Gaussian Process Regression

Justin Silverman

Penn State University

02/18/2021

Table of Contents

Sources

A collection of slides in this talk are taken from the following sources:

- Iain Murray, Introduction to Gaussian Processes
- David Duvenaud, The Kernel Cookbook

Section 1

Motivation

Recall: Non-linear basis functions

What does the "Linear" in Linear Regression mean?

Linear refers to linear in the parameters β only. This model is a linear regression model

$$y_i \sim N(\beta_0 + \beta_1 \sin(x) + \beta_2 x^3, \sigma^2)$$

It may make you more comfortable to realize that we can always just write $w = sin(x)$ and $z = x^3$ and then we have

$$y_i \sim N(\beta_0 + \beta_1 w + \beta_2 z, \sigma^2)$$

We can write linear regression using any non-linear basis functions $\phi(x)$:

$$y_i \sim N(\beta_0 + \beta_1 \phi_1(x_i) + \cdots + \beta_p \phi_p(x_i), \sigma^2)$$

and still use the same numerical and analogical solutions to solve for β.

Limitations of non-linear basis functions in linear models

- But how do you choose the non-linear basis functions $\{\phi_1, \ldots, \phi_p\}$?

Limitations of non-linear basis functions in linear models

- But how do you choose the non-linear basis functions $\{\phi_1, \ldots, \phi_p\}$?
- How many do we need to choose?

Limitations of non-linear basis functions in linear models

- But how do you choose the non-linear basis functions $\{\phi_1, \dots, \phi_p\}$?
- How many do we need to choose?
- You could do something like a Taylor series expansion and pick a polynomial basis functions (e.g., $\{1, x, x^2, \dots, x^p\}$).

Limitations of non-linear basis functions in linear models

- But how do you choose the non-linear basis functions $\{\phi_1, \dots, \phi_p\}$?
- How many do we need to choose?
- You could do something like a Taylor series expansion and pick a polynomial basis functions (e.g., $\{1, x, x^2, \dots, x^p\}$).
- If you want a really good approximation to a non-linear relationship then you need to have a large value of p and then you start running into $p >> n$ problems.

Limitations of non-linear basis functions in linear models

- But how do you choose the non-linear basis functions $\{\phi_1, \dots, \phi_p\}$?
- How many do we need to choose?
- You could do something like a Taylor series expansion and pick a polynomial basis functions (e.g., $\{1, x, x^2, \dots, x^p\}$).
- If you want a really good approximation to a non-linear relationship then you need to have a large value of p and then you start running into $p >> n$ problems.
- So lets look at modeling functions directly.

Functions as infinite dimensional objects

Let $x, y \in \mathcal{R}$

$$y = f(x) \Leftrightarrow y = \begin{bmatrix} f(-\infty) \\ \vdots \\ f(0) \\ \vdots \\ f(-\infty) \end{bmatrix}$$

So how do we work with infinite dimensional vectors? You obviously can't even represent these on a computer in closed form...

Section 2

Gaussian Processes Take 1

Proceduralist Approach to Gaussian Process Regression

- Intuition: Two observations i and j should have more similar y_i and y_j if x_i and x_j is similar.

Proceduralist Approach to Gaussian Process Regression

- Intuition: Two observations i and j should have more similar y_i and y_j if x_i and x_j is similar.
- We like the multivariate normal distribution, so why don't we just model all the $(y_1, \ldots, y_n)$ as:

$$y \sim N(\mu, \Sigma)$$

Proceduralist Approach to Gaussian Process Regression

- Intuition: Two observations i and j should have more similar y_i and y_j if x_i and x_j is similar.
- We like the multivariate normal distribution, so why don't we just model all the $(y_1, \ldots, y_n)$ as:

$$y \sim N(\mu, \Sigma)$$

where μ_i is just our prior expectation of y given x, e.g., I believe that y_i is most likely to have a value of

$$E[y_i] = \mu_i = m(x_i)$$

Proceduralist Approach to Gaussian Process Regression

- Intuition: Two observations i and j should have more similar y_i and y_j if x_i and x_j is similar.
- We like the multivariate normal distribution, so why don't we just model all the $(y_1, \ldots, y_n)$ as:

$$y \sim N(\mu, \Sigma)$$

where μ_i is just our prior expectation of y given x, e.g., I believe that y_i is most likely to have a value of

$$E[y_i] = \mu_i = m(x_i)$$

and where we determine the covariance between any two y_i and y_j to be a function of how similar x_i and x_j are, e.g.,

$$Cov(y_i, y_j) = \Sigma_{ij} = K(x_i, x_j)$$

We call m a mean function and K a kernel function. Note there are some properties K must follow to ensure that Σ is positive semi-definite.

Proceduralist Approach to Gaussian Process

With this specification we can then predict new values of y^* given x^* using the conditional properties of the multivariate normal. Given a new x^* we have

$$\begin{bmatrix} y \\ y^* \end{bmatrix} \sim N\left(\begin{bmatrix} \mu \\ m(x^*) \end{bmatrix}, \begin{bmatrix} \Sigma & K_* \\ K_*^T & K_{**} \end{bmatrix}\right).$$

Here we have introduced the shorthand

$$K_*^T = \begin{bmatrix} K(x^*, x_1) & \cdots & K(x^*, x_n) \end{bmatrix}$$

and finally

$$K_{**} = K(x^*, x^*).$$

Notice that K_*^T provides the necessary information to determine how the training test points influence our belief in the value of the new test point.

