COMP4131: Data Modelling and Analysis

Lecture 7: Gaussian Process Regression and Classification

Daokun Zhang

University of Nottingham Ningbo China

daokun.zhang@nottingham.edu.cn

31 March, 2025

Overview

Gaussian Process

Linear Regression

Recall the probabilistic analysis of the standard linear regression model with Gaussian noise

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}\right), \quad y = f(\mathbf{x}) + \varepsilon,$$

where ε is the additive noise that follows an i.i.d Gaussian distribution with zero mean and variance σ_n^2:

$$\varepsilon \sim \mathcal{N}\left(0, \sigma_n^2\right).$$

Given the training data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\mathbf{y} = \{y_1, \cdots, y_N\}$, we can use maximum likelihood estimation (MLE) to find the solution of $\mathbf{w}$ as

$$\mathbf{w}^* = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{y},$$

where $\mathbf{\Phi} = \left[\phi\left(\mathbf{x}_1\right), \cdots, \phi\left(\mathbf{x}_N\right)\right]^{\mathrm{T}}$ is an $N \times M$ matrix.

Bayesian Linear Regression

In the Bayesian formalism, we can specify a prior over the parameters, expressing our beliefs about the parameters before we look at the observations. We put a Gaussian prior with mean vector $\boldsymbol{\mu}_{\mathbf{w}}$ and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{w}}$ on the weights

$$\mathbf{w} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{w}}, \boldsymbol{\Sigma}_{\mathbf{w}}\right).$$

Due to the randomness in $\mathbf{w}$, $f(x) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x})$ is no longer a deterministic but a random function. Let f_p denote the linear function value $f(\mathbf{x}_n)$ at sample $\mathbf{x}_n$, then $f_1, \cdots f_N$ follow a multivariate Gaussian distribution with

$$\mathbb{E}\left[f_p\right] = \boldsymbol{\mu}_{\mathbf{w}}^{\mathrm{T}}\phi(\mathbf{x}_p),$$
$$\mathrm{Cov}\left(f_n, f_m\right) = \phi\left(\mathbf{x}_n\right)^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{w}}\phi\left(\mathbf{x}_m\right).$$

Now we get a Gaussian process $f(\mathbf{x})$ with mean function $m(\mathbf{x}) = \boldsymbol{\mu}_{\mathbf{w}}^{\mathrm{T}}\phi(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}') = \phi\left(\mathbf{x}\right)^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{w}}\phi\left(\mathbf{x}'\right)$.

Gaussian Process

Definition: *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

Implications

- "a collection of random variables" means a set that has
  - a finite number of random variables
  - an infinite number of random variables
- "any finite number of which have a joint Gaussian distribution" means:
  - multiple random variables follow a joint multivariate Gaussian distribution
  - after marginalization, a single random variable follows a univariate Gaussian distribution

Gaussian Process

A Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[\{f(\mathbf{x}) - m(\mathbf{x})\}\{f(\mathbf{x}') - m(\mathbf{x}')\}\right], \end{aligned}$$

and write the Gaussian process as

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right).$$

Usually, for notational simplicity we will take the mean function to be zero, i.e., set $m(\mathbf{x}) = 0$.

Example: For Bayesian linear regression model $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x})$ with prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{w}})$, we have the mean and covariance function

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] = \phi(\mathbf{x})^{\mathrm{T}}\mathbb{E}[\mathbf{w}] = 0, \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[f(\mathbf{x})f(\mathbf{x}')\right] = \phi(\mathbf{x})^{\mathrm{T}}\mathbb{E}\left[\mathbf{w}\mathbf{w}^{\mathrm{T}}\right]\phi(\mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{w}}\phi(\mathbf{x}'). \end{aligned}$$

Kernel Trick

The covariance function $k(\mathbf{x}, \mathbf{x}')$ can be extended to any kernel functions.

