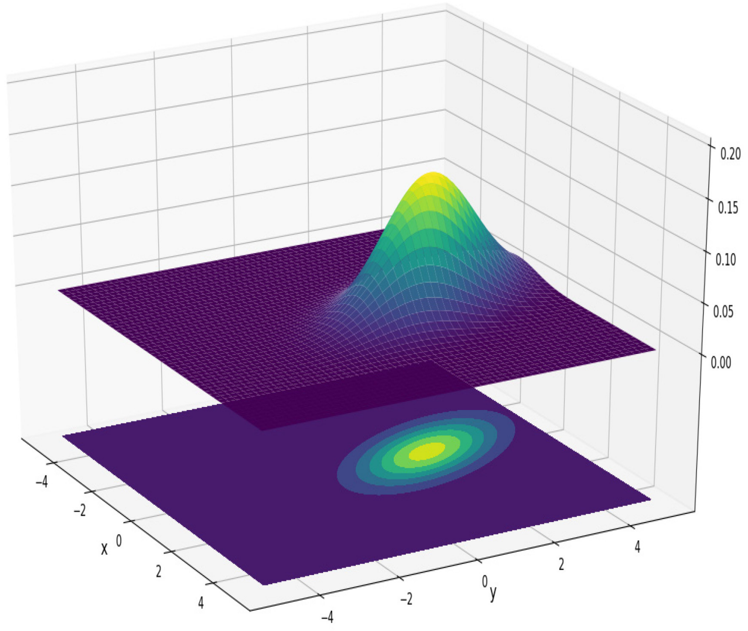
NTB
Interstaatliche Hochschule
für Technik Buchs

FHO Fachhochschule Ostschweiz



MSE MachLe Gaussian Processes

Christoph Würsch
Institute for Computational Enineering ICE
Interstaatliche Hochschule für Technik Buchs, FHO

Educational objectives

- You are able to apply Bayesian learning, especially **Bayes' theorem, Bayesian Regression** and the Bayes classifier
- You can explain the difference between a **maximum likelihood** (ML) estimate and a **Bayesian posterior estimation** of the parameters for a regression problem.
- You know the most important properties of Gaussian distributions (conditional, marginal, product and sum)
- You know a **Gaussian Process** as a generalization of a multivariate Gaussian distribution.
- You are able to construct appropriate **kernel functions** for a regression problem using a Gaussian Process and
- You are able to **sample functions** from a Gaussian process and to **fit functions** to given data (infernence) and to use Gaussian processes for **making predictions**

Based on material by
- Stuart Russell, UC Berkeley
- T. Stadelmann, R. Ewerth & B. Freisleben, U Marburg

Why probabilistic inference in Machine Learning?

- *probabilities* provide a language for representing uncertainty
- in *learning systems* knowledge is typically uncertain
- observed data provide evidence for and against explanations
- a learner must be able to "weigh the evidence"
- probabilities are used to
  - **represent knowledge**
  - **infer the new state of knowledge in the light of new observations**
  - **make optimal decisions**
- probabilistic models are equivalent to other views of learning:
  - information theoretic
  - physical analogies: minimizing free energy

What is a Gaussian process?

- Frequently, it is referred to as the **infinite-dimensional extension of the multivariate normal distribution**.
- This may be confusing, because we typically don't observe random variables with infinitely many components. However, when we work with GPs, the intuition is that we observe some **finite-dimensional subset** of infinite-dimensional data, and this finite subset follows a multivariate normal distribution, as would every finite subset.
- **Example**: we measure the temperature every day of the year at noon, resulting in a 365-dimensional vector. In reality, temperature is a continuous process, and the choice to take a measurement every day at noon is arbitrary. What would happen if we took the temperature in the evening instead? What if we took measurements every hour or every week? If we model the data with a GP, we are assuming that each of these possible data collection schemes would yield data from a multivariate normal distribution.

Importance of Gaussians

Gaussian random variables are extremely useful in machine learning and statistics for two main reasons.

- First, they are extremely common when **modeling "noise"** in statistical algorithms.
- Second, Gaussian random variables are convenient for many analytical manipulations, because many of the integrals involving Gaussian distributions that arise in practice have **simple closed form solutions**.

Properties of Multivariate Gaussians

Covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}$$

Precision matrix

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

- **Conditional** probability distribution

$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}\left(\mathbf{x}_a|\boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}\right)$$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b)$$

$$\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Lambda}_{aa}^{-1}$$

Conditional Probability Distribution of a Gaussian

- **Marginal** probability distribution:

$$p(\mathbf{x}_a) = \int_{\mathbf{x}_b} p(\mathbf{x}_a, \mathbf{x}_b|\boldsymbol{\mu}, \boldsymbol{\Sigma})d\mathbf{x}_b = \mathcal{N}\left(\mathbf{x}_a|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}\right)$$

$$p(\mathbf{x}_b) = \int_{\mathbf{x}_a} p(\mathbf{x}_a, \mathbf{x}_b|\boldsymbol{\mu}, \boldsymbol{\Sigma})d\mathbf{x}_a = \mathcal{N}\left(\mathbf{x}_b|\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_{bb}\right)$$

I. Bayesian Regression

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta) \cdot p(\theta)}{\int_{\theta'} p(\mathcal{D}|\theta') \cdot p(\theta') d\theta'}$$

Parameter Posterior

$$p(y_*|x_*\mathcal{D}) = \int_{\theta} p(y_*|x_*, \theta) \cdot p(\theta|\mathcal{D}) d\theta$$

posterior predictive distribution

Bayesian Linear Regression

- Let $D = \{x^{(i)}, y^{(i)}\}$ $(i = 1 \dots m)$ be a training set of iid. examples from some unknown distribution. The standard probabilistic interpretation of linear regression states that

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$$

- Where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is iid. noise. It follows that:

$$y^{(i)} - \theta^T x^{(i)} \sim \varepsilon^{(i)} = \mathcal{N}(0, \sigma^2)$$
$$p(y^{(i)} | x^{(i)}, \theta) = \frac{1}{(2\pi\sigma^2)} \exp\left\{-\frac{1}{2\sigma^2}\left(y^{(i)} - \theta^T x^{(i)}\right)^2\right\}$$

- In Bayesian linear regression**, we assume that a prior distribution over parameters** is also given; a typical choice, for instance, is:

$$p(\theta) \sim \mathcal{N}(0, \sigma_0^2)$$

Bayesian Linear Regression II

- Using Bayes's rule, we obtain the **parameter posterior**:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)\cdot p(\theta)}{\int_{\theta'} p(\mathcal{D}|\theta')\cdot p(\theta')d\theta'} = \frac{p(\theta)\cdot \prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)}{\int_{\theta'} p(\theta)\cdot \prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)d\theta'} \tag{1}$$

- Using this posterior, it is now possible, the **make predictions** on some unknown data: the «output» of Bayesian linear regression on a new test point x_* is not just a single guess y_*, but rather an **entire probability distribution** over possible outputs, known as the **posterior predictive distribution**:

$$p(y_*|x_*\mathcal{D}) = \int_{\theta} p(y_*|x_*,\theta)\cdot p(\theta|\mathcal{D})d\theta \tag{2}$$

Bayesian Linear Regression III

- For many types of models, the integrals in (1) and (2) are difficult to compute, and hence, we often resort to approximations, **such as MAP** estimation.
- In the case of **Bayesian linear regression**, however, the integrals actually are tractable: (after much work!)

$$p(\theta|\mathcal{D}) \sim \mathcal{N}\left(\frac{1}{\sigma^2}\mathbf{\Lambda}^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{Y}, \Lambda^{-1}\right)$$

$$p(y_*|x_*, \mathcal{D}) \sim \mathcal{N}\left(\frac{1}{\sigma^2}x_*^T\mathbf{\Lambda}^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{Y}, x_*^T\mathbf{\Lambda}^{-1}x_* + \sigma^2\right)$$

$$\mathbf{\Lambda} = \frac{1}{\sigma^2}\mathbf{X}^{\mathbf{T}}\mathbf{X} + \frac{1}{\sigma_0^2}\mathbb{1}$$

