training

Training in Naïve Bayes is **easy :**

we have to select **feature distribution** (aka likelihood function)
[usually Gaussian distribution]

and **estimate parameters** of model of feature distribution

This corresponds to Maximum Likelihood estimation of model parameters.

Is there other than Gaussian Naïve Bayes?

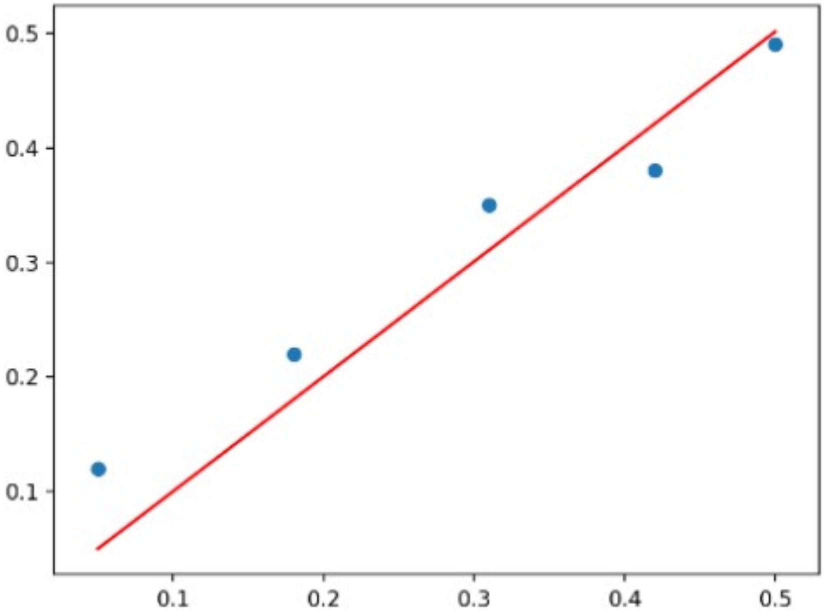
Sure, as a likelihood function, we could select different distributions.

In Scikit-learn there is implementation for Gaussian, Multinomial, Bernoulli etc.

Topics

- Probability recap
- Naïve Bayes
- **Gaussian process**
- Bayesian hyperparameter optimization

Gaussian Process: ML perspective



Gaussian Process: Definition

Continuous stochastic process — random functions — a set of random variables indexed by a continuous variable: $f(x)$

Set of 'inputs' $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$; corresponding set of random function variables $\mathbf{f} = \{f_1, f_2, \ldots, f_N\}$

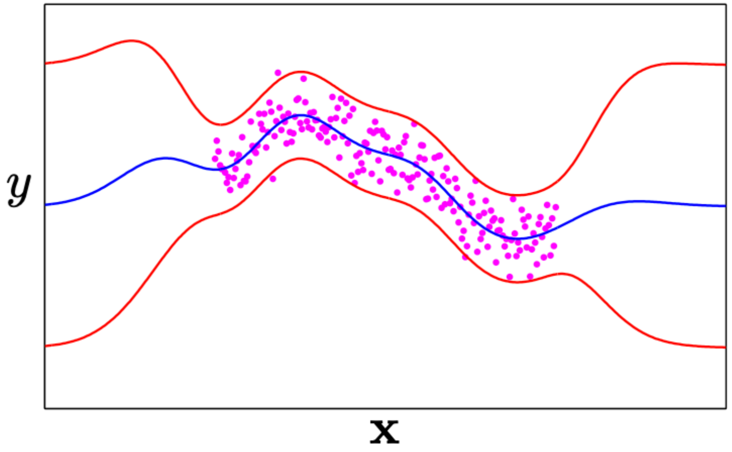
GP: Any set of function variables $\{f_n\}_{n=1}^{N}$ has joint (zero mean) Gaussian distribution:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

Gaussian Process: ML perspective

Consider the problem of nonlinear regression:

You want to learn a function f with error bars from data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$



A Gaussian process is a prior over functions $p(f)$ which can be used for Bayesian regression:

$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

Gaussian Process: vs Gaussian distributions

Gaussian distributions
$\mathcal{N}(\mu, \Sigma)$

Distribution over vectors.
Fully specified by a mean and covariance.
The position of the random variable in the vector plays the role of the index.

Gaussian processes
$\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$

Distribution over functions.
Fully specified by a mean function and covariance function.
The argument of the random function plays the role of the index.

Gaussian Process: covariance matrix

Gaussian processes are merely based on the good old Gaussian

$$\mathcal{N}\left(\mathbf{x} \middle| \boldsymbol{\mu}, \mathbf{K}\right) = \frac{1}{\sqrt{|2\pi \mathbf{K}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right]$$

Covariance matrix or kernel matrix

Covariance matrix constructed from *covariance function*:

$$\mathbf{K}_{ij} = K(x_i, x_j)$$

Covariance function characterizes correlations between different points in the process:

$$K(x, x') = \mathcal{E}[f(x)f(x')]$$

Gaussian Process: covariance matrix

Gaussian processes are merely based on the good old Gaussian

$$\mathcal{N}\left(\mathbf{x}\,\middle|\,\boldsymbol{\mu},\mathbf{K}\right)=\frac{1}{\sqrt{|2\pi\mathbf{K}|}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{K}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

Covariance matrix or kernel matrix

Covariance matrix constructed from *covariance function*:

$$\mathbf{K}_{ij}=K(x_i,x_j)$$

Covariance function characterizes correlations between different points in the process:

$$K(x,x')=\mathcal{E}[f(x)f(x')]$$

Gaussian Process: kernel examples

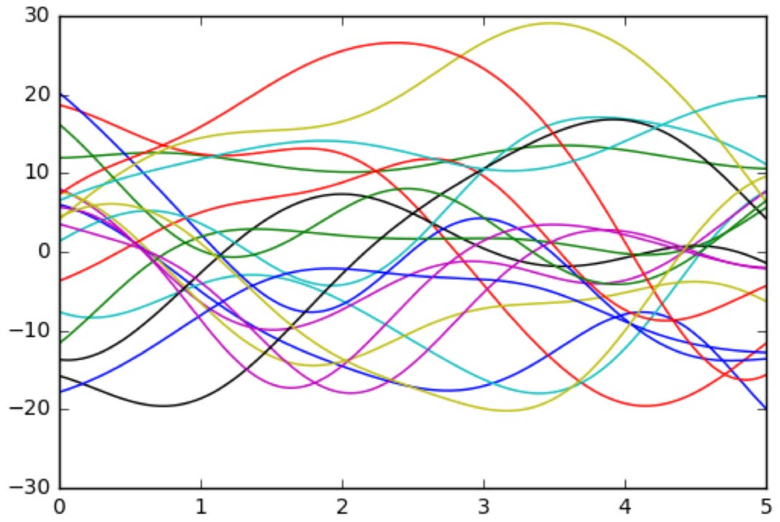
Kernels -- are a crucial ingredient of GPs which determine the shape of prior and posterior of the GP.
They **encode the assumptions** on the function being learned by defining the "similarity" of two datapoints combined with the assumption that similar datapoints should have similar target values.

Squared exponential (SE), RBF

$$K(x, x') = \sigma_0{}^2 \exp\left[-\frac{1}{2}\left(\frac{x - x'}{\lambda}\right)^2\right]$$

Intuition: function variables close in input space are highly correlated, whilst those far away are uncorrelated

λ, σ_0 — hyperparameters. λ: lengthscale, σ_0: amplitude



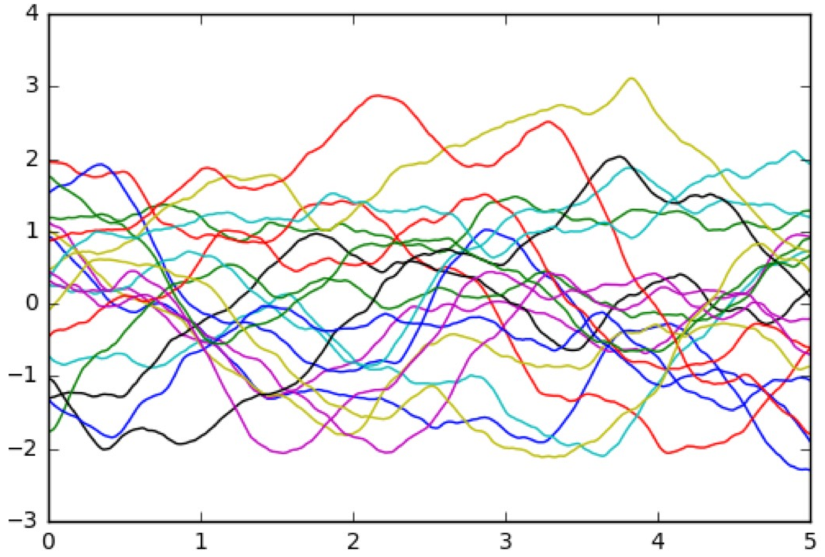
Gaussian Process: kernel examples

Kernels -- are a crucial ingredient of GPs which determine the shape of prior and posterior of the GP.
They **encode the assumptions** on the function being learned by defining the “similarity” of two datapoints combined with the assumption that similar datapoints should have similar target values.

Mat́ern class [usually ν = 1/2 or 3/2]

$$K(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|x - x'|}{\lambda} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}|x - x'|}{\lambda} \right)$$

where K_ν is a modified Bessel function.

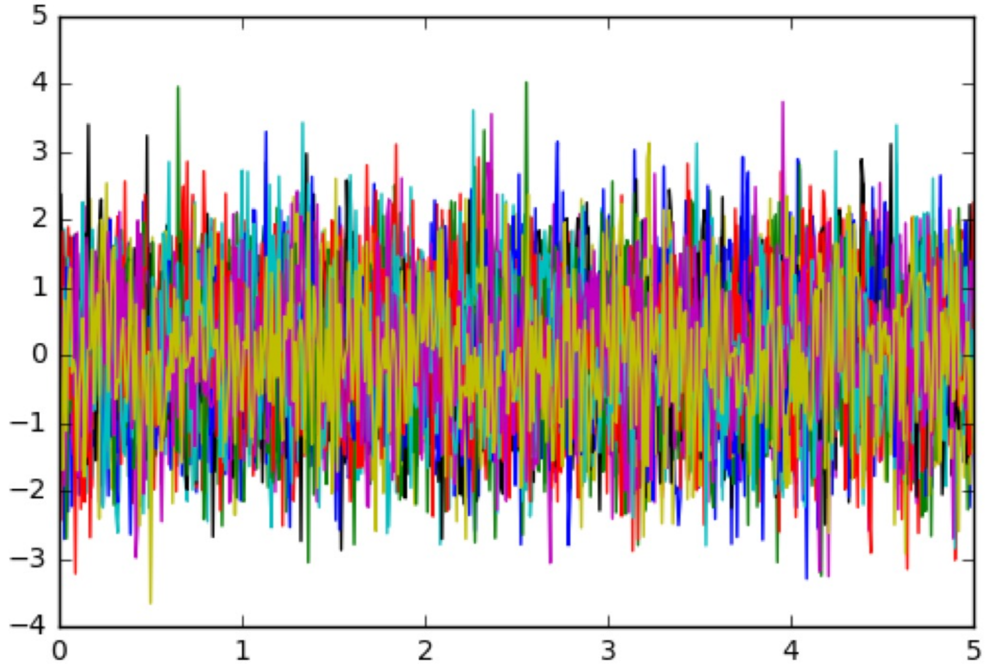


Gaussian Process: kernel examples

Kernels -- are a crucial ingredient of GPs which determine the shape of prior and posterior of the GP.
They **encode the assumptions** on the function being learned by defining the “similarity” of two datapoints combined with the assumption that similar datapoints should have similar target values.

