

# Gaussian Processes for Time Series Forecasting

Dr. Juan Orduz

PyData Berlin 2019



# Overview

**Aim:** Give a better intuition on the notion of Gaussian Process Regression

Introduction

Bayesian Linear Regression

The Kernel Trick

Gaussian Process Regression

Kernel Examples

Non-Linear Example (RBF)

The Kernel Space

Example: Time Series

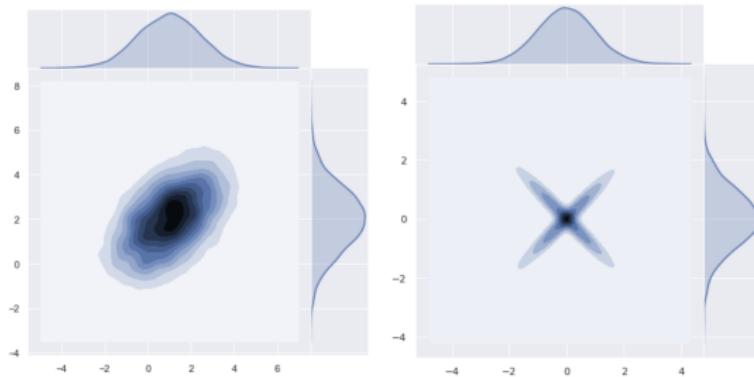
# Multivariate Normal Distribution

[5]

$X = (X_1, \dots, X_d)$  has a **multinormal distribution** if every linear combination is normally distributed. The joint density has the form

$$p(x|m, K_0) = \frac{1}{\sqrt{(2\pi)^d |K_0|}} \exp\left(-\frac{1}{2}(x - m)^T K_0^{-1}(x - m)\right)$$

where  $m \in \mathbb{R}^d$  is the **mean vector** and  $K_0 \in M_d(\mathbb{R})$  is the (symmetric, positive definite) **covariance matrix**.



**Figure:** Left: Multivariate Normal Distribution, Right: Non-Multivariate Normal Distribution

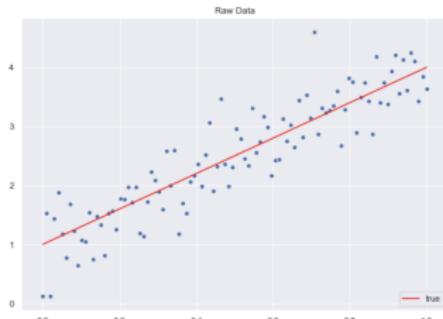
# Bayesian Linear Regression

[2], [7, Chapter 2.1]

Let  $x_1, \dots, x_n \in \mathbb{R}^d$  and  $y_1, \dots, y_n$  be a set of observations (data). We want to fit the linear model

$$f(x) = x^T b \quad \text{and} \quad y = f(x) + \varepsilon, \quad \text{with} \quad \varepsilon \sim N(0, \sigma_n^2)$$

where  $b \in \mathbb{R}^d$  denotes the parameter vector. Let  $X \in M_{d \times n}$  be denote the observation matrix.



**Figure:** The true parameters in this example are  $b = (1, 3)$  and  $\sigma_n = 0.5$

We want to compute  $p(b|X, y)$  using the Bayes theorem

$$p(b|X, y) = \frac{p(y|X, b)p(b)}{p(y|X)} \propto \text{likelihood} \times \text{prior}$$

# Prior Distribution

## ► Prior

$$b \sim N(0, \Sigma_p), \quad \Sigma_p \in M_d(\mathbb{R})$$

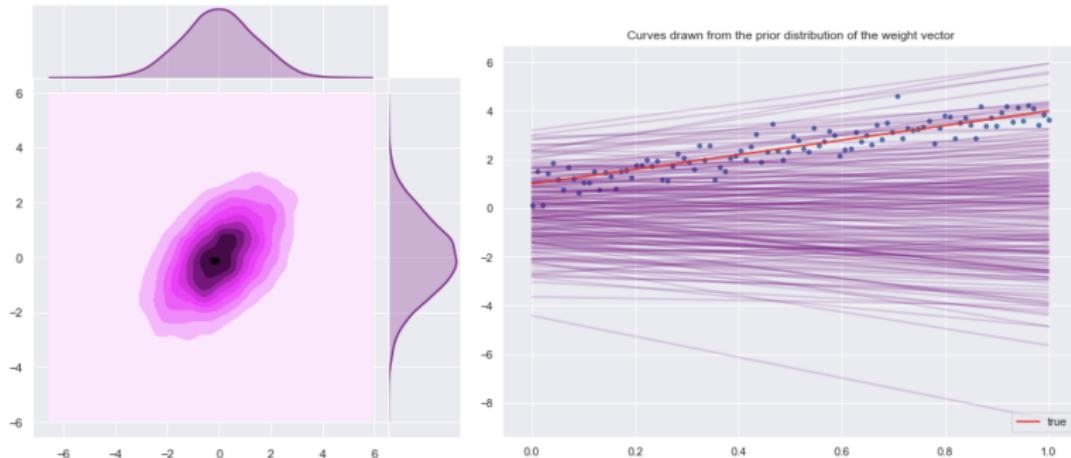


Figure: Prior Distribution, for this example  $\Sigma_p = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ .

## ► Likelihood

$$p(y|X, b) = \prod_{i=1}^n p(y_i|x_i, b) = N(X^T b, \sigma_n^2 I)$$



# Posterior Distribution Sampling

[9]

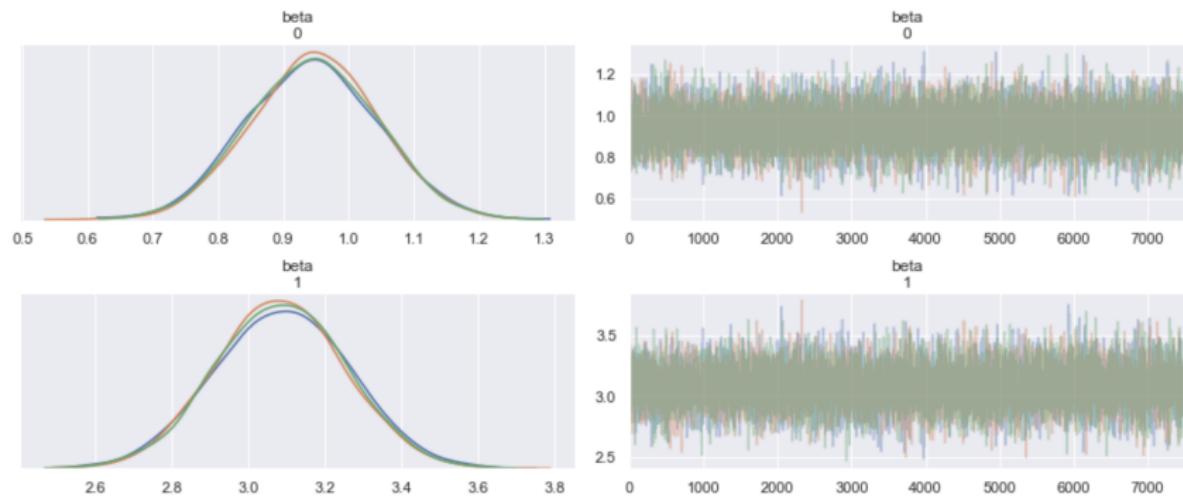


Figure: Posterior distribution of the  $b$  weight estimation using MCMC sampling (PyMC3). Here we use 3 chains.

# Posterior Distribution - Analytical Solution

[7, Chapter 2.1.1]

## ► Posterior

$$p(b|y, X) = N \left( \bar{b} = \frac{1}{\sigma_n^2} A^{-1} X y, A^{-1} \right), \quad A = \sigma_n^{-2} X X^T + \Sigma_p^{-1}$$

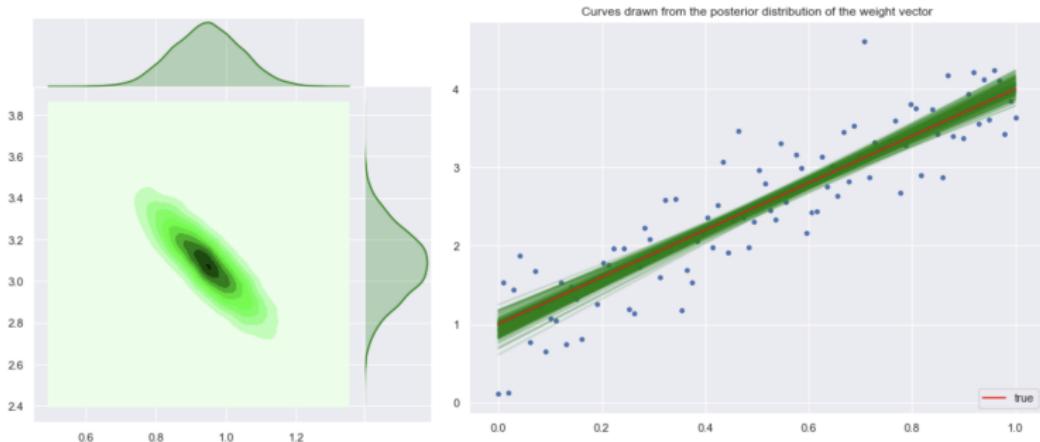


Figure: Posterior Distribution.



