Lecture 6 Gaussian Process

Shiwei Lan[1]

[1]School of Mathematical and Statistical Sciences
Arizona State University

STP598 Machine Learning and Deep Learning
Fall 2021

Gaussian Process Modeling Function Data

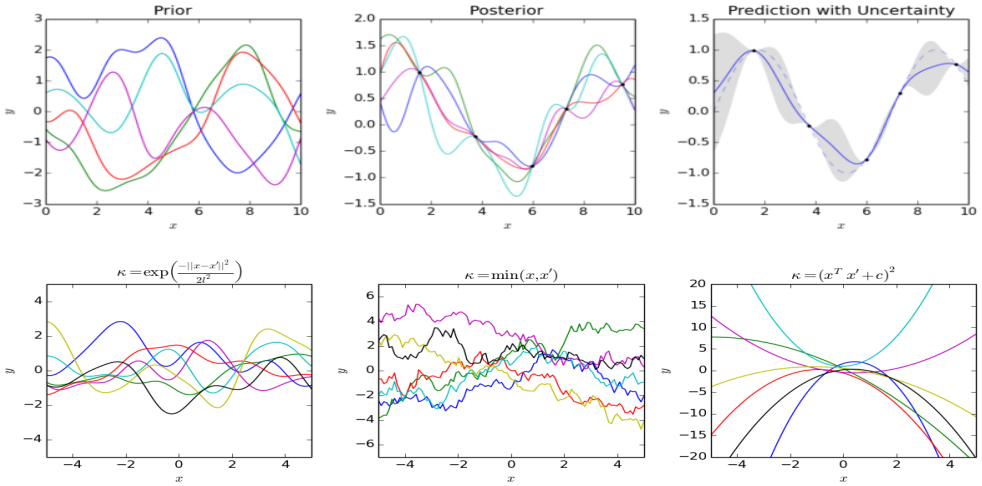


Figure: Top: Gaussian Process Regression; Bottom: GP Priors.

Table of Contents

Introduction

- A Gaussian process (GP) on the real line is a random real-valued function $y(t)$, which is completely determined by its mean function $\mathrm{E}y(s)$ and covariance function (kernel) $C_{st} = \mathrm{Cov}(y(s), y(t))$.
- A finite sample $(y(t_1), \ldots, y(t_n))$ has a multivariate Gaussian distribution with mean $(\mathrm{E}y(t_1), \ldots, \mathrm{E}y(t_n))$, and covariance matrix $(C_{t_i t_j})$.
- Note that we are limited to kernels providing positive semi-definite covariance matrices.
- In this lecture, we discuss Gaussian process models for regression and classification, so the process is indexed by a set of predictors x.
- In this case, Gaussian process is used as a distribution over functions $y(x)$.
- We usually add an extra parameter to account for observation noise.

Gaussian process models

- To introduce this concept, we start with a simple linear regression model.
- Recall that we presented a linear regression model as

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... \beta_p x_{ip} + \varepsilon_i$$

- Using normal priors (with mean zero, and in general, different variances) for β's

$$\beta_j | \sigma_j \sim N(0, \sigma_j^2) \qquad j = 0, ..., p$$

Gaussian process models

- In prior, β has a $(p+1)$ dimensional multivariate normal distribution

$$\beta|\Sigma_\beta \sim N(0, \Sigma_\beta)$$

- ε also has an n dimensional multivariate normal distribution

$$\varepsilon|\Sigma_\varepsilon \sim N(0, \Sigma_\varepsilon)$$

- To obtain the distribution of y we multiply β by the matrix x and add ε to it.
- Based on the properties of multivariate normal distribution, the resulting distribution would still be multivariate normal $N(0, C)$ where

$$C = x\Sigma_\beta x^T + \Sigma_\varepsilon$$

Gaussian process models

- This gives us the prior distribution on the function $y(x)$.
- Since any finite subset of $y(x)$ (e.g., for the n observed cases) would have a Gaussian distribution, the prior distribution on $y(x)$ is a *Gaussian process*.
- As mentioned above, analogous to Gaussian distributions, Gaussian processes are defined by their mean (here, the mean is 0 in prior) and covariance function C.
- For the above linear model, the elements of C are

$$C_{ij} = Cov(y_i, y_j) = \sigma_0^2 + \sum_{u=1}^{p} x_{iu} x_{ju} \sigma_u^2 + \delta_{ij} \sigma_\varepsilon^2$$

where δ_{ij} is equal to 1 if $i = j$, and 0 otherwise.

Gaussian process models

- Setting up the model this way, we are putting the prior directly on the relationship between x and y as opposed to on some parameters that represent this relationship (i.e., we cut out the middleman).
- This is specially useful if our objective is to predict future cases as opposed to making inference about the relationship between x and y.
- Note that the prior here is implicit and reflects our choice of the functional form.
- In the above example, we are assuming the relationship is linear. In general, we could use other covariance functions, C, to create nonlinear relationship.

Kernelization

- GP regression can be viewed as a generalization of linear regression

$$y_i = \sum_{d=1}^{D} \phi_d(\mathbf{x}_i)\beta_d + \varepsilon_i, \quad e.g.\, \phi_d(\mathbf{x}_i) = x_{id}$$
$$\beta_d \sim \mathcal{N}(\cdot\,|0, \lambda_d)$$
$$\varepsilon_i \sim \mathcal{N}(\cdot\,|0, \sigma^2)$$

- Integrating the coefficients β_d's (middleman) we get

$$y \sim \mathcal{GP}(0, K)$$
$$K(y_i, y_j) = \sum_{d=1}^{D} \phi_d(\mathbf{x}_i)\lambda_d\phi_d(\mathbf{x}_j) + \sigma^2\delta_{ij}$$

- Let $D \to +\infty$, this GP corresponds to infinitely many basis functions (infinite-dimensional feature spaces).

Gaussian Process: a formal definition

- A Gaussian process defines a distribution over functions, $p(f)$, where $f : \mathcal{X} \to \mathbb{R}$ is a function mapping some input space $\mathcal{X}$ to $\mathbb{R}$.

Definition (Gaussian process)

$p(f)$ is a *Gaussian process* if for any finite subset $\{x_1, \cdots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset $p(\mathbf{f})$ with $\mathbf{f} := (f(x_1), \cdots, f(x_n))$ is *multivariate Gaussian*.

- Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, or kernel, $K(x, x')$

$$p(f(x), f(x')) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$\boldsymbol{\mu} = \begin{bmatrix} \mu(x) \\ \mu(x') \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} K(x,x) & K(x,x') \\ K(x',x) & K(x'x') \end{bmatrix}$$

Gaussian Process: a flexible modeling tool

- An exemplary covariance (kernel) function could be

$$K(x_i, x_j) = v_0 \exp\left\{-\left(\frac{|x_i - x_j|}{r}\right)^{\alpha}\right\} + v_1 + v_2\delta_{ij}$$

- The parameters $(v_0, v_1, v_2, r, \alpha)$ are interpretable and can be learned from data

- v_0 signal variance
- v_1 variance of bias
- v_2 noise variance
- r lengthscale
- α roughness

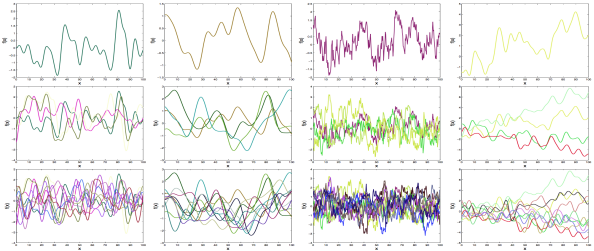


Table of Contents

Gaussian process for nonlinear regression

- For example, the following covariance function is very useful and includes a wide range of smooth nonlinear functions:

$$Cov(y_i, y_j) = \lambda^2 + \eta^2 \exp\Big(-\sum_{u=1}^{p} \rho_u^2 (x_{iu} - x_{ju})^2 \Big) + \delta_{ij}\sigma_\varepsilon^2$$

- The constant part is used to make sure the model fit functions where the mean of y is not zero (the x matrix does not have a vector of 1's anymore). However, it is better to center y before analysis so we don't have to use a large constant.
- There is one ρ for each predictor.
- The noise parameter, σ_ε^2 (also called *jitter*), accounts for random variations and is essential to improve computation.

The effect of parameters in the covariance function

- By using different η, ρ's, λ and σ_ε, we can generate a large variety of functions.

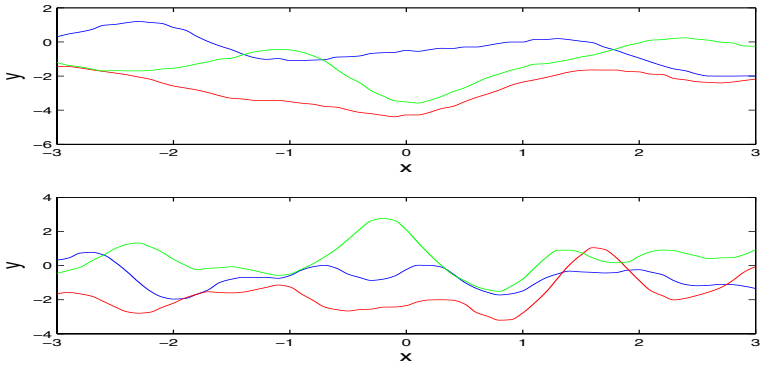


Figure: The top panel shows samples based on $\eta = 1$, $\rho = 1$, $\lambda = 1$, and $\sigma_\varepsilon = 0.01$. The bottom panel is base on the same priors except we set $\rho = 2$.

Prediction

- As mentioned above, using a Gaussian process prior is especially useful if our goal is predicting future cases for which we only know the value of predictors, $\tilde{x}$.
- Assume that we have observed (x, y) for n cases, and we want to predict $\tilde{y}$ for a new observation with predictor values $\tilde{x}$.
- Since the covariance function depends on x, we can find C_{n+1} for n the training cases and the new observation, i.e., for $\binom{x}{\tilde{x}}$. To avoid confusion we denote the covariance matrix for just the training cases as C_n.
- We can write down C_{n+1} as follows:

$$C_{n+1} = \begin{pmatrix} C_n & K \\ K^T & v \end{pmatrix}$$

where K is the $n \times 1$ covariance vector between $\tilde{y}$ and the n observed y. v is the prior variance of $\tilde{y}$ obtained based on the covariance function C.

Prediction

- Based the above setting, we can obtain the posterior predictive distribution for the new case.
- This distribution is also Gaussian with the following mean and variance:

$$\begin{aligned} E(\tilde{y}|y) &= K^T C_n^{-1} y \\ Var(\tilde{y}|y) &= v - K^T C_n^{-1} K \end{aligned}$$

- If we need a point estimate, we can use $E(\tilde{y}|y)$.