White noise

$$k(x, x') = \begin{cases} 1 \text{ if } x = x' \\ 0 \text{ otherwise} \end{cases}$$



Gaussian Process: most standard kernel

Kernels -- are a crucial ingredient of GPs which determine the shape of prior and posterior of the GP.
They **encode the assumptions** on the function being learned by defining the “similarity” of two datapoints combined with the assumption that similar datapoints should have similar target values.

We can make new kernels by summing or multiplying kernels

Most default kernel K=RBF+White_noise

Gaussian Process Regression: training

$\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x'}))$

I) Select mean function *m*(x) [usually constant]
II) Select kernel function *k*(x,x’) [usually RBF + White_nise]

Gaussian Process Regression: training

$\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x'}))$

I) Select mean function *m*(x) [usually constant]
II) Select kernel function *k*(x,x') [usually RBF + White_nise]

$$k(x, x') = \sigma_y^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right) + \sigma_n^2 \delta_{ii'}$$

Gaussian Process Regression: training

$\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x'}))$

I) Select mean function *m*(x) [usually constant]
II) Select kernel function *k*(x,x’) [usually RBF + White_nise]

$$k(x, x') = \sigma_y^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right) + \sigma_n^2 \delta_{ii'}$$

III) Find parameters [fit GP to data]

Gaussian Process: as a regression [in 2 words]

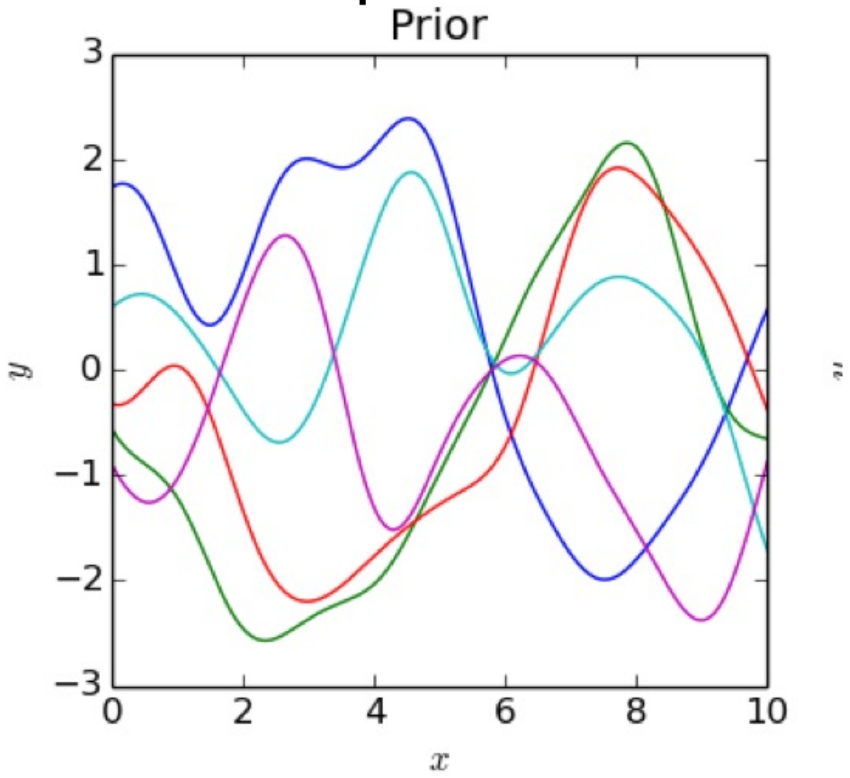
Covariance function defines the properties in the function space.

Data points "anchor" the function as specific locations.

Gaussian Process: as a regression

Covariance function defines the properties in the function space.

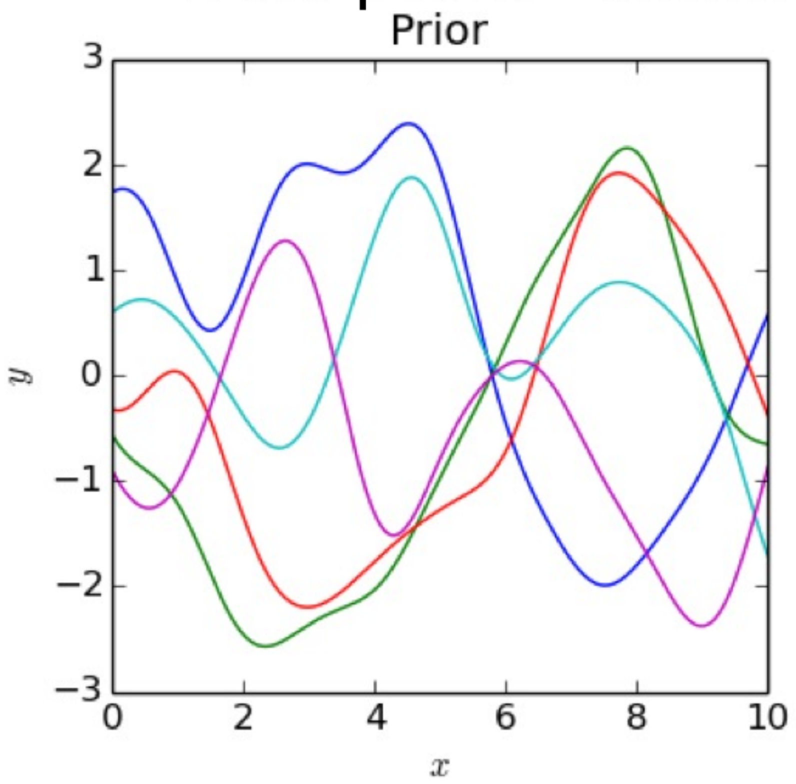
Data points "anchor" the function as specific locations.



Gaussian Process: as a regression

Covariance function defines the properties in the function space.

Data points "anchor" the function as specific locations.

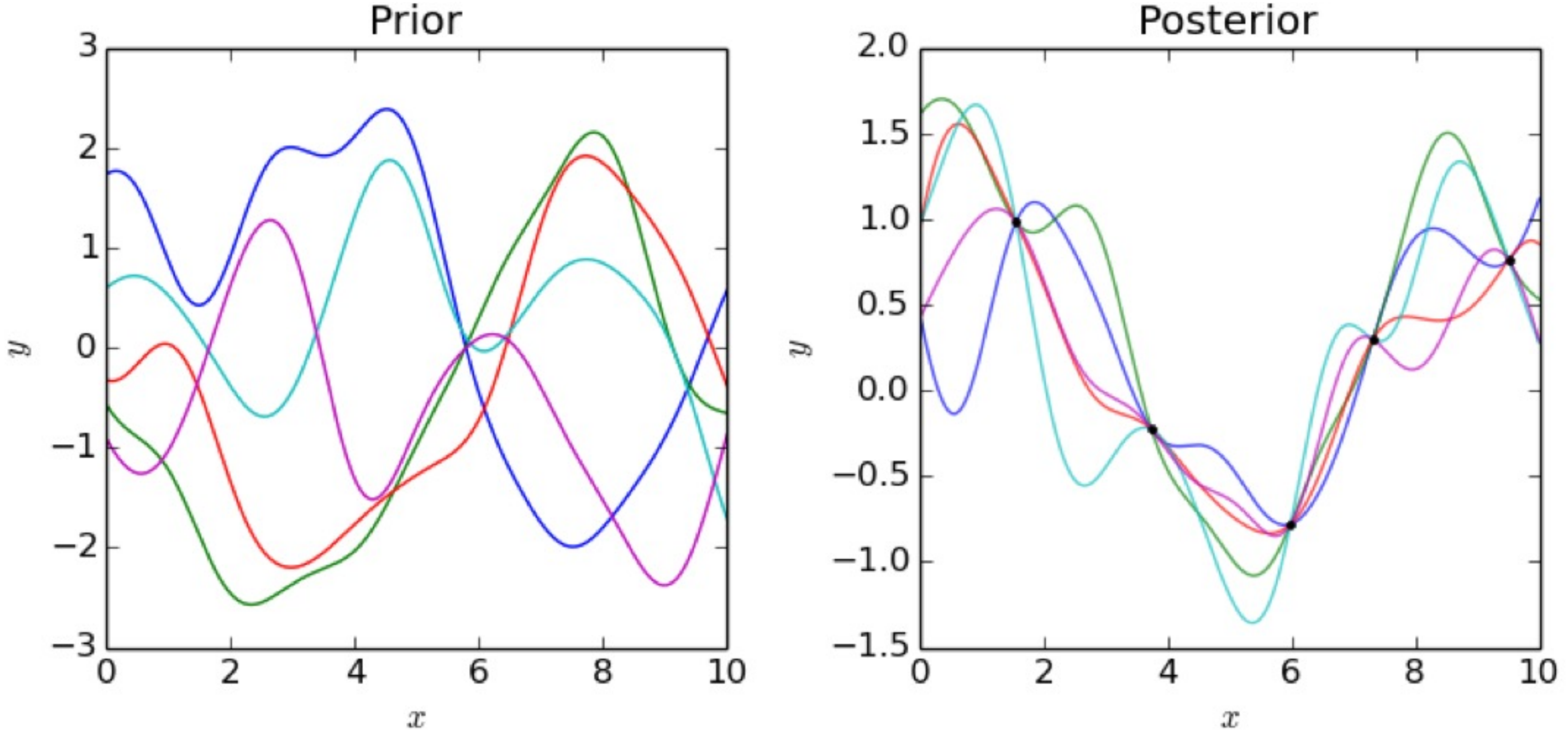


I) So we a fitting an **infinite number of functions** to the data

Gaussian Process: as a regression

Covariance function defines the properties in the function space.

Data points "anchor" the function as specific locations.