# Predictive Distribution - Analytical Solution

[7, Chapter 2.1.1]

$$p(f_*|x_*, X, y) = \int p(f_*|x_*, b)p(b|X, y)db = N\left(\frac{1}{\sigma_n^2}x_*^T A^{-1} X y, x_*^T A^{-1} x_*\right)$$

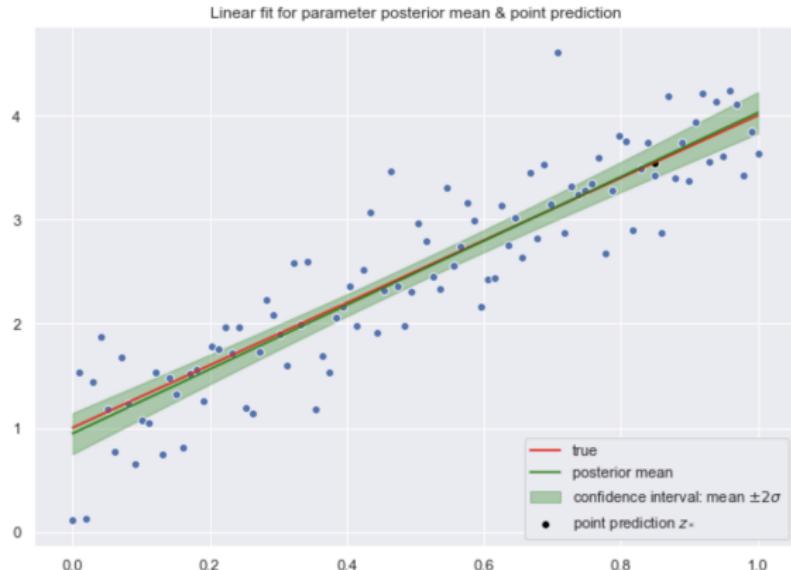


Figure: Prediction Interval.

# The Kernel Trick

[7, Chapter 2.1.2]

Let us consider a map  $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^N$  and the model

$$f(x) = \phi(x)^T b \quad \text{and} \quad y = f(x) + \varepsilon, \quad \text{with} \quad \varepsilon \sim N(0, \sigma_n^2).$$

It is easy to verify that the analysis for this model is analogous to the standard linear model replacing  $X$  with  $\Phi := \phi(X)$ . Set  $\phi_* = \phi(x_*)$ ,

$$p(f_* | x_*, X, y) = N\left(\underbrace{\frac{1}{\sigma_n^2} \phi_*^T \Phi^{-1} \Phi y}_{(1)}, \underbrace{\phi_*^T \Phi^{-1} \phi_*}_{(2)}\right)$$

$$(1) = \phi_*^T \Sigma_p \Phi (\Phi^T \Sigma_p \Phi + \sigma_n^2 I)^{-1} y$$

$$(2) = \phi_*^T \Sigma_p \phi_* - \phi_*^T \Sigma_p \Phi (\Phi^T \Sigma_p \Phi + \sigma_n^2 I)^{-1} \Phi^T \Sigma_p \phi_*$$

This motivates the definition of the **covariance function** or **kernel**

$$k(x, x') := \phi(x)^T \Sigma_p \phi(x')$$

# Gaussian Process

[1, Chapter 21], [7, Chapter 2.2]

## Main Idea

The specification of a covariance function implies a distribution over functions.

## Gaussian Process

- ▶ A **Gaussian Process** is a collection of random variables, any finite number of which have a joint multinormal distribution.
- ▶ A Gaussian process  $f \sim \mathcal{GP}(m, k)$  is completely specified by its mean function  $m(x)$  and covariance function  $k(x, x')$ . Here  $x \in \mathcal{X}$  denotes a point on the index set  $\mathcal{X}$ .

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

# Gaussian Process

[1, Chapter 21], [7, Chapter 2.2]

## Main Idea

The specification of a covariance function implies a distribution over functions.

## Gaussian Process

- ▶ A **Gaussian Process** is a collection of random variables, any finite number of which have a joint multinormal distribution.
- ▶ A Gaussian process  $f \sim \mathcal{GP}(m, k)$  is completely specified by its mean function  $m(x)$  and covariance function  $k(x, x')$ . Here  $x \in \mathcal{X}$  denotes a point on the index set  $\mathcal{X}$ .

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

## Example

The map  $f(x) = \phi(x)^T b$  (with prior  $b \sim N(0, \Sigma_p)$ ) defines a Gaussian process with  $m(x) = 0$  and  $k(x, x') = \phi(x)^T \Sigma_p \phi(x')$ .