For all $\mathbf{x}$ and $\mathbf{x}'$ in the input space $\mathcal{X}$, the function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel function, if there exists a feature map $\varphi : \mathcal{X} \to \mathcal{V}$ that satisfies

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \left\langle \varphi(\mathbf{x}), \varphi\left(\mathbf{x}'\right) \right\rangle_{\mathcal{V}},$$

where $\mathcal{V}$ is a inner product space, and $\langle \cdot, \cdot \rangle_{\mathcal{V}}$ is the inner product operation defined by $\mathcal{V}$.

An alternative definition can be formulated by the *positive semidefinite (PSD)* property: For any points $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ in $\mathcal{X}$, and all choices of n real-valued coefficients $(c_1, \ldots, c_n)$, the function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel function, if

$$\sum_{i=1}^{n} \sum_{j=1}^{n} k\left(\mathbf{x}_i, \mathbf{x}_j\right) c_i c_j \geq 0.$$

The spanned matrix, $\boldsymbol{K} \in \mathbb{R}^n \times \mathbb{R}^n$ with its ij-th entry $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, is called Gram matrix.

Kernel Trick

Many kernels can be chosen for various application scenarios

- Fisher kernel
- Polynomial kernel
- Radial basis function kernel (RBF)
- String kernels
- Graph kernels

The Radial basis function kernel (RBF) is also called squared-exp or Gaussian kernel, which is formulated as

$$k_{\mathrm{SE}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\ell^2}\right).$$

Gaussian Process Regression

Prediction with Noise-free Observations

We first consider the ideal case where the observation y can be described by a Gaussian process $f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$ in a noise-free manner, i.e., $y = f(x)$. For a set of test samples with size n_*, and feature matrix $\boldsymbol{X}_*$, its function value vector $\mathbf{f}_*$ follows a multivariate Gaussian distribution

$$\mathbf{f}_* \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{K}\left(\boldsymbol{X}_*, \boldsymbol{X}_*\right)\right),$$

where $\boldsymbol{K}(\cdot,\cdot)$ is the covariance matrix between two collections of input feature vectors.

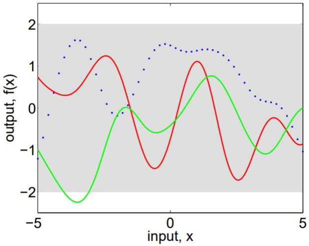
If we are given a set of training samples with size n, and feature matrix $\boldsymbol{X}$, its function value vector $\mathbf{f}$ should follow a multivariate Gaussian distribution jointly with $\mathbf{f}_*$

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) & \boldsymbol{K}\left(\boldsymbol{X}, \boldsymbol{X}_*\right) \\ \boldsymbol{K}\left(\boldsymbol{X}_*, \boldsymbol{X}\right) & \boldsymbol{K}\left(\boldsymbol{X}_*, \boldsymbol{X}_*\right) \end{bmatrix}\right).$$

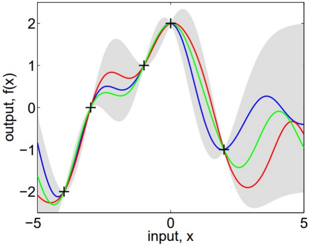
Prediction with Noise-free Observations

Given the observations of $\mathbf{f}$, the posterior distribution of $\mathbf{f}_*$ is also a Gaussian distribution

$$\mathbf{f}_* | \boldsymbol{X}_*, \boldsymbol{X}, \mathbf{f} \sim \mathcal{N}\left(\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1} \mathbf{f} \right. \\ \left. \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \right).$$



(a), prior



(b), posterior

The prior and posterior distributions of $\mathbf{f}_*$ with the RBF kernel function $k(\mathbf{x}_p, \mathbf{x}_q) = \exp(-\frac{1}{2}\|\mathbf{x}_p - \mathbf{x}_q\|_2^2)$.

Prediction using Noisy Observations

It is typical for more realistic modelling situations that we do not have access to function values themselves, but only noisy versions thereof $y = f(\mathbf{x}) + \varepsilon$.

