Gaussian Processes as a Time Series Forecasting Tool

Tyler Chan
Eric Zhong
Mario Gutierrez

CSCI 6962/4962
Machine Learning and Optimization
Fall 2023
Project Final Presentation
Undergraduate Group 3

Overview

- Introduction
  - Time series forecasting: Motivation
  - Multivariate Normal Distribution
  - Bayesian Methods and Linear Regression
  - The Kernel Trick
- Gaussian Process Regression
- Gaussian Process Kernels
- Priors and Posteriors for Hyperparameter Estimation
- Time Series Example (Our Implementation and Results)
- Concluding Remarks

Time Series Forecasting: Motivation

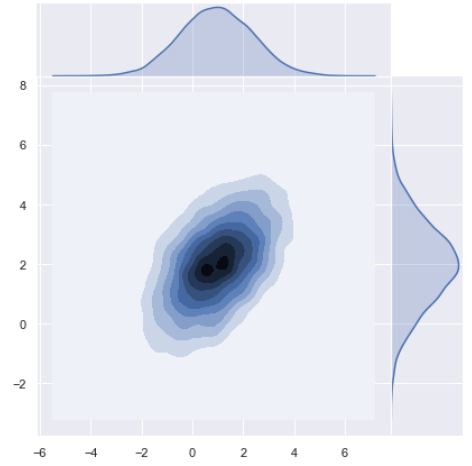
- **Rapid, Precise Forecasting:** Introduce a swift model for accurate forecasts across diverse time series without exhaustive searches.
- **Adaptable Modeling:** Find a fixed objective function and hyperparameters for the model, offering automatic adjustments to diverse time series, surpassing complexities of traditional models (ARIMA and neural networks).
- **Reliable Parameter Estimation:** Ensure stable hyperparameter estimation via statistical methods, vital for accurate forecasts with limited data and minimal computational burden.

Multivariate normal distribution [7]

- Multivariate normal distribution is a generalization of univariate normal distribution to higher dimensions.
- Random vector $\mathrm{x} = (x_1, \ldots, x_k)^T$ has a multinormal distribution if every linear combination is normally distributed.
- Joint density function is:

$$f_{\mathbf{X}}(x_1, \ldots, x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}$$

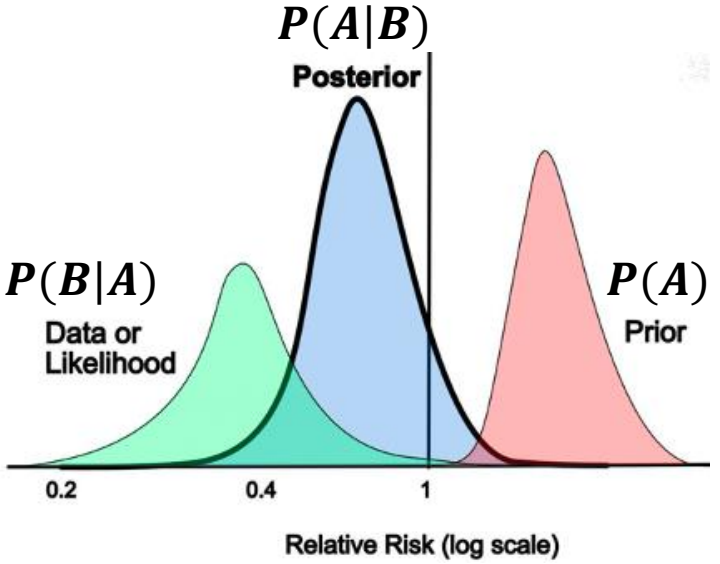
- Where:
  - μ = Mean vector
  - Σ = Covariance metric (symmetric, positive definite)



Normal

Bayesian Methods for ML

- Bayesian methods use Bayes' theorem to compute and update probabilities after obtaining new data.
- Use prior beliefs (prior distribution) and observed data to update and derive a posterior distribution via Bayes' theorem.



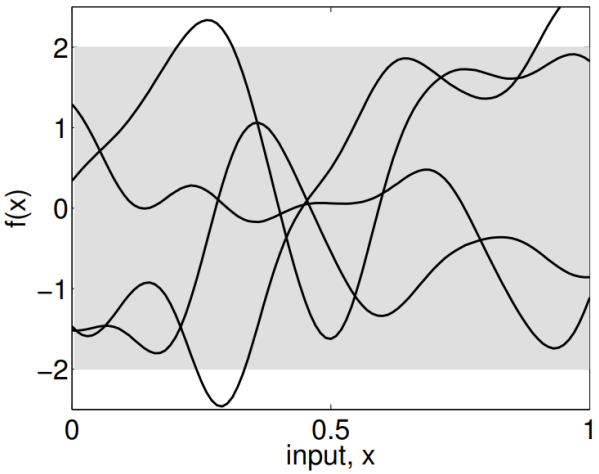
- Bayes' Rule for Bayes Inference

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

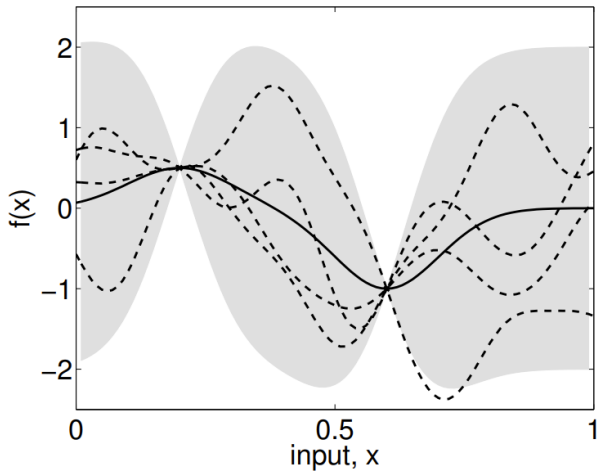
  - P(A) is the prior (Strength in our belief)
  - P(B) is the normalizing factor
  - P(B|A) is the likelihood (probability that data is generated by model with parameter A)
  - P(A|B) is the posterior (Refined strength in our belief)

Gaussian Processes: A specific Bayesian method

- Gaussian Processes are:
  - Completely specified by a mean vector and a covariance matrix.
  - Since Gaussian processes let us describe probability distributions over functions, we can use Bayes' rule to update our distribution of functions by observing training data.
- Graph (a) shows samples drawn from the prior distributions.
- Graph (b) shows the situation after two datapoints have been observed.



