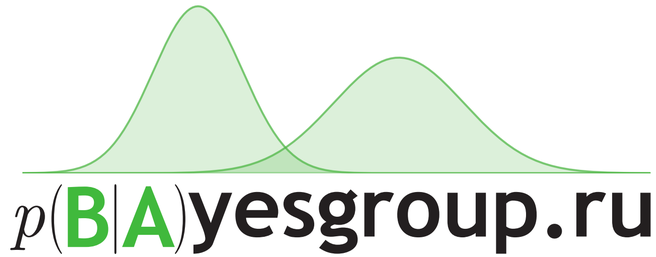
Gaussian processes

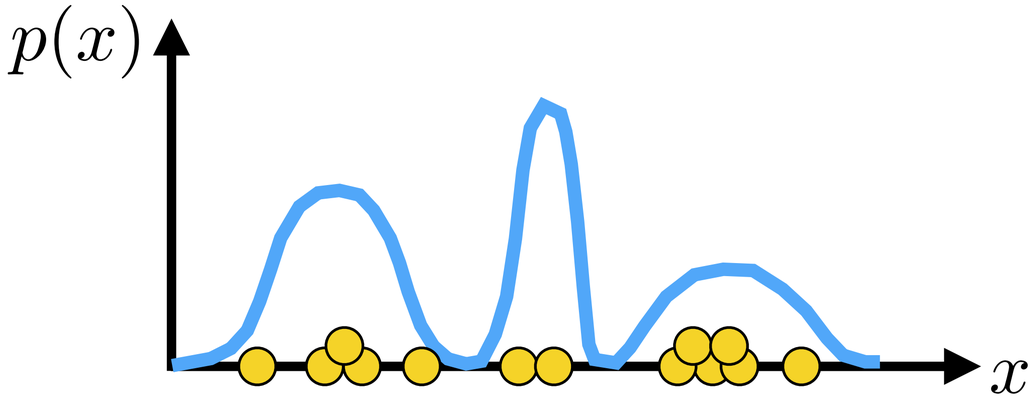
Nadia Chirkova

Higher School of Economics, Samsung-HSE Laboratory
Moscow, Russia

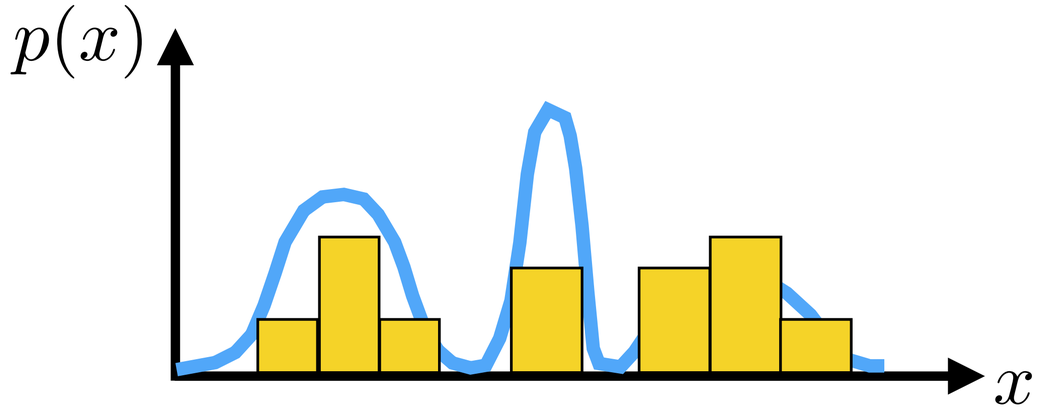
SAMSUNG
Research



Sampling points from a distribution



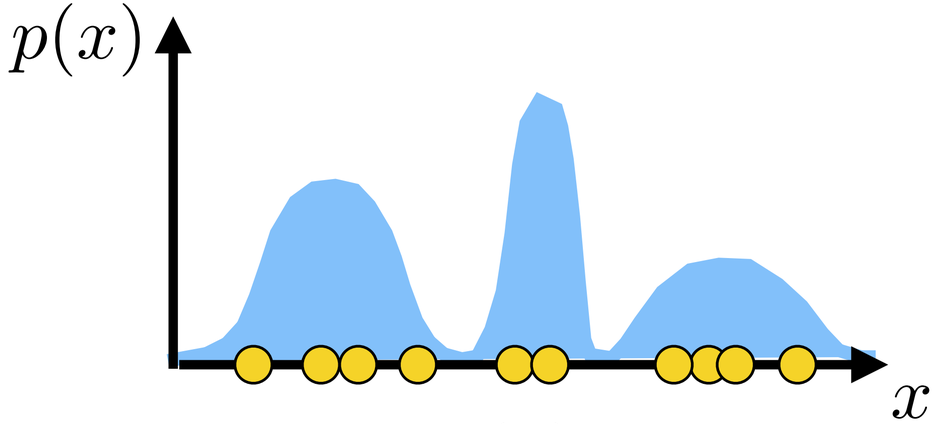
$x \sim p(x),\ x \in \mathbb{R}$



histogram approaches $p(x)$
when the number of points grows

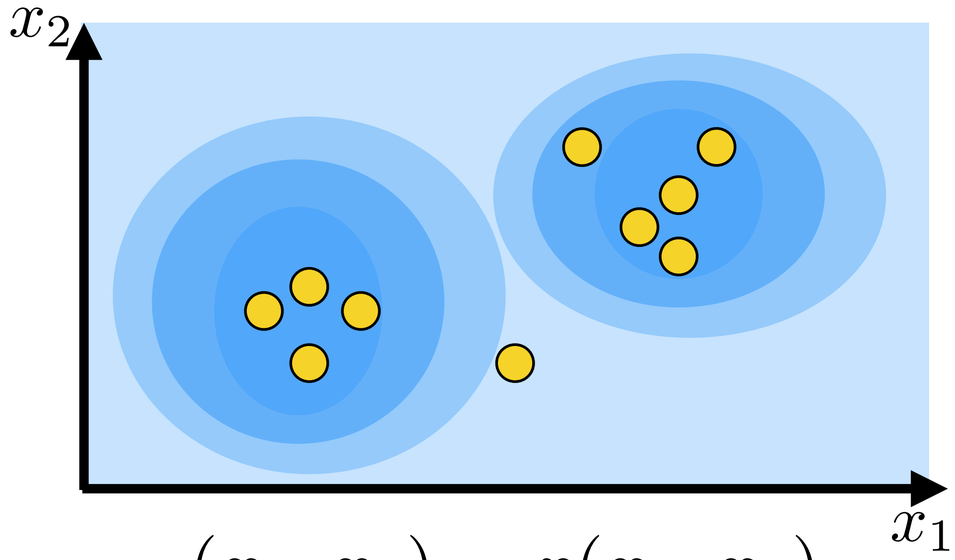
Sampling points from a multivariate distribution

1-dimensional distribution



$$x \sim p(x),\ x \in \mathbb{R}$$

2-dimensional distribution



$$(x_1, x_2) \sim p(x_1, x_2)$$

$$x \sim p(x),\ x \in \mathbb{R}^2$$

... n-dimensional distribution:

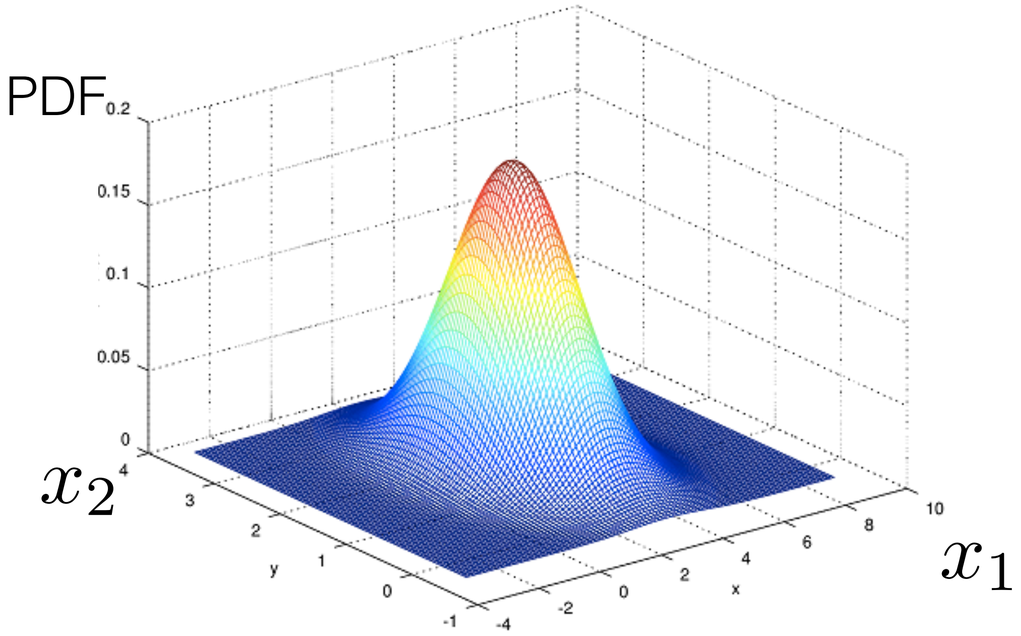
$$(x_1, \ldots, x_n) \sim p(x_1, \ldots, x_n)$$

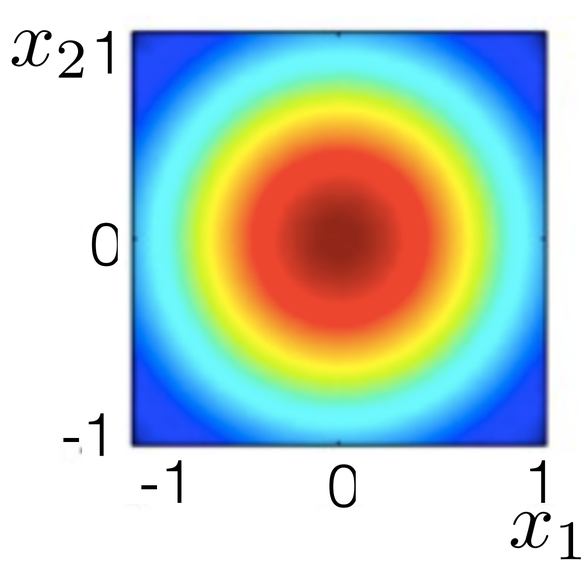
$$x \sim p(x),\ x \in \mathbb{R}^n$$

Multivariate normal (Gaussian) distribution

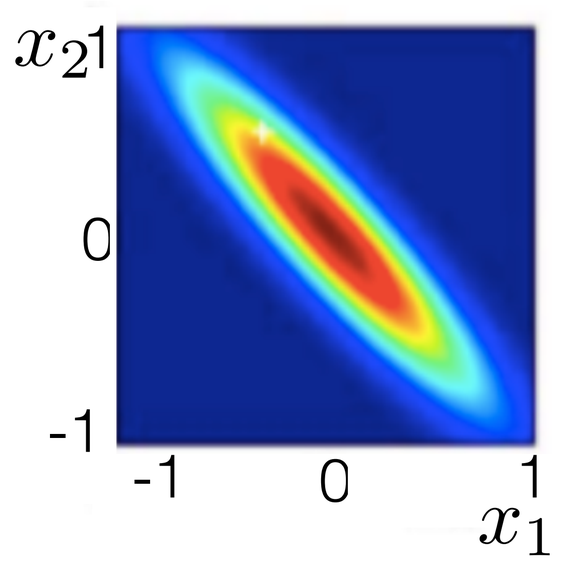
$$\mathcal{N}(\mu, \Sigma) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)),$$

$$x \in \mathbb{R}^d$$
$$\mu \in \mathbb{R}^d$$
$$\Sigma \in \mathbb{R}^{d\times d}$$





diagonal Σ



non-diagonal Σ

Images from https://medium.com/ming-learns-thing/machine-learning-bayesian-linear-regression-f160c4eaef99, C. Bishop. Pattern Recognition and Machine Learning

Multivariate normal (Gaussian) distribution

$$\mathcal{N}(\mu, \Sigma) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)),$$

$$x \in \mathbb{R}^d$$
$$\mu \in \mathbb{R}^d$$
$$\Sigma \in \mathbb{R}^{d\times d}$$

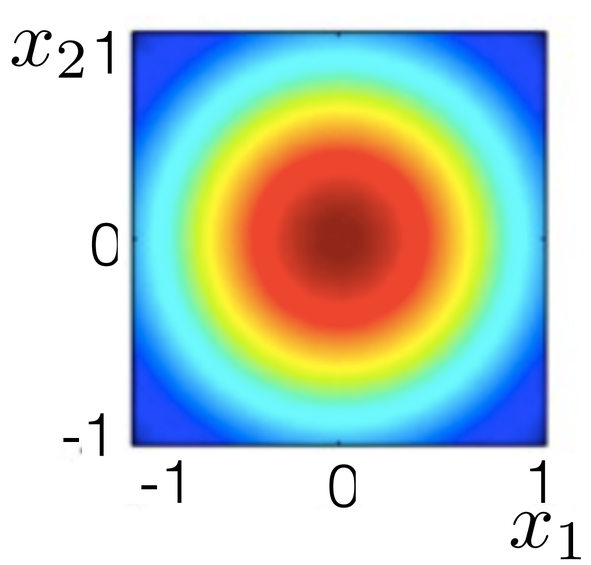
$\mathrm{Var}(x_1)$

$\mathrm{Cov}(x_1, x_2)$

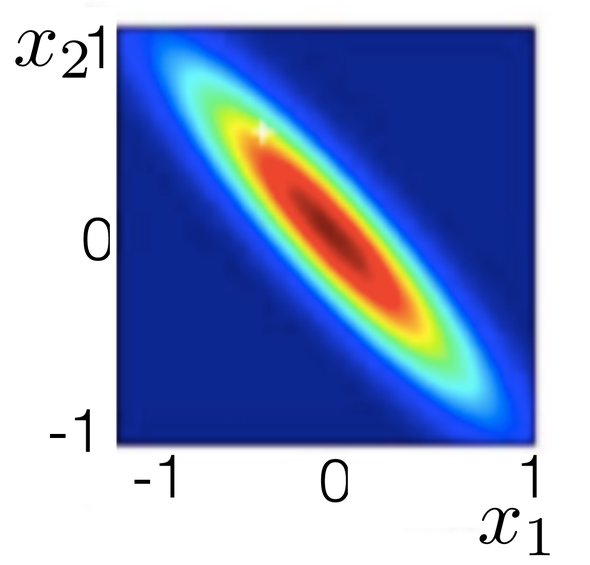
$$\Sigma \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{bmatrix}$$

diagonal $\Sigma \Rightarrow$

independent $x_1, \ldots x_d$



diagonal Σ

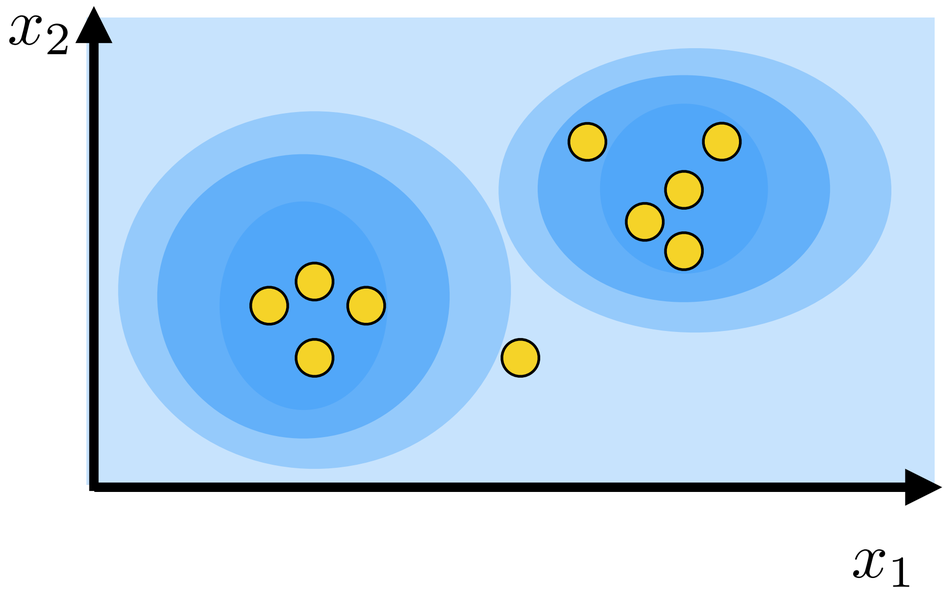


non-diagonal Σ

Images from https://medium.com/ming-learns-thing/machine-learning-bayesian-linear-regression-f160c4eaef99, C. Bishop. Pattern Recognition and Machine Learning

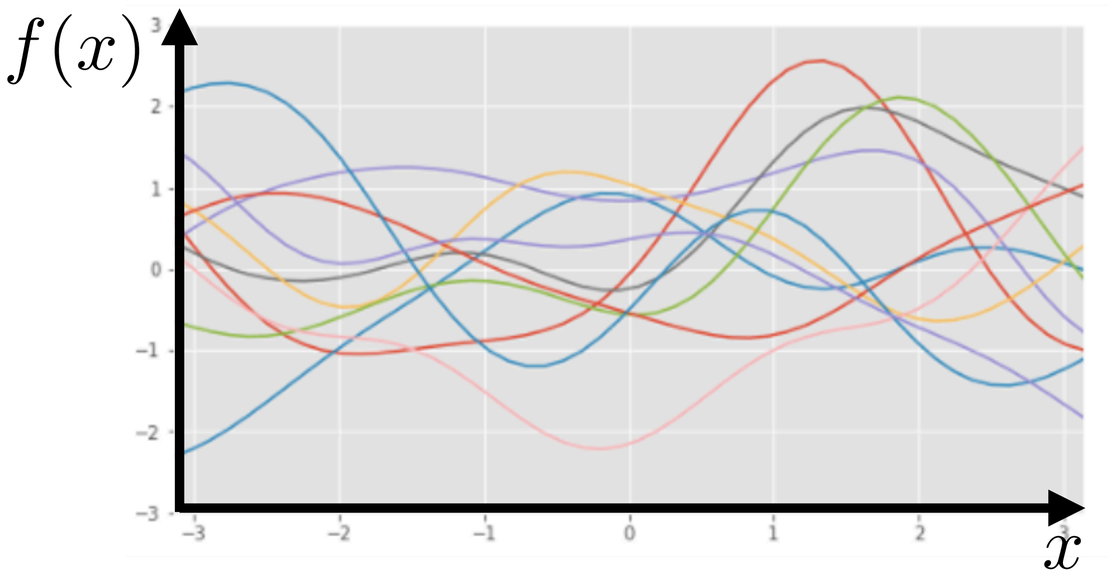
Sampling functions from a process

Multivariate distribution
— sample **points**



$$x \sim p(x)$$

Process
— sample **functions**



$$f(x) \sim GP(m(x), k(x, x'))$$

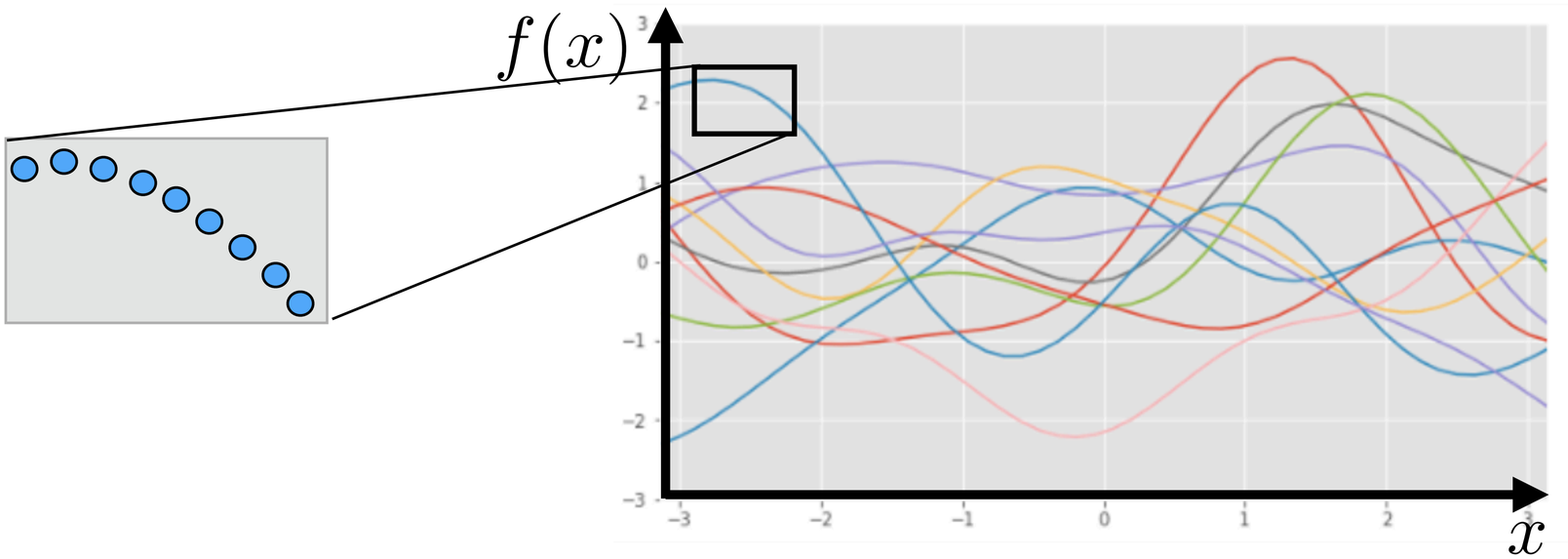
each line is a sample from the process

Right image from https://towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d

