# Gaussian Processes

Cesar Pardede, Marc Sanchez, and Isaac Zhou

December 16, 2020

## 1 Introduction

A Gaussian Process (GP) is a generalization of a Multivariate Normal Distribution (MVN) and as such it shares many of the MVN's properties. Gaussian Process regression also known as kriging is a stochastic process which originated from the field of geo-statistics in South Africa. Danie S. Krieg developed the process to estimate the distribution of gold deposits between data sites which made finding gold easier. Gaussian processes are used to interpolate between data points for sparse datasets and gives confidence intervals for these interpolations. It can be used for regression and classification.

So we start with a quick review of the MVN and some important properties.

**Definition 1** (PDF for the MVN in $d$ dimensions)**.** *The PDF $p(\mathbf{x}|\mu, \Sigma)$ for the Multivariate Normal Distribution in $\mathbb{R}^d$ is given by:*

$$p(\mathbf{x}|\mu, \sigma) = \frac{1}{(2\pi)^d |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right]$$

*Where $x = (X_1, X_2, ..., X_d) \in \mathbb{R}^d$, $\mu = E[\mathbf{x}]$, and $\Sigma = cov(\mathbf{x})$ is the covariance matrix. The covariance matrix is symmetric and positive definite. It's entries $\Sigma_{ij}$ are given by $\Sigma_{ij} = cov(X_i, X_j)$*

This first set of properties will be shared with our GP.

**Properties 1.** *Here are some of the properties of an MVN in $\mathbb{R}^d$*

1. *a sample from an MVN is a collection of vectors from $\mathbb{R}^d$:*
2. *all marginal distributions of an MVN are MVN*
3. *all conditional distributions of an MVN that are obtained by conditioning on a subset of the variables are also MVN*

This second set of properties will give us some computational tools to use with our GP.

**Properties 2.** *Here are some of the properties of conditional distributions of an MVN. If we partition* $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2})^T$*, then we have the following properties:*

1. $\mu = (\mu_1, \mu_2) = (E[\mathbf{x_1}], E[\mathbf{x_2}])$
2. $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \begin{pmatrix} cov(\mathbf{x_1}, \mathbf{x_1}) & cov(\mathbf{x_1}, \mathbf{x_2}) \\ cov(\mathbf{x_2}, \mathbf{x_1}) & cov(\mathbf{x_2}, \mathbf{x_2}) \end{pmatrix}$
3. *If* $\mu_{1|2}$ *represents the mean of* $\mathbf{x_1}|\mathbf{x_2}$*, then* $\mu_{1|2} = E[\mathbf{x_1}|\mathbf{x_2}] = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x_2} - \mu_2)$
4. *If* $\Sigma_{1|2}$ *is the covariance matrix of* $\mathbf{x_1}|\mathbf{x_2}$*, then* $\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$
5. $p(\mathbf{x_1}|\mathbf{x_2}) = N(\mathbf{x_1}|\mu_{1|2}, \Sigma_{1|2})$

Now we can define the GP

**Definition 2** (Gaussian Process on a set $\Omega$)**.** *A Gaussian Process is a collection of random variables indexed by* $\Omega$*, any finite number of which have a Multivariate Normal Distribution.*

So the GP is this (potentially very large) collection of random variables, but when we just look at any finite subset of them we end up with an MVN. Notice that if we only look at one of the random variables, it has to be univariate normal.

As an example if our indexing set is finite, then it is easy to convince yourself that we have an MVN. If the indexing set is infinite, it is best to think of a sample as a function from the indexing set into $\mathbb{R}$. Typically the indexing set will be $\mathbb{R}^N$.

The GP is completely specified by it's mean function $m(\mathbf{x})$ and it's covariance function $k(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are both members of the indexing set.

The covariance (or kernel) function $k(\mathbf{x}, \mathbf{x}')$ of a Gaussian Process (GP) is a function that generalizes the covariance function $cov(\mathbf{x}, \mathbf{x}')$, it serves as a measure of how similar are $\mathbf{x}$ and $\mathbf{x}'$. Similarity here means the function is has larger values when $\mathbf{x}$ and $\mathbf{x}'$ are "close" and smaller values when they are "far".

**Properties 3.** *The covariance function has the following properties:*

- *It is defined for all pairs* $(\mathbf{x}, \mathbf{x}')$ *of vectors from the indexing space.*
- *It is symmetric,* $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$*.*
- *It is non-negative,* $k(\mathbf{x}, \mathbf{x}) \geq 0$*.*
- *It is positive definite and this means for each set of vectors* $\mathbf{x}_1$*,* $\mathbf{x}_2$*, ...*$\mathbf{x}_N$*, the matrix with entries* $k(\mathbf{x}_i, \mathbf{x}_j)$ *is positive definite.*

It should be noted that showing a function satisfies the non-negative property is typically easy to verify, but proving the positive definite property requires more work.

An example of a commonly used covariance function is the squared exponential covariance function $k(\mathbf{x}, \mathbf{x}') = \tau^2 \exp\left(-\frac{|\mathbf{x}, -\mathbf{x}'|^2}{2l^2}\right)$.

Once the GP process has been defined and a set of points $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}$ has been selected, the covariance matrix $K$ is constructed by making the $(i, j)$ entry equal to $k(\mathbf{x_i}, \mathbf{x_j})$. With these selections, we now have a MVN namely $N(\boldsymbol{\mu}, K)$, where $\boldsymbol{\mu} = (m(\mathbf{x_1}), m(\mathbf{x_2}), ..., m(\mathbf{x_N}))$ from where we can sample a vector $(f_1, f_2, ..., f_N)$. The assignment $f(\mathbf{x_i}) = f_i$ can be viewed as part of a sampled function from the GP. So the GP is the structure that exists before any observed data is considered, which is the role of the prior in Bayesian Theory and the sampled function is just a sample from the prior. Summarizing, once we have selected a set of points from the indexing set, we can construct a MVN and the samples from that MVN can be interpreted as parts of functions from the GP and these functions in turn will be the samples from the prior determined by the GP.