(a) Prior



(b) Posterior

Bayesian Regression as a Gaussian Process [7]

- Let $\mathbf{X} = x_1, \ldots, x_n \in \mathbb{R}^d$ and $\boldsymbol{y} = y_1, \ldots, y_n$ be a set of observations.
- We fit a simple Gaussian Process: Bayesian linear regression model:
  - $f(x) = \mathrm{X}^T w$ and $\mathrm{y} = f(x) + \epsilon$ where $\epsilon \sim \mathrm{N}(0, \sigma_n^2)$
- Bayesian linear model is based on the posterior distribution over the weights, computed by Bayes' rule:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \qquad p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)}$$

Where: $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$

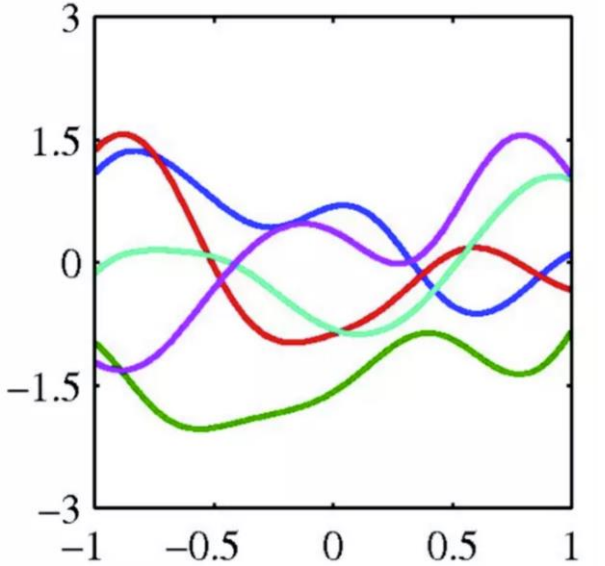
$$p(\mathbf{y}|X, \mathbf{w}) = \mathcal{N}(X^\top \mathbf{w}, \sigma_n^2 I)$$

- Thus, the posterior distributions are also Gaussian

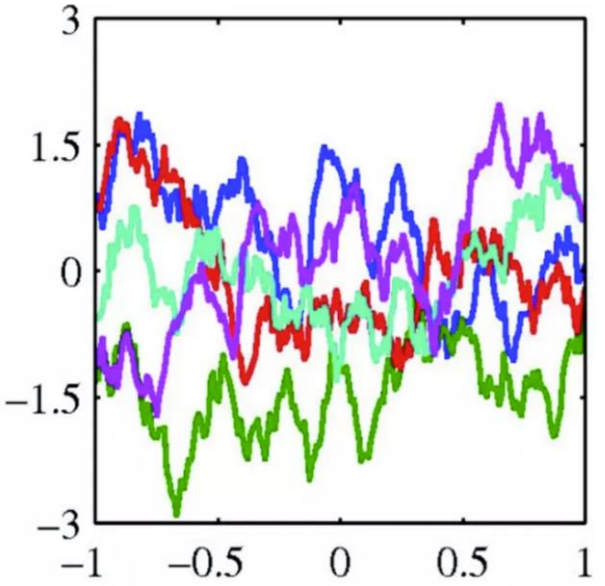
$$p(\mathbf{w}|X, \mathbf{y}) \sim \mathcal{N}(\bar{\mathbf{w}} = \frac{1}{\sigma_n^2} A^{-1} X\mathbf{y}, A^{-1}) \qquad A = \sigma_n^{-2} X X^\top + \Sigma_p^{-1}$$

The Kernel Trick [8]

- The Kernel Trick implicitly maps data into a higher dimensional space without explicitly computing the transformation.
  - $\mathbb{R}^d \xrightarrow{\perp} \mathbb{R}^N \;\; d < N$
- This mapping allows the model to capture complex, non-linear relationships by transforming the input features into a space where relationships are found.



$$k(x_n, x_m) = \exp\left(-\frac{\theta}{2}(x_n - x_m)^2\right)$$



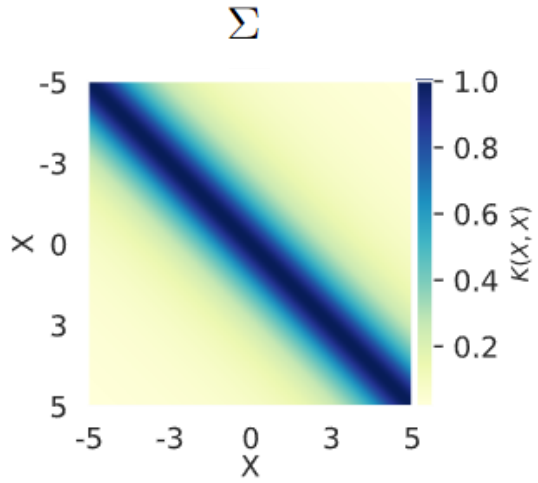
$$k(x_n, x_m) = \exp\left(-\theta|x_n - x_m|\right)$$

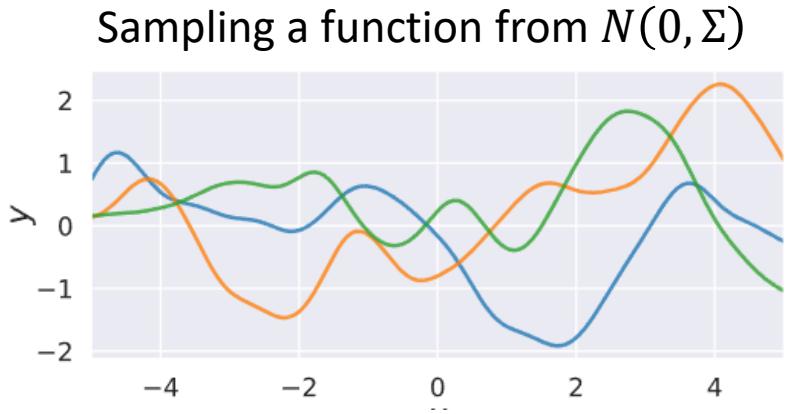
Covariance functions (Kernels) [4]

- In Gaussian Process models the kernel defines the covariance between two function values at different locations
  - The kernel is parameterized by hyperparameters θ

$$Cov\big(f(x_1), f(x_2)\big) = K_\theta(x_1, x_2)$$

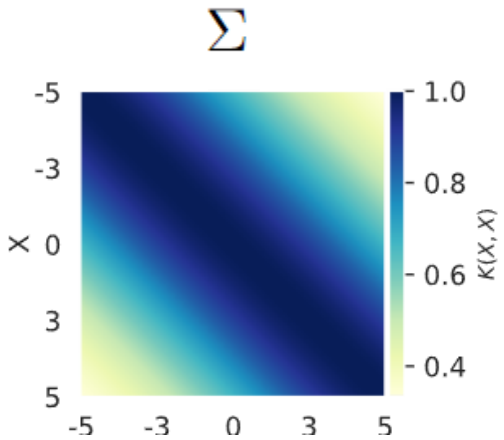
- The heatmap of the covariance matrix below is diagonal, meaning entries close to each other will be similar but entries far away are different.