## Notation

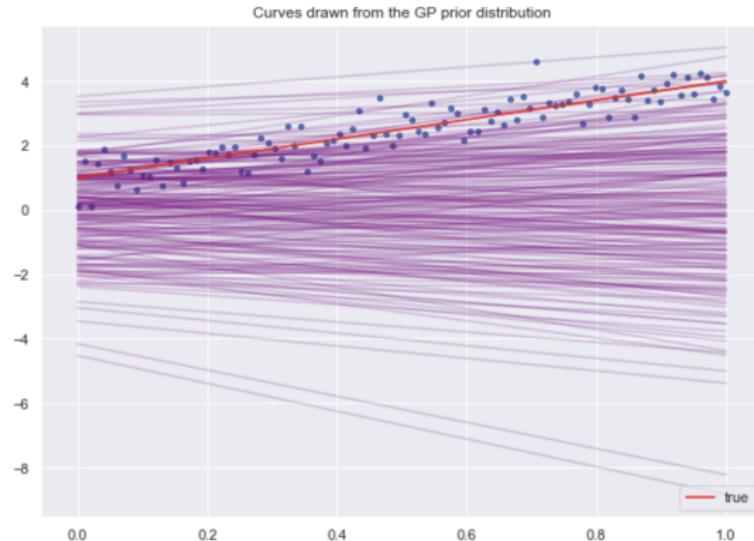
Let  $K(X, X)$  denote the matrix of the point-wise kernel images.

# Linear Regression - Function Space View

[7, Chapter 2.2]

- ▶ Let us consider input points  $X_*$  (test set).
- ▶ Prior

$$f_* \sim N(0, K(X_*, X_*))$$



**Figure:** We sample from the prior space of functions by plotting their realization on  $n_* = 80$  points.

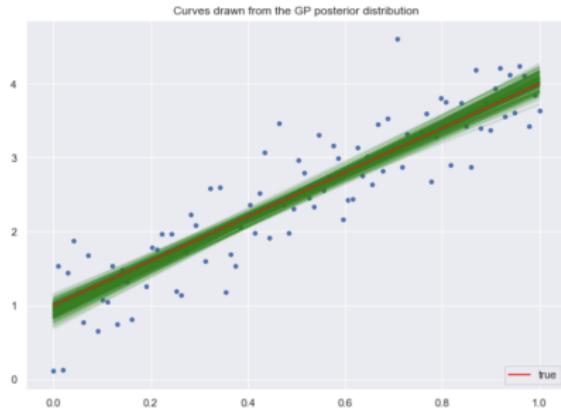
# Linear Regression - Function Space View

[7, Chapter 2.2]

- ▶ Joint Distribution

$$\begin{pmatrix} y \\ f_* \end{pmatrix} \sim N\left(0, \begin{pmatrix} K(X, X) + \sigma_n^2 & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right)$$

- ▶ Conditional Distribution  $f_*|X, y, X_* \sim N(\bar{f}_*, \text{cov}(f_*))$



$$\bar{f}_* = K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}y$$

$$\text{cov}(f_*) = K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}K(X, X_*)$$

# Kernel Examples

[8], [7, Chapter 4.2]

Symmetric and positive semi-definite functions  $k : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ .

- Dot Product

$$k_{DOT}(x, x') = (\sigma_0^2 + x^T \Sigma_p x')^m$$

- Squared Exponential

$$k_{SE}(x, x') = \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$

- Rational Quadratic

$$k_{RQ}(x, x') = \left(1 + \frac{(x - x')^2}{2\alpha\ell^2}\right)^{-\alpha}$$

- Exp-Sine-Squared

$$k_{ESS}(x, x') = \exp\left(-2\left(\frac{\sin(\pi(x - x')/T)}{\ell}\right)^2\right)$$



# Example: Non-Linear Function

[4]

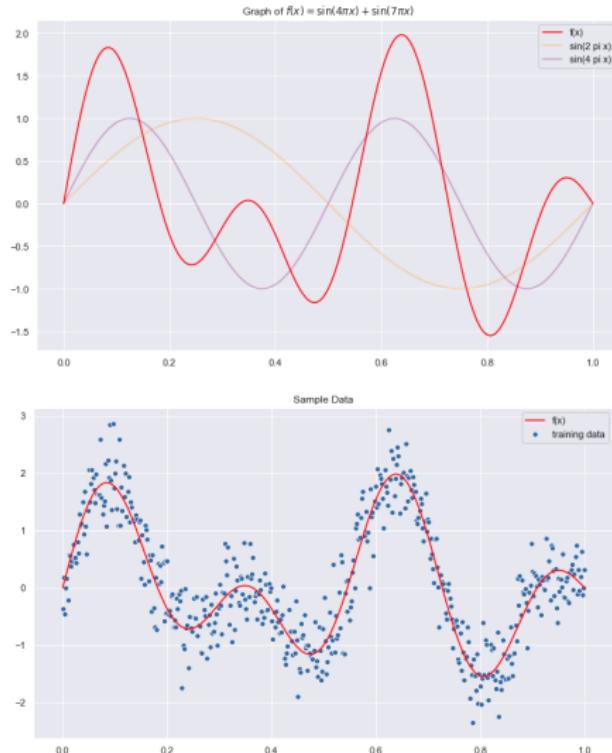


Figure: Non-Linear Example with  $n = 500$  training points.

# Prior Distribution

$$f \sim N(0, K(X_*, X_*))$$

```
fig, ax = plt.subplots()
for i in range(0, 100):
    # Sample from prior distribution.
    z_star = np.random.multivariate_normal(mean=np.zeros(n_star), cov=K_star2)
    # Plot function.
    sns.lineplot(x=x_star, y=z_star, color='blue', alpha=0.2)

# Plot "true" linear fit.
sns.lineplot(x=x, y=f_x, color='red', label='f(x)')
ax.set(title='Samples of Prior Distribution')
ax.legend(loc='lower right');
```

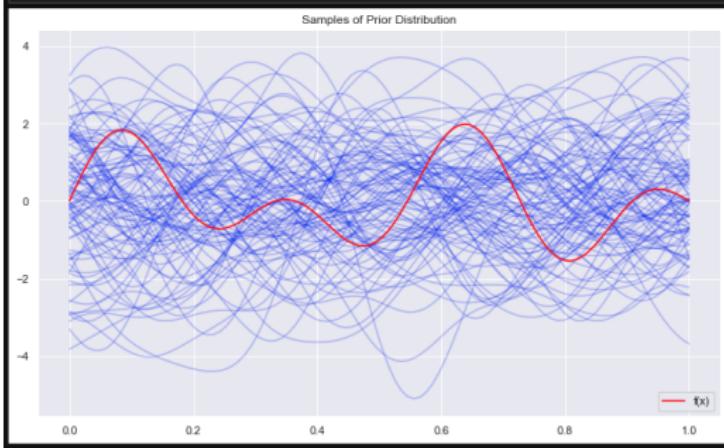
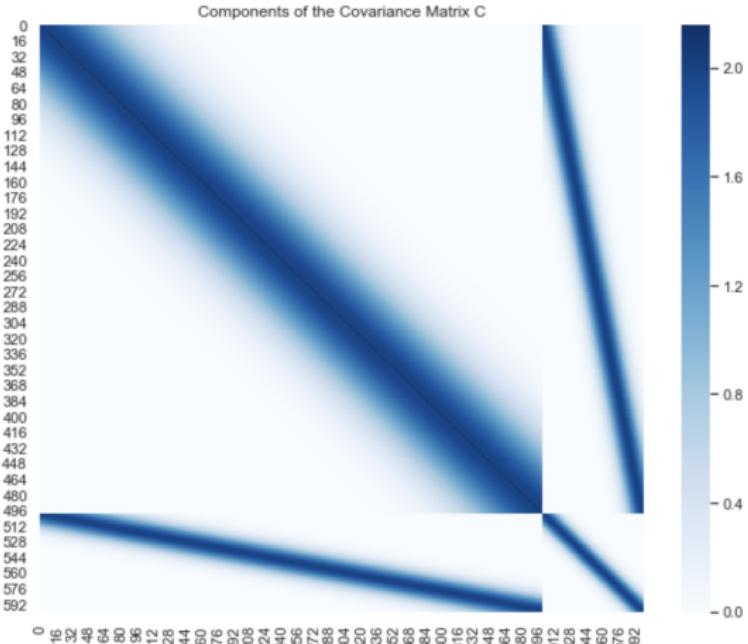


Figure: We sample from the prior space of functions by plotting their realization on  $n_* = 100$  points.

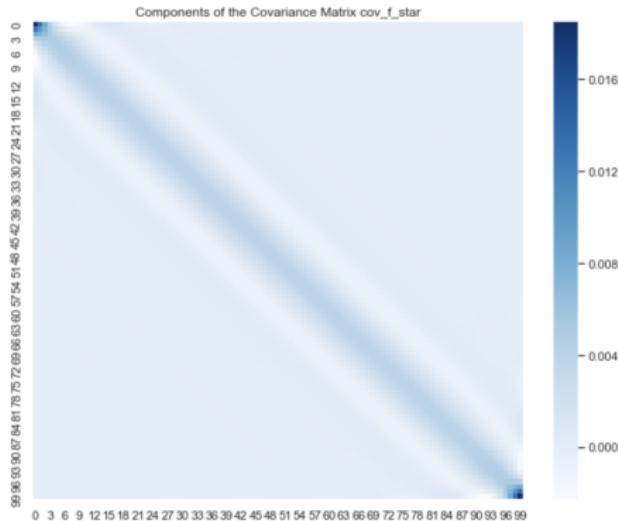
# Joint Distribution

$$\begin{pmatrix} y \\ f_* \end{pmatrix} \sim N \left( 0, \begin{pmatrix} K(X, X) + \sigma_n^2 & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix} \right)$$