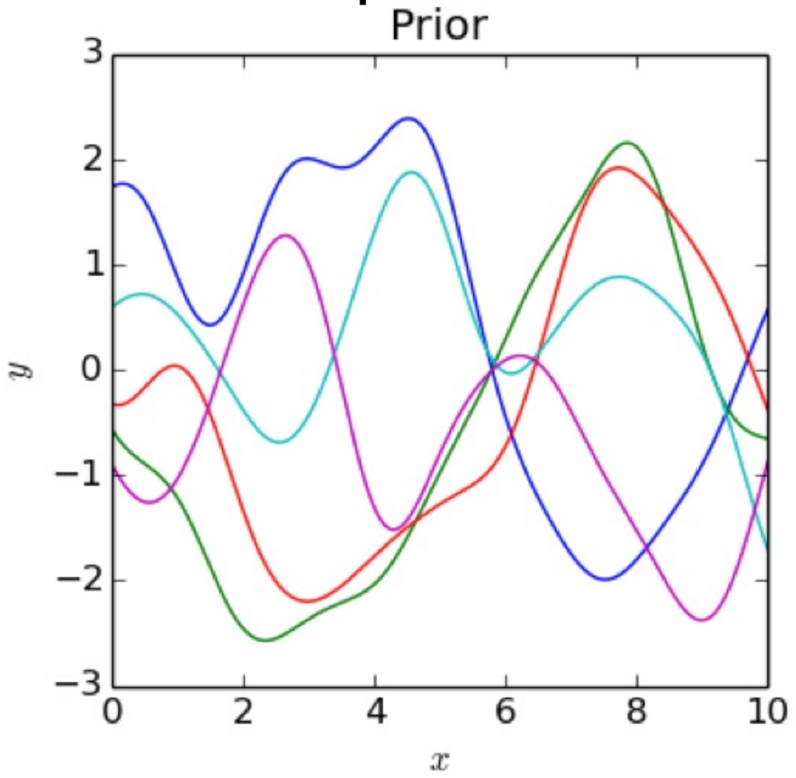


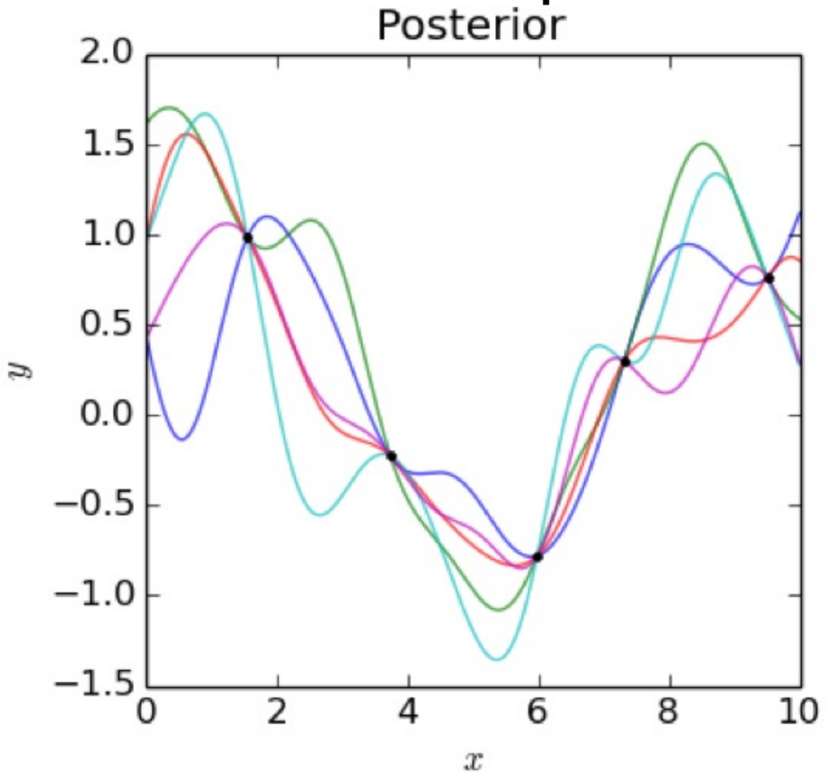
I) So we a fitting an **infinite number of functions** to the data

Gaussian Process: as a regression

Covariance function defines the properties in the function space.

Data points "anchor" the function as specific locations.





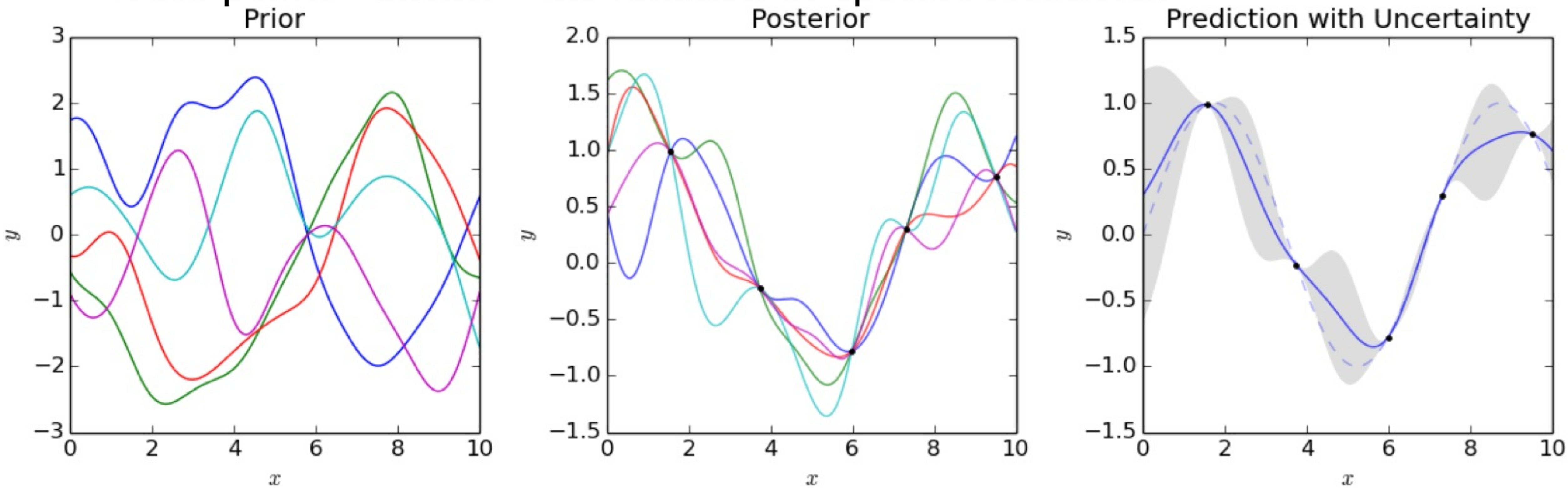
I) So we a fitting an **infinite number of functions** to the data

II) and than **sampling this functions from GP** and calculate values of those **sampled functions for particular input**

Gaussian Process: as a regression

Covariance function defines the properties in the function space.

Data points "anchor" the function as specific locations.



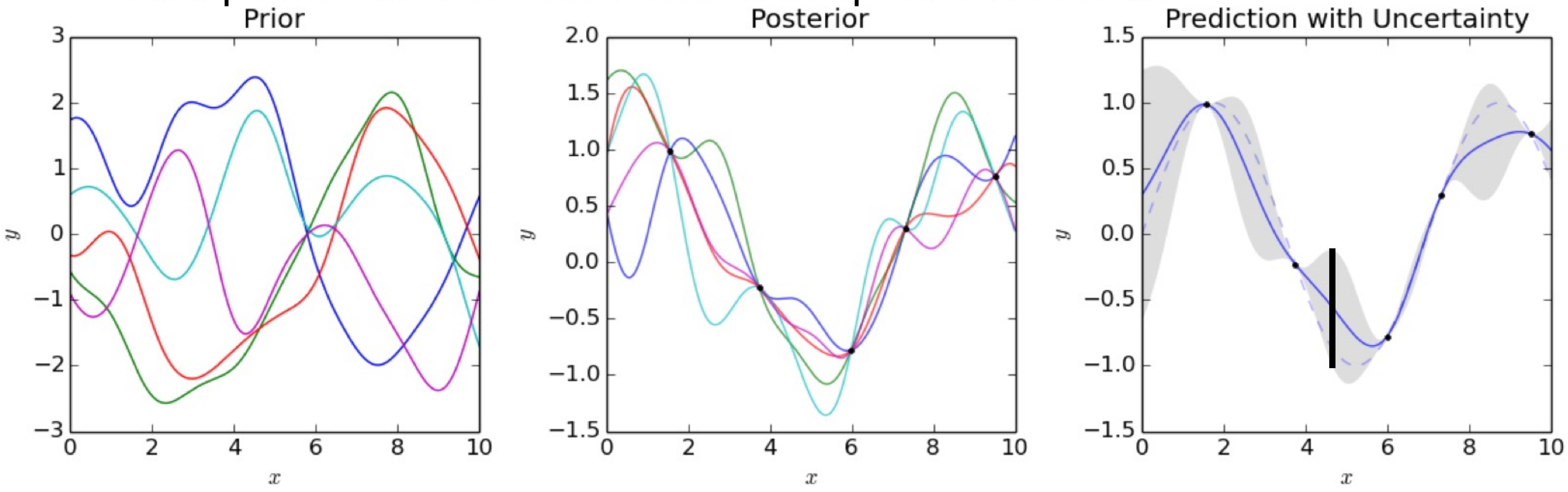
I) So we a fitting an **infinite number of functions** to the data

II) and than **sampling this functions from GP** and calculate values of those **sampled functions for particular input**

Gaussian Process: as a regression

Covariance function defines the properties in the function space.

Data points "anchor" the function as specific locations.



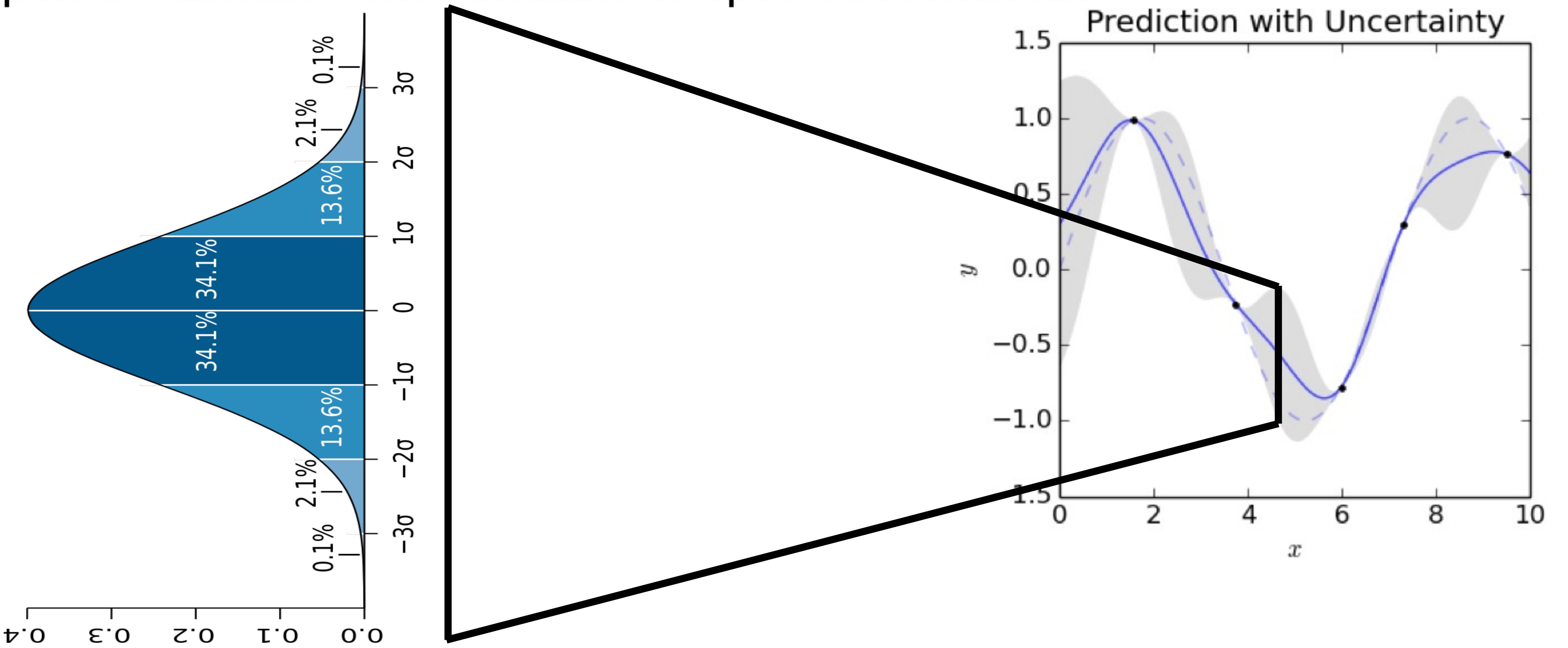
I) So we a fitting an **infinite number of functions** to the data

II) and than **sampling this functions from GP** and calculate values of those **sampled functions for particular input**

Gaussian Process: as a regression

Covariance function defines the properties in the function space.