Sampling functions from a process

When we plot
a function in python,
we define a function as
a sequence of points

Process
— sample **function**



$$f(x) \sim GP(m(x), k(x, x'))$$

each line is a sample from the process

Right image from https://towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d

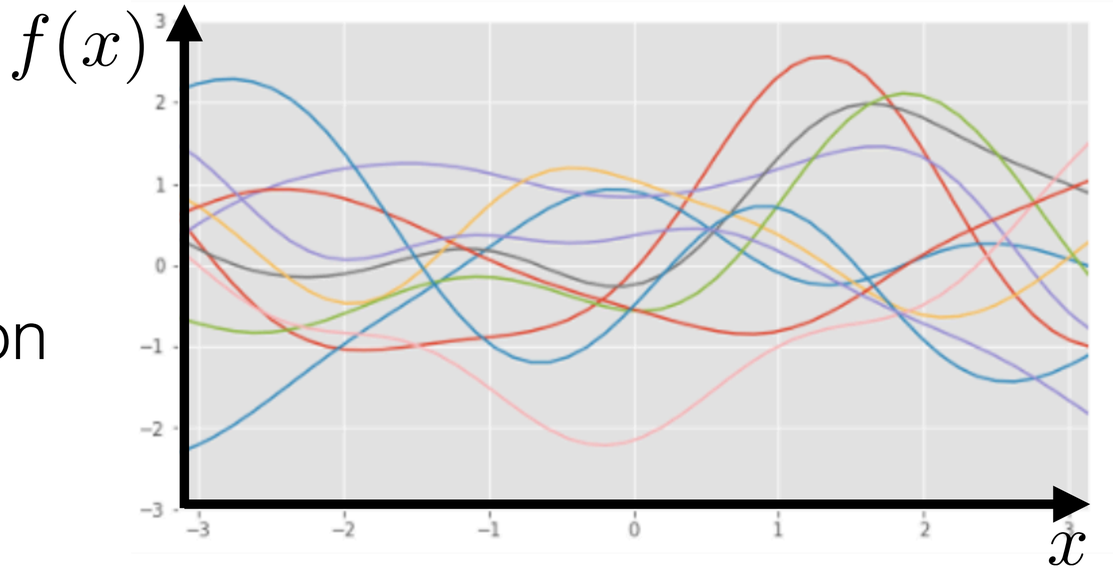
Gaussian process

$$f(x) \sim GP(m(x), k(x, x'))$$

$m(x)$ — mean function

$k(x, x')$ — covariance (or kernel) function

(x may be a vector in general case)



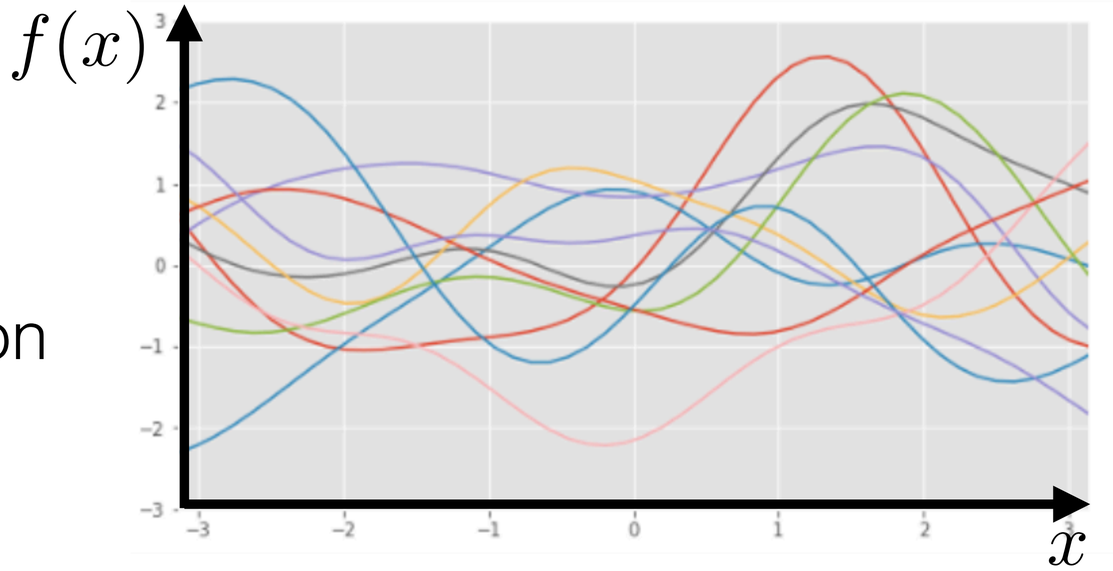
Gaussian process

$$f(x) \sim GP(m(x), k(x, x'))$$

$m(x)$ — mean function

$k(x, x')$ — covariance (or kernel) function

(x may be a vector in general case)



Definition of Gaussian process:
every finite set of function values has a multivariate normal distribution

$$\forall n \quad \forall (x_1, \ldots, x_n) \quad (f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu = \{m(x_i)\}_{i=1}^{n} \quad \Sigma = \{k(x_i, x_j)\}_{i,j=1}^{n,n}$$

Example 1: white noise

$$f(x) \sim GP(m(x), k(x, x'))$$

$$m(x) = 0$$

$$k(x, x') = \sigma^2[x = x']$$

$[\text{condition}] = 1$ if condition is True else 0

Example 1: white noise

$$f(x) \sim GP(m(x), k(x, x'))$$

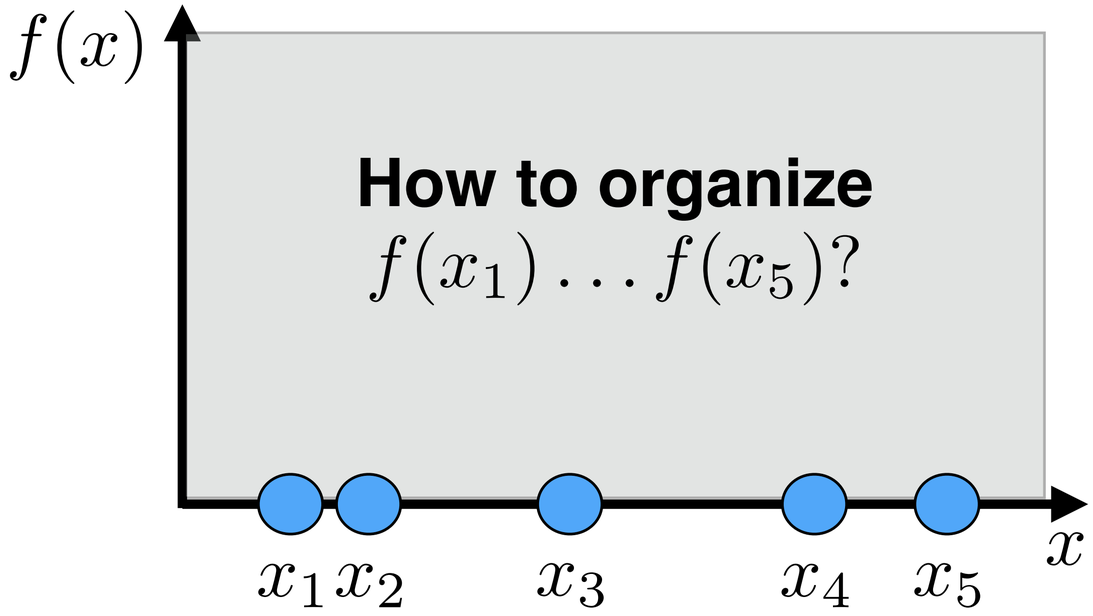
$$m(x) = 0$$

$$k(x, x') = \sigma^2 [x = x']$$

[condition] = 1 if condition is True else 0

$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu = 0 \qquad \Sigma = \sigma^2 I$$



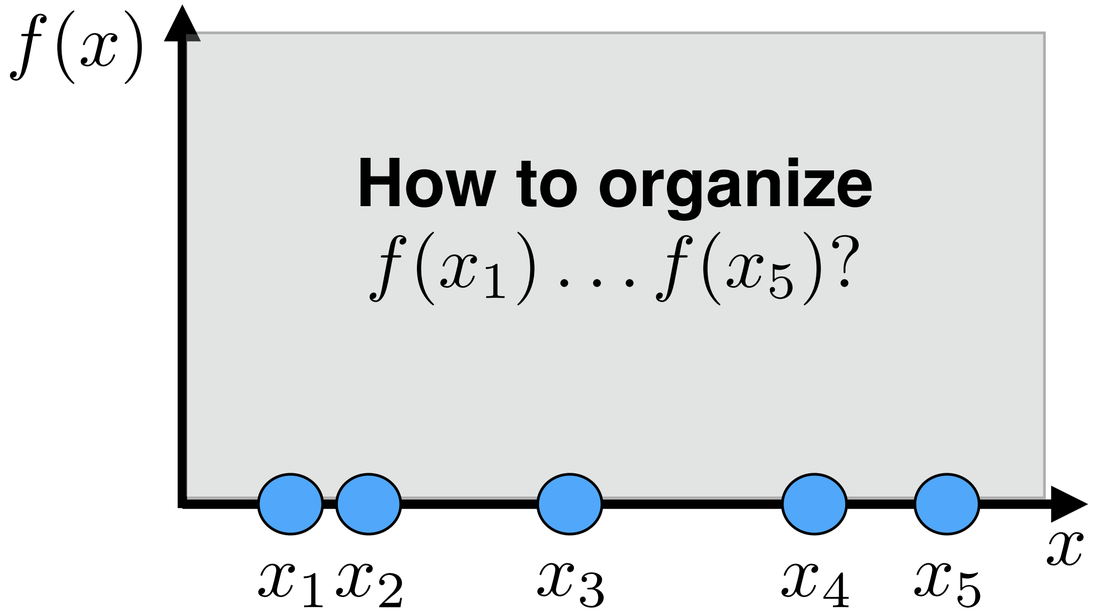
Example 1: white noise

$$f(x) \sim GP(m(x), k(x, x'))$$

$$m(x) = 0$$

$$k(x, x') = \sigma^2 [x = x']$$

[condition] = 1 if condition is True else 0



$$(f(x_1), \dots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu = 0 \qquad \Sigma = \sigma^2 I$$

$$p(f(x_1), \dots, f(x_n)) = \prod_{i=1}^{n} \mathcal{N}(0, \sigma^2)$$

(all x are independent on each other)

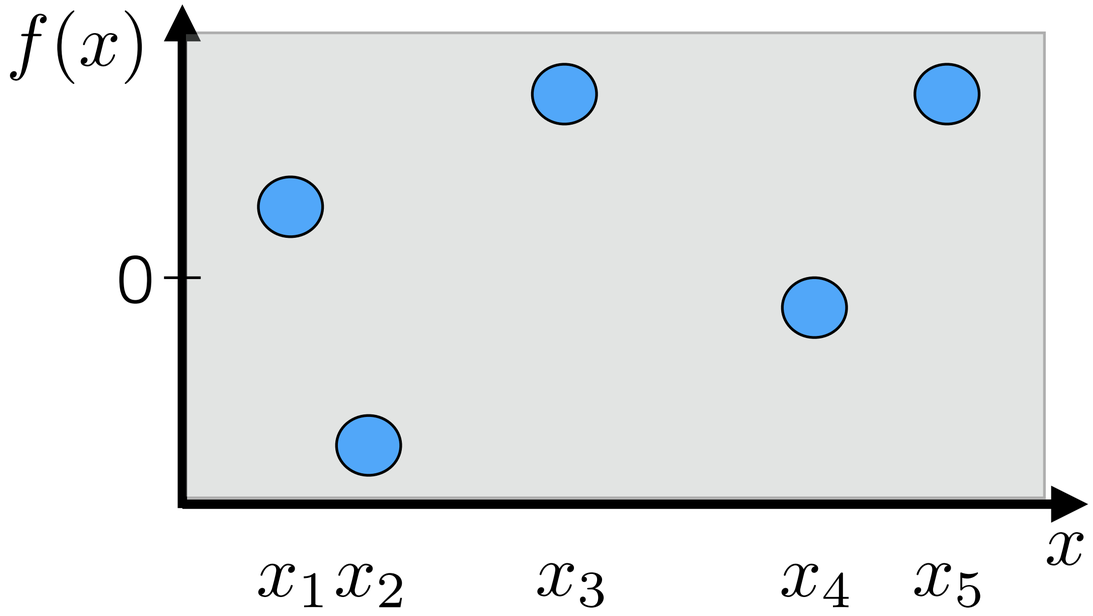
Example 1: white noise

$$f(x) \sim GP(m(x), k(x, x'))$$

$$m(x) = 0$$

$$k(x, x') = \sigma^2 [x = x']$$

[condition] = 1 if condition is True else 0



$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu = 0 \qquad \Sigma = \sigma^2 I$$

$$p(f(x_1), \ldots, f(x_n)) = \prod_{i=1}^{n} \mathcal{N}(0, \sigma^2)$$

for any x
$f(x)$ is sampled
independently

(all x are independent on each other)

Example 2: constant function

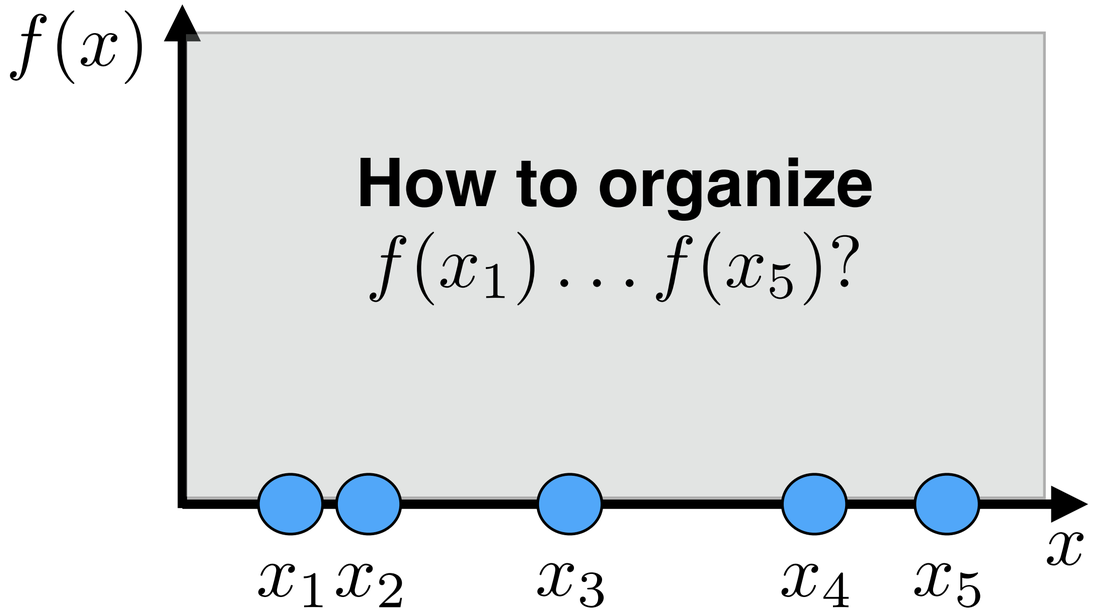
$$f(x) \sim GP(m(x), k(x, x'))$$

$$m(x) = 0$$

$$k(x, x') = \mathrm{C}$$

$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$

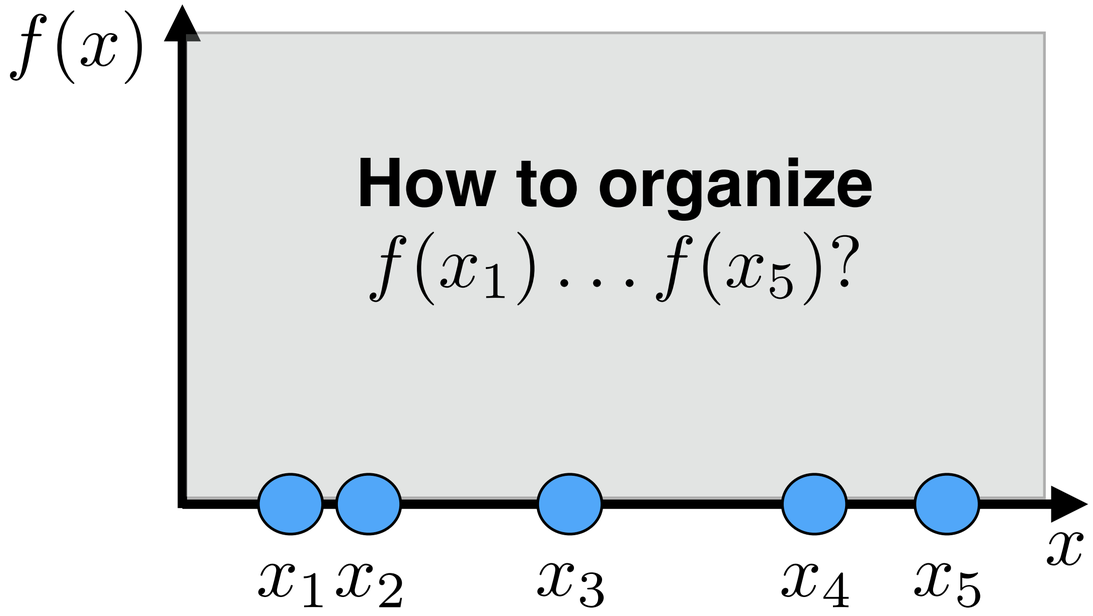
$$\mu = 0 \qquad \Sigma = \{C\}_{i,j=1}^{n,n}$$