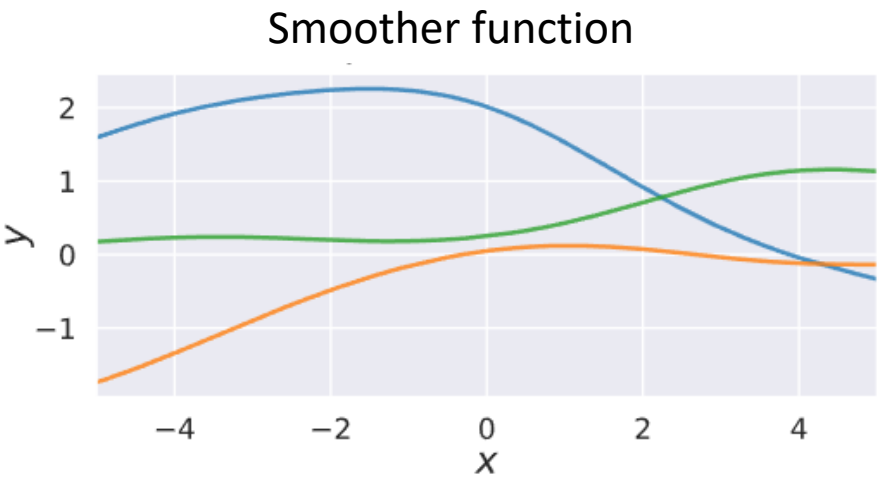




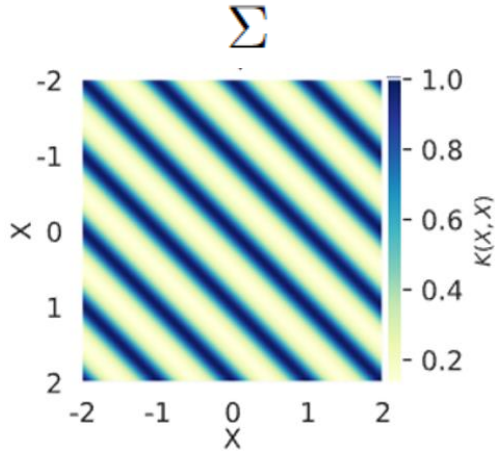
Visualizing Covariance Matrices [3]

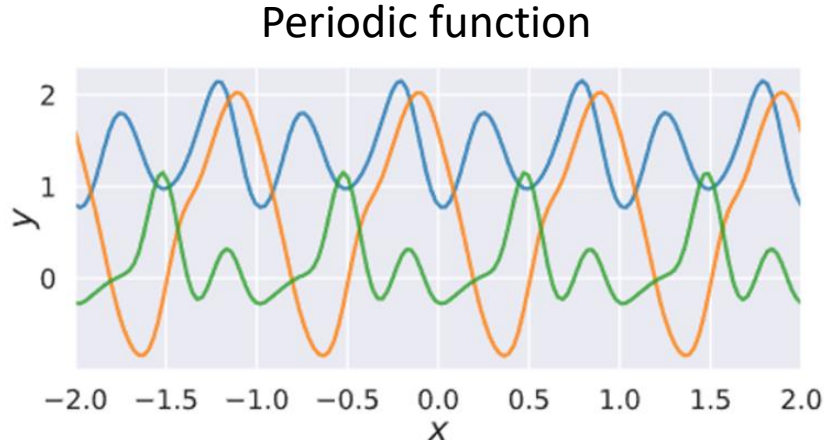
- A smoother function yields entries that far away would have high covariance.





- On periodic kernels, the diagonals in the covariance matrix repeat themselves which correspond to the periods from the function drawings.





Different types of Kernels [3]

- RBF Kernel
  - Produces nonlinear functions
- Periodic Kernel
  - Produces Periodic functions
- Linear Kernel
  - Produces Bayesian linear functions

$$k_{\boldsymbol{\theta}}(x_1, x_2) = s_r^2 \exp\left(-\frac{(x_1 - x_2)^2}{2\ell_r^2}\right)$$

$$k_{\boldsymbol{\theta}}(x_1, x_2) = s_p^2 \exp\left(-\frac{(2\sin^2(\pi|x_1 - x_2|/p_e)}{\ell_p^2}\right)$$

$$k_{\boldsymbol{\theta}}(x_1, x_2) = s_b^2 + s_l^2 x_1 x_2$$

- Kernels define the general structure of the function, but can be fine-tuned with hyperparameters, which are denoted as theta. For example, in the RBF kernel, there are 2 hyperparameters s and l. s determines the average distance from the mean while l determines the smoothness of the function.

Kernels continued [2][3]

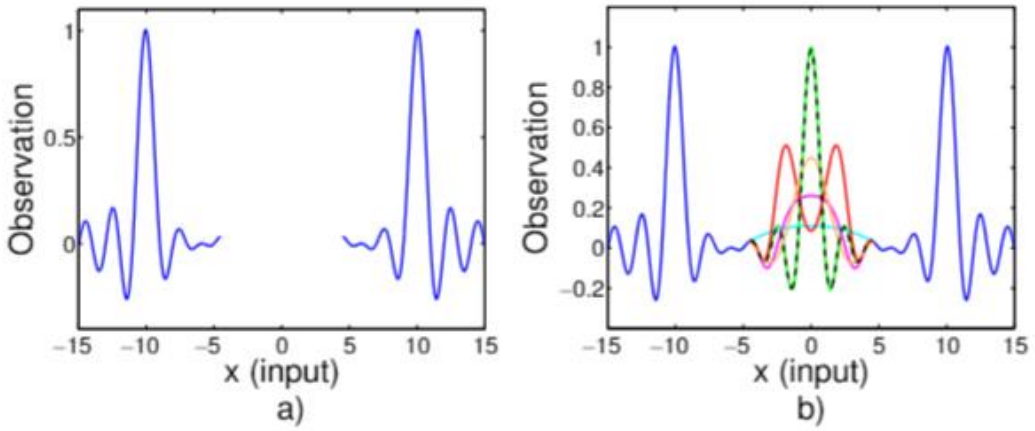
- White noise Kernel

$$k_{\boldsymbol{\theta}}(x_1, x_2) = \text{noise\_level if } x_1 == x_2 \text{ else } 0$$

  - Accounts for noise in the input by adding random noise to the function
- Spectral Mixture Kernel

$$k_{\boldsymbol{\theta}}(x_1, x_2) = s_{m_i}^2 \exp\left(-\frac{(x_1 - x_2)^2}{2\ell_{m_i}^2}\right) \cos\left(\frac{x_1 - x_2}{\tau_{m_i}}\right)$$

  - RBF Kernel * cos Kernel
  - As seen by the plots below, compared to other kernels, the SM kernel can fit a complex function like the sinc function.



$$y(x) = \text{sinc}(x + 10) + \text{sinc}(x) + \text{sinc}(x - 10)$$
$$\text{sinc}(x) = \sin(\pi x)/(\pi x)$$

Kernel composition [1]

- Kernels are closed under addition and multiplication
  - To add and multiply kernels, we do elementwise addition and multiplication on the two corresponding matrices
- In our problem, we use the following Kernel composition:

$$K = PER + LIN + RBF + WN + SM_1 + SM_2$$

- The paper Gaussian Processes needs priors [1] claims that this is a general kernel since it captures the useful components for modelling timeseries and thus we can keep this composition fixed for time series, eliminating the need for kernel search
- Some time series may exhibit certain trends such as linearity or periodicity, yet this kernel is suitable since it is able to scale hyperparameters based on their importance. This is called Automatic Relevance Determination (ARD)

Drawing Posterior Distributions [7]

- We can write the joint distribution of the observed target values ($\mathbf{f}$) and the function values ($\mathbf{f}_*$) under the prior as:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

Gaussian Process follows a multivariate normal distribution.
Thus, the mean vector is defined at **0**.
This makes the model flexible during learning.