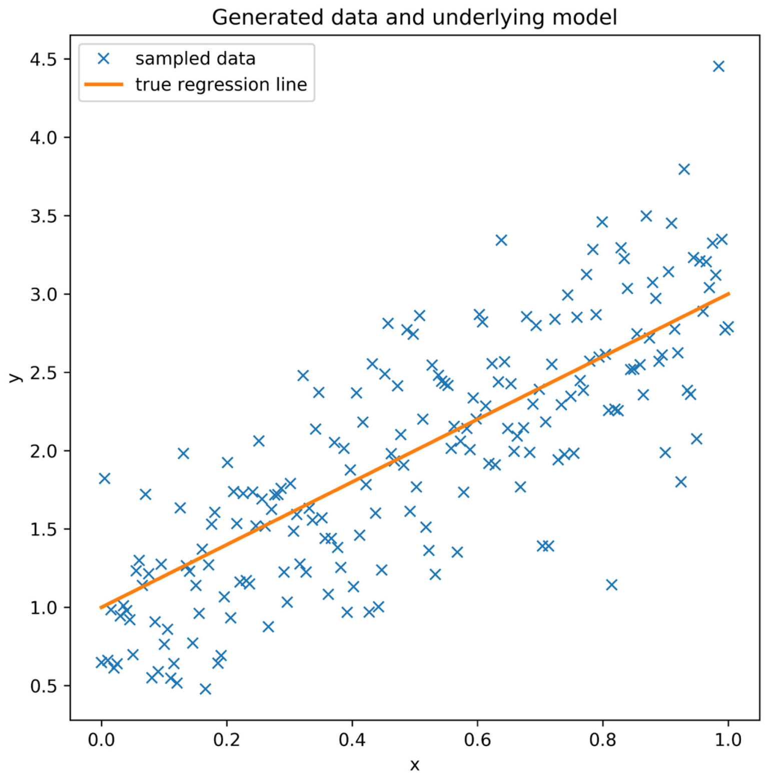
- In case of general prior distributions, **Monte Carlo Methods** and smart sampling methods are used to estimate the posterior distribution (eg. using `pymc3`)

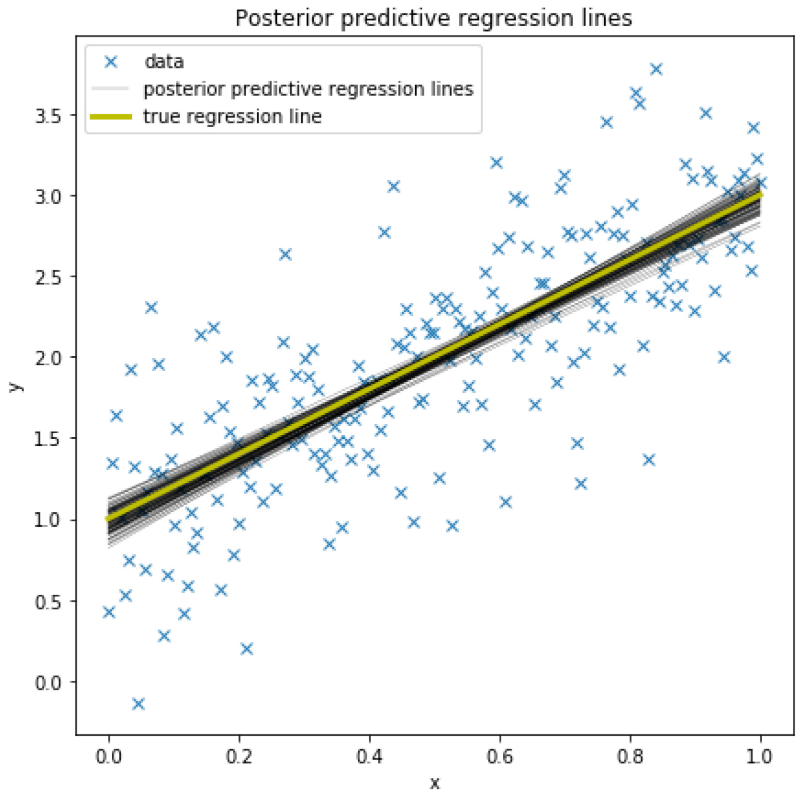
Example: Bayesian Linear Regression using `pymc3`

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- True model:

$$y^{(i)} = 2.0\,x^{(i)} + 1.0 + \varepsilon^{(i)}$$
$$\varepsilon \sim \mathcal{N}(0, 0.5^2)$$





Example: Bayesian Linear Regression using pymc3

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

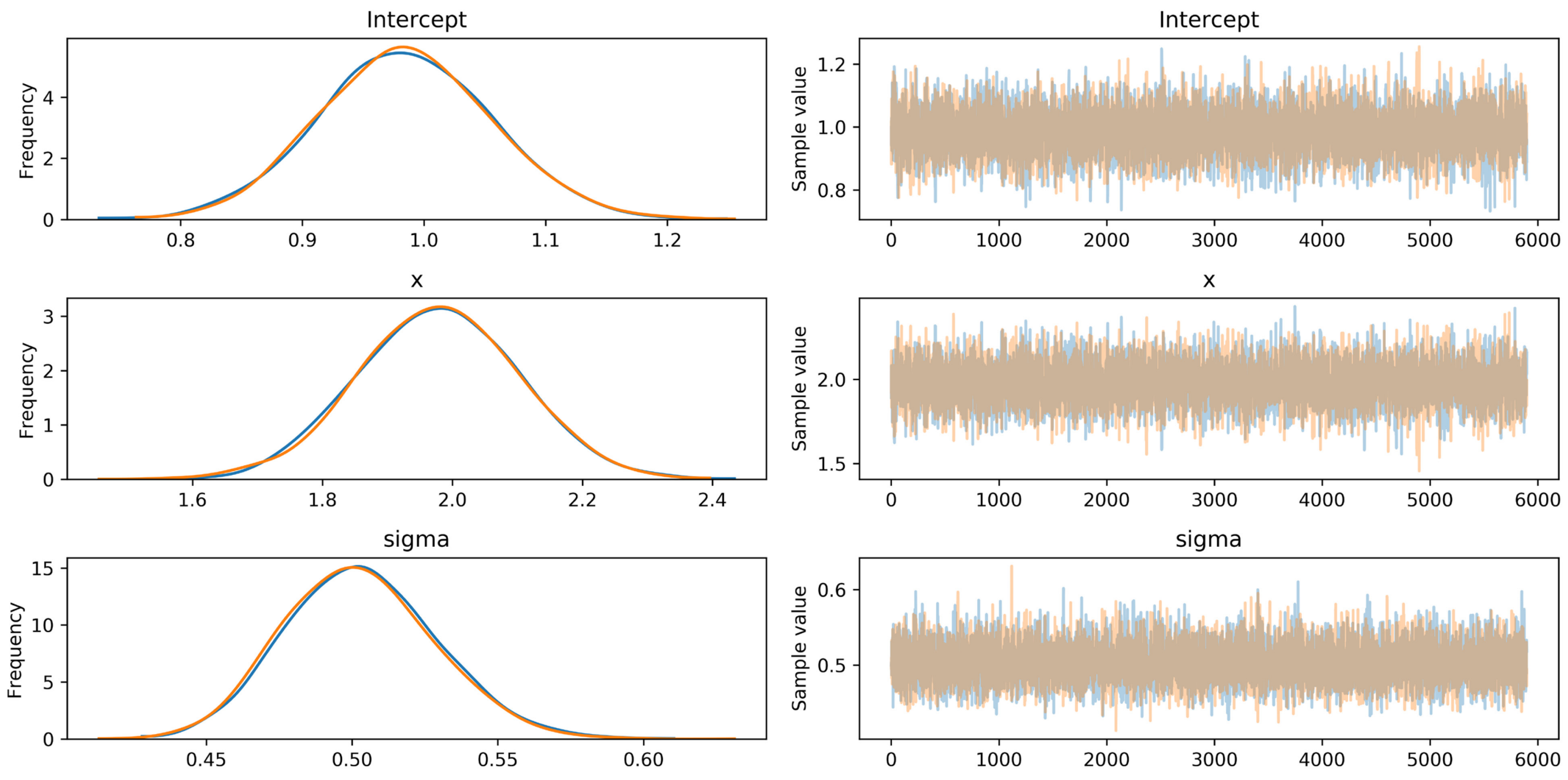
- **Pymc3** is a comfortable package for Bayesian Inference

```
from pymc3 import *

with Model() as model:
# model specifications in PyMC3 are wrapped in a with-statement

# Define priors (#tau = precision = 1/variance = 1/sigma^2)
sigma = HalfCauchy('sigma', beta=10, testval=1.)
intercept = Normal('Intercept', 0, tau=1/20**2)
x_coeff = Normal('x', 0, tau=1/20**2)

# Define likelihood
likelihood = Normal('y', mu=intercept + x_coeff * x,
     tau=1/sigma**2, observed=y)

# Inference! trace = sample(3000, cores=2)

# draw 3000 posterior samples using NUTS sampling
plt.figure(figsize=(7, 7))
traceplot(trace[100:]); plt.tight_layout();
```

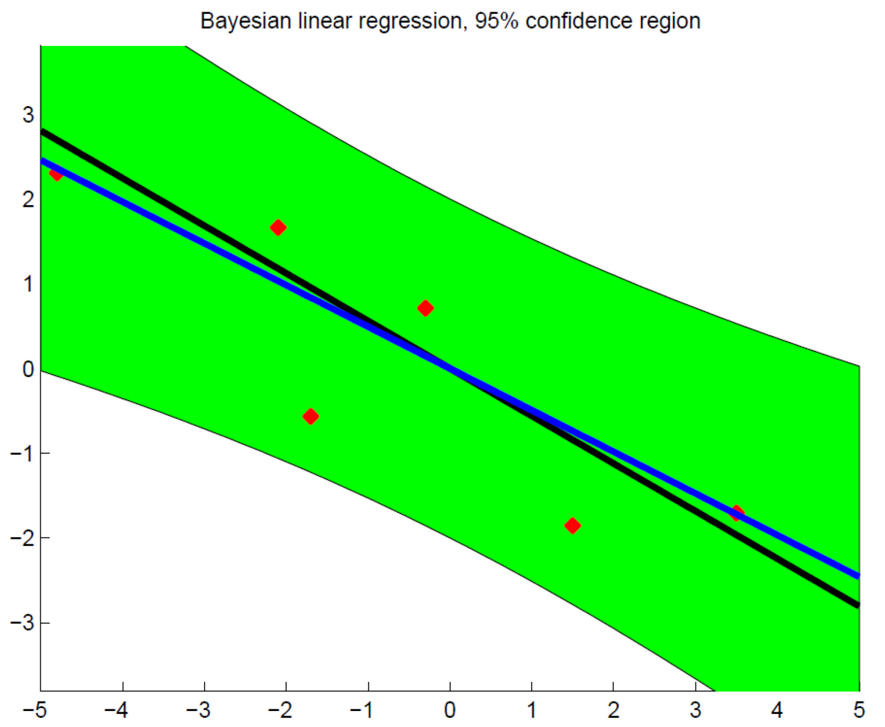
See also: https://docs.pymc.io/notebooks/GLM-linear.html

Advantage of Byesian Infererence: Posterior !

- Bayesian inference does not give us only one best fitting line (as maximum likelihood does) but rather a **whole posterior distribution of likely parameters**. → similar to p-values of statsmodels, but more general.



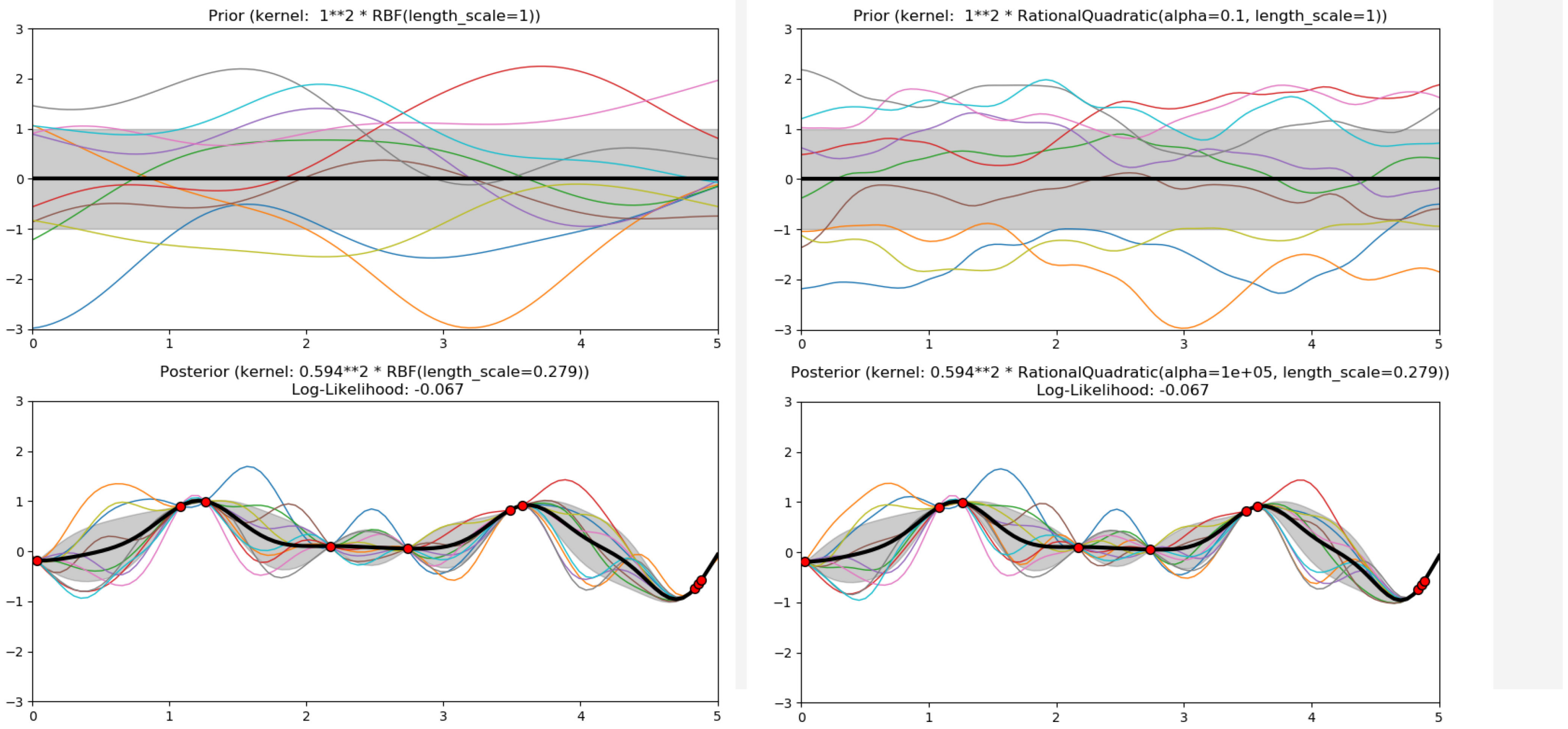
Confidence region



Bayesian linear regression for a one-dimensional linear regression problem, $y^{(i)} = \theta x^{(i)} + \varepsilon^{(i)}$ with $\varepsilon^{(i)} \sim \mathcal{N}(0,1)$ i.i.d. noise. The green region denotes the 95% confidence region for predictions of the model. Note that the (vertical) width of the green region is largest at the ends but narrowest in the middle. This region reflects the uncertain in the estimates for the parameter θ. In contrast, a classical linear regression model would display a confidence region of constant width, reflecting only the $\mathcal{N}(0,\sigma^2)$ noise in the outputs.

II. Gaussian Processes

$$h(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{m}(\mathbf{x}), \mathbf{k}(\mathbf{x}, \mathbf{x}')\right)$$



Gaussian Processes $(\mathcal{GP})$

$$h(\mathbf{x}) \sim \mathcal{N}\left(\mathbf{m}(\mathbf{x}), \mathbf{k}(\mathbf{x}, \mathbf{x}')\right)$$

- **Gaussian processes** $(\mathcal{GP})$ are the extension of multivariate Gaussians to infinite-sized collections of real-valued variables.
- Think of Gaussian processes as distributions not just over random vectors but in fact **distributions over random functions**.
- A $\mathcal{GP}$ can be seen as a **prior over functions**. Samples of a Gaussian process are functions whose values are drawn from a infinite multivariate Gaussian distribution.
- **A stochastic process** is a **collection of random variables**, $\{h(x): x \in \mathcal{X}\}$ indexed by elements from some set $\mathcal{X}$, known as the **index set**.

Definition of a Gaussian Process $(\mathcal{GP})$

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- A Gaussian process is a stochastic process such that **any finite subcollection of random variables** has a **multivariate Gaussian distribution**.

- In particular, a collection of random variables $\{h(x): x \in \mathcal{X}\}$ is said to be drawn from a Gaussian process with **mean function $m(\cdot)$** and **covariance function $k(\cdot\,,\cdot\,)$** if for any finite set of elements $x_1, x_2, \dots x_m \in \mathcal{X}$, the associated finite set of random variables $h(x_1), h(x_2), \dots, h(x_m)$ have a distribution:

$$\begin{bmatrix} h(x_1) \\ \vdots \\ h(x_m) \end{bmatrix} \sim \mathcal{N}\left\{ \begin{bmatrix} m(x_1) \\ \vdots \\ m(x_m) \end{bmatrix}, \underbrace{\begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_m) \\ \vdots & \ddots & \vdots \\ k(x_m, x_1) & \dots & k(x_m, x_m) \end{bmatrix}}_{\text{Kernel Matrix}} \right\}$$

$$h(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{m}(\mathbf{x}), \mathbf{k}(\mathbf{x}, \mathbf{x}')\right)$$

Random functions from a Gaussian Process

- Example **one dimensional** Gaussian process:

$$p\left(f(x)\right) \sim \mathcal{GP}\left\{m(x) = 0, k(x, x') = \exp\left(-\frac{1}{2}(x - x')^2\right)\right\}$$

- To get an indication of what this distribution over functions looks like, focus on a finite subset of function values $\mathbf{f} = \left(f(x_1), f(x_2), \ldots, f(x_n)\right)$ for which

$$\mathbf{f} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}\right)$$
$$\mathbf{\Sigma}_{ij} = k(x_i, x_j)$$

- Then plot the coordinates of $\mathbf{f}$ as a function of the corresponding x values.

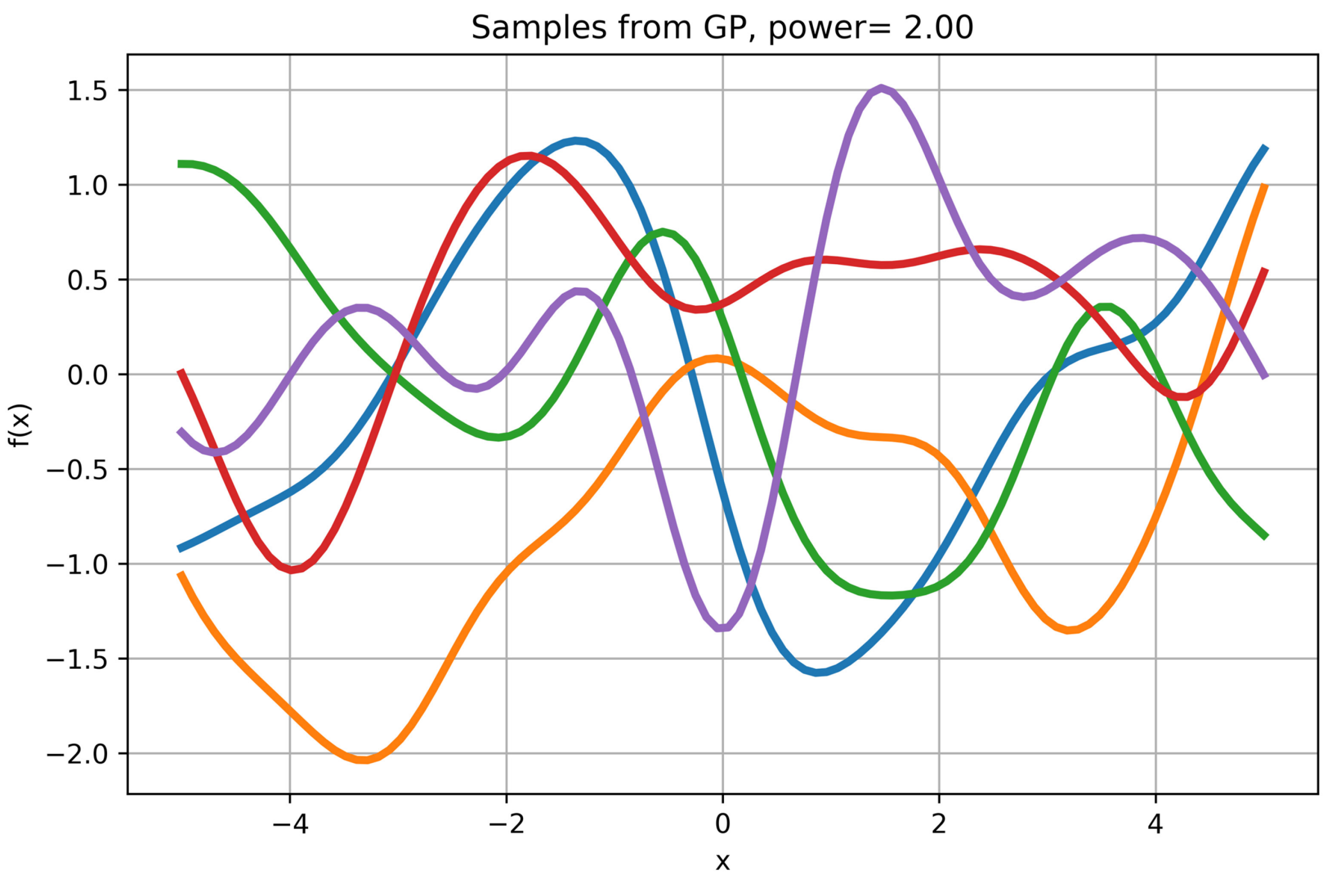
Sampling from a gaussian Process (Prior)

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

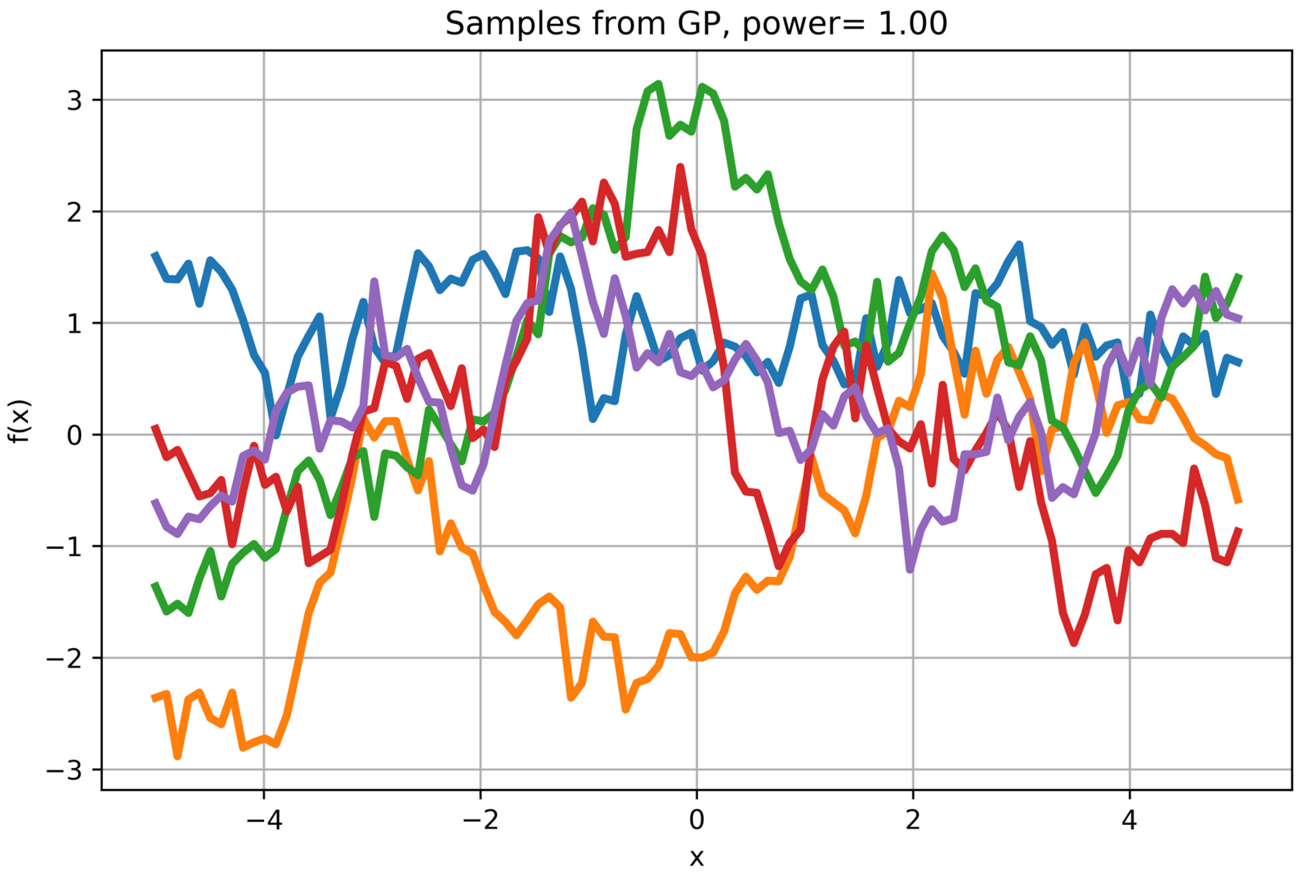
```
import numpy as np
from matplotlib import pyplot as plt
from scipy.spatial import distance_matrix as dm

#%% Sample a function from a Gaussian Process (GP)
N=100; power=2;
X = np.linspace(-5, 5, N).reshape(N,1) ;

#calculate the distance matrix
D=dm(X,X,p=2) mean =np.zeros(N)
cov =np.exp(-1/2*np.power(D,power))
y=np.random.multivariate_normal(mean, cov, 5).T

fig1=plt.figure(figsize=(8,5))
plt.plot(X,y,lw=3); plt.grid(True);
plt.xlabel('x'); plt.ylabel('f(x)')
```

Samples from a Gaussian Process

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

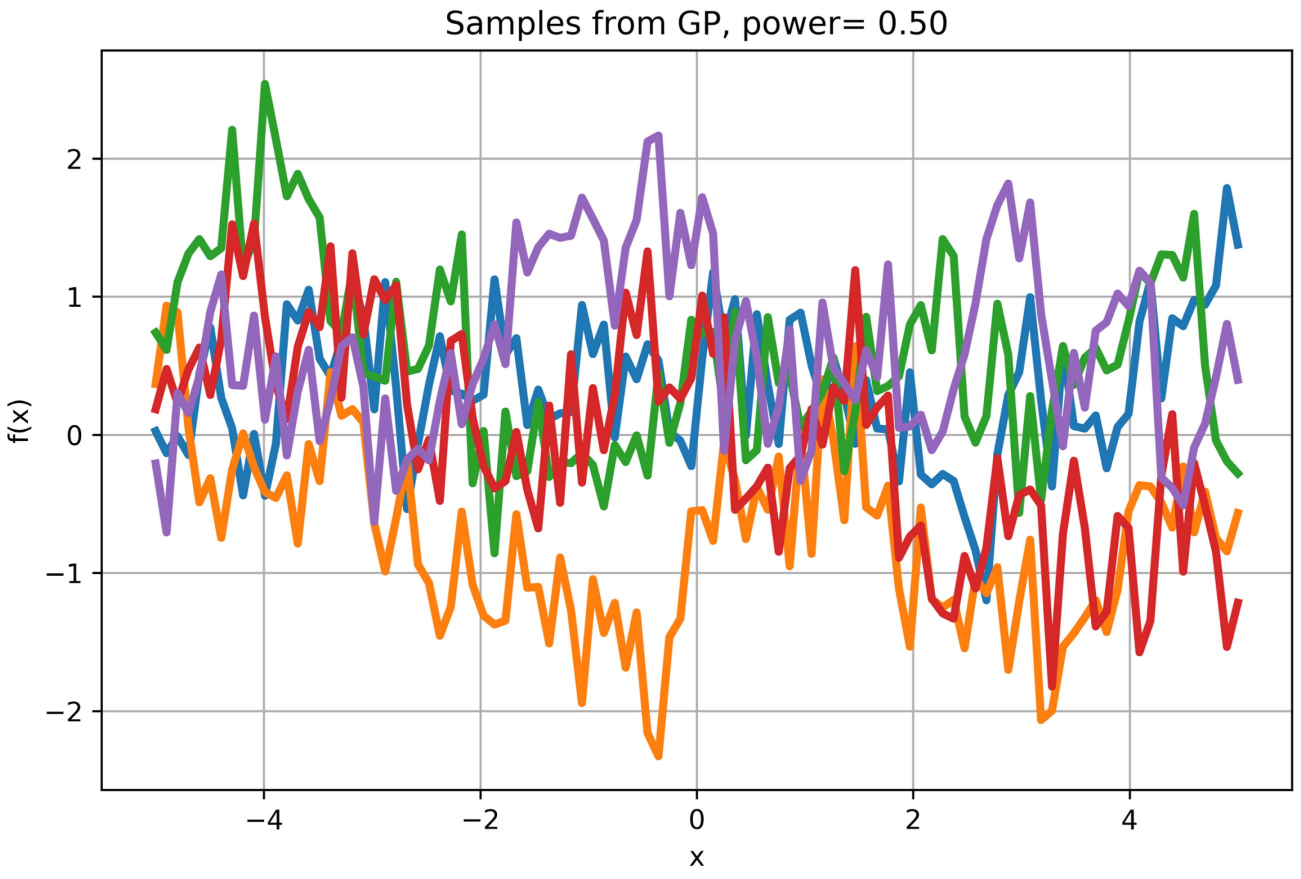


Samples from a Gaussian Process



Samples from a Gaussian Process

NTB
Interstaatliche Hochschule
für Technik Buchs
FHO Fachhochschule Ostschweiz



Non-parametric Gaussian process models

- In our non-parametric model, the "parameters" is the function itself!
- **Gaussian likelihood**

$$p(\mathbf{y}|\mathbf{x}, f(x), M_i) \sim \mathcal{N}\left(\mathbf{f}, \mathbb{1}\sigma_n^2\right)$$

- (Zero mean) Gaussian **process prior**:

$$p(f(x)|M_i) \sim \mathcal{GP}\left\{m(x) \equiv 0, k(x, x')\right\}$$

Non-parametric Gaussian process models

- Leads to a Gaussian **process posterior**

$$p(f(x)|\mathbf{x},\mathbf{y},M_i) \sim \mathcal{GP}\left\{m_p, k_p(x,x')\right\}$$

$$m_p(x) = k(x,\mathbf{x})\left[K(\mathbf{x},\mathbf{x}) + \mathbb{1}\sigma_n^2\right]^{-1}\mathbf{y}$$

$$k_p(x,x') = k(x,x') - k(x,\mathbf{x})\left[K(\mathbf{x},\mathbf{x}) + \mathbb{1}\sigma_n^2\right]^{-1}k(\mathbf{x},x')$$

- And a Gaussian **predictive distribution**:

$$p(y^*|x^*,\mathbf{x},\mathbf{y},M_i) \sim \mathcal{GP}\left\{m^*, k^*(x,x')\right\}$$

$$m^*(x) = k(x^*,\mathbf{x})\left[K(\mathbf{x},\mathbf{x}) + \mathbb{1}\sigma_n^2\right]^{-1}\mathbf{y}$$

$$k^*(x^*,x^*) = k(x^*,x^*) - k(x^*,\mathbf{x})\left[K(\mathbf{x},\mathbf{x}) + \mathbb{1}\sigma_n^2\right]^{-1}k(x^*,\mathbf{x})$$

Non-parametric Gaussian process models (in short)

- In our non-parametric model, the "parameters" is the function itself!
- **Gaussian likelihood**

$$p(\mathbf{y}|\mathbf{x}, f, M_i) \sim \mathcal{N}\left(\mathbf{f}, \mathbb{1}\sigma_n^2\right)$$

- (Zero mean) Gaussian **process prior**:

$$p(f|M_i) \sim \mathcal{GP}\left\{\mathbf{m}, \mathbf{k}\right\}$$

- **Posterior «functions» distribution** (over functions)

$$p(f|\mathbf{x}, \mathbf{y}, M_i) = \frac{p(\mathbf{y}|\mathbf{x}, f, M_i) \cdot p(f|M_i)}{p(\mathbf{y}|\mathbf{x}, M_i)}$$

- **Predictive distribution**:

$$p(y^*|x^*, \mathbf{x}, \mathbf{y}, M_i) = \int p(y^*|x^*, f, M_i) \cdot p(f|\mathbf{x}, \mathbf{y}, M_i) df$$

Marginalization Property → finite subsets

- Intuitively, one can think of a function f(·) drawn from a Gaussian process prior as an extremely high-dimensional vector drawn from an extremely high-dimensional multivariate Gaussian.
- Here, each dimension of the Gaussian corresponds to an element x from the index set $\mathcal{X}$, and the corresponding component of the random vector represents the value of f(x).
- Using the **marginalization property** for multivariate Gaussians, we can obtain the marginal multivariate Gaussian density corresponding to any *finite subcollection of variables*.

What sort of functions m(·) and k(·, ·) give rise to valid Gaussian processes?

Covariance Function k = Mercer kernel

NTB Interstaatliche Hochschule für Technik Buchs

FHO Fachhochschule Ostschweiz

- The positive semidefiniteness requirement for covariance matrices computed based on arbitrary input points is identical to **Mercer's condition for kernels!**
- A function k(·, ·) is a valid kernel provided the resulting **kernel matrix K** defined as above is always positive semidefinite for any set of input points $x_1, x_2, \ldots x_m \in \mathcal{X}$.
- **Gaussian processes, therefore, are kernel-based probability distributions in the sense that any valid kernel function can be used as a covariance function!**

Kernel properties

- If $k_1(x,y)$ and $\mathrm{k}_2(x,y)$ are kernels, then the following functions are also kernels:

$$k(x,y) = \langle \phi(x) | \phi(y) \rangle_{\mathcal{F}}$$

$$k(x,y) = k_1(x,y) + k_2(x,y)$$
$$k(x,y) = \alpha \cdot k_1(x,y) \qquad \text{where } \alpha > 0$$
$$k(x,y) = f(x) \cdot f(y) \qquad \text{for any function } f(x)$$
$$k(x,y) = k_1(x,y) \cdot k_2(x,y)$$

gp = GaussianProcessRegressor(kernel=kernel)

NTB Interstaatliche Hochschule für Technik Buchs

FHO Fachhochschule Ostschweiz

```
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF

kernel = 1.0 * RBF(length_scale=1.0,
        length_scale_bounds=(1e-1, 10.0))
# Specify Gaussian Process
gp = GaussianProcessRegressor(kernel=kernel)
X_ = np.linspace(0, 5, 100)
y_mean, y_std = gp.predict(X_[:, np.newaxis], return_std=True) #

Generate data and fit GP
rng = np.random.RandomState(4)
X = rng.uniform(0, 5, 10)[:, np.newaxis]
y = np.sin((X[:, 0] - 2.5) ** 2)
gp.fit(X, y)
y_mean, y_std = gp.predict(X_[:, np.newaxis], return_std=True) #sample
from the posterior
y_samples = gp.sample_y(X_[:, np.newaxis], 10)
```

Start: `Lab9b_plot_gpr_prior_posterior.jypnb`

1. RBF kernel (squared exponential)

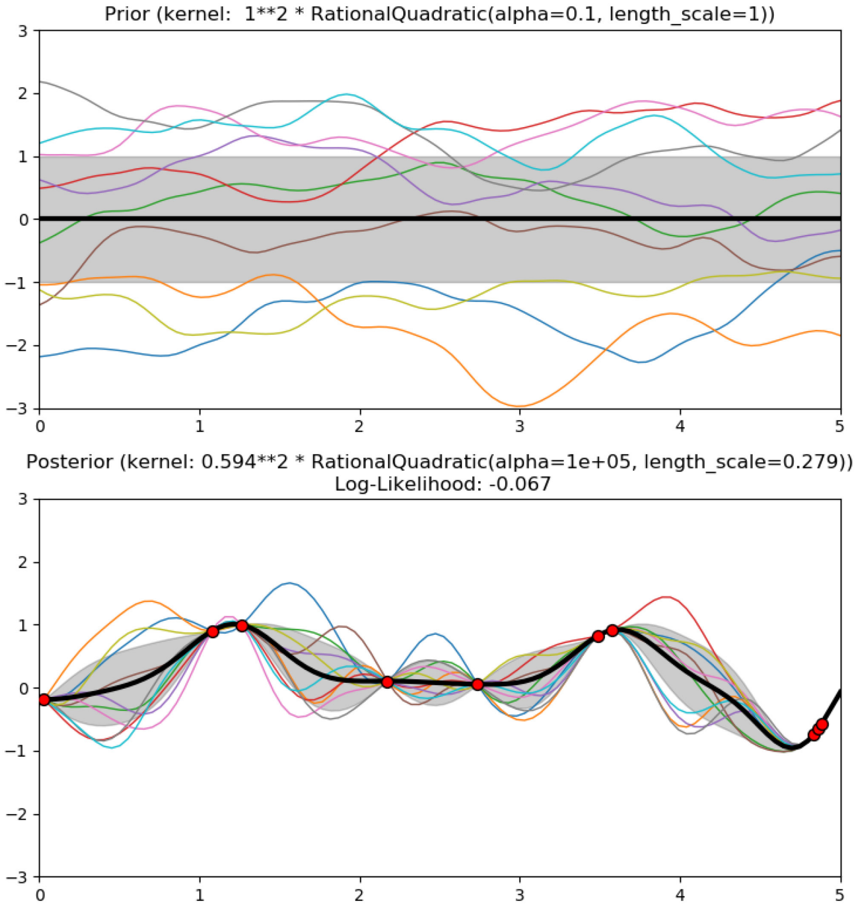
$$k(x,x') = \sigma_0^2 \exp\left[-\frac{1}{2}\left(\frac{x-x'}{\lambda}\right)^2\right]$$

- The **radial basis function kernel or short RBF kernel** is a stationary kernel. Stationary means, the kernel $k(x,x') = k(x-x')$ is invariant to translations.
- It is also known as the “squared exponential” kernel. It is parameterized by a length-scale parameter λ, which can either be a scalar (isotropic variant of the kernel) or a vector with the same number of dimensions as the inputs (anisotropic variant of the kernel).
- This kernel is **infinitely differentiable**, which implies that a $\mathcal{GP}$ with this kernel as covariance function have mean square derivatives of all orders, and **are thus very smooth**.

