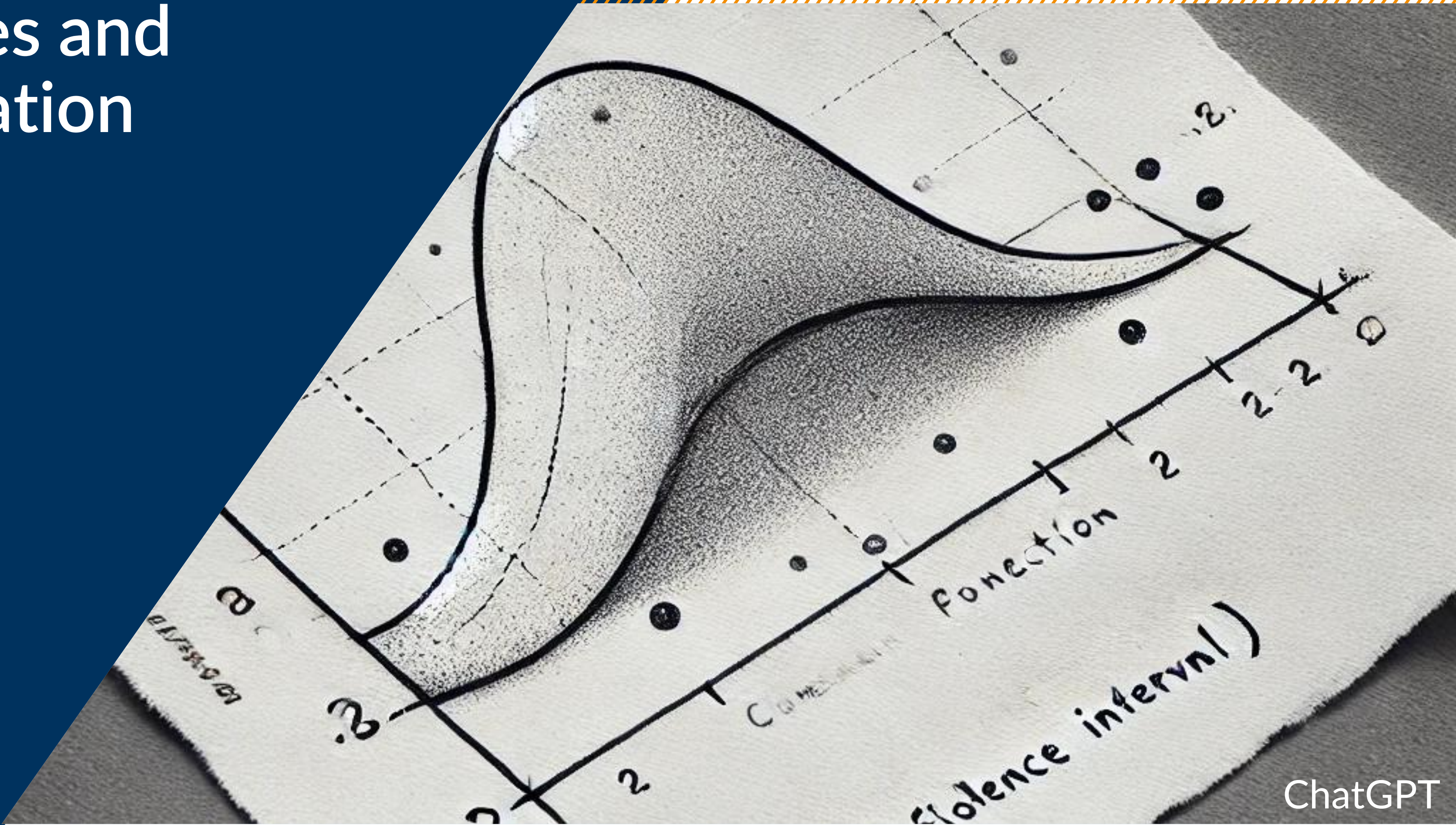
Leibniz MMS AI School 2025
Oct. 6th to 9th, 2025, Leipzig, Germany

IOM
Leibniz Institute of Surface Engineering

Gaussian Processes and Bayesian Optimization

- an introduction to core ideas and practical implementation



ChatGPT

Martin Rudolph

Leibniz Institute of Surface Engineering (IOM)
Modelling and Simulation

Agenda

- 0 Challenges in Process Optimization — 11h START
- 1 Gaussian Process Intuition
- 2 GP Regression: from prior to posterior
  - Stochastic process
  - Bayesian conditioning / updating

--------------------------------------- 13h LUNCH

  - Log marginal likelihood

--------------------------------------- 16h COFFEE

- 3 Bayesian Optimization

--------------------------------------- 19h FESTIVAL OF LIGHTS

Opening comments

- Non-mathematical introduction to GP and BO
- Focus on using the GP/BO in scikit-learn for efficiently building response surfaces
  - Lab experiments
  - Computer simulations
  - Not included: hyperparameter optimization in ML
- Learn the concepts of GP and BO
  - Includes looking at equations, abbreviations are marked by 
  - Point out the differences to other ML techniques
  - Practical sessions with (limited) code to illustrate the process in general
- Big group of students, one trainer
- Learning units: Lecture + Code examples + Practical session + Discussion

0 Challenges in process optimization

Challenges in process optimization

delonghi.de

Challenges in process optimization

- large number of possible process parameter combinations
  - grain size
  - water temperature
  - amount of powder
  - brew duration
  - weather (!!)
  - …

5 parameter with 10 variations each:
10^5 = 100,000 possible combinations
→ „**combinatorial explosion**“

delonghi.de

Challenges in process optimization

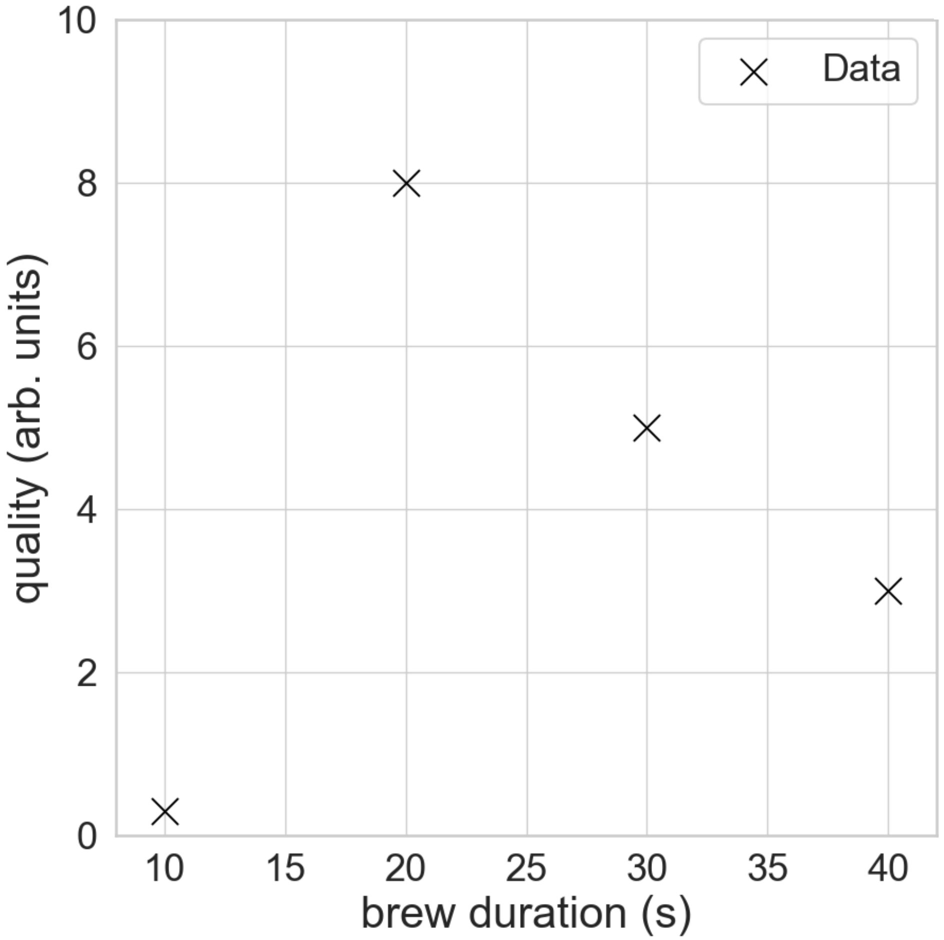
- large number of possible process parameter combinations
  - grain size
  - water temperature
  - amount of powder
  - brew duration
  - weather (!!)
  - ...

5 parameter with 10 variations each:
10^5 = 100,000 possible combinations
→ „**combinatorial explosion**"

delonghi.de

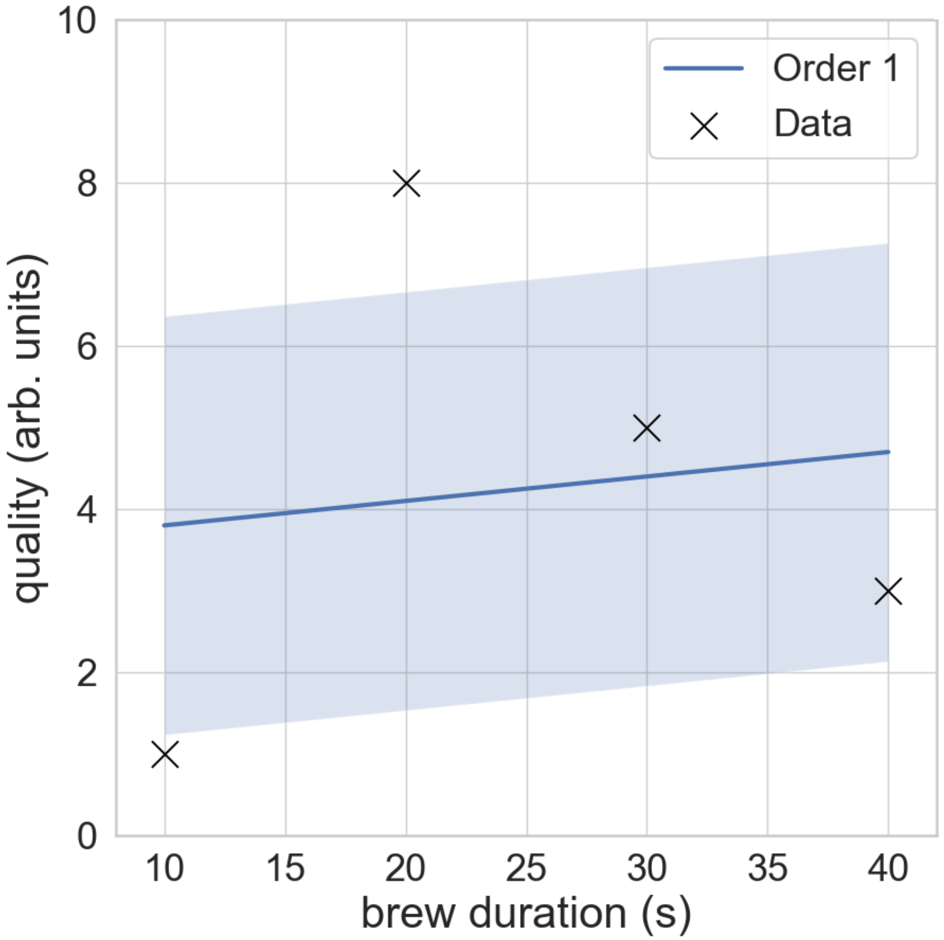
Response surfaces using polynomials

- Single parameter: brew duration



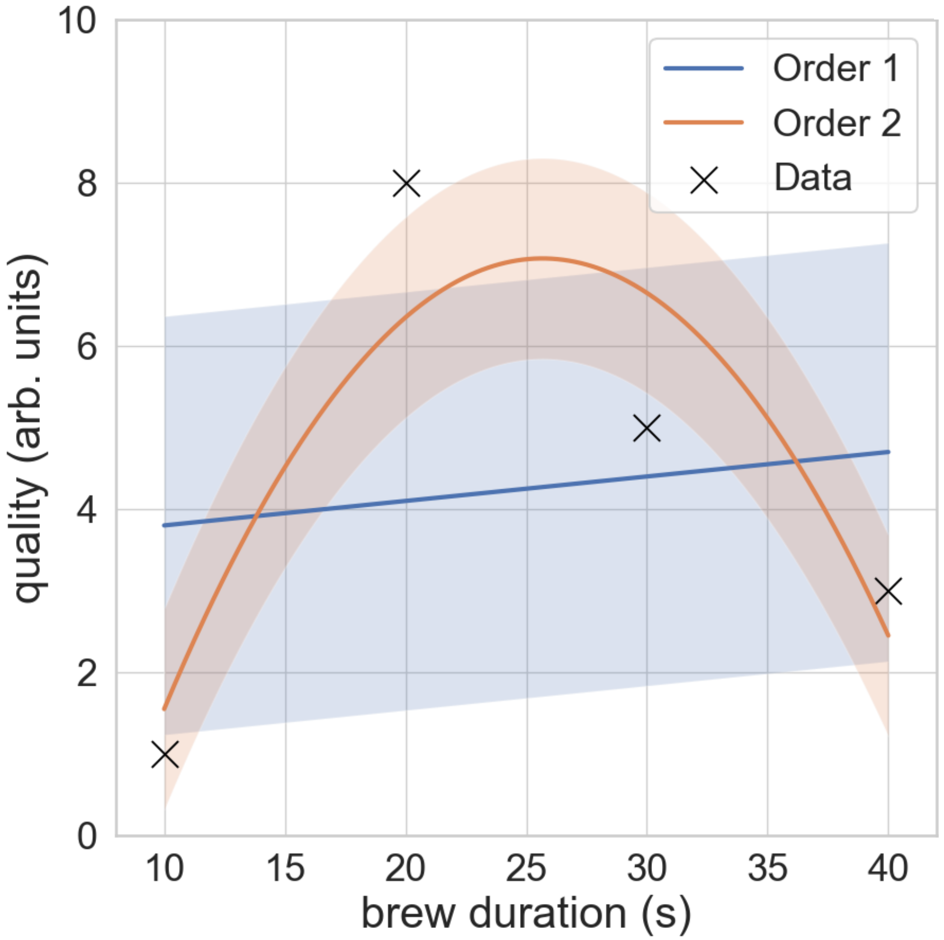
Response surfaces using polynomials

- Single parameter: brew duration



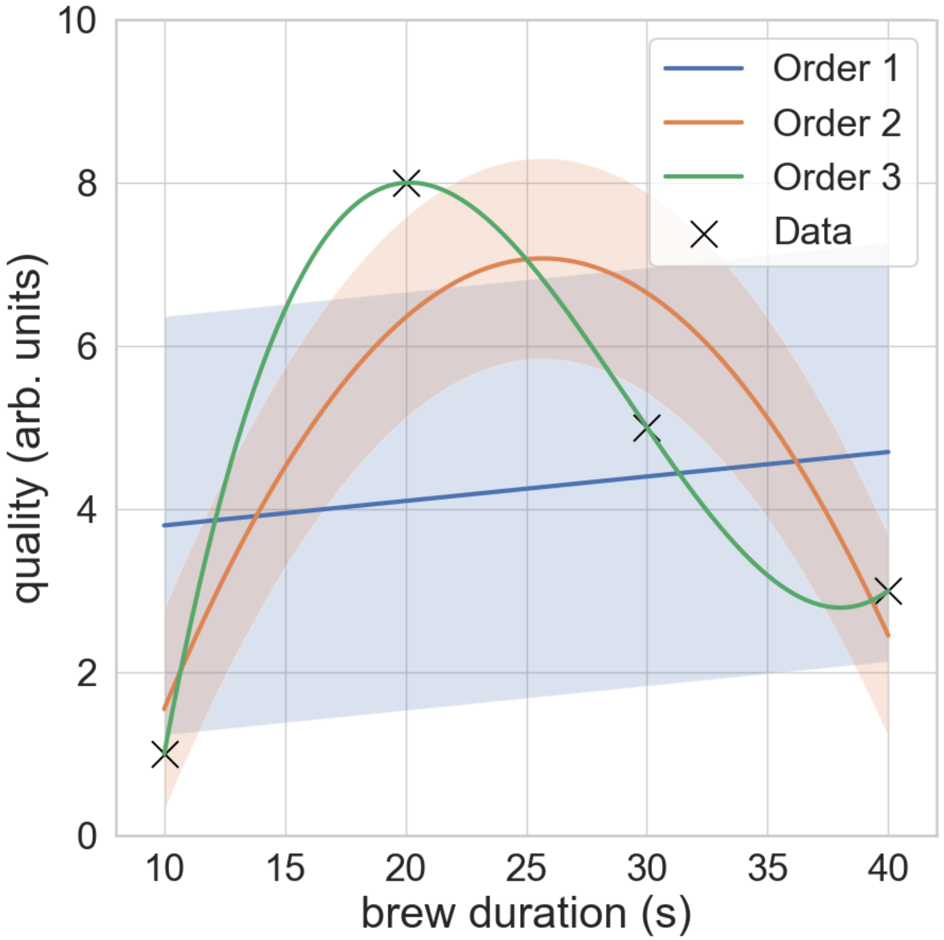
Response surfaces using polynomials

- Single parameter: brew duration



Response surfaces using polynomials

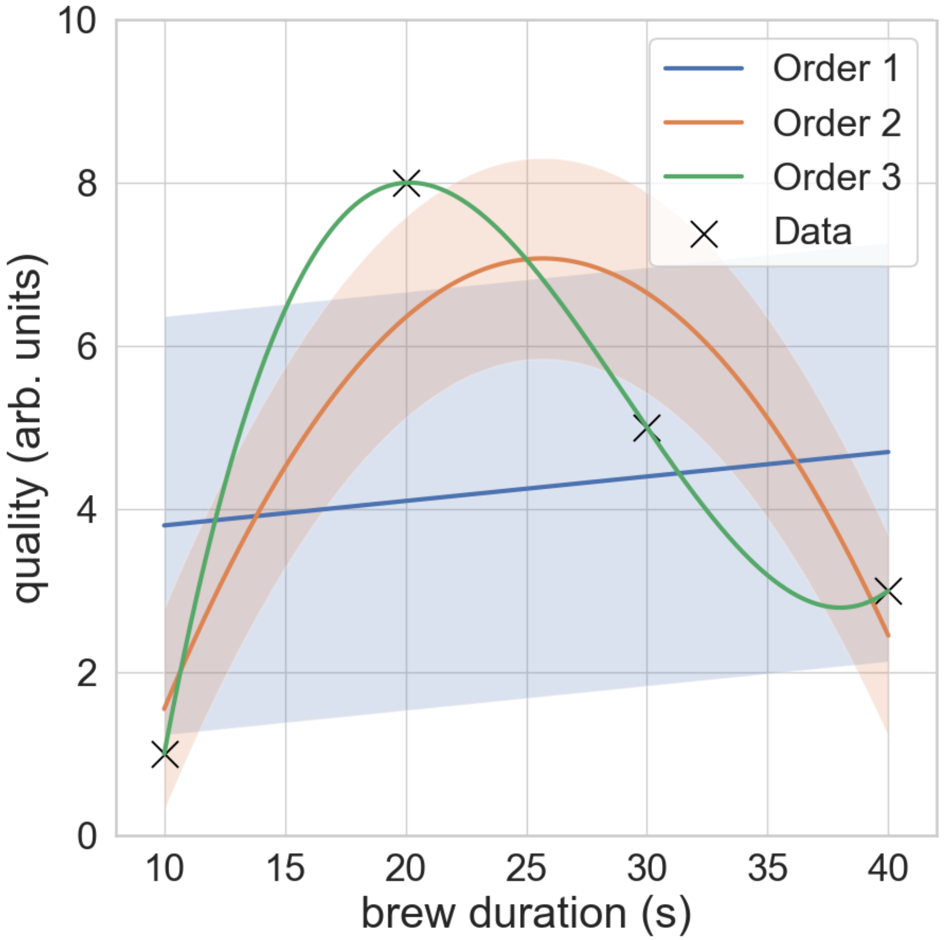
- Single parameter: brew duration



Response surfaces using polynomials

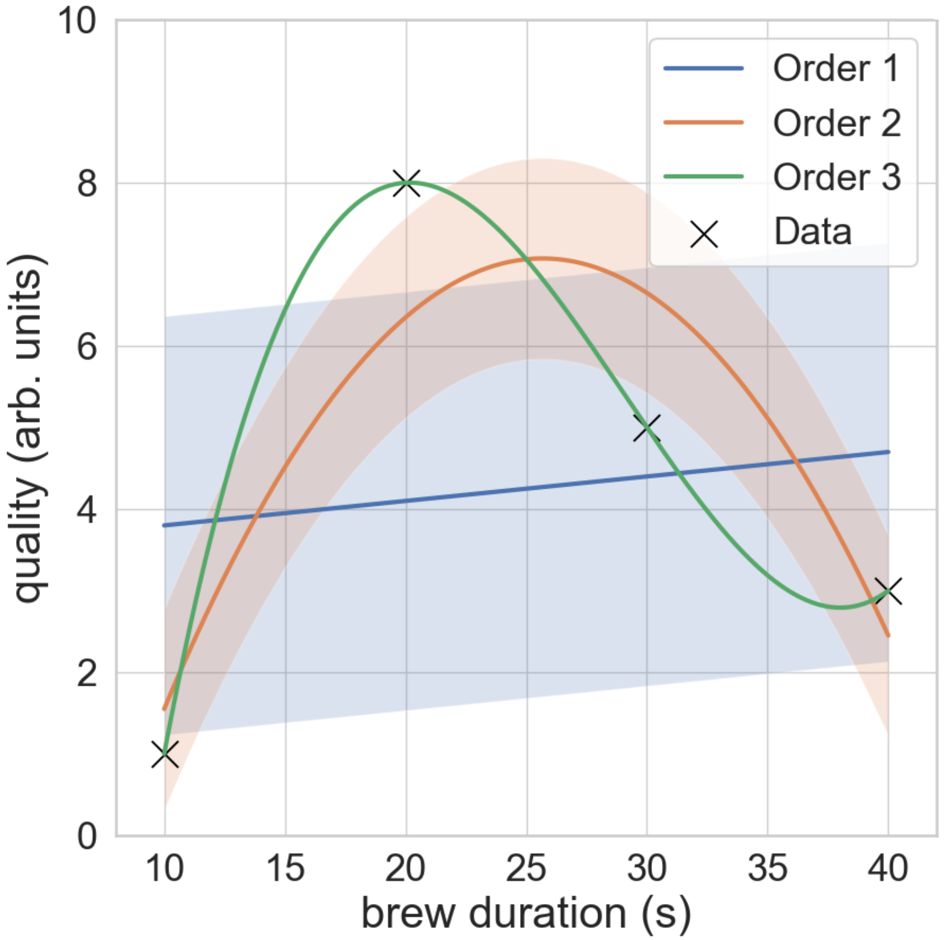
- Single parameter: brew duration

- Limitations:
  - Need to choose a (rigid) model
  - Constant uncertainty (even in the vicinity of data points)
  - Perfect fit (no uncertainty) for 3rd order polynomial