Assuming additive independent identically distributed Gaussian noise ε with zero mean and variance σ_n^2, the prior on the noisy observations becomes another Gaussian distribution with mean $\mathbb{E}[y] = \mathbb{E}[f(\mathbf{x}) + \varepsilon] = 0$, and covariance

$$\operatorname{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \text{ or } \operatorname{cov}(\mathbf{y}) = K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I},$$

where δ_{pq} is a Kronecker delta symbol which is one if and only if $p = q$ and zero otherwise.

Gaussian Process Regression

We can write the joint distribution of the observed target values $\mathbf{y}$ and the function values $\mathbf{f}_*$ of test samples under the prior as

$$\left[\begin{array}{c} \mathbf{y} \\ \mathbf{f}_* \end{array}\right] \sim \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{cc} \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}_*) \end{array}\right]\right).$$

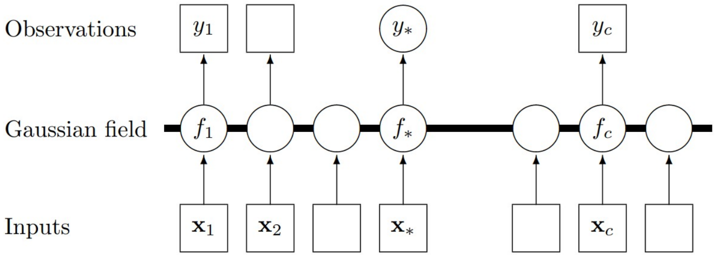
Deriving the conditional distribution, we arrive at the key predictive equations for Gaussian process regression

$$\mathbf{f}_* | \boldsymbol{X}, \mathbf{y}, \boldsymbol{X}_* \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)\right), \text{ where}$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}\left[\mathbf{f}_* | \boldsymbol{X}, \mathbf{y}, \boldsymbol{X}_*\right] = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}\right]^{-1} \mathbf{y},$$

$$\operatorname{cov}(\mathbf{f}_*) = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*).$$

Gaussian Process Regression



Graphical model (chain graph) for a Gaussian process for regression. Squares represent observed variables and circles represent unknowns.

Gaussian Process Regression

By using a compact form of the notation setting $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ and $\mathbf{k}_* = \boldsymbol{K}(\boldsymbol{X}, \mathbf{x}_*)$ for a test sample $\mathbf{x}_*$, the prediction f_* for the single test sample $\mathbf{x}_*$ has mean $\bar{f}_*$ and variance $\mathbb{V}[f_*]$ as

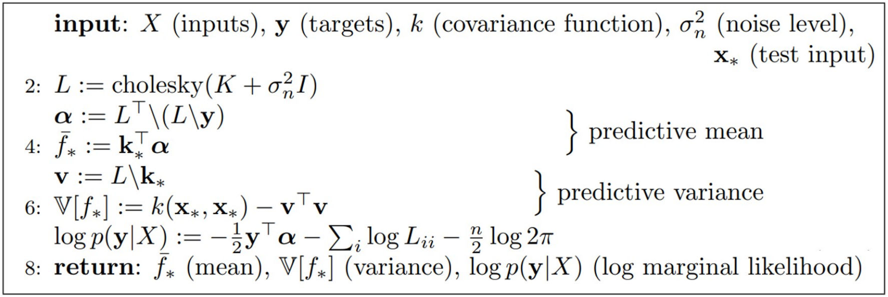
$$\bar{f}_* = \mathbf{k}_*^\top \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \mathbf{y},$$
$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \mathbf{k}_*.$$

Another way to look at the expression of the mean $\bar{f}_*$ is to see it as a linear combination of N kernel functions, each one centered on a training point, by writing

$$\bar{f}_* = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*),$$

where $\boldsymbol{\alpha} = \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \mathbf{y}$.