- Thus, the key predictive equation is:

$$\mathbf{f}_*|X_*,X,\mathbf{f} \sim \mathcal{N}\big(K(X_*,X)K(X,X)^{-1}\mathbf{f}, K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)\big)$$

- Function values $\mathbf{f}_*$ (corresponding to test inputs X_*) are sampled from the joint posterior distribution:
  - Evaluate mean and covariance matrix.
  - Generating samples using the mean and diagonal of the covariance matrix.

Practical Algorithm for GP Regression [7]

Given:

- Training Points (X)
- Training Values (y)
- Testing Points (X^*)
- Kernel (K)

Returns:

- $\bar{f}_*$ (mean)
- $\mathbb{V}[f_*]$ (variance)
- $\log p(\mathrm{y} \mid \mathrm{X})$ (log marginal likelihood)

1. $\mathrm{L} \coloneqq \mathrm{cholesky}\big(K(\mathrm{X}, \mathrm{X})\big)$
2. $\boldsymbol{\alpha} \coloneqq \mathrm{L}^T \backslash (\mathrm{L} \backslash \mathrm{y})$
3. $\bar{f}_* \coloneqq \mathrm{K}(X^*, X)\boldsymbol{\alpha}$
4. $\mathbf{v} \coloneqq \mathrm{L} \backslash K(\mathrm{X}, \mathrm{X}^*)$
5. $\mathbb{V}[f_*] \coloneqq K(\mathrm{X}^*, \mathrm{X}^*) - \mathbf{v}^T\mathbf{v}$
6. $\log p(\mathrm{y} \mid \mathrm{X}) \coloneqq -\frac{1}{2}\mathrm{y}^T\boldsymbol{\alpha} - \sum_i \log\, \mathrm{L}_{ii} - \frac{n}{2}\log 2\pi$

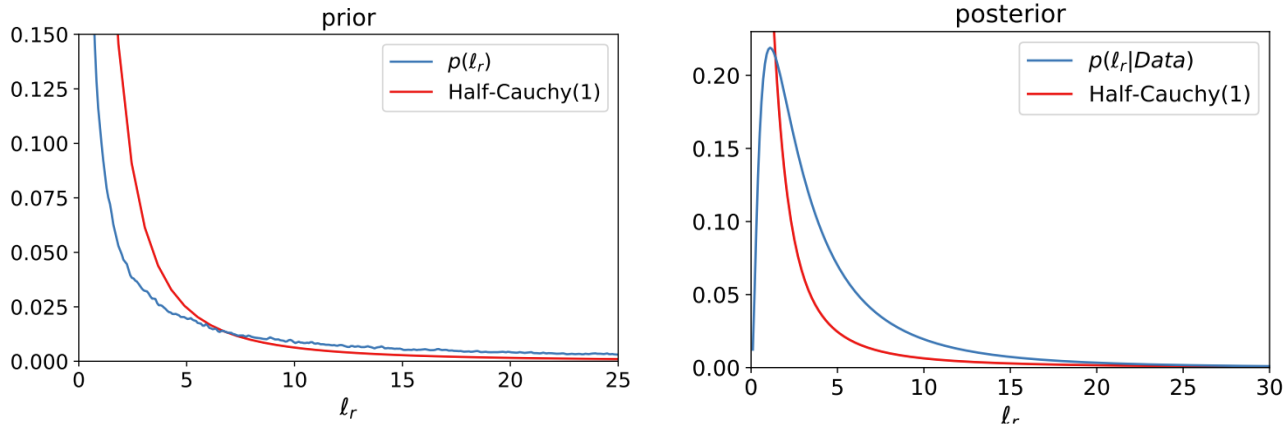
- Computational complexity is $O(n^3/6)$ due to the Cholesky decomposition
- Memory complexity is $O(n^2)$
- Where n is the number of training points

Example How Priors Are Used [1][3]

- RBF Kernel has two hyperparameters:
  - s_r and ℓ_r

$$k_{\boldsymbol{\theta}}(x_1, x_2) = s_r^2 \exp\left(-\frac{(x_1 - x_2)^2}{2\ell_r^2}\right)$$

- These hyperparameters are unknown and can be modeled by some distribution:
  - $s_r \sim \mathcal{N}(-2, 0.1)$
  - $\ell_r \sim \mathcal{N}(0, 1)$
- Through training, the mean and variance for these hyperparameters can be narrowed down to values that better fit the data.



Properties and Guarantees [1]

- Table below shows the performance of different time-series machine learning models:
  - GP (with priors)
  - Exponential Smoothing (ets)
  - arima
  - GP_0 (without priors)
- Research has shown that GP models require priors on hyperparameters to produce highly accurate forecasts.
- Why use GP with priors?
  - It is mostly more accurate than the traditional methods.
  - It is faster to compute than the other methods.
  - It avoids having to do kernel searches.

competition	freq	score	GP	*ets*	*arima*	GP_0
M1	monthly	MAE	**0.58**	0.59	0.62*	0.72*
M1	monthly	CRPS	**0.41**	0.45*	0.45*	0.53*
M1	monthly	LL	**-1.13**	-1.27*	-1.28*	-1.67*
M1	quarterly	MAE	**0.57**	0.63*	0.62*	0.75*
M1	quarterly	CRPS	**0.39**	0.47*	0.44*	0.59*
M1	quarterly	LL	**-1.07**	-1.41*	-1.44*	-2.66*
M3	monthly	MAE	**0.48**	0.51*	0.51*	0.59*
M3	monthly	CRPS	**0.35**	0.38*	0.37*	0.42*
M3	monthly	LL	**-1.01**	-1.05*	-1.06*	-1.23*
M3	quarterly	MAE	0.42	**0.41**	**0.41**	0.54*
M3	quarterly	CRPS	**0.30**	0.31	0.31	0.40*
M3	quarterly	LL	**-0.85**	-0.90*	-0.94*	-1.61*