Response surfaces using polynomials

- Single parameter: brew duration
- Limitations:
  - Need to choose a (rigid) model
  - Constant uncertainty (even in the vicinity of data points)
  - Perfect fit (no uncertainty) for 3rd order polynomial



- Wishlist:
  - No need to choose a rigid model
  - Uncertainty measure that lives up to its name

1 Looking at a Gaussian Process

Danie Krige

Danie Krige
1919-2013

1938 B.Sc. Mining Engineering

1938 Anglovaal: sampling and valuation of gold in the ground

> *„decisions on new gold mines of critical importance to the State […] were being taken on a limited number of drillholes,* ***without any scientific analysis*** *of the risks of failure“ **

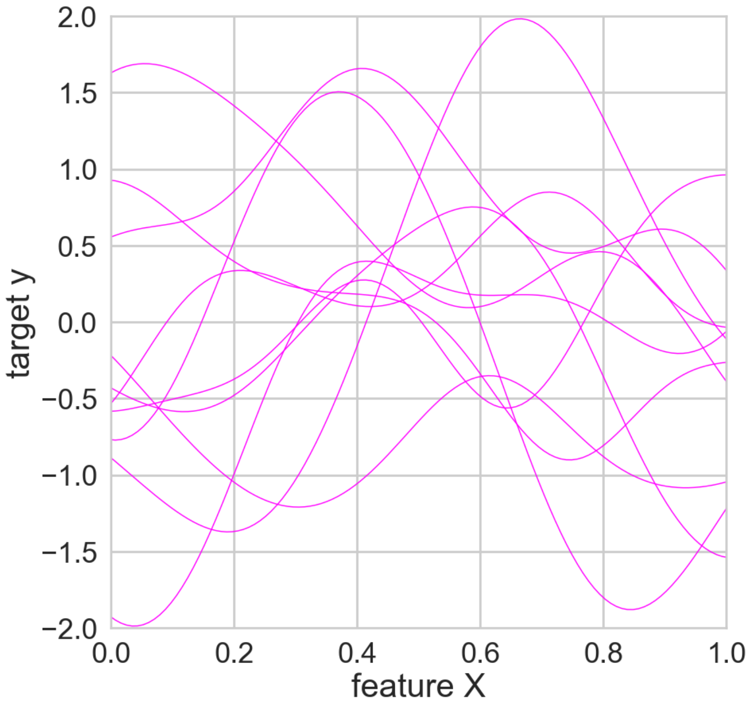
1943 Government Mining Engineer

> *Krige, Danie G. (1951). "A statistical approach to some basic mine valuation problems on the Witwatersrand". J. of the Chem., Metal. and Mining Soc. of South Africa.* ***52*** *(6): 119–139.*
> *(today cited 4600 times)*

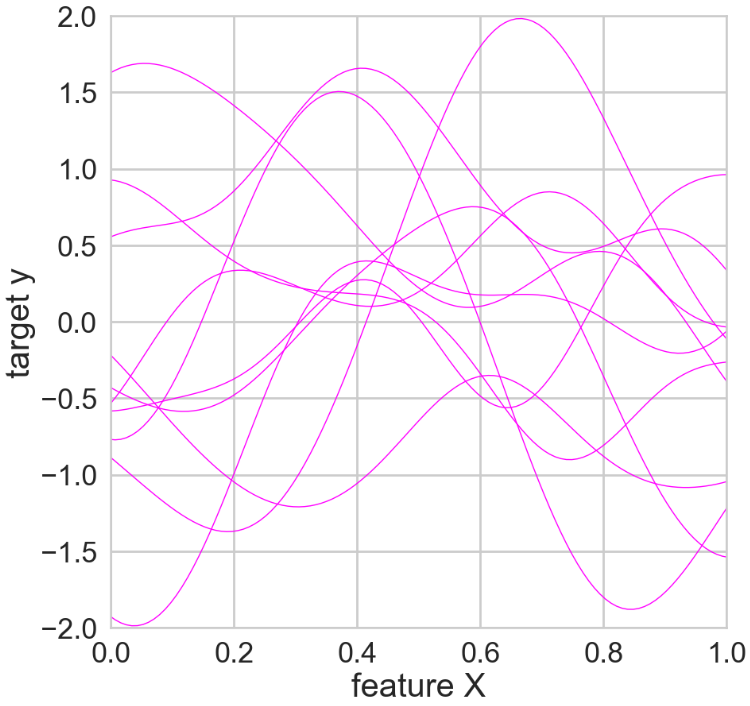
1951 Anglovaal: application of „kriging“ of ore reserves

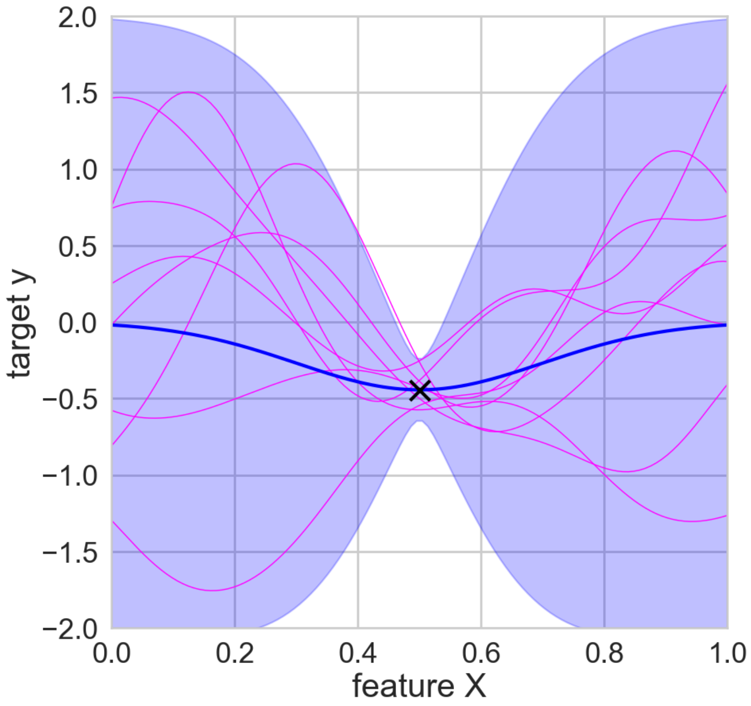
1981 Witwatersrand University: Professor of Mineral Economics

Gaussian Process in action

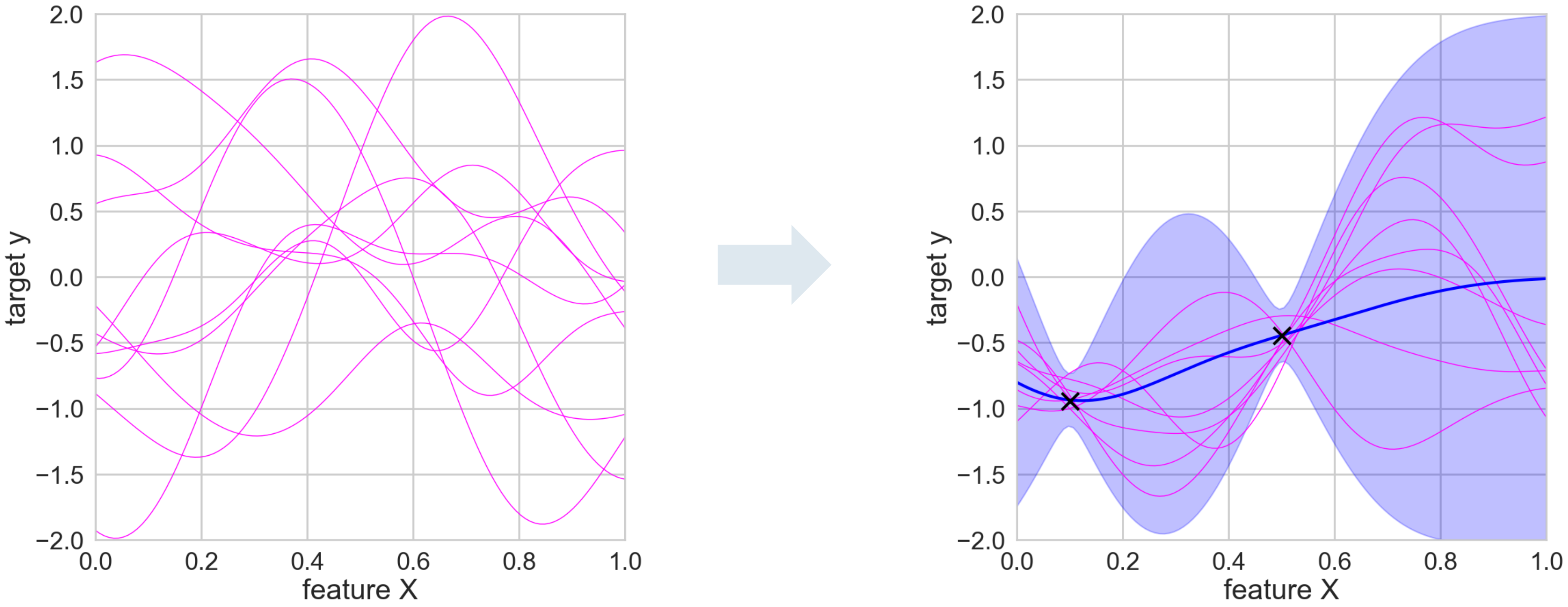


Gaussian Process in action

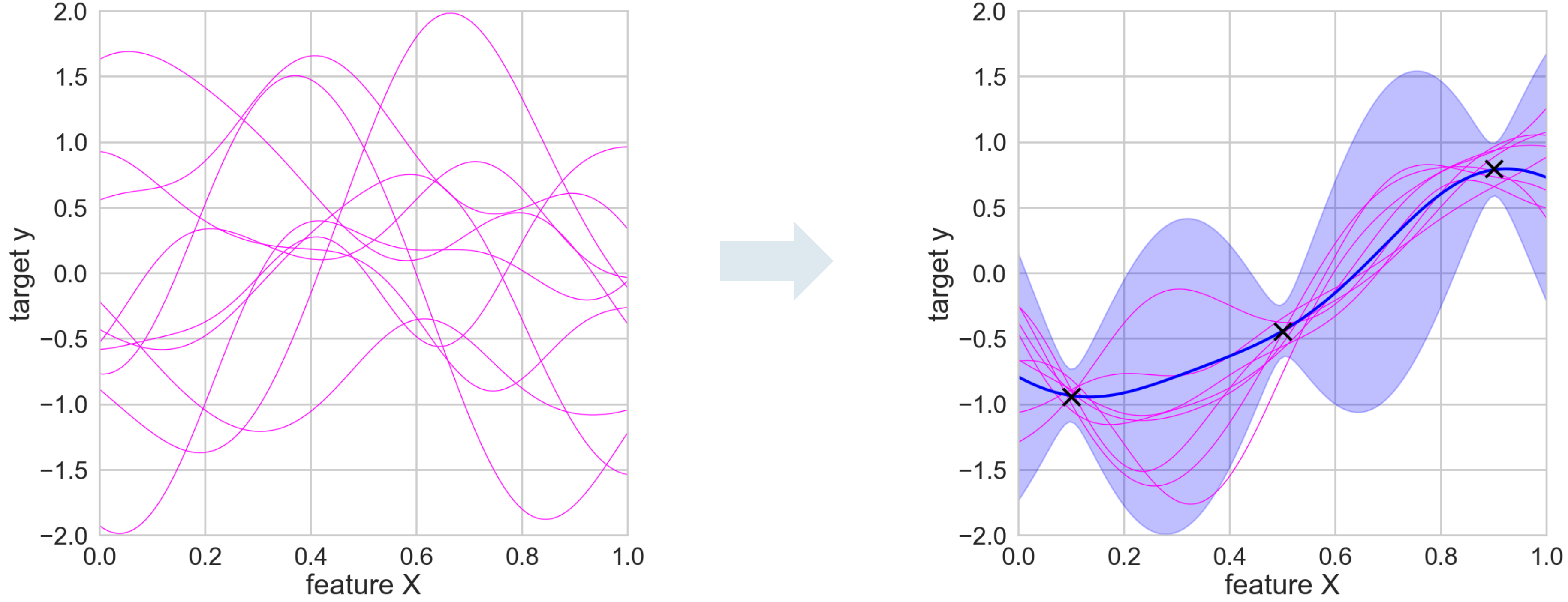




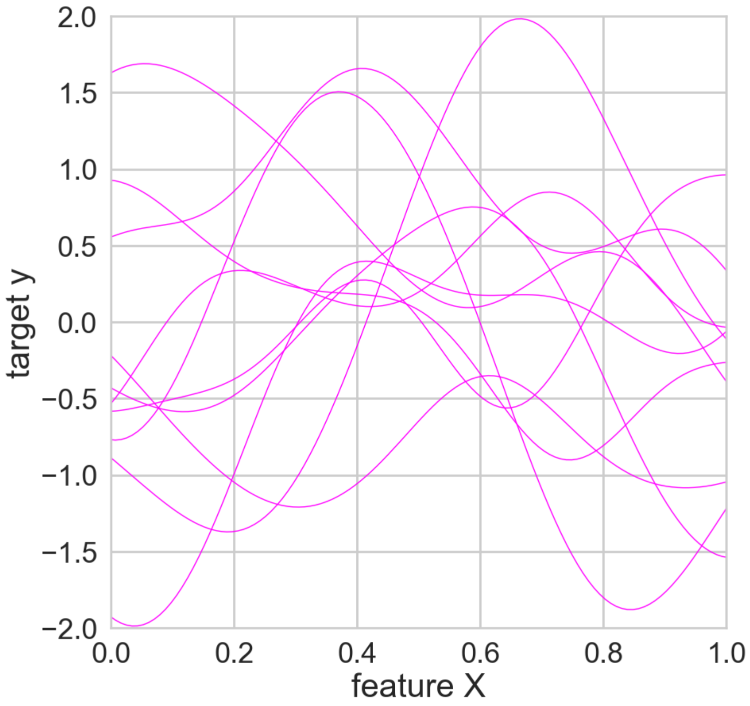
Gaussian Process in action

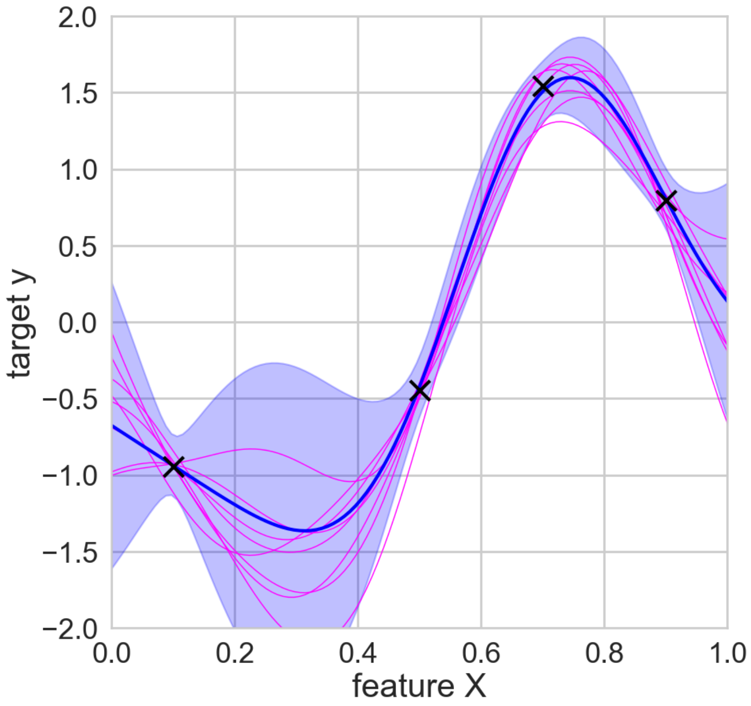


Gaussian Process in action

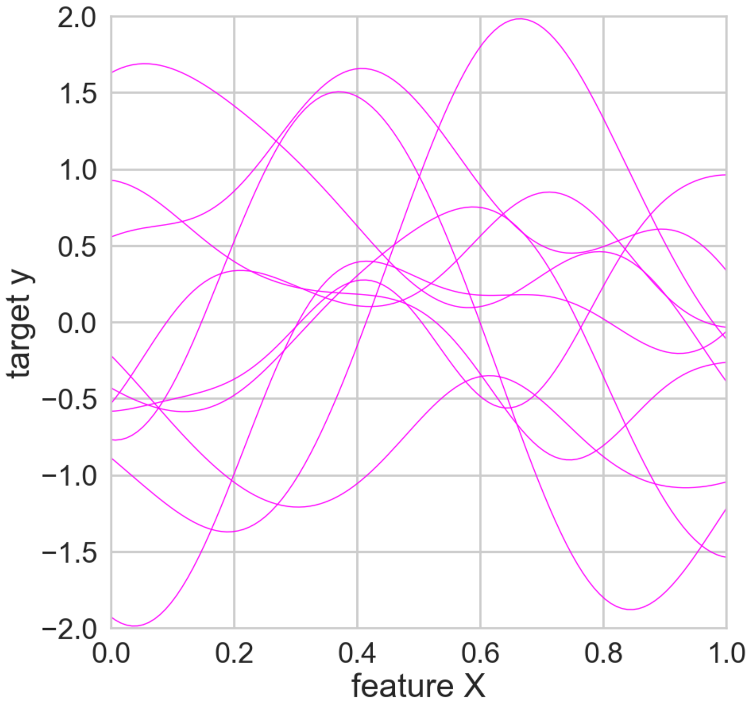


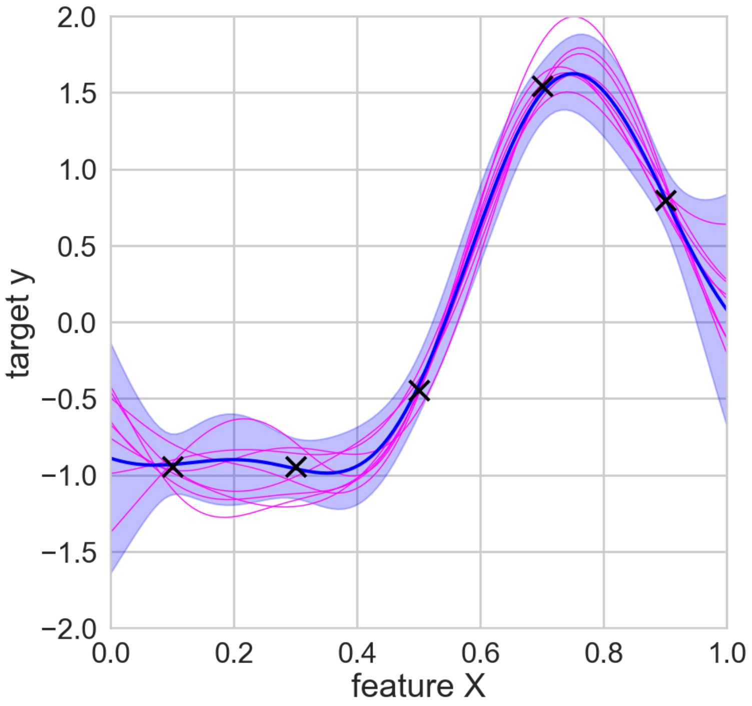
Gaussian Process in action





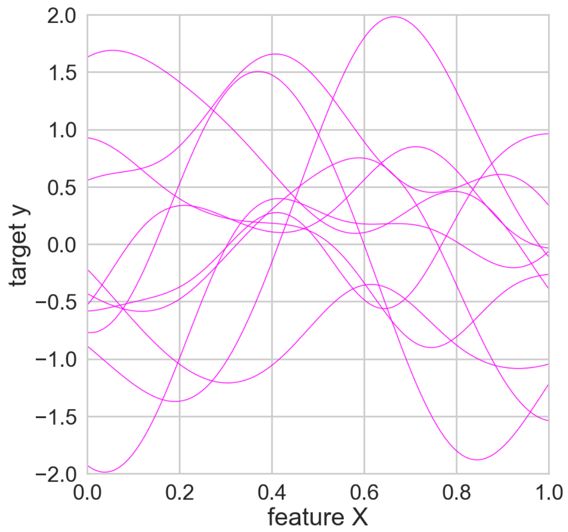
Gaussian Process in action

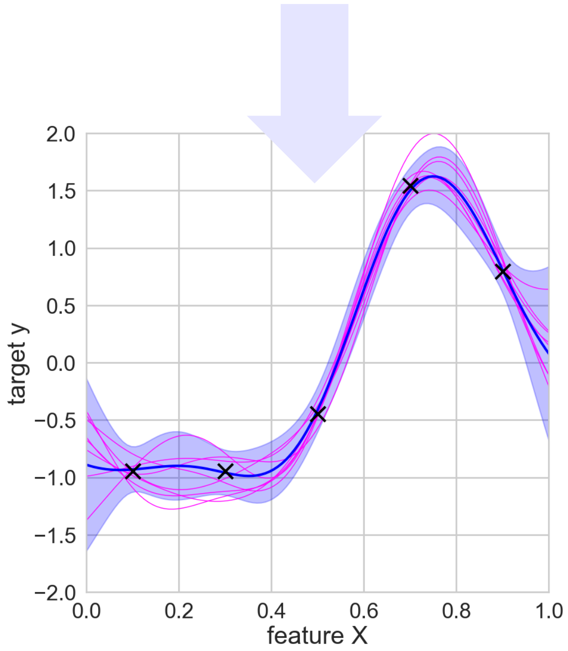




Advantages

- Non-parametric model
  - traditional parametric models: parameters define a specific functional shape
    - e.g. linear fit $y = a_0 + a_1$ x
  - (hyper-)parameters in GP define the *function space* *(= infinite set of possible functions)*
- Adaptive
  - add data points as experimental data comes in
  - GP: model complexity can grow with more data points
- Measure of uncertainty
  - Collect data where uncertainty is high (exploration)
  - Collect data where model prediction is best (exploitation)
  - (→ Bayesian optimization, later today)





Questions?