Then we can just use the conditional properties of the multivariate normal to solve for $p(y^*|y, x, x^*)$

Section 3

Gaussian Processes Take 2

Bayesian Linear Regression as Functional Regression

$$y_i \sim N(\beta x_i, \sigma^2)$$
$$\beta \sim N(\mu, \tau^2 I)$$

Bayesian Linear Regression as Functional Regression

$$y_i \sim N(\beta x_i, \sigma^2)$$
$$\beta \sim N(\mu, \tau^2 I)$$

Let X be of dimension $q \times n$, defined as:

$$X = \begin{bmatrix} | & \cdots & | \\ x_1 & \cdots & x_n \\ | & \cdots & | \end{bmatrix}.$$

Then the above model can be alternatively written as:

$$y \sim N(\beta X, \sigma^2 I)$$
$$\beta \sim N(\mu, \tau^2 I)$$

Bayesian Linear Regression as Functional Regression

$$y \sim N(\beta X, \sigma^2 I)$$
$$\beta \sim N(\mu, \tau^2 I)$$

This can be written alternatively as:

$$y = \beta x + \epsilon^y \quad \epsilon_i^y \sim N(0, \sigma^2 I) \tag{1}$$
$$\beta = \mu + \epsilon^\beta \quad \epsilon^\beta \sim N(0, \tau^2 I) \tag{2}$$

Bayesian Linear Regression as Functional Regression

$$y \sim N(\beta X, \sigma^2 I)$$
$$\beta \sim N(\mu, \tau^2 I)$$

This can be written alternatively as:

$$y = \beta x + \epsilon^y \quad \epsilon_i^y \sim N(0, \sigma^2 I) \tag{1}$$
$$\beta = \mu + \epsilon^\beta \quad \epsilon^\beta \sim N(0, \tau^2 I) \tag{2}$$

Substituting (2) into (1) gives:

$$y = \mu x + \epsilon^\beta x + \epsilon^y \quad \epsilon^y \sim N(0, \sigma^2 I) \quad \epsilon^\beta \sim N(0, \tau^2 I)$$

Bayesian Linear Regression as Functional Regression

$$y \sim N(\beta X, \sigma^2 I)$$
$$\beta \sim N(\mu, \tau^2 I)$$

This can be written alternatively as:

$$y = \beta x + \epsilon^y \quad \epsilon_i^y \sim N(0, \sigma^2 I) \tag{1}$$
$$\beta = \mu + \epsilon^\beta \quad \epsilon^\beta \sim N(0, \tau^2 I) \tag{2}$$

Substituting (2) into (1) gives:

$$y = \mu x + \epsilon^\beta x + \epsilon^y \quad \epsilon^y \sim N(0, \sigma^2 I) \quad \epsilon^\beta \sim N(0, \tau^2 I)$$

This has the following marginal distribution for y as (review slides on multivariate normal):

$$y \sim N(\mu x, \sigma^2 I + \tau^2 X X^T)$$

Bayesian Linear Regression as Functional Regression

We now have two models that are identical in how they model the observations y.

Standard Representation of Bayesian Linear Regression

$$y_i \sim N(\beta x_i, \sigma^2 I)$$
$$\beta \sim N(\mu, \tau^2 I)$$

Here we model each y_i and y_j as conditionally independent given β.

This New Representation of Bayesian Linear Regression

$$y \sim N(\mu x, \sigma^2 I + \tau^2 X X^T)$$

Here we have marginalized out β. This later representation is a Gaussian process.

Bayesian Linear Regression as Functional Regression

$$y \sim N(\mu x, \sigma^2 I + \tau^2 X X^T)$$

Lets look at the special case where there are just two observed data points (y_1, x_1) and (y_2, x_2). For some chosen μ, τ^2 and σ^2 this gives us:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu x_1 \\ \mu x_2 \end{bmatrix}, \begin{bmatrix} \sigma^2 + \tau x_1 x_1^T & \tau x_1 x_2^T \\ \tau x_2 x_1^T & \sigma^2 + \tau x_2 x_2^T \end{bmatrix}\right).$$

Bayesian Linear Regression as Functional Regression

$$y \sim N(\mu x, \sigma^2 I + \tau^2 X X^T)$$

Lets look at the special case where there are just two observed data points (y_1, x_1) and (y_2, x_2). For some chosen μ, τ^2 and σ^2 this gives us:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu x_1 \\ \mu x_2 \end{bmatrix}, \begin{bmatrix} \sigma^2 + \tau x_1 x_1^T & \tau x_1 x_2^T \\ \tau x_2 x_1^T & \sigma^2 + \tau x_2 x_2^T \end{bmatrix} \right).$$

We can instead think of this as:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim N\left(\begin{bmatrix} m(x_1) \\ m(x_2) \end{bmatrix}, \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) \\ K(k_2, x_1) & K(x_2, x_2) \end{bmatrix} \right).$$

where $m(x) = \mu x$ is called a **mean function** and $K(x_i, x_j) = \sigma^2 I_{i=j} + \tau^2 x_i x_j^T$ is called a **kernel function**.