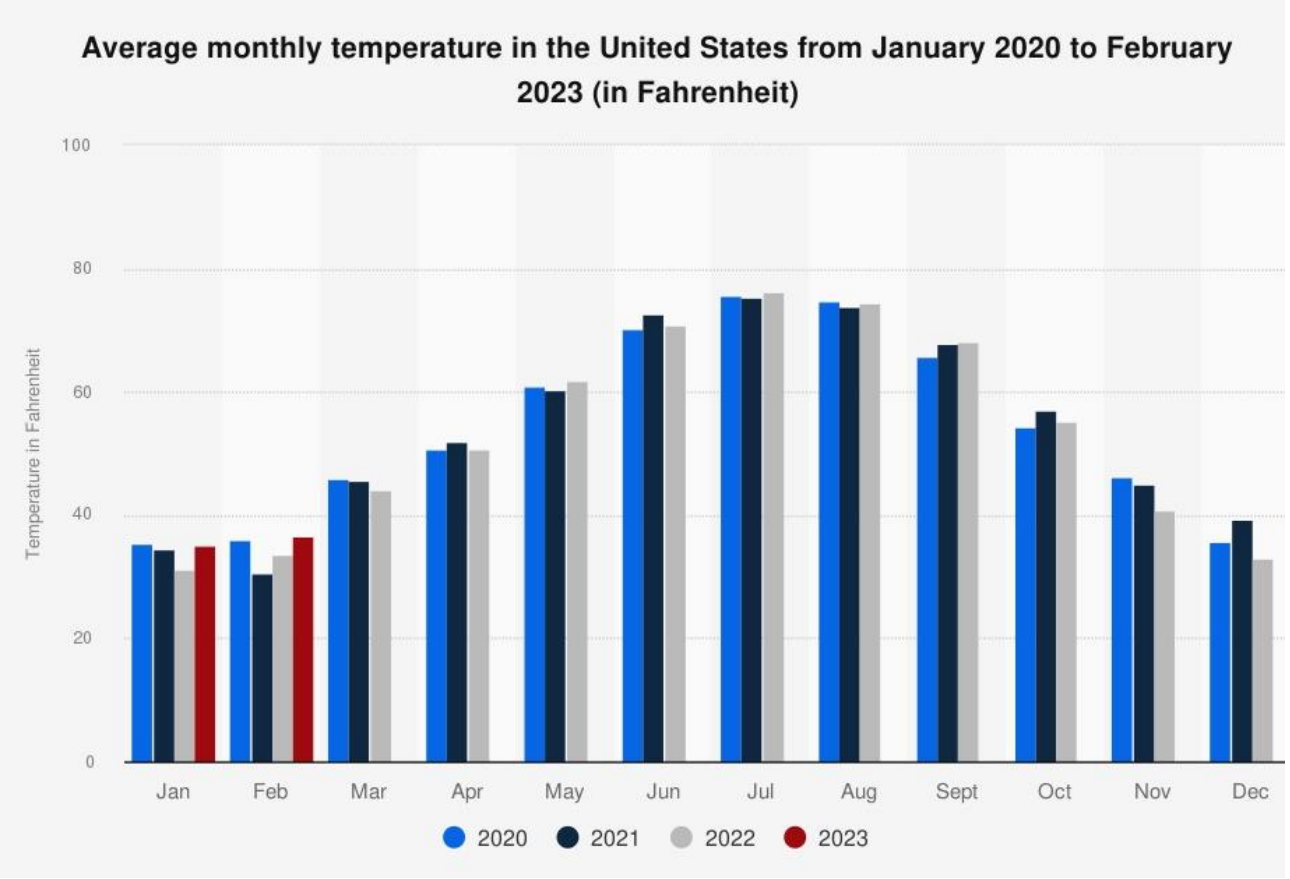
- M1 and M3 = Monthly time series competition datasets.
- **Higher scores for LL and lower scores for MAE and CRPS are better**

Time Series Forecasting: Revisiting Motivation

- **Rapid, Precise Forecasting**: Introduce a swift “Gaussian Process model” for accurate forecasts across diverse time series without exhaustive searches.
- **Adaptable Modeling**: Find a fixed kernel composition in the GP model, offering automatic adjustments to diverse time series, surpassing complexities of ARIMA and neural networks.
- **Reliable Parameter Estimation**: Ensure stable hyperparameter estimation via Bayesian methods, vital for accurate forecasts with limited data and minimal computational burden.

Timeseries forecasting on temperature dataset

- Our dataset consisted of average monthly observations of US temperature(in Fahrenheit) between 1895 and 2023 (Dataset)
- We used 70% of the data for the training set and the other 30% for our test set
  - X_train: (1895 – 1985)
  - X_test: (1985 – 2023)
- Highly periodic. Period = 1 year.



Our Implementation: [1]

- We used the Sklearn Gaussian Process Regressor (GPR) to learn our data.
- We had to define which kernels to use and the bounds for the hyperparameters.
  - As mentioned previously, we used this composition of kernels:

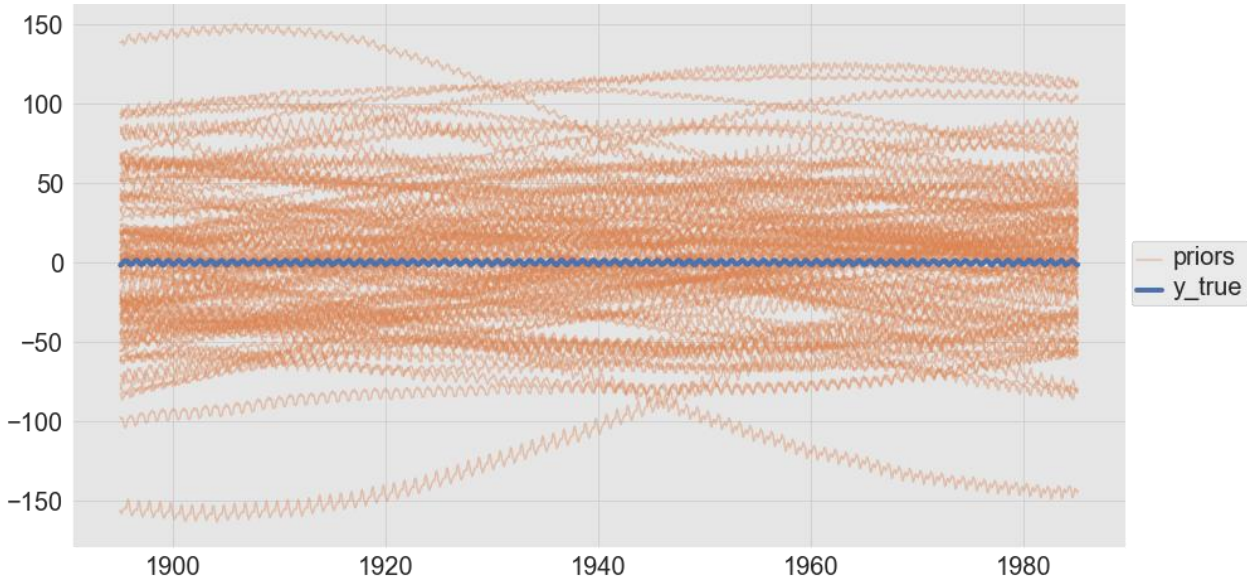
$$K = PER + LIN + RBF + WN + SM_1 + SM_2$$

- We scaled the data with 0 mean and 1 variance.
- We then fit the priors by using the fit method from the GPR class