Questions?

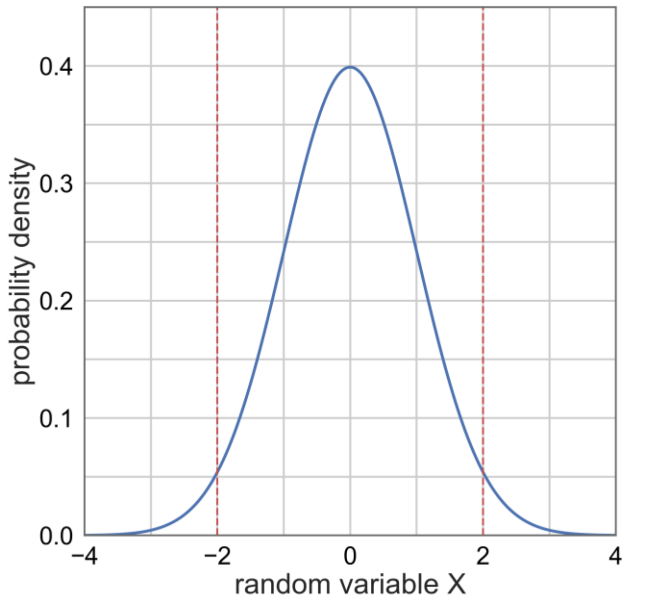
- How to generate the curves?
- Do these curves have a common property?

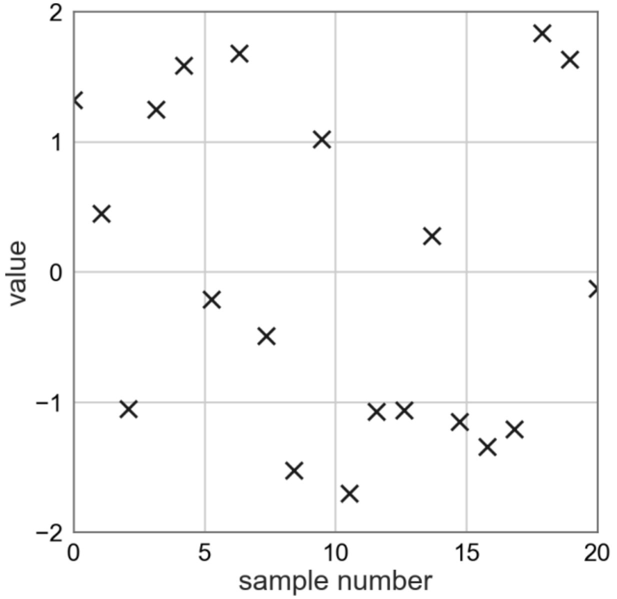
2 GP Regression

2.1 Stochastic Process
2.2 Bayesian Conditioning / Updating
2.3 Log marginal likelihood

Stochastic process

- sampling from a (normal) distribution

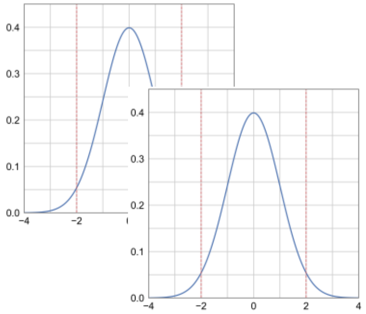




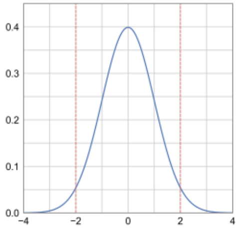
mean μ
variance σ^2

Stochastic process

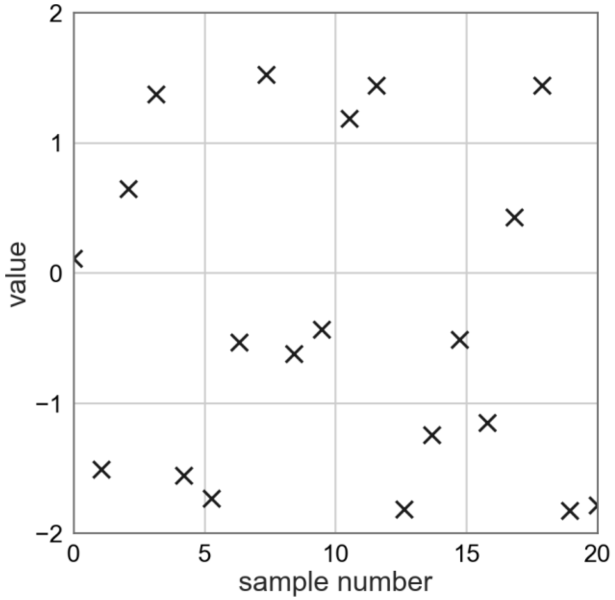
- sampling from an **independent** multivariate normal distribution



...

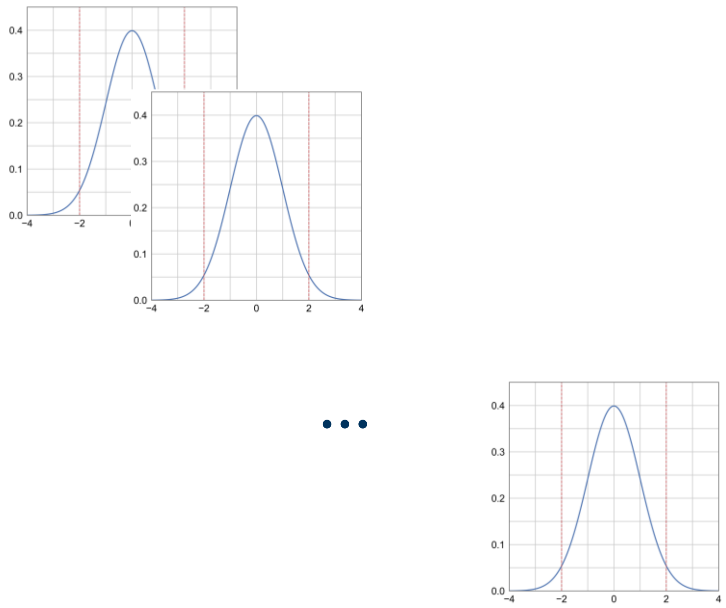


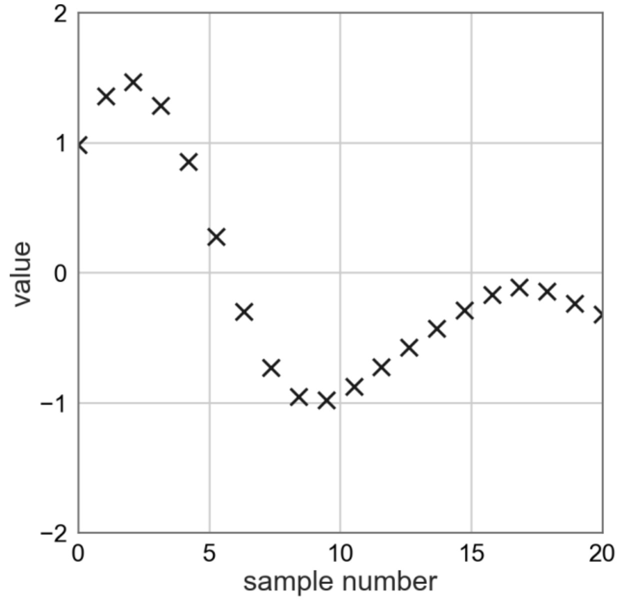
mean vector $\boldsymbol{\mu}$
variance vector $\boldsymbol{\sigma}^2$



Stochastic process

- sampling from a **general** multivariate normal distribution





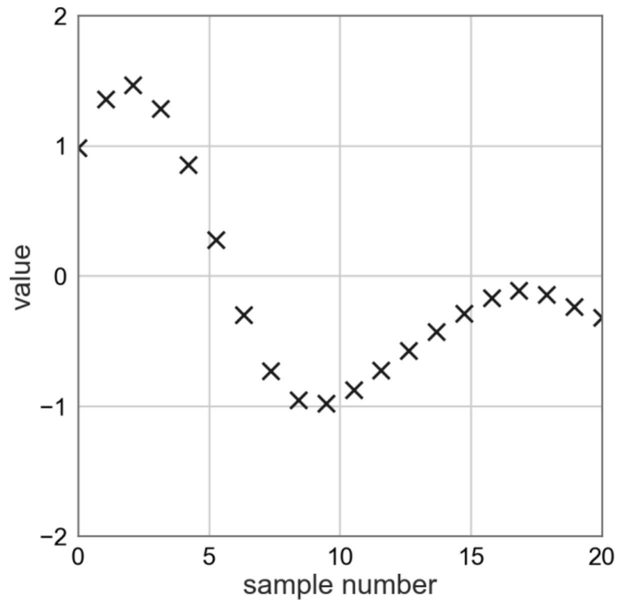
mean vector **μ**
covariance matrix **σ²**

stochastic process based on normal distributions: **Gaussian process**

Covariance matrix

- specifies covariance („correlation“) between each pair of random variables
  - Cov = 1 → strongest covariance
  - Cov = 0 → no covariance

- is a measure for distance between two points



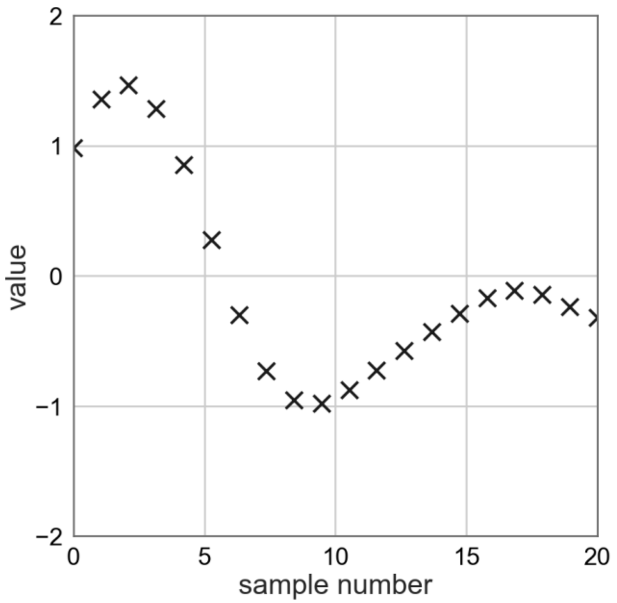
Covariance matrix

- specifies covariance („correlation“) between each pair of random variables

20

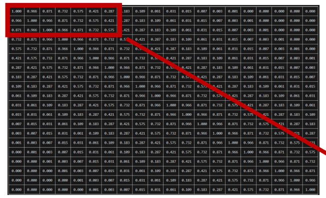
20

1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015	0.007	0.003	0.001	0.000	0.000	0.000	0.000	0.000
0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015	0.007	0.003	0.001	0.000	0.000	0.000	0.000
0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015	0.007	0.003	0.001	0.000	0.000	0.000
0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015	0.007	0.003	0.001	0.000	0.000
0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015	0.007	0.003	0.001	0.000
0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015	0.007	0.003	0.001
0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015	0.007	0.003
0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015	0.007
0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031	0.015
0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061	0.031
0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109	0.061
0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183	0.109
0.007	0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287	0.183
0.003	0.007	0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421	0.287
0.001	0.003	0.007	0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575	0.421
0.000	0.001	0.003	0.007	0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732	0.575
0.000	0.000	0.001	0.003	0.007	0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871	0.732
0.000	0.000	0.000	0.001	0.003	0.007	0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966	0.871
0.000	0.000	0.000	0.000	0.001	0.003	0.007	0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000	0.966
0.000	0.000	0.000	0.000	0.000	0.001	0.003	0.007	0.015	0.031	0.061	0.109	0.183	0.287	0.421	0.575	0.732	0.871	0.966	1.000



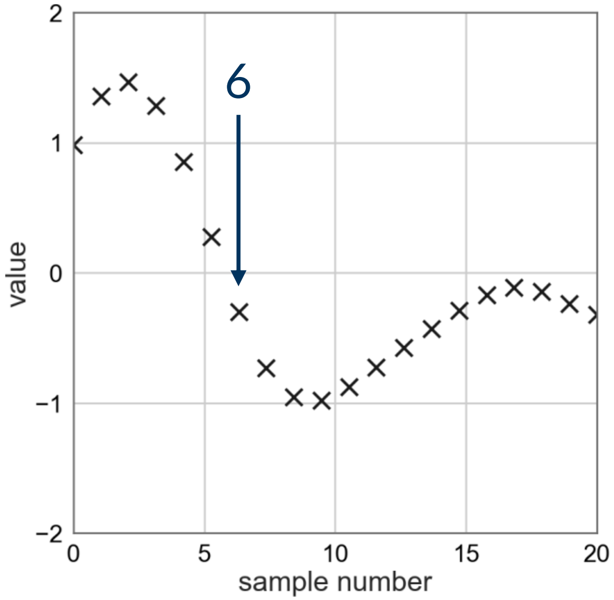
Covariance matrix

- specifies covariance („correlation“) between each pair of random variables



	0	1	2	3	4	5	6	
0	1.000	0.966	0.871	0.732	0.575	0.421	0.287	
1	0.966	1.000	0.966	0.871	0.732	0.575	0.421	…
2	0.871	0.966	1.000	0.966	0.871	0.732	0.575	

…

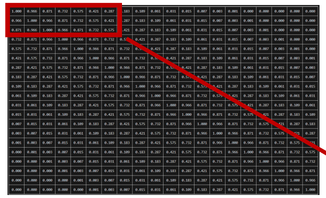


- Covariance function (or kernel function)
  - Radial Basis Kernel

$$cov(f(x), f(x')) = k(x, x') = \exp\left(-\frac{(x-x')^2}{2l^2}\right)$$

Covariance matrix

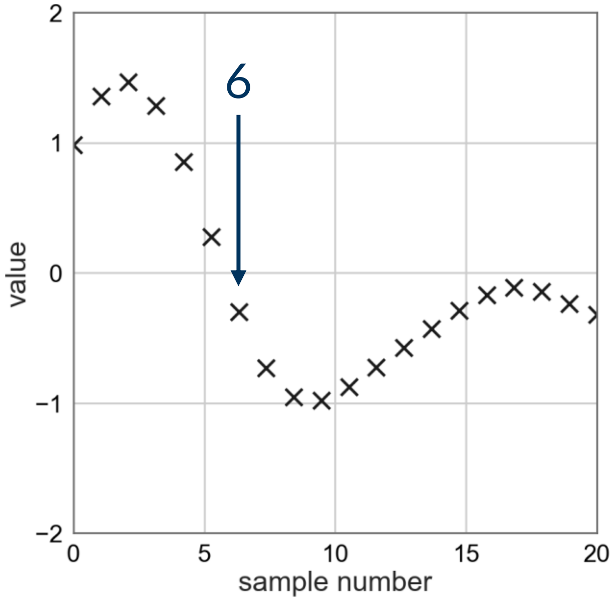
- specifies covariance („correlation“) between each pair of random variables



	0	1	2	3	4	5	6
0	1.000	0.966	0.871	0.732	0.575	0.421	0.287
1	0.966	1.000	0.966	0.871	0.732	0.575	0.421
2	0.871	0.966	1.000	0.966	0.871	0.732	0.575

…

…

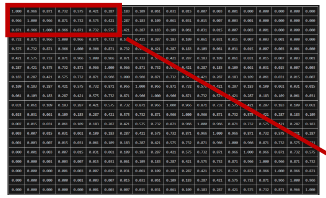


- Covariance function (or kernel function)
  - Radial Basis Kernel (most common kernel function, but there are many more!)

$$cov(f(x), f(x')) = k(x, x') = \exp\left(-\frac{(x-x')^2}{2l^2}\right)$$

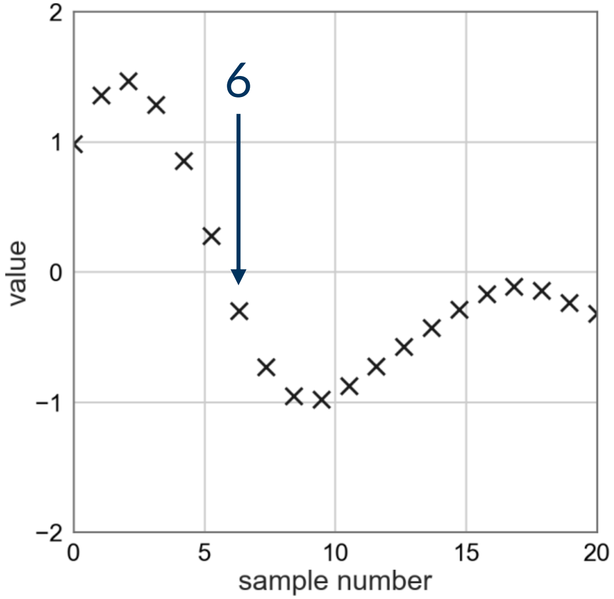
Covariance matrix

- specifies covariance („correlation“) between each pair of random variables



	0	1	2	3	4	5	6	…
0	1.000	0.966	0.871	0.732	0.575	0.421	0.287	
1	0.966	1.000	0.966	0.871	0.732	0.575	0.421	…
2	0.871	0.966	1.000	0.966	0.871	0.732	0.575	

…



- Covariance function (or kernel function)
  - Radial Basis Kernel (most common kernel function, but there are many more!)