# Conditional Distribution

$$f_*|X, y, X_* \sim N(\bar{f}_*, \text{cov}(f_*))$$



**Figure:** Covariance matrix of the posterior (conditional) distribution.

$$\bar{f}_* = K(X_*, X)(K(X, X) + \sigma_n^2 I)y$$

$$\text{cov}(f_*) = K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}K(X, X_*)$$

# Posterior Distribution

```
fig, ax = plt.subplots()

for i in range(0, 100):
    # Sample from posterior distribution.
    z_star = np.random.multivariate_normal(mean=f_bar_star.squeeze(), cov=cov_f_star)
    # Plot function.
    sns.lineplot(x=x_star, y=z_star, color="blue", alpha=0.2);

# Plot "true" linear fit.
sns.lineplot(x=x, y=f_x, color='red', label = 'f(x)')
ax.set(title='Samples of Posterior Distribution, sigma_f = {} and l = {}'.format(sigma_f, l))
ax.legend(loc='upper right');
```

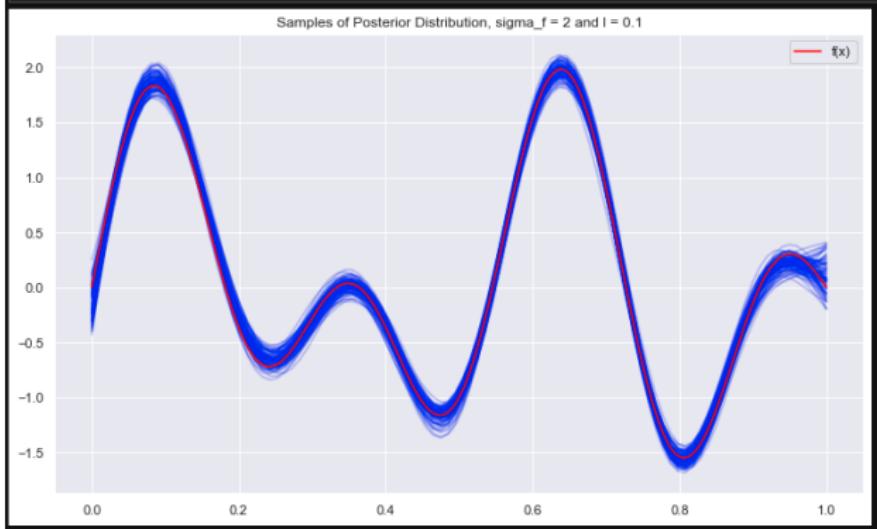


Figure: We sample from the posterior space of functions by plotting their realization on  $n_* = 100$  points.

# Hyperparameter Estimation

[7, Chapter 2.3, 5]

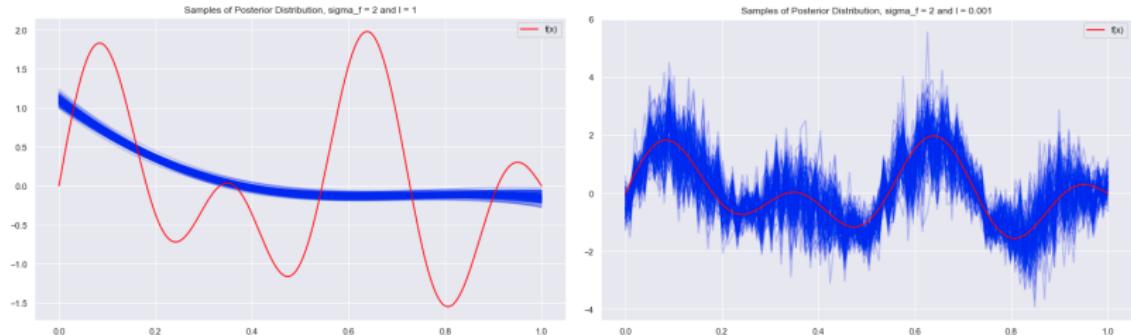


Figure: Samples from two gaussian processes with different scale parameters (left:  $\ell = 1$  and right:  $\ell = 0.001$ ).

## Methods

- Marginal Likelihood ( $\theta$  = parameter vector)

$$\log(p(y|X, \theta)) = -\frac{1}{2}y^T(K + \sigma_n^2 I)^{-1}y - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log(2\pi)$$

- Cross Validation

# The Kernel Space

[8], [7, Chapter 4]

Let  $k_1, k_2 : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  be two kernels, then the following are also kernels

- ▶  $k_1 + k_2$
- ▶  $k_1 \times k_2$
- ▶  $k_1 * k_2$  (convolution)

If  $k : \mathcal{X}_1 \times \mathcal{X}_1 \rightarrow \mathbb{R}$  and  $h : \mathcal{X}_2 \times \mathcal{X}_2 \rightarrow \mathbb{R}$  are two kernels, then the following are also kernels (on  $\mathcal{X}_1 \times \mathcal{X}_2$ )