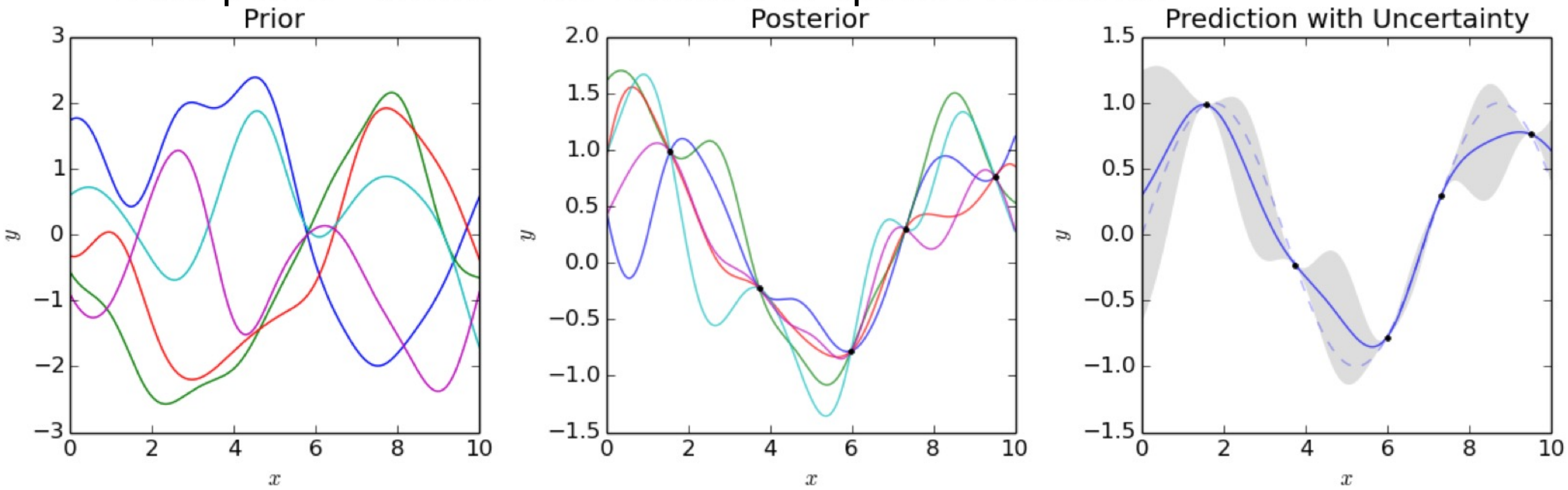
Data points "anchor" the function as specific locations.



Gaussian Process: as a regression

Covariance function defines the properties in the function space.

Data points "anchor" the function as specific locations.



I) So we a fitting an **infinite number of functions** to the data

II) and than **sampling this functions from GP** and calculate values of those **sampled functions for particular input**

Topics

- Probability recap
- Naïve Bayes
- Gaussian process
- **Bayesian hyperparameter optimization**

Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

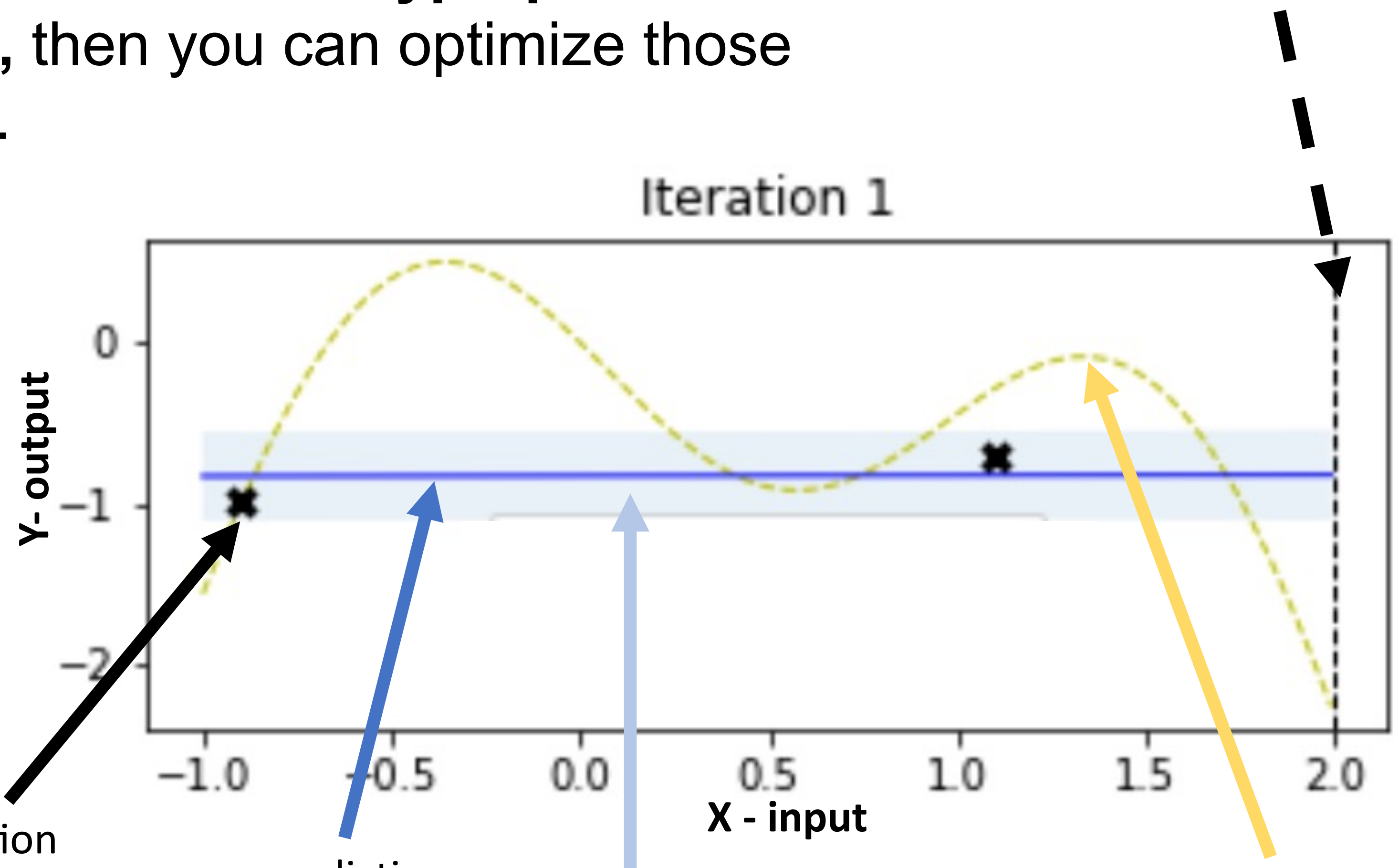
An experiment could be:

- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost

Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

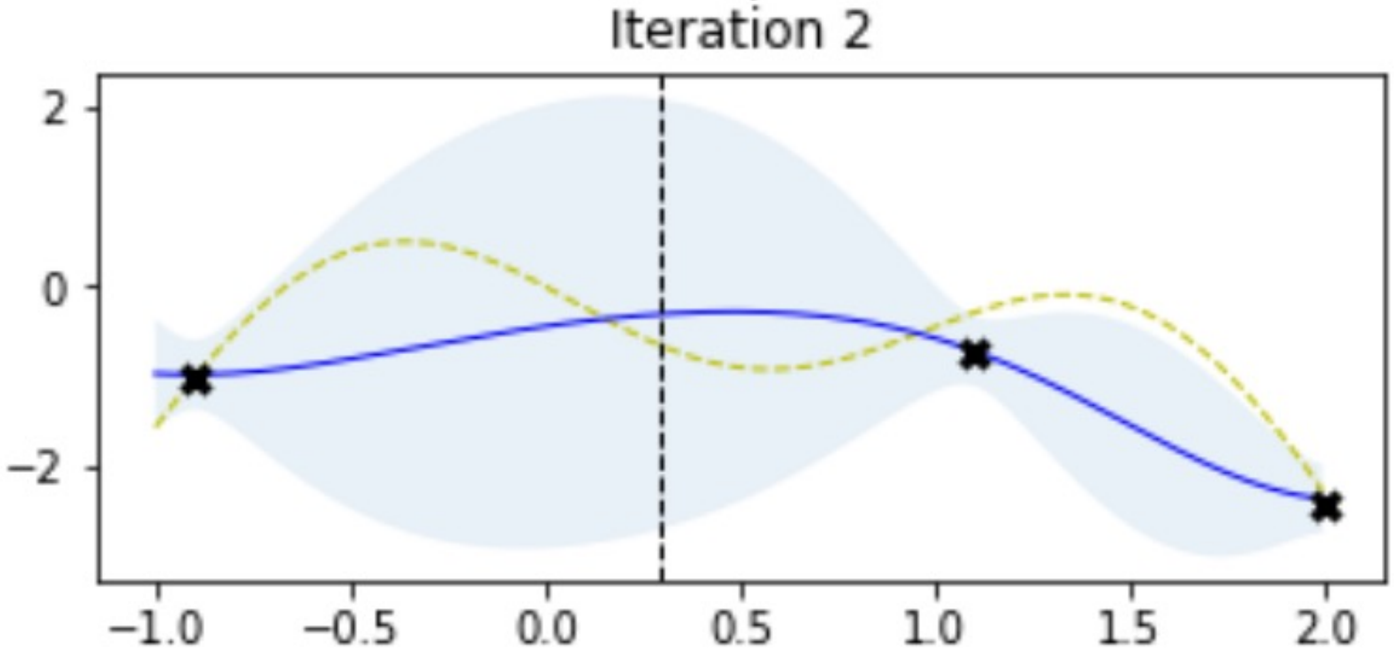
An experiment could be:
- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost



Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

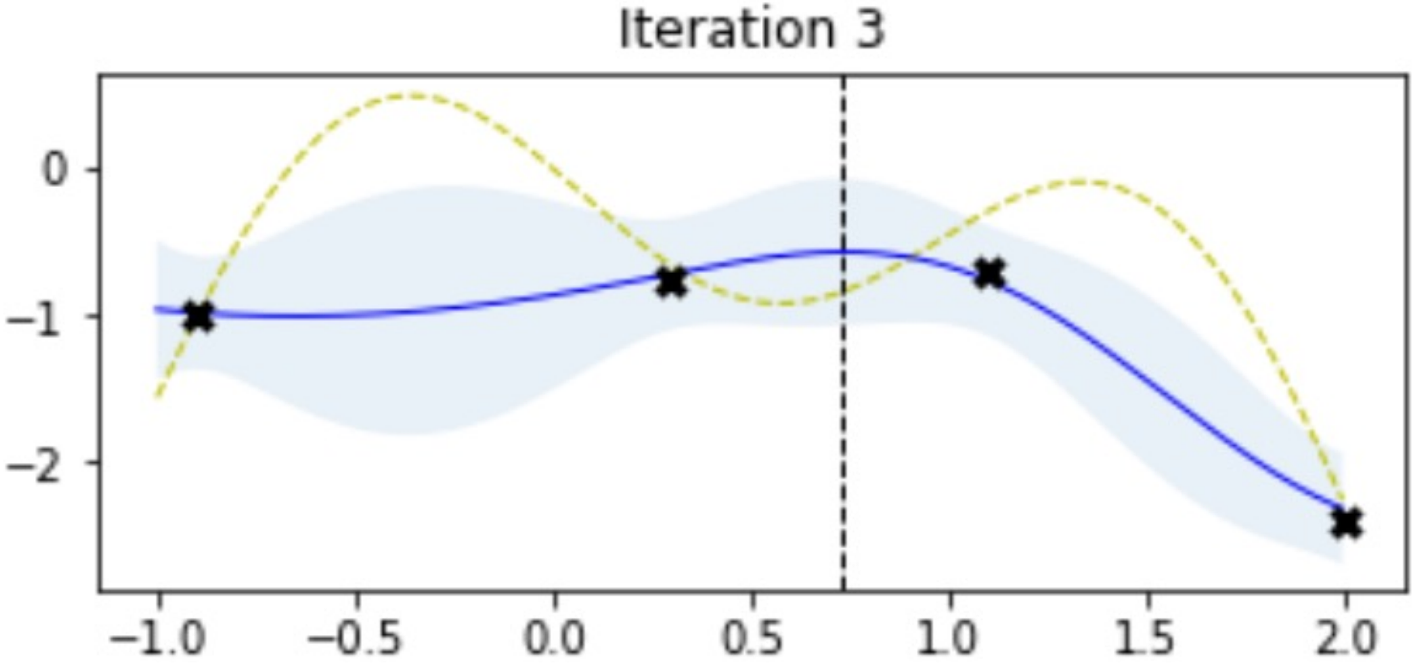
An experiment could be:
- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost



Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

An experiment could be:
- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost

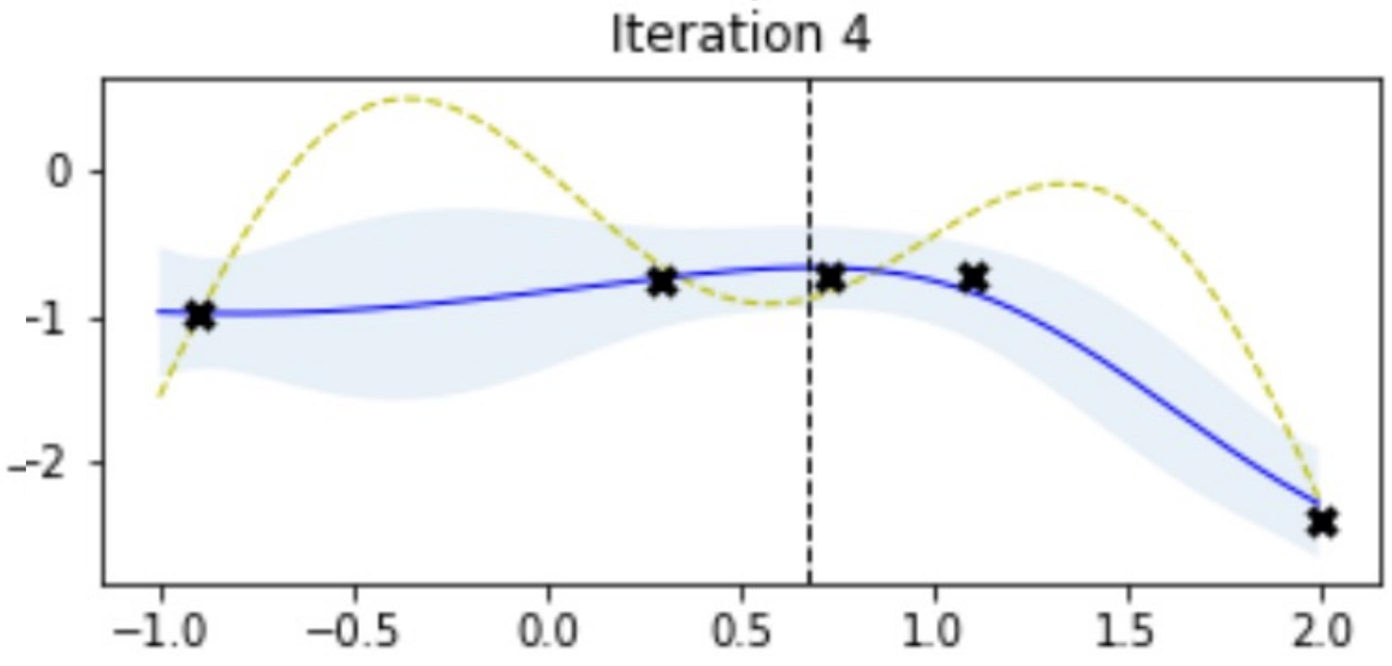


Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

An experiment could be:

- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost

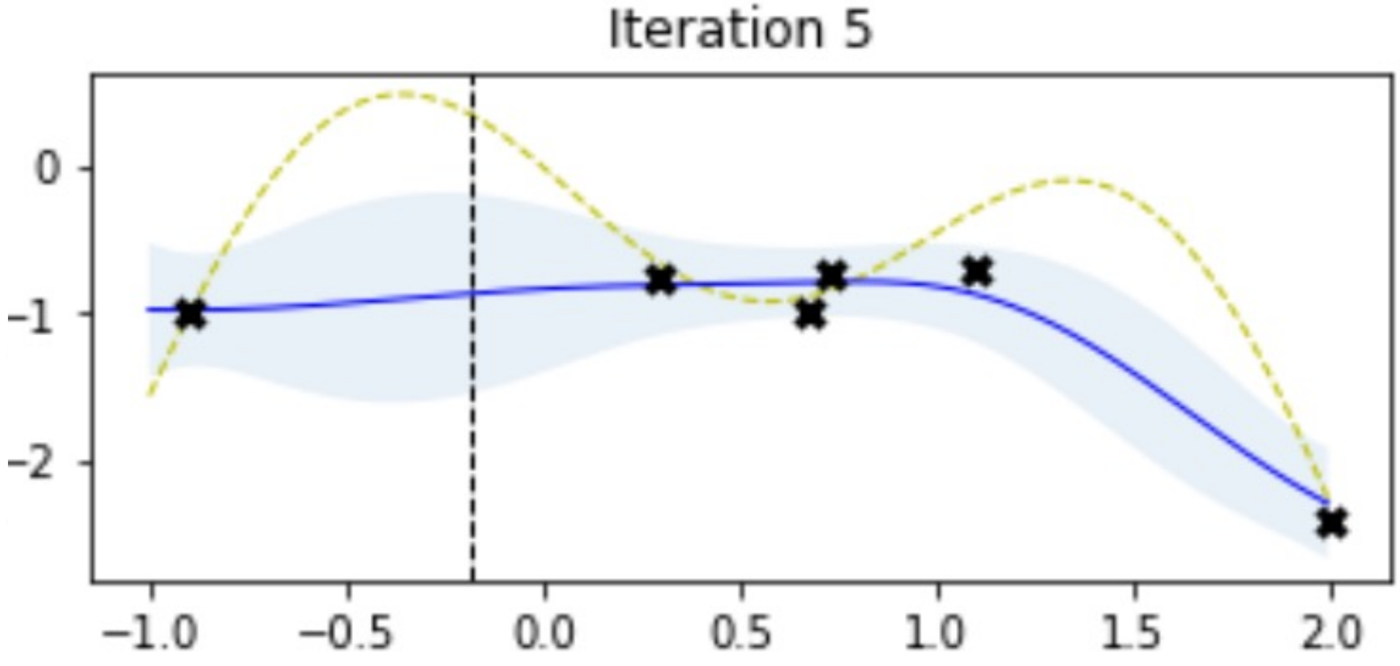


Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

An experiment could be:

- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost

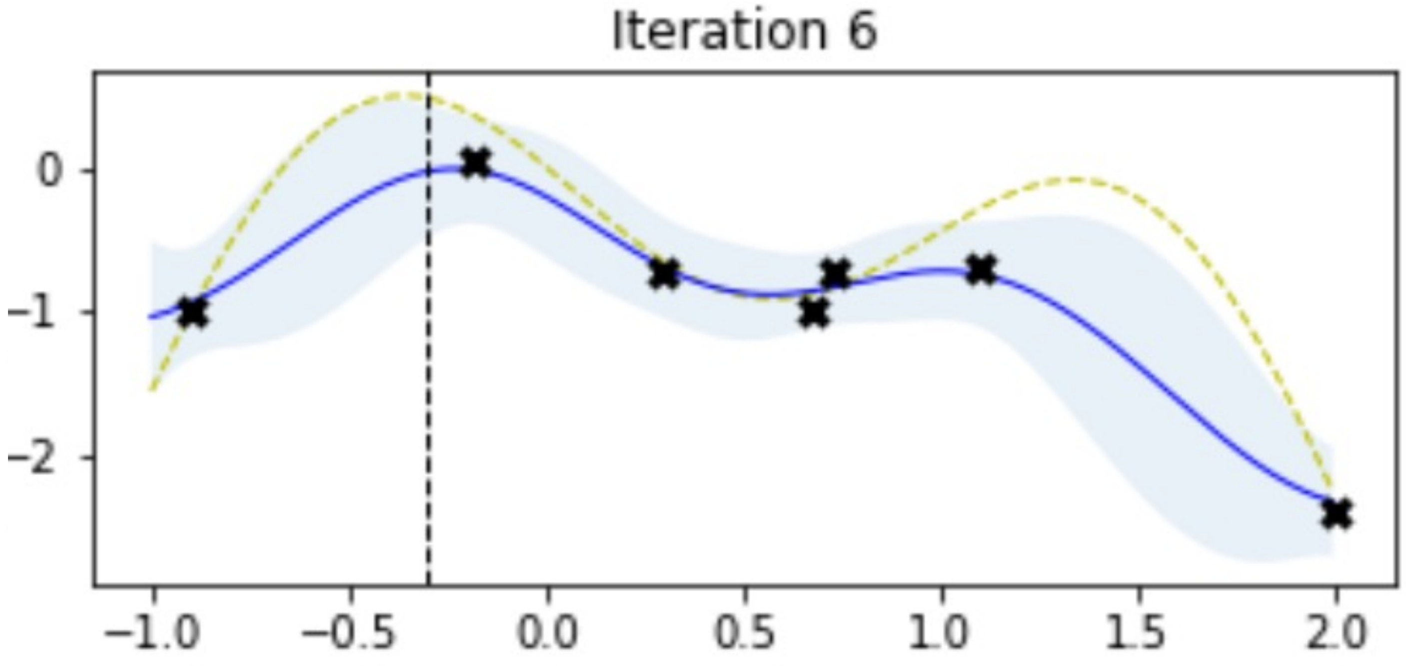


Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

An experiment could be:

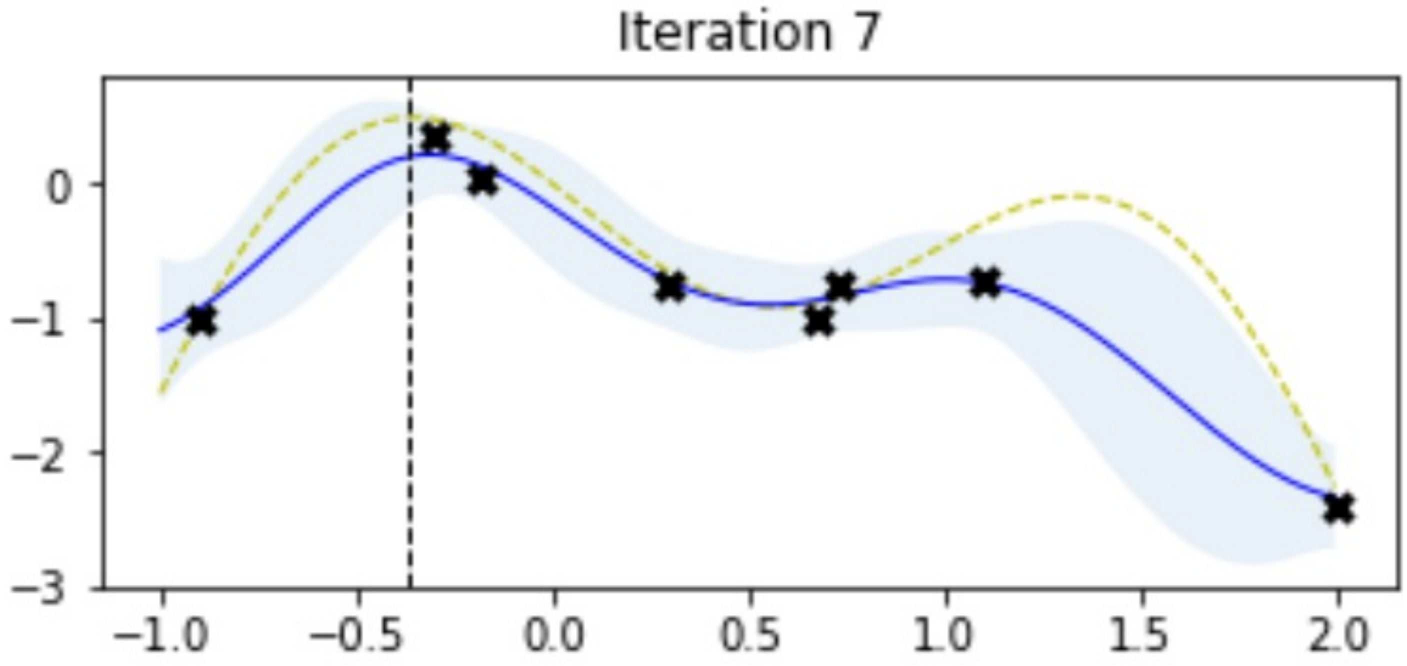
- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost



Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

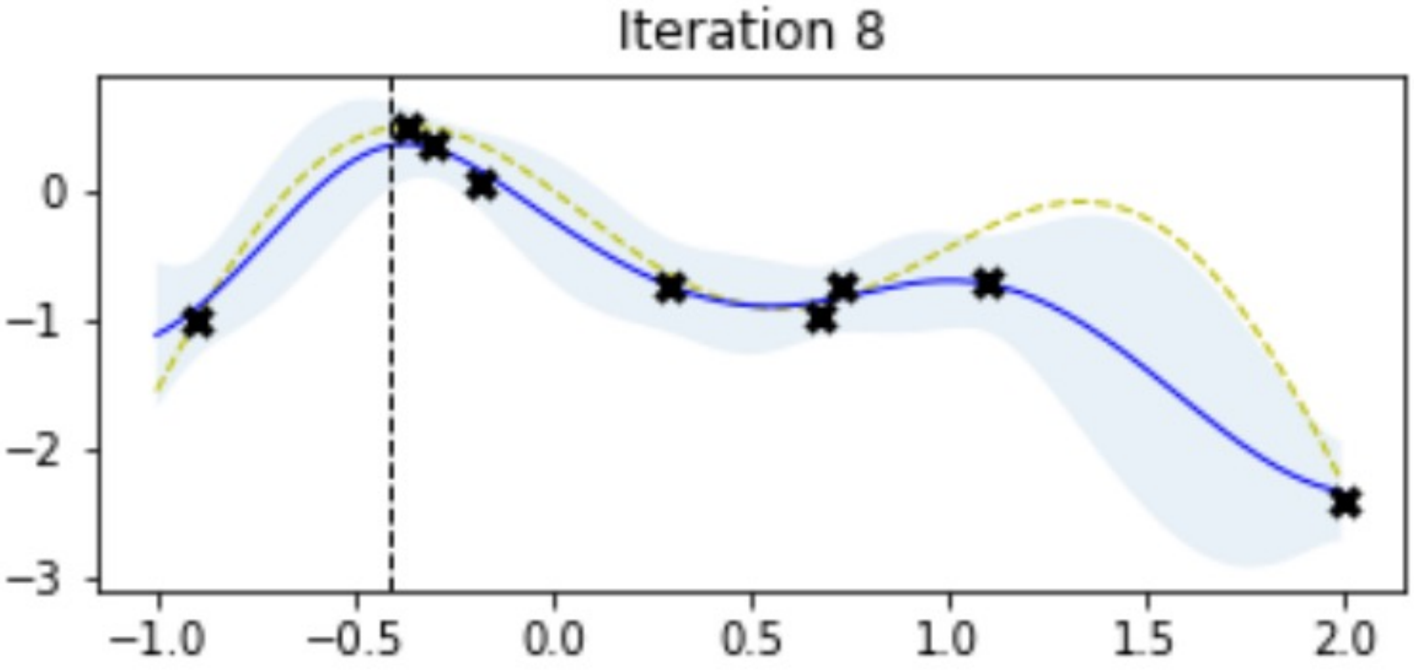
An experiment could be:
- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost



Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

An experiment could be:
- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost

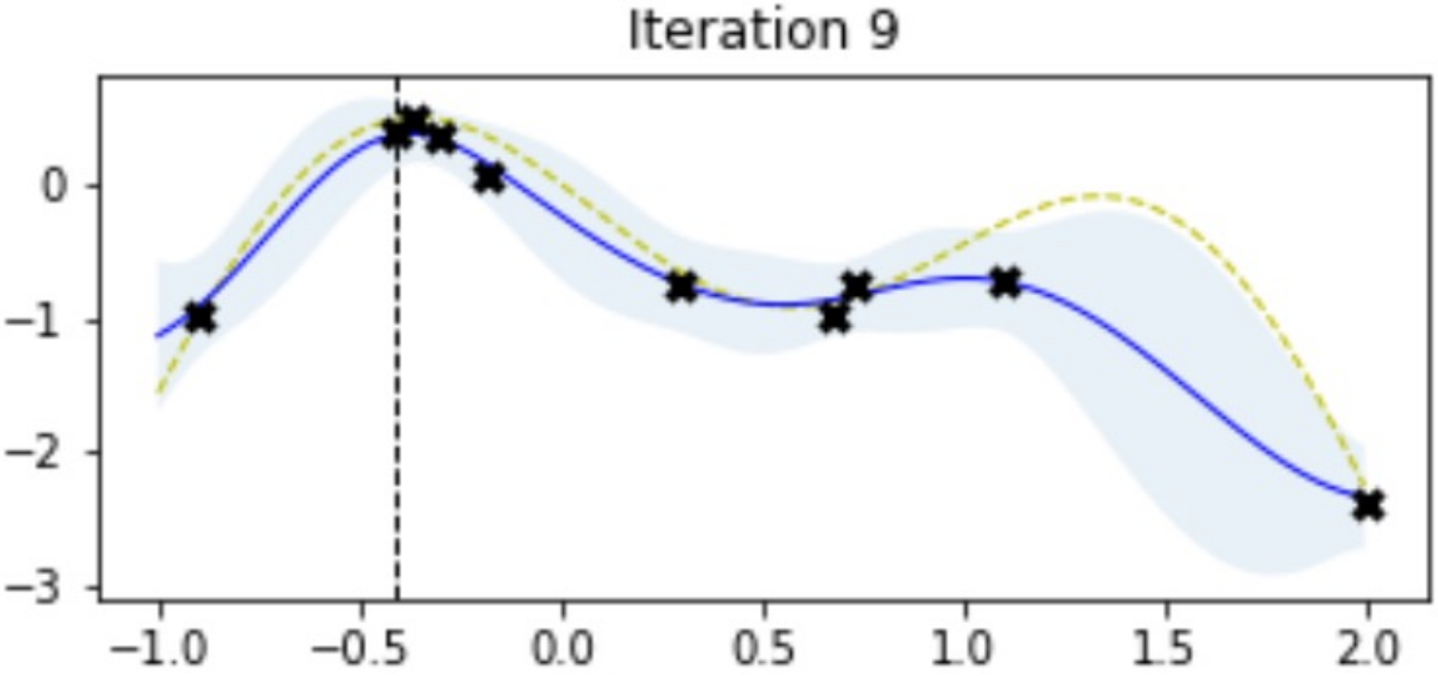


Bayesian hyperparameter optimization

If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

An experiment could be:

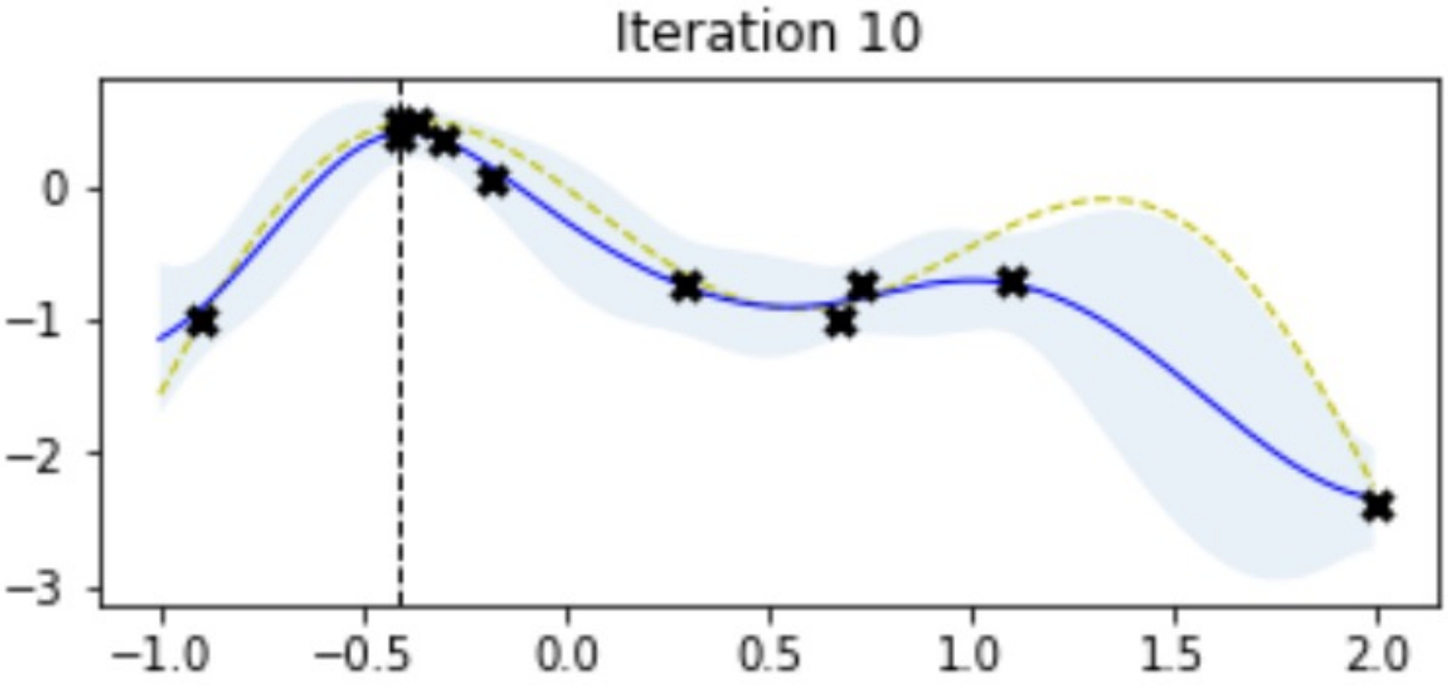
- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost



Bayesian hyperparameter optimization

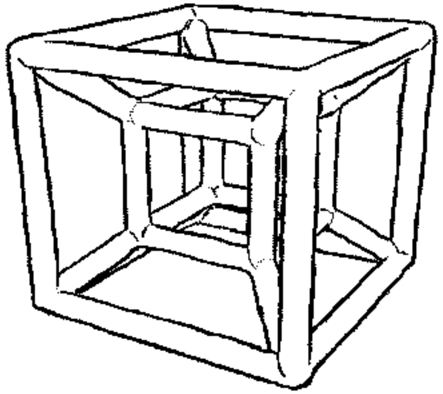
If you have an **experiment** with **set of hyperparameters** and some **objective function,** then you can optimize those hyperparameters with GP.

An experiment could be:
- Actual wet-lab experiment
- Machine learning model with big set of hyperparameters, like neural network or xgboost

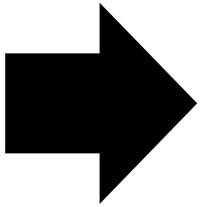


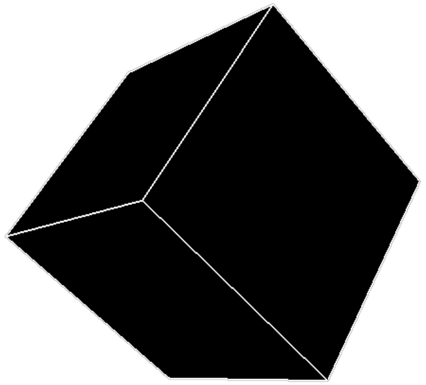
Bayesian hyperparameter optimization

Graduate student perspective



multidimensional
data

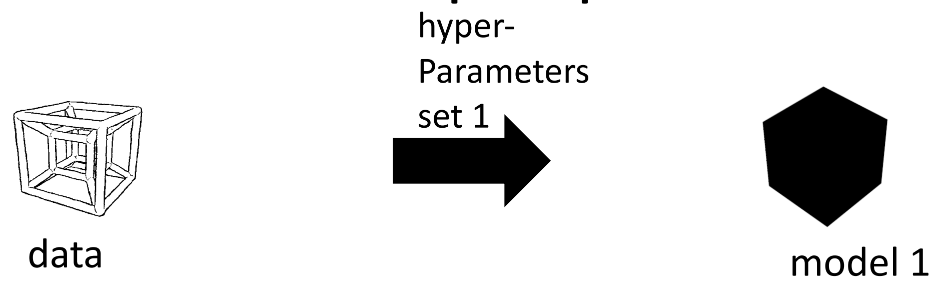




Machine learning
model

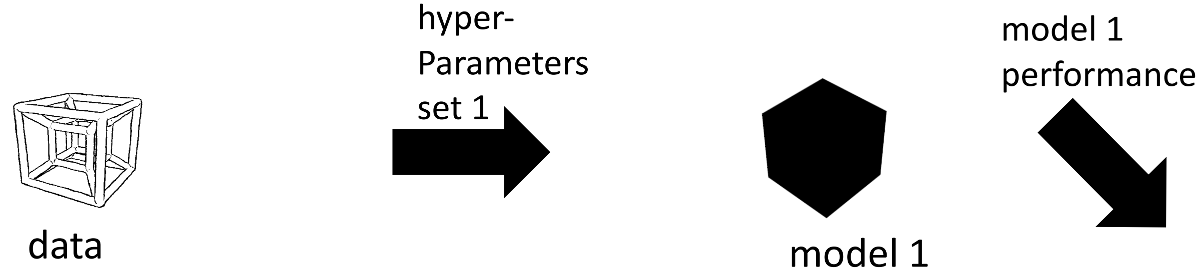
Bayesian hyperparameter optimization

Graduate student perspective



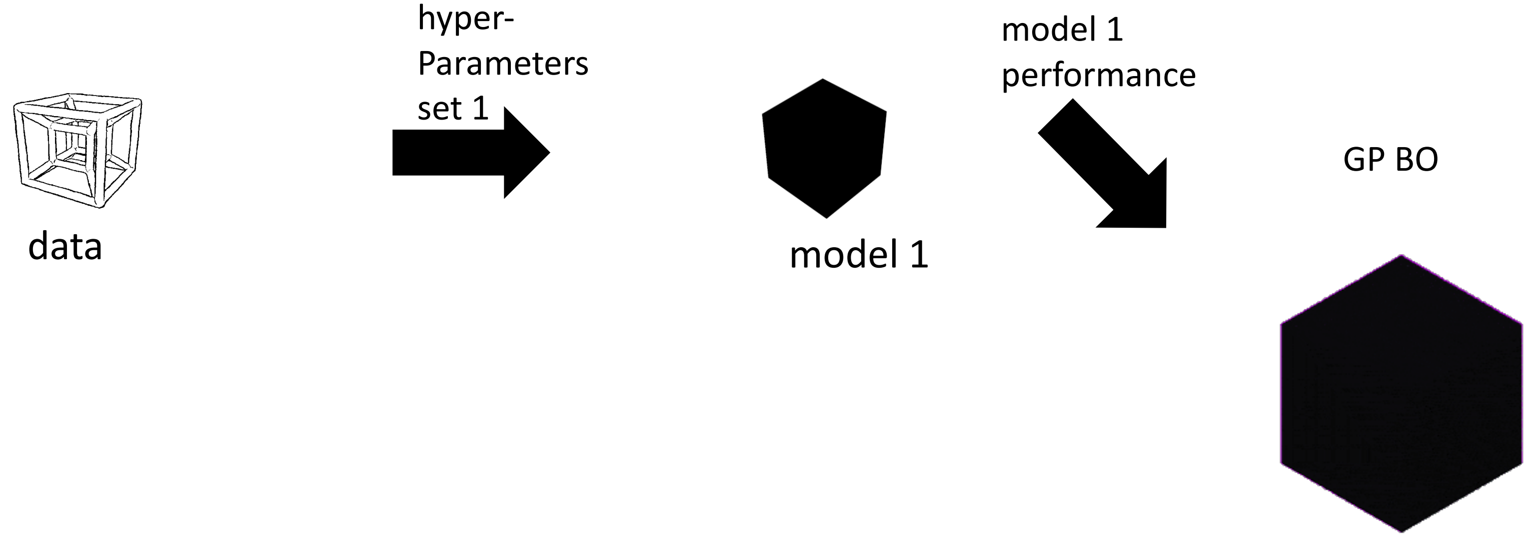
Bayesian hyperparameter optimization

Graduate student perspective



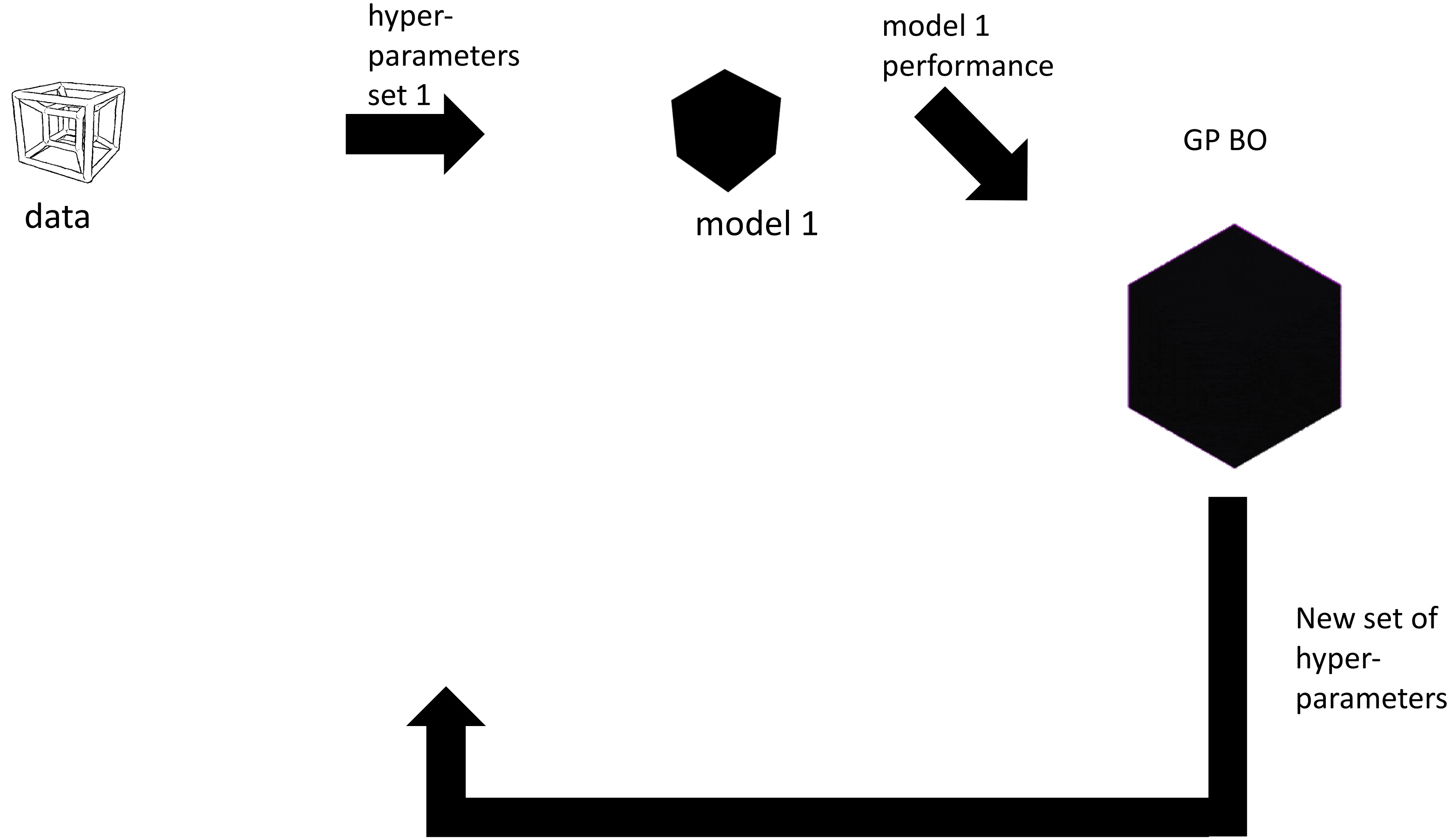
Bayesian hyperparameter optimization

Graduate student perspective



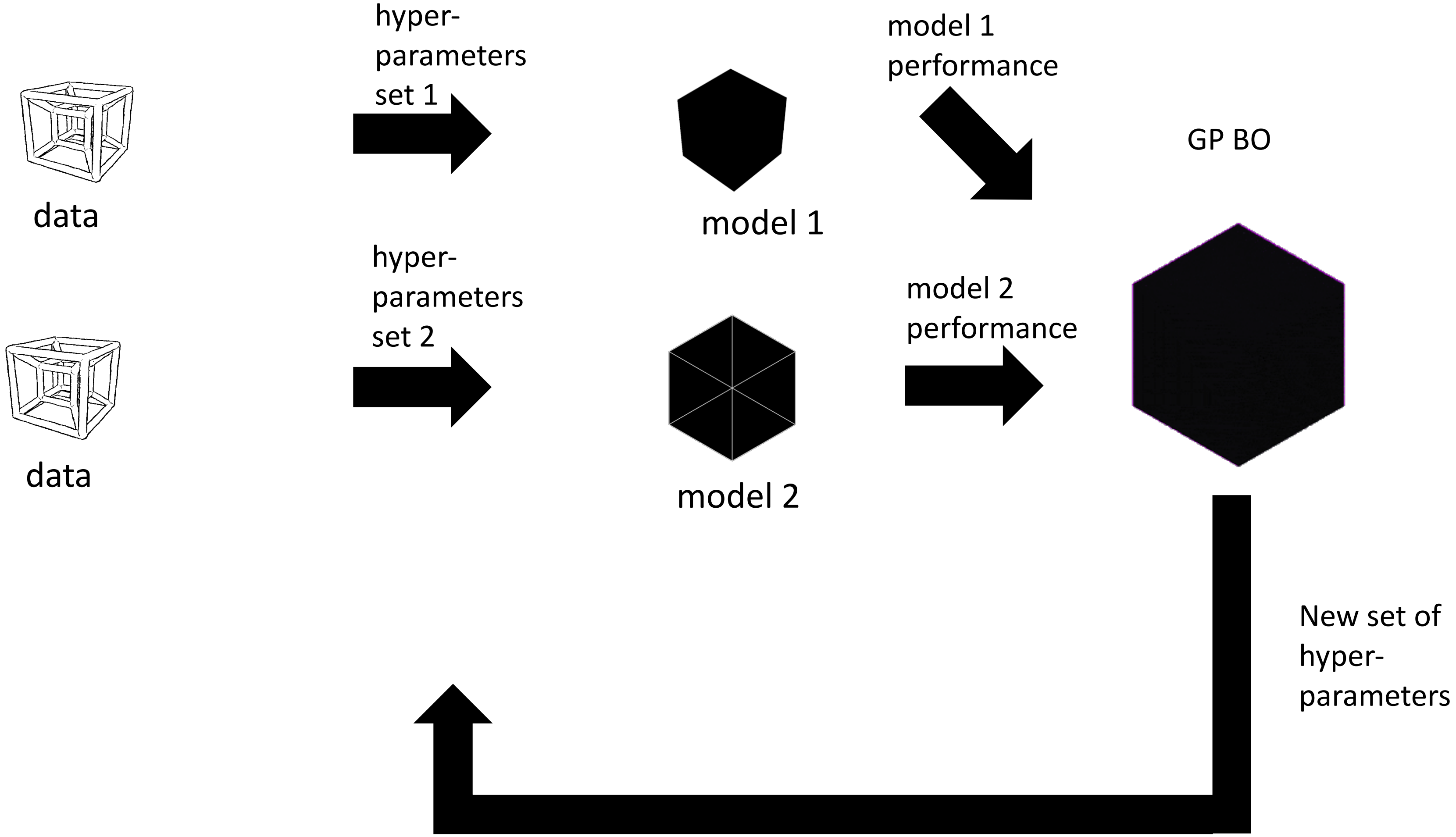
Bayesian hyperparameter optimization

Graduate student perspective



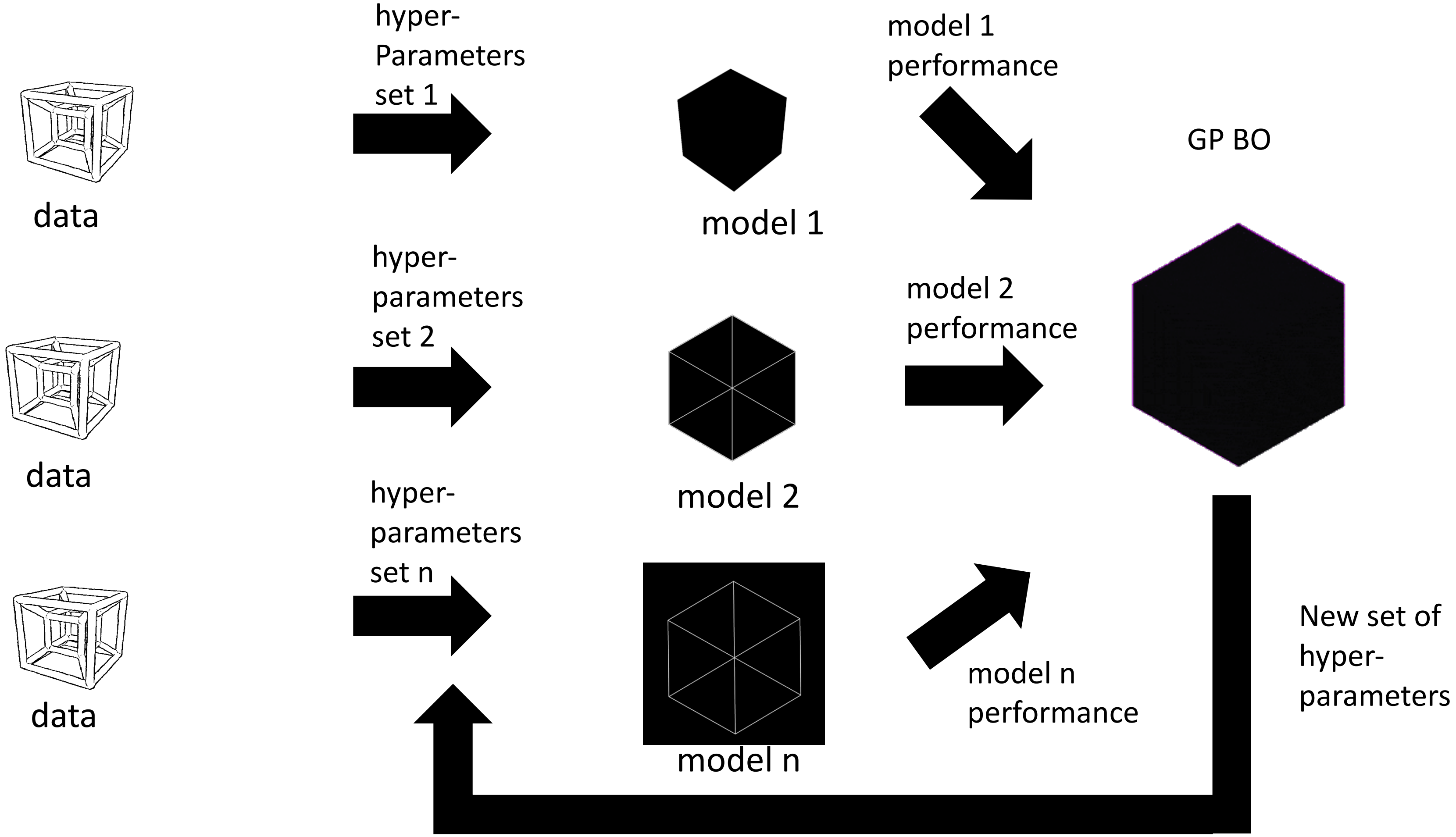
Bayesian hyperparameter optimization

Graduate student perspective



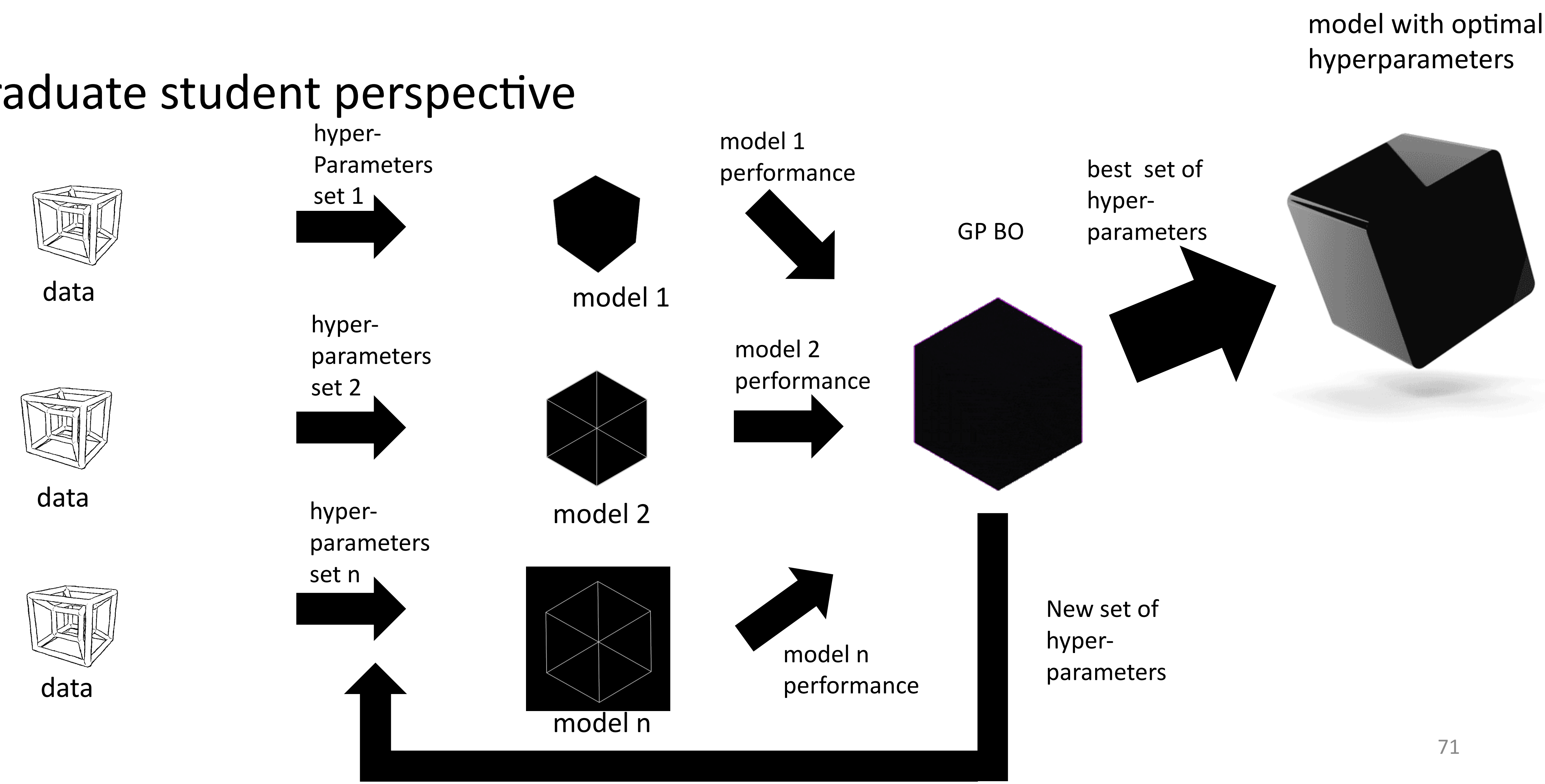
Bayesian hyperparameter optimization

Graduate student perspective



Bayesian hyperparameter optimization

Graduate student perspective



Bayesian hyperparameter optimization

Graduate student perspective