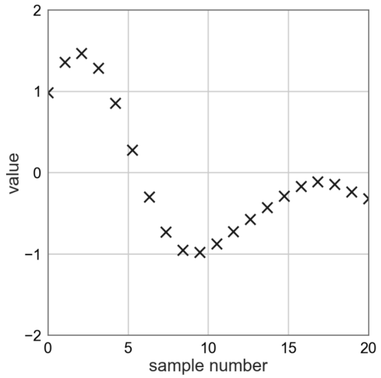
$$cov(f(x), f(x')) = k(x, x') = \exp\left(-\frac{(x-x')^2}{2\boldsymbol{l}^2}\right)$$

length scale parameter $\boldsymbol{l}$

Length scale parameter

$l = 4$

	0	1	2	3	4	5	6	…
0	1.000	0.966	0.871	0.732	0.575	0.421	0.287	
1	0.966	1.000	0.966	0.871	0.732	0.575	0.421	…
2	0.871	0.966	1.000	0.966	0.871	0.732	0.575	
…								



…

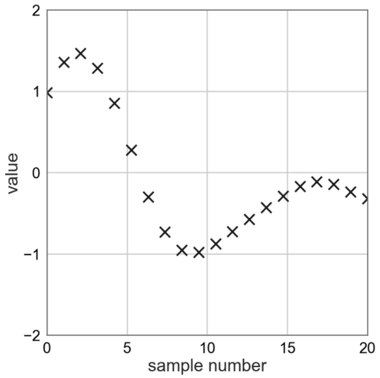
Length scale parameter

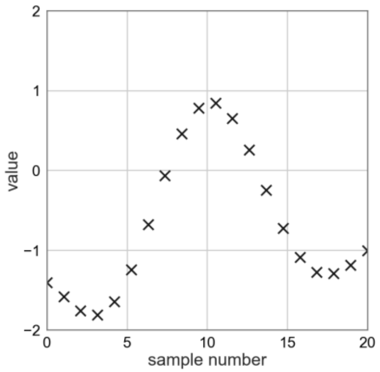
l = 4

	0	1	2	3	4	5	6
0	1.000	0.966	0.871	0.732	0.575	0.421	0.287
1	0.966	1.000	0.966	0.871	0.732	0.575	0.421
2	0.871	0.966	1.000	0.966	0.871	0.732	0.575

…

…





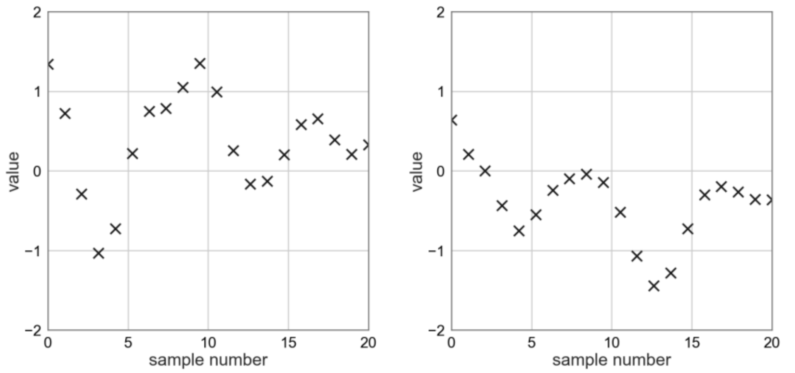
…

Length scale parameter

l = 2

	0	1	2	3	4	5	6	
0	1.000	0.871	0.575	0.287	0.109	0.031	0.007	
1	0.871	1.000	0.871	0.575	0.287	0.109	0.031	…
2	0.575	0.871	1.000	0.871	0.575	0.287	0.109	

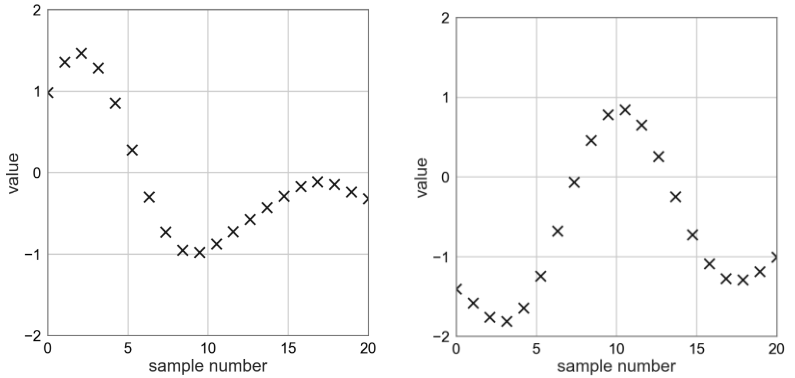
…



l = 4

	0	1	2	3	4	5	6	
0	1.000	0.966	0.871	0.732	0.575	0.421	0.287	
1	0.966	1.000	0.966	0.871	0.732	0.575	0.421	…
2	0.871	0.966	1.000	0.966	0.871	0.732	0.575	

…



Length scale parameter

$l = 2$

	0	1	2	3	4	5	6	
0	1.000	0.871	0.575	0.287	0.109	0.031	0.007	
1	0.871	1.000	0.871	0.575	0.287	0.109	0.031	…
2	0.575	0.871	1.000	0.871	0.575	0.287	0.109	

…

$l = 4$

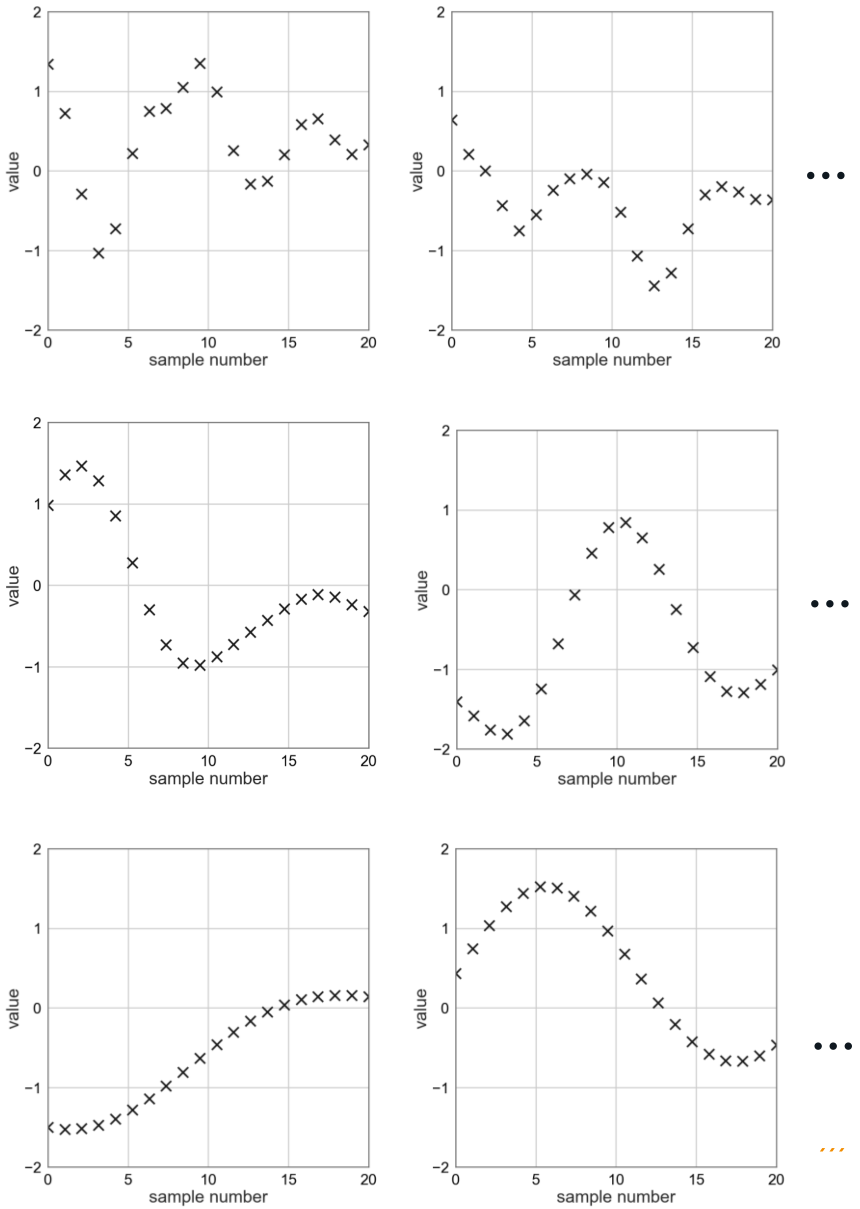
	0	1	2	3	4	5	6	
0	1.000	0.966	0.871	0.732	0.575	0.421	0.287	
1	0.966	1.000	0.966	0.871	0.732	0.575	0.421	…
2	0.871	0.966	1.000	0.966	0.871	0.732	0.575	

…

$l = 8$

	0	1	2	3	4	5	6	
0	1.000	0.991	0.966	0.925	0.871	0.805	0.732	
1	0.991	1.000	0.991	0.966	0.925	0.871	0.805	…
2	0.966	0.991	1.000	0.991	0.966	0.925	0.871	

…



Tasks

- Run 01_gaussian_process.py
- What does the plot show?
- Vary the length scale parameter. What do you observe?
- What do you observe when changing the RBF kernel to
  - Matern kernel
  - Periodic kernel (exp-sine-squared)
  - Linear kernel

Tasks

- Run 01_gaussian_process.py
- What does the plot show?
- Vary the length scale parameter. What do you observe?
- What do you observe when changing the RBF kernel to
  - Matern kernel
    (becomes RBF kernel for ν = ∞)

    $$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right)$$

  - Periodic kernel (ExpSineSquared)

    $$k(x_i, x_j) = \exp\left(-\frac{2\sin^2(\pi d(x_i, x_j)/p)}{l^2}\right)$$

  - Linear kernel (DotProduct)

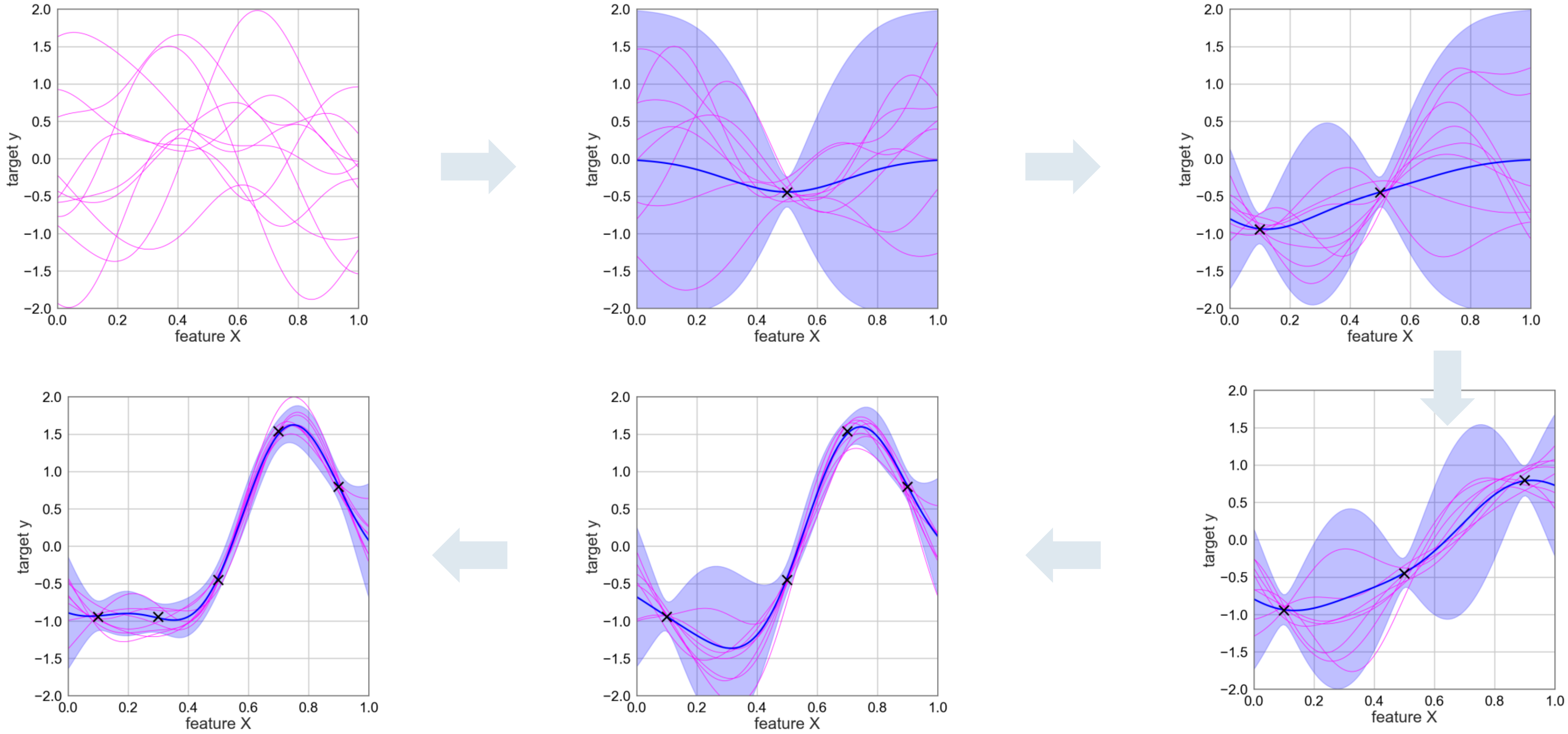
    $$k(x_i, x_j) = \sigma_0^2 + x_i \cdot x_j$$

scikit learn

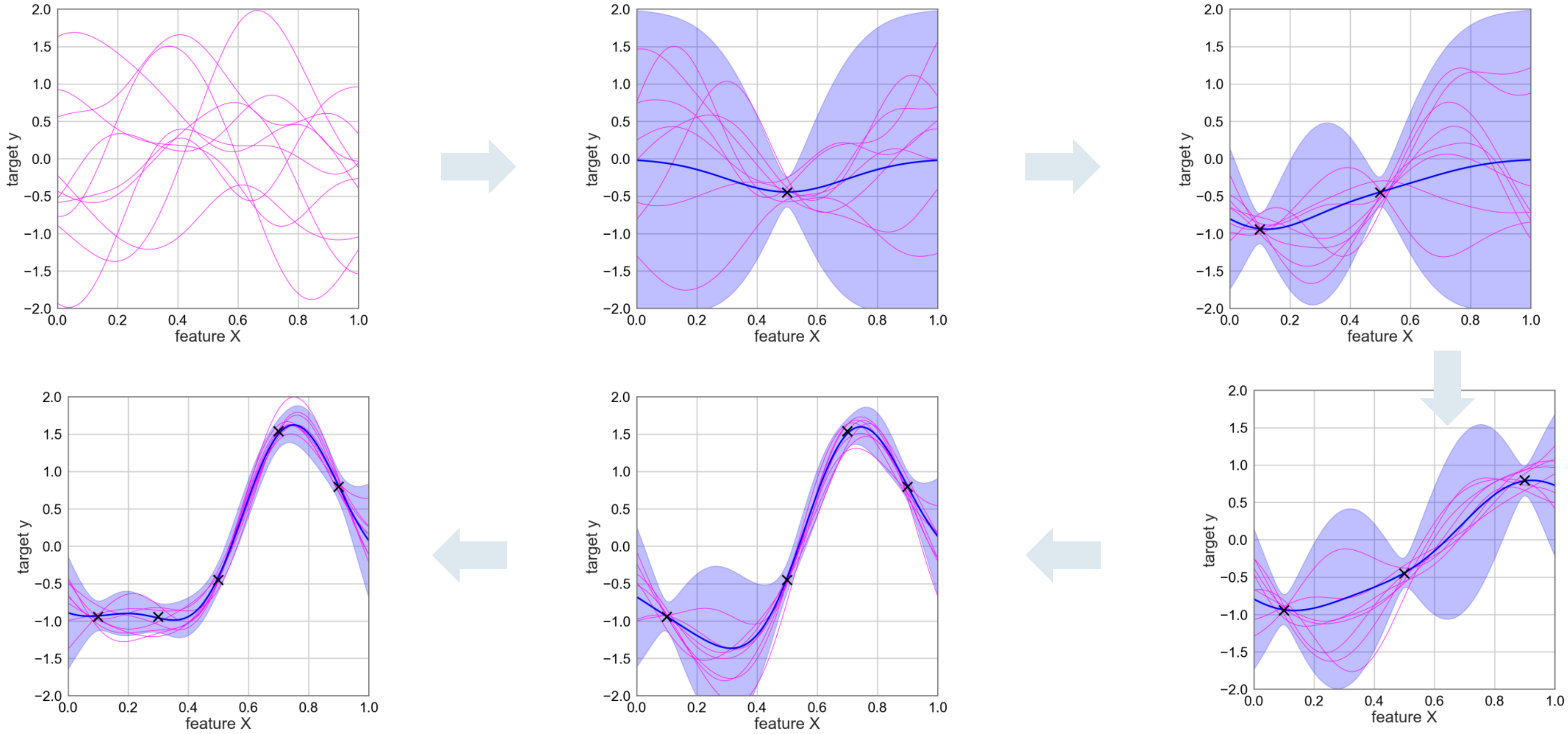
2 GP Regression

2.1 Stochastic Process
2.2 Conditioning / Updating the GP
2.3 Model Assessment

Conditioning / Updating the GP



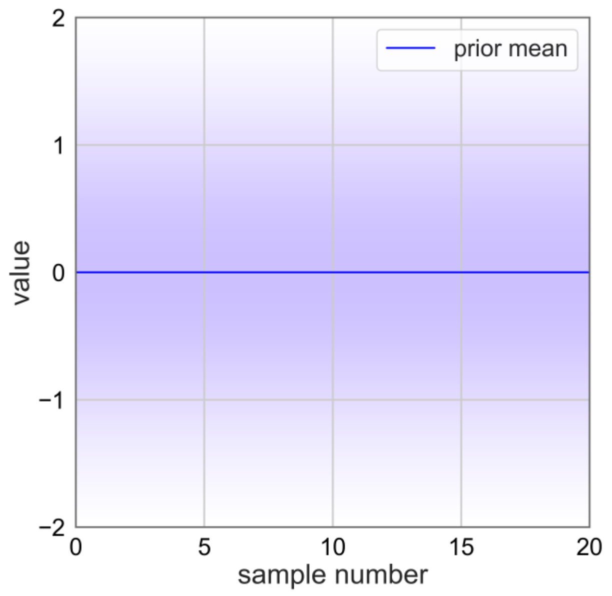
Conditioning / Updating the GP



Very intuitive picture, but in reality one works with the underlying probability distributions.

GP Prior (before any data)

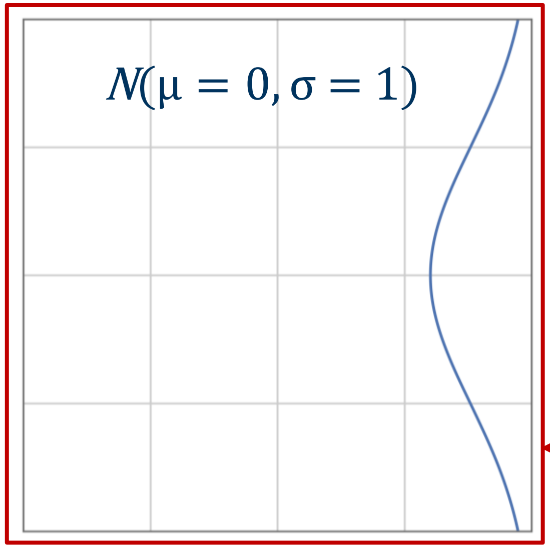
- Prior: probability distribution over functions before adding observations

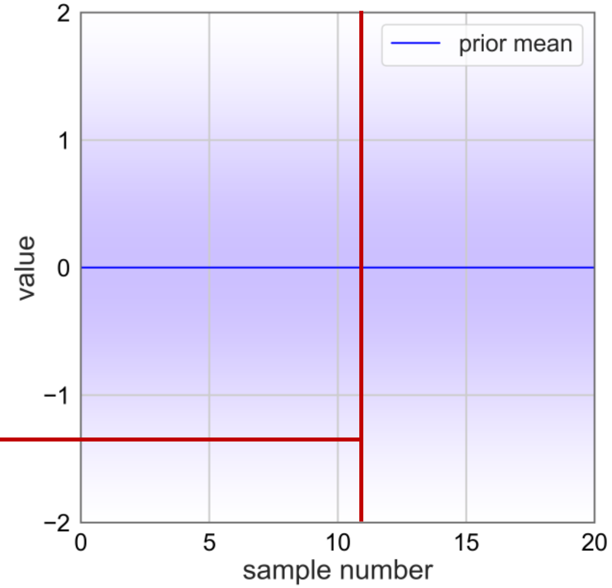


infinite number of normal distributions with
a **mean vector** and a **covariance matrix**

GP Prior (before any data)

- Prior: probability distribution over functions before adding observations

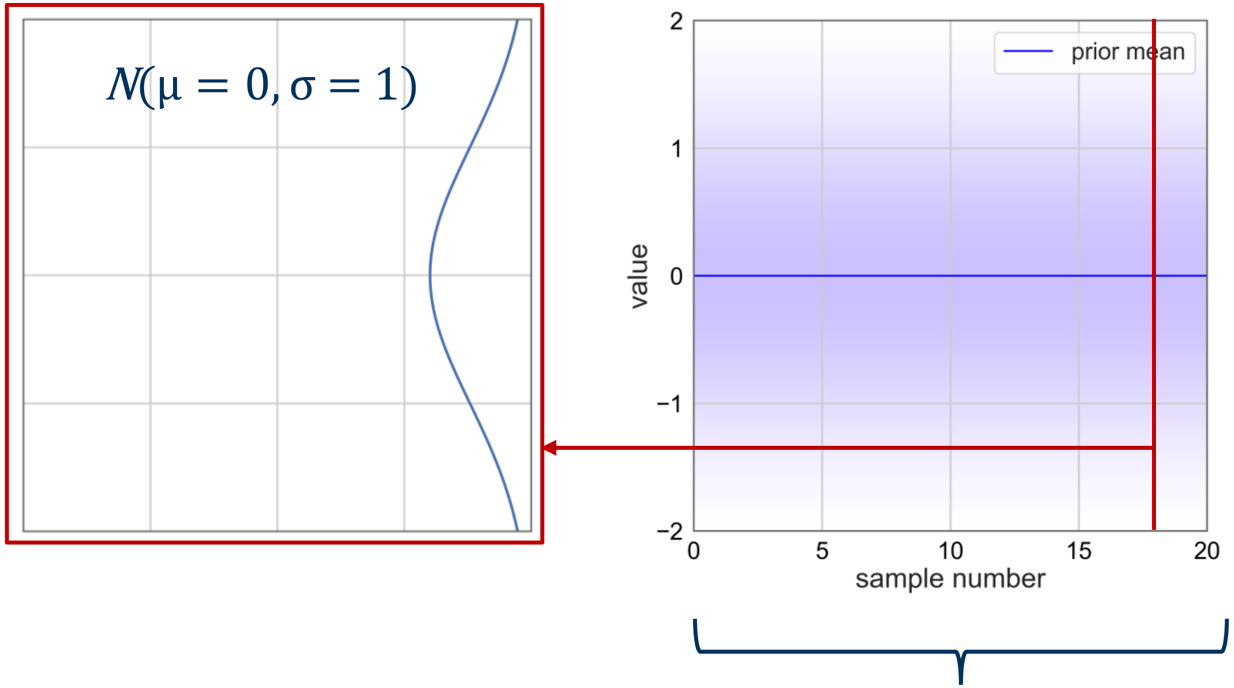




infinite number of normal distributions with a **mean vector** and a **covariance matrix**