The next step is to determine how to generate samples for the posteriors. Suppose we have $X = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$ so we can approximate samples from the prior and we also have a set of observed values $(X^*, Y^*)$. We can use this to construct a new MVN $N(\mu', K')$ with mean $\mu' = (m(X), Y^*)^T$ and covariance $K' = \begin{pmatrix} K(X, X) & K(X, X^*) \\ K(X^*, X^*) & K(X^*, X^*) \end{pmatrix}$. By conditioning on the observed values and using the formulas for the conditional distribution parameters of a MVN we get that a new MVN with parameters:

- Mean: $m(X) + K(X, X^*) K(X^*, X^*)^{-1} (Y^* - m(X^*))$
- Covariance: $K(X, X) - K(X, X^*)K(X^*, X^*)^{-1}K(X*, X)$

If we have a sampled value from this conditional MVN we can reason (as we did above for the prior) to view this as part of a function and since this function was obtained from a distribution that was conditioned on observed values, the function can be viewed as a sample of the posterior distribution.

## 2 Priors

A Gaussian process prior is a random function where the values at $n$ discrete points are a draw $\mu$ from an $n$-dimensional normal distribution with a mean vector $m$ (of size $n$) and a covariance matrix $K$ $(n \times n)$. In other words, we take an interval and break it up into $n$ discrete points $x_1, x_2, ..., x_n \in \mathbf{x}$, define a mean $m(x_i)$ for each $x_i$ and a covariance matrix $K$, then draw

$$\mu(x_1), \mu(x_2), ..., \mu(x_n) \sim N((m(x_1), m(x_2), ..., m(x_n)), K(x_1, x_2, ..., x_n)).$$

In Figure 1 (top left), we show an example of specifying points within an interval, and define the mean function to be $m(\cdot) = 0$ at each point. Then, in Figure 1 (top right) we show an example of one draw from our specified prior. As we take more and more

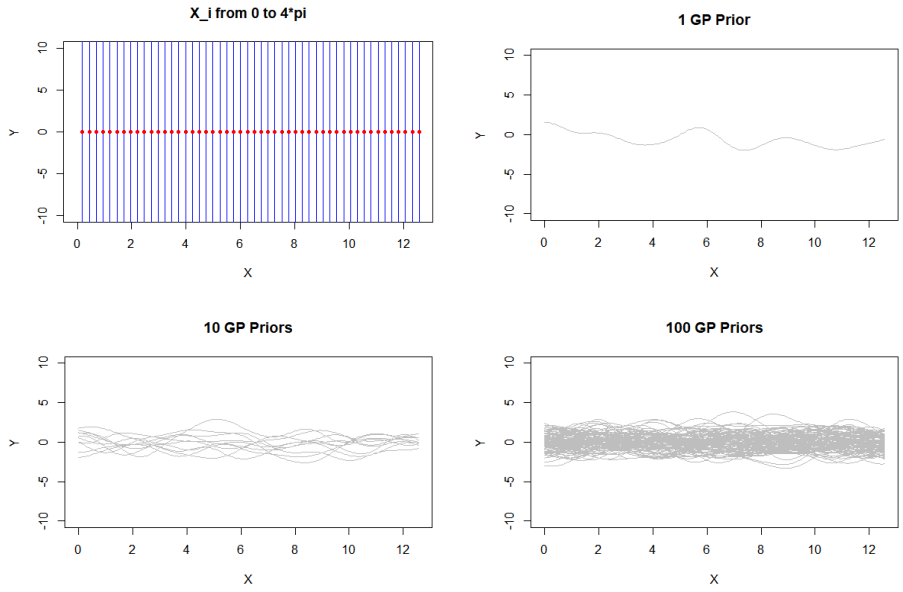



Figure 1: Pre-specified $\mathbf{x}$ with mean function $m(x) = 0$ for all $x_i \in \mathbf{x}$

draws (Figure 1 bottom row), the draws begin to resemble a density of functions, with a distribution of values at each $x_i$. In fact, if we were to take a slice across these functions at any $x_i$, we would see a univariate Gaussian at every slice as illustrated in Figure 2.

As a quick note, we mention there are other choices of GP prior. Figure 3 shows some examples of potential priors, which prior to specify using the mean function $m(\cdot)$ depends on your prior belief of what underlying function produces your data. The mean function we chose, $m(\cdot)$, is akin to choosing a non-informative prior, since we can center any set of data around zero by subtracting the mean of the data. For the rest of our example, assume we are working with the "non-informative" zero-centered GP prior.

# 3 Posteriors

Next, we look at how we can fit our GP priors to data to create GP posteriors. Suppose we have the data points shown in blue in Figure 4, we overlay it on our GP priors. We want to incorporate the data into our draws from an n-dimensional Gaussian by defining a joint distribution of our priors and data, then making draws from a conditional multivariate distribution conditioned on our data.

We create a joint distribution of our priors and data by concatenating our data $y'$ onto the mean vector (previously defined solely by our $m(\cdot)$s), and define a new covariance matrix $\mathbf{K}$ with blocks being the covariance matrices between our original $\mathbf{x}$ and the $\mathbf{x}$' from

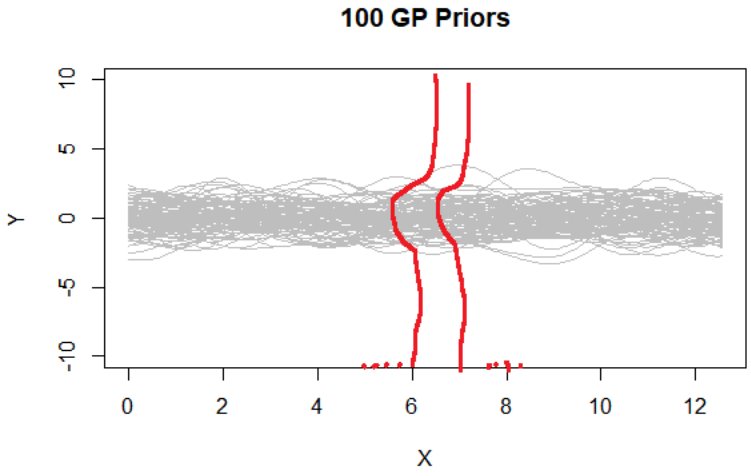



Figure 2: Univariate Gaussians at each specified x.

