

CEE 697M: Probabilistic Machine Learning

M4 Nonparametric Methods:

L4b: Gaussian Processes

Jimi Oke

UMass**Amherst**

College of Engineering

Fri, Apr 21, 2023

Outline

- ① Introduction
- ② Mercer kernels
- ③ Joint MVNs
- ④ Noise-free
- ⑤ Noisy
- ⑥ Kernel learning
- ⑦ Outlook

Gaussian processes

Given a domain \mathcal{X} , a Gaussian process (GP) defines a joint distribution over functions of the form $f : \mathcal{X} \rightarrow \mathbb{R}$.

Assumptions

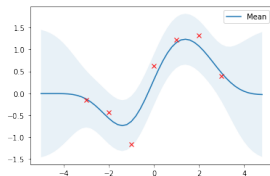
- Function values for $M > 0$ inputs $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_M)]$ is jointly Gaussian
 - Mean: $\boldsymbol{\mu} = m(\mathbf{x}_1, \dots, \mathbf{x}_M)$, where m is a mean function
 - Covariance: $\Sigma_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$, where \mathcal{K} is a Mercer kernel

More formally, a GP can be considered as a finite number of r.v.'s which have a joint Gaussian distribution

$$\mathbf{f}(\mathbf{x}) = \mathcal{GP}(\boldsymbol{\mu}, \Sigma_{ij}) \quad (1)$$

Gaussian processes and kernels

- The structure of the covariance Σ_{ij} is specified by a **kernel function** (which imposes a prior assumption of the distribution)
- Gaussian processes can be used for:
 - regression
 - classification
 - clustering



Source: <http://krasserm.github.io/2018/03/19/gaussian-processes/>

Mercer kernel

- Nonparametric methods require an approach to map **prior knowledge** regarding the pairwise similarity of input vectors $(\mathbf{x}_i, \mathbf{x}_j)$
- This is achieved using a kernel function \mathcal{K}
- A Mercer kernel is any symmetric function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$ such that:

$$\sum_{i=1}^N \sum_{j=1}^N \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) c_i c_j \geq 0 \quad (2)$$

for any set of N points $\mathbf{x}_i \in \mathcal{X}$ and any $c_i \in \mathbb{R}$

Gram matrix view

The Gram (or kernel) matrix is defined as:

$$\mathbf{K} = \begin{pmatrix} \mathcal{K}(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \mathcal{K}(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\mathbf{x}_N, \mathbf{x}_1) & \cdots & \mathcal{K}(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix} \quad (3)$$

If \mathbf{K} is **positive definite** for any set of unique inputs $\mathbf{x}_i \in \mathcal{X}, n = 1 : N$, then \mathcal{K} is a Mercer kernel.

- Thus, a Mercer kernel is also known as a positive definite kernel

Mercer's theorem

Any [symmetric] positive definite matrix \mathbf{K} can be eigendecomposed as:

$$\mathbf{K} = \mathbf{U}^\top \mathbf{\Lambda} \mathbf{U} \quad (4)$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues $\lambda_i > 0$ and \mathbf{U} the matrix of eigenvectors.

- Each element k_{ij} can be written as:

$$k_{ij} = (\mathbf{\Lambda}^{\frac{1}{2}} \mathbf{u}_i)^\top (\mathbf{\Lambda}^{\frac{1}{2}} \mathbf{u}_j) \equiv \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \quad (5)$$

where \mathbf{u}_i is the i -th column/row of \mathbf{U} .

- We can then express each element as an inner product of feature vectors specified by \mathbf{u}_i :

$$k_{ij} = \sum_m \phi_m(\mathbf{x}_i) \phi_m(\mathbf{x}_j) \quad (6)$$

Popular Mercer kernels

- **Squared exponential (SE)/RBF** kernel (below: $\ell = 1$)



Source: <https://peterroelants.github.io/posts/gaussian-process-kernels/>

- **Periodic** kernels (below: $\ell = 1, p = 1$)



Source: <https://peterroelants.github.io/posts/gaussian-process-kernels/>

- **Automatic relevancy determination (ARD)** kernel
- Kernels can be composed by addition, multiplication and other operations

Squared exponential kernel

The squared exponential (SE) kernel is given by:

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp \left[-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2} \right] \quad (7)$$

where ℓ is the length scale (or **bandwidth**) parameter

- Note that the SE kernel measures similarity between \mathbf{x} and \mathbf{x}' using a scaled Euclidean distance
- SE is also referred to as Gaussian, RBF or exponentiated quadratic

Joint Gaussian r.v.'s

If two random vectors \mathbf{x}_1 and \mathbf{x}_2 are jointly Gaussian, then their joint probability is given by:

$$p \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N} \left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix} \right) \quad (8)$$

Marginalization

Provides the distribution on each vector (partial information).
The marginal distributions are given by:

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1 | \mathbf{m}u_1, \mathbf{\Sigma}_{11}) \quad (9)$$

$$p(\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_2 | \mathbf{m}u_2, \mathbf{\Sigma}_{22}) \quad (10)$$

Marginal distributions are also Gaussian.

Conditioning

The **posterior** conditional distribution is:

$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1|\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2}) \quad (11)$$

where:

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2) \quad (\text{posterior mean})$$

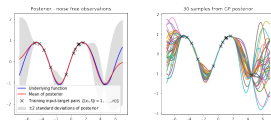
$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \quad (\text{posterior covariance})$$

Noise-free observations

In standard GP, we assume that observations in the training set are noise-free (exact output based on a known function f):

$$\mathcal{D} = \{(\mathbf{x}_n, y_n) : n = 1 : N\} \quad (12)$$

$$y_n = f(\mathbf{x}_n) \quad (\text{noise-free observation}) \quad (13)$$



Source: <https://www.aidanscannell.com/post/gaussian-process-regression/>

To predict function outputs for new inputs \mathbf{x}^* , we estimate the *posterior conditional distribution*

Joint distribution

The joint distribution $p(\mathbf{f}_X, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*)$ of function outputs is given by:

$$\begin{pmatrix} \mathbf{f}_X \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \boldsymbol{\mu}_X \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{X,X} & \mathbf{K}_{X,*} \\ \mathbf{K}_{X,*}^\top & \mathbf{K}_{*,*} \end{pmatrix} \right) \quad (14)$$

where:

- \mathbf{f}_X : function outputs over training set (interpolator): $[f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]$
- \mathbf{f}_* : function outputs over test set: $[f(\mathbf{x}_1^*), \dots, f(\mathbf{x}_{N_*}^*)]$
- Test inputs: $\mathbf{X}_* \in \mathbb{R}^{N_* \times D}$; training inputs: $\mathbf{X} \in \mathbb{R}^{N \times D}$
- Mean vectors: $\boldsymbol{\mu}_X = m(\mathbf{x}_1, \dots, \mathbf{x}_N)$ and $\boldsymbol{\mu}_* = m(\mathbf{x}_1^*, \dots, \mathbf{x}_{N_*}^*)$
- Block covariance matrix:
 - $\mathbf{K}_{X,X} = \mathcal{K}(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{N \times N}$
 - $\mathbf{K}_{X,*} = \mathcal{K}(\mathbf{X}, \mathbf{X}_*) \in \mathbb{R}^{N \times N_*}$
 - $\mathbf{K}_{*,*} = \mathcal{K}(\mathbf{X}_*, \mathbf{X}_*) \in \mathbb{R}^{N_* \times N_*}$

Posterior conditional distribution

We sample predictions from the posterior conditional distributon, which is specified as:

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D}) = \mathcal{N}(\mathbf{f}_*|\mathbf{u}_{*|X}, \Sigma_{*|X}) \quad (15)$$

$$\boldsymbol{\mu}_{*|X} = m(\mathbf{X}_*) + \mathbf{K}_{X,*}^\top \mathbf{K}_{X,X}^{-1}(\mathbf{f}_X - m(\mathbf{X})) \quad (16)$$

$$\Sigma_{*|X} = \mathbf{K}_{*,*} - \mathbf{K}_{X,*}^\top \mathbf{K}_{X,X}^{-1} \mathbf{K}_{X,*} \quad (17)$$

Noisy observations

If we have a noisy realization of the underlying function, then:

$$y_n = f(\mathbf{x}_n) + \epsilon_n \quad (18)$$

where $\epsilon_n \sim \mathcal{N}(0, \sigma_y^2)$.