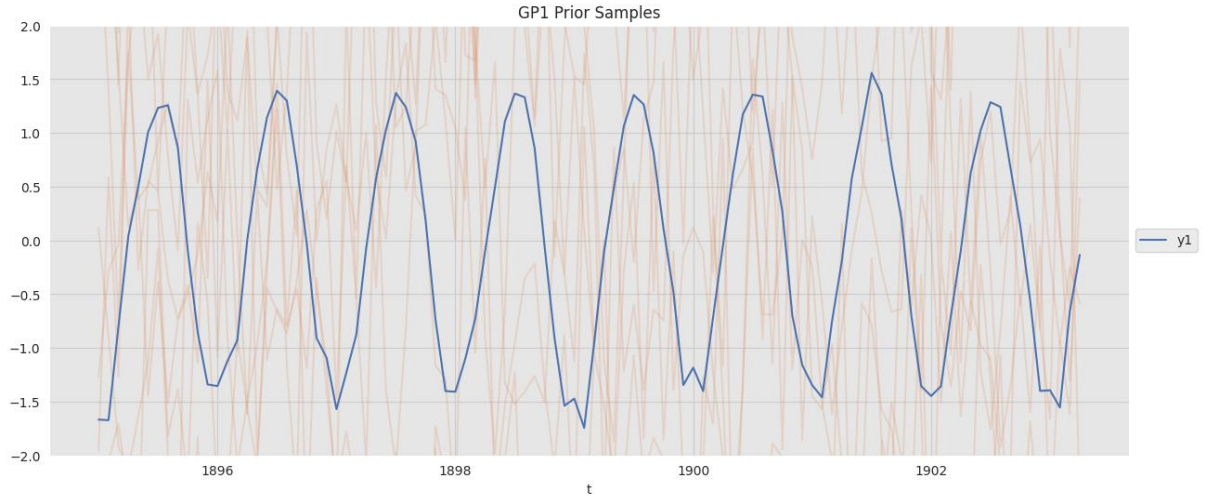
Prior Samples

- To the right is 100 samples from our Gaussian process prior space of functions.
- Sampling the priors allow us to determine bounds on the hyperparameters to reduce the search space
- The variance of the samples is also large

Training Set and GP Functions from the Priors



Zoomed in Version of Training Set and Prior Samples



Training and Posterior Results

- We would like to compute the posterior but some integrals we cannot compute directly (intractable).
- Maximizing the marginal likelihood with respect to the hyperparameters (often denoted as θ) is a common approach to approximate the posterior distribution over the hyperparameters.

$$p(y|X,\theta) = \int p(y|f,X,\theta)p(f|X)df$$

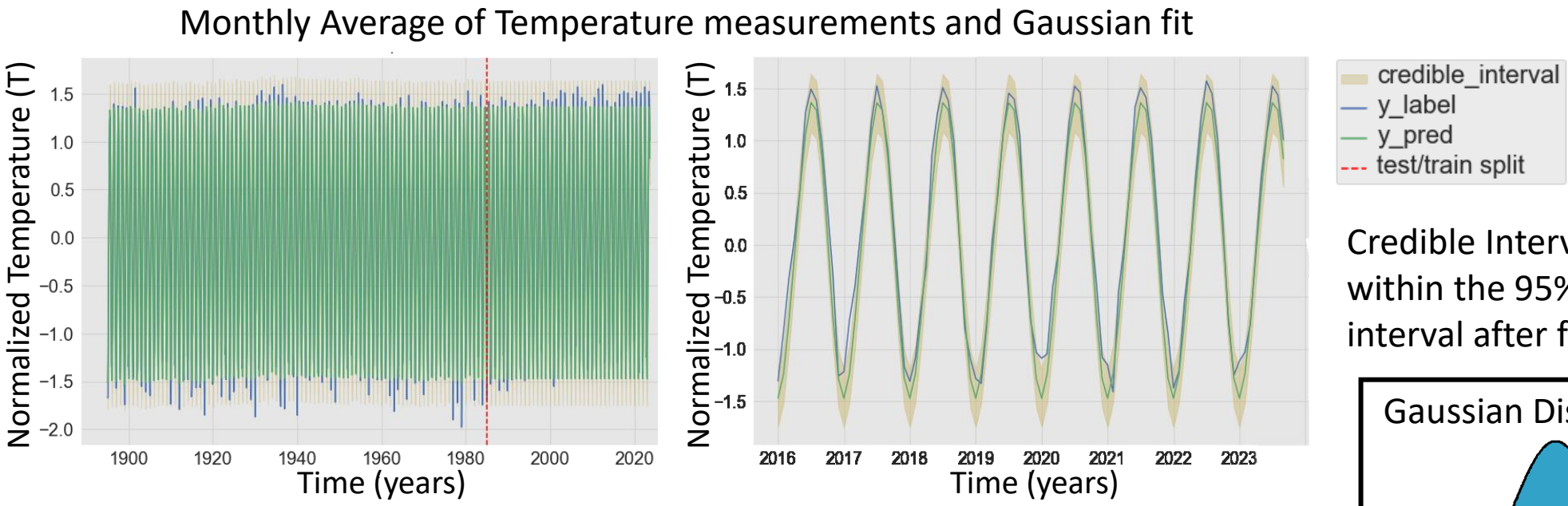
- We can explicitly compute the log marginal likelihood:

$$\log p(\mathbf{y}|\theta,X) = \underbrace{-\frac{1}{2}\mathbf{y}^\top[K_\theta(X,X)+\sigma^2 I]^{-1}\mathbf{y}}_{\text{Fitting the Data}} \underbrace{-\frac{1}{2}\log|K_\theta(X,X)|}_{\text{Complexity Penalty}} + \underbrace{c}_{\text{Normalization Constant}}$$

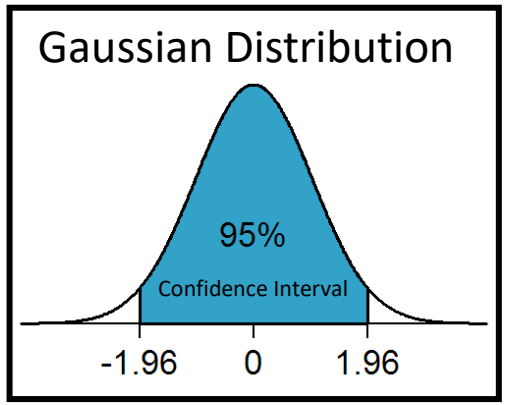
- The above expression encodes a trade-off within fitting the data and penalizing for model complexity

Training and Posterior Results

- In Sklearn, the optimizer minimized the negative log marginal likelihood with respect to theta using gradient descent.
- Sklearn uses the L-BFGS-B algorithm by default, a quasi-Newton method.

