Introduction to Gaussian Process

Matheus Schossler

January 29, 2023

A **Gaussian process** are used to define probability distribution over functions. With this concept, we can build the Gaussian process regression algorithm (GPR), which is a data-based prediction method with good performance in dealing with high dimension small samples, nonlinear problems. Let us start with a simpler algorithm to build our intuition around Gaussian processes: the Bayesian linear regression.

Bayesian Linear Regression

Let $S = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$be a training set of i.i.d. examples from a unknown distribution. The linear regression states that

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon \tag{1}$$

where ε are i.i.d. noise with $\mathcal{N}\left(0, \sigma^2\right)$ distribution. This implies

$$p\left(y^{(i)}|x^{(i)}, \theta\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^{(i)} - \theta^T x^{(i)}}{2\sigma^2}\right). \tag{2}$$

In Bayesian linear regression, we assume a **prior distribution** over parameters is also given,

$$\theta \sim \mathcal{N}\left(0, \tau^2 I\right), \tag{3}$$

which from Bayes's rule, the θ distribution conditioned to S is

$$p(\theta|S) = \frac{p(\theta)p(S|\theta)}{\int_{\theta'} p(\theta')p(S|\theta')d\theta'}. \tag{4}$$

In this model $p(S|\theta) = \prod_i p\left(y^{(i)}|x^{(i)}, \theta\right)$, where $p\left(y^{(i)}|x^{(i)}, \theta\right)$ is the normal distribution defined in 2. The output of this model on a new test point $x_*$is given by the **posterior predictive distribution**

$$p(y_*|x_*, S) = \int_\theta p(y_*|x_*, \theta)p(\theta|S)d\theta. \tag{5}$$

For the linear regression model above, we can express both $p(\theta|S)$ and $p(y_*|x_*, S)$ in terms of a closed formula of X_*, S, σ, τ. It can be shown that BLR is a GPR with the kernel given by the Dot-Product.

Gaussian Process

A Gaussian Process is a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions. A Gaussian process is fully specified by its mean function $m(x)$ and covariance function $k(x, x')$:

$$f \sim \mathcal{GP}(m, k), \tag{6}$$

which reads "the function f is distributed as a GP with mean function m and covariance function k". We use a set of indexes $\{x_i \in \mathcal{X}\}_{i=1}^m$ to denote the equation above in terms of their vector notation $\boldsymbol{f}$:

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_m) \end{bmatrix}, \begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_m) \\ \vdots & \ddots & \vdots \\ k(x_m,x_1) & \cdots & k(x_m,x_m) \end{bmatrix} \right), \tag{7}$$

where $\mathcal{N}(\mu, \Sigma)$ denotes a multivariate Gaussian distribution with mean $\mu \in \mathbb{R}^n$ and covariance matrix Σ as defined above.

Example

The Bayesian linear model suffers from limited expressiveness. A very simple idea to overcome this problem is to project the inputs into some high dimensional space using a set of basis feature space functions, e.g. $\{1, x, x^2, \ldots, x^{d-1}\}$, and then apply the linear model in this space instead of directly on the inputs themselves. This type of map also defines a simple example of a Gaussian process:

$$f(\boldsymbol{x}) = \phi(\boldsymbol{x})^T \boldsymbol{\theta} \tag{8}$$

with prior $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$, where Σ_p is a a $p \times p$ covariance matrix, which implies

$$\begin{aligned} \mathbb{E}[f(\boldsymbol{x})] &= \phi(\boldsymbol{x})^T \mathbb{E}[\boldsymbol{\theta}] = 0 \\ \mathbb{E}[f(\boldsymbol{x}) f(\boldsymbol{x}')] &= \phi(\boldsymbol{x})^T \mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^T] \phi(\boldsymbol{x}') = \phi(\boldsymbol{x})^T \Sigma_p \phi(\boldsymbol{x}'). \end{aligned} \tag{1}$$

$f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ are jointly Gaussian with zero mean and covariance $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \Sigma_p \phi(\boldsymbol{x}')$. Therefore any combination of $f(\boldsymbol{x}^{(i)})$, with $\boldsymbol{x}^{(i)}$ in the input set X is a Gaussian process. Using the kernel trick it is possible to show that, fundamentally all we need to know about the feature map $\phi(\boldsymbol{x})$ is encapsulated in the corresponding kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ $(= \Sigma_p)$.