The covariance of the observed noisy responses \mathbf{y} is thus given by:

$$\text{Cov}[\mathbf{y}|\mathbf{X}] = \mathbf{K}_{X,X} + \sigma_y^2 \mathbf{I}_N \equiv \hat{\mathbf{K}}_{X,X} \quad (19)$$

In scalar form, this can be written as:

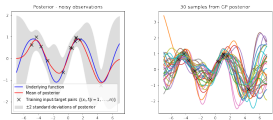
$$\text{Cov}[y_i, y_j] = \text{Cov}[f_i, f_j] + \text{Cov}[\epsilon_i, \epsilon_j] = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) + \sigma_y^2 \delta_{ij} \quad (20)$$

Joint density

The joint distribution of the observed data \mathbf{y} and the noise-free function on test points \mathbf{f}_* is:

$$\begin{pmatrix} \mathbf{f}_X \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \boldsymbol{\mu}_X \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \hat{\mathbf{K}}_{X,X} & \mathbf{K}_{X,*} \\ \mathbf{K}_{X,*}^\top & \mathbf{K}_{*,*} \end{pmatrix} \right) \tag{21}$$

Note that this has the same form as the joint distribution in the noise-free case, except: $\mathbf{K}_{X,X}$ is replaced by $\hat{\mathbf{K}}_{X,X}$, which is the covariance of the training inputs with a constant variance σ_y^2 added to all the diagonal terms.



GPR example with noisy inputs. Source: <https://www.aidancannell.com/post/gaussian-process-regression/>

Posterior predictive density

The conditional posterior distribution (posterior predictive) is given by:

$$p(\mathbf{f}_* | \mathbf{X}_*, \mathcal{D}) = \mathcal{N}(\mathbf{f}_* | \boldsymbol{\mu}_{*|X}, \boldsymbol{\Sigma}_{*|X}) \quad (22)$$

$$\boldsymbol{\mu}_{*|X} = m(\mathbf{X}_*) + \mathbf{K}_{X,*}^\top \hat{\mathbf{K}}_{X,X}^{-1} (\mathbf{y} - m(\mathbf{X})) \quad (23)$$

$$\boldsymbol{\Sigma}_{*|X} = \mathbf{K}_{*,*} - \mathbf{K}_{X,*}^\top \hat{\mathbf{K}}_{X,X}^{-1} \mathbf{K}_{X,*} \quad (24)$$

Optimizing kernel parameters

The value of kernel [hyper]parameters θ affects prediction performance.

- In SE kernels, we want to optimize the length scale ℓ
- For ARD kernels, we want to learn/optimize the characteristic length scale ℓ_d and the variance σ^2
- Hyperparameters can be learned via **maximum marginal likelihood**

Maximum marginal likelihood

Assuming the mean function is zero, the prior is given by:

$$p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}) \quad (25)$$

and the likelihood of each observation (conditioned on the latent function \mathbf{f}) can be written as:

$$p(\mathbf{y}|\mathbf{f}, \mathbf{X}) = \prod_{n=1}^N \mathcal{N}(y_n|f_n, \sigma_y^2) \quad (26)$$

The marginal likelihood is thus given by:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) d\mathbf{f} \quad (27)$$

To find the optimal kernel parameters $\boldsymbol{\theta}$, we maximize the **log marginal likelihood**:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \log \mathcal{N}(\mathbf{y}|\mathbf{0}, \hat{\mathbf{K}}_{\mathbf{X},\mathbf{X}}) = -\frac{1}{2} \mathbf{y}^\top \hat{\mathbf{K}}_{\mathbf{X},\mathbf{X}}^{-1} \mathbf{y} - \frac{1}{2} \log |\hat{\mathbf{K}}_{\mathbf{X},\mathbf{X}}| - \frac{N}{2} \log(2\pi) \quad (28)$$

Reading

- **PMLI 17.1-2**
- https://colab.research.google.com/github/krasserm/bayesian-machine-learning/blob/dev/gaussian-processes/gaussian_processes.ipynb
- <https://peterroelants.github.io/posts/gaussian-process-kernels/>
- <https://github.com/aidanscannell/probabilistic-modelling/blob/master/notebooks/gaussian-process-regression.ipynb>