Bayesian Linear Regression as Functional Regression

$$y \sim N(\mu x, \sigma^2 I + \tau^2 X X^T)$$

Lets look at the special case where there are just two observed data points (y_1, x_1) and (y_2, x_2). For some chosen μ, τ^2 and σ^2 this gives us:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu x_1 \\ \mu x_2 \end{bmatrix}, \begin{bmatrix} \sigma^2 + \tau x_1 x_1^T & \tau x_1 x_2^T \\ \tau x_2 x_1^T & \sigma^2 + \tau x_2 x_2^T \end{bmatrix} \right).$$

We can instead think of this as:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim N\left(\begin{bmatrix} m(x_1) \\ m(x_2) \end{bmatrix}, \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) \\ K(k_2, x_1) & K(x_2, x_2) \end{bmatrix} \right).$$

where $m(x) = \mu x$ is called a **mean function** and
$K(x_i, x_j) = \sigma^2 I_{i=j} + \tau^2 x_i x_j^T$ is called a **kernel function**.

We can specify lots of different functions for m and K and get powerful methods for non-linear regression.

Section 4

Gaussian Processes

Gaussian Processes (GPs)

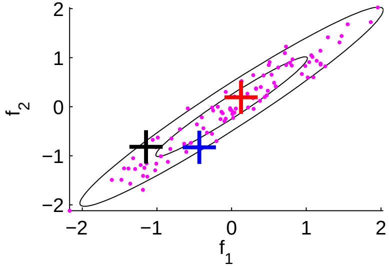
$$f \sim \mathcal{GP}(m(\cdot), K(\cdot,\cdot))$$

You can think of a Gaussian process as a distribution over functions f (aka a stochastic process) parameterized by a mean function $m(\cdot)$ and a kernel function $K(\cdot,\cdot)$ such that for any finite collection of points $(x_1, \dots, x_n)$ the probability distribution for $f(x) = (f(x_1), \dots, f(x_n))$ is given by

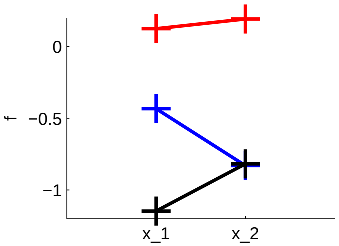
$$f(x) = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim N\left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_n) \end{bmatrix}, \begin{bmatrix} K(x_1,x_1) & \cdots & K(x_1,x_n) \\ \vdots & \ddots & \vdots \\ K(x_n,x_1) & \cdots & K(x_n,x_n) \end{bmatrix} \right).$$

Laying out Gaussians

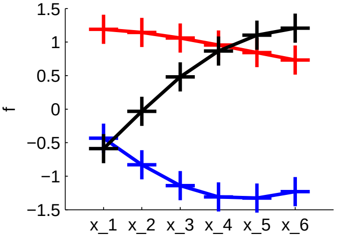
A way of visualizing draws from a 2D Gaussian:



$\Leftrightarrow$

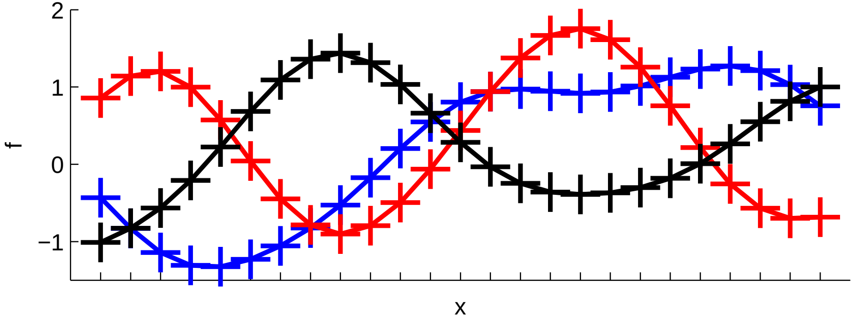


Now it's easy to show three draws from a 6D Gaussian:



Building large Gaussians

Three draws from a 25D Gaussian:



To produce this, we needed a mean: I used `zeros(25,1)`

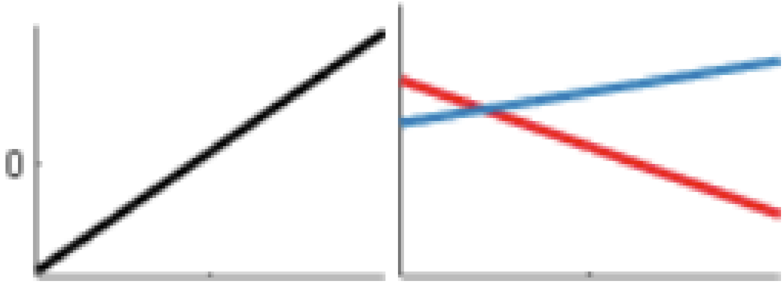
The covariances were set using a *kernel* function: $\Sigma_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

The $\mathbf{x}$'s are the positions that I planted the tics on the axis.

Later we'll find k's that ensure Σ is always positive semi-definite.

GPs with Linear Kernel

Taking $m(\cdot) = 0$ *just to illustrate kernel effect*



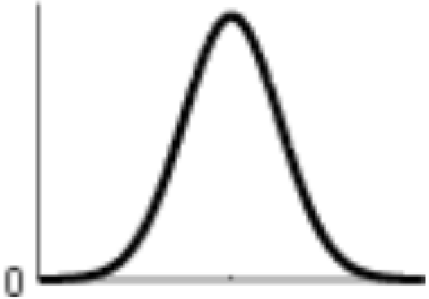
$$k_{\mathrm{Lin}}(x, x') = \sigma_b^2 + \sigma_v^2 (x - c)(x' - c)$$

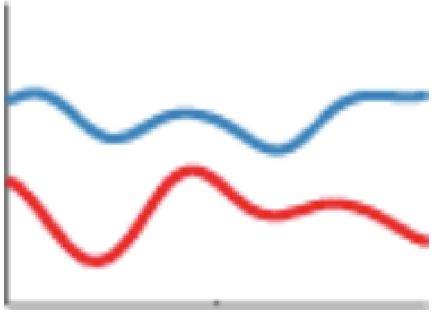
GPs with RBF Kernels

Taking $m(\cdot) = 0$ *just to illustrate kernel effect*

$$K(x_i, x_j) = \sigma^2 e^{-\frac{(x_i - x_j)^2}{2\ell^2}}$$

RBF kernels = Radial Basis Function Kernels. These are also called the Gaussian Kernels or the Squared Exponential Kernels. These are often the *de-facto* kernels chosen in applied projects.



