Gaussian Processes

General Principles

Through varying intercepts and slopes, we have seen how to quantify some of the unique features that generate variation across clusters and covariance among the observations within each cluster. But through the covariance matrix that is used to account for correlation between clusters, we are inherently assuming linear relationships between clusters. What if we want to model the relationship between two variables that are not linearly related? In this case, we can use a Gaussian Process (GP) to model the relationship between two variables. Basically, a GP is a varying-slope model with a covariance matrix where each element of the matrix is a kernel function .

Considerations

Caution

- To capture complex, non-linear relationships in data where the underlying function is smooth but has an unknown functional form, GPs use a kernel .
- GPs assume normally distributed errors and may not be appropriate for all types of noise.
- The choice of kernel hyperparameters can significantly impact results; thus, GPs require choosing an appropriate kernel function that captures the expected behavior of your data.
- Through kernel definition, we can incorporate domain knowledge.
- They scale poorly with dataset size ($O(n^3)$ complexity) due to matrix operations; thus, memory requirements can be substantial for large datasets, which has led to neural networks being used instead to resolve large non-linear problems.

Example

Below is an example code snippet demonstrating Gaussian Process regression using the Bayesian Inference (BI) package. Data consist of a continuous dependent variable (*total_tools*), representing the number of tools invented in the islands, and a continuous independent variable (*population*), representing the population of the islands. The goal is to estimate the effect of population on the total tools. We use the distance matrix of the islands for the kernel function in order to capture the spatial dependence of the relationship. This example is based on McElreath (2018).

Python

```
from BI import bi
import jax.numpy as jnp
import numpyro
# Setup device--------------------------------------------------
m = bi(platform='cpu')

# Import Data & Data Manipulation ------------------------------------------------
# Import
from importlib.resources import files
data_path = files('BI.resources.data') / 'Kline2.csv'
m.data(data_path, sep=';')

data_path2 = files('BI.resources.data') / 'Kline2.csv'
islandsDistMatrix = pd.read_csv(data_path2, index_col=0)

m.data_to_model(['total_tools', 'population'])
m.data_on_model["society"] = jnp.arange(0,10)# index observations
m.data_on_model["Dmat"] = islandsDistMatrix.values # Distance matrix


def model(Dmat, population, society, total_tools):
    a = m.dist.exponential(1, name = 'a')
    b = m.dist.exponential(1, name = 'b')
    g = m.dist.exponential(1, name = 'g')

    # non-centered Gaussian Process prior
    etasq = m.dist.exponential(2, name = 'etasq')
    rhosq = m.dist.exponential(0.5, name = 'rhosq')
    SIGMA = cov_GPL2(Dmat, etasq, rhosq, 0.01)
```

```
    k = m.dist.multivariatenormal(0, SIGMA, name = 'k')
    #k = m.gaussian.gaussian_process(Dmat, etasq, rhosq, 0.01, shape = (10,))
    k = m.gaussian.kernel_L2(Dmat, etasq, rhosq, 0.01)
    lambda_ = a * population**b / g * jnp.exp(k[society])

    m.dist.poisson(lambda_, obs=total_tools)

# Run sampler ----------------------------------------------
m.fit(model)
m.summary()
```

R

```
library(BI)
pd=import('pandas')
# setup platform-------------------------------------------------
m=importbi(platform='cpu')

# Import data ----------------------------------------------
m$data(paste(system.file(package = "BI"),"/data/Kline2.csv", sep = ''), sep=';')
islandsDistMatrix = pd$read_csv(paste(system.file(package = "BI"),"/data/islandsDistMatrix.cs
m$data_to_model(list('total_tools', 'population'))
m$data_on_model$society = jnp$arange(0,10, dtype='int64')
m$data_on_model$Dmat = jnp$array(islandsDistMatrix)


# Define model ---------------------------------------------
model <- function(Dmat, population, society, total_tools){
  a = bi.dist.exponential(1, name = 'a')
  b = bi.dist.exponential(1, name = 'b')
  g = bi.dist.exponential(1, name = 'g')

  # non-centered Gaussian Process prior
  etasq = bi.dist.exponential(2, name = 'etasq')
  rhosq = bi.dist.exponential(0.5, name = 'rhosq')
  z = bi.dist.normal(0,1, name = 'z', shape = c(10))
  r = m$kernel_sq_exp(Dmat, z, etasq, rhosq, 0.01)
  SIGMA = r[[1]]
  L_SIGMA = r[[2]]
  k = r[[3]]
  lambda_ = a * population**b / g * jnp$exp(k[society])
```

```
  m$poisson(lambda_, obs=total_tools)
}

# Run MCMC ------------------------------------------------
m$fit(model) # Optimize model parameters through MCMC sampling

# Summary -------------------------------------------------
m$summary() # Get posterior distribution
```

Mathematical Details

Formula

The following equation allows us to evaluate the relationship between the dependent variable Y and the independent variable X while incorporating a GP for variable Z:

$$Y_i = \alpha + \beta X_i + \gamma_{Z_i}$$

where: - Y_i is the i-th value for the dependent variable Y.

- α is the intercept term.
- β is the regression coefficient term.
- X_i is the i-th value for the independent variable X.
- γ_{Z_i} is the Gaussian process i-th value for the independent variable Z.

The GP γ_{Z_i} follows a multivariate normal distribution:

$$\begin{pmatrix} Z_1 \\ \vdots \\ Z_n \end{pmatrix} \sim \text{MVNormal} \left(\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, K \right)$$

where:

- $(Z_1, ..., Z_n)$ represents a collection of all values of the random variable Z.
- $(0, ..., 0)$ represents the mean vector of the multivariate normal distribution of the same size as the number of random variables and set to zero .
- K is the covariance matrix of the random variable Z. Each element K_{ij} of the matrix is given by the kernel function evaluated at the corresponding points: $K_{ij} = k(Z_i, Z_j)$

$$K = \begin{pmatrix} k(Z_1, Z_1) & k(Z_1, Z_2) & \cdots & k(Z_1, Z_n) \\ k(Z_2, Z_1) & k(Z_2, Z_2) & \cdots & k(Z_2, Z_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(Z_n, Z_1) & k(Z_n, Z_2) & \cdots & k(Z_n, Z_n) \end{pmatrix}$$

- Multiple kernel functions exist and will be discussed in the Note(s) section. But the most common one is the quadratic kernel:

$$K_{ij} = \eta^2 \exp(-p^2 D_{ij}^2) + \delta_{ij}\sigma^2$$

Where:

- η is the signal variance, representing the overall variance of the outputs of the Gaussian process. It scales the influence of the kernel function. A larger η^2 indicates a wider range of values the function can take.
- p determines the rate of decline.
- D_{ij} is the distance between the i-th and j-th points.
- δ_{ij} is the Kronecker delta, taking a value of one when $i = j$ and zero otherwise, allowing the self-covariance to be included in the calculation.
- σ^2 is the noise variance, which accounts for the observation noise in the data. It represents the uncertainty or variability in the measurements or outputs at each point. The term effectively adds this noise variance only when $i = j$, ensuring that the diagonal elements of the covariance matrix represent the total variance at each input point.

Bayesian model

In the Bayesian formulation, we define each parameter with priors . We can express a Bayesian version of this GP using the following model:

$$Y_i = \alpha + \beta X_i + \gamma_{Z_i}$$

$$\gamma \sim \text{MVNormal}\left(\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, K\right)$$

$$K_{ij} = \eta^2 \exp(-p^2 D_{ij}^2) + \delta_{ij}\sigma^2$$

$$\alpha \sim \text{Normal}(0, 1)$$

$$\eta^2 \sim \text{HalfCauchy}(0, 1)$$

$$p^2 \sim \text{HalfCauchy}(0, 1)$$

where:

- Y_i is the i-th value for the dependent variable Y.
- α is the intercept term with a prior of Normal$(0, 1)$.
- β is the regression coefficient term with a prior of Normal$(0, 1)$.
- X_i is the i-th value for the independent variable X.
- γ_{Z_i} is the Gaussian process i-th value for the independent variable Z.
- γ is the latent function modeled by the GP.
- K_{ij} is the kernel function evaluated at the corresponding points, $K_{ij} = k(Z_i, Z_j)$, with priors of HalfCauchy(0,1) for η^2 and p^2 to ensure positive values.

Notes

Note

Common kernel functions include:

- *Radial Basis Function* (RBF) or Squared Exponential Kernel:

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

- *Rational Quadratic Kernel*, this kernel is equivalent to adding together many RBF kernels with different length scales:

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

- *Periodic kernel* allows for modeling functions that repeat themselves exactly:

$$k(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi||x - x'||/p)}{l^2}\right)$$

- *Locally Periodic Kernel*:

$$k(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi||x - x'||/p)}{l^2}\right) \exp\left(-\frac{||x - x'||^2}{2l^2}\right)$$

- GPs can be extended to classification problems using link functions.
- Multi-output problems can be addressed using matrix-valued kernels.
- Deep learning can be combined with GPs through Deep Kernel Learning.
- Computational tricks for large datasets include:
  - Sparse approximations (e.g., FITC, VFE)
  - Inducing points methods
  - Random Fourier features

Reference(s)

McElreath, Richard. 2018. *Statistical Rethinking: A Bayesian Course with Examples in r and Stan*. Chapman; Hall/CRC.

https://www.cs.toronto.edu/~duvenaud/cookbook/