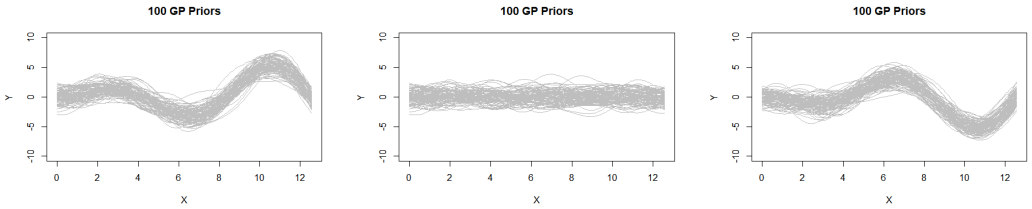



Figure 3: Different choices of GP priors.

our data.

$$\mu(x_1), \mu(x_2), ..., \mu(x_n), \mu(x_1'), ..., \mu(x_4') \sim N((m(x_1), m(x_2), ..., m(x_n), y_1', ..., y_4'), \mathbf{K})$$

where $\mathbf{K} = \begin{pmatrix} K(\mathbf{x}, \mathbf{x}) & K(\mathbf{x'}, \mathbf{x}) \\ K(\mathbf{x'}, \mathbf{x}) & K(\mathbf{x'}, \mathbf{x'}) \end{pmatrix}$.

Then, using the equations for conditional means and variances for multivariate normal distributions mentioned previously, we draw functions from our new multivariate normal to get GP posteriors. Figure 5 shows 100 GP posteriors that incorporate each of the data points and have 95% uncertainty bounds that increase the further away a point is from data.

Thus, we have created a very rough model taking into account our data and uncertainty using Gaussian processes. We squeezed the functions together where we have data, and have some estimate for uncertainty everywhere else. Our rough model tells us where we have a

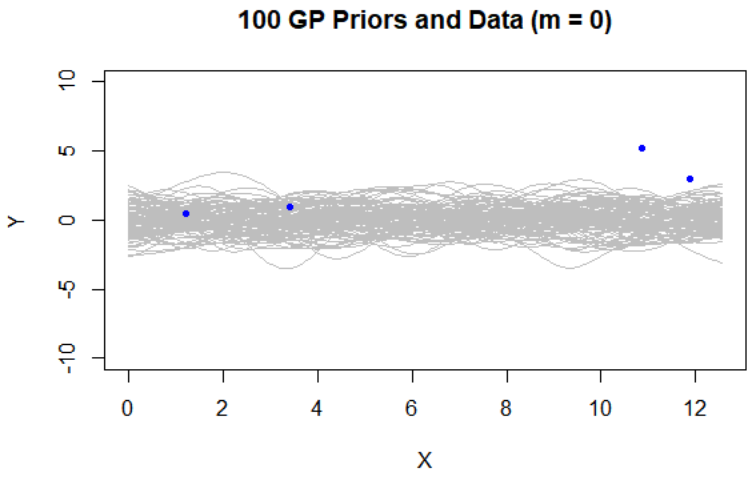



Figure 4: Sample data with previous GP prior.

lot of uncertainty, so we have an idea where to perform experiments or take measurements next.

Figure 6 shows the effect on our model as we obtain new data. Seeing a large gap and high uncertainty between $3 < x < 11$, we decide to gather data at $x = 7$ and repeat the process to create posterior GP's and create a model. This alters the model to what is shown in Figure 6 (top right). As we gather data, we update our model, and update the uncertainty of our model. With enough data points, we are able to get a much better estimate of the underlying function which generated these points (Figure 6 bottom right, Figure 7).

For instructional purposes, we sampled from $f(x) = \frac{x}{2}sin(\frac{3x}{4})$ to create our data. We show the true function as the blue line going through the blue points, our "data". With 9 points, we were able to model our true function with tight 95% uncertainty bounds pretty well, except for the last bit between $11 < x < 12$. A benefit of Gaussian process model is that predictions come with 95% confidence bounds where there is no data.

# 4 Tuning Parameters

In this section, we look at the effect of tuning parameters on GP models using the squared exponential kernel. A kernel $k(x, x')$ is a measure of similarity or closeness between points $x$ and $x'$ using some metric. Throughout our paper, we have used the squared exponential kernel, which we define below.

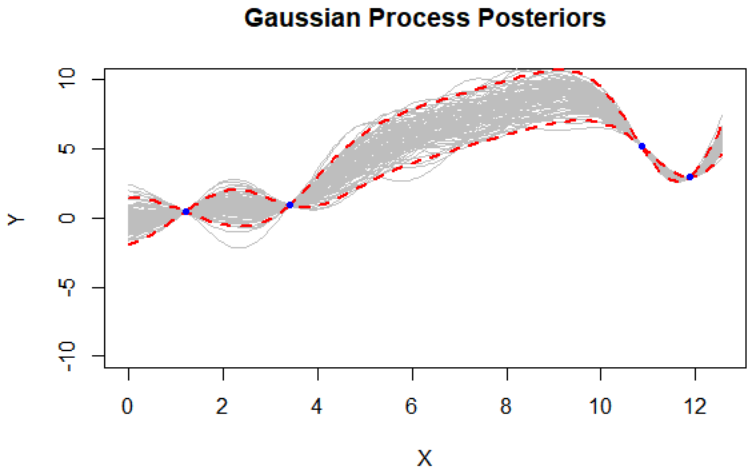



Figure 5: Posterior GP fitted to data.

$$k(x, x') = \tau^2 exp\left(-\frac{|x - x'|^2}{2l^2}\right).$$

This kernel has two tuning parameters: $\tau$ affects the amplitude of the Gaussian processes, and $l$ affects its wiggliness (length scale). Figure 8 shows the affect of different choices of parameters, with $\tau = 1, l = 1$ being the choice of parameters we've used throughout this paper in the top left.