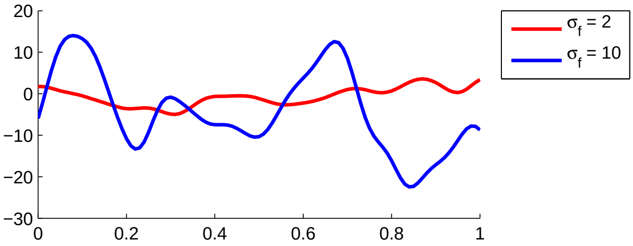


Meaning of hyper-parameters

Many kernels have similar types of parameters:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\tfrac{1}{2} \sum_{d=1}^{D} (x_{d,i} - x_{d,j})^2 / \ell_d^2 \right),$$

Consider $\mathbf{x}_i = \mathbf{x}_j$, $\Rightarrow$ marginal function variance is σ_f^2

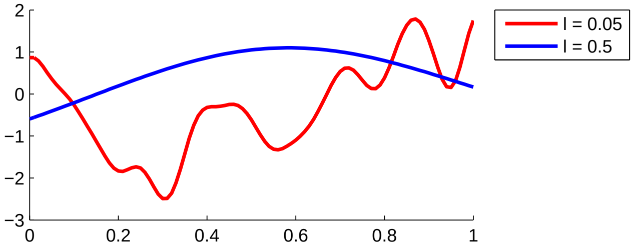


Meaning of hyper-parameters

The ℓ_d parameters give the overall lengthscale in dimension-d

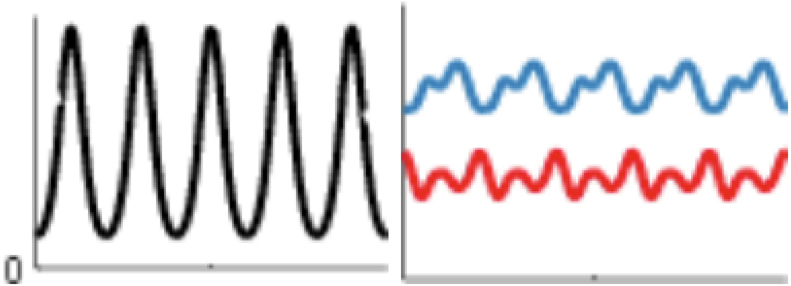
$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\tfrac{1}{2} \sum_{d=1}^{D} (x_{d,i} - x_{d,j})^2 / \ell_d^2 \right),$$

Typical distance between peaks $\approx \ell$



Periodic Kernels

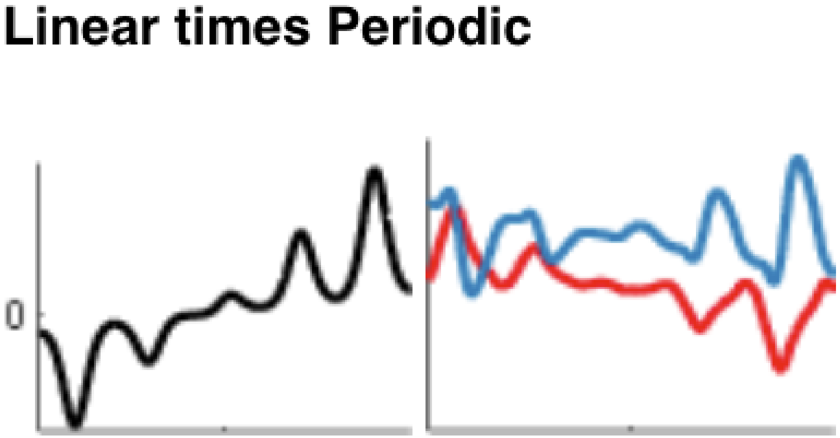
Taking $m(\cdot) = 0$ *just to illustrate kernel effect*



$$k_{\mathrm{Per}}(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x-x'|/p)}{\ell^2}\right)$$

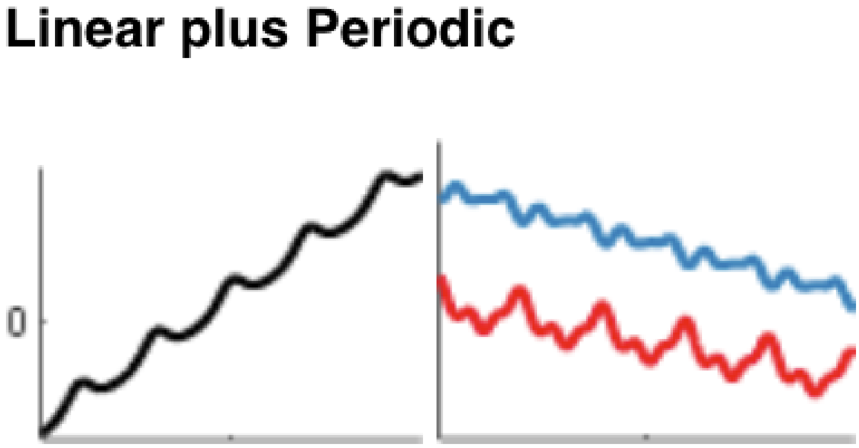
Multiplying Kernels

When you multiply two kernels together you get something like an **AND** function (two points will be highly correlated if they are highly correlated in both kernels):



Adding Kernels

When you multiply two kernels together you get something like an **OR** function (two points will be highly correlated if they are highly correlated in either kernel):



Other Types of Kernels

- Multidimensional Kernels
- Symmetric Kernels
- Changepoint Kernels
- Time-series Kernels
- And so much more. . .