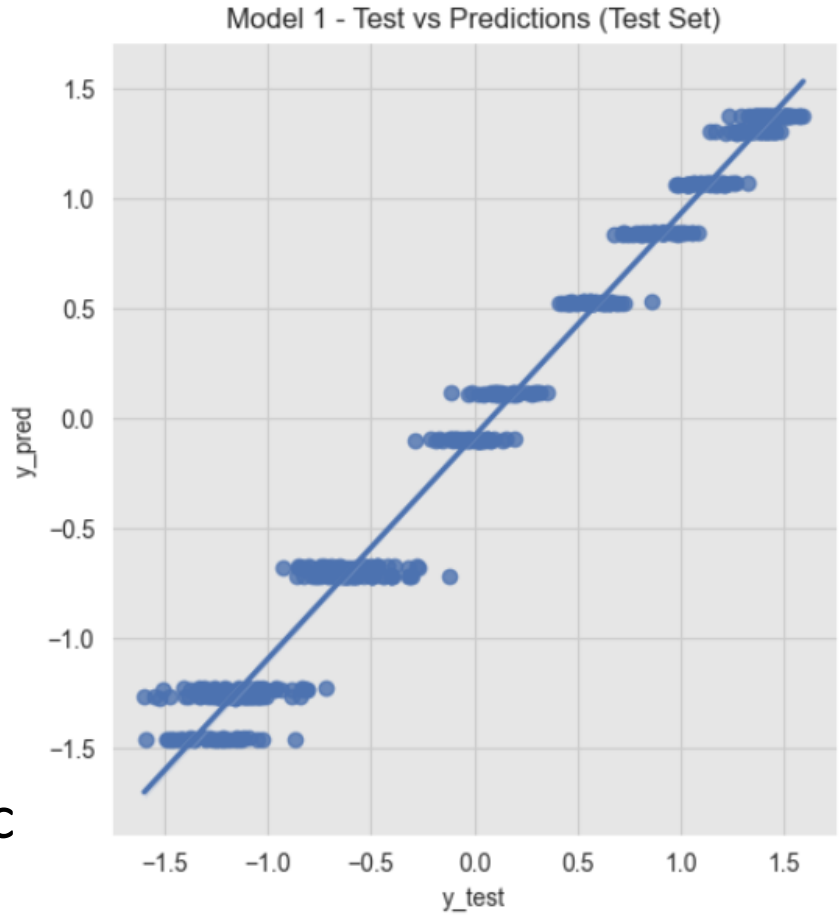


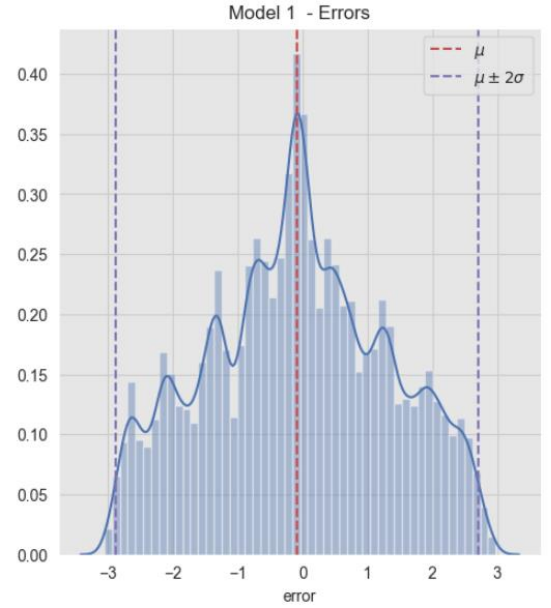
Credible Interval: Functions within the 95% confidence interval after fit



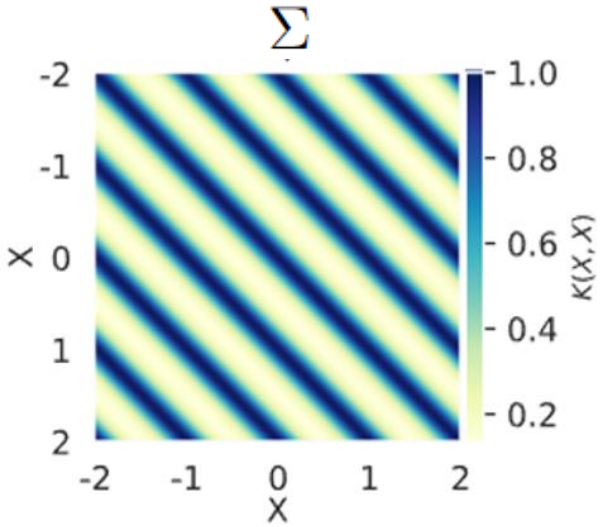
Results continued

- Performance:
  - R2 Score Train = 0.982
  - R2 Score Test = 0.974
  - MAE Train = 0.101 & MAE Test = 0.121
- Model predominantly learnt about the data's periodicity.
- Thus, the model has discrete values for each time step which on average are correct and deviate as a normal distribution.
- The discrete values in the predictions remind us of periodic kernel's heatmap.





Periodic Kernel's Correlation Matrix



Gaussian processes offer a powerful framework for time series forecasting, providing a nuanced understanding of uncertainty and adaptability crucial in dynamic environments.

Thank you!

Questions?

Sources

[1] Giorgio Corani, Alessio Benavoli, Marco Zaffalon, "Time series forecasting with Gaussian Processes needs priors", arXiv:2009.08102 [stat.ML], 2020. Available: https://arxiv.org/abs/2009.08102

[2] Andrew Gordon Wilson, Ryan Prescott Adams, "Gaussian Process Kernels for Pattern Discovery and Extrapolation", arXiv:1302.4245 [stat.ML], 2013. Available: https://arxiv.org/abs/1302.4245.

[3] Peter Roelants, “Gaussian processes (3/3) - exploring kernels,” Github Pages, January 7, 2019. Available: https://peterroelants.github.io/posts/gaussian-process-kernels/

Sources continued

[4] Mutual Information, “Gaussian Processes,” Youtube, Aug 22, 2021. [Video File]. Available: https://www.youtube.com/watch?v=UBDgSHPxVME. Accessed November 10, 2023.

[5] Juan Camilo Orduz, “PyData Berlin 2019: Gaussian Processes for Time Series Forecasting (scikit-learn),” Github Pages, October 10, 2019. Available: https://juanitorduz.github.io/gaussian_process_time_series/

[6] PyMC Labs, "L3: Hierarchical Modeling (State of Bayes Lecture Series),“ Youtube, May 4, 2023. [Video File]. Available: https://www.youtube.com/watch?v=pnJgDSdgqVg. Accessed: November 10, 2023.

Sources continued

[7] C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. c 2006 Massachusetts Institute of Technology. www.GaussianProcess.org/gpml

[8] Brochu, E., Cora, V.M. and de Freitas, N. (2010) A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning.