Example 2: constant function

$$f(x) \sim GP(m(x), k(x, x'))$$
$$m(x) = 0$$
$$k(x, x') = \mathrm{C}$$

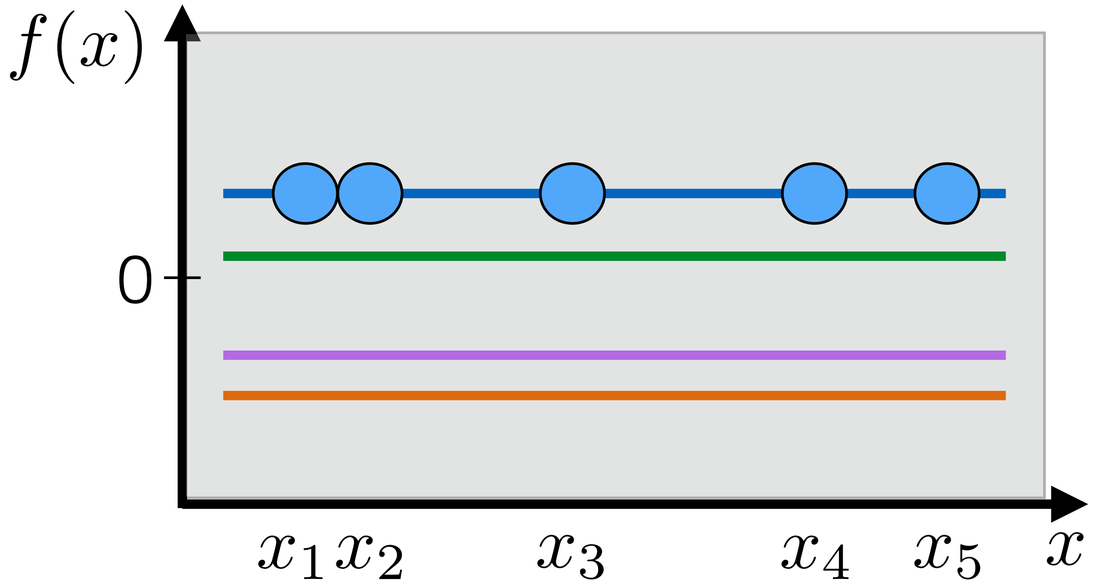
$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$
$$\mu = 0 \qquad \Sigma = \{C\}_{i,j=1}^{n,n}$$



$$\forall i \neq j$$
$$\mathrm{Corr}(f(x_i), f(x_j)) = \frac{\mathrm{Cov}(f(x_i), f(x_j))}{\sqrt{\mathrm{Var}(f(x_i))\mathrm{Var}(f(x_j))}} = \frac{C}{\sqrt{C}^2} = 1$$
$$\mathrm{Var}(f(x_i)) = \mathrm{Var}(f(x_j)) = C, \quad \mathbb{E}f(x_i) = \mathbb{E}f(x_j) = 0$$
$$\Rightarrow f(x_i) = f(x_j)$$

Example 2: constant function

$$f(x) \sim GP(m(x), k(x, x'))$$
$$m(x) = 0$$
$$k(x, x') = \mathrm{C}$$



$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$
$$\mu = 0 \qquad \Sigma = \{C\}_{i,j=1}^{n,n}$$

$$\left.\begin{array}{l} \forall i \neq j \\ \mathrm{Corr}(f(x_i), f(x_j)) = \dfrac{\mathrm{Cov}(f(x_i), f(x_j))}{\sqrt{\mathrm{Var}(f(x_i))\mathrm{Var}(f(x_j))}} = \dfrac{C}{\sqrt{C}^2} = 1 \\ \mathrm{Var}(f(x_i)) = \mathrm{Var}(f(x_j)) = C, \quad \mathbb{E}f(x_i) = \mathbb{E}f(x_j) = 0 \end{array}\right\} \Rightarrow f(x_i) = f(x_j)$$

Example 3: RBF-kernel

$$f(x) \sim GP(m(x), k(x, x'))$$
$$m(x) = 0$$
$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$

$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$
$$\mu = 0 \qquad \Sigma = \{k(x_i, x_j)\}_{i,j=1}^{n,n}$$

Example 3: RBF-kernel

$$f(x) \sim GP(m(x), k(x, x'))$$
$$m(x) = 0$$
$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$

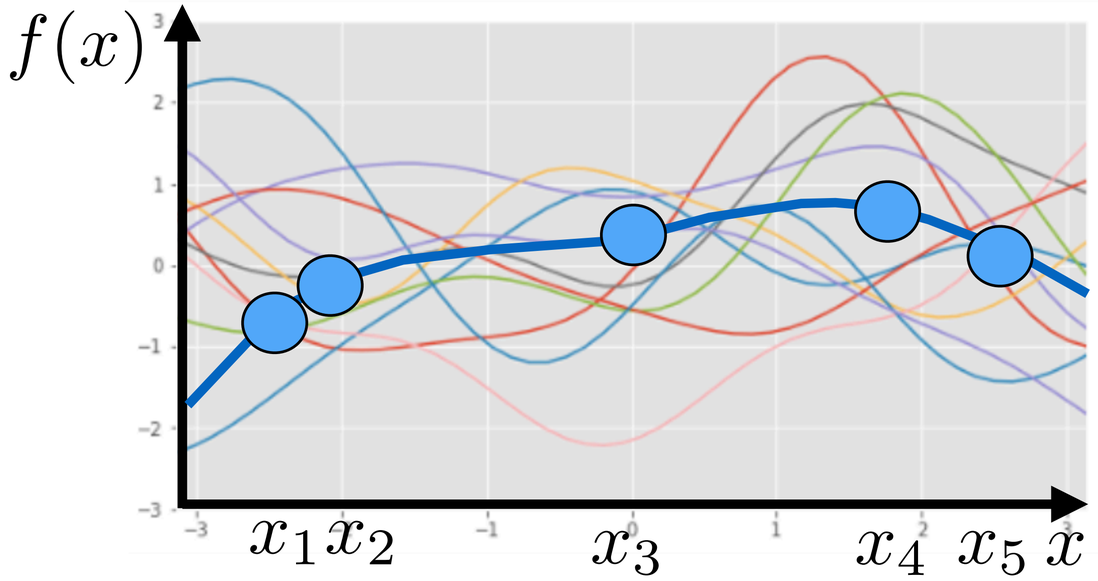
$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$
$$\mu = 0 \qquad \Sigma = \{k(x_i, x_j)\}_{i,j=1}^{n,n}$$

$$\text{if } \|x_i - x_j\| \approx 0 \quad \Rightarrow \quad \Sigma_{ij} \approx \sigma^2 = \Sigma_{ii} = \Sigma_{jj} \quad \Rightarrow \quad f(x_i) \approx f(x_j)$$

$$\text{if } \|x_i - x_j\| \gg 0 \quad \Rightarrow \quad \Sigma_{ij} \approx 0, \;\; f(x_i) \text{ and } f(x_j) \text{ are not correlated}$$

Example 3: RBF-kernel

$$f(x) \sim GP(m(x), k(x, x'))$$
$$m(x) = 0$$
$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$



$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$
$$\mu = 0 \qquad \Sigma = \{k(x_i, x_j)\}_{i,j=1}^{n,n}$$

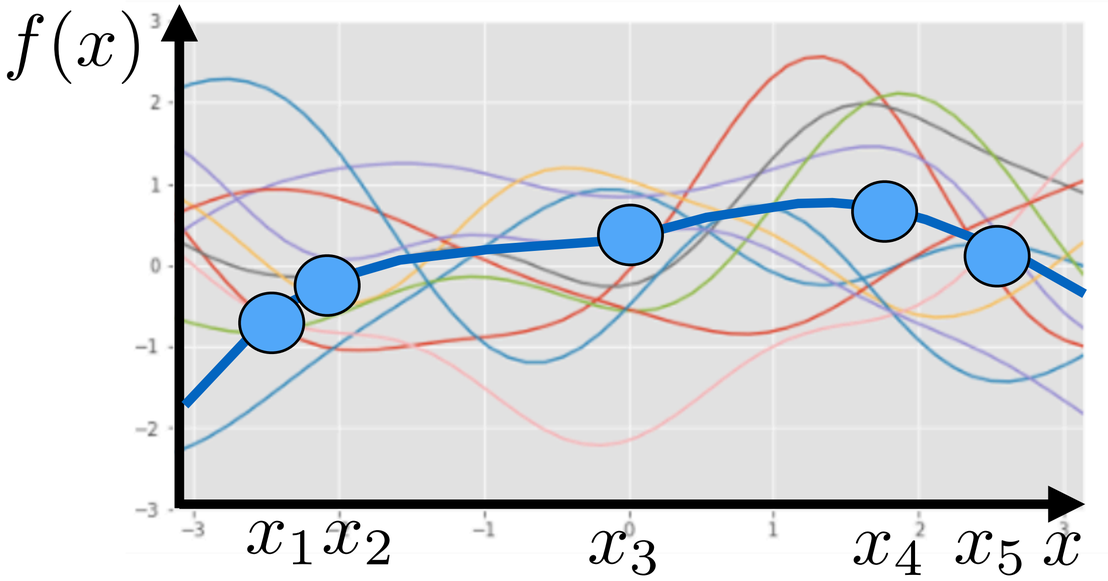
$$\text{if } \|x_i - x_j\| \approx 0 \quad \Rightarrow \quad \Sigma_{ij} \approx \sigma^2 = \Sigma_{ii} = \Sigma_{jj} \quad \Rightarrow \quad f(x_i) \approx f(x_j)$$

$$\text{if } \|x_i - x_j\| \gg 0 \quad \Rightarrow \quad \Sigma_{ij} \approx 0, \;\; f(x_i) \text{ and } f(x_j) \text{ are not correlated}$$

Example 3: RBF-kernel

$$f(x) \sim GP(m(x), k(x, x'))$$
$$m(x) = 0$$
$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$

$$(f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$
$$\mu = 0 \qquad \Sigma = \{k(x_i, x_j)\}_{i,j=1}^{n,n}$$



σ^2 defines the "height" of the function
ℓ^2 defines the frequency of fluctuations

Example 3: RBF-kernel

ℓ^2 defines the frequency of fluctuations

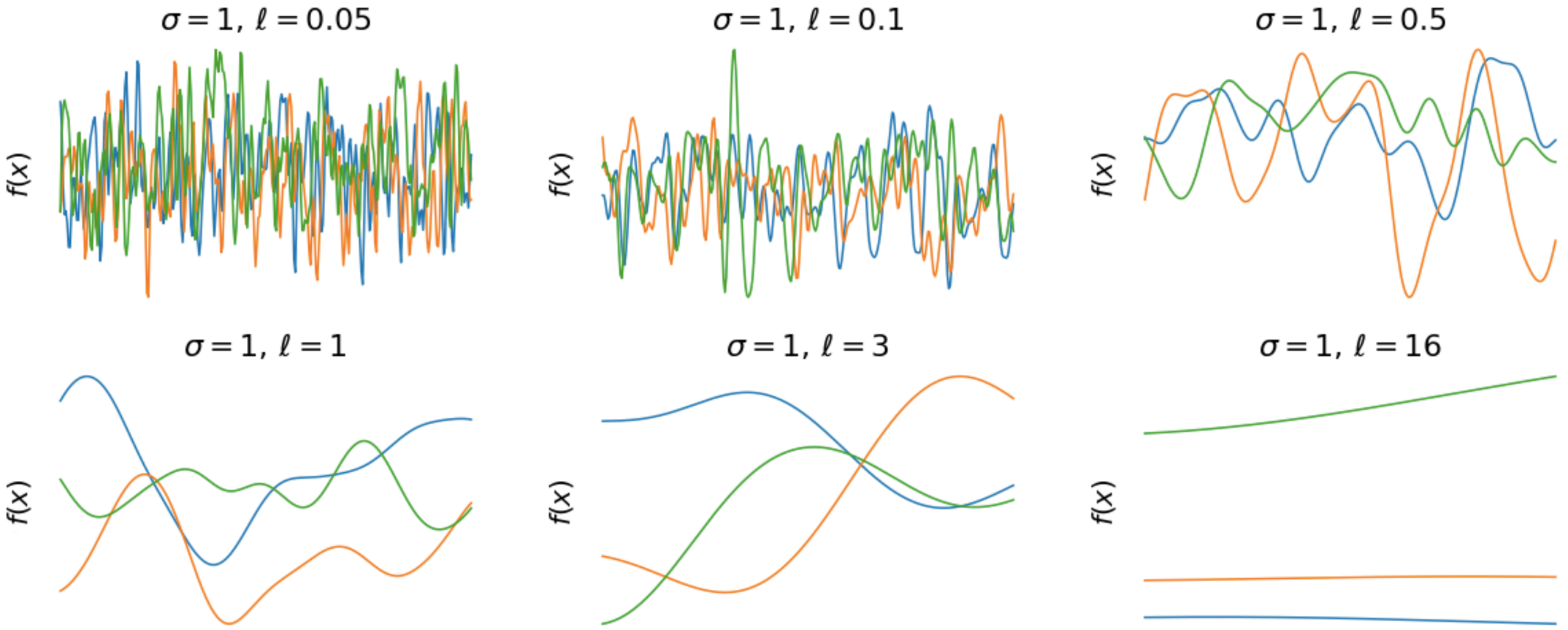
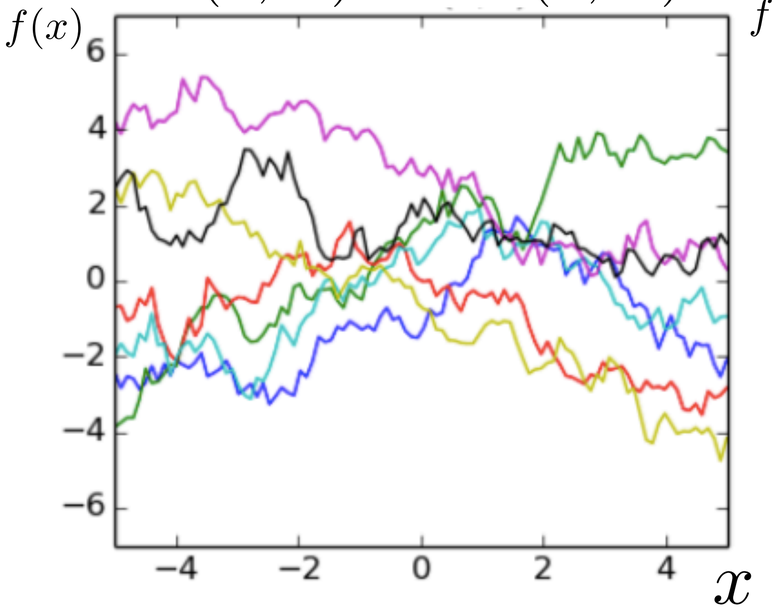


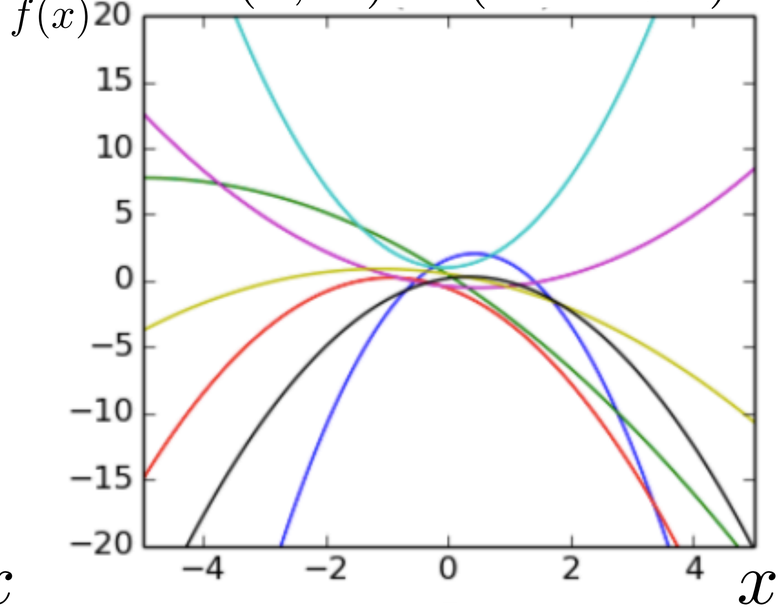
Image from https://relguzman.blogspot.com/2018/03/gaussian-processes-explained.html

More kernels

$k(x, x') = \min(x, x')$



$k(x, x') = (x \cdot x' + c)^2$



$k(x, x') = x \cdot x'$

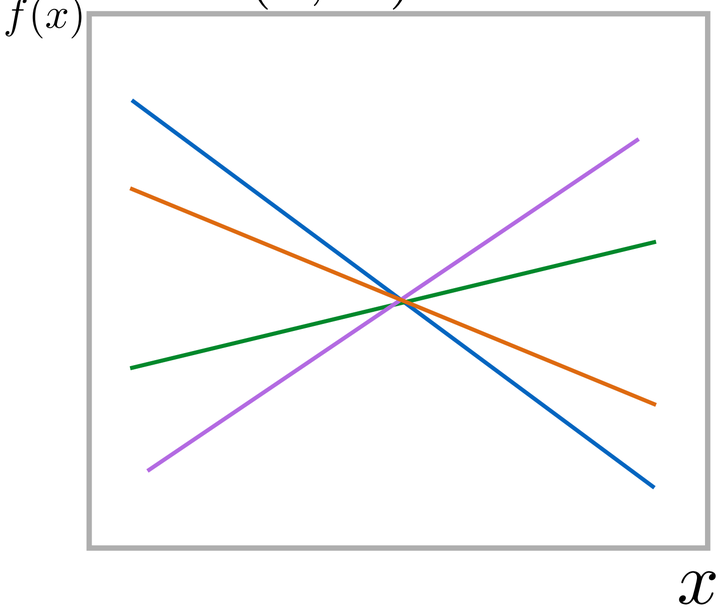
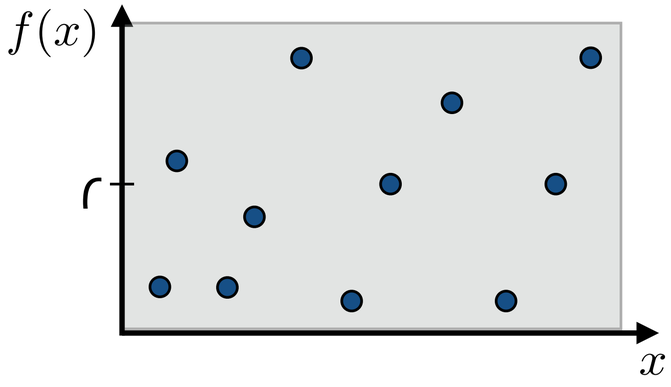
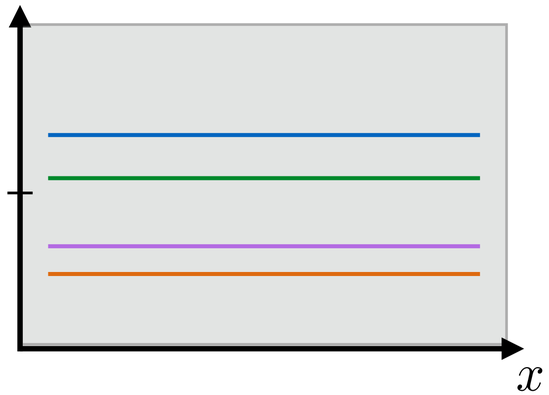


Image from https://www.kaggle.com/residentmario/gaussian-process-regression-and-classification

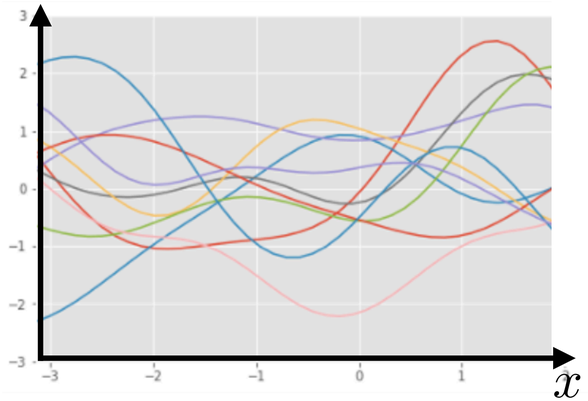
Sum-kernel



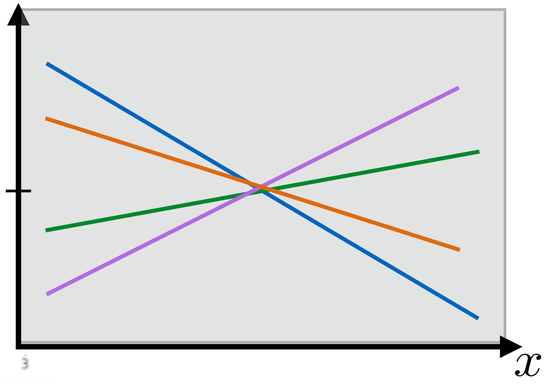
$$k(x, x') = \sigma^2[x = x']$$



$$k(x, x') = \mathrm{C}$$



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

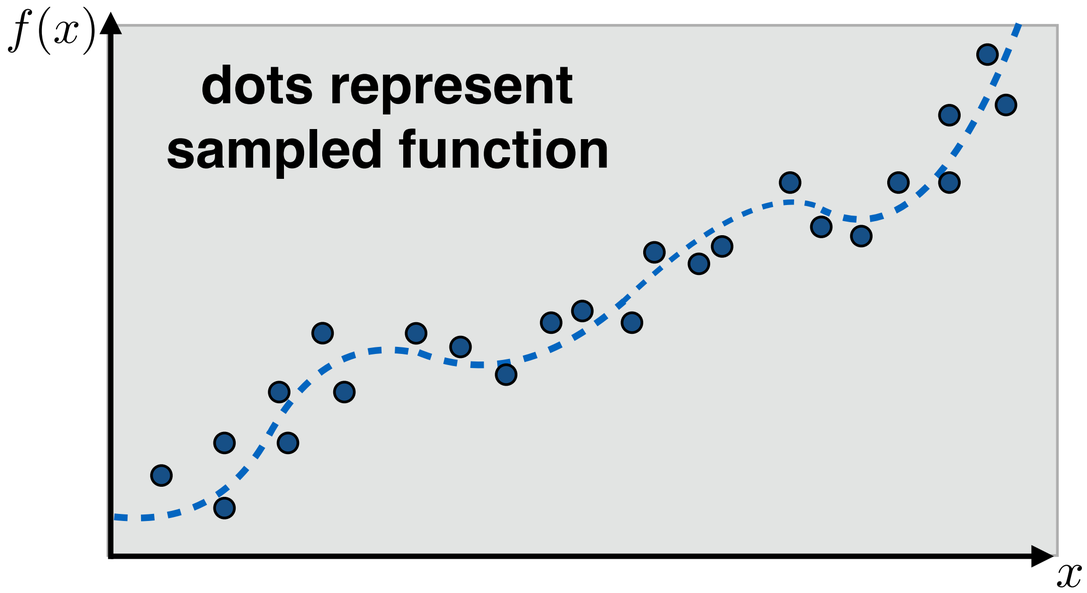


$$k(x, x') = x \cdot x'$$

Sum-kernel (multidimensional case):

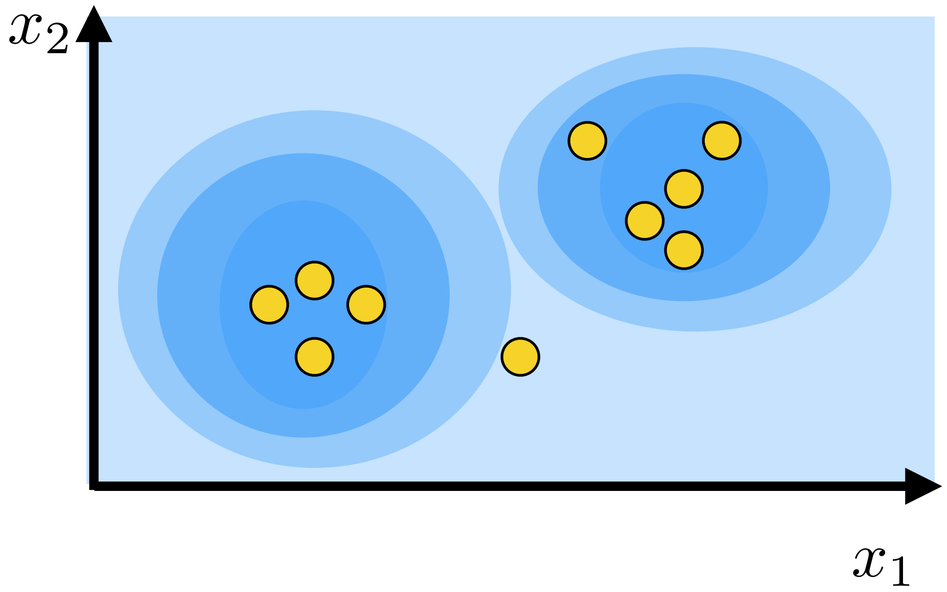
$$k(x, x') = x^T x' + \sigma_1^2 \exp\left(-\frac{||x - x'||^2}{2\ell^2}\right) + \sigma_2^2[x = x'] + \sigma_3^2$$

$$x, x' \in \mathbb{R}^d, d - \text{number of features}$$



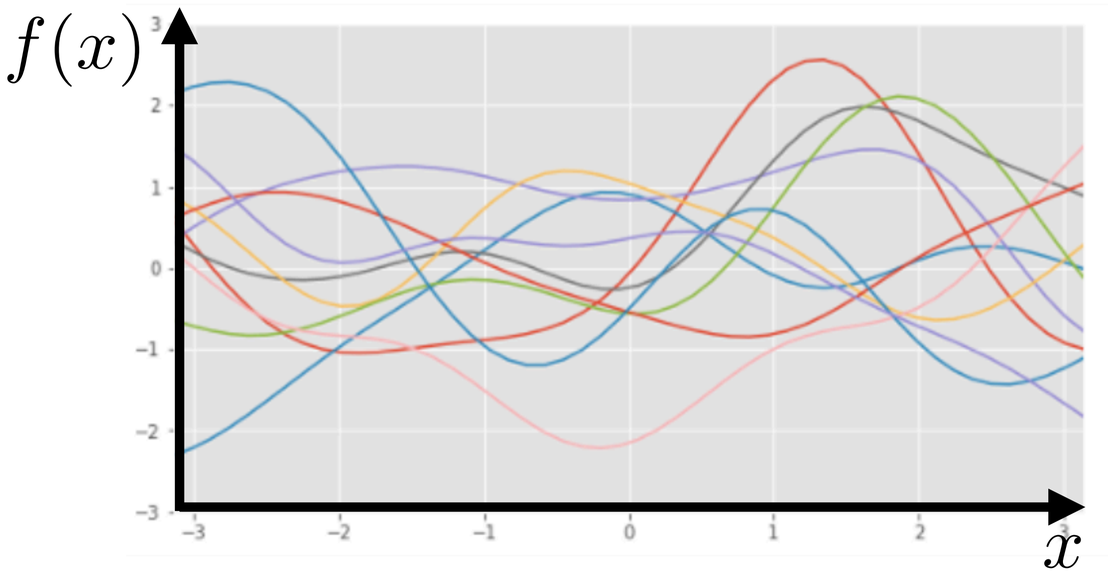
Sampling functions from a process

Multivariate distribution
— sample **points**



$$x \sim p(x)$$

Process
— sample **functions**

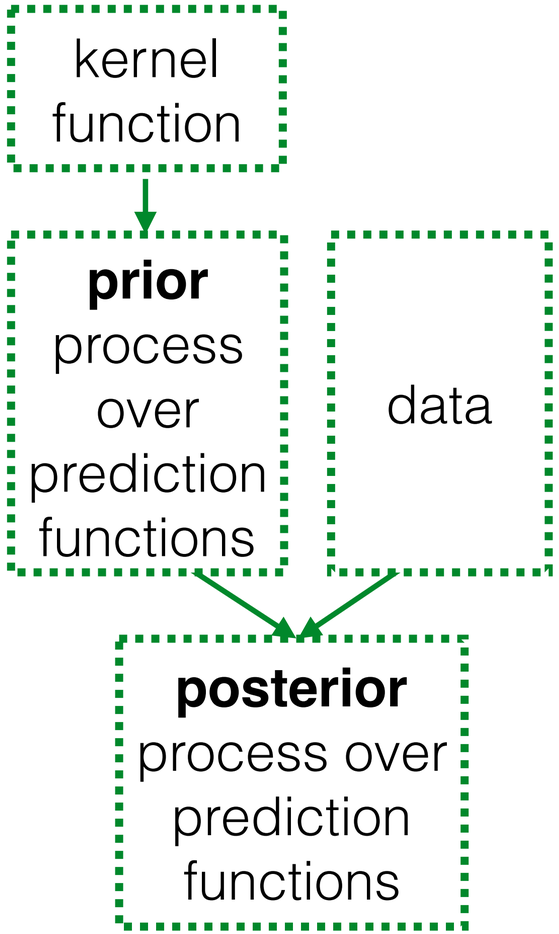


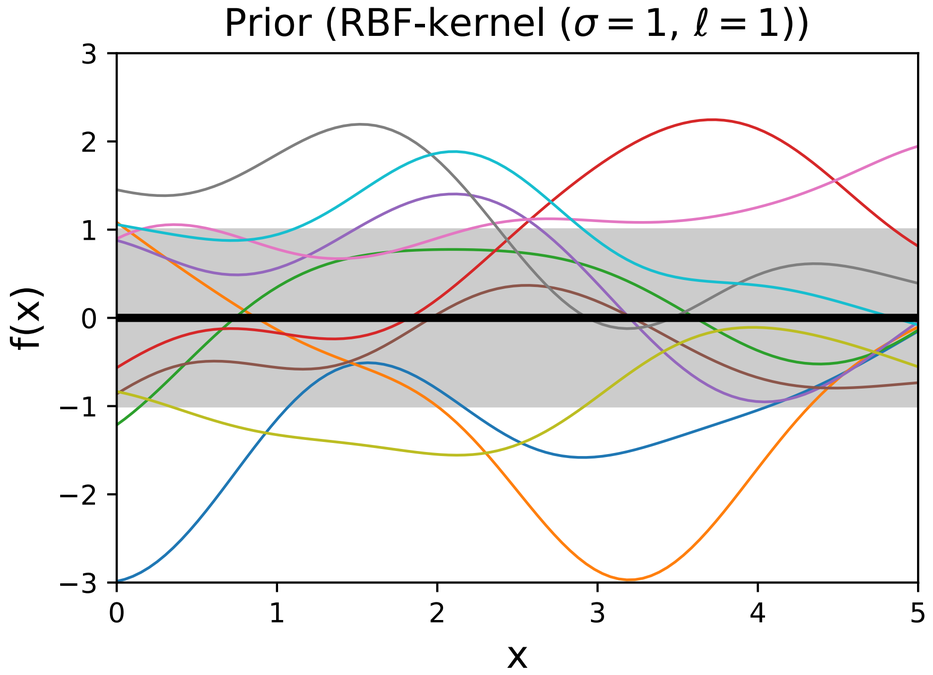
$$f(x) \sim GP(m(x), k(x, x'))$$

each line is a sample from the process

Right image from https://towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d

Gaussian processes for regression





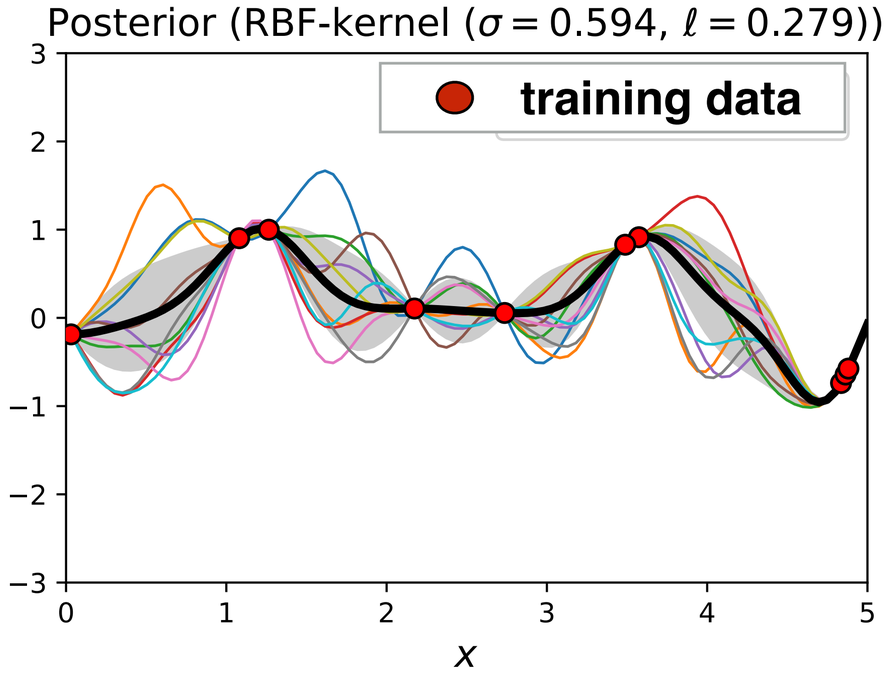
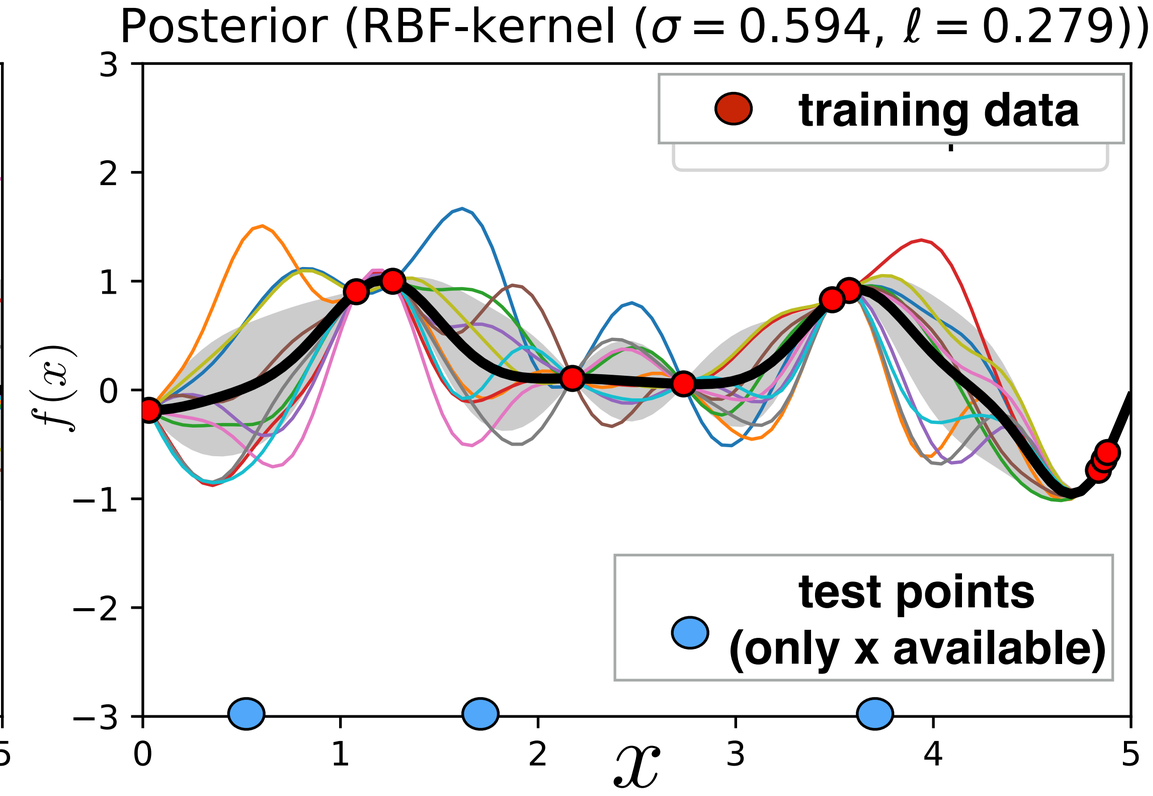


Image source: https://scikit-learn.org/stable/modules/gaussian_process.html

Gaussian processes for regression



Training data ● ● ● :

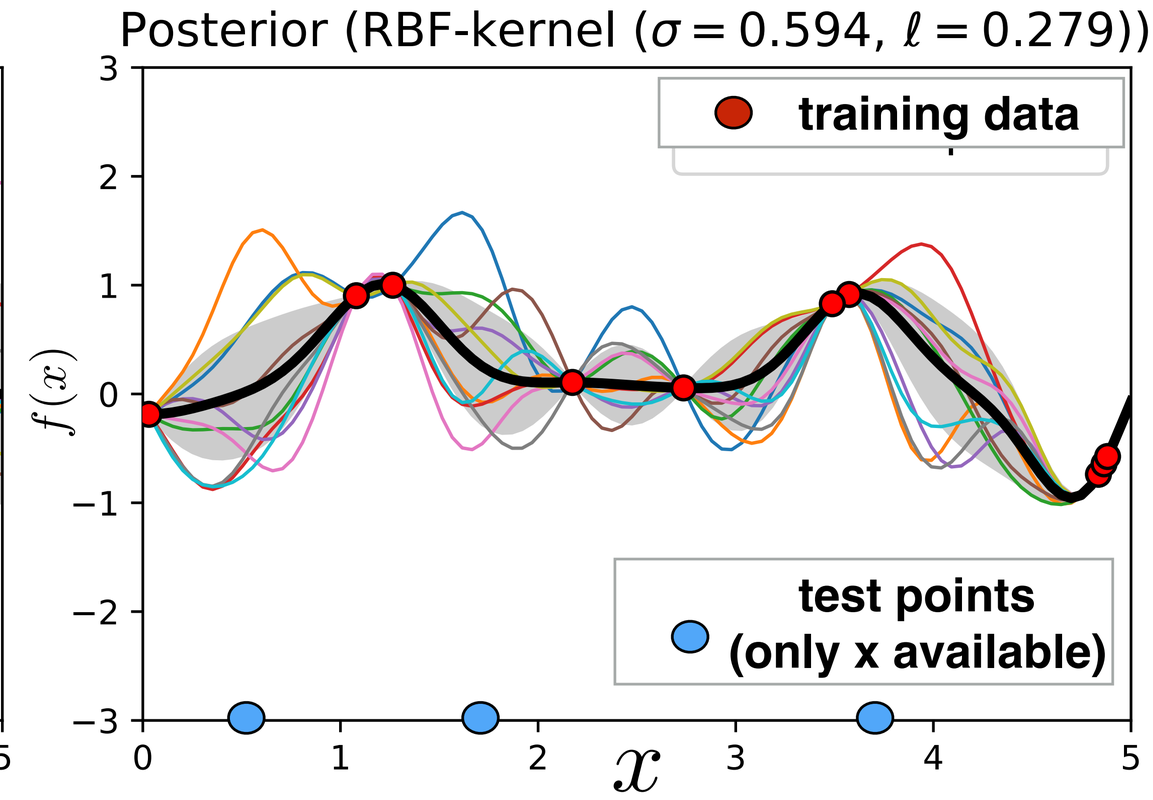
$X^{tr} = \{x_i^{tr}\}_{i=1}^N$, $x_i^{tr} \in \mathbb{R}^d$ — input data