A good resource is the Kernel Cookbook.

Section 5

Gaussian Processes Take 3

Gaussian Process Regresion

Our goal is to learn $f \in \mathcal{H}_K$[1] for the following type of model:

$$y_i \sim N(f(x_i), \sigma^2)$$

We want to infer f but it is infinite dimensional, e.g., this is the worst $p >> n$ problem. So we are going to put a prior on f and do Bayesian inference.

[1] $\mathcal{H}_K$ is a Reproducing Kernel Hilbert Space, don't get too bogged down here, just think of it as a mathematical space of functions.

Gaussian Process Regression

We are going to do Bayesian inference by placing a Gaussian Process prior on f such that the overall model can be written as:

$$\begin{aligned} y_i &\sim N(f(x_i), \sigma^2) \\ f &\sim \mathcal{GP}(m(\cdot), K(\cdot, \cdot)) \end{aligned}$$

Gaussian Process Regression

We are going to do Bayesian inference by placing a Gaussian Process prior on f such that the overall model can be written as:

$$\begin{aligned} y_i &\sim N(f(x_i), \sigma^2) \\ f &\sim \mathcal{GP}(m(\cdot), K(\cdot, \cdot)) \end{aligned}$$

Since f is infinite dimensional we cannot evaluate/sample the posterior $p(f|y, x)$ directly, instead we focus on evaluating $p(f(x^*)|y, x)$ for some test point x^*. **That is, we focus on predicting the function f on some new test point x^*.**

Gaussian Process Regression

Letting f^* be shorthand for $f(x^*)$, the joint density of the observed data and the latent, noise-free function evaluated at the test point x^* is given by

$$\begin{bmatrix} y \\ f^* \end{bmatrix} \sim N\left(\begin{bmatrix} m(x) \\ m(x^*) \end{bmatrix}, \begin{bmatrix} K_x & K_* \\ K_*^T & K_{**} \end{bmatrix} \right).$$

Here we have introduced the shorthand

$$K_x = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \cdots & K(x_n, x_n) \end{bmatrix}$$

and

$$K_*^T = \begin{bmatrix} K(x^*, x_1) & \cdots & K(x^*, x_n) \end{bmatrix}$$

and finally

$$K_{**} = K(x^*, x^*).$$

Notice that K_*^T provides the necessary information to determine how the training test points influence our belief in the value of the new test point.

Gaussian Process Regression

$$\begin{bmatrix} y \\ f^* \end{bmatrix} \sim N\left(\begin{bmatrix} m(x) \\ m(x^*) \end{bmatrix}, \begin{bmatrix} K_x & K_* \\ K_*^T & K_{**} \end{bmatrix}\right)$$

We can now use the conditional properties of the multivariate normal to get:

$$\begin{aligned} f^*|y, x, x^* &\sim N(\mu^*, \Sigma^*) \\ \mu^* &= m(x^*) + K_*^T K_x^{-1}[y - m(x)] \\ \Sigma^* &= K_{**} - K_*^T K_x^{-1} K_* \end{aligned}$$

Gaussian Process Regression

$$\begin{bmatrix} y \\ f^* \end{bmatrix} \sim N\left(\begin{bmatrix} m(x) \\ m(x^*) \end{bmatrix}, \begin{bmatrix} K_x & K_* \\ K_*^T & K_{**} \end{bmatrix}\right)$$

We can now use the conditional properties of the multivariate normal to get:

$$\begin{aligned} f^*|y, x, x^* &\sim N(\mu^*, \Sigma^*) \\ \mu^* &= m(x^*) + K_*^T K_x^{-1}[y - m(x)] \\ \Sigma^* &= K_{**} - K_*^T K_x^{-1} K_* \end{aligned}$$

Weighting the Training Points

Letting $m = 0$ another way to write the posterior mean is as:

$$\mu^*_{m=0} = K_*^T K_x^{-1} y = \sum_{i=1}^{n} \alpha_i K(x_i, x^*)$$

where $\alpha = K_x^{-1} y$.

Computational Issues

K_x is a matrix of dimension $n \times n$, it is unwise to invert this matrix directly. Instead other methods for solving linear systems are faster and more numerically stable.

Computational Issues

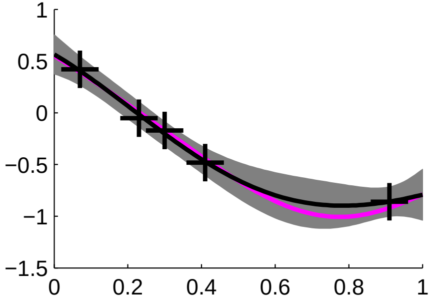
K_x is a matrix of dimension $n \times n$, it is unwise to invert this matrix directly. Instead other methods for solving linear systems are faster and more numerically stable.