Gaussian Process: Regression

- GP can be used for nonlinear regression. Observe a data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)_{n=1}^N\} = (\mathbf{X}, \mathbf{y})$. We can model them

$$\begin{aligned} y_n &= f(\mathbf{x}_n) + \varepsilon_n \\ f &\sim \mathcal{GP}(\cdot|0, K) \\ \varepsilon_n &\sim \mathcal{N}(\cdot|0, \sigma^2) \end{aligned}$$

- The posterior on f is tractable

$$p(\mathbf{f}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$\boldsymbol{\mu} = (\mathbf{K}_N^{-1} + \sigma^{-2}\mathbf{I}_N)^{-1}(\sigma^{-2}\mathbf{y}), \quad \boldsymbol{\Sigma} = (\mathbf{K}_N^{-1} + \sigma^{-2}\mathbf{I}_N)^{-1}$$

- So is the predictive probability on new predictors $\mathbf{x}_*$

$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p(y_*|\mathbf{x}_*, \mathbf{f}, \mathcal{D}) p(\mathbf{f}|\mathcal{D}) d\mathbf{f} = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$
$$\boldsymbol{\mu}_* = \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2\mathbf{I}_N)^{-1}\mathbf{y}, \quad \boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2\mathbf{I}_N)^{-1}\mathbf{K}_{N*} + \sigma$$

Example

- The following example shows a Gaussian process model trained on 100 data points uniformly sampled from -3 to 3 with the following covariance function:

$$Cov(y_i, y_j) = 2 + \exp\left(-0.5(x_i - x_j)^2\right) + \delta_{ij} \times 0.1$$

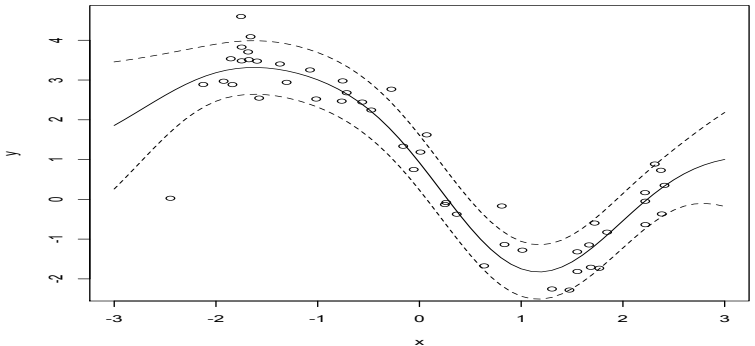


Figure: The solid line is expected function based on a grid test points between -3 and 3. The dashed lines show the 95% interval for predictions.

Hyperparameters

- In reality, we might not have enough information to fix the parameters of the covariance functions.
- In general, we would treat these parameters (e.g., η, ρ's, λ and σ_ε) as hyperparameters.
- Therefore, we need to use MCMC simulations in order to obtain samples from the posterior distributions of these hyperparameters, and as usual, we integrate over these posterior distributions to obtain the posterior predictive probabilities.
- The log-(marginal) likelihood function in this case is as follows:

$$\ell = \log P(y|X) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\det C - \frac{1}{2}y^T C^{-1} y, \quad C = K + \sigma^2 I$$

- Note that the computational cost of C^{-1} is in general $\mathcal{O}(n^3)$.

Example

- For the following example, η^2, ρ^2, λ^2, and σ^2 are hyperparameters with Gamma(1, 1) priors.

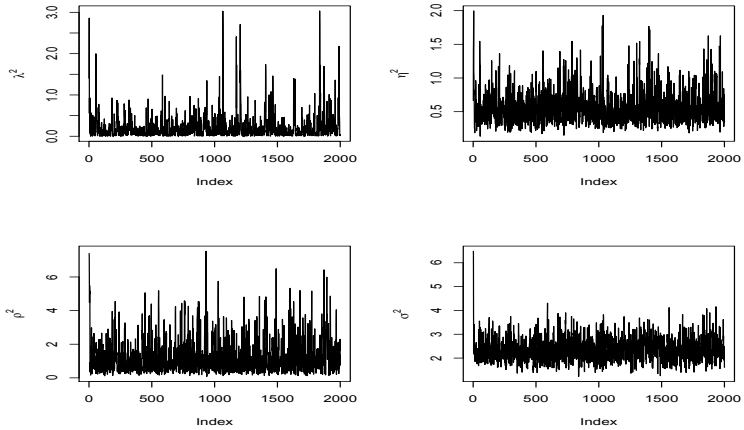
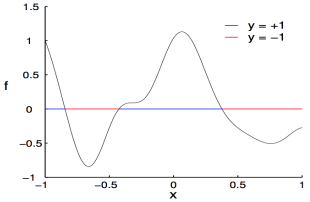
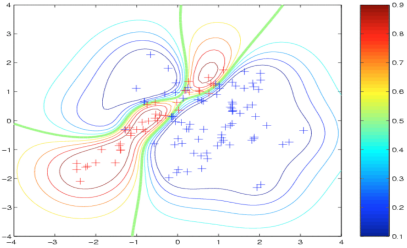


Table of Contents

Gaussian Process: Classification

- Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\} = (\mathbf{X}, \mathbf{y})$ with binary class labels $y_i \in \{\pm 1\}$, we want to infer the class label probabilities at new points.





- Many ways to relate function values (latent) $f_i = f(\mathbf{x}_i)$ to class probabilities

$$p(y_i|f_i) = \begin{cases} \frac{1}{1+\exp(-y_i f_i)}, & \text{sigmoid (logistic)} \\ \Phi(y_i f_i), & \text{cumulative normal (probit)} \\ H(y_i f_i), & \text{threshold} \\ \varepsilon + (1-\varepsilon) H(y_i f_i), & \text{robuts threshold} \end{cases}$$

Kernelization

- GP classification is related to kernel methods, e.g. kernelized *Support Vector Machines (SVM)*. Consider the soft-margin SVM:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i (1 - y_i f_i)_+$$

  where $(\cdot)_+$ is the hinge loss and $f_i = f(\mathbf{x}_i) = \mathbf{w}\cdot\mathbf{x}_i + w_0$.
- Let's kernelize this

$$\mathbf{x}_i \to \phi(\mathbf{x}_i) = k(\cdot, \mathbf{x}_i), \quad \mathbf{w} \to f(\cdot), \quad \langle k(\cdot, \mathbf{x}_i), f(\cdot)\rangle = f(\mathbf{x}_i)$$

- By representation theorem,

$$f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) \Longrightarrow \boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{f}$$

- Then the regularizer $\|\mathbf{w}\|^2 = \|f\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{K}\boldsymbol{\alpha}$ and the kernelized SVM loss is

$$\min_{\mathbf{f}} \frac{1}{2}\mathbf{f}^{\mathsf{T}}\mathbf{K}^{-1}\mathbf{f} + C\sum_i (1 - y_i f_i)_+$$

Gaussian Process: Classification

- Compare the kernelized SVM

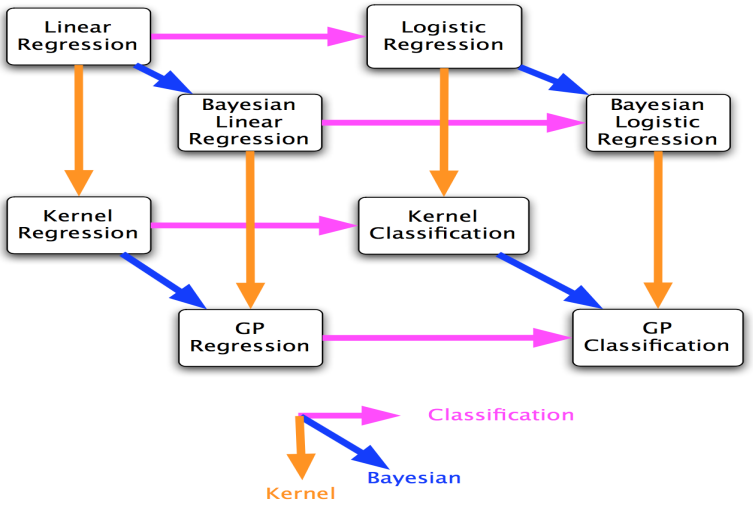
$$\min_{\mathbf{f}} \frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{f} + C\sum_i (1 - y_i f_i)_+$$

against GP classification

$$\frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{f} - \sum_i \log p(y_i|f_i) + c$$

- With GP, we can
  - Handle uncertainty in unknown function f by averaging, not minimization
  - Compute $p(y = +1|\mathbf{x}) \neq p(y = +1|\hat{\mathbf{f}}, \mathbf{x})$
  - Learn the kernel parameters automatically from data.
  - Learn the regularization parameter C without cross-validation.
  - Incorporate interpretable noise models and priors over functions.
  - Combine automatic feature selection with learning using ARD.

Gaussian Process: Kernelization (Zoubin Ghahramani)



Gaussian process models for classification

- For categorical outcome variables, we assume the Gaussian process prior over a continuous *latent function*, $u(x)$.
- We define the distribution of the response variable in terms of this latent function.
- For example, if the outcome variable y is binary, we can use the following logistic model:

$$P(y_i = 1|u(x_i)) = \frac{\exp(u(x_i))}{1 + \exp(u(x_i))}$$

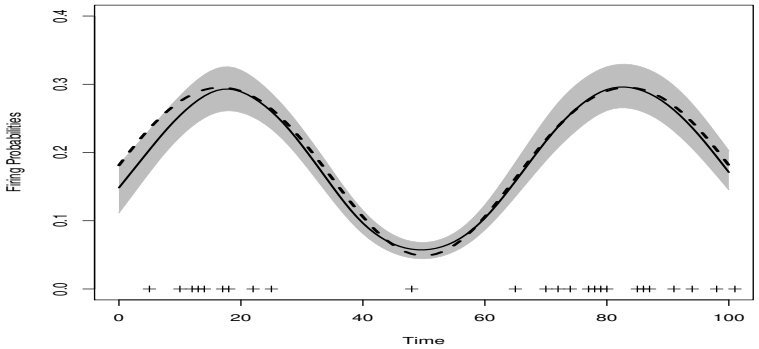
or alternatively,

$$P(y_i = 1|u(x_i)) = \frac{1}{1 + \exp(-u(x_i))}$$

- We can use a multinomial logit model for outcome variables with multiple categories.

Example

- Here, we are using a GP model to estimate the underlying firing rates of a neuron (i.e., $y_t = 1$ when the neuron fires, $y_t = 0$ otherwise).
- The dashed line shows the true firing probability and the plus signs show the firing time.



Gaussian Process Classification: multi-class

- Suppose we have the targets t falling into K classes $1, 2, \cdots, K$. We model those targets in a generalized expit function of latent variables $y_1, y_2, \cdots, y_K$ in the following form: for each observation i,

$$\Pr[t^{(i)} = k | y^{(i)}] = \frac{\exp(y_k^{(i)})}{\sum_{k'=1}^{K} \exp(y_{k'}^{(i)})}, \quad \text{for } k = 1, \cdots, K$$

- Then we model the prior of latent variables y_k as GP with mean zero and covariance in terms of covariates $\{x_u\}$:

$$\text{Cov}[y_k^{(i)}, y_k^{(j)}] = \lambda^2 + \sum_{u=1}^{p} \tau_u^2 x_u^{(i)} x_u^{(j)} + \eta^2 \exp\left(- \sum_{u=1}^{p} \rho_u^2 (x_u^{(i)} - x_u^{(j)})^2 \right) + \gamma^2 \delta_{ij}$$

for any fixed k and $y_k, y_{k'}$ are independent vectors for $k \neq k'$.

