STA3010 Regression Analysis

Feng Yin

The Chinese University of Hong Kong (Shenzhen)

yinfeng@cuhk.edu.cn

May 7, 2020

Overview

Outline

Background

We start with a general regression model

$$y_i = f(\boldsymbol{x}_i) + e_i, \quad i = 1, 2, ..., n, \tag{1}$$

where

- n is the number of the data points;
- $\boldsymbol{x}_i$ is the i-th input of dimension p;
- y_i is the i-th output (assumed to be scalar);
- $f(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}$ is the underlying regression function to be approximated;
- e_i is an error term.

Background

In general, a machine learning algorithm involves the following two phases:

Learning phase: Assume a (parametric or nonparametric) model of $f(\boldsymbol{x})$ and train the model given a set of training data, denoted by $\mathcal{D} \triangleq \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_n, y_n)\} = \{\boldsymbol{X}, \boldsymbol{y}\}$.

Prediction phase: Given a novel data point with input $\boldsymbol{x}_*$, we aim to predict the output y_*.

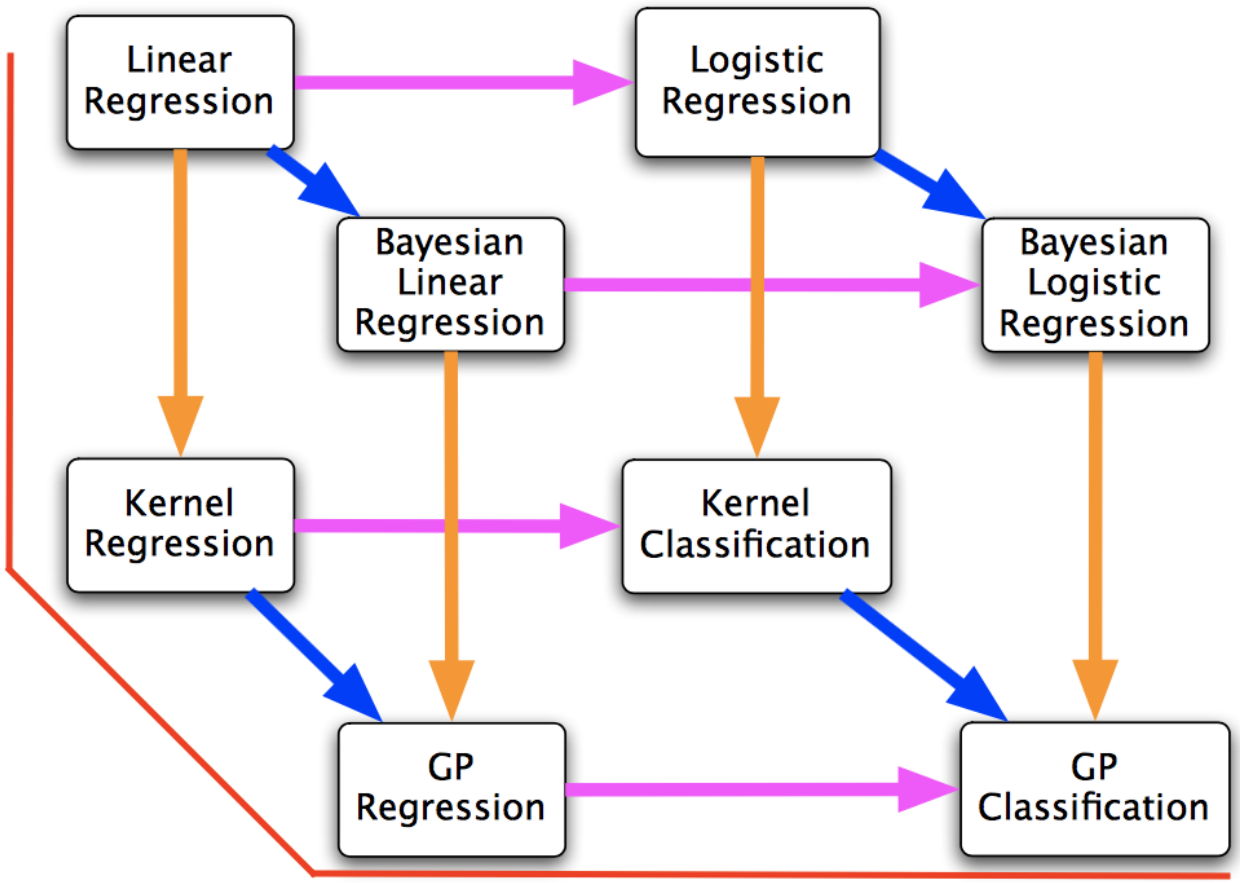
Two options prevail:

1. Discard the training data and make predictions purely based on the model that has been trained $\rightarrow$ parametric model.
2. Use the training data during the prediction phase $\rightarrow$ nonparametric model.

In this presentation, we mainly focus on the second option.

Background

"Roadmap" of this presentation:



Outline

Regularized Least-Squares in Canonical Form

Let us approximate the underlying function by a linear regression model

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^T \phi(\boldsymbol{x}_i), \tag{2}$$

where $\boldsymbol{w}$ is an $m \times 1$ vector of weighting factors of a fixed nonlinear feature space mapping $\phi(\boldsymbol{x}_i) : \mathbb{R}^p \to \mathbb{R}^m$, which returns a vector of m basis function values evaluated at point $\boldsymbol{x}_i$.

The classic regularized least-squares (RLS) method for solving the weighting factors $\boldsymbol{w}$ can be expressed as:

$$\boldsymbol{w}_{RLS} = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{1}{2} \sum_{i=1}^{n} \left(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) - y_i \right)^2 + \frac{\lambda}{2} \boldsymbol{w}^T \boldsymbol{w}, \tag{3}$$

where λ is the regularization parameter.

Regularized Least-Squares in Dual Form

The point estimate of the weighting factors can be formulated as

$$\boldsymbol{w} = \sum_{i=1}^{n} a_i \phi(\boldsymbol{x}_i) = \boldsymbol{\Phi}^T \boldsymbol{a}, \tag{4}$$

where $\boldsymbol{\Phi}$ is the design matrix of size $n \times m$, whose ith row is given by $\phi(\boldsymbol{x}_i)^T$ and the vector $\boldsymbol{a} = [a_1, a_2, ..., a_n]^T$ with the ith entry defined by

$$a_i \triangleq -\frac{1}{\lambda}\left(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) - y_i\right). \tag{5}$$

Regularized Least-Squares in Dual Form

The dual representation of the above RLS can be obtained by substituting $\boldsymbol{w} = \boldsymbol{\Phi}^T \boldsymbol{a}$ into the original cost function $J(\boldsymbol{w})$ and expanding the square term, namely,

$$J(\boldsymbol{a}) \triangleq \frac{1}{2}\boldsymbol{a}^T \boldsymbol{K}\boldsymbol{K}\boldsymbol{a} - \boldsymbol{a}^T \boldsymbol{K}\boldsymbol{y} + \frac{1}{2}\boldsymbol{y}^T \boldsymbol{y} + \frac{\lambda}{2}\boldsymbol{a}^T \boldsymbol{K}\boldsymbol{a}, \tag{6}$$

where $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$ and $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ is called the Gram matrix or kernel matrix, which is an $n \times n$ symmetric matrix with the (i, j)-th entry (the ith row and jth column of $\boldsymbol{K}$)

$$[\boldsymbol{K}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j), \tag{7}$$

where the term kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ is introduced eventually.

Regularized Least-Squares in Dual Form

Taking the derivative of $J(\boldsymbol{a})$ w.r.t. $\boldsymbol{a}$ and set it equal to zero, yields

$$\boldsymbol{a}_{RLS} = (\boldsymbol{K} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{y}. \tag{8}$$

Inserting this result back into the linear parametric regression model, we obtain the following prediction for a new input $\boldsymbol{x}_*$ as follows:

$$y_* = \phi(\boldsymbol{x}_*)^T \boldsymbol{w}_{RLS} = \phi(\boldsymbol{x}_*)^T \boldsymbol{\Phi}^T \boldsymbol{a}_{RLS} = \boldsymbol{k}(\boldsymbol{x}_*)^T (\boldsymbol{K} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{y} \tag{9}$$

where we have defined the vector $\boldsymbol{k}(\boldsymbol{x}_*)$ with entries $k_i(\boldsymbol{x}_*) \triangleq k(\boldsymbol{x}_i, \boldsymbol{x}_*)$.

Remark: The dual form rewrites the RLS solution entirely in terms of the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$.

Regularized Least-Squares in Dual Form

The cons and pros of the dual representation are the following:

- Pros: The prediction works directly with the kernel function and avoids the selection of fixed basis functions $\phi(\boldsymbol{x})$, which allows us to use feature spaces of higher, even infinite, dimension.
- Cons: In the dual representation, we need to invert an $n \times n$ matrix for computing $\boldsymbol{a}$ rather than invert a smaller $m \times m$ matrix for $\boldsymbol{w}$ in the original problem.

In summary, larger model flexibility is gained with higher computational complexity.

Constructing Valid Kernels

According to [J. Shawe-Taylor and N. Cristianini, 04], a necessary and sufficient condition for a function $k(\boldsymbol{x}, \boldsymbol{x}')$ to be a valid kernel is that the corresponding kernel matrix $\boldsymbol{K}$ should be positive semidefinite (PSD) for all possible choices of $\boldsymbol{x} \in \mathcal{X}$.

For clarity, we give the kernel matrix $\boldsymbol{K}$ explicitly as follows:

$$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta}_h) = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1; \boldsymbol{\theta}_h) & k(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{\theta}_h) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_n; \boldsymbol{\theta}_h) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1; \boldsymbol{\theta}_h) & k(\boldsymbol{x}_2, \boldsymbol{x}_2; \boldsymbol{\theta}_h) & \dots & k(\boldsymbol{x}_2, \boldsymbol{x}_n; \boldsymbol{\theta}_h) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1; \boldsymbol{\theta}_h) & k(\boldsymbol{x}_n, \boldsymbol{x}_2; \boldsymbol{\theta}_h) & \dots & k(\boldsymbol{x}_n, \boldsymbol{x}_n; \boldsymbol{\theta}_h) \end{bmatrix}, \quad (10)$$

where $\boldsymbol{\theta}_h$ is a set of parameters that control the characteristic of the kernel function. $\boldsymbol{\theta}_h$ is often called hyper-parameters and later $\boldsymbol{K}$ or $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ is short for $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta}_h)$.

Constructing Valid Kernels

We have two options for constructing valid kernels:

1. Naive solution: Choose a feature space mapping $\phi(\boldsymbol{x})$ and use the definition

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}') = \sum_{i=1}^{m} \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{x}'), \tag{11}$$

   where $\phi_i(\boldsymbol{x})$, $i = 1, 2, ..., m$ are the basis functions.

2. More elegant solution: Construct valid kernel functions without having to construct the feature space mapping $\phi(\boldsymbol{x})$ first. Build complex but valid kernels out of elementary kernels under valid operations.

Constructing Valid Kernels

According to [C. Bishop, 06], some fundamental properties can be exploited for constructing a new valid kernel $k(\boldsymbol{x}, \boldsymbol{x}')$, they are:

$$k(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x}) f(\boldsymbol{x}') \tag{12a}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = c k_1(\boldsymbol{x}, \boldsymbol{x}') \tag{12b}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x}) k_1(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}') \tag{12c}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(k_1(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{12d}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}') \tag{12e}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') \cdot k_2(\boldsymbol{x}, \boldsymbol{x}') \tag{12f}$$

where $k_1(\boldsymbol{x}, \boldsymbol{x}')$ and $k_2(\boldsymbol{x}, \boldsymbol{x}')$ are both known valid kernels; $f(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}$ is any function; $c \geq 0$ is a constant.

Elementary Kernels: SE Kernel

Squared exponential (SE) kernel (a.k.a. radial basis function kernel or Gaussian kernel), has the form:

$$k_{SE}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2l^2}\right). \tag{13}$$

The SE kernel is often deemed as the default kernel for Gaussian Processes and SVM.

The set of hyper-parameters to be tuned is $\boldsymbol{\theta}_h = [\sigma^2, l]^T$, where

- The length scale l determines the length of the 'wiggles' in the function.
- The variance σ^2 determines the average distance of the function away from its mean.

Elementary Kernels: SE Kernel

- Simple task I: Show that the SE kernel is a valid kernel.

  Answer:

  1. Expand $||\boldsymbol{x} - \boldsymbol{x}^{'}||^2$ to be $\boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{x}^{'} + (\boldsymbol{x}^{'})^T\boldsymbol{x}^{'}$.
  2. Express $k_{SE}(\boldsymbol{x}, \boldsymbol{x}^{'}) = \sigma^2 \cdot f(\boldsymbol{x})k_1(\boldsymbol{x}, \boldsymbol{x}^{'})f(\boldsymbol{x}^{'})$, where $\sigma^2 > 0$, $l^2 > 0$, $f(\boldsymbol{x}) = \exp\left[\frac{-\boldsymbol{x}^T\boldsymbