Gaussian Process Regression Summaries

1.Mathematical Description

Gaussian Process Regression is a Bayesian model of regression which focus on calculating the posterior distribution over models, which measures the uncertainty in model predictions and make more robust for new test points. It is not in the way that classical machine learning algorithms does: to solve optimization problem and find the best model or to estimate a best model which can get good prediction on future test input points. It is a kernel-based fully Bayesian regression algorithm.

Firstly, we begin to the idea of Bayesian linear regression. Suggesting we have a model following:

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}, \qquad i = 1,2,\dots,m$$

Where the ε denotes for noise which has gaussian distribution:$\aleph(0,\sigma^2)$, then we can also suggest that $y^{(i)} - \theta^T x^{(i)} \sim \aleph(0,\sigma^2)$. Using concepts from statistics we can define the likelihood and marginal likelihood as:

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

$$P(\boldsymbol{y}|\boldsymbol{X}) = \int_{\theta_{min}}^{\theta_{max}} P(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}$$ ①

Then, we estimate the probability of θ as we know the information of **X**,**y** leveraging Bayes' theorem:

$$Posterior = \frac{likelihood \times prior}{marginal\ likelihood}$$

$$P(\theta|\boldsymbol{X},\boldsymbol{y}) = \frac{P(\theta)\prod_{i=1}^{m} P\left(y^{(i)}\middle|x^{(i)},\theta\right)}{\int P(\theta')\prod_{i=1}^{m} P\left(y^{(i)}\middle|x^{(i)},\theta'\right) d\theta'}$$

Now, we obtained the distribution of parameter (Note: in tradition regression algorithm, parameter θ is a exact value, but here is a probability distribution). Then, we can apply it to predict new points assuming that noise over all dataset have the same distribution, i.e $\aleph(0,\sigma^2)$. We can define a **posterior predictive distribution** for prediction. In statistics, it is noting but a likelihood known the new input and the information of old dataset.

$$P(y^*|x^*,\boldsymbol{X},\boldsymbol{y}) = \int P(y^*|x^*,\theta)P(\theta|\boldsymbol{X},\boldsymbol{y})d\theta$$

To solve the both equations above, we need to use a technique called **maximum a posteriori (MAP)**, which can estimate the weight θ. The results show here(S denotes $(\boldsymbol{X},\boldsymbol{y})$):

$$\theta \mid S \sim \mathcal{N}\left(\frac{1}{\sigma^2}A^{-1}X^T\vec{y}, A^{-1}\right)$$

$$y_* \mid x_*, S \sim \mathcal{N}\left(\frac{1}{\sigma^2}x_*^T A^{-1}X^T\vec{y}, x_*^T A^{-1}x_* + \sigma^2\right)$$

① Bold uppercase letter such as **X** denotes a matrix containing input variables; bold lowercase letter such as **y**,**θ** denotes a vector of output or labels and a vector of parameters.

Where $A = \frac{1}{\sigma^2} X^T X + \frac{1}{\tau^2} I$.

A Gaussian Process Regression is a complex problem of Bayesian model. Gaussian process is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution. If we have a set of functions, denotes $\{f(x): x \in \chi\}$ which has gaussian distribution with mean function m $(\cdot)$ and covariance function k$(\cdot,\cdot)$. For a series of x inputs, we have:

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

Then, we can define a Gaussian process as:

$$f(\cdot) \sim GP(m(\cdot), k(\cdot,\cdot))$$

Because covariance matrix is positive semidefinite on every random input points, it can be identified as a kernel. In the Gaussian process, we use Gaussian kernel to calculate coveriences, i.e:

$$k_{SE}(x, x') = \exp\left(-\frac{1}{2\tau^2} ||x - x'||^2 \right)$$

Same assuming for Bayesian regression model but we may not have one function for calculating y, in contrast, here we have a set of functions for calculate y.

$$y^{(i)} = f(x^{(i)}) + \varepsilon^{(i)}, \qquad i = 1, \ldots, m$$

Firstly, we assume a prior distribution for f $(\cdot)$, which is similar as we did in Bayesian regression. It is very important step for the problem. It is the basic assumption of Gaussian Process Regression.

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

Secondly, plitting the overall dataset as training set denotes $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ and testing set $S = \{(x_*^{(i)}, y_*^{(i)})\}_{i=1}^{m*}$

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix} \in \mathbf{R}^{m \times n} \quad \vec{f} = \begin{bmatrix} f(x^{(1)}) \\ f(x^{(2)}) \\ \vdots \\ f(x^{(m)}) \end{bmatrix}, \quad \vec{\varepsilon} = \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(m)} \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbf{R}^{m},$$

$$X_* = \begin{bmatrix} — & (x_*^{(1)})^T & — \\ — & (x_*^{(2)})^T & — \\ & \vdots & \\ — & (x_*^{(m_*)})^T & — \end{bmatrix} \in \mathbf{R}^{m_* \times n} \quad \vec{f}_* = \begin{bmatrix} f(x_*^{(1)}) \\ f(x_*^{(2)}) \\ \vdots \\ f(x_*^{(m_*)}) \end{bmatrix}, \quad \vec{\varepsilon}_* = \begin{bmatrix} \varepsilon_*^{(1)} \\ \varepsilon_*^{(2)} \\ \vdots \\ \varepsilon_*^{(m_*)} \end{bmatrix}, \quad \vec{y}_* = \begin{bmatrix} y_*^{(1)} \\ y_*^{(2)} \\ \vdots \\ y_*^{(m_*)} \end{bmatrix} \in \mathbf{R}^{m_*}.$$

For prediction, we are supposed to calculate **posterior predictive distribution**.