Gaussian Process Regression



```
input: $X$ (inputs), $\mathbf{y}$ (targets), $k$ (covariance function), $\sigma_n^2$ (noise level), $\mathbf{x}_*$ (test input)
$L := \text{cholesky}(K + \sigma_n^2 I)$
$\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$            } predictive mean
$\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$
$\mathbf{v} := L \backslash \mathbf{k}_*$                          } predictive variance
$\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$
$\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^\top \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi$
return: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)
```

Algorithm for Gaussian process regression, where the predictions (the target value's mean and variance) at the test sample $\mathbf{x}_*$ and the log marginal likelihood on training set are returned.

Varying the Hyperparameters

Setting the RBF kernel $k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp(-\frac{1}{2\ell^2}\|\mathbf{x}_p - \mathbf{x}_q\|_2^2)$ as the covariance function, the covariance between the observations y_p and y_q becomes

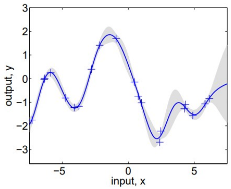
$$\operatorname{cov}(y_p, y_q) = \sigma_f^2 \exp(-\frac{1}{2\ell^2}\|\mathbf{x}_p - \mathbf{x}_q\|_2^2) + \sigma_n^2 \delta pq,$$

which implies that the model relies on three hyperparameters:

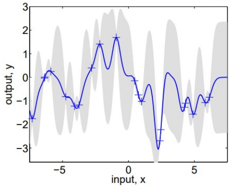
- signal variance σ_f^2
- noise variance σ_n^2
- length-scale ℓ

The log marginal likelihood can be used to tune the hyperparameters.

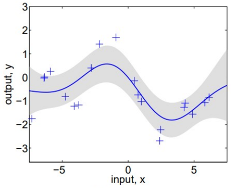
Varying the Hyperparameters



(a), $\ell = 1$



(b), $\ell = 0.3$



(c), $\ell = 3$

Small length-scale $\ell = 0.3 \rightarrow$ a sharply varying function f with small noise.
Large length-scale $\ell = 3 \rightarrow$ a slowly varying function f with large noise.
Moderate length-scale $\ell = 1 \rightarrow$ exact fitting of the underlying function f.

Gaussian Process Classification

Binary Gaussian Process Classifier

For binary classification with label candidate set $\{+1, -1\}$, linear regression uses the sigmoid function to squash the linear regression output into the label probability $p(y = +1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}))$.

In Gaussian process, the latent function value $f(\mathbf{x})$ is used to construct the linear regression output.

Similarly, for Gaussian process classification, we can also construct the label probability via the sigmoid function

$$p(y = +1|f(\mathbf{x})) = \sigma(f(\mathbf{x})).$$

Different from $\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x})$, $f(\mathbf{x})$ is Gaussian random variable, we need to marginalize it to get the posterior probability

$$p(y = +1|\mathbf{x}) = \int p(y = +1|f)p(f|\mathbf{x})\mathrm{d}f.$$

Binary Gaussian Process Classifier

Given a training set $\{\boldsymbol{X}, \mathbf{y}\}$, for a new test sample $\mathbf{x}_*$, Gaussian process classifier predicts its label by estimating the posterior probability

$$\begin{aligned}\bar{\pi}_* \triangleq p\left(y_* = +1 | \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*\right) &= \int p\left(y_* = +1 | f_*\right) p\left(f_* | \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*\right) \mathrm{d} f_* \\ &= \int \sigma(f_*) p\left(f_* | \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*\right) \mathrm{d} f_*.\end{aligned}$$

To achieve the estimation, we need first calculate the posterior distribution of the latent variable f_* corresponding to the test sample $\mathbf{x}_*$

$$p\left(f_* | \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*\right) = \int p\left(f_* | \boldsymbol{X}, \mathbf{x}_*, \mathbf{f}\right) p(\mathbf{f} | \boldsymbol{X}, \mathbf{y}) \mathrm{d} \mathbf{f},$$

where $p(\mathbf{f}|\boldsymbol{X}, \mathbf{y}) = p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\boldsymbol{X}) / p(\mathbf{y}|\boldsymbol{X})$ is the posterior over the latent variables and $p(\mathbf{y}|\mathbf{f})$ is computed by applying the elementwise sigmoid function to $\mathbf{f}$.