Gaussian Process Classification: multi-class

- Denote $\theta = (\lambda, \tau, \eta, \rho, \gamma)$ as our parameters. We put priors and hyper-priors

$$\begin{aligned} \theta_h|\mu_h, \sigma_h^2 &\sim N(\mu_h, \sigma_h^2),\ h = 1, \cdots, 5 \\ \mu_h|M, V &\sim N(M, V) \\ \sigma_h^2|\alpha, \beta &\sim \text{Inv-Gamma}(\alpha, \beta) \end{aligned}$$

- The log-likelihood and its gradient can be calculated

$$\begin{aligned} l &= -\frac{K}{2}\log\det C - \frac{1}{2}\sum_{k=1}^{K} y_k^T C^{-1} y_k \\ \frac{\partial l}{\partial \theta_h} &= -\frac{K}{2}\text{tr}\left(C^{-1}\frac{\partial C}{\partial \theta_h}\right) + \frac{1}{2}\sum_{k=1}^{K} y_k^T C^{-1}\frac{\partial C}{\partial \theta_h}C^{-1}y_k \end{aligned}$$

Gaussian Process Classification: multi-class

- For fixed k, the potential energy $U_k(y)$ of y_k is

$$U_k(y) = -\log(P(y_k|t,\theta)) = -\sum_{i:t^{(i)}=k} y_k^{(i)} + \sum_{i=1}^{n} \log \sum_{k'=1}^{K} \exp(y_{k'}^{(i)}) + \frac{1}{2} y_k^T C^{-1} y_k$$

- Sampling parameters τ and ρ is challenging. We calculate their potentials

$$U(\tau) = -\log(P(\tau|y,\theta_{-2})) = \frac{K}{2}\log\det C(\tau) + \frac{1}{2}\sum_{k=1}^{K} y_k^T C^{-1}(\tau) y_k + \frac{(\tau-\mu_2 1_p)}{2\sigma_2^2}$$

$$U(\rho) = -\log(P(\rho|y,\theta_{-4})) = \frac{K}{2}\log\det C(\rho) + \frac{1}{2}\sum_{k=1}^{K} y_k^T C^{-1}(\rho) y_k + \frac{(\rho-\mu_4 1_p)}{2\sigma_4^2}$$

- Hyper-parameters μ_h, σ_h^2 can be updated by Gibb's sampler

Table of Contents

Covariance functions (kernels)

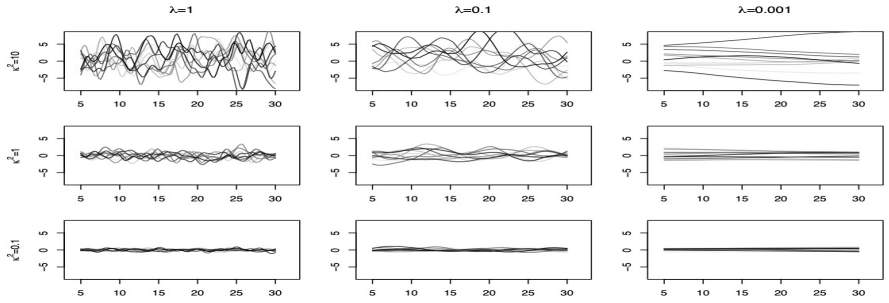
- Choosing an appropriate covariance function is an important step in setting Gaussian process models.
- Usually, from a class of valid kernels we choose a kernel that represents our beliefs regarding the underlying function, $y(x)$.
- Sometimes, we might choose a kernel that is computationally convenient computationally.
- In what follows, we discuss some of these alternative kernels.
- Note that we can create new kernels using their products and linear combinations.

Squared exponential

- The covariance function we discussed earlier is called *squared exponential*, which has the following form:

$$C_{ij} = \kappa^2 \exp[-\lambda(x_i - x_j)^2]$$

- Here, λ controls the correlation length, while κ^2 accounts for the height of oscillations in realizations of the GP.



Matérn process

- The Matérn class of kernels is defined in terms of the smoothness parameter, ν, length-scale, ρ, and variance σ^2 as follows:

$$C_{ij} = \sigma^2 \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\sqrt{2\nu} \frac{|x_i - x_j|}{\rho} \right)^{\nu} K_\nu \left(\sqrt{2\nu} \frac{|x_i - x_j|}{\rho} \right)$$

- Here, Γ and K are the Gamma and modified Bessel functions respectively.
- For $\nu = \frac{1}{2} + n$ and $n = 0, 1, \ldots$, the corresponding GP is n times continuously differentiable.

Ornstein-Uhlenbeck (OU) process

- Setting $\nu = 1/2$, we obtain a special case of GP called Ornstein-Uhlenbeck (OU) process,

$$C_{ij} = \sigma^2 \exp\Big(- \frac{|x_i - x_j|}{\rho} \Big)$$

- Compared to the squared exponential kernel, the resulting model is not very flexible.

Brownian motion (Wiener process)

- Brownian motion (Wiener process) is still simpler (rougher) than the OU process

$$C_{ij} = \sigma^2 \min(x_i, x_j)$$

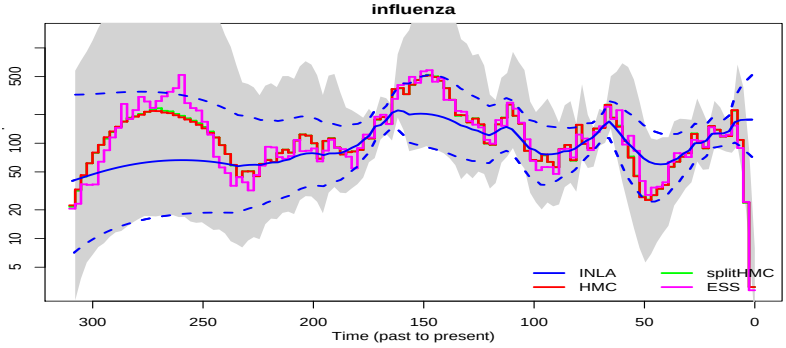
- In this case, the computational cost of inverting C is $\mathcal{O}(n)$.
- The OU process and the Wiener process are related through the following SDE:

$$dY_t = -\rho(Y_t - \mu)dt + dW_t,$$

where Y_t is an OU process and W_t is Brownian motion.