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

Similarly, as the Bayesian regression did, we leveraging the conditional probability method:

$$P(\vec{y}_*|\vec{y},\boldsymbol{X},\boldsymbol{X}_*) = \frac{P(\vec{y}_*,\vec{y}|\boldsymbol{X},\boldsymbol{X}_*)}{P(\vec{y}|\boldsymbol{X},\boldsymbol{X}_*)}$$

$$P(\vec{y}|\boldsymbol{X},\boldsymbol{X}_*) = \int P(\vec{y}_*,\vec{y}|\boldsymbol{X},\boldsymbol{X}_*)d\vec{y}_*$$

The results show below:

$$\vec{y_*} \mid \vec{y}, X, X_* \sim \mathcal{N}(\mu^*, \Sigma^*)$$

where

$$\mu^* = K(X_*,X)\left(K(X,X)+\sigma^2 I\right)^{-1}\vec{y}$$
$$\Sigma^* = K(X_*,X_*)+\sigma^2 I - K(X_*,X)\left(K(X,X)+\sigma^2 I\right)^{-1}K(X,X_*).$$

There is a picture shows how it work:

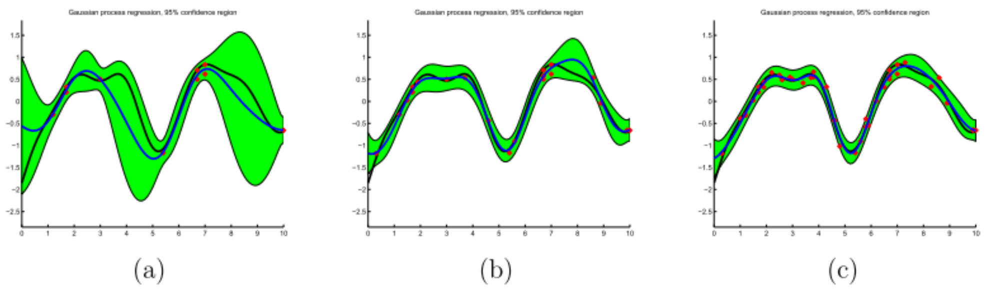


Figure 3: Gaussian process regression using a zero-mean Gaussian process prior with $k_{SE}(\cdot,\cdot)$ covariance function (where $\tau = 0.1$), with noise level $\sigma = 1$, and (a) $m = 10$, (b) $m = 20$, and (c) $m = 40$ training examples. The blue line denotes the mean of the posterior predictive distribution, and the green shaded region denotes the 95% confidence region based on the model's variance estimates. As the number of training examples increases, the size of the confidence region shrinks to reflect the diminishing uncertainty in the model estimates. Note also that in panel (a), the 95% confidence region shrinks near training points but is much larger far away from training points, as one would expect.

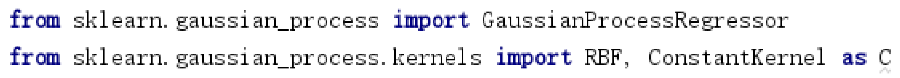
I'd like to mention some additional information about the figures: the black line is the true model calculated from ground knowledge. The blue line is the prediction line which is μ*which denotes the mean of posterior predictive. Green ranging is the area has 95% confidence. Comparing the different decision of numbers of training points using to calculate, we can see the more training points select, the less confidence area would be contained. That is, different selection of training points may cause the model underfitting or overfitting.

2. Python coding for Gaussian Process Regression

Python sklearn package has Gaussian Process Regression algorithm. In that case, researchers can write a few codes to finish their work using Gaussian Process Regression. I found a simple example online which can be used to have a better understand how it works

To do a gaussian Process Regression with sklearn, we provide a dataset using to train and test algorithm in the beginning. Then a kernel function needs to be defined. After defining a good kernel, we can do the Gaussian Process Regression immediately to fit and predict new points. To evaluate its performance, we usually need a ground knowledge model for the dataset points and compare it with the prediction model. The codes are attached below, and more explanations provides here for the codes:

Packages we need to import:



```
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel as C
```

GaussianProcessRegressor: support the regressor to execute the GPR process (see documentation). The kernel parameters calculate by maximum the log-marginal likelihood using Limited-memory BFGS.

RBF: Gaussian kernel setter leverages to set a Gaussian kernel format to calculate the kernel (covariance) in data. It returns a RBF type object (see documentation).

ConstantKernel: It provides a scale factor for kernel, usually it can be a variance (see documentation and supplement documentation).

Reference

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

2.cs229-gaussian_processes notes, Stanford

3.C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X.

4.Python documentations