- ▶  $k_1 \oplus k_2$
- ▶  $k_1 \otimes k_2$

**Remark ([7, Chapter 4.3])**

There is a rich theory of spectral theory for kernels by considering the integral operator  $T_k : L^2(\mathcal{X}, \mu) \rightarrow L^2(\mathcal{X}, \mu)$  (where  $(\mathcal{X}, \mu)$  is a finite measure space and  $k \in L^\infty(\mathcal{X} \times \mathcal{X}, \mu \times \mu)$ ).

$$(T_k\phi)(x) = \int_{\mathcal{X}} k(x, x')\phi(x')d\mu(x')$$

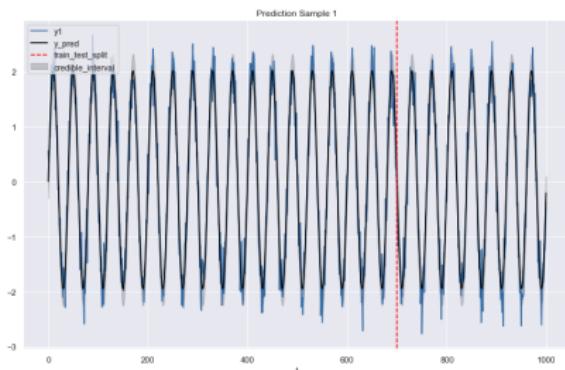
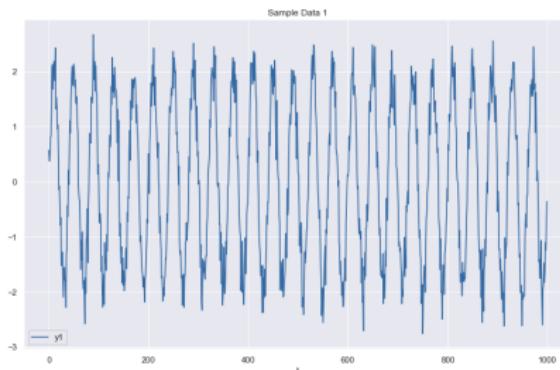


# Example: Periodic Component I ([3])

[6, Section 1.7. Gaussian Processes]

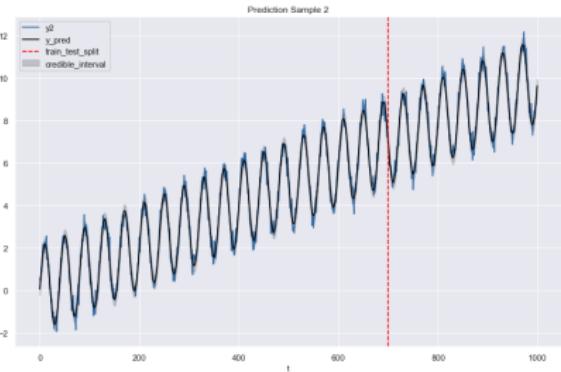
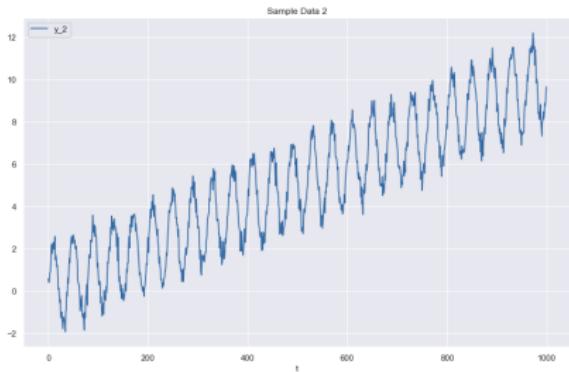
```
from sklearn.gaussian_process.kernels import WhiteKernel, ExpSineSquared, ConstantKernel  
  
k0 = WhiteKernel(noise_level=0.3**2, noise_level_bounds=(0.1**2, 0.5**2))  
  
k1 = ConstantKernel(constant_value=2) * ExpSineSquared(length_scale=1.0, periodicity=40, periodicity_bounds=(35, 45))  
  
kernel_1 = k0 + k1
```

```
from sklearn.gaussian_process import GaussianProcessRegressor  
  
gp1 = GaussianProcessRegressor(  
    kernel=kernel_1,  
    n_restarts_optimizer=10,  
    normalize_y=True,  
    alpha=0.0  
)
```