GP Prior (before any data)

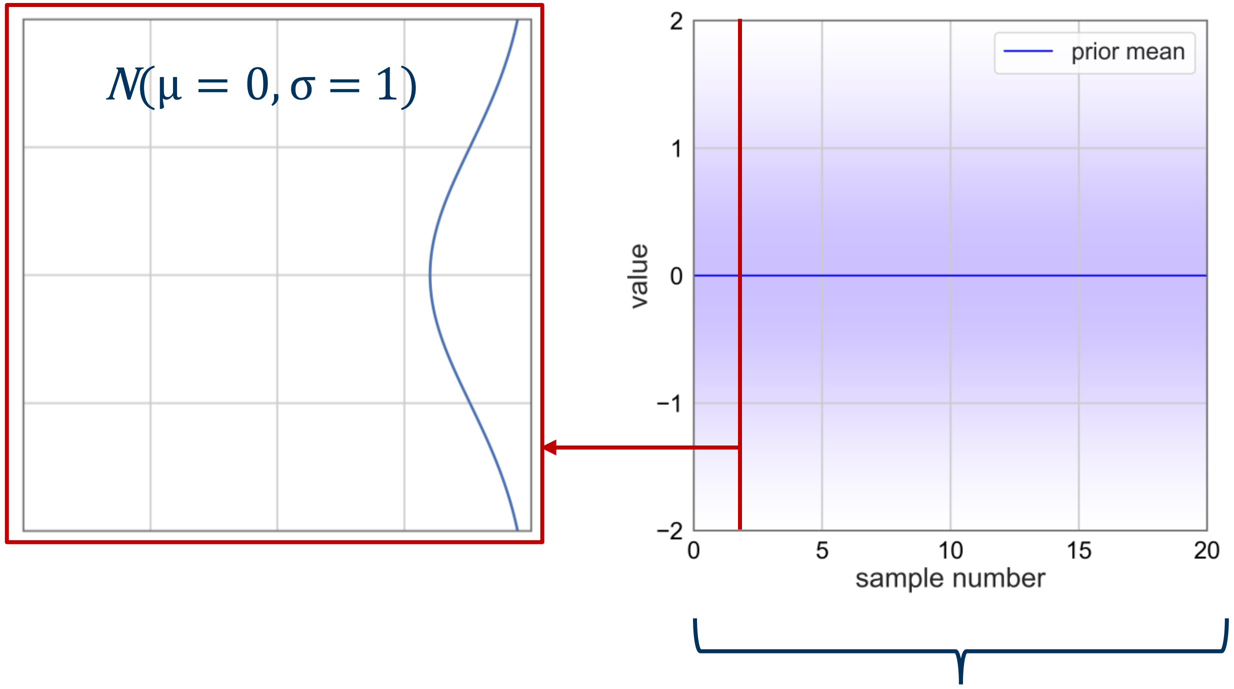
- Prior: probability distribution over functions before adding observations



infinite number of normal distributions with
a **mean vector** and a **covariance matrix**

GP Prior (before any data)

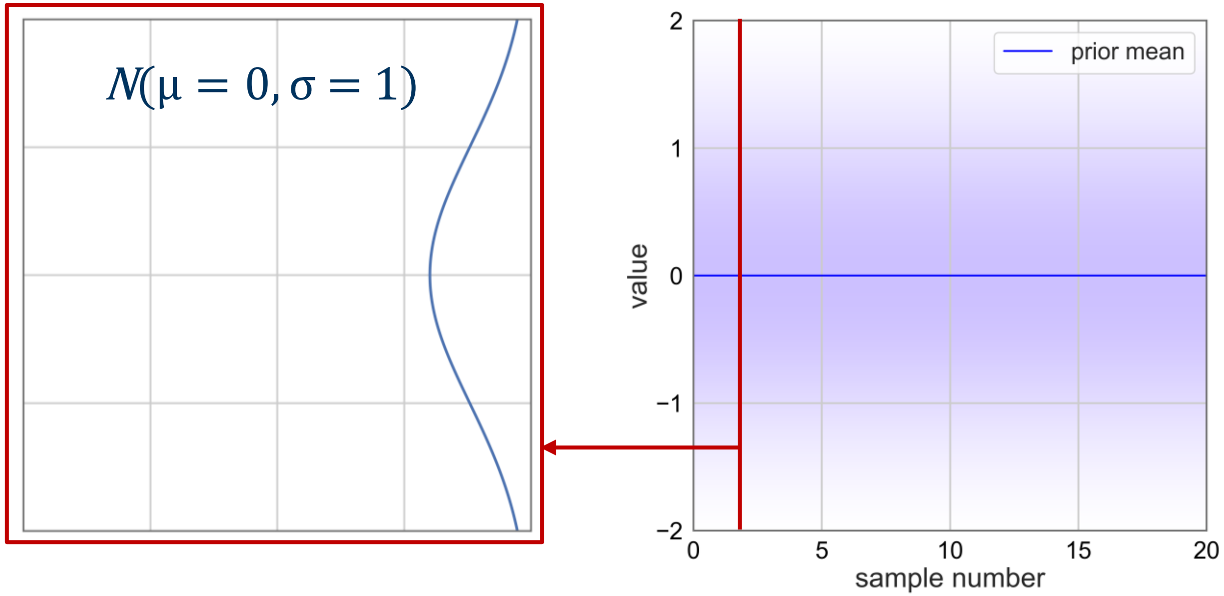
- Prior: probability distribution over functions before adding observations



infinite number of normal distributions with
a **mean vector** and a **covariance matrix**

GP Prior (before any data)

- Prior: probability distribution over functions before adding observations

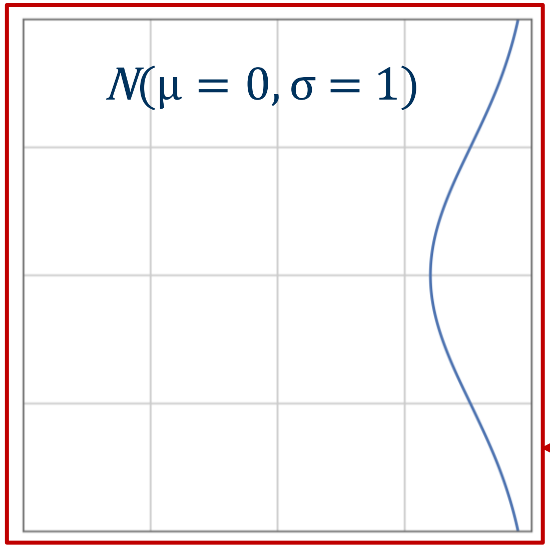


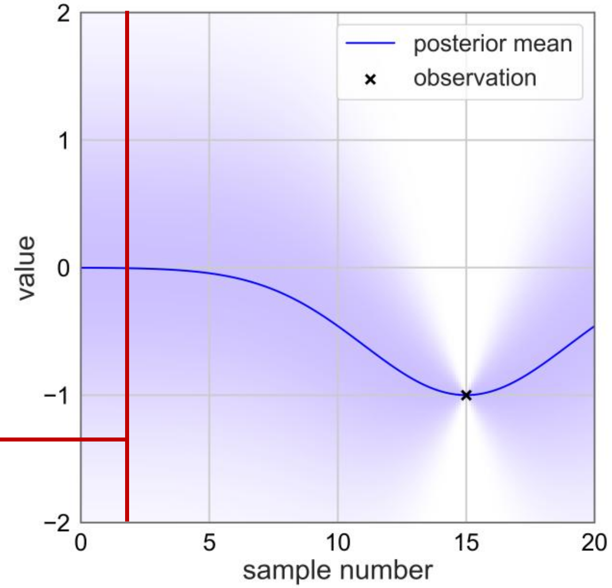
Conditioning: Posterior $\propto$ Prior $\times$ Likelihood

Martin Rudolph

GP Posterior (after adding data)

- Prior: probability distribution over functions before adding observations

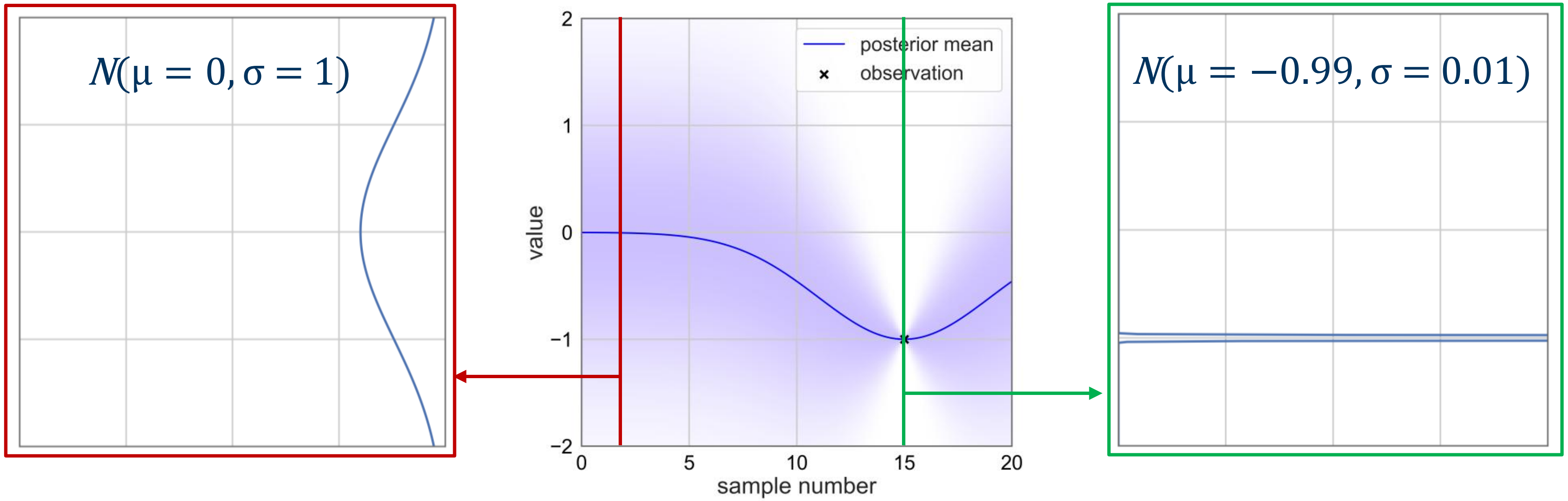




Conditioning: Posterior $\propto$ Prior $\times$ Likelihood

GP Posterior (after adding data)

- Prior: probability distribution over functions before adding observations



Conditioning: Posterior $\propto$ Prior $\times$ Likelihood

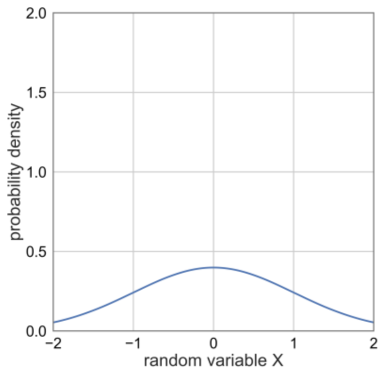
Likelihood

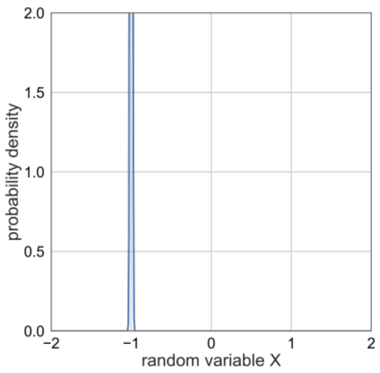
Conditioning: Posterior $\propto$ Prior $\times$ Likelihood

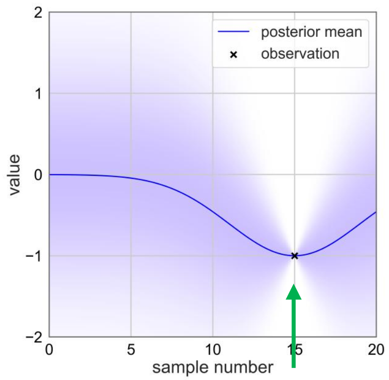
- Likelihood: normal distribution with μ and sigma

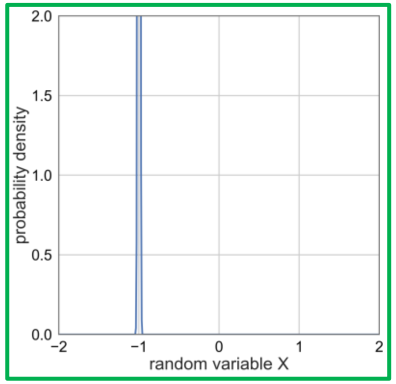
Prior $\times$ **Likelihood** $\propto$ Posterior „slice through posterior“

$\mathcal{N}(\mu = 0, \sigma = 1)$ $\mathcal{N}(\mu = -1, \sigma = 0.01)$ $\mathcal{N}(\mu = -0.99, \sigma = 0.01)$









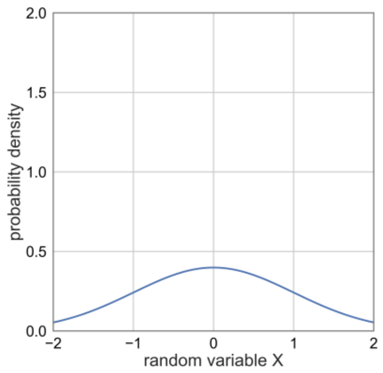
Likelihood

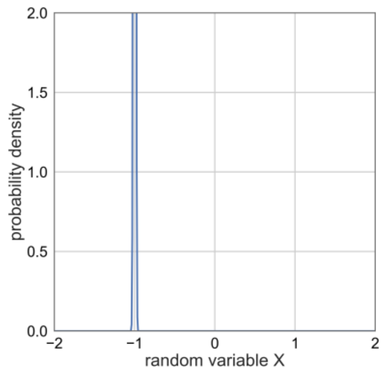
Conditioning: Posterior $\propto$ Prior $\times$ Likelihood

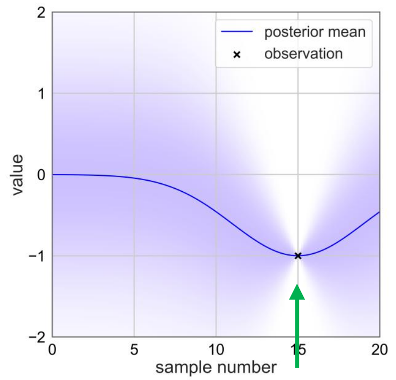
- Likelihood: normal distribution with μ and sigma

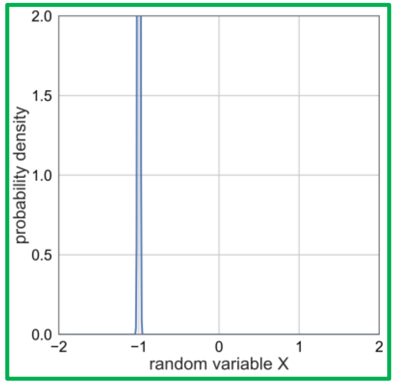
Prior $\times$ **Likelihood** $\propto$ Posterior „slice through posterior“

$N(\mu = 0, \sigma = 1)$ $N(\mu = -1, \sigma = 0.01)$ $N(\mu = -0.99, \sigma = 0.01)$

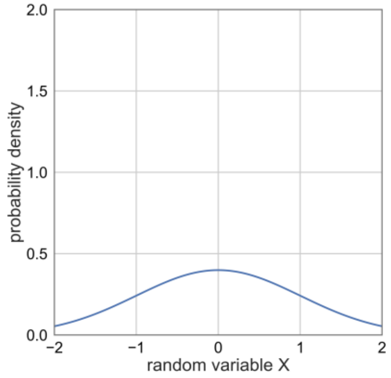


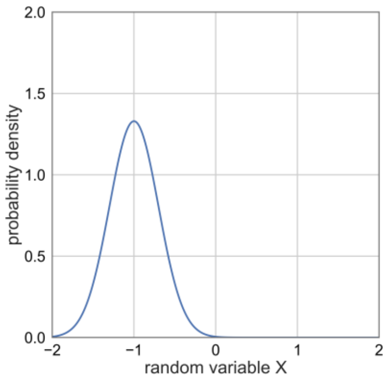


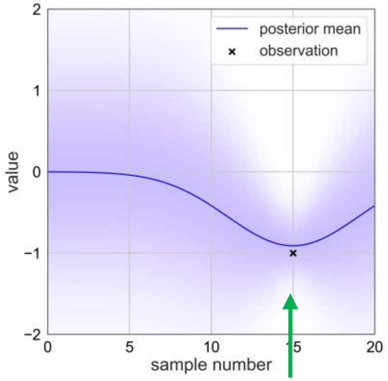


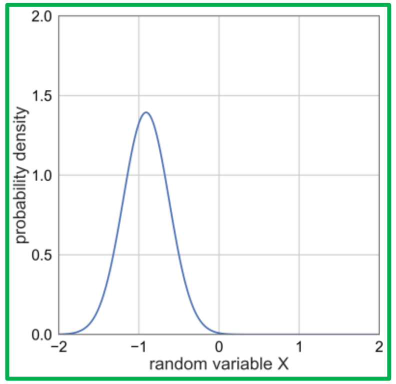


$N(\mu = -1, \sigma = 0.3)$ $N(\mu = -0.91, \sigma = 0.29)$









Tasks

- Open 02_bayesian_condition.py
- What does the plot show?
- Reduce the length scale. What do you observe?
- Increase the length scale. What do you observe?
- Add additional data points. What do you observe?

Tasks

- Open 02_bayesian_condition.py
- What does the plot show?
- Reduce the length scale. What do you observe?
- Increase the length scale. What do you observe?
  - numerical instabilities due to ill-conditioned matrix
- Add additional data points. What do you observe?

2 GP Regression

2.1 Stochastic Process
2.2 Conditioning / Updating the GP
2.3 Model Assessment

Martin Rudolph

Model assessment

objective: choose hyperparameters for
- goodness of fit
- generalization

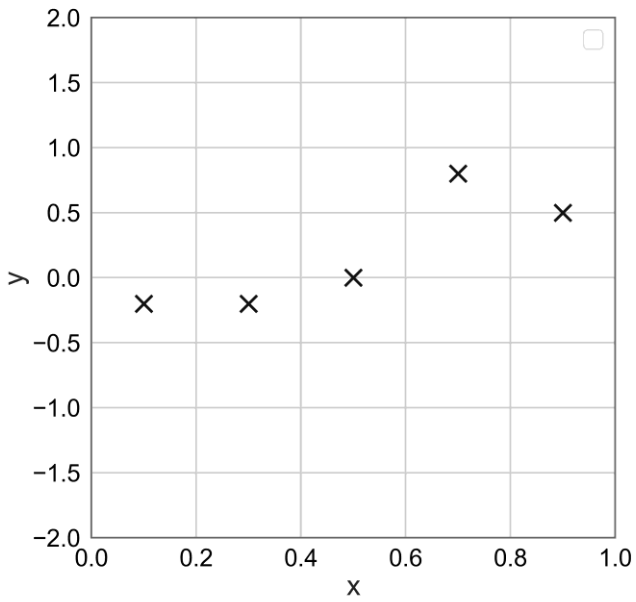
- in classical (deterministic) ML methods (support vector machines, random forrests,...):
  - hyperparameters are properties of the method
  - need to be adjusted to
    - predict test data **(generalization)**
    - with low error **(goodness of fit)**

Cross-validation to balance generalization and goodness of fit

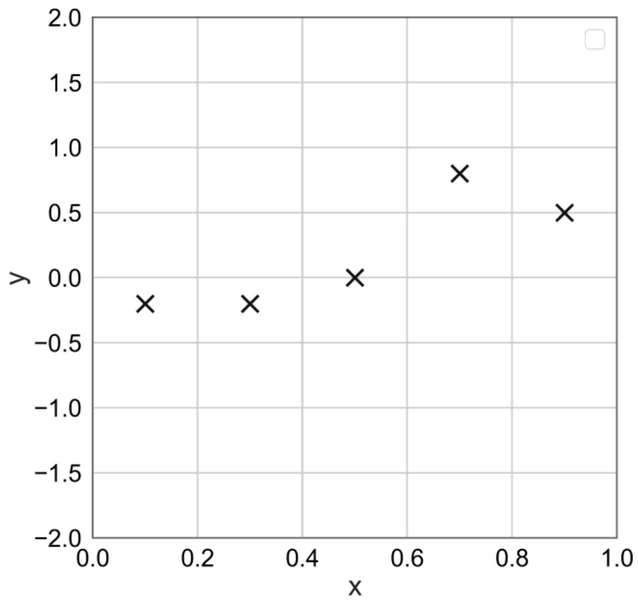
- in **probabilistic ML methods** (GP)
  - **hyperparameter (length scale in a GP) defines the probability distribution, that can be regarded as generating the data.**
  - (in practice: One can still do cross-validation to check the generalization)

Relation between lengthscale of prior and data

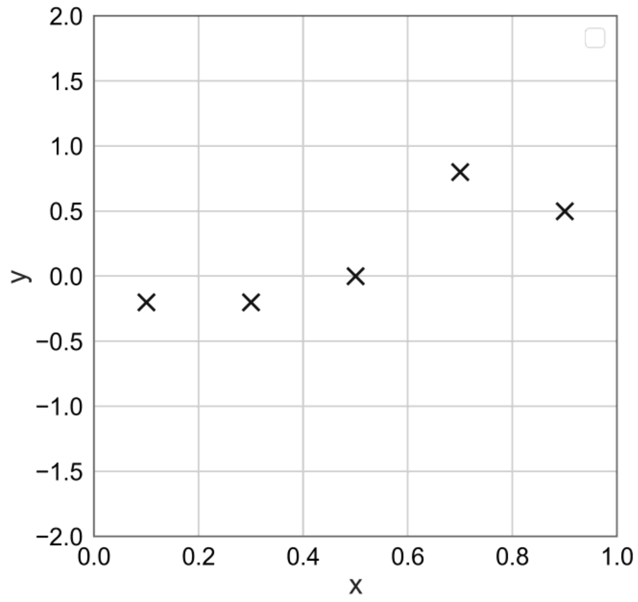
$l = 0.03$



$l = 0.17$

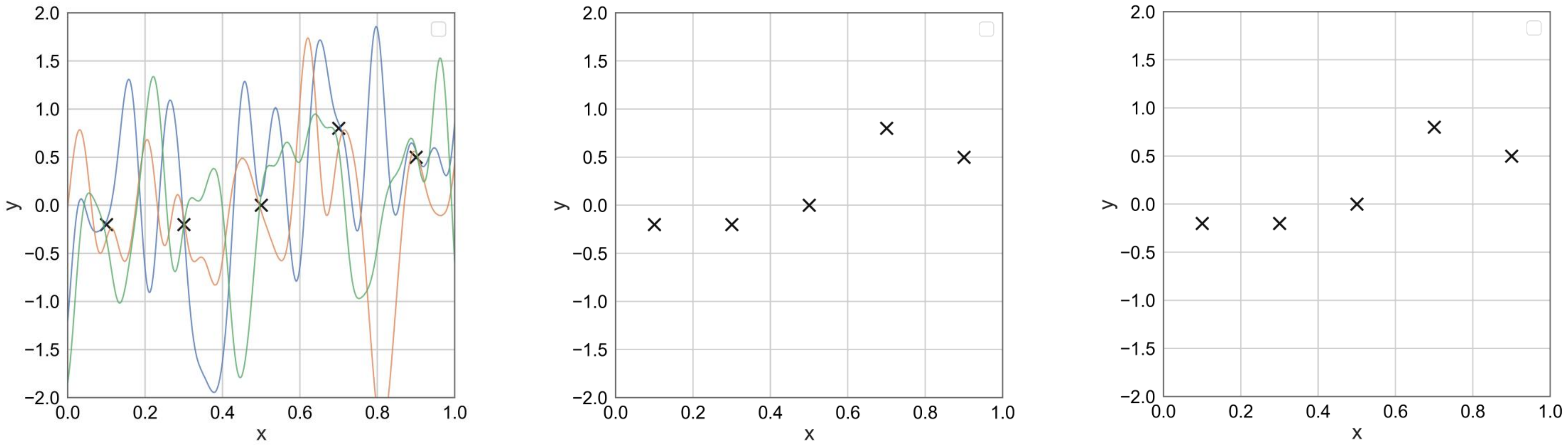


$l = 0.84$



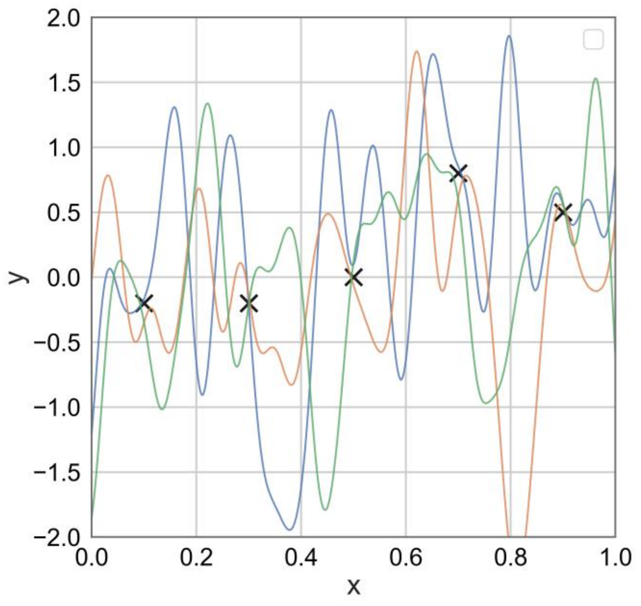
Relation between lengthscale of prior and data

$l = 0.03$ | $l = 0.17$ | $l = 0.84$

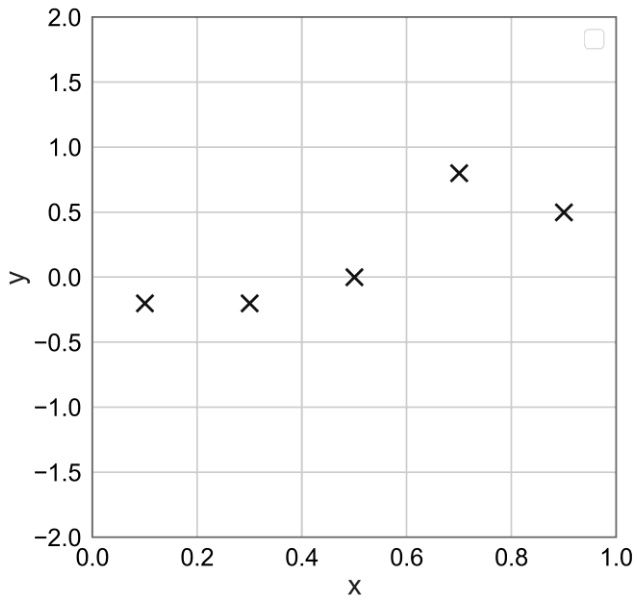


Relation between lengthscale of prior and data

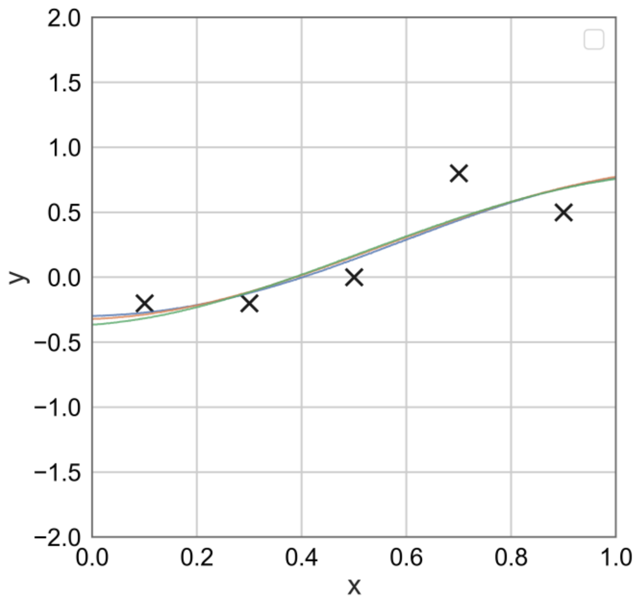
l = 0.03



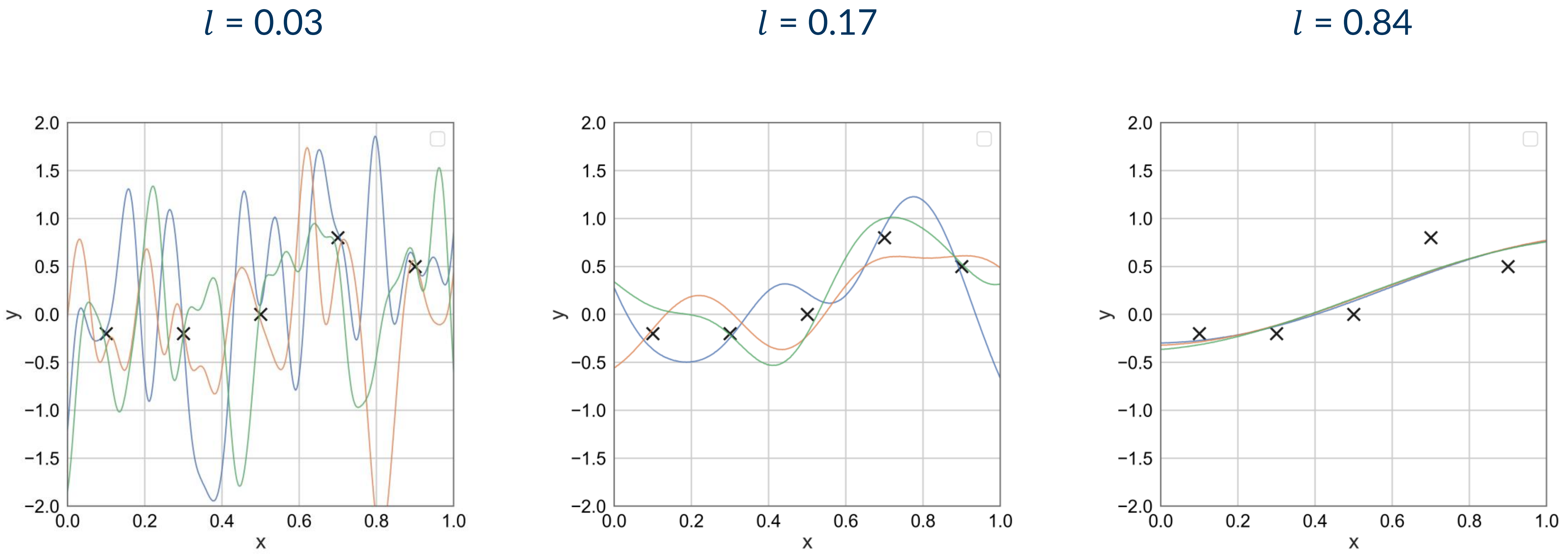
l = 0.17



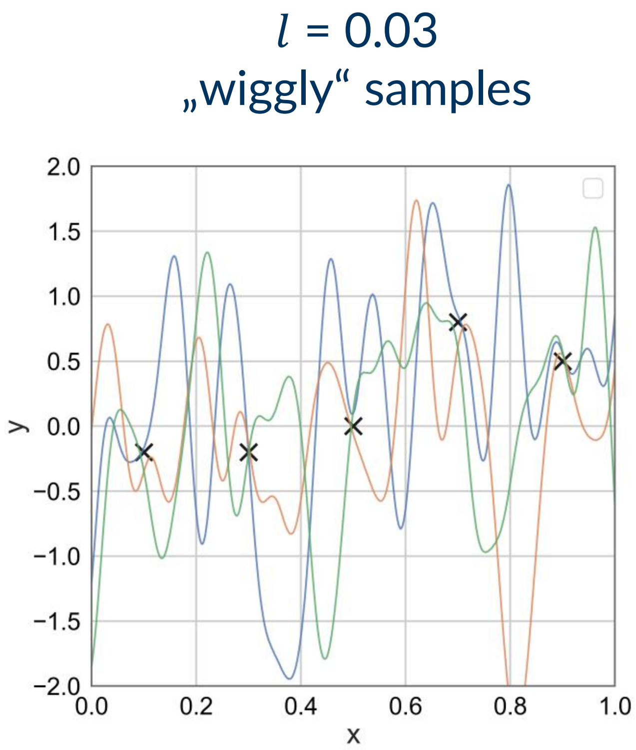
l = 0.84

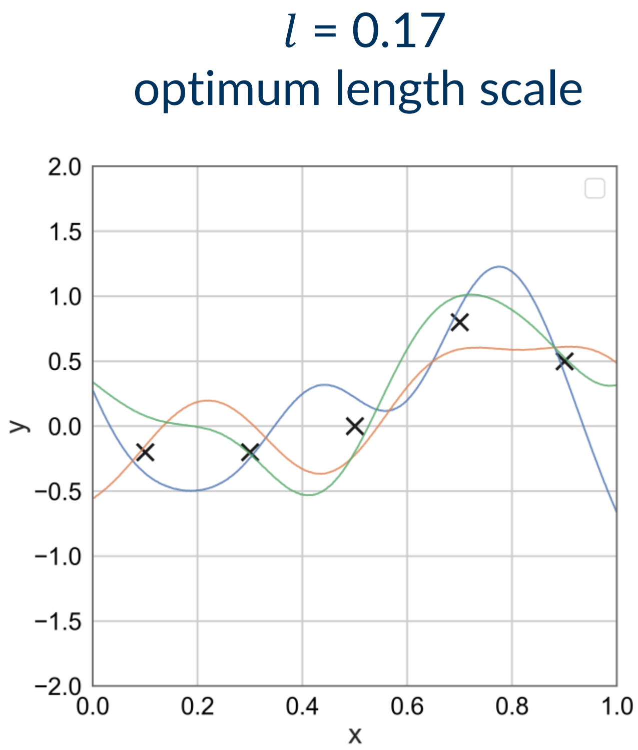


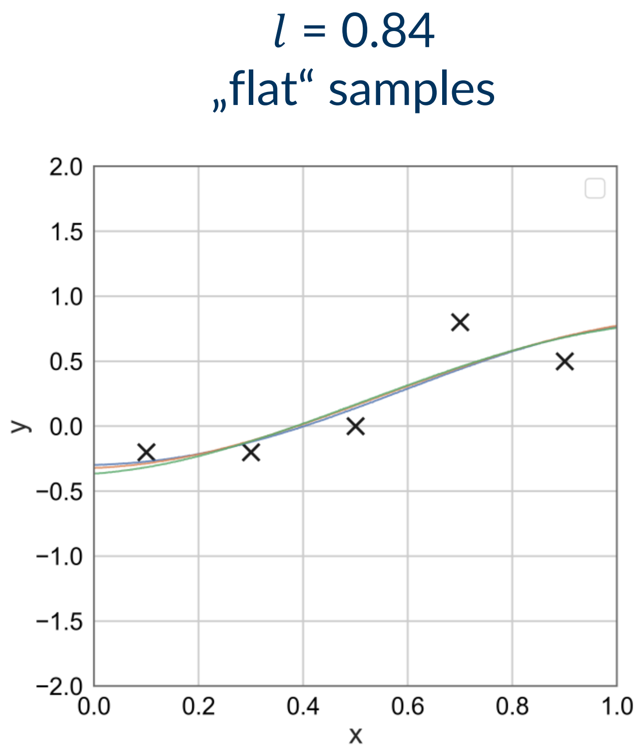
Relation between lengthscale of prior and data



Relation between lengthscale of prior and data

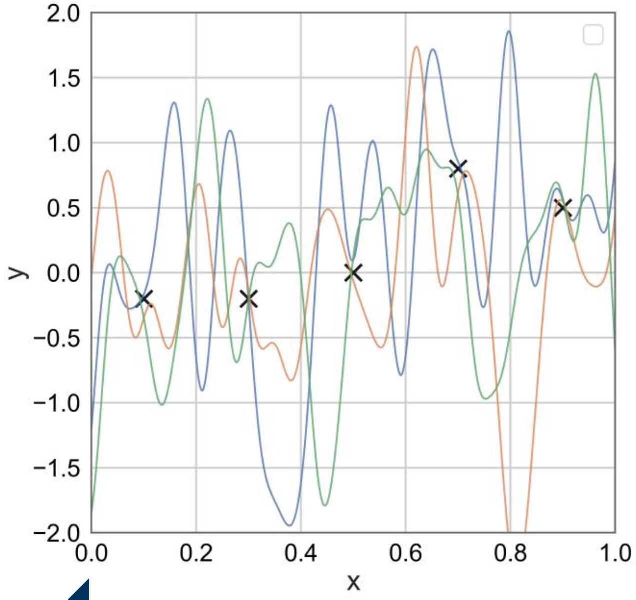




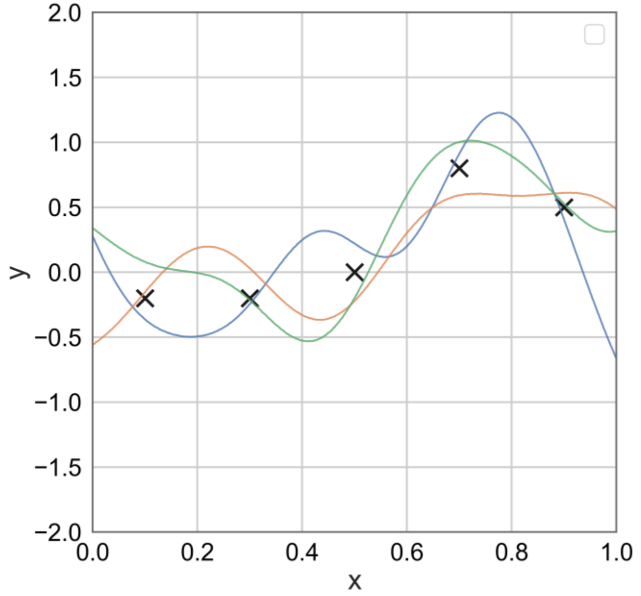


Relation between lengthscale of prior and data

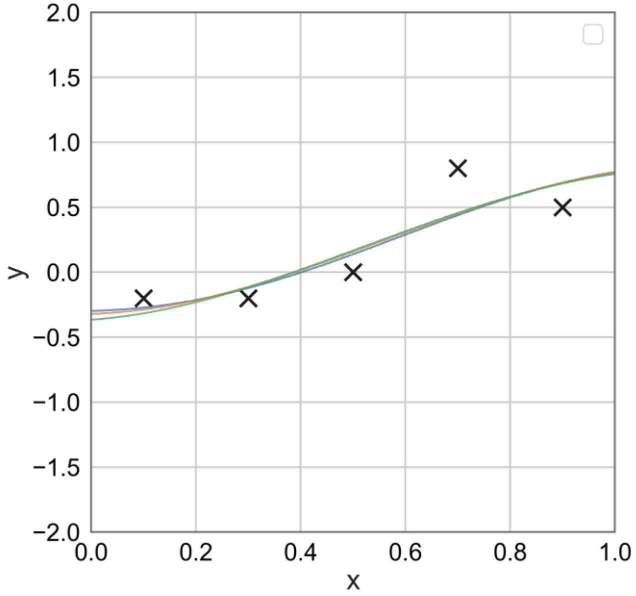
l = 0.03
„wiggly“ samples



l = 0.17
optimum length scale



l = 0.84
„flat“ samples



Increased goodness of fit

Reduced model complexity (Volume of function space → determinant of covariance matrix |K|)

→ Possibility to select an optimum lengthscale from the knowledge of the data!