Laplace Approximation for Binary GP Classifier

In the regression case, $\mathbf{y}$ and f_* jointly follow a multivariate Gaussian distribution, the conditional probability distribution $p\left(f_* | \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*\right)$ can be computed analytically.

However, in the classification case, due to the computational intractability of the distribution $p(\mathbf{f} | \boldsymbol{X}, \mathbf{y})$, the posterior distribution $p\left(f_* | \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*\right)$ cannot be computed analytically with the intractable integral.

To solve this problem, we resort to the Laplace approximation that approximates the distribution $p(\mathbf{f} | \boldsymbol{X}, \mathbf{y})$ with a Gaussian distribution $q(\mathbf{f} | \boldsymbol{X}, \mathbf{y})$, obtained by doing a second order Taylor expansion of $\log p(\mathbf{f} | \boldsymbol{X}, \mathbf{y})$ around the maximum of the posterior

$$q(\mathbf{f} | \boldsymbol{X}, \mathbf{y}) = \mathcal{N}\left(\mathbf{f} | \hat{\mathbf{f}}, \boldsymbol{A}^{-1}\right) \propto \exp\left(-\frac{1}{2}(\mathbf{f} - \hat{\mathbf{f}})^\top \boldsymbol{A}(\mathbf{f} - \hat{\mathbf{f}})\right),$$

where $\hat{\mathbf{f}} = \operatorname{argmax}_{\mathbf{f}} p(\mathbf{f} | \boldsymbol{X}, \mathbf{y})$ and $\boldsymbol{A} = -\nabla\nabla \log p(\mathbf{f} | \boldsymbol{X}, \mathbf{y})\big|_{\mathbf{f}=\hat{\mathbf{f}}}$ is the Hessian matrix of the negative log posterior at that point.

Laplace Approximation for Binary GP Classifier

By Bayes' rule, the posterior over the latent variables $p(\mathbf{f}|\boldsymbol{X}, \mathbf{y})$ is given by

$$p(\mathbf{f}|\boldsymbol{X}, \mathbf{y}) = \frac{p(\mathbf{y}, \mathbf{f}|\boldsymbol{X})}{p(\mathbf{y}|\boldsymbol{X})} = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{X})}{p(\mathbf{y}|\boldsymbol{X})},$$

but $p(\mathbf{y}|\boldsymbol{X})$ is independent of $\mathbf{f}$, we only need to consider the un-normalized posterior $p(\mathbf{y}, \mathbf{f}|\boldsymbol{X}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{X})$ when maximizing w.r.t. $\mathbf{f}$.

Taking the logarithm on the un-normalized posterior, we get

$$\begin{aligned}\boldsymbol{\Psi}(\mathbf{f}) &\triangleq \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|\boldsymbol{X}) \\ &= \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^{\mathrm{T}}\boldsymbol{K}^{-1}\mathbf{f} - \frac{1}{2}\log|\boldsymbol{K}| - \frac{n}{2}\log 2\pi,\end{aligned}$$

where $p(\mathbf{f}|\boldsymbol{X})$ is a Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{K})$, $\boldsymbol{K}$ is short for the kernel matrix $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$, and $|\boldsymbol{K}|$ is the determinant of the kernel matrix $\boldsymbol{K}$.

Laplace Approximation for Binary GP Classifier

Differentiating $\mathbf{\Psi}(\mathbf{f})$ w.r.t. $\mathbf{f}$, we obtain

$$\nabla\mathbf{\Psi}(\mathbf{f}) = \nabla \log p(\mathbf{y}|\mathbf{f}) - \boldsymbol{K}^{-1}\mathbf{f},$$
$$\nabla\nabla\mathbf{\Psi}(\mathbf{f}) = \nabla\nabla \log p(\mathbf{y}|\mathbf{f}) - \boldsymbol{K}^{-1},$$

where $\nabla\nabla \log p(\mathbf{y}|\mathbf{f})$ is diagonal, since the distribution for $y_i = +1$ or -1 depends only on f_i , not on $f_{j \neq i}$.