Gaussian Process Regression

Gaussian process regression (GPR) is a typical data-based prediction method, which has a better performance in dealing with **high dimension small sample**, **nonlinear problems**. Let $S = \left\{ \left(x^{(i)}, y^{(i)} \right) \right\}_{i=1}^m$ be a training set of i.i.d. examples from an unknown distribution. A noisy regression algorithm problem is concerned with estimating a function f such that

$$y^{(i)} = f\left(x^{(i)} \right) + \varepsilon, \ i = 1, \ldots, m \tag{9}$$

where ε are i.i.d. noise with $\mathcal{N}\left(0, \sigma^2\right)$ distribution.

We assume a **prior distribution** over functions $f(\cdot)$ following a zero-mean Gaussian process

$$f \sim \mathcal{GP}(0, k), \tag{10}$$

for some valid covariance function (or kernel) $k(\cdot, \cdot)$. It can be shown that the squared-exponential covariance function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f \exp\left(-\frac{1}{2l^2} ||\boldsymbol{x} - \boldsymbol{x}'||^2 \right), \tag{11}$$

where σ_f and l are the hyperparameters, corresponds to a Bayesian linear regression model with an infinite dimensional basis (we can make the train data set as large as we want and the space the kernel operates in keeps growing without bound).

Now, let $T = \left\{ \left(x_*^{(i)}, y_*^{(i)} \right) \right\}_{i=1}^m$ be a training set of i.i.d. examples from the same unknown distribution as S. We want to calculate the posterior predictive distribution over the testing outputs $y_*^{(i)}$. We write the joint distribution of both sets $\{f(x) : x \in X\}$ and $\{f_*(x) : x \in X_*\}$

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f_*} \end{bmatrix} | X, X_* \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix} \right) \Rightarrow \boldsymbol{f_*} | \boldsymbol{f} \sim \text{Normal distribution.} \tag{12}$$

Using the formula for conditioning a joint Gaussian distribution we can find an expression for $\boldsymbol{f_*}|\boldsymbol{f}$. The joint Gaussian i.i.d. noise vector becomes

$$\begin{bmatrix} \boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon_*} \end{bmatrix} |X, X_* \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma^2 I & 0 \\ 0 & \sigma^2 I \end{bmatrix}\right). \tag{13}$$

Finally the **posterior predictive distribution** is found,

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y_*} \end{bmatrix} |X, X_* = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f_*} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon_*} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X)+\sigma^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*)+\sigma^2 I \end{bmatrix}\right). \tag{2}$$

$$\Rightarrow \tag{3}$$

$$\boxed{\boldsymbol{y_*}|\boldsymbol{y}, X, X_* \sim \mathcal{N}(\mu_*, \Sigma_*)} \tag{14}$$

Where μ_*, Σ_* are expressed in terms of $X_*, S, k(\cdot,\cdot), \sigma$

$$\mu_* = K(X_*, X)\Lambda^{-1}\boldsymbol{y} \tag{4}$$

$$\Sigma_* = K(X_*, X_*) - K(X_*, X)\Lambda^{-1}K(X_*, X), \tag{5}$$

where $\Lambda = K(X,X) + \sigma^2 I$.

Bayesian Linear Regression as a Gaussian Process Regressor

A Gaussian process $f(x)$ is completely specified by its mean function $m(x)$ and covariance function $k(x, x\prime)$. Here $x \in X$ denotes a point on the index set X. These functions are defined by

$$m(x) = \mathbb{E}[f(x)] \tag{15}$$

and

$$k(x, x\prime) = \mathbb{E}[(f(x) - m(x))(f(x\prime) - m(x\prime))]. \tag{16}$$

Ref link shows that $f(x)$ as defined by (1) defines a Gaussian Process because $\theta \sim \mathcal{N}\left(0, \tau^2 I\right)$. In addition, link, shows that $m(x) = 0$ and $k(x, x\prime) = \tau^2 x^T x + \sigma^2$.

Training a Gaussian Process Regressor

Gaussian Process Regressor is useful if we have enough prior information about a dataset at hand to specify prior mean and covariance functions confidently. However, the availability of such detailed prior information is not the typical case in machine learning applications. We use hierarchical specification to specify vague prior information in the form of m and k in a simple way and optimize the introduced hyperparameters based on the data.

We start writing a general model:

$$f \sim \mathcal{GP}(m, k), \tag{18}$$

where

$$m(\boldsymbol{x}) = a\boldsymbol{x}^2 + b\boldsymbol{x} + c \tag{6}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{2}{2l^2}\left(\boldsymbol{x} - \boldsymbol{x}'\right)^T M \left(\boldsymbol{x} - \boldsymbol{x}'\right)\right) + \sigma_n^2 \delta_{ii'} \tag{7}$$

we define the hyperparameters set as :

$$H = \{a, b, c, \{M\}, \sigma_f, \sigma_n\}, \tag{19}$$

where M can be defined in terms of the characteristic length-scales of correlations, l_i, or the factor analysis distance matrix, Λ. For high-dimensional datasets, this matrix could identify a few directions

in the input space with especially high "relevance" (in the sense of PCA - high variance). Although we are using squared exponential, any positive definite function can be used as covariance function.

Given the marginal likelihood

$$L = \log p(\boldsymbol{y}|\boldsymbol{x}, \theta) = -\frac{1}{2}\left(\log|\Sigma| + (\boldsymbol{y} - m)^T \Sigma^{-1} (\boldsymbol{y} - m)\right) \tag{8}$$

we can maximize L using its partial derivatives with respect to H:

$$\frac{\partial L}{\partial H_m}, \frac{\partial L}{\partial H_k}, \tag{20}$$

where H_m and H_k are used to indicate hyperparameters of the mean and covariance functions, respectively, and optimization algorithms such as the gradient descent, momentum gradient descent, or the Adam algorithm.

The trade-off between penalty and data fit in the GP model is automatic (based on the L exact expression).

Disadvantages of Gaussian Process Regression

- It is expensive to build a model when there is a lot of training data. An implementation of the algorithm explained here requires the inversion of the covariance matrix $\Sigma = K(X, X)$ using (Cholesky factorization), with a memory complexity of $O(n^2)$ and a computational complexity of $O(n^3)$. There has been work on GP models that only include a **subset of the data** (Gramacy and Apley 2015) which can be implemented efficiently (Gramacy et al. 2014).

- Another issue of the model is stationary covariance function is assumed, but for many problems, the character of the covariance function needs to change in different regimes. Gramacy and Lee (2008) developed a **hybrid tree-Gaussian process model** that allows the covariance function to change over the range of input data.

Alternatives

- Alternatives of GPR are the Bayesian Multiple Adaptive Regression Splines (**MARS**) which can automatically handle some of the issues with nonstationary covariances, and the underlying computation in fitting a Bayesian MARS model is a least-squares solve, which can be done efficiently.

- The **twin Gaussian process** is a structured prediction method which uses Gaussian process priors [8] on both covariates and responses, both multivariate. This method predicts outputs by **minimizing the Kullback-Leibler divergence between two GP** modeled as normal distributions over finite index sets of training and testing examples. It's based on the fact that similar inputs should produce similar percepts. (Mohammad Mojaddady, Moin Nabi, and Shahram Khadivi - Stock Market Prediction using Twin Gaussian Process Regression).

Dataset Name	Mean Square Error	
	Twin Gaussian Process	***Gaussian Process***
Adobe Systems Incorporated (NASDAQ:ADBE)	5.5129	31.664
Autodesk Inc (NASDAQ:adsk)	6.9853	38.1152
Electronic Art Inc (NASDAQ:erts)	9.3889	45.4864
Microsoft Corporation (NASDAQ:msft)	3.8729	23.2404
Oracle Corporation (NAZDAQ:orcl)	9.1444	14.0564
SAP (NAZDAQ:sap)	10.6153	53.8016
Symantec Corporation (NAZDAQ:symc)	13.218	16.5636
VeriSign Inc (NAZDAQ:vrsn)	13.2947	22.6892
Tehran Stock Index	0.79102	4.4541

Figure 1: Mean Square Error of Stock Market prediction

- Use it together with other models (auto-encoder, recurrent neural network (RNN) and long short-term memory (LSTM)) to construct the interval prediction of the input signal and analyze the uncertainties (e.g. stock market). Ref: Stock index prediction and uncertainty analysis using multi-scale nonlinear ensemble paradigm of optimal feature extraction, two-stage deep learning and Gaussian process regression, Wang et al. 2021.