# Example: Add Linear Trend

```
from sklearn.gaussian_process.kernels import RBF  
  
k0 = WhiteKernel(noise_level=0.3**2, noise_level_bounds=(0.1**2, 0.5**2))  
  
k1 = ConstantKernel(constant_value=2) * ExpSineSquared(length_scale=1.0, periodicity=40, periodicity_bounds=(35, 45))  
  
k2 = ConstantKernel(constant_value=10, constant_value_bounds=(1e-2, 1e3)) * RBF(length_scale=100.0, length_scale_bounds=(1, 1e4))  
  
kernel_2 = k0 + k1 + k2
```



# Example: Add Periodic Component II

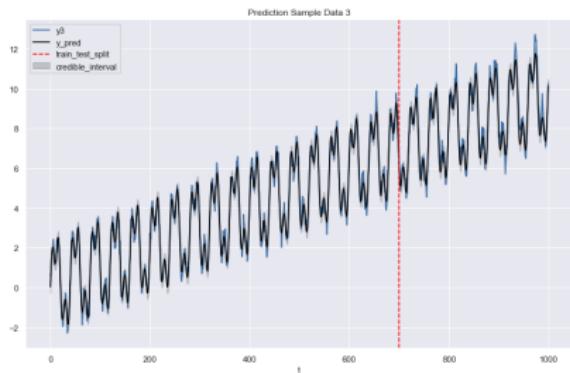
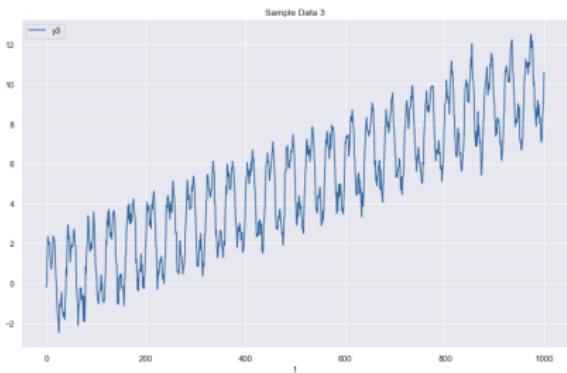
```
k0 = WhiteKernel(noise_level=0.3**2, noise_level_bounds=(0.1**2, 0.5**2))

k1 = ConstantKernel(constant_value=2) * ExpSineSquared(length_scale=1.0, periodicity=40, periodicity_bounds=(35, 45))

k2 = ConstantKernel(constant_value=10, constant_value_bounds=(1e-2, 1e3)) * RBF(length_scale=100.0, length_scale_bounds=(1, 1e4))

k3 = ConstantKernel(constant_value=1) * ExpSineSquared(length_scale=1.0, periodicity=12, periodicity_bounds=(10, 15))

kernel_3 = k0 + k1 + k2 + k3
```



# Computational Challenges

- ▶ Calculating the posterior mean and covariance matrix requires  $\mathcal{O}(n^3)$  computations.
- ▶ A practical implementation of Gaussian process regression is described in [7, Algorithm 2.1], where the Cholesky decomposition is used instead of inverting the matrices directly.
- ▶ There are remarkable approximation methods for Gaussian processes to speed up the computation ([1, Chapter 20.1])

# References I

- [1] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin.  
*Bayesian Data Analysis, Third Edition.*  
Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013.
- [2] Juan Orduz.  
Bayesian regression as a gaussian process.  
[https://juanitorduz.github.io/reg\\_bayesian\\_regression/](https://juanitorduz.github.io/reg_bayesian_regression/), Apr 2019.
- [3] Juan Orduz.  
Gaussian processes for time series forecasting.  
[https://juanitorduz.github.io/gaussian\\_process\\_time\\_series/](https://juanitorduz.github.io/gaussian_process_time_series/), Oct 2019.
- [4] Juan Orduz.  
An introduction to gaussian process regression.  
[https://juanitorduz.github.io/gaussian\\_process\\_reg/](https://juanitorduz.github.io/gaussian_process_reg/), Apr 2019.
- [5] Juan Orduz.  
Sampling from a multivariate normal distribution.  
[https://juanitorduz.github.io/multivariate\\_normal/](https://juanitorduz.github.io/multivariate_normal/), Mar 2019.

## References II

- [6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay.  
Scikit-learn: Machine learning in Python.  
*Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [7] Carl Edward Rasmussen and Christopher K. I. Williams.  
*Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*.  
The MIT Press, 2005.
- [8] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain.  
Gaussian processes for timeseries modelling.  
*Philosophical Transactions of the Royal Society*.
- [9] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck.  
Probabilistic programming in python using PyMC3.  
*PeerJ Computer Science*, 2:e55, apr 2016.



# Thank You!

## Contact

- ▶ [🚀 https://juanitorduz.github.io](https://juanitorduz.github.io)
- ▶ [⌚ github.com/juanitorduz](https://github.com/juanitorduz)
- ▶ [🐦 juanitorduz](https://twitter.com/juanitorduz)
- ▶ [✉️ juanitorduz@gmail.com](mailto:juanitorduz@gmail.com)

