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.

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 .
- 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

```python
from BI import bi, jnp
import pandas as pd
# Setup device------------------------------------------------
m = bi(platform='cpu')

# Import Data & Data Manipulation ------------------------------------------------
# Import
from importlib.resources import files
data_path = m.load.kline2(only_path=True)
m.data(data_path, sep=';')


data_path2 = files('BI.Resources') / 'islandsDistMatrix.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 = etasq * jnp.exp(-rhosq * jnp.square(Dmat))
    SIGMA = SIGMA.at[jnp.diag_indices(Dmat.shape[0])].add(etasq)
    k = m.dist.multivariate_normal(0, SIGMA, name = 'k')

    lambda_ = a * population**b / g * jnp.exp(k[society])

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

# Run sampler ------------------------------------------------
```

```
m.fit(model)
m.summary()
```

```
jax.local_device_count 32
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]warmup:   0%|          | 1/1000 [00:00<12:52,  1.29i
arviz - WARNING - Shape validation failed: input_shape: (1, 500), minimum_shape: (chains=2, c
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	1.39	1.01	0.17	2.81	0.07	0.05	133.98	214.31	NaN
b	0.29	0.08	0.17	0.41	0.01	0.00	184.17	119.24	NaN
etasq	0.08	0.07	0.01	0.15	0.01	0.01	136.57	149.24	NaN
g	0.64	0.59	0.02	1.31	0.03	0.05	194.77	235.45	NaN
k[0]	-0.21	0.30	-0.64	0.24	0.02	0.03	170.75	120.39	NaN
k[1]	0.03	0.28	-0.34	0.48	0.02	0.02	166.01	144.33	NaN
k[2]	-0.07	0.27	-0.40	0.39	0.03	0.03	115.89	108.80	NaN
k[3]	0.33	0.23	-0.01	0.74	0.02	0.02	136.59	96.46	NaN
k[4]	0.02	0.24	-0.34	0.33	0.02	0.03	118.65	96.09	NaN
k[5]	-0.41	0.28	-0.78	-0.00	0.03	0.03	120.78	94.91	NaN
k[6]	0.11	0.23	-0.23	0.42	0.02	0.03	105.71	78.35	NaN
k[7]	-0.24	0.25	-0.64	0.11	0.03	0.03	109.64	71.99	NaN
k[8]	0.24	0.24	-0.07	0.61	0.03	0.03	97.87	68.36	NaN
k[9]	-0.23	0.34	-0.71	0.26	0.04	0.04	103.11	73.45	NaN
rhosq	1.69	1.65	0.01	3.62	0.08	0.11	246.84	201.20	NaN

Python (Build in function)

```
from BI import bi, jnp
import pandas as pd
# Setup device------------------------------------------------
m = bi(platform='cpu')

# Import Data & Data Manipulation ------------------------------------------------
# Import
from importlib.resources import files
data_path = m.load.kline2(only_path=True)
m.data(data_path, sep=';')
```

```
islandsDistMatrix = m.load.islands_dist_matrix(frame = False)['data']

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

    k = m.gaussian.gaussian_process(Dmat)

    lambda_ = a * population**b / g * jnp.exp(k[society])

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

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

```
jax.local_device_count 32

  0%|          | 0/1000 [00:00<?, ?it/s]warmup:   0%|          | 1/1000 [00:00<10:36,  1.57i
arviz - WARNING - Shape validation failed: input_shape: (1, 500), minimum_shape: (chains=2, 
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	1.42	1.11	0.07	2.72	0.07	0.09	162.72	159.21	NaN
b	0.28	0.08	0.17	0.44	0.01	0.00	80.44	174.63	NaN
etasq	0.20	0.18	0.01	0.39	0.01	0.02	145.06	171.68	NaN
g	0.65	0.66	0.02	1.35	0.07	0.06	70.85	107.84	NaN
kernel[0]	-0.17	0.33	-0.74	0.27	0.04	0.03	54.29	92.78	NaN
kernel[1]	-0.02	0.32	-0.54	0.48	0.05	0.03	48.22	79.88	NaN
kernel[2]	-0.07	0.32	-0.48	0.50	0.05	0.04	51.01	55.60	NaN
kernel[3]	0.35	0.28	-0.10	0.78	0.04	0.03	51.18	83.79	NaN
kernel[4]	0.07	0.29	-0.35	0.50	0.04	0.03	53.67	57.59	NaN
kernel[5]	-0.40	0.29	-0.80	0.05	0.04	0.04	59.51	68.94	NaN
kernel[6]	0.12	0.29	-0.21	0.60	0.04	0.04	51.47	63.64	NaN
kernel[7]	-0.22	0.29	-0.59	0.27	0.04	0.04	55.68	76.35	NaN

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
kernel[8]	0.26	0.28	-0.17	0.67	0.04	0.03	57.48	50.38	NaN
kernel[9]	-0.19	0.38	-0.76	0.39	0.04	0.04	81.79	95.49	NaN
rhosq	1.38	1.73	0.03	3.37	0.12	0.13	134.65	216.24	NaN

R