[9] https://compass.blogs.bristol.ac.uk/2022/01/25/gaussian-process-emulation/

Supplementary Slides

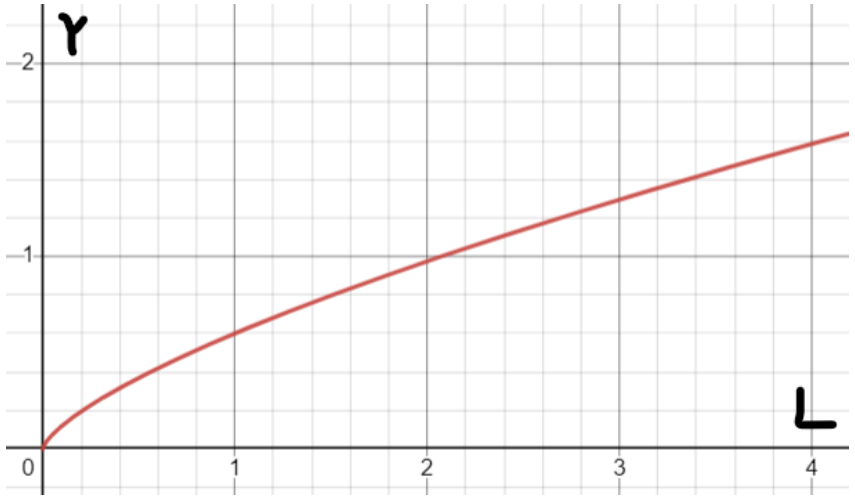
For the Go-Getters ☺

Priors: Toy Example [6]

- We want to model a simplified Cobb-Douglas function in this form:

$$Y \approx A \cdot L^{\beta}$$

- Where:
  - Y is the output
  - L is the input labor
  - A is the total production factor
  - β is the discount labor



Example Graph: $A = 0.6, \beta = 0.7$

Priors: Toy Example [6]

- We should do a log-log model instead:

$$\log Y \approx \log A + \beta \log L$$

- The prior predictive now can be modeled by a Gaussian:

$$\log Y \approx \text{Normal}(\log A + \beta \log L\,, \epsilon)$$

Where:

- ϵ is the variance from the expected model

Priors: Toy Example [6]

$$\log Y \approx \text{Normal}(\log A + \beta \log L\,, \epsilon)$$

Now we have 3 parameters that we don't know:

- $A = ??$
- $\beta = ??$
- $\epsilon = ??$

Priors: Toy Example [6]

$$\log Y \approx \text{Normal}(\log A + \beta \log L\,, \epsilon)$$

They can be represented by distributions whose parameters are selected from domain expertise:

- $A = \text{LogNormal}(-0.5,1)$
- $\beta = \text{LogitNormal}(0,1)$
- $\epsilon = \text{LogNormal}(-2,0.1)$

Hierarchies [6]

- Let's say there is a population and k groups within that population we want to model.
- Different groups would have different Total Production Factors:
  - A_k : Total Production Factor for group k
  - A_{pop} : Total Production Factor for entire population

Hierarchies [6]

$$\log Y \approx \text{Normal}(\log A_k + \beta \log L\,, \epsilon)$$

We can now redefine A with a hierarchy:

- $A_k = \text{LogNormal}(A_{pop}, \sigma_A)$
- $A_{pop} = \text{LogNormal}(-0.5,1)$
- $\sigma_A = \text{LogNormal}(-2,0.1)$
- $\beta = \text{LogitNormal}(0,1)$
- $\epsilon = \text{LogNormal}(-2,0.1)$

Priors in GP [1]

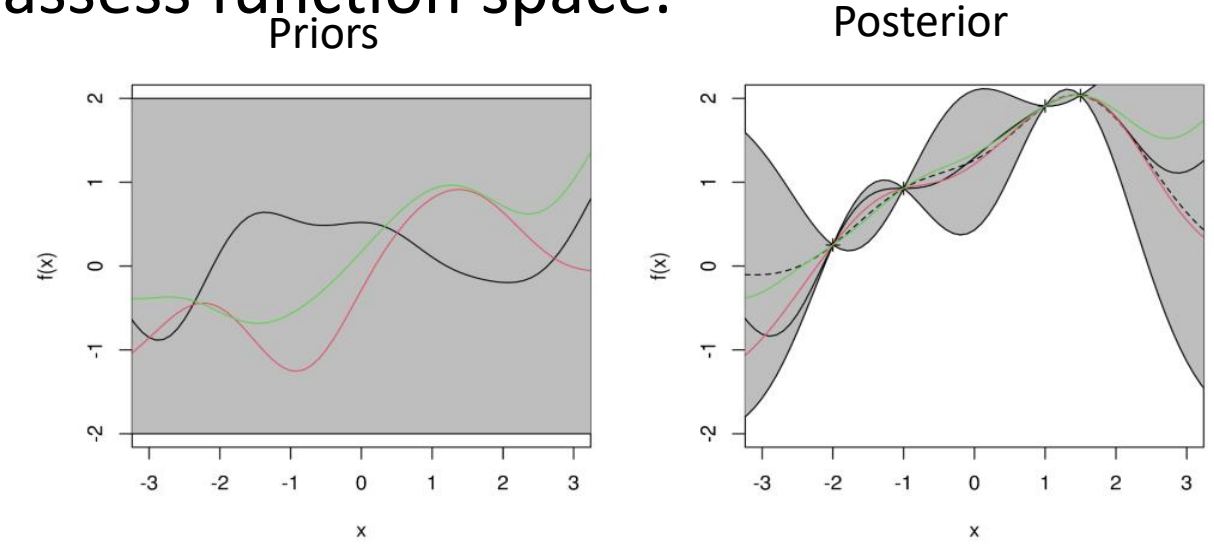
- How do priors connect to GP?
  - Some parameters of the kernels can be represented with priors.
- Example:
  - The Variance (σ^2) and the Length Scale (ℓ) in the RBF kernel can be represented by priors:

$$k(x_a, x_b) = \sigma^2 \exp\left(-\frac{\|x_a - x_b\|^2}{2\ell^2}\right)$$

  - $\sigma^2 \sim N(0, 1)$
  - $\ell \sim N(0.1, 2)$

Priors [9]

- Priors may also be analyzed via Markov Chain MonteCarlo (MCMC) Simulations. The priors realization can be plotted along with the training set to assess function space.



- Finally, assigning priors from previously known domain knowledge about the dataset can also be applied to narrow down the prior distribution.

Covariance matrix and mean vector

- Once you have your kernel function, you will want to produce four matrices:
  - K_{X,X^*} : the covariance between the training and testing points ($K_{X,X^*} = K_{X^*,X}^T$)
  - $K_{X^*,X}$: the covariance between the testing and training points ($K_{X^*,X} = K_{X,X^*}^T$)
  - K_{X^*,X^*} : the covariance between just the testing points
  - $K_{X,X}$: the covariance between just the training points
  - Where X and X^* are the training and test points respectively
- Then the mean and covariance can be easily computed for the gaussian process:

$$\mu = K(X^*, X)K(X, X)^{-1}y$$
$$\Sigma = K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*)$$

Conclusions

- Gaussian processes offer a powerful framework for time series forecasting, providing a nuanced understanding of uncertainty and adaptability crucial in dynamic environments.
- GPs can help modeling complex patterns and incorporate prior knowledge, this makes them versatile due to the Kernel Trick.
  - Proper kernel engineering involves understanding the data's characteristics and choosing a kernel function that best reflects those properties.
- Additionally, setting informed priors based on domain knowledge can significantly enhance the GP's performance by providing meaningful constraints.