We can apply the Newton-Raphson method to find the latent function value vector $\hat{\mathbf{f}}$ that maximizes $\mathbf{\Psi}(\mathbf{f})$, and obtain the approximated probability distribution as

$$q(\mathbf{f}|\boldsymbol{X}, \mathbf{y}) = \mathcal{N}\left(\hat{\mathbf{f}}, \left(\boldsymbol{K}^{-1} + \boldsymbol{W}\right)^{-1}\right),$$

where $\boldsymbol{W} \triangleq -\nabla\nabla \log p(\mathbf{y}|\mathbf{f})|_{\mathbf{f}=\hat{\mathbf{f}}}$.

Laplace Approximation for Binary GP Classifier

With $q(\mathbf{f}|\boldsymbol{X}, \mathbf{y})$ approximating $p(\mathbf{f}|\boldsymbol{X}, \mathbf{y})$, we can get the approximation for the posterior $p(f_*|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_*)$ as

$$q(f_*|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\boldsymbol{X}, \mathbf{x}_*, \mathbf{f})\, q(\mathbf{f}|\boldsymbol{X}, \mathbf{y})\mathrm{d}\mathbf{f},$$

where both $p(f_*|\boldsymbol{X}, \mathbf{x}_*, \mathbf{f})$ and $q(\mathbf{f}|\boldsymbol{X}, \mathbf{y})$ are Gaussian distributions. It can be proved that $q(f_*|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_*)$ is also a Gaussian distribution with mean and variance as

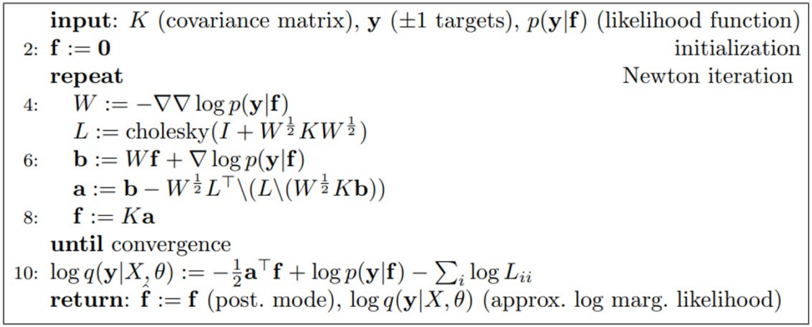
$$\mathbb{E}_q[f_*|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_*] = \mathbf{k}_*^{\mathrm{T}} \boldsymbol{K}^{-1}\hat{\mathbf{f}},$$
$$\mathbb{V}_q[f_*|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^{\mathrm{T}}\left(\boldsymbol{K} + \boldsymbol{W}^{-1}\right)^{-1}\mathbf{k}_*.$$

Finally, the label probability for the test sample $\mathbf{x}_*$ can computed as

$$\bar{\pi}_* \triangleq p(y_* = +1|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_*) := \int \sigma(f_*) q(f_*|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_*)\,\mathrm{d}f_*.$$

Numeric integration techniques are generally used to calculate the one-dimensional integral.

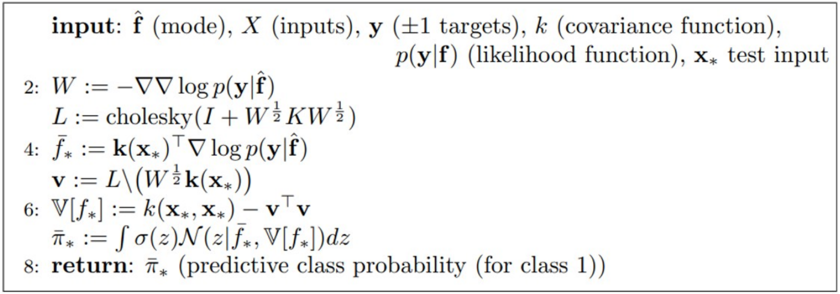
Binary Gaussian Process Classifier



```
input: $K$ (covariance matrix), $\mathbf{y}$ ($\pm 1$ targets), $p(\mathbf{y}|\mathbf{f})$ (likelihood function)
$\mathbf{f} := \mathbf{0}$                                                    initialization
repeat                                                                     Newton iteration
    $W := -\nabla\nabla \log p(\mathbf{y}|\mathbf{f})$
    $L := \text{cholesky}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})$
    $\mathbf{b} := W\mathbf{f} + \nabla \log p(\mathbf{y}|\mathbf{f})$
    $\mathbf{a} := \mathbf{b} - W^{\frac{1}{2}} L^\top \backslash (L \backslash (W^{\frac{1}{2}} K \mathbf{b}))$
    $\mathbf{f} := K\mathbf{a}$
until convergence
$\log q(\mathbf{y}|X, \theta) := -\frac{1}{2}\mathbf{a}^\top \mathbf{f} + \log p(\mathbf{y}|\mathbf{f}) - \sum_i \log L_{ii}$
return: $\hat{\mathbf{f}} := \mathbf{f}$ (post. mode), $\log q(\mathbf{y}|X, \theta)$ (approx. log marg. likelihood)
```

Training algorithm for the binary Gaussian process classifier with Laplace approximation. The mode $\hat{\mathbf{f}}$ for the posterior probability distribution $p(\mathbf{f}|\boldsymbol{X}, \mathbf{y})$ is returned, as well as the approximated log marginal likelihood, which can be use to tune hyperparameters.

Binary Gaussian Process Classifier



```
input: $\hat{\mathbf{f}}$ (mode), $X$ (inputs), $\mathbf{y}$ ($\pm 1$ targets), $k$ (covariance function),
        $p(\mathbf{y}|\mathbf{f})$ (likelihood function), $\mathbf{x}_*$ test input
$W := -\nabla\nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$
$L := \text{cholesky}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})$
$\bar{f}_* := \mathbf{k}(\mathbf{x}_*)^\top \nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$
$\mathbf{v} := L \backslash (W^{\frac{1}{2}} \mathbf{k}(\mathbf{x}_*))$
$\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$
$\bar{\pi}_* := \int \sigma(z) \mathcal{N}(z|\bar{f}_*, \mathbb{V}[f_*]) dz$
return: $\bar{\pi}_*$ (predictive class probability (for class 1))
```

Prediction algorithm for the binary Gaussian classifier with Laplace approximation.

Multi-Class Gaussian Process Classifier

For the multi-class classification problem, where the class label y belongs to the label candidate set $\{1, \cdots, C\}$ with size C, we need to construct a Gaussian process for each class $c \in \{1, \cdots, C\}$, with the latent function value $f^c(\mathbf{x})$ evaluated at $\mathbf{x}$.

The conditional label probability $p\left(y = c | f^1(\mathbf{x}), \cdots, f^C(\mathbf{x})\right)$ can be constructed by the softmax function

$$p\left(y = c | f^1(\mathbf{x}), \cdots, f^C(\mathbf{x})\right) = \frac{\exp\left(f^c(\mathbf{x})\right)}{\sum_{c'=1}^{C} \exp\left(f^{c'}(\mathbf{x})\right)}.$$

Following the same procedure as the binary classification case, given a test sample $\mathbf{x}_*$, we can estimate the posterior probabilities $p(y_* = c | \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*)$ to predict $\mathbf{x}_*$'s class label.

References

Williams CK, Rasmussen CE. **Gaussian Processes for Machine Learning**. Cambridge, MA: MIT press; 2006.

- Section 1 - Section 3

The End