Log likelihood

Likelihood function measures how probable it is to observe data **y** at input **X** given the hyperparameters θ.

to be maximized

$$\log p(\boldsymbol{y}|\boldsymbol{X},\theta) = \underbrace{-\frac{1}{2}\boldsymbol{y}^T K_y^{-1}\boldsymbol{y}}_{\textbf{data fit term}} \underbrace{-\frac{1}{2}\log\left|\mathrm{K_y}\right|}_{\textbf{complexity penality}} \underbrace{-\frac{\mathrm{n}}{2}\log 2\pi}_{\text{normalization (= constant)}}$$

Log likelihood

Likelihood function measures how probable it is to observe data **y** at input **X** given the hyperparameters θ.

to be maximized

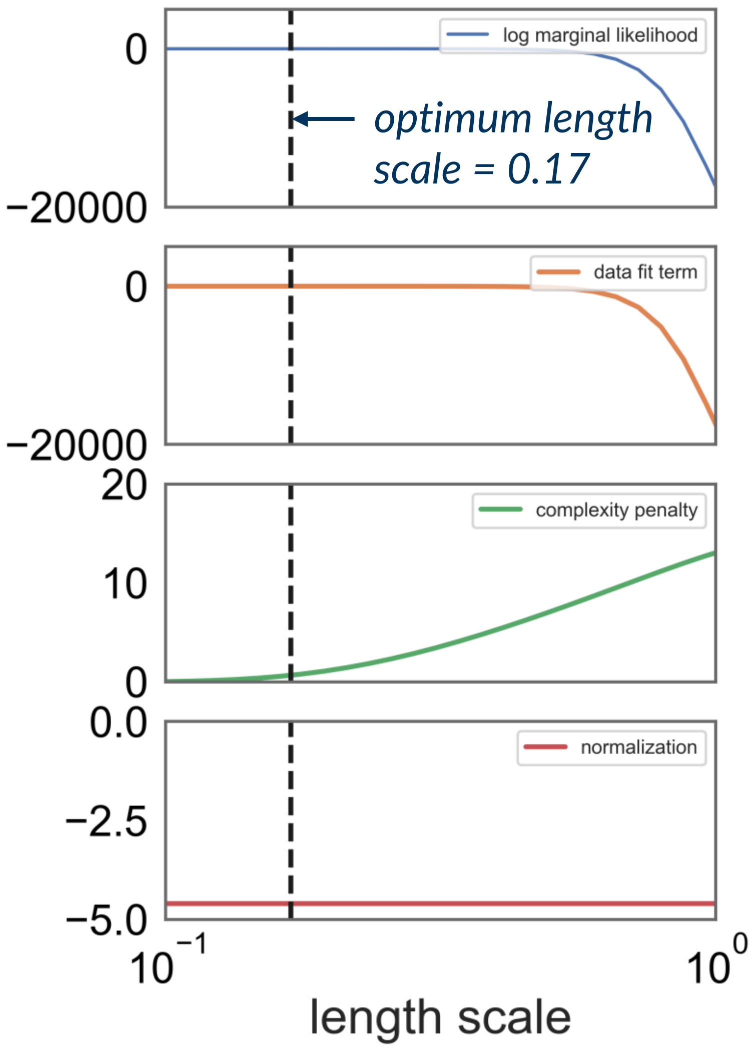
$$\log p(\boldsymbol{y}|\boldsymbol{X},\theta) = \underbrace{-\frac{1}{2}\boldsymbol{y}^T K_y^{-1}\boldsymbol{y}}_{\text{data fit term}} \quad \underbrace{-\frac{1}{2}\log|\mathrm{K_y}|}_{\text{complexity penality}} \quad \underbrace{-\frac{\mathrm{n}}{2}\log 2\pi}_{\text{normalization (= constant)}}$$

Mahalanobis distance^2 = generalized squared error

(more negative for worse fits)

determinant $|K_y|$ becomes larger for small length scales (vectors become linearly independent)

Log likelihood



Likelihood function measures how probable it is to observe data **y** at input **X** given the hyperparameters θ.

$$\log p(\boldsymbol{y}|\boldsymbol{X},\theta) = \underbrace{-\frac{1}{2}\boldsymbol{y}^T K_y^{-1}\boldsymbol{y}}_{\textbf{data fit term}} \underbrace{-\frac{1}{2}\log|\mathrm{K_y}|}_{\textbf{complexity penality}} \underbrace{-\frac{\mathrm{n}}{2}\log 2\pi}_{\text{normalization (= constant)}}$$

Tasks

- Open 03_lengthscale_optimization.py

- Run the script and optimize the lengthscale manually.

- Have the lengthscale optimized automatically
  - Set attribute *optimizer = „fmin_l_bfgs_b"*

- Print the optimum length scale to the screen and compare to your lengthscale.
  - Kernel can be accessed through *gp.kernel_*
  - Lengthscale can be accessed through *gp.kernel_.length_scale*

- Optimize the noise as well
  - Use a combination of a *WhiteKernel* and an *RBF* kernel (composite kernel)
  - Noise can be accessed through gp.kernel_.k1.length_scale
    gp.kernel_.k2.noise_level

Tasks

- Open 03_lengthscale_optimization.py

- Run the script and optimize the lengthscale manually.

- Have the lengthscale optimized automatically
  - Set attribute *optimizer* = *„fmin_l_bfgs_b"*
  - Possibly set the bounds *kernel = RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1))*
  - Possibly set restarts *gp = GaussianProcessRegressor(… , n_restarts_optimizer=10)*

- Print the optimum length scale to the screen and compare to your lengthscale.
  - Kernel can be accessed through *gp.kernel_*
  - Lengthscale can be accessed through *gp.kernel_.length_scale*

- Optimize the noise as well
  - Use a combination of a *WhiteKernel* and an *RBF* kernel (composite kernel)
  - Noise can be accessed through gp.kernel_.k1.length_scale
    gp.kernel_.k2.noise_level
  - Possibly add data points ;)

3 Bayesian Optimization

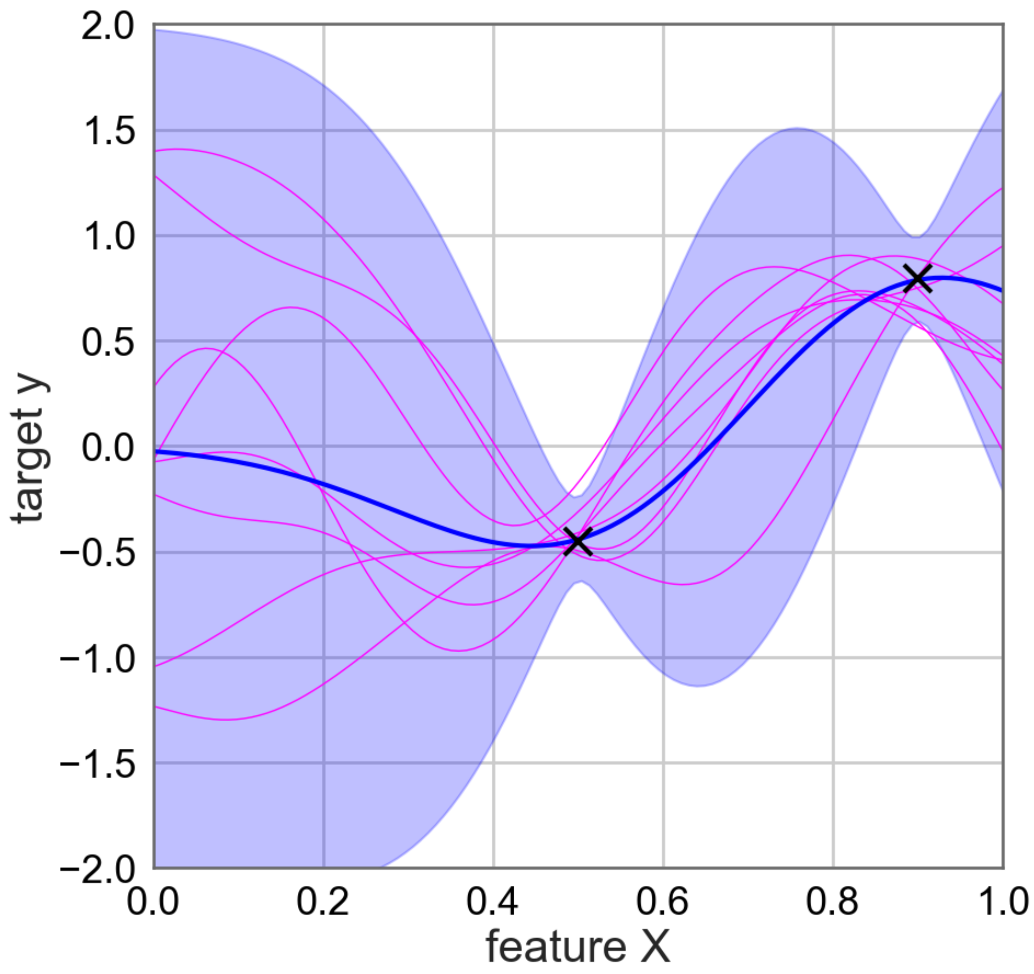
Where to sample next?

Probabilistic models offer

- Predictive mean
- Uncertainty estimate

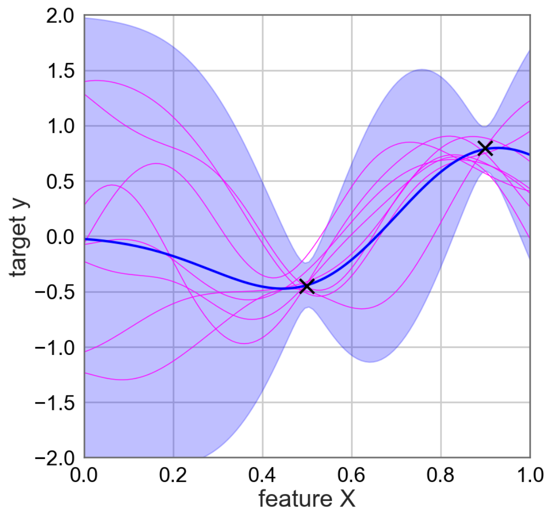
Two strategies

- **Exploitation**
  - sample at the models' **best prediction** (max/min. of the predictive mean)
- **Exploration**
  - Sample at model's highest uncertainty for **maximum model improvement**



Acquistion function

- acquisition function:
  - calculates a score for sampling at a location given the current state of the model



- Simple example: upper confidence bound
  → „pick the point with the largest optimistic estimate“

$$\alpha_{UCB}(x) = \mu(x) + \beta\sigma(x)$$ (for maximization problems)

$$\alpha_{UCB}(x) = \mu(x) - \beta\sigma(x)$$ (for minization problems)

exploration parameter β:

$\beta > 1$: promoting exploration
$\beta < 1$: promoting exploitation

possibility to adjust with each data point

Acquistion function

- acquisition function:
  - calculates a score for sampling at a location given the current state of the model

- Simple example: upper confidence bound
  → „pick the point with the largest optimistic estimate“

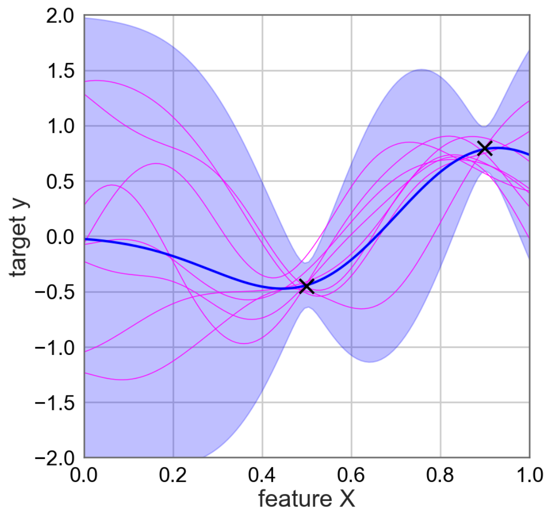
$$\alpha_{UCB}(x) = \mu(x) + \beta\sigma(x)$$ (for maximization problems)

$$\alpha_{UCB}(x) = \mu(x) - \beta\sigma(x)$$ (for minization problems)

exploration parameter β:

β > 1: promoting exploration
β < 1: promoting exploitation

possibility to adjust with each data point

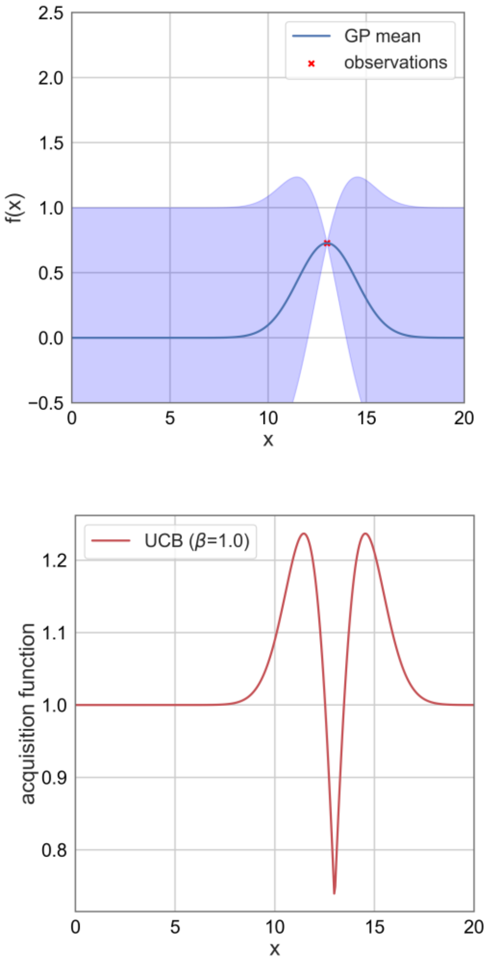


How does the UBC acquisition function look like for this plot?

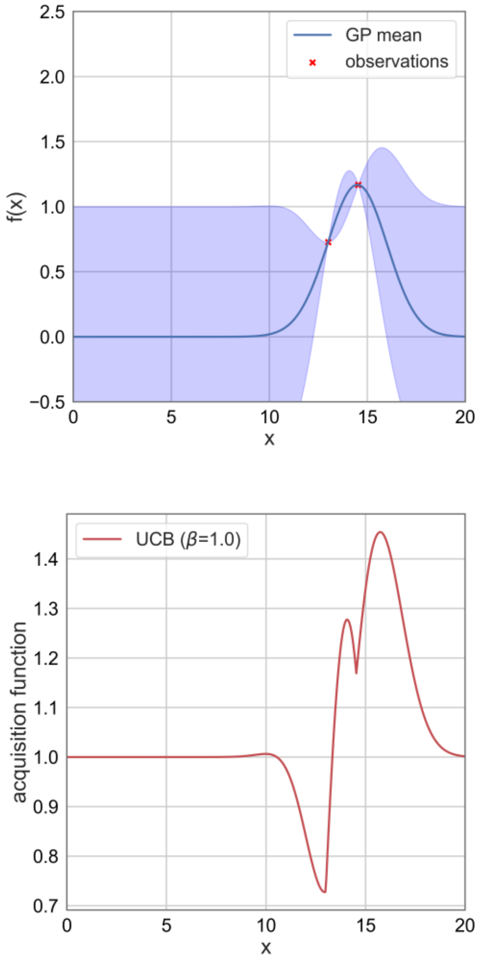
Martin Rudolph

Upper confidence bound (β = 1)

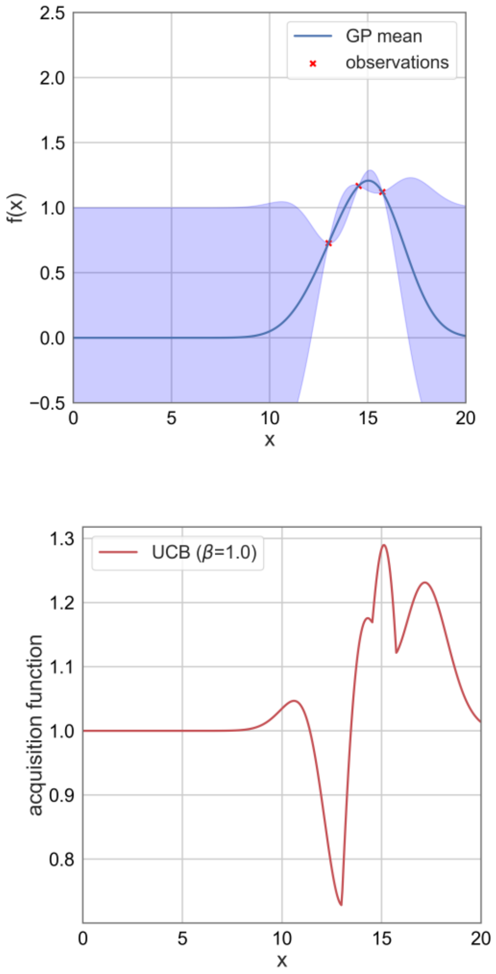
1



2

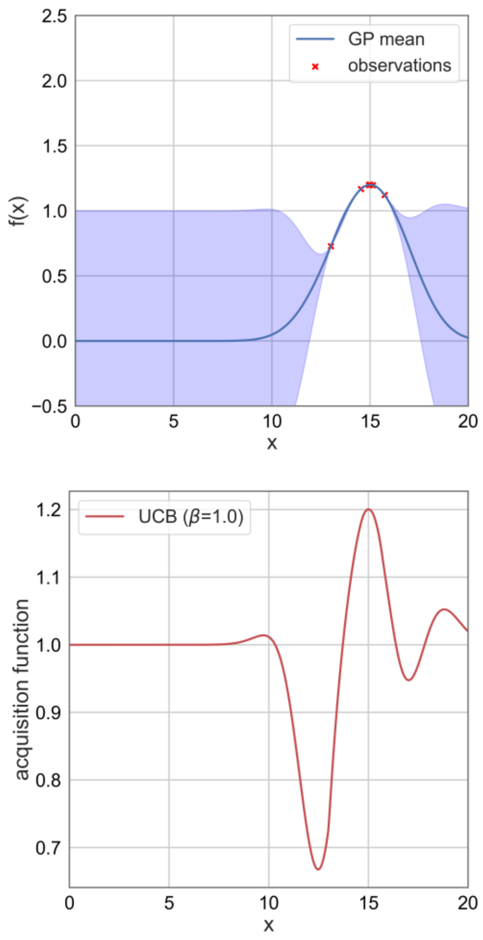


3

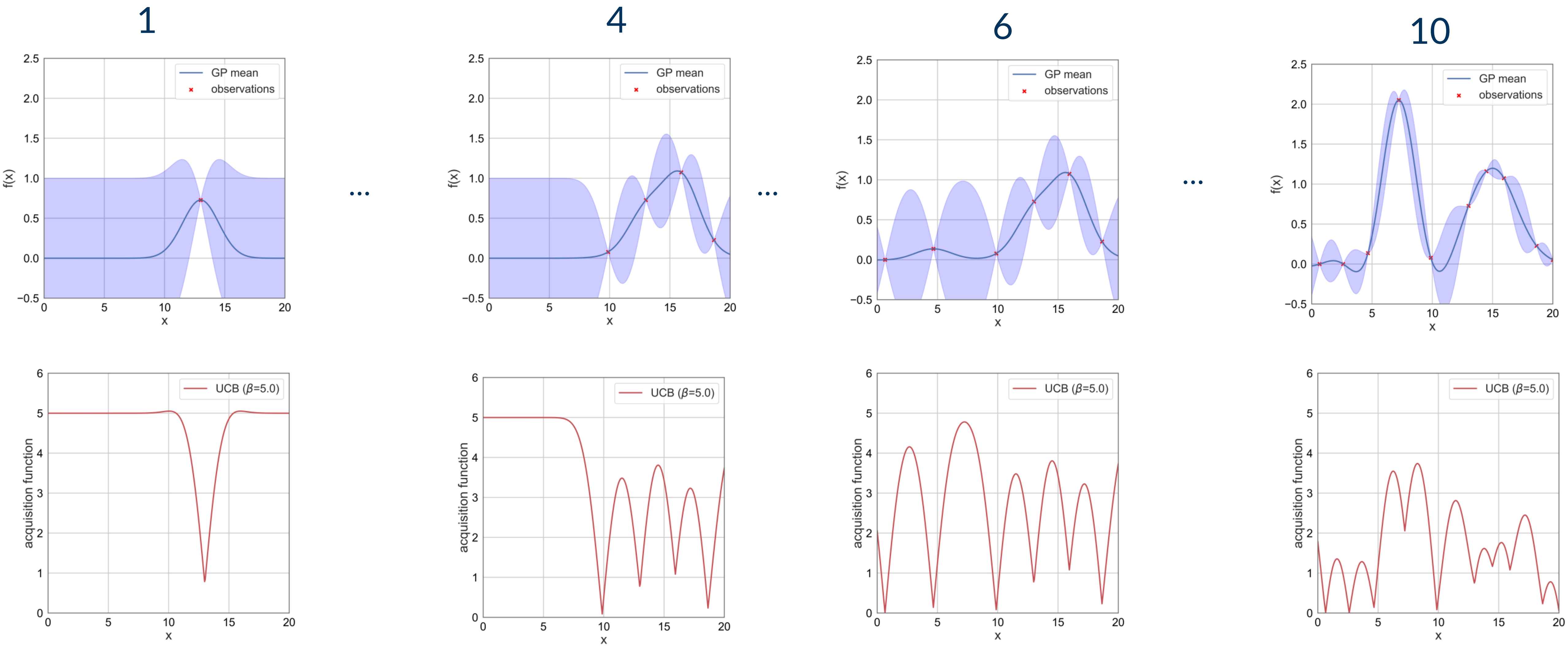


…

10



Upper confidence bound (β = 5 → promoting exploration)



Acquisition function II: Expected improvement

- Expected improvement

$$\alpha_{EI}(x) = (\mu(x) - f(x^*))\, \phi(z) \quad + \quad \sigma(x)\, \varphi(z)$$

$f(x^*)$ current best value

$$z(x) = \frac{\mu(x) - f(x^*)}{\sigma(x)}$$

$\phi(z)$ cumulative distribution function

$\varphi(z)$ probability distribution function

Acquisition function II: Expected improvement

- Expected improvement

$$\alpha_{EI}(x) = \underbrace{(\mu(x) - f(x^*))\,\phi(z)}_{\text{„exploitation term“}} + \underbrace{\sigma(x)\,\varphi(z)}_{\text{„exploration term“}}$$

expected gain
of improvement

chance
of achieving it

uncertainty

maximum, when μ(x) = f(x*),
and symmetric decrease
in both directions

$f(x^*)$ current best data point

$$z(x) = \frac{\mu(x) - f(x^*)}{\sigma(x)}$$

$\phi(z)$ cumulative distribution function

$\varphi(z)$ probability distribution function

Acquisition function II: Expected improvement

- Expected improvement in practice

$$\alpha_{EI}(x) = \underbrace{(\mu(x) - f(x^*) - \xi)\, \Phi(z)}_{\text{„exploitation term“}} + \underbrace{\sigma(x)\, \varphi(z)}_{\text{„exploration term“}}$$

$$z(x) = \frac{\mu(x) - f(x^*) - \xi}{\sigma(x)}$$

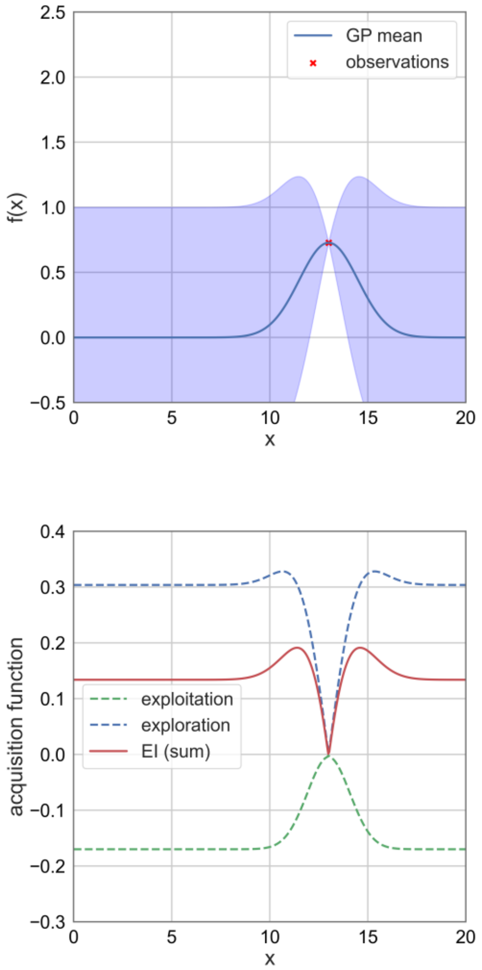
$\Phi(z)$ cumulative distribution function

$\varphi(z)$ probability distribution function

- Choose $\xi > 0$ for promoting exploration (e.g. 0.05 to 0.5)

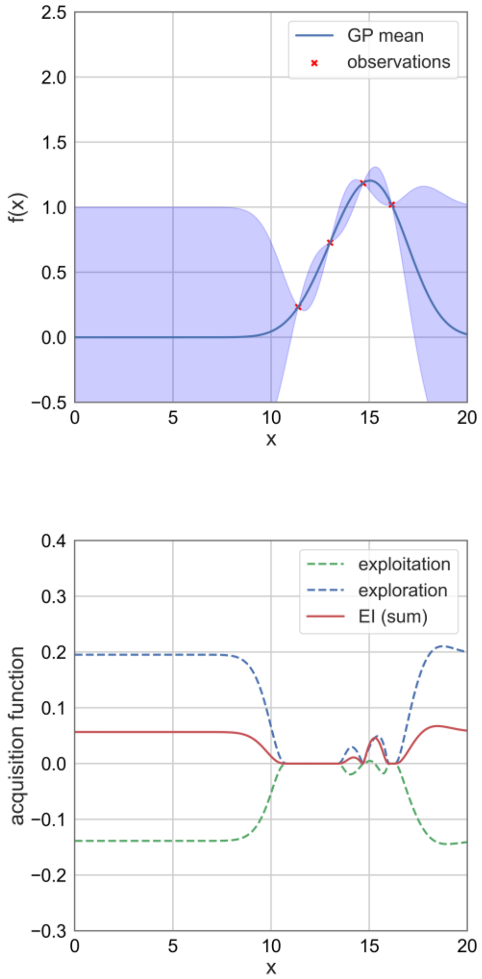
Expected improvement

1



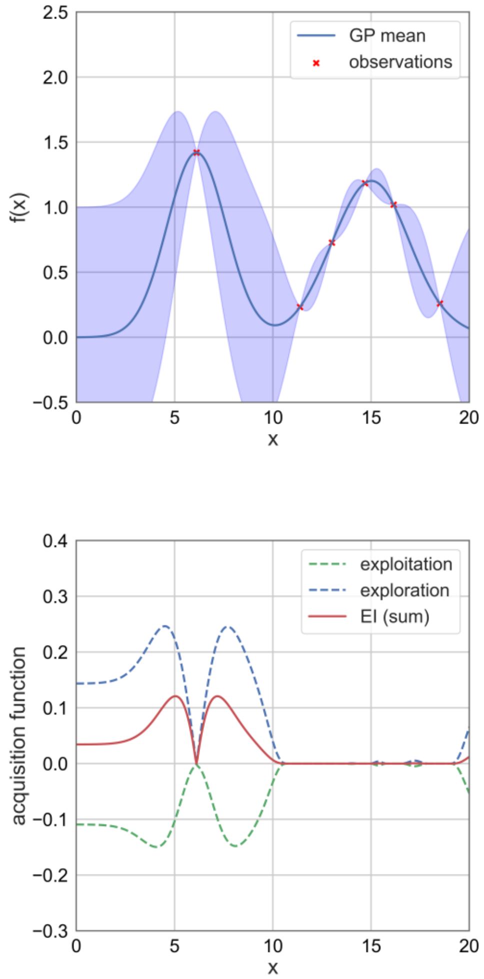
…

4



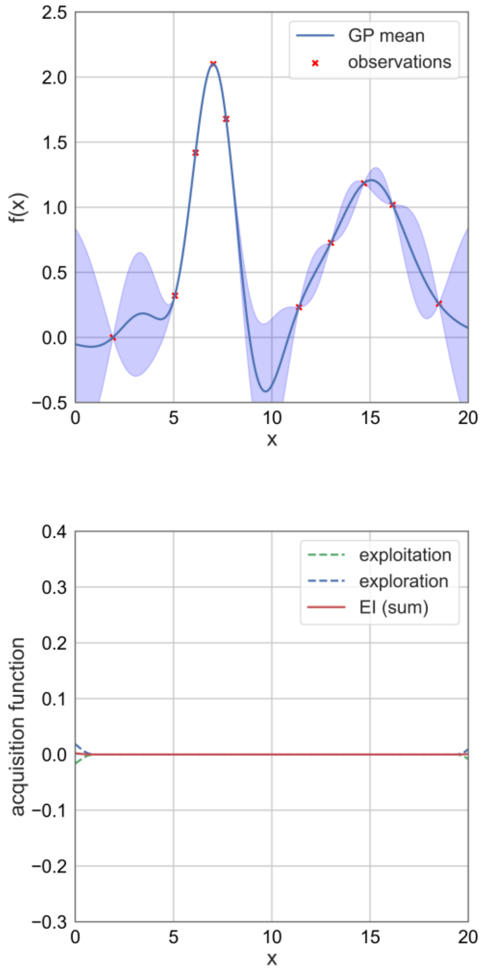
…

6



…

10



Tasks

- Open 04_bayesian_optimization.ipynb

- Approximate the black-box function
  - Do Gaussian Process Regression
  - Calculate the acquisition function (upper confidence bound)
  - Evaluate the black-box function and repeat

- Now use the expected improvement acquisition function

```python
def expected_improvement(X, gp, y_best, xi=0.01):

    mu, sigma = gp.predict(X, return_std=True)
    sigma = sigma.reshape(-1, 1)
    mu = mu.reshape(-1, 1)

    # avoid division by zero
    sigma = np.maximum(sigma, 1e-9)

    imp = mu - y_best - xi
    Z = imp / sigma

    ei = imp * norm.cdf(Z) + sigma * norm.pdf(Z)
    return ei.ravel()
```

Tasks

- Open 04_bayesian_optimization.ipynb

- Approximate the black-box function
  - Do Gaussian Process Regression
  - Calculate the acquisition function (upper confidence bound)
  - Evaluate the black-box function and repeat

  if the optimization gets stuck, adjust β

- Now use the expected improvement acquisition function

  if the optimization gets stuck, adjust ξ

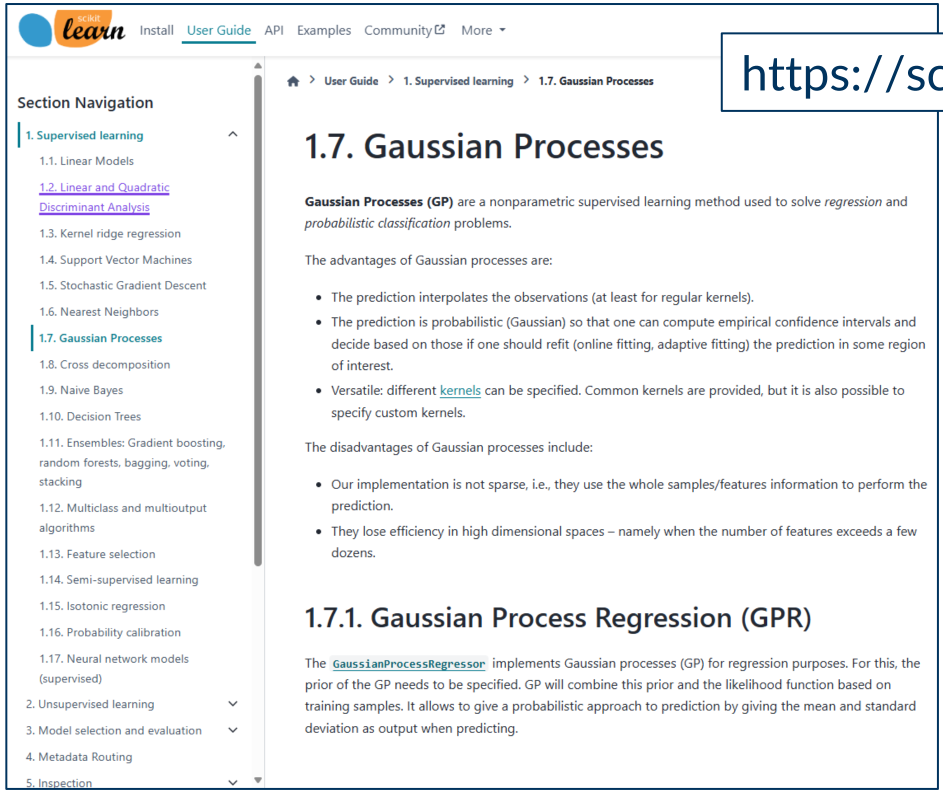
```
def expected_improvement(X, gp, y_best, xi=0.01):

    mu, sigma = gp.predict(X, return_std=True)
    sigma = sigma.reshape(-1, 1)
    mu = mu.reshape(-1, 1)

    # avoid division by zero
    sigma = np.maximum(sigma, 1e-9)

    imp = mu - y_best - xi
    Z = imp / sigma

    ei = imp * norm.cdf(Z) + sigma * norm.pdf(Z)
    return ei.ravel()
```

4 Final remarks

Summary

- „Coffee machine“ challenge
  - Numerical explosion
  - Drawbacks with polynomial regression
- Gaussian Process Regression
  - Provides mean and uncertainty
  - Non-parametric function
- Co-variance matrix / kernel function
  - Radial basis function kernel (RBF)
  - Periodic kernel
  - Linear kernel
- Bayesian optimization / acquisition function
  - Upper/lower confidence bound
  - Expected improvement

Literature: Online docs for practioners


Install User Guide API Examples Community More

Section Navigation

1. Supervised learning
1.1. Linear Models
1.2. Linear and Quadratic Discriminant Analysis
1.3. Kernel ridge regression
1.4. Support Vector Machines
1.5. Stochastic Gradient Descent
1.6. Nearest Neighbors
1.7. Gaussian Processes
1.8. Cross decomposition
1.9. Naive Bayes
1.10. Decision Trees
1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking
1.12. Multiclass and multioutput algorithms
1.13. Feature selection
1.14. Semi-supervised learning
1.15. Isotonic regression
1.16. Probability calibration
1.17. Neural network models (supervised)
2. Unsupervised learning
3. Model selection and evaluation
4. Metadata Routing
5. Inspection

User Guide > 1. Supervised learning > 1.7. Gaussian Processes

1.7. Gaussian Processes

Gaussian Processes (GP) are a nonparametric supervised learning method used to solve *regression* and *probabilistic classification* problems.

The advantages of Gaussian processes are:

- The prediction interpolates the observations (at least for regular kernels).
- The prediction is probabilistic (Gaussian) so that one can compute empirical confidence intervals and decide based on those if one should refit (online fitting, adaptive fitting) the prediction in some region of interest.
- Versatile: different kernels can be specified. Common kernels are provided, but it is also possible to specify custom kernels.

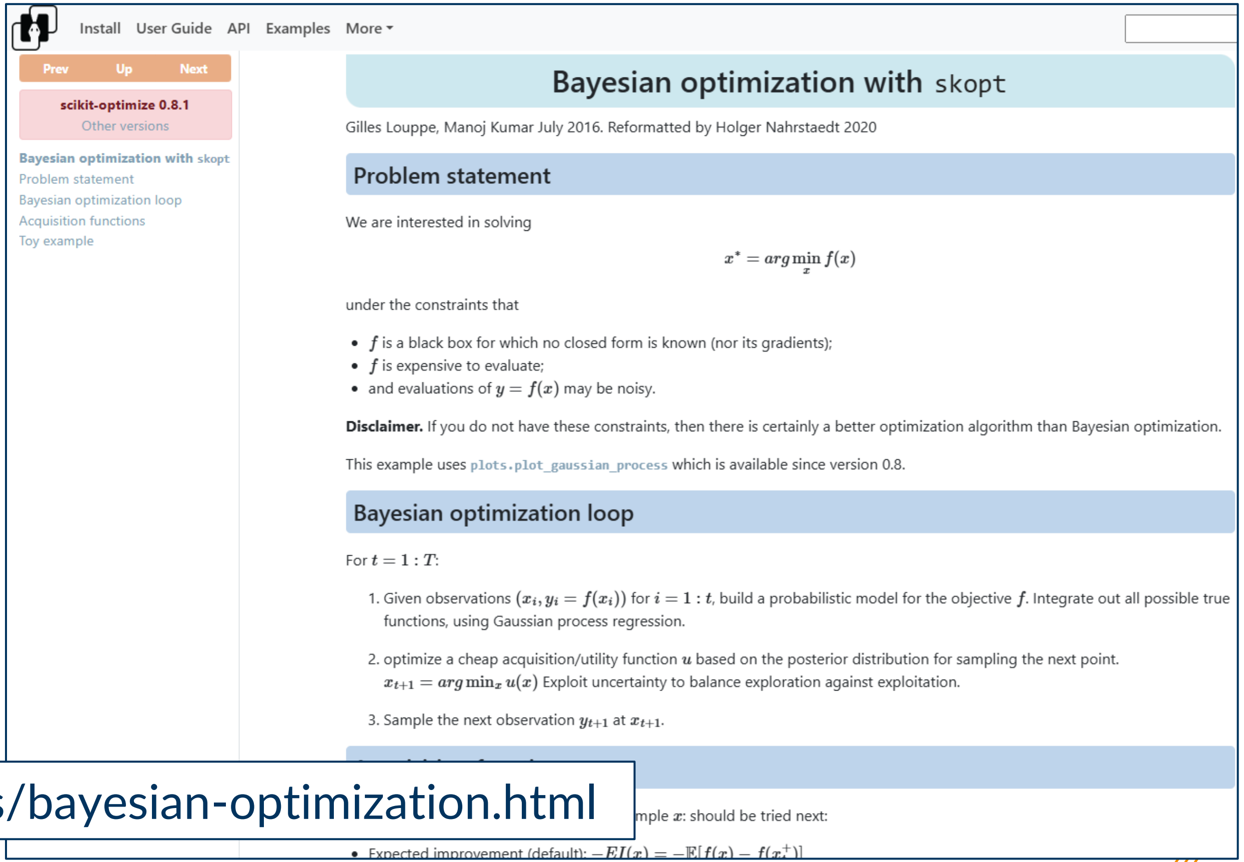
The disadvantages of Gaussian processes include:

- Our implementation is not sparse, i.e., they use the whole samples/features information to perform the prediction.
- They lose efficiency in high dimensional spaces – namely when the number of features exceeds a few dozens.

1.7.1. Gaussian Process Regression (GPR)

The `GaussianProcessRegressor` implements Gaussian processes (GP) for regression purposes. For this, the prior of the GP needs to be specified. GP will combine this prior and the likelihood function based on training samples. It allows to give a probabilistic approach to prediction by giving the mean and standard deviation as output when predicting.

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


Install User Guide API Examples More

Prev Up Next

scikit-optimize 0.8.1
Other versions

Bayesian optimization with skopt
Problem statement
Bayesian optimization loop
Acquisition functions
Toy example

Bayesian optimization with `skopt`

Gilles Louppe, Manoj Kumar July 2016. Reformatted by Holger Nahrstaedt 2020

Problem statement

We are interested in solving

$$x^* = arg \min_x f(x)$$

under the constraints that

- f is a black box for which no closed form is known (nor its gradients);
- f is expensive to evaluate;
- and evaluations of $y = f(x)$ may be noisy.

Disclaimer. If you do not have these constraints, then there is certainly a better optimization algorithm than Bayesian optimization.

This example uses `plots.plot_gaussian_process` which is available since version 0.8.

Bayesian optimization loop

For $t = 1 : T$:

1. Given observations $(x_i, y_i = f(x_i))$ for $i = 1 : t$, build a probabilistic model for the objective f. Integrate out all possible true functions, using Gaussian process regression.
2. optimize a cheap acquisition/utility function u based on the posterior distribution for sampling the next point. $x_{t+1} = arg \min_x u(x)$ Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

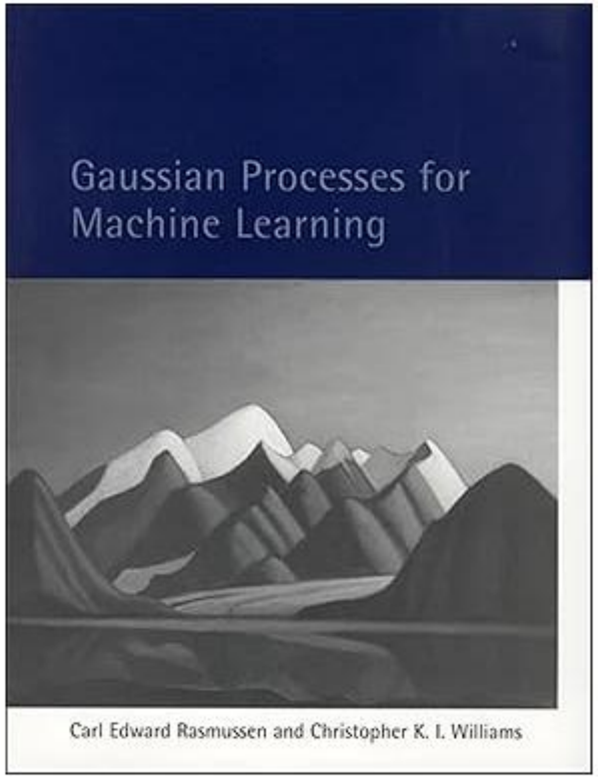
mple x: should be tried next:

- Expected improvement (default): $-EI(x) = -\mathbb{E}[f(x) - f(x_t^+)]$

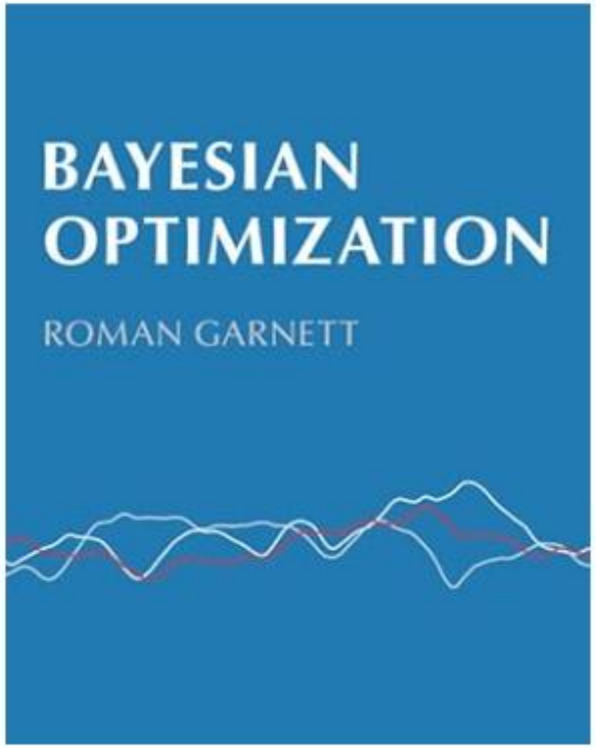
https://scikit-optimize.github.io/stable/auto_examples/bayesian-optimization.html

Books for further studies

Rasmussen and Williams: Gaussian Processes for Machine Learning, MIT Press, 2005.



Garnett: Bayesian Optimization, Cambridge University Press,2023.



Questions