$Y^{tr} = \{y_i^{tr}\}_{i=1}^N$, $y_i^{tr} \in \mathbb{R}$ — targets

N – number of objects, d – number of features

Given: (1) training data and
(2) prior Gaussian process over
prediction functions $a(x)$

Gaussian processes for regression



Given: (1) training data and
(2) prior Gaussian process over
prediction functions $a(x)$

Training data ● ● ● :

$X^{tr} = \{x_i^{tr}\}_{i=1}^{N}$, $x_i^{tr} \in \mathbb{R}^d$ — input data

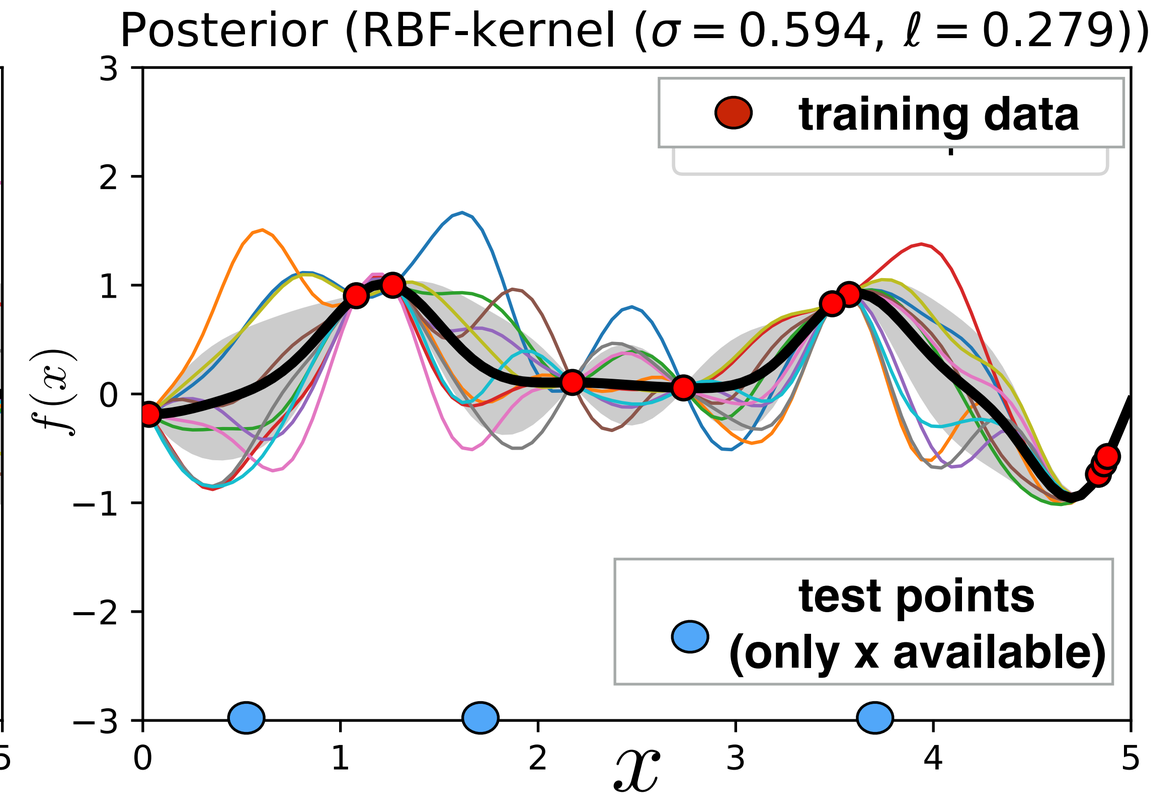
$Y^{tr} = \{y_i^{tr}\}_{i=1}^{N}$, $y_i^{tr} \in \mathbb{R}$ — targets

N – number of objects, d – number of features

Test points ● ● ● (any set of points):

$X^{te} = \{x_i^{te}\}_{i=1}^{M}$, $x_i^{te} \in \mathbb{R}^d$

Gaussian processes for regression



Given: (1) training data and
(2) prior Gaussian process over
prediction functions $a(x)$

Training data ● ● ● :

$X^{tr} = \{x_i^{tr}\}_{i=1}^N$, $x_i^{tr} \in \mathbb{R}^d$ — input data

$Y^{tr} = \{y_i^{tr}\}_{i=1}^N$, $y_i^{tr} \in \mathbb{R}$ — targets

N – number of objects, d – number of features

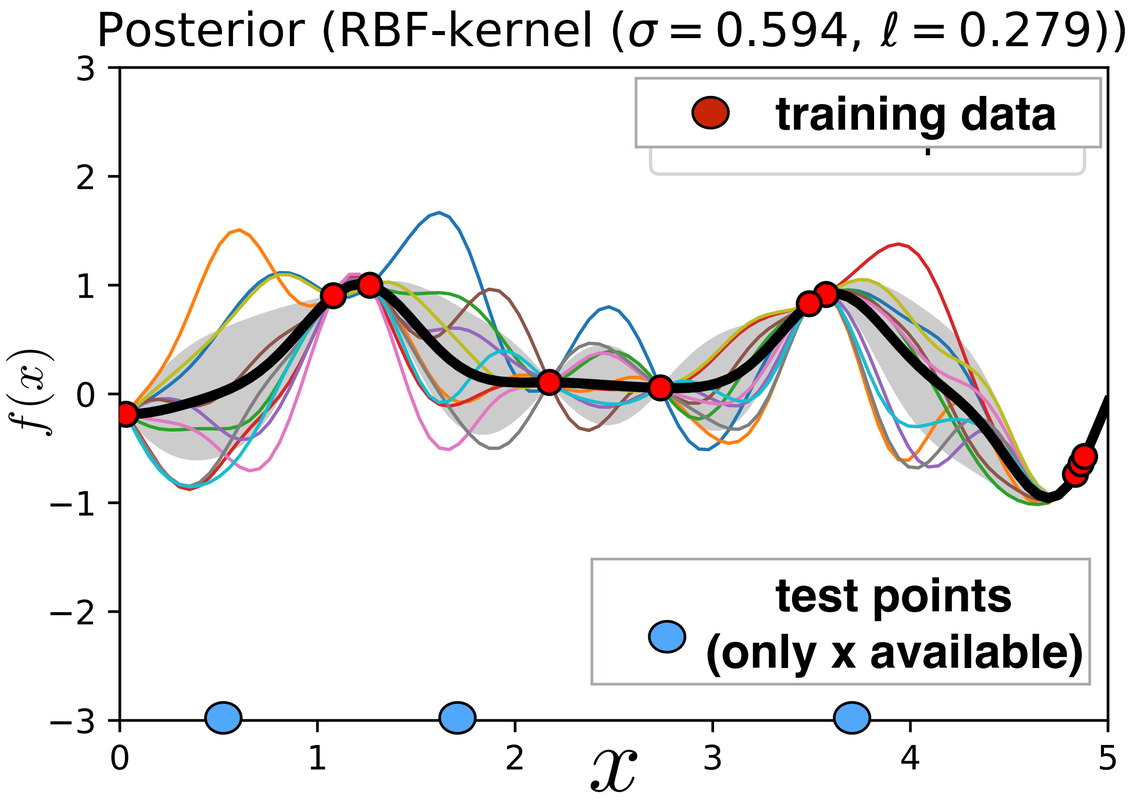
Test points ● ● ● (any set of points):

$X^{te} = \{x_i^{te}\}_{i=1}^M$, $x_i^{te} \in \mathbb{R}^d$

Find:

$p(a(x_1^{te}), \ldots, a(x_M^{te}))-?$ **p(●●●)**

Conditioning in multivariate normal distribution

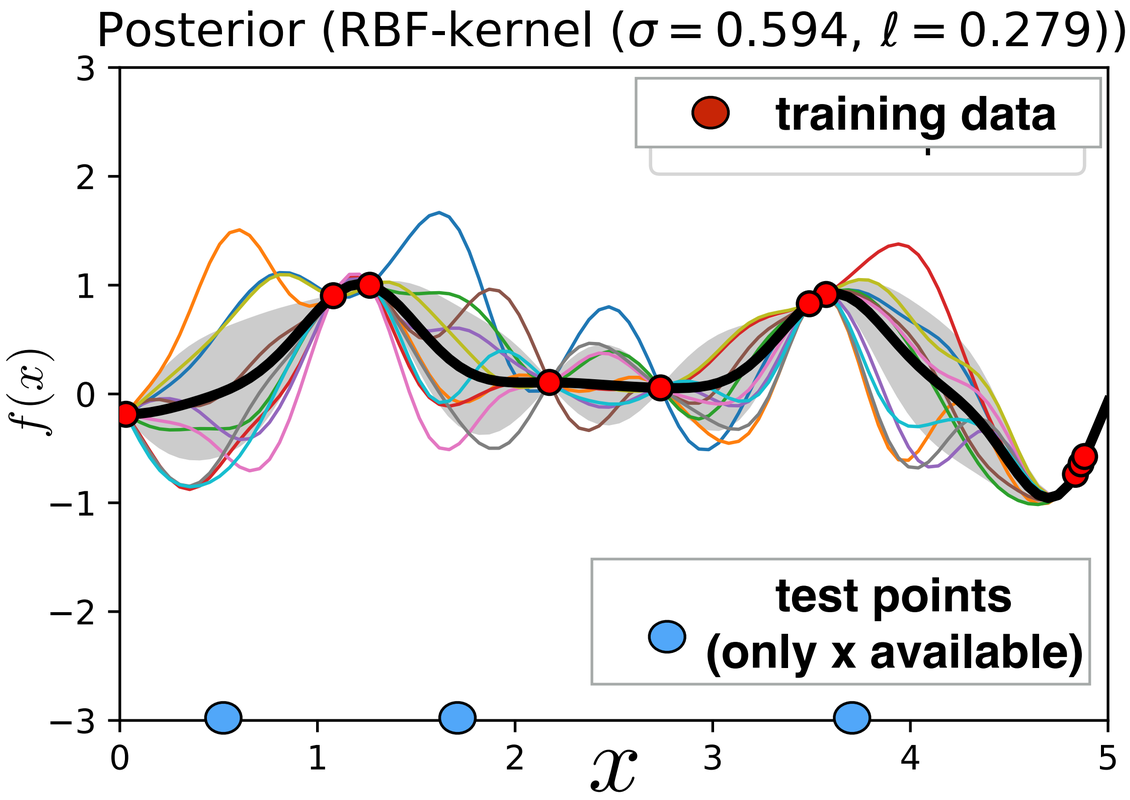


Definition of Gaussian process:
every finite set of function values
has a multivariate normal distribution

$$\forall n \quad \forall (x_1, \ldots, x_n) \quad (a(x_1), \ldots, a(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu = \{m(x_i)\}_{i=1}^{n} \quad \Sigma = \{k(x_i, x_j)\}_{i,j=1}^{n,n}$$

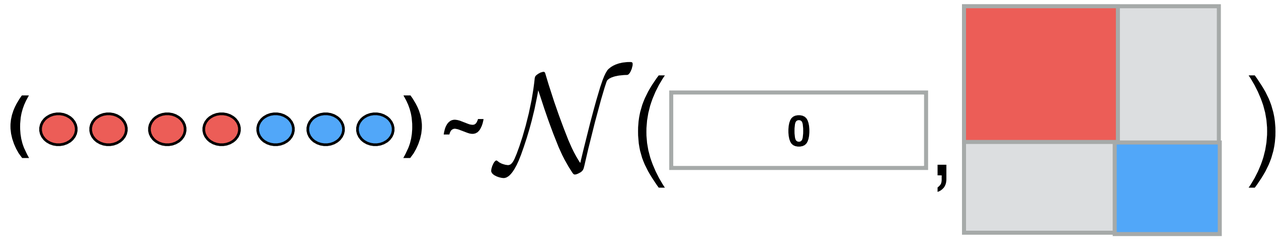
Conditioning in multivariate normal distribution



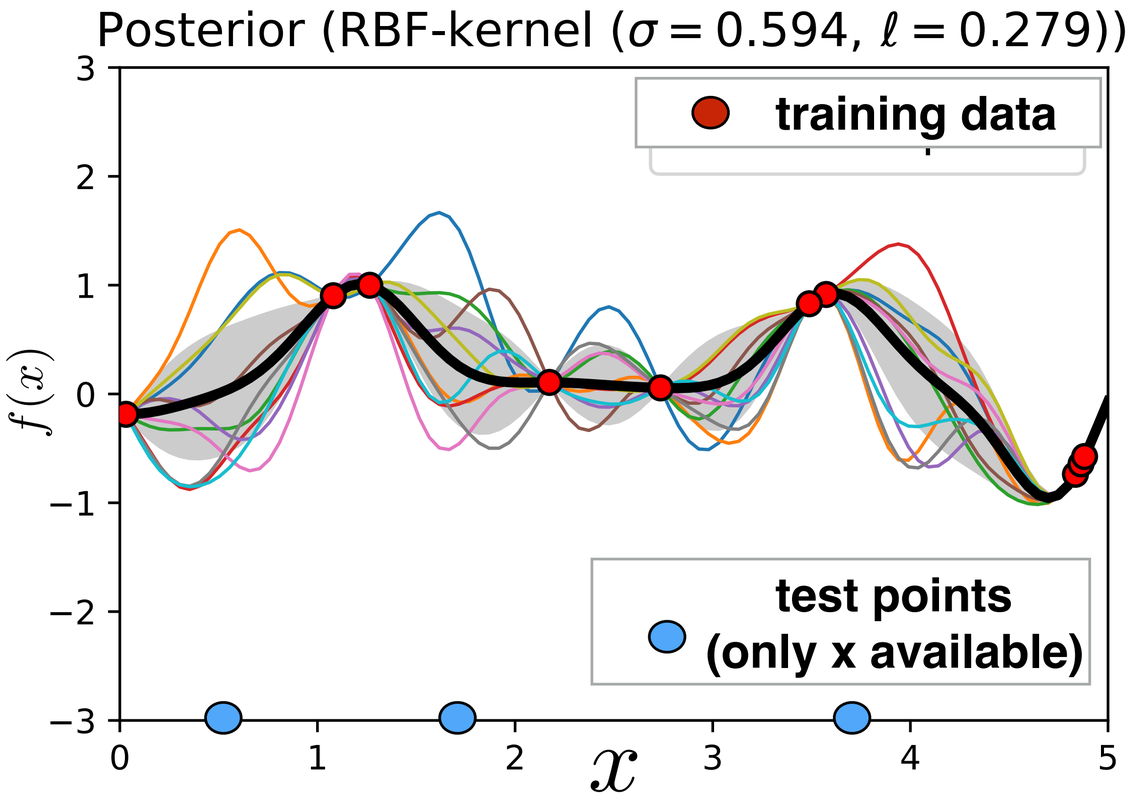
Definition of Gaussian process:
every finite set of function values
has a multivariate normal distribution

$$\forall n \quad \forall (x_1, \ldots, x_n) \quad (a(x_1), \ldots, a(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu = \{m(x_i)\}_{i=1}^{n} \quad \Sigma = \{k(x_i, x_j)\}_{i,j=1}^{n,n}$$



Conditioning in multivariate normal distribution

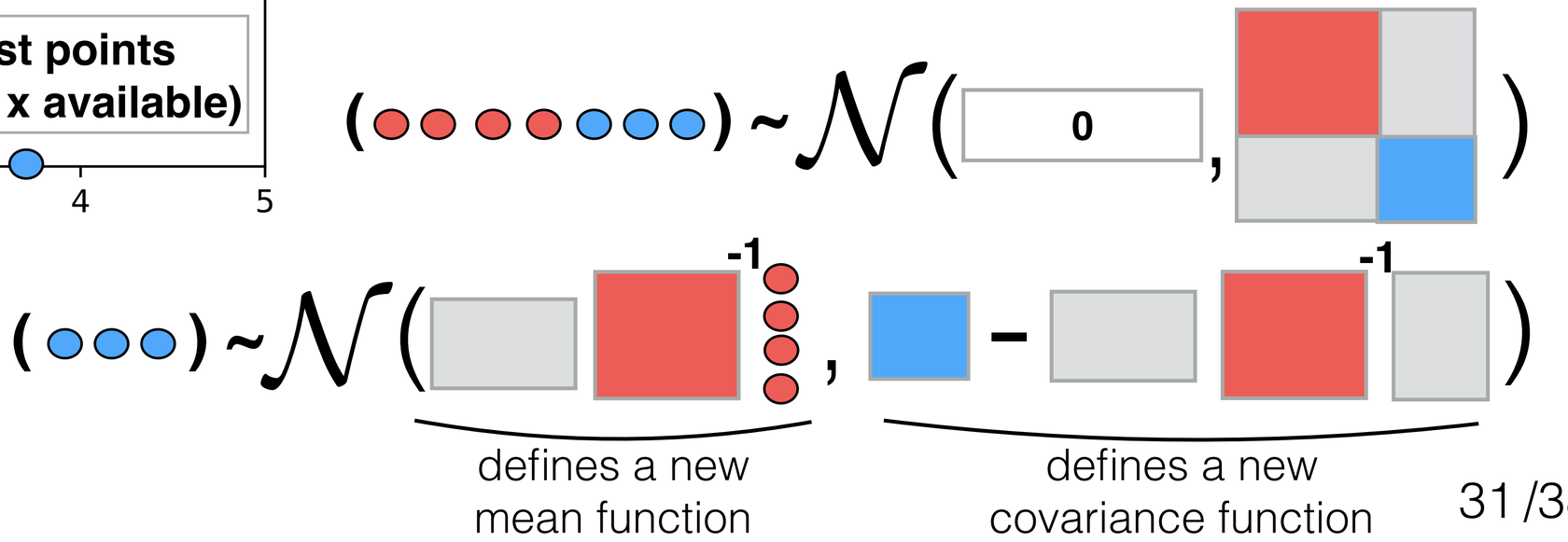


Definition of Gaussian process:
every finite set of function values
has a multivariate normal distribution

$$\forall n \quad \forall (x_1, \ldots, x_n) \quad (a(x_1), \ldots, a(x_n)) \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu = \{m(x_i)\}_{i=1}^{n} \quad \Sigma = \{k(x_i, x_j)\}_{i,j=1}^{n,n}$$

According to properties of normal distribution:



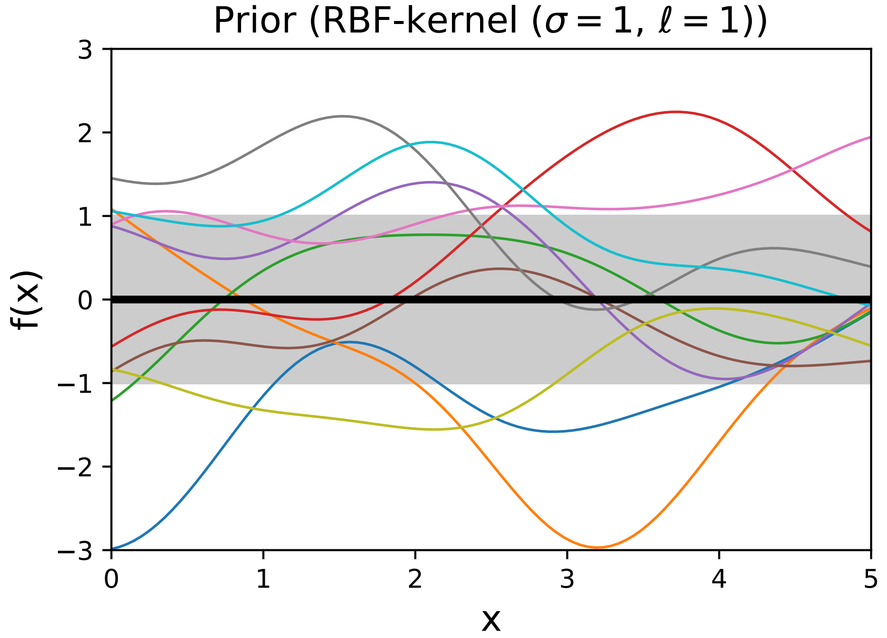
Gaussian processes for regression

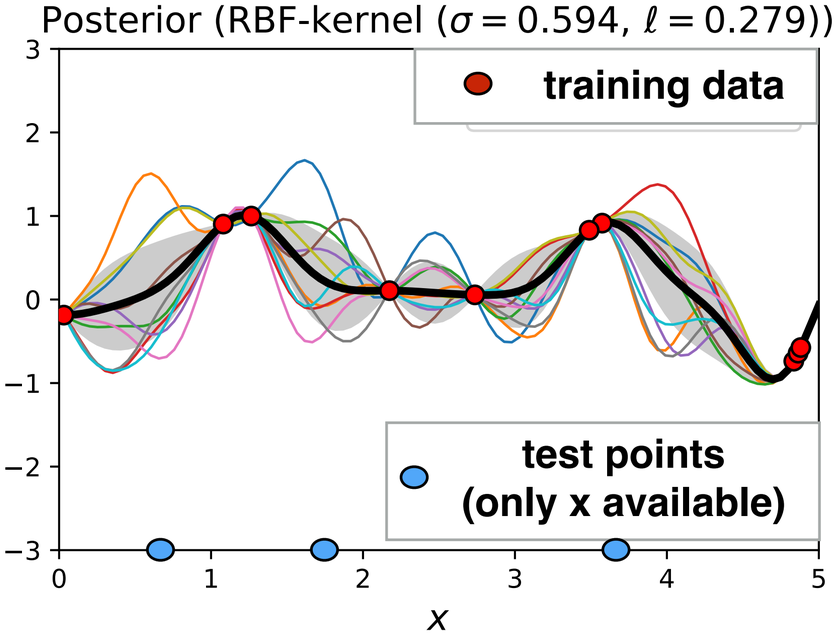
kernel function

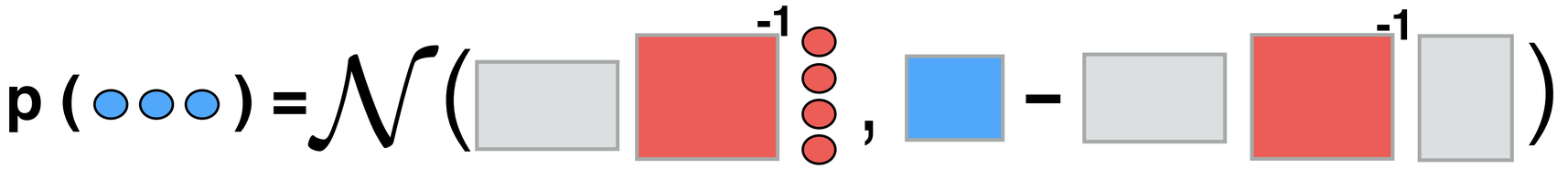
prior process over prediction functions

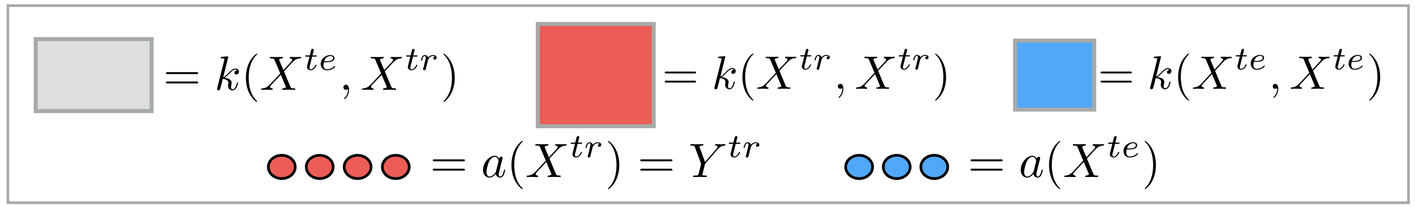
data

posterior process over prediction functions









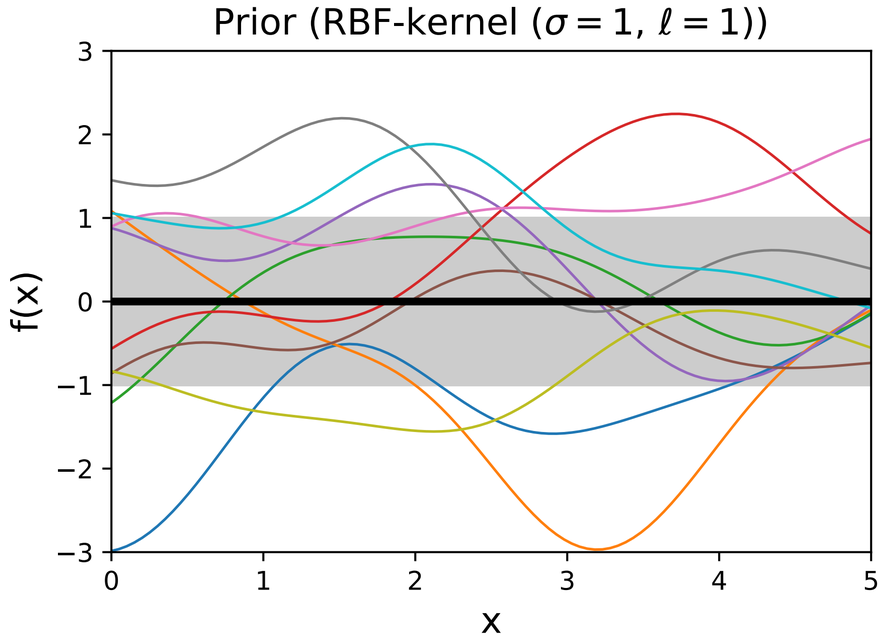
Gaussian processes for regression

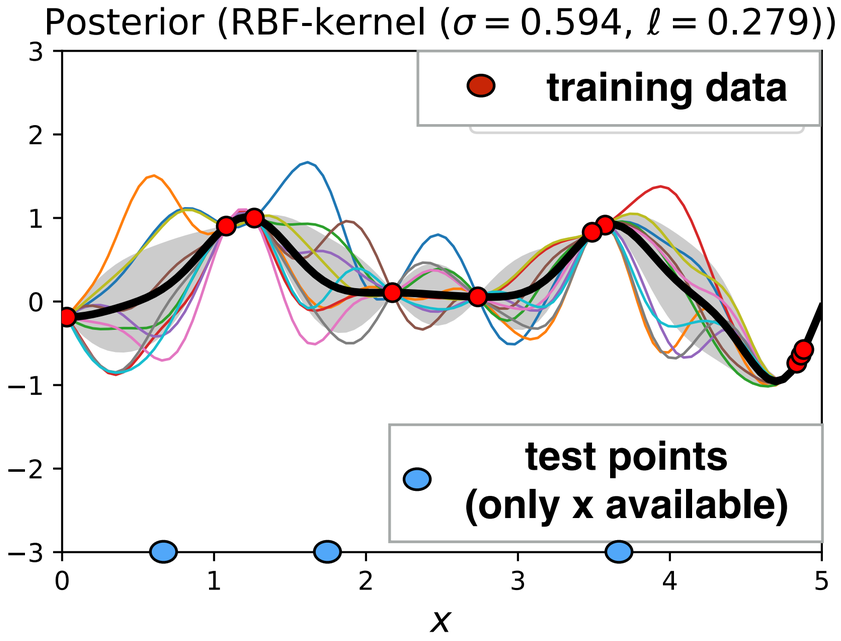
kernel function

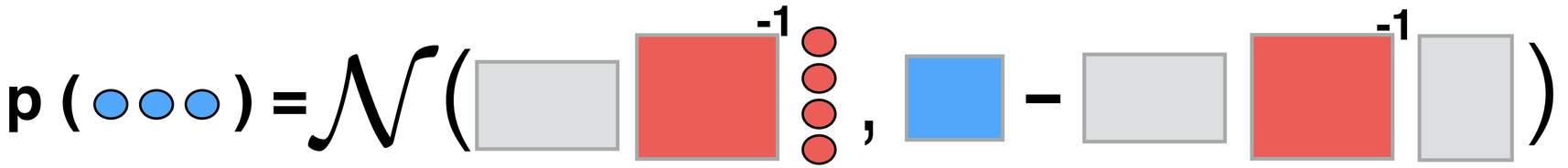
prior process over prediction functions

data

posterior process over prediction functions

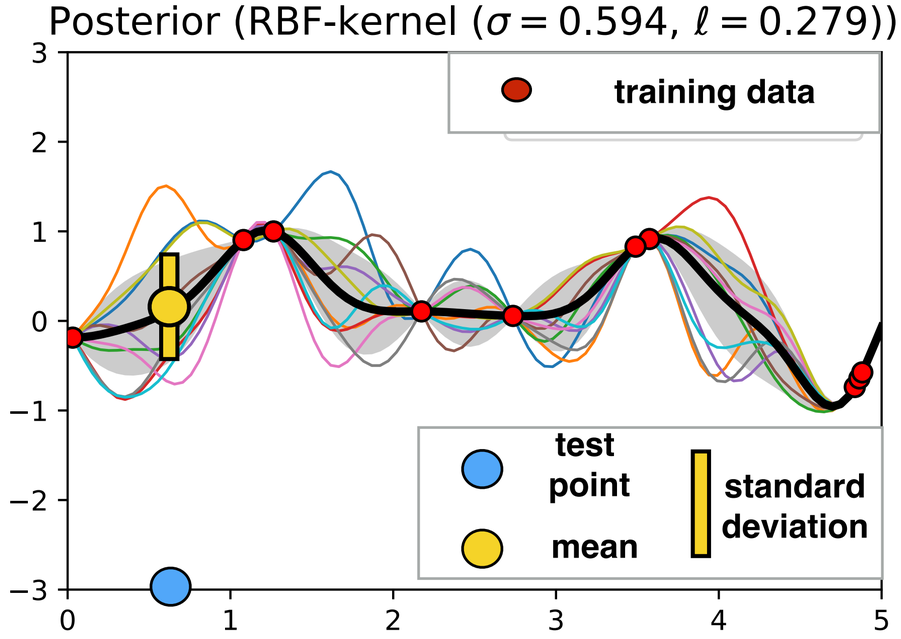






Training? **Prediction?**

Training and prediction in GP for regression



Prediction:

$$p(\circ) = \mathcal{N}\left(\; , \; - \; \right)$$

mean variance

$= k(x_*, X^{tr})$ $= k(X^{tr}, X^{tr})$ $= k(x_*, x_*)$

$= a(X^{tr}) = Y^{tr}$ $= a(x_*)$

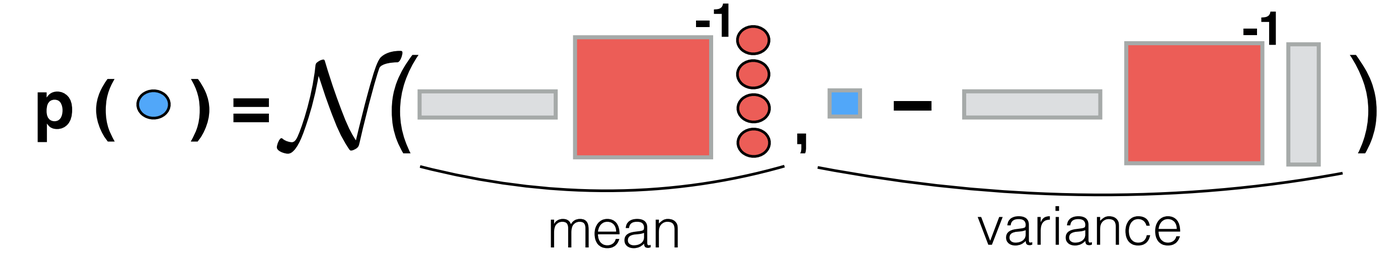
Training and prediction in GP for regression

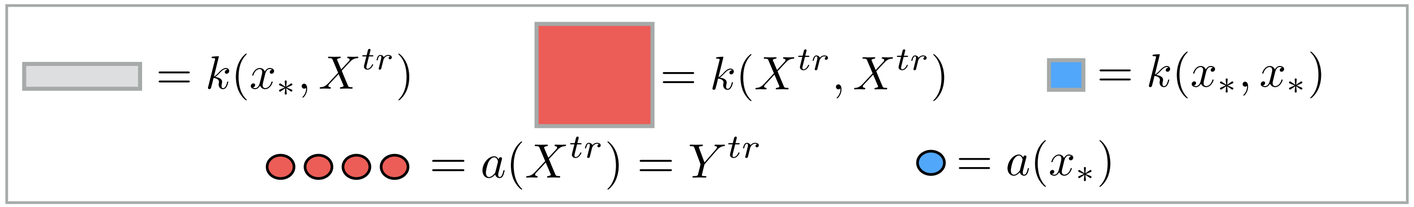
Training:

$$k(x, x') = x^T x' + \sigma_1^2 \exp\left(-\frac{||x - x'||^2}{2\ell^2}\right) + \sigma_2^2[x = x'] + \sigma_3^2$$

parameters of the kernel (covariance) function

Prediction:

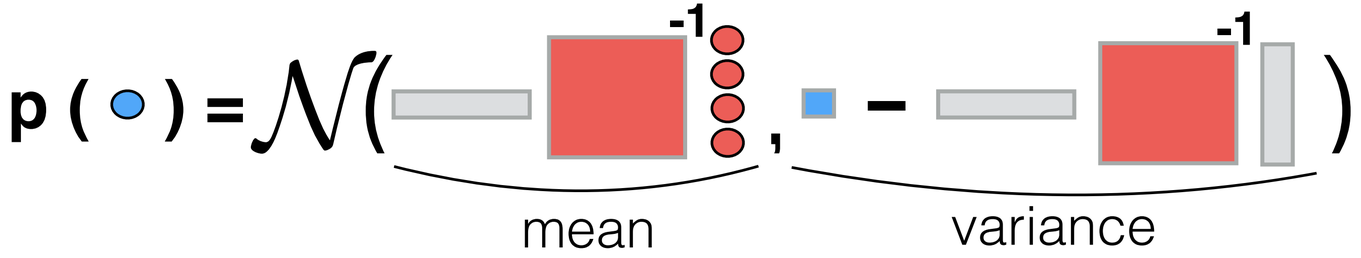




Training and prediction in GP for regression

$$p(a(x_*)) = \mathcal{N}(m_*, \sigma_*) \quad m_* = k_*^T K^{-1} Y, \quad \sigma_*^2 = k_{**} - k_*^T K^{-1} k_*$$
$$k_{**} = k(x_*, x_*), \quad k_* = \{k(x_i, x_*)\}_{i=1}^N, \quad K = \{k(x_i, x_j)\}_{i,j=1}^{N,N}, \quad Y = \{y_i\}_{i=1}^N$$

Prediction:



Exercise

- Consider we are given the following training data (1 feature):

x	y
-1.5	1
0.5	3
0.7	2.5

We use zero mean function and RBF-kernel: $k(x, x') = 0.5 \exp\left(-\frac{(x - x')^2}{2}\right)$

- What prediction will we make for a new object $x_* = 0?$ for $x_* = 3?$

Formulas:

$$p(a(x_*)) = \mathcal{N}(m_*, \sigma_*) \quad m_* = k_*^T K^{-1} Y, \quad \sigma_*^2 = k_{**} - k_*^T K^{-1} k_*$$

$$k_{**} = k(x_*, x_*), \quad k_* = \{k(x_i, x_*)\}_{i=1}^N, \quad K = \{k(x_i, x_j)\}_{i,j=1}^{N,N}, \quad Y = \{y_i\}_{i=1}^N$$

Exercise

x	y
-1.5	1
0.5	3
0.7	2.5

$$K = 0.5 \cdot \begin{bmatrix} 1 & \exp\left(-\frac{(-1.5-0.5)^2}{2}\right) & \exp\left(-\frac{(-1.5-0.7)^2}{2}\right) \\ \exp\left(-\frac{(-1.5-0.5)^2}{2}\right) & 1 & \exp\left(-\frac{(0.5-0.7)^2}{2}\right) \\ \exp\left(-\frac{(-1.5-0.7)^2}{2}\right) & \exp\left(-\frac{(0.5-0.7)^2}{2}\right) & 1 \end{bmatrix}$$

Formulas:

$$p(a(x_*)) = \mathcal{N}(m_*, \sigma_*) \quad m_* = k_*^T K^{-1} Y, \quad \sigma_*^2 = k_{**} - k_*^T K^{-1} k_*$$

$$k_{**} = k(x_*, x_*), \quad k_* = \{k(x_i, x_*)\}_{i=1}^N, \quad K = \{k(x_i, x_j)\}_{i,j=1}^{N,N}, \quad Y = \{y_i\}_{i=1}^N$$

Exercise

x	y
-1.5	1
0.5	3
0.7	2.5

$$K = 0.5 \cdot \begin{bmatrix} 1 & \exp\left(-\frac{(-1.5-0.5)^2}{2}\right) & \exp\left(-\frac{(-1.5-0.7)^2}{2}\right) \\ \exp\left(-\frac{(-1.5-0.5)^2}{2}\right) & 1 & \exp\left(-\frac{(0.5-0.7)^2}{2}\right) \\ \exp\left(-\frac{(-1.5-0.7)^2}{2}\right) & \exp\left(-\frac{(0.5-0.7)^2}{2}\right) & 1 \end{bmatrix}$$

For $x_* = 0$: $\quad k_* = 0.5 \cdot \begin{bmatrix} \exp\left(-\frac{(0+1.5)^2}{2}\right) & \exp\left(-\frac{(0-0.5)^2}{2}\right) & \exp\left(-\frac{(0-0.7)^2}{2}\right) \end{bmatrix} \quad k_{**} = \begin{bmatrix} 0.5 \end{bmatrix}$

Formulas:

$$p(a(x_*)) = \mathcal{N}(m_*, \sigma_*) \quad m_* = k_*^T K^{-1} Y, \quad \sigma_*^2 = k_{**} - k_*^T K^{-1} k_*$$

$$k_{**} = k(x_*, x_*), \quad k_* = \{k(x_i, x_*)\}_{i=1}^N, \quad K = \{k(x_i, x_j)\}_{i,j=1}^{N,N}, \quad Y = \{y_i\}_{i=1}^N$$

Exercise

x	y
-1.5	1
0.5	3
0.7	2.5

$$K = 0.5 \cdot \begin{bmatrix} 1 & \exp\left(-\frac{(-1.5-0.5)^2}{2}\right) & \exp\left(-\frac{(-1.5-0.7)^2}{2}\right) \\ \exp\left(-\frac{(-1.5-0.5)^2}{2}\right) & 1 & \exp\left(-\frac{(0.5-0.7)^2}{2}\right) \\ \exp\left(-\frac{(-1.5-0.7)^2}{2}\right) & \exp\left(-\frac{(0.5-0.7)^2}{2}\right) & 1 \end{bmatrix}$$

For $x_* = 0$: $\quad k_* = 0.5 \cdot \begin{bmatrix} \exp\left(-\frac{(0+1.5)^2}{2}\right) & \exp\left(-\frac{(0-0.5)^2}{2}\right) & \exp\left(-\frac{(0-0.7)^2}{2}\right) \end{bmatrix} \quad k_{**} = [0.5]$

$$\mu_* = \begin{bmatrix} 0.162 & 0.441 & 0.391 \end{bmatrix} \begin{bmatrix} 0.5 & 0.067 & 0.044 \\ 0.067 & 0.5 & 0.490 \\ 0.044 & 0.490 & 0.5 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 3 \\ 2.5 \end{bmatrix} \quad \sigma_*^2 = 0.5 - \begin{bmatrix} 0.162 & 0.441 & 0.391 \end{bmatrix} \begin{bmatrix} 0.5 & 0.067 & 0.044 \\ 0.067 & 0.5 & 0.490 \\ 0.044 & 0.490 & 0.5 \end{bmatrix}^{-1} \begin{bmatrix} 0.162 \\ 0.441 \\ 0.391 \end{bmatrix}$$

Formulas:

$$p(a(x_*)) = \mathcal{N}(m_*, \sigma_*) \quad m_* = k_*^T K^{-1} Y, \quad \sigma_*^2 = k_{**} - k_*^T K^{-1} k_*$$

$$k_{**} = k(x_*, x_*), \quad k_* = \{k(x_i, x_*)\}_{i=1}^N, \quad K = \{k(x_i, x_j)\}_{i,j=1}^{N,N}, \quad Y = \{y_i\}_{i=1}^N$$

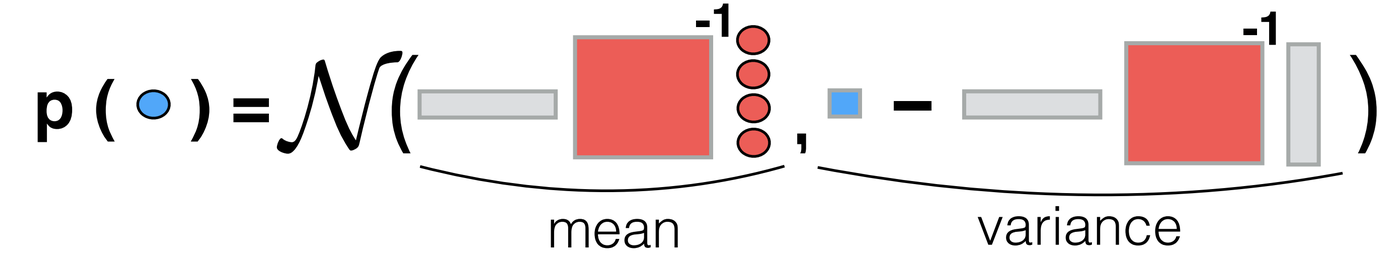
Training and prediction in GP for regression

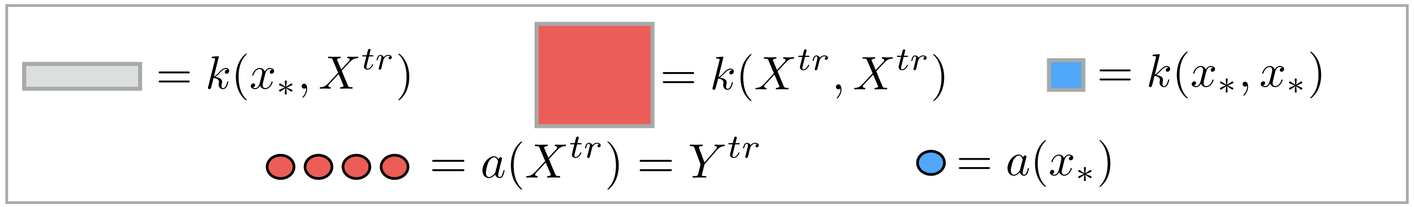
Training:

$$k(x,x') = x^T x' + \sigma_1^2 \exp\left(-\frac{||x-x'||^2}{2\ell^2}\right) + \sigma_2^2[x = x'] + \sigma_3^2$$

parameters of the kernel (covariance) function

Prediction:



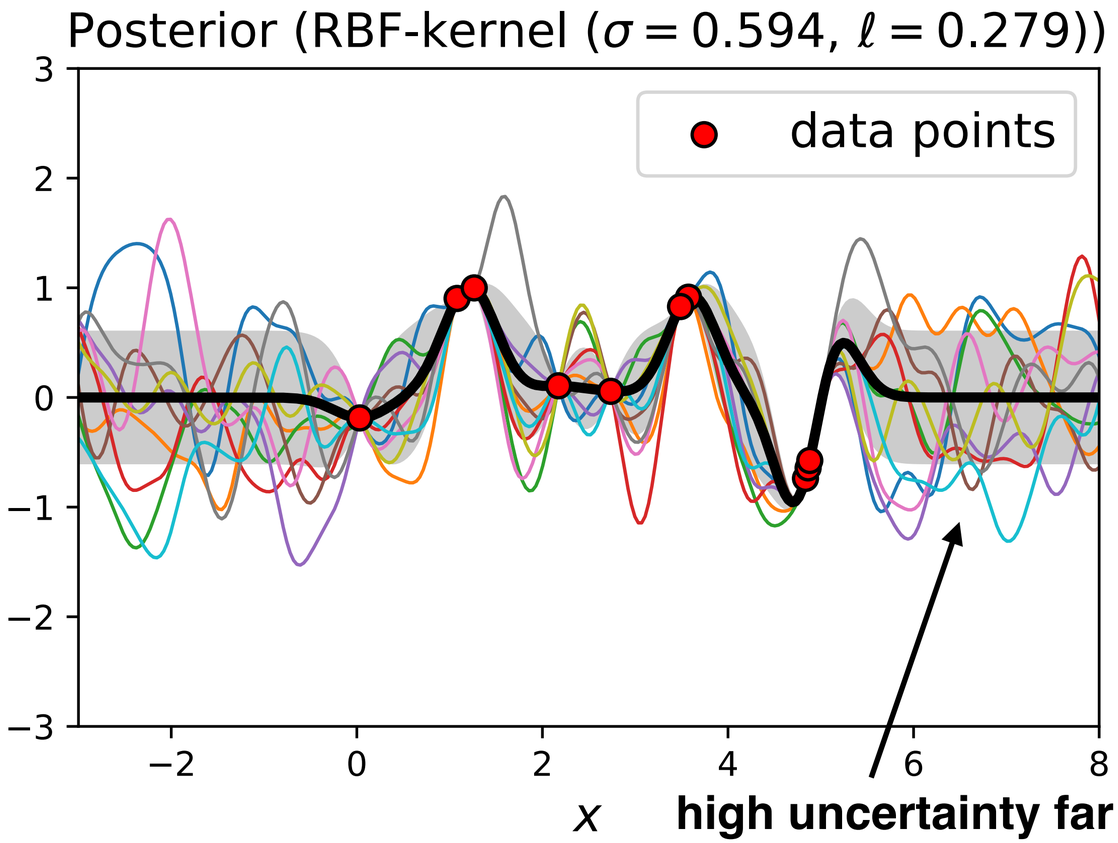


Parametric vs non-parametric models

	Parametric models:	**Non-parametric models:**
Prediction:	$a(x)$ – function of x and parameters θ	$a(x)$ – function of x and training data
Training:	finding θ based on training data	none (or tuning a small number of parameters)
Examples:	decision trees (Bayesian) linear regression	kNN Gaussian processes

Pros and cons of Gaussian processes

+ uncertainty estimation



high uncertainty far from training data

— kernel (covariance) function?

— slow computation

Training: O(N³)

$$p(\bullet\bullet\bullet) = \mathcal{N}(0, \blacksquare) \rightarrow \max_{\sigma_1, \sigma_2, \sigma_3, \ell}$$

Prediction: O(N) — mean, O(N²) — std

$$p(\bullet) = \mathcal{N}(\blacksquare^{-1}\bullet\bullet\bullet\bullet, \blacksquare - \blacksquare^{-1})$$

mean variance

N — number of training objects

Summary

- Gaussian process is a “distribution” over *functions*
- Regression with Gaussian Process generalizes kNN in a probabilistic manner
- Gaussian Processes provide reliable uncertainty estimates but require careful choice of kernel function and are slow in training and testing

Optimization of expensive to evaluate function

$$f(x) \to \min_x$$

computing $f(x)$
takes hours or more

Examples:

- choosing locations for extraction of minerals / oil / gold etc
- choosing hyperparameters of neural network (weight decay, dropout rate etc)
- generating new moleculas (optimizing w. r. t. molecula embedding)

Optimization of expensive to evaluate function

$$f(x) \to \min_x$$

computing $f(x)$
takes hours or more

Examples:

- choosing locations for extraction of minerals / oil / gold etc
- choosing hyperparameters of neural network (weight decay, dropout rate etc)
- generating new moleculas (optimizing w. r. t. molecula embedding)

General idea 1:
model $f(x)$ and
optimize model w.r.t. x

Optimization of expensive to evaluate function

$$f(x) \to \min_x$$

computing $f(x)$
takes hours or more

Examples:

- choosing locations for extraction of minerals / oil / gold etc
- choosing hyperparameters of neural network (weight decay, dropout rate etc)
- generating new moleculas (optimizing w. r. t. molecula embedding)

General idea 1:
model $f(x)$ and
optimize model w.r.t. x

imperfect model

General idea 2:
model both $f(x)$ and
uncertainty in its prediction

Gaussian process

Bayesian optimization

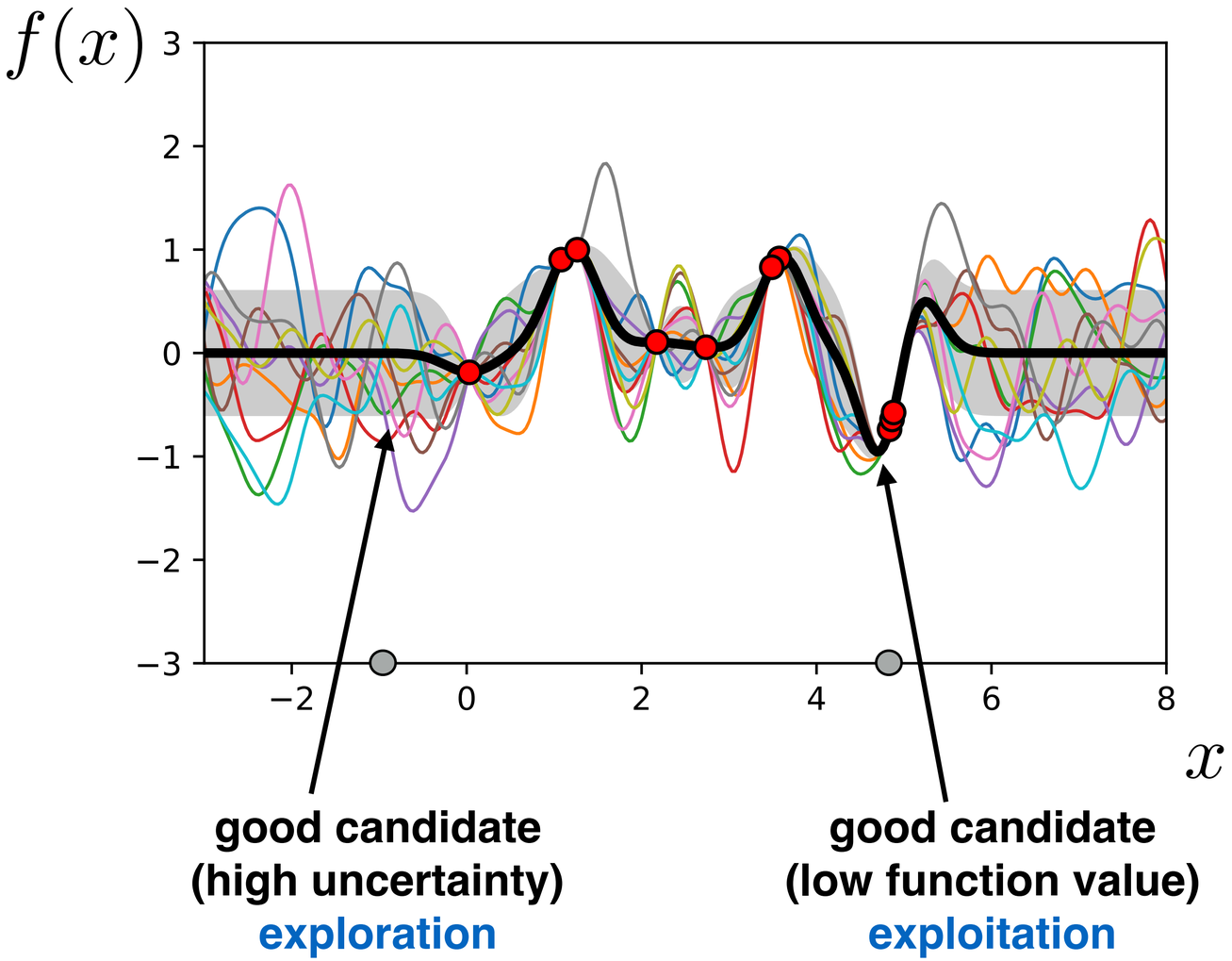
$$f(x) \to \min_x$$

points with known function value

predicted function value

uncertainty in prediction

Next point for computing function value?



Bayesian optimization

$$f(x) \to \min_x$$

computing $f(x)$
takes hours or more

Choosing a new point x_{n+1} for evaluation:

Given: $\mathcal{D}_n = \{(x_1, f(x_1)), \dots, (x_n, f(x_n))\}$ — set of currently known fun. values

1. Model $f(x)$ with a Gaussian process using $\mathcal{D}_n$:
   ability to compute $\mu(x)$, $\sigma(x)$ for any x

2. Acquaintance function: $\alpha(x) = \alpha(\mu(x), \sigma(x))$
   Optimize acquaintance function w. r. t. x : $x_{n+1} = \operatorname{argmin}_x \alpha(x)$

   Example: $\alpha(x) = \mu(x) - \beta\sigma(x)$ (β — hyperparameter)

Bayesian optimization

$$f(x) \to \min_x$$

computing $f(x)$
takes hours or more

Choosing a new point x_{n+1} for evaluation:

Given: $\mathcal{D}_n = \{(x_1, f(x_1)), \dots, (x_n, f(x_n))\}$ — set of currently known fun. values

1. Model $f(x)$ with a Gaussian process using $\mathcal{D}_n$:
   ability to compute $\mu(x)$, $\sigma(x)$ for any x

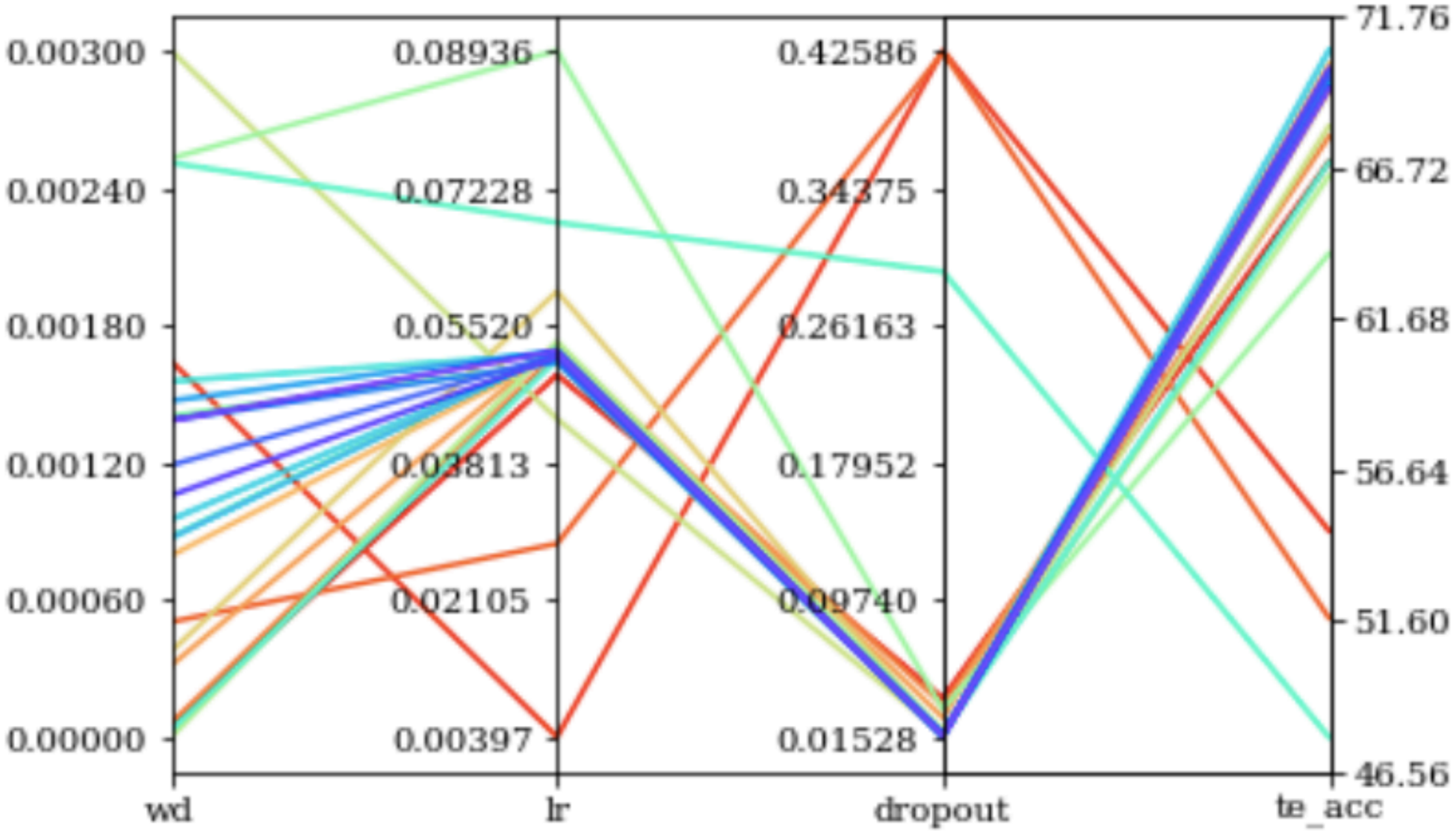
2. Acquaintance function: $\alpha(x) = \alpha(\mu(x), \sigma(x))$
   Optimize acquaintance function w. r. t. x : $x_{n+1} = \operatorname{argmin}_x \alpha(x)$