Brownian motion (Wiener process)

- Here is an example of using Brownian motion to model population dynamics of influenza (Lan, et el., 2014) using different sampling algorithms, Hamiltonian Monte Carlo (HMC), SplitHMC, and Elliptical Slice Sampling (ESS), and comparing the results to Integrated Nested Laplace Approximation (INLA).



Other resources

- To learn more about this topic, you could refer to
  - "Regression and classification using Gaussian process priors" (with discussion), by Neal, R. M. (1998).
  - "Gaussian Processes for Machine Learning," by Rasmussen and Williams (2006)
  - "Hierarchical Modeling and Analysis for Spatial Data," by Banerjee, Carlin, and Gelfand (2014).
  - "Dependent Matérn Processes for Multivariate Time Series, Vandenberg-Rodes, A. and Shahbaba, B. (2015).
- Python packages for GP modeling:
  - sklearn.gaussian_process.GaussianProcessRegressor , sklearn.gaussian_process.GaussianProcessClassifier
  - GPflow , GPy , PyMC , etc.

Table of Contents

Application: GP emulation of potential energy

- We need log-posterior for all MCMCs:

$$U(\boldsymbol{\theta}) = -\log L(\mathbf{x};\boldsymbol{\theta}) - \log P(\boldsymbol{\theta})$$

- Big data – the log-likelihood has huge amount items to add up;
  Complex models – they are computationally expensive to simulate.
- We need cheaper substitutes – Gaussian Process emulation.

$$\begin{aligned} U(\cdot) &\sim \mathcal{GP}(\mu(\cdot), \mathcal{C}(\cdot,\cdot)) \\ \mu(\boldsymbol{\theta}) &= \mathbf{h}(\boldsymbol{\theta})\boldsymbol{\beta}, \quad \mathbf{h}(\boldsymbol{\theta}) := [1, \boldsymbol{\theta}^{\mathsf{T}}, (\boldsymbol{\theta}^2)^{\mathsf{T}}] \text{ a } 1 \times (D+1)D \text{ vector} \\ \mathcal{C}(\cdot,\cdot) &= \sigma^2 \mathbf{C}(\cdot,\cdot), \quad \mathbf{C}(\boldsymbol{\theta}^i, \boldsymbol{\theta}^j) := \exp\{-(\boldsymbol{\theta}^i - \boldsymbol{\theta}^j)^{\mathsf{T}} \operatorname{diag}(\boldsymbol{\rho})(\boldsymbol{\theta}^i - \boldsymbol{\theta}^j)\} \end{aligned}$$

- Other parametrizations $\boldsymbol{\rho} = \mathbf{r}^{-2}$ ($\mathbf{r}$ correlation length), $\boldsymbol{\rho} = \mathrm{e}^{-\boldsymbol{\tau}}$.

GP emulation of potential energy

- Given design points $\mathfrak{D}_{\mathfrak{e}} := \{\boldsymbol{\theta}^1, \cdots, \boldsymbol{\theta}^n\}$, and conditioned on functional outputs $\mathbf{u}_{\mathfrak{D}} := U(\mathfrak{D}_{\mathfrak{e}})$, we can predict $U(\boldsymbol{\theta}^*)$ at $\mathfrak{E} := \{\boldsymbol{\theta}^{*1}, \cdots, \boldsymbol{\theta}^{*m}\}$, denoted as $\mathbf{u}_{\mathfrak{E}}$.
- Assume $p(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-2}$.
- Integrating $\boldsymbol{\beta}, \sigma^2$ out yields

$$\mathbf{u}_{\mathfrak{E}} | \mathbf{u}_{\mathfrak{D}}, \boldsymbol{\rho} \sim \mathcal{T}_{n-(D+1)}(\boldsymbol{\mu}^{**}, \widehat{\sigma}^2 \mathbf{C}^{**})$$

$$\boldsymbol{\mu}^{**} = \mathbf{H}_{\mathfrak{E}} \widehat{\boldsymbol{\beta}} + \mathbf{C}_{\mathfrak{E}\mathfrak{D}} \mathbf{C}_{\mathfrak{D}}^{-1} (\mathbf{u}_{\mathfrak{D}} - \mathbf{H}_{\mathfrak{D}} \widehat{\boldsymbol{\beta}})$$

$$\mathbf{C}^{**} = \mathbf{C}_{\mathfrak{E}} - \begin{bmatrix} \mathbf{H}_{\mathfrak{E}} & \mathbf{C}_{\mathfrak{E}\mathfrak{D}} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{H}_{\mathfrak{D}}^{\mathsf{T}} \\ \mathbf{H}_{\mathfrak{D}} & \mathbf{C}_{\mathfrak{D}} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{H}_{\mathfrak{E}}^{\mathsf{T}} \\ \mathbf{C}_{\mathfrak{D}\mathfrak{E}} \end{bmatrix}$$

$$\widehat{\boldsymbol{\beta}} = (\mathbf{H}_{\mathfrak{D}}^{\mathsf{T}} \mathbf{C}_{\mathfrak{D}}^{-1} \mathbf{H}_{\mathfrak{D}})^{-1} \mathbf{H}_{\mathfrak{D}}^{\mathsf{T}} \mathbf{C}_{\mathfrak{D}}^{-1} \mathbf{u}_{\mathfrak{D}}$$

$$\widehat{\sigma}^2 = (n - (D+1) - 2)^{-1} \mathbf{u}_{\mathfrak{D}}^{\mathsf{T}} \mathbf{Q}_{\mathfrak{D}} \mathbf{u}_{\mathfrak{D}}$$