2. Rational-Quadratic kernel

$$k(x, x') = \left(1 + \frac{(x - x')^2}{2\alpha\lambda^2}\right)^{-\alpha}$$

- The **RationalQuadratic kernel** can be seen as a scale mixture (an infinite sum) of RBF kernels with different characteristic length scales.
- It is parameterized by a length-scale parameter λ and a scale mixture parameter α.

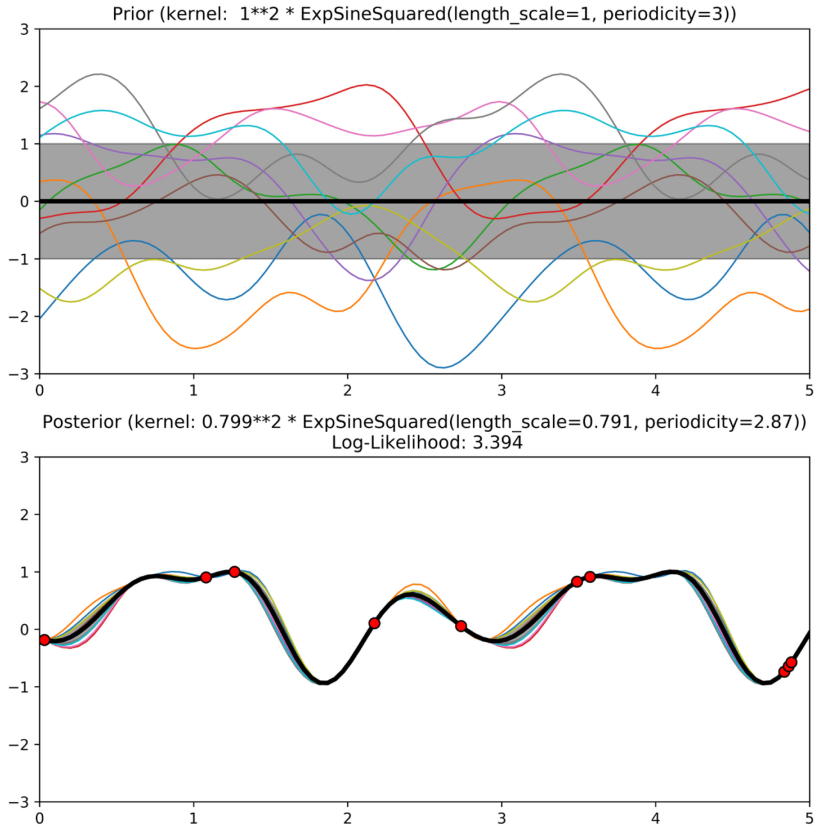


3. Exp-Sine-Squared kernel

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

$$k(x, x') = \exp\left[-\frac{2 \cdot \sin^2\left(\pi/p \cdot |x - x'|\right)}{\lambda^2}\right]$$

- The **Exp-Sine-Squared** kernel allows modeling periodic functions.
- It is parameterized by a **length-scale parameter λ** and a **periodicity parameter p**.
- Example: modeling periodic changes, e.g. daily temperature changes, seasonally changing quantities, such as the CO_2 concentration.

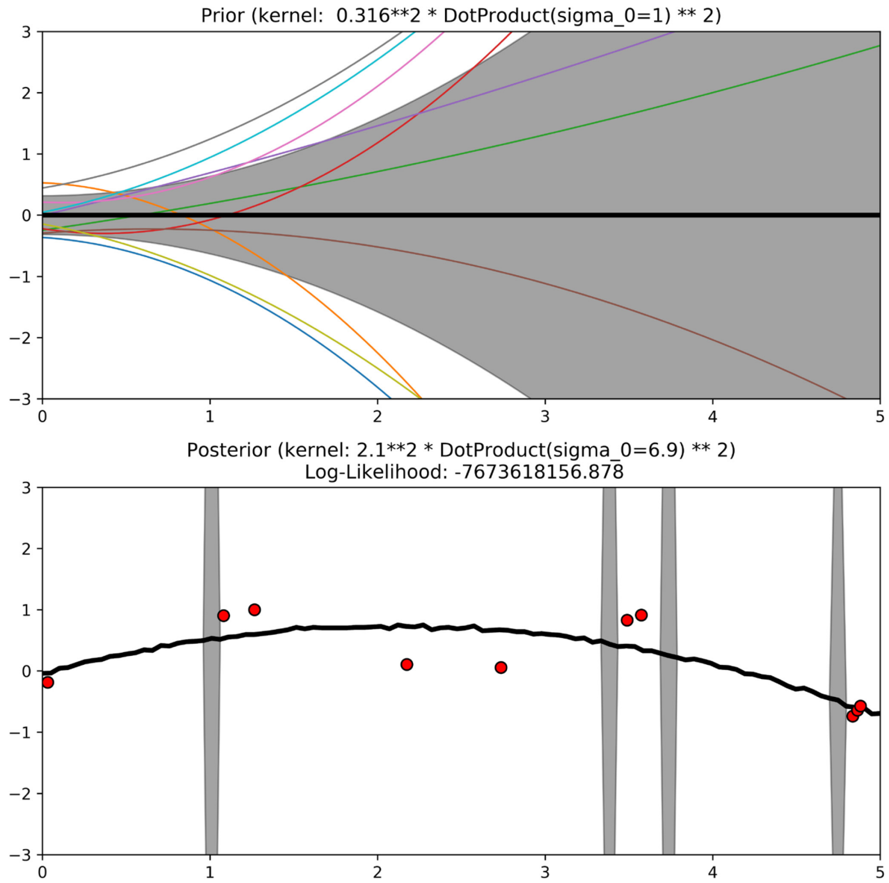


4. Dot-Product kernel

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

$$k(x, x') = \sigma_0^2 + x \cdot x'$$

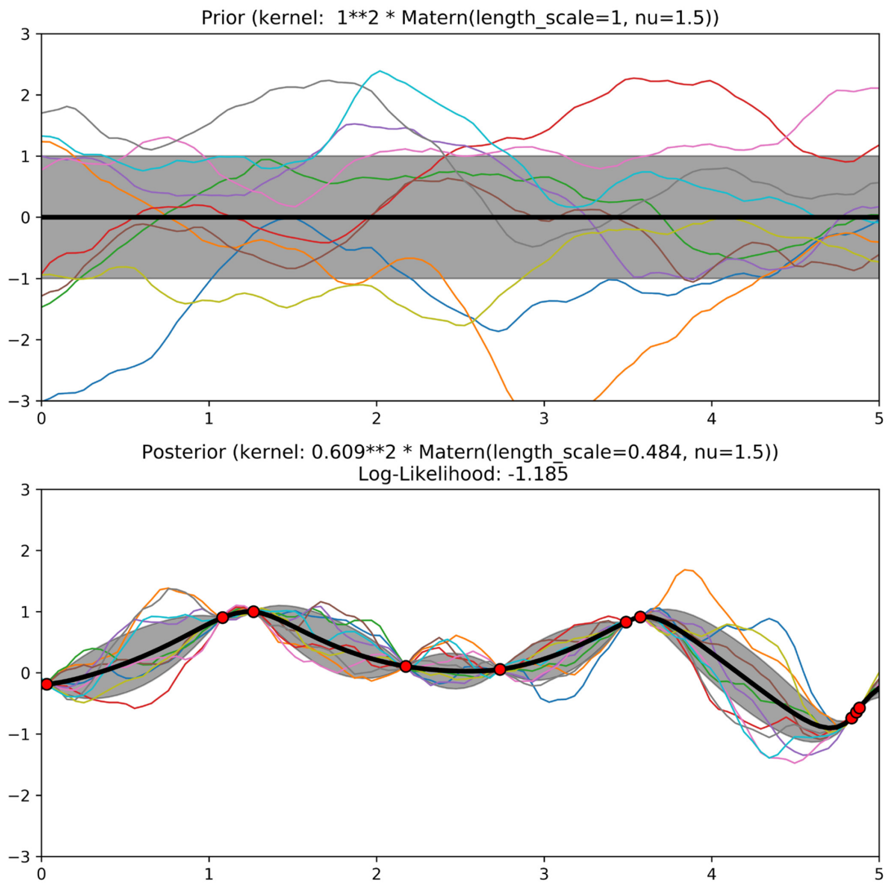
- The Dot-Product kernel is **non-stationary** and can be obtained from linear regression by putting $\mathrm{a_d} \sim \mathcal{N}(0,1)$ priors on the coefficients of $x_d (d = 1 \dots D)$ and a prior of $\mathrm{b_0} \sim \mathcal{N}(0, \sigma_0^2)$ on the bias.
- The Dot-Product kernel is **invariant to a rotation of the coordinates** about the origin, but not translations.
- It is parameterized by a parameter σ_0^2. For $\sigma_0^2 = 0$, the kernel is called the homogeneous linear kernel, otherwise it is inhomogeneous.



5. Matérn kernel

$$k(x_i, x_j) = \sigma^2 \frac{1}{\Gamma(\nu)2^{\nu-1}} \Bigg(\gamma\sqrt{2\nu}\frac{|x-x'|}{\lambda}) \Bigg)^{\nu} \cdot K_{\nu} \Bigg(\gamma\sqrt{2\nu}\frac{|x-x'|}{\lambda}) \Bigg)$$

- The Matern kernel is a **stationary** kernel and **a generalization of the RB**F kernel. It has an additional parameter ν which controls the smoothness of the resulting function.
- It is parameterized by a length-scale parameter λ, which can either be a scalar (isotropic variant of the kernel) or a vector with the same number of dimensions as the inputs (anisotropic variant of the kernel).
- for a finite ν, the matérn kernel generates much rougher sample functions than RBF.
- for the special case of $\nu = \frac{1}{2}$, the kernel gets K(x,x′)=exp(−|x−x′| λ) . This is called a **Ornstein-Uhlenbeck process**, yielding very rough sample functions

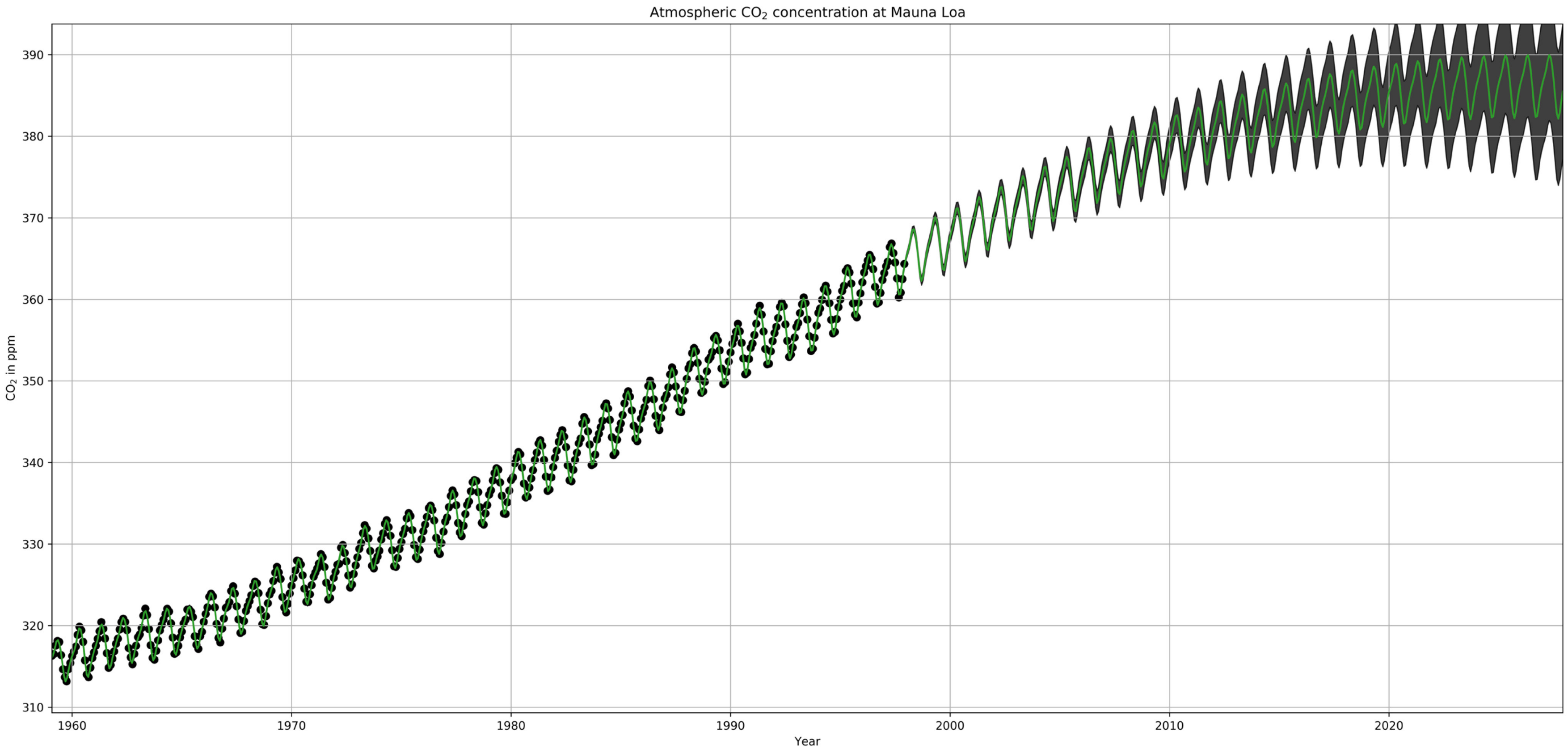


Bayesian inference summary

- *Assumptions* are made explicit in the form of a **prior**
- *Predictions* are make by **averaging over the posterior**, taking all possible interpretations of the data into account
- Bayesian inference does not involve any maximization so there is **no possibility of over-fitting**. Instead, parameters are integrated out.
- Bayesian inference is *usually difficult*, because it is difficult to do integrals.

Example: CO_2 prediction

- CO_2 concentration at Manua Loa



```
gp.kernel_
34.4**2 * RBF(length_scale=41.7) + 3.2**2 * RBF(length_scale=179) *
ExpSineSquared(length_scale=1.41, periodicity=1) + 0.445**2 *
RationalQuadratic(alpha=18.2, length_scale=0.957) + 0.198**2 * RBF(length_scale=0.138) +
WhiteKernel(noise_level=0.0336)
```

Example: CO_2 prediction: Kernel

```
# long term smooth rising trend
k1 = 66.0**2 * RBF(length_scale=67.0)

# seasonal component
k2 = 2.4**2 * RBF(length_scale=90.0) *
    ExpSineSquared(length_scale=1.3, periodicity=1.0)

# medium term irregularity
k3 = 0.66**2 * RationalQuadratic(length_scale=1.2, alpha=0.78)

# noise terms
k4 = 0.18**2 * RBF(length_scale=0.134) +
    WhiteKernel(noise_level=0.19**2)

kernel_gpml = k1 + k2 + k3 + k4
gp = GaussianProcessRegressor(kernel=kernel_gpml, alpha=0,
optimizer=None, normalize_y=True) gp.fit(X, y)
```

Summary

- As Bayesian methods, Gaussian process models **allow one to quantify uncertainty in predictions** resulting not just from intrinsic noise in the problem but also the errors in the parameter estimation procedure. Furthermore, many methods for model selection and hyperparameter selection in Bayesian methods are immediately applicable to Gaussian processes (though we did not address any of these advanced topics here).
- Like locally-weighted linear regression, Gaussian process regression is **non-parametric** and hence can model essentially arbitrary functions of the input points.
- Gaussian process regression models provide a natural way to introduce **kernels into a regression modeling framework**. By careful choice of kernels, Gaussian process regression models can sometimes take advantage of structure in the data.
- Gaussian process regression models, though perhaps somewhat tricky to understand conceptually, nonetheless lead to simple and straightforward linear algebra implementations.