```r
library(BayesianInference)
jnp = reticulate::import('jax.numpy')
pd = reticulate::import('pandas')
# setup platform------------------------------------------------
m=importBI(platform='cpu')

# Import data ------------------------------------------------
m$data(m$load$kline2(only_path=T), sep=';')
islandsDistMatrix = m$load$islands_dist_matrix(frame = FALSE)$data
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')
  k = m$gaussian$gaussian_process(Dmat, etasq, rhosq, 0.01)

  lambda_ = a * population**b / g * jnp$exp(k[society])
  m$dist$poisson(lambda_, obs=total_tools)
}

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

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

Julia

```
using BayesianInference

# Setup device------------------------------------------------
m = importBI(platform="cpu")

# Import Data & Data Manipulation ------------------------------------------------
# Import
data_path = m.load.kline2(only_path = true)
m.data(data_path, sep=";")

islandsDistMatrix = m.load.islands_dist_matrix(frame = false)["data"]
m.data_to_model(["total_tools", "population"])
m.data_on_model["society"] = jnp.arange(0,10)# index observations
m.data_on_model["Dmat"] = jnp.array(islandsDistMatrix) # Distance matrix



# Define model ------------------------------------------------
@BI function 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 = etasq * jnp.exp(-rhosq * jnp.square(Dmat))
    SIGMA = SIGMA.at[jnp.diag_indices(Dmat.shape[0])].add(etasq)
    k = m.dist.multivariate_normal(0, SIGMA, name = "k")

    lambda_ = a * population^b / g * jnp.exp(k[society])

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

end
```

```
# Run mcmc ------------------------------------------------
m.fit(model)  # Optimize model parameters through MCMC sampling

# Summary -------------------------------------------------
m.summary() # Get posterior distributions
```

Mathematical Details

Formula

The following equation allows us to evaluate the relationship between the dependent variable Y distributed normal, and the independent variable X while incorporating a GP for the effect of variable Q:

$$Y_{[i]} \sim \text{Normal}(\alpha + \beta X_{[i]} + \gamma_{[Q_{[i]}]}, \sigma)$$

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.
- $Q_{[i]}$ is an integer-valued independent variable (e.g., year-of-birth, age, year) for observation i.
- γ is a vector output from a Gaussian process:

$$\gamma \sim \text{MVNormal}\left(Z, \varsigma \Omega \varsigma\right)$$

where:

- Z represents the mean vector of the multivariate normal distribution and set to zero .
- ς is a diagonal matrix of standard deviations.
- Ω is a correlation matrix.
- Multiple kernel functions for Ω exist and will be discussed in the Note(s) section. But the most common one is the quadratic kernel:

$$\Omega_{[i,j]} = \eta \exp(-\phi^2 D^2_{[i,j]})$$

Where:

- η is the maximal correlation.
- ϕ determines the rate of decline.
- $D_{[i,j]}$ is the distance between the i-th and j-th categories.

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:

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

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

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

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

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

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

where:

- Y_i is the i-th value for the dependent variable Y.
- α is the intercept term with a prior of $\text{Normal}(0, 1)$.
- β is the regression coefficient term with a prior of $\text{Normal}(0, 1)$.
- 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*:

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

- Any slope or intercept in your model can be defined using a Gaussian Process.

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/