Still, even with numerical tricks, GPs are typically $\mathcal{O}(n^3)$ which can be too slow for big problems.

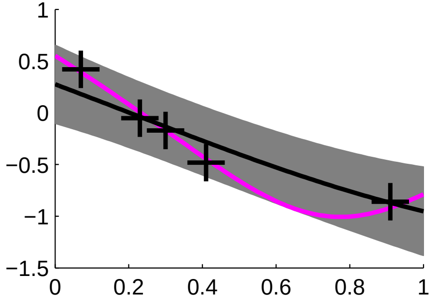
see Murphy, Machine Learning a Probabilistic Perspective, pg 526 for more details.

Effect of hyper-parameters

Different (SE) kernel parameters give different explanations of the data:



$\ell = 0.5, \ \ \sigma_n = 0.05$



$\ell = 1.5, \ \ \sigma_n = 0.15$

Infering Hyperparameters

Often there will be hyper-parameters in the Kernel function that you will want to learn from data.

e.g., the bandwidth ℓ in the RBF kernel:

$$K(x_i, x_j) = \sigma^2 e^{-\frac{(x_i - x_j)^2}{2\ell^2}}.$$

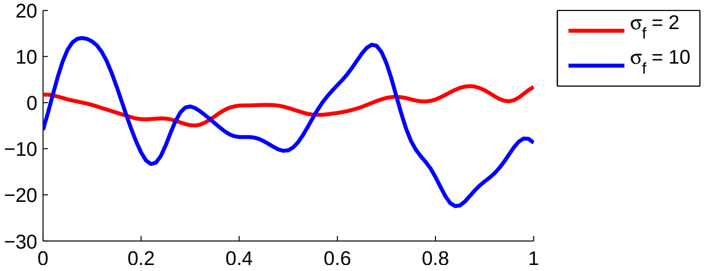
While there are multiple approaches for learning these parameters, two of the most common are:

1. Cross-validation
2. Maximum Marginal Likelihood, e.g., optimizing

$$p(y|[\text{hyper-parameters}])$$

Mean function

Using $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, K)$ is common



If your data is not zero-mean this is a poor model.
Center your data, or use a parametric mean function $m(\mathbf{x})$.

We can do this: the posterior is a GP with non-zero mean.

Other tricks

To set initial hyper-parameters, **use domain knowledge wherever possible**. Otherwise. . .

Standardize input data and set lengthscales to ~ 1.

Standardize targets and set function variance to ~ 1.

Often useful: **set initial noise level high**, even if you think your data have low noise. The optimization surface for your other parameters will be easier to move in.

If optimizing hyper-parameters, (as always) random restarts or other tricks to **avoid local optima** are advised.

Section 6

Gaussian Process Classification

Gaussian Process Classification

Just like linear models, we can create generalized GP models. A particularly useful one is the GP equivalent to logistic regression.

$$\begin{aligned} y_i &\sim \text{Bernoulli}(\pi_i) \\ \pi_i &= \text{Logit}^{-1}(\eta_i) \\ \eta &\sim \mathcal{GP}(m(\cdot), K(\cdot, \cdot)) \end{aligned}$$

This permits flexible non-linear decision boundaries to be learned from the data without having to use a-priori defined basis functions.

As with logistic regression, no closed from exists for the posterior and numerical methods must be used.

Section 7

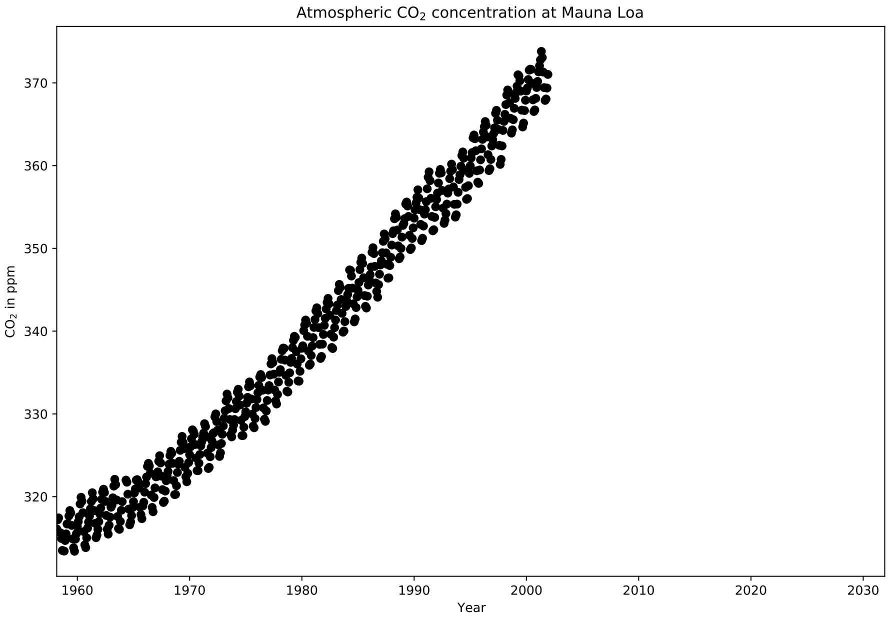
Gaussian Process Regression Example

Gaussian Process Regression Example

We will follow the often used Mauna Loa CO2 data example, code can be found here.

The goal is to model and predict Atmospheric CO2 concentration at Mauna Loa (which is in Hawaii).