Repeat until time limit and choose $x_* = \underset{x = x_1, \dots, x_N}{\operatorname{argmin}} f(x)$

Bayesian optimization for hyperparameter search

WideResNet, CIFAR100, thin network
Choosing weight decay, learning rate, dropout

2 initial trials + 20 BO iterations
red: early iters **blue**: late iters

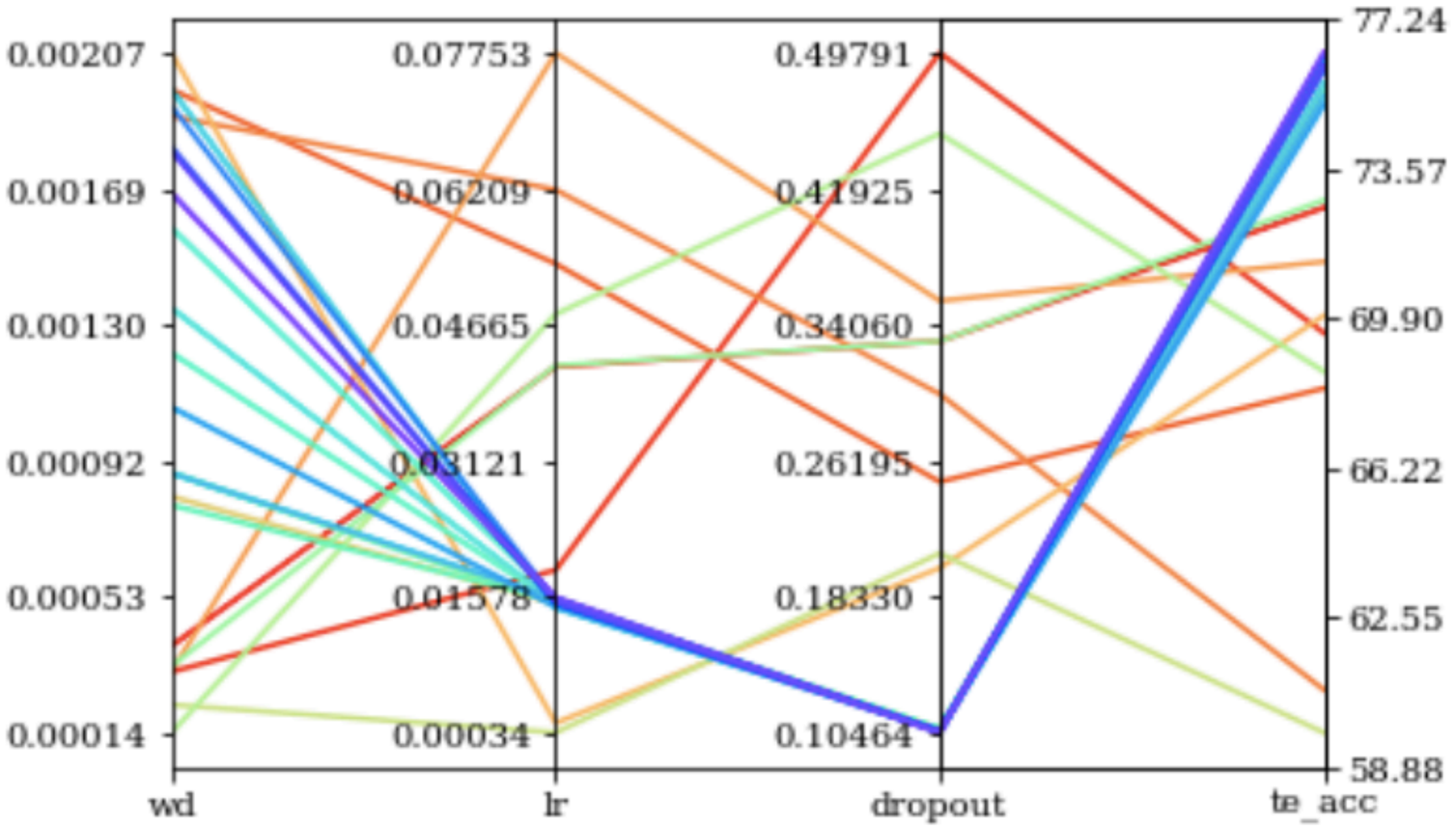


wd	lr	dropout	te_acc
5.614498e-05	0.049026	0.038425	67.68
1.642479e-03	0.003974	0.425863	54.14
5.029366e-04	0.028018	0.424888	50.94
6.099681e-05	0.051738	0.034468	68.60
3.129397e-04	0.052818	0.026868	70.36
7.926548e-04	0.051617	0.017005	71.26
3.743663e-04	0.059411	0.016053	70.28
3.000000e-03	0.043537	0.017029	68.98
1.000000e-07	0.053162	0.019330	67.16
2.529865e-03	0.089355	0.031902	64.30
1.404169e-03	0.050548	0.016009	70.44
8.739062e-04	0.052254	0.016042	70.60
2.509659e-03	0.067971	0.293746	46.56
2.973249e-05	0.050953	0.016744	67.66
1.554621e-03	0.051628	0.016651	70.50
9.497889e-04	0.051552	0.016384	71.76
8.709719e-04	0.051907	0.016620	70.76
1.469804e-03	0.052046	0.015283	70.88
1.384762e-03	0.050585	0.016426	70.72
1.187798e-03	0.051475	0.015822	70.82
1.055963e-03	0.051222	0.016678	71.10
1.383532e-03	0.052000	0.015923	70.36

Bayesian optimization for hyperparameter search

WideResNet, CIFAR100, thin network
Choosing weight decay, learning rate, dropout

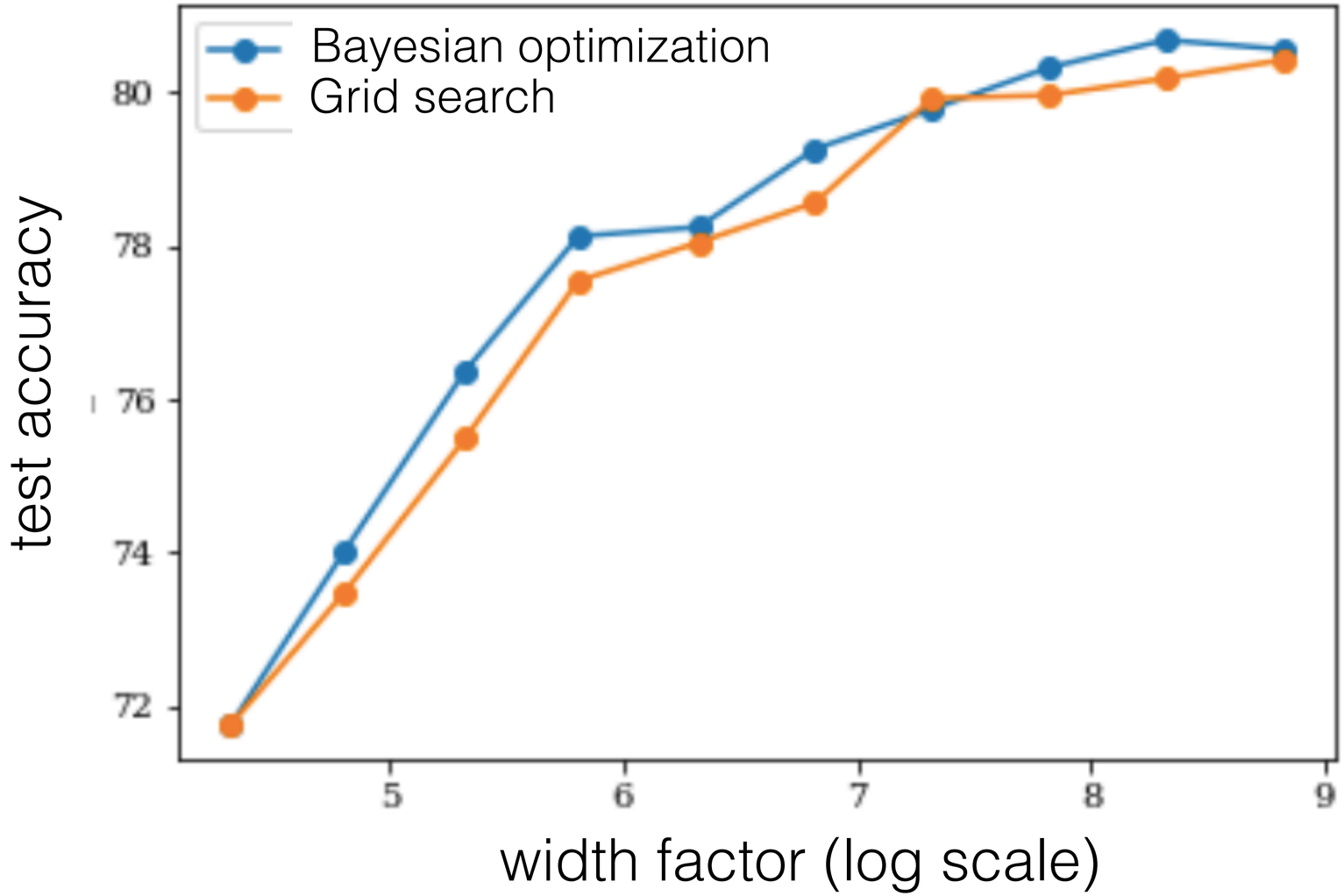
2 initial trials + 20 BO iterations
red: early iters **blue**: late iters



wd	lr	dropout	te_acc
0.000390	0.041890	0.331147	73.06
0.000317	0.018800	0.497906	69.68
0.001971	0.053580	0.249726	68.18
0.001895	0.061989	0.300542	60.04
0.000319	0.077525	0.354442	71.60
0.002075	0.001431	0.199903	70.14
0.000814	0.014996	0.106773	76.14
0.000223	0.000341	0.208378	58.88
0.000144	0.047858	0.451247	68.64
0.000333	0.042020	0.331092	73.26
0.000882	0.015054	0.106817	76.16
0.000790	0.015362	0.106535	76.36
0.001222	0.015142	0.106228	76.84
0.001577	0.015526	0.106035	76.80
0.001350	0.015079	0.105402	77.06
0.001970	0.014531	0.105515	76.40
0.000881	0.015591	0.105086	76.10
0.001068	0.014617	0.105467	75.84
0.001923	0.015366	0.105173	77.16
0.001808	0.015190	0.105586	76.76
0.001795	0.015050	0.104644	76.90
0.001677	0.015691	0.104903	77.24

Bayesian optimization for hyperparameter search

WideResNet, CIFAR100,
Choosing weight decay, learning rate, dropout for different widths



Bayesian methods for supervised learning: overview

Bayesian methods for supervised learning: overview

Model type

Parametric

Model definition:

$p(y|x, \theta)$ — likelihood

$p(\theta)$ — prior distribution over model parameters

Non-parametric

Model definition:

$p(y|f(x))$ — likelihood

$p(f(x))$ — prior process

Bayesian methods for supervised learning: overview

Model type

Parametric

Model definition:

$p(y|x, \theta)$ — likelihood

$p(\theta)$ — prior distribution over model parameters

Examples:

Bayesian linear regression:

$$p(y|x, w) = \mathcal{N}(y|w^T x, \beta)$$
$$p(w) = \mathcal{N}(w|0, \alpha I)$$

Sparse variational dropout (Bayesian neural net.)

$$p(y|x, w) = \text{Mult}(y|\text{NN}(x, w))$$
$$p(w) = \prod_{w_i} \frac{1}{|w_i|}$$

Non-parametric

Model definition:

$p(y|f(x))$ — likelihood

$p(f(x))$ — prior process

Gaussian process regression:

$$p(y|f(x)) = \delta(f(x))$$
$$p(f(x)) = GP(f(x)|\mu(x), k(x, x'))$$

or $p(y|f(x)) = \mathcal{N}(y|f(x), \beta)$

Bayesian methods for supervised learning: overview

X, Y — training data

Model type

Parametric

Model definition:

$p(y|x, \theta)$ — likelihood

$p(\theta)$ — prior distribution over model parameters

Training:

Finding posterior $p(\theta|X, Y)$

Prediction:

$$p(y_{new}|x_{new}, X, Y) = \int p(y_{new}|x_{new}, \theta)p(\theta|X, Y)d\theta$$

Non-parametric

Model definition:

$p(y|f(x))$ — likelihood

$p(f(x))$ — prior process

Training:

None or
Finding parameters of
the prior process

Prediction:

finding
$p(y_{new}|x_{new}, X, Y)$
directly from data X, Y

Bayesian methods for supervised learning: overview

X, Y — training data

Model type

Parametric

Non-parametric

lecture 5

Model definition:

$p(y|x, \theta)$ — likelihood

$p(\theta)$ — prior distribution over model parameters

Inference type (training)

Full Bayesian inference

$$p(\theta|X, Y) = \frac{p(Y|X, \theta)p(\theta)}{\int p(Y|X, \tilde{\theta})p(\tilde{\theta})d\tilde{\theta}}$$

lectures 1, 2

Maximum posterior

$$\theta_{MAP} = \text{argmax}_\theta p(\theta|X, Y) = \text{argmax}_\theta p(Y|X, \theta)p(\theta)$$

$$p(\theta|X, Y) \approx \delta(\theta_{MAP})$$

lectures 1, 2

Bayesian methods for supervised learning: overview

X, Y — training data

Model type

Parametric

Non-parametric

lecture 5

Inference type (training)

Full Bayesian inference

$p(\theta|X,Y)$

from Bayes formula

lectures 1, 2

Variational inference

$p(\theta|X,Y) \approx q(\theta|\lambda)$

lecture 3

MCMC sampling

$\hat{\theta}_1, \ldots, \hat{\theta}_K \sim$

$\sim p(\theta|X,Y)$

lecture 4

Maximum posterior

$p(\theta|X,Y) \approx \delta(\theta_{MAP})$

lectures 1, 2

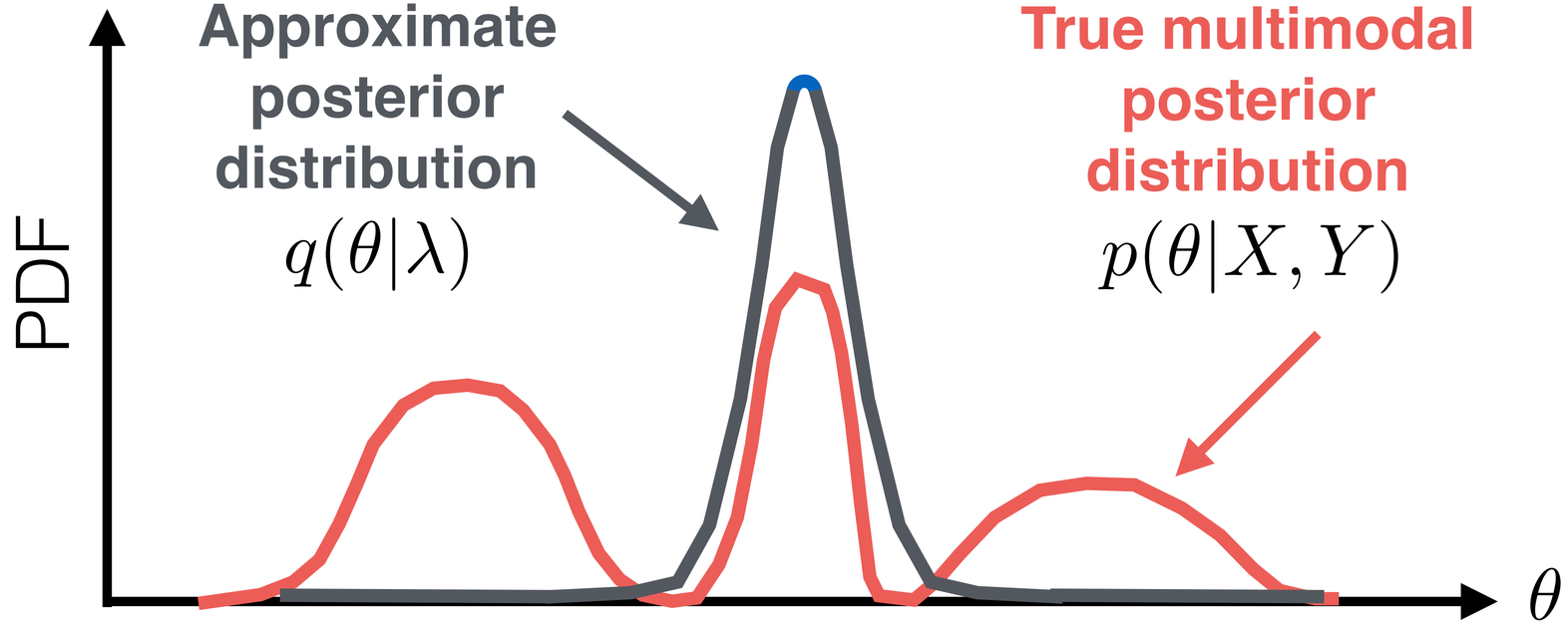
$p(y|x,\theta)$ — likelihood

$p(\theta)$ — prior

Variational inference

equivalent task

$$KL(q(\theta|\lambda)||p(\theta|X,Y)) \to \min_{\lambda}$$



$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(\theta|\lambda)} \log p(y^i|x^i,\theta)}_{\text{Data term}} - \underbrace{KL(q(\theta|\lambda)||p(\theta))}_{\text{Regularizer}} \to \max_{\lambda}$$

$p(y|x,\theta)$ — likelihood

$p(\theta)$ — prior

MCMC Sampling

$$p(y_*|x_*, X, Y) = \mathbb{E}_{p(\theta|X,Y)} p(y_*|x_*, \theta) \approx \frac{1}{K} \sum_{k=1}^{K} p(y_*|x_*, \theta^k)$$

$$\theta^k \sim p(\theta|X, Y)$$

We only need to know how to **sample** from the **unnormalized** posterior!

unnormalized posterior

$$p(\theta|X, Y) = \frac{p(Y|X, \theta)p(\theta)}{\int p(Y|X, \tilde{\theta})p(\tilde{\theta})d\tilde{\theta}}$$

Bayesian methods for supervised learning: overview

X, Y — training data

Model type

Parametric

Non-parametric

lecture 5

Inference type (training)

Full Bayesian inference

Variational inference

MCMC sampling

Maximum posterior

$p(\theta|X,Y)$

from Bayes formula

lectures 1, 2

:) all pros of Bayes. methods

:(may be inapplicable

$p(\theta|X,Y) \approx q(\theta|\lambda)$

lecture 3

:) frequently used in practice

:(lose some information from posterior

$\hat{\theta}_1, \ldots, \hat{\theta}_K \sim$

$\sim p(\theta|X,Y)$

lecture 4

:) sometimes used in practice

:(many samples needed => very slow

$p(\theta|X,Y) \approx \delta(\theta_{MAP})$

lectures 1, 2

:(always applicable

:(lose almost all pros of Bayes. methods