These are four potential models, we could have ended up with any of these depending on our choice of tuning parameters. Although the model where $\tau = 1/2, l = 1$ has tight bounds and seems to model the function correctly, in application we would not know the true function and wider bounds might be desired. We also see that a choice of $l = 1/2$ is not a good choice whatsoever; the function we are trying to model does not lend itself well to the smaller length scale parameter.

## 5 R Code Example

We provide a short snippet of R code from our gaussian_process.R to demonstrate how GP models can be implemented. The function *kov*() computes the squared exponential distance between any two pairs of points X and X1. We use the *kov*() function extensively in the *gaussianProcess*() function to build the blocks for a new covariance matrix S that incorporates data and the specified priors. The *gaussianProcess*() function utilizes the equations for conditional expectations and covariances of Gaussian distributions to create

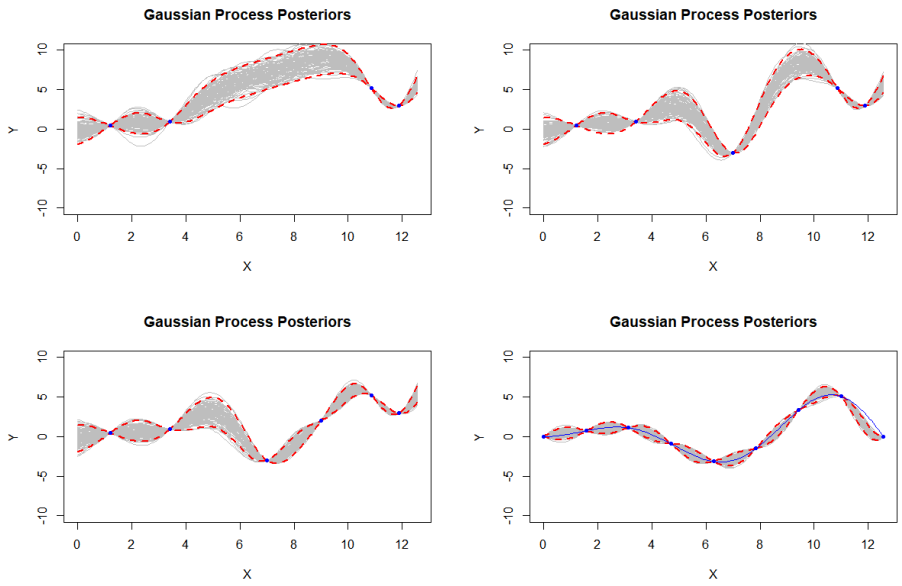



Figure 6: Posterior GP fitted to additional data.

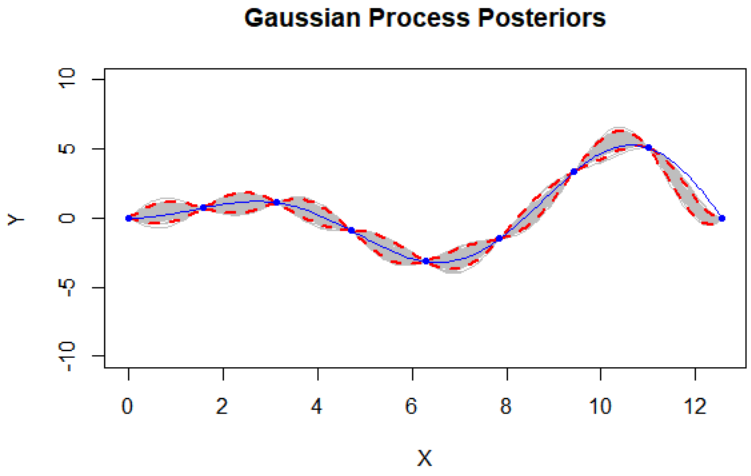



Figure 7: Posterior GP with 9 samples roughly estimates true function.

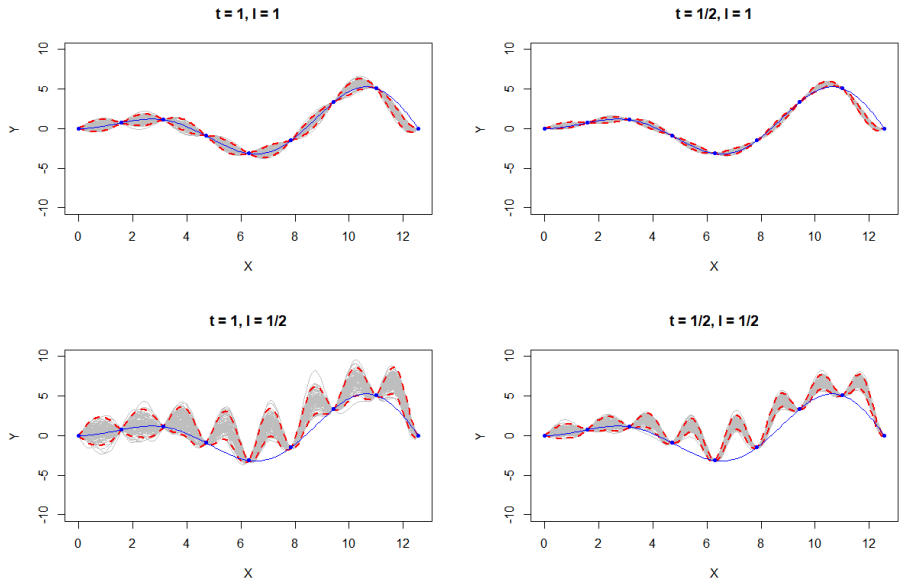



Figure 8: Effect of various choices of tuning parameters.

the parameters of a multivariate normal distribution conditioned on the data, and then draws functions from that distribution.

```
kov = function(X, X1 = X, t = 1, l = 1){
# covariance/kernel function
# t: amplitude parameter
# l: width parameter
d = distance(X, X1)
k = t^2 * exp(-d/(2*l^2))
k
}


gaussianProcess = function(X, data, t, l, n_gp){
# n_gp: number of GP's to generate
# calculating parameters of GPs
x_data = data$x
y_data = data$y
SXX = kov(X, t = t, l = l) # NOTE: UPPERCASE x
Sxx = kov(x_data, t = t, l = l) # note: lowercase x
SXx = kov(X, x_data, t = t, l = l)
```

```
Sxx_inv = solve(Sxx)
# mu = SXx %*% Sxx_inv %*% y_data # mu vector with data
mu = X + SXx %*% Sxx_inv %*% (y_data-x_data) # mu vector with data
S = SXX - SXx %*% Sxx_inv %*% t(SXx) # sigma matrix w/ data

# Gaussian Processes posterior
Y_gp = rmvnorm(n_gp, mu, S)
gp = list(Y = Y_gp, m = mu, S = S)
gp
}
```

# 6 Example

One application of Gaussian Processes comes from from analyzing for patterns in birthday frequencies from a birthday dataset. The dataset contains records of birthdays on each day from 1969-1988. Gaussian Processes are components of a larger model that look at factors such as special days in the year including holidays like Thanksgiving or Christmas, day of the week effects, seasonal patterns during the year, and long term effects. The model comes from Chapter 21 of our textbook "Bayesian Data Analysis" 3rd edition by Gellman, Carlin, et.al. Based on the knowledge of the calendar we use an additive model where:

$$y_t(t) = f_1(t) + f_2(t) + f_3(t) + f_4(t)) + f_5(t) + \epsilon_t$$

$t$ is the time in days starting with $t = 1$ for January 1st, 1969 and the different terms represent variation with different scales and periodicity.

1. Long term trends modeled by Gaussian Processes with squared exponential covariance function

$$f_1(t) \sim GP(0, k_1), \quad k_1(t, t') = \sigma_1^2 \exp\left(-\frac{|t - t'|^2}{2l_1^2}\right)$$

2. Short term trends modeled by Gaussian Processes with squared exponential covariance function with different amplitude and scale

$$f_2(t) \sim GP(0, k_2), \quad k_2(t, t') = \sigma_2^2 \exp\left(-\frac{|t - t'|^2}{2l_2^2}\right)$$

3. Weekly quasi-periodic pattern(allowed to change over time) modeled as product of periodic and squared exponential covariance function

$$f_3(t) \sim GP(0, k_3) \quad k_3(t, t') = \sigma_3^2 \exp\left(-\frac{2\sin^2(\pi(t - t')/7}{l_{3,1}^2}\right) \exp\left(-\frac{|t - t'|^2}{2l_{3,2}^2}\right)$$

4. Yearly smooth seasonal pattern modeled using product of periodic and squared exponential covariance function. 365.25 is the period and matches average length of year

$$f_4(t) \sim GP(0, k_4) \quad k_4(s, s') = \sigma_4^2 \exp\left(-\frac{2\sin^2(\pi(s-s')/365.25}{l_{4,1}^2}\right) \exp\left(-\frac{|s-s'|^2}{2l_{4,2}^2}\right)$$

where $s = s(t) = t \mod 365.25$. Using 365.25 allows us to align with the calendar every 4 years accounting for Leap Day.

5. Special days include interaction term with weekend. Based on prior knowledge inspection of data we choose: New Year's, Valentine's, Leap Day, April Fool's, Independence Day, Halloween, Christmas, and days between Christmas and New Years

$$f_5(t) = I_{special\ day}(t)\beta_\alpha + I_{weekend}(t) \cdot I_{special\ day}(t)\beta_b$$

$I_{specialday}$ is the vector of 13 indicator variables corresponding to each special day. The 13 special days include New Year's Day, Valentine's Day, Leap Day, April Fool's Day, Independence Day, Halloween, Christmas and the week between Christmas and New Year's. $I_{weekend}$ is an indicator equal to 1 if day is a weekend and 0 if otherwise. $\beta_\alpha$ and $\beta_b$ are both vectors of length 13 which have corresponding effects of each special day for weekdays and weekends respectively

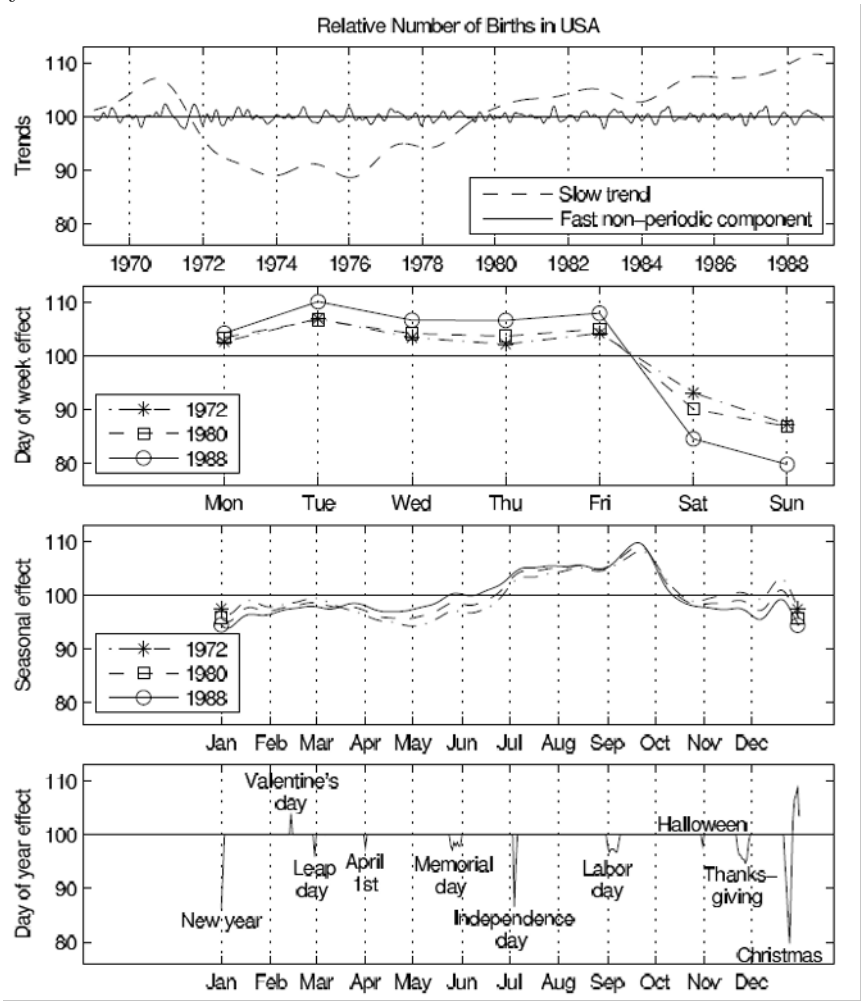
6. Unstructured residuals

$$\epsilon \sim N(0, \sigma^2)$$

We can use an additive model because the sum of Gaussian processes is also a Gaussian process and the kernel or covariance function is just the sum of the kernel for each component.

$$k(t, t') = k_1(t, t') + k_2(t, t') + k_3(t, t') + k_4(t, t') + k_5(t, t')$$

Plot shows slow trend, faster non periodic correlated variation, weekly trend and its change over time period of dataset, seasonal effect and its change over time, and day of the year effects:




In plot there are highest amount of births from August to September. This means there is seasonal effect has inverse relationship with temperature 9 months ago when conception occurs. This is because people stay inside more during colder weather. There are a smaller number of births on the weekends and more on weekdays and this trend increases over time. There are less births also on holidays except Valentine's day. Both these trends can be explained by c-sections and induced births. People want births on weekdays as they think the best doctors only work on the weekdays and want to avoid their kids having the same birthday as a major holiday.

The major issue with the model is that we see the effect of special days but not the days before or after the birth where we should see a change in births mostly increases.

If births are avoided during special days, we would see more births before and after that special day.

In new model we have special effects for each day of the year so we are no longer looking at only special holidays. A short time-scale non periodic component is added and the yearly periodic components are slightly changed. The model comes from Chapter 21 of "Bayesian Data Analysis" by Gellman, Carlin, et.al.

New Model:

$$y(t) = f_1(t) + f_2(t) + f_3(t) + f_4(t) + f_5(t) + f_6(t) + f_7(t) + f_8(t) + \epsilon_t$$

$t$ is the time in days starting with $t = 1$ for January 1st, 1969 and the different terms represent variation with different scales and periodicity.

1. Long term trends modeled by Gaussian Processes with squared exponential covariance function

$$f_1(t) \sim GP(0, k_1), \quad k_1(t, t') = \sigma_1^2 \exp\left(-\frac{|t - t'|^2}{2l_1^2}\right)$$

2. Short term trends modeled by Gaussian Processes with squared exponential covariance function with different amplitude and scale

$$f_2(t) \sim GP(0, k_2), \quad k_2(t, t') = \sigma_2^2 \exp\left(-\frac{|t - t'|^2}{2l_2^2}\right)$$

3. Weekly quasi-periodic pattern(allowed to change over time) modeled as product of periodic and squared exponential covariance function

$$f_3(t) \sim GP(0, k_3) \quad k_3(t, t') = \sigma_3^2 \exp\left(-\frac{2\sin^2(\pi(t - t')/7}{l_{3,1}^2}\right) \exp\left(-\frac{|t - t'|^2}{2l_{3,2}^2}\right)$$

4. Yearly smooth seasonal pattern modeled using product of periodic and squared exponential covariance function.

$$f_4(t) \sim GP(0, k_4) \quad k_4(s, s') = \sigma_4^2 \exp\left(-\frac{2\sin^2(\pi(s - s')/365}{l_{4,1}^2}\right) \exp\left(-\frac{|s - s'|^2}{2l_{4,2}^2}\right)$$

where $s = s(t)$ is modified from original model. The time before and after Leap Day are each incremented by 0.5 so that 365 is the length of the year for all years including Leap year. This is an easier implementation of periodicity.

5. Yearly fast changing pattern for weekday(day of the year effect) using a periodic covariance function.

$$f_5(t) \sim GP(0, k_5), \quad k_5(s, s') = I_{\text{weekday}}(t, t')\sigma_5^2 \exp\left(-\frac{2\sin^2(\pi(s-s')/365)}{l_5^2}\right)$$

$I_{weekday}(t, t')$ is indicator variable equal to 1 if both $t$ and $t'$ are weekdays and 0 otherwise

6. Yearly fast changing pattern for weekend(day of the year effect) using a periodic covariance function.

$$f_6(t) \sim GP(0, k_6), \quad k_6(t, t') = I_{\text{weekend}}(t, t')\sigma_6^2 \exp\left(-\frac{2\sin^2(\pi(s-s')/365)}{l_6^2}\right)$$

$I_{weekend}(t, t')$ is indicator variable equal to 1 if both $t$ and $t'$ are weekdays and 0 otherwise

7. Effects of specials days that do not have constant dates from year to year

$$f_7(t) = I_{special\ day}(t)\beta$$

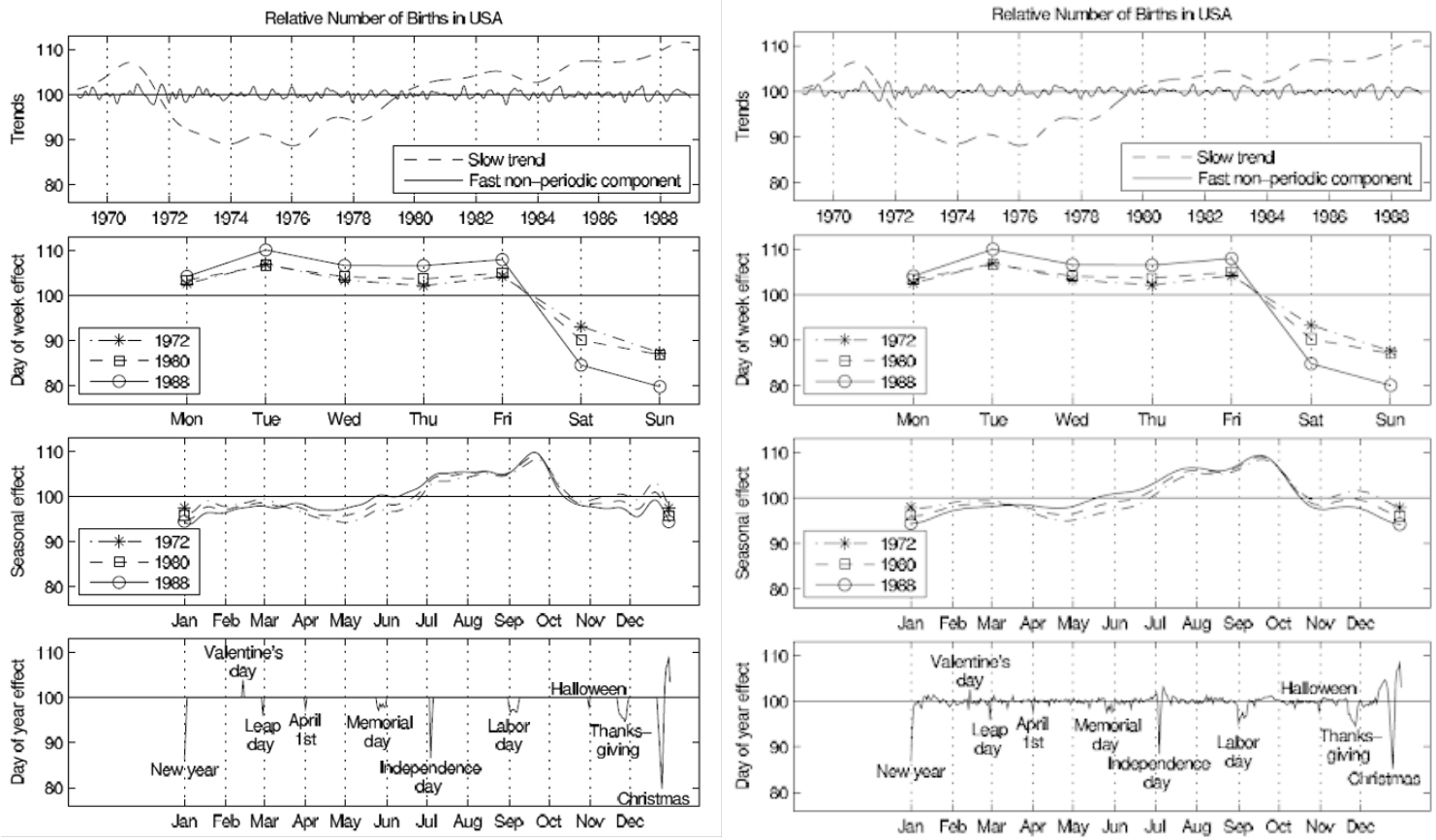
$I_{specialday}(t, t')$ is row vector of 4 indicator variables corresponding to 4 holidays

8. Short term variation modeled by Gaussian Processes with squared exponential covariance function

$$f_8(t) \sim GP(0, k_8), \quad k_8(t, t') = \sigma_2^2 \exp\left(-\frac{|t-t'|^2}{2l_8^2}\right)$$

9. Unstructured residuals

$$\epsilon \sim N(0, \sigma^2)$$




In new model we see fluctuations in the amount of births before and after special days as all days are accounted for in the day of year effect instead of just special days. This also means the seasonal effect graph has a smoother curve because it is no longer modeling for these days. It is only modeling for month of year as intended. This example shows how Gaussian Processes can be used as components of an additive model.

# 7 Conclusion

We have only scratched the surface of the topic of Gaussian processes. Since their origins in geo-statistics, GP have found numerous applications, including hyperparameter tuning and classification. They are a flexible non-parametric model that can be used to make predictions with uncertainty estimations.There are extensions that include modeling noise of multiple types on data, which we showed in the birthday example but did not elaborate on the process. The birthday example also shows how the sum of Gaussian processes is also a Gaussian process.

Despite their advantages, GP scale cubic with the number of training points. So we wouldn't want to use them for data sets with a large number of records, but they're perfect in a Bayesian setting with high-dimensional small data. Their ease of use and flexibility make them a valuable tool for the Bayesian statistician. We're glad to have researched this topic and look forward to using in our future work as statisticians.

# 8 References


1. Bayesian Data Analysis, Gellman, Carlin, et.al., Chapter 21
2. Gaussian Processes for Machine Learning, Rasmussen & Williams
3. Google search: Gaussian Process
    - https://www.cs.ubc.ca/ nando/540-2013/lectures/l6.pdf
    - https://bookdown.org/rbg/surrogates/chap5.html
4. YouTube search: Gaussian Process
    - Kilian Weinberger
    - Nando de Freitas
    - Pascal Poupart
5. Wikipedia: Gaussian process, Kriging

# 9 Appendix: R Code

```
#### Cesar Pardede ####
# Final Project
# Math 538
# 2020-12-01

library(plgp) # distance() - creates symmetric distance matrix
library(mvtnorm) # rmvnorm() - samples from MVN distn


f = function(x){
# true function of x
f = x/2*sin(3*x/4)
f
}


kov = function(X, X1 = X, t = 1, l = 1){
# covariance/kernel function
# t: amplitude parameter
# l: width parameter
d = distance(X, X1)
k = t^2 * exp(-d/(2*l^2))
k
}


gaussianProcess = function(X, data, t, l, n_gp){
# n_gp: number of GP's to generate
# calculating parameters of GPs
x_data = data$x
y_data = data$y
SXX = kov(X, t = t, l = l) # NOTE: UPPERCASE x
Sxx = kov(x_data, t = t, l = l) # note: lowercase x
SXx = kov(X, x_data, t = t, l = l)
Sxx_inv = solve(Sxx)
# mu = SXx %*% Sxx_inv %*% y_data # mu vector with data
mu = X + SXx %*% Sxx_inv %*% (y_data-x_data) # mu vector with data
S = SXX - SXx %*% Sxx_inv %*% t(SXx) # sigma matrix w/ data
```

```
# Gaussian Processes posterior
Y_gp = rmvnorm(n_gp, mu, S)
gp = list(Y = Y_gp, m = mu, S = S)
gp
}


gp_plotter = function(X, data, title = 'Gaussian Process Posteriors', t = 1, l = 1, n_gp =
# plotting gp's
x_data = data$x
y_data = data$y
gp = gaussianProcess(X, data, t, l, n_gp)
Y_gp = gp$Y
mu = gp$m
S = gp$S

plot(1, type = 'n', xlim = c(x_a, x_b), ylim = c(-10, 10), xlab = 'X', ylab = 'Y', main = t
for (i in 1:n_gp){
lines(X, Y_gp[i, ], col = 8)
}

# 95% bounds
var = diag(S)
var[which(var < 0)] = 0
sd = sqrt(var)
q_lower = mu + qnorm(0.025, 0, sd)
q_upper = mu + qnorm(0.975, 0, sd)
lines(X, q_lower, lty = 2, col = 2, lwd = 2)
lines(X, q_upper, lty = 2, col = 2, lwd = 2)

# plotting true function
plot_truth_and_data(data, fp, dp)
}


plot_truth_and_data = function(data, func_plot = T, data_plot = T, new_plot = F){
x_data = data$x
y_data = data$y
if (new_plot){plot(1, type = 'n', xlim = c(x_a, x_b), ylim = c(-10, 10), xlab = 'X', ylab =
if (func_plot){lines(X, Y, col = 4)}
if (data_plot){points(x_data, y_data, pch = 20, col = 4)}
```

```
}


gp_priors = function(n_prior, title = 'Gaussian Process Priors'){
# Gaussian Process priors
SXX = kov(X)
plot(1, type = 'n', xlim = c(x_a, x_b), ylim = c(-10, 10), xlab = 'X', ylab = 'Y', main = t
for (i in 1:n_prior){
mu = rep(0, ncol(SXX))
# mu = -f(X)
Y_gp_prior = rmvnorm(1, mu, sigma = SXX)
lines(X, Y_gp_prior, col = 8)
}
}


# True function
n = 200 # length of grid/discretization/quantization, and dimension of MVN
x_a = 0; x_b = 4*pi
X = matrix(seq(x_a, x_b, length = n), ncol = 1)
Y = f(X)
DATA = data.frame(x = X, y = Y)

# plotting gp priors
# Gaussian Processes are n-variate MVN distributions
gp_priors(1, title = '1 GP Prior')
gp_priors(10, '10 GP Priors')
gp_priors(100, '100 GP Priors')

# define "data"
n_data = 4 # 15 is max number of x's before numerical instability
set.seed(538)
x_data = as.matrix(sample(X, n_data)) # random data x
# x_data = as.matrix(seq(x_a, x_b, length = n_data)) # uniform data x
y_data = f(x_data)
data = data.frame(x = x_data, y = y_data)
data0 = data

# Posterior GP conditioned on data
plot_truth_and_data(data, new_plot = T, data_plot = T, func_plot = F)
gp_plotter(X, data, fp = F, dp = T)
```

```
# measure new data based on distribution of functions
x_new = 7
y_obs = f(x_new)
data_new = data.frame(x = x_new, y = y_obs)
data = rbind(data, data_new)
gp_plotter(X, data, fp = F, dp = T)

x_new = 9
y_obs = f(x_new)
data_new = data.frame(x = x_new, y = y_obs)
data = rbind(data, data_new)
gp_plotter(X, data, fp = F, dp = T)

x_new = seq(x_a, x_b, length = 9)
y_obs = f(x_new)
data_new = data.frame(x = x_new, y = y_obs)
gp_plotter(X, data_new, fp = F, dp = T)

# plot true function
plot_truth_and_data(DATA, func_plot = T, data_plot = F)

# plot comparisons of tuning parameters
data = data_new
gp_plotter(X, data, t = 1, l = 1, title = 't = 1, l = 1')
gp_plotter(X, data, t = 1/2, l = 1, title = 't = 1/2, l = 1')
gp_plotter(X, data, t = 1, l = 1/2, title = 't = 1, l = 1/2')
gp_plotter(X, data, t = 1/2, l = 1/2, title = 't = 1/2, l = 1/2')

# additional plots for slides
plot(1, type = 'n', xlim = c(x_a, x_b), ylim = c(-10, 10), xlab = 'X', ylab = 'Y', main = '
for (i in 1:(length(X)/4)){
abline(v = X[4*i], lty = 1, col = 4)
points(X[4*i], 0, pch = 20, col = 2)
}

gp_priors(100, '100 GP Priors and Data (m = 0)')
plot_truth_and_data(data0, func_plot = F)
```