The Data


Atmospheric CO_2 concentration at Mauna Loa
CO_2 in ppm
320
330
340
350
360
370
1960
1970
1980
1990
2000
2010
2020
2030
Year

The Kernel

Rasmussen and Williams proposed a kernel with 4 parts (kernel addition).

- RBF kernel with long band-width to explain smooth rising trend observed over multiple years. *The specific length-scale and the amplitude are free hyper-parameters to be optimized.*

The Kernel

Rasmussen and Williams proposed a kernel with 4 parts (kernel addition).

- RBF kernel with long band-width to explain smooth rising trend observed over multiple years. *The specific length-scale and the amplitude are free hyper-parameters to be optimized.*
- Periodic Kernel with fixed 1 year periodicity to model seasonality. In order to allow decaying away from exact periodicity, the product with an RBF kernel is taken. *The length-scales of these two components (periodic and RBF kernels) are free parameters.*

The Kernel

Rasmussen and Williams proposed a kernel with 4 parts (kernel addition).

- RBF kernel with long band-width to explain smooth rising trend observed over multiple years. *The specific length-scale and the amplitude are free hyper-parameters to be optimized.*
- Periodic Kernel with fixed 1 year periodicity to model seasonality. In order to allow decaying away from exact periodicity, the product with an RBF kernel is taken. *The length-scales of these two components (periodic and RBF kernels) are free parameters.*
- Rational Quadratic Kernel to explain smaller and medium term irregularities. RQ kernel can accommodate a variety of length-scales whereas an RBF kernel requires a single length-scale.

The Kernel

Rasmussen and Williams proposed a kernel with 4 parts (kernel addition).

- RBF kernel with long band-width to explain smooth rising trend observed over multiple years. *The specific length-scale and the amplitude are free hyper-parameters to be optimized.*
- Periodic Kernel with fixed 1 year periodicity to model seasonality. In order to allow decaying away from exact periodicity, the product with an RBF kernel is taken. *The length-scales of these two components (periodic and RBF kernels) are free parameters.*
- Rational Quadratic Kernel to explain smaller and medium term irregularities. RQ kernel can accommodate a variety of length-scales whereas an RBF kernel requires a single length-scale.
- An RBF kernel + Noise term to explain short-term fluctuations such as weather

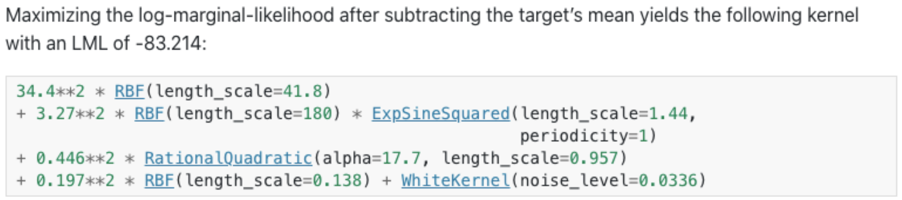
The Mean Function

Scikit-learn doesn't give us much flexibility here:

The prior mean is assumed to be constant and zero (for `normalize_y=False`) or the training data's mean (for `normalize_y=True`).

We use `normalize_y=TRUE` so we assume $m = 0$.

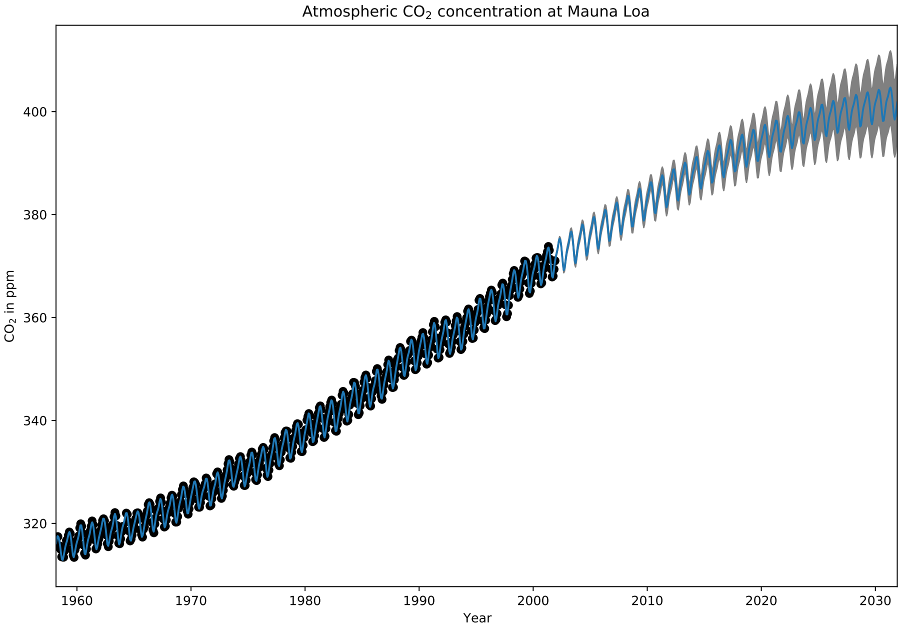
The Optimized Kernel



Maximizing the log-marginal-likelihood after subtracting the target's mean yields the following kernel with an LML of -83.214:

```
34.4**2 * RBF(length_scale=41.8)
+ 3.27**2 * RBF(length_scale=180) * ExpSineSquared(length_scale=1.44,
                                                   periodicity=1)
+ 0.446**2 * RationalQuadratic(alpha=17.7, length_scale=0.957)
+ 0.197**2 * RBF(length_scale=0.138) + WhiteKernel(noise_level=0.0336)
```

The fitted Model



Section 8

Further Resources

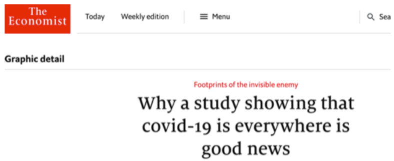
Further Resources

- If you want to play with Gaussian Processes, Check this out
- The definitive guide to modeling with Gaussian Processes, Rasmussen and Williams

Section 9

Gaussian Processes for COVID Surveillance

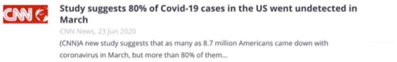
I was planning on a quite move to PSU this past summer


The Economist Today Weekly edition Menu Sea

Graphic detail

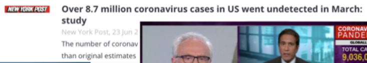
Footprints of the invisible enemy

Why a study showing that covid-19 is everywhere is good news


Study suggests 80% of Covid-19 cases in the US went undetected in March

CNN News, 23 Jun 2020

(CNN)A new study suggests that as many as 8.7 million Americans came down with coronavirus in March, but more than 80% of them...


Over 8.7 million coronavirus cases in US went undetected in March: study

New York Post, 23 Jun 2

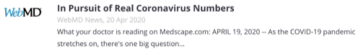
The number of coronav

than original estimates

Why We Don't Know the True Death Rate for Covid-19

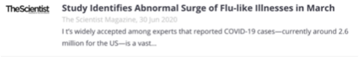
New York Times, 17 Apr 2020

The Coronavirus Outbreak debugid:204 • Latest Updates Supported by Determining what percentage of those infected by the...


In Pursuit of Real Coronavirus Numbers

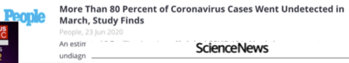
WebMD News, 20 Apr 2020

What your doctor is reading on Medscape.com: APRIL 19, 2020 -- As the COVID-19 pandemic stretches on, there's one big question...


Study Identifies Abnormal Surge of Flu-like Illnesses in March

The Scientist Magazine, 30 Jun 2020

I t's widely accepted among experts that reported COVID-19 cases—currently around 2.6 million for the US—is a vast...


More Than 80 Percent of Coronavirus Cases Went Undetected in March, Study Finds

People, 23 Jun 2020

An estim

undiagn

ScienceNews

NEWS Health & Medicine

Millions of COVID-19 cases in the U.S. may have gone undiagnosed in March

A surge of flulike illnesses suggests the virus was already much more widespread than thought

Why was this picked up by mainstream news in Late March 2020?

What was widely accepted at the time:

- 6.4 day doubling time
- 120,000 infections between March 8 and March 28
- Infection Fatality Rate as high as 3.4 (few researchers suggest closer to 1%)

What we suggested

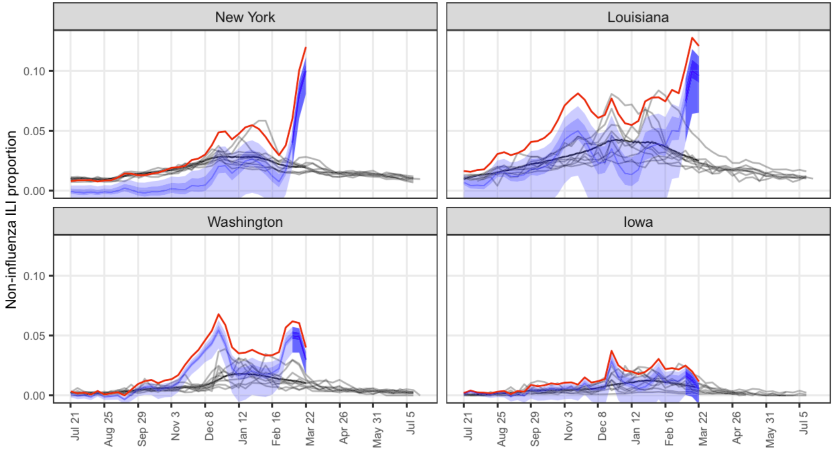
- 3 day doubling time (faster)
- $\geq$ 8.7 million infections between March 8 and 28th
- (weak suggestion) IFR may be lower and closer to 0.1-0.6%

Influenza-Like Illness (ILI)

- Early Warning System for Flu-like illness
- ILI = Fever + Cough and/or runny nose
- $\approx$ 2600 (sentinel) outpatient health care provides
- $\approx$ 60 million outpatient encounters recorded each year
- Aggregated data, reported at state-level going back to 2010 (Number of patients seen per week and Number of those who have ILI)

The ILI Surge

SCIENCE TRANSLATIONAL MEDICINE | REPORT



Silverman et al. (2020) Science Translational Medicine

The Gaussian Process Model

$$\tilde{y}_{it} \sim \text{Binomial}(\pi_{it}, n_{it})$$

$$\pi_{it} = \frac{\exp(\eta_{it})}{1 + \exp(\eta_{it})}$$

$$\eta_{it} \sim N(\lambda_i(t), \sigma^2)$$

$$\lambda_i(t) \sim \text{GP}(\theta(t), \sigma^2 \boldsymbol{\Gamma})$$

$$\sigma^2 \sim \text{InverseGamma}(\nu, \xi)$$

$$\theta(t) = \theta$$

$$\Gamma(t, t+s) = \alpha \exp\left(\frac{-s^2}{2\rho^2}\right)$$

Silverman et al. (2020) Science Translational Medicine