GP emulation of potential energy

$$\mathbf{B}_{\mathfrak{D}} := (\mathbf{H}_{\mathfrak{D}}^{\mathrm{T}}\mathbf{C}_{\mathfrak{D}}^{-1}\mathbf{H}_{\mathfrak{D}})^{-1}$$
$$\mathbf{P}_{\mathfrak{D}} := \mathbf{B}_{\mathfrak{D}}\mathbf{H}_{\mathfrak{D}}^{\mathrm{T}}\mathbf{C}_{\mathfrak{D}}^{-1}$$
$$\mathbf{Q}_{\mathfrak{D}} := \mathbf{C}_{\mathfrak{D}}^{-1}[\mathbf{I} - \mathbf{H}_{\mathfrak{D}}\mathbf{P}_{\mathfrak{D}}]$$
$$\mathbf{L}_{\mathfrak{E}} := \mathbf{H}_{\mathfrak{E}}\mathbf{P}_{\mathfrak{D}} + \mathbf{C}_{\mathfrak{E}\mathfrak{D}}\mathbf{Q}_{\mathfrak{D}}$$

- *Best Linear Unbiased Predictor*

$$\mathbf{u}_{\mathfrak{E}}|\mathbf{u}_{\mathfrak{D}}, \boldsymbol{\rho} \approx \boldsymbol{\mu}^{**} = \mathbf{L}_{\mathfrak{E}}\mathbf{u}_{\mathfrak{D}}$$

- $\boldsymbol{\rho}$ is fixed at MLE.

$$\mathbf{P}_{\mathfrak{D}}\mathbf{H}_{\mathfrak{D}} = \mathbf{I}, \quad \mathbf{H}_{\mathfrak{D}}^{\mathrm{T}}\mathbf{Q}_{\mathfrak{D}} = \mathbf{Q}_{\mathfrak{D}}\mathbf{H}_{\mathfrak{D}} = \mathbf{0}$$

GP emulation with derivative information

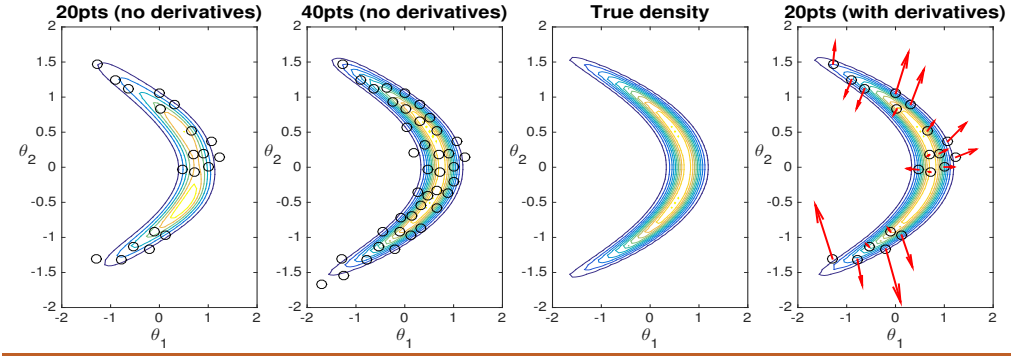
- Derivative information, $d\mathbf{u}_{\mathfrak{D}} = \nabla \otimes U(\mathfrak{D}\mathfrak{e})$ helps GP emulation.
- The differential operator is linear thus $dU(\cdot)$ is still a Gaussian Process (Papoulis and Pillai, 2002)

$$\mathrm{E}\left[\frac{\partial U(\boldsymbol{\theta}^i)}{\partial \theta^i_k}\right] = \frac{\partial}{\partial \theta^i_k}\mathrm{E}[U(\boldsymbol{\theta}^i)]$$

$$\mathrm{Cor}\left[\frac{\partial U(\boldsymbol{\theta}^i)}{\partial \theta^i_k}, U(\boldsymbol{\theta}^j)\right] = \frac{\partial}{\partial \theta^i_k}\mathbf{C}(\boldsymbol{\theta}^i, \boldsymbol{\theta}^j) = -2\rho_k(\theta^i_k - \theta^j_k)\mathbf{C}(\boldsymbol{\theta}^i, \boldsymbol{\theta}^j)$$

$$\begin{aligned}\mathrm{Cor}\left[\frac{\partial U(\boldsymbol{\theta}^i)}{\partial \theta^i_k}, \frac{\partial U(\boldsymbol{\theta}^j)}{\partial \theta^j_l}\right] &= \frac{\partial^2}{\partial \theta^i_k \partial \theta^j_l}\mathbf{C}(\boldsymbol{\theta}^i, \boldsymbol{\theta}^j) \\ &= [2\rho_k\delta_{kl} - 4\rho_k\rho_l(\theta^i_k - \theta^j_k)(\theta^i_l - \theta^j_l)]\mathbf{C}(\boldsymbol{\theta}^i, \boldsymbol{\theta}^j)\end{aligned}$$

Effect of derivative information on emulation

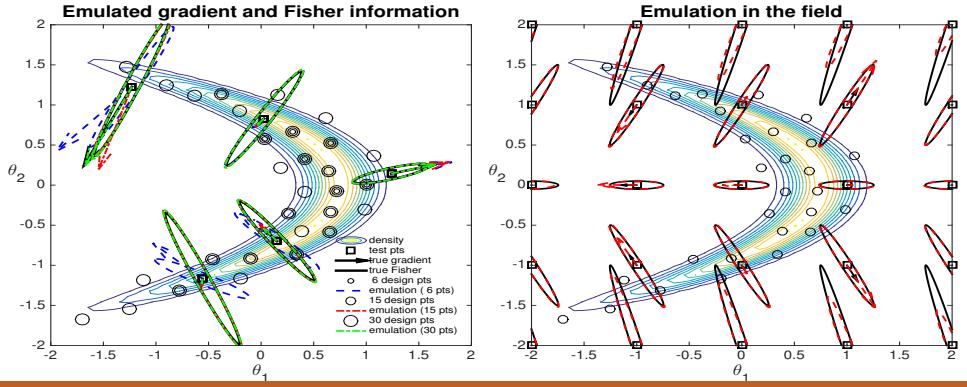


Proposition

Denote $\tilde{\mathbf{u}}_{\mathfrak{D}} = [\mathbf{u}_{\mathfrak{D}}^{\mathsf{T}}, d\mathbf{u}_{\mathfrak{D}}^{\mathsf{T}}]^{\mathsf{T}}$. *Given the same design set* $\mathfrak{D}\mathfrak{e}$, *we have*

$$E[(U(\boldsymbol{\theta}^*) - \hat{U}(\boldsymbol{\theta}^*)|\tilde{\mathbf{u}}_{\mathfrak{D}})^2] \leq E[(U(\boldsymbol{\theta}^*) - \hat{U}(\boldsymbol{\theta}^*)|\mathbf{u}_{\mathfrak{D}})^2]$$

Effect of design size on emulation



Proposition (Benjamin Haaland, Vaibhav Maheshwari)

For design sets $\mathfrak{D}_{\mathfrak{e}1} \subseteq \mathfrak{D}_{\mathfrak{e}2}$, we have

$$E[(U(\boldsymbol{\theta}^*) - \hat{U}(\boldsymbol{\theta}^*|\mathfrak{D}_{\mathfrak{e}2}))^2] \leq E[(U(\boldsymbol{\theta}^*) - \hat{U}(\boldsymbol{\theta}^*|\mathfrak{D}_{\mathfrak{e}1}))^2]$$

Gaussian Process on Hilbert Spaces

- Let X be a Hilbert space $\mathcal{H} = L^2(\mathcal{D}; \mathbb{R})$ on bounded open $\mathcal{D} \subset \mathbb{R}^d$ with Lipschitz boundary, and with inner-product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$
- Consider the following covariance operator C

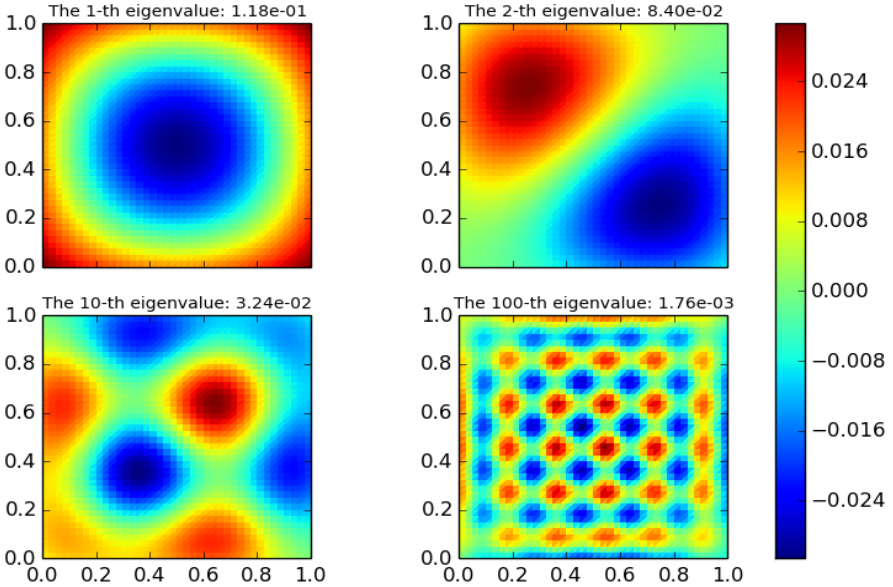
$$C := \sigma^2(\alpha I - \Delta)^{-s}$$

- Let $\{\lambda_i^2\}$ and $\{\phi_i(x)\}$ denote eigenvalues and eigenfunctions of C.
- If $s > d/2$ and $\lambda_i \asymp i^{-\frac{s}{d}}$, C defines a Gaussian measure $\mathcal{N}(0, C)$ that each draw $u(\cdot) \sim \mathcal{N}(0, C)$ admits Karhunen-Loève (K-L) expansion (Adler, 1981; Bogachev, 1998; Dashti and Stuart, 2015):

$$u(x) = \sum_{i=0}^{+\infty} u_i \lambda_i \phi_i(x), \qquad u_i \overset{iid}{\sim} \mathcal{N}(0, 1)$$

- Particularly useful in Bayesian Inverse Problems.

Gaussian Process: kernel eigenfunctions



Gaussian Process: Extended Reading

- Nice Intro-notes *Gaussian Processes for Machine Learning* by Carl Edward Rasmussen and Christopher K. I. Williams.
- Neal, R. M. (1996) *Bayesian learning for neural networks*. Springer Verlag. for connection to Neural networks.
- For regression:
  - P. Boyle and M. Frean (2005). *Dependent Gaussian processes* for multi-variate outputs.
  - C. E. Rasmussen and Z. Ghahramani (2002). *Infinite mixtures of Gaussian process experts* for combining GP and Dirichlet process mixture.
  - A. O'Hagan (1978). *Curve fitting and optimal design for prediction*.
- For classification:
  - D. Barber and C. K. I. Williams (1997). *Gaussian processes for Bayesian classification via hybrid Monte Carlo*.
  - M. Girolami and S. Rogers (2006). *Variational Bayesian multinomial probit regression with Gaussian process priors*.
  - R. M. Neal (1998). *Regression and classification using Gaussian process priors*.
- More on http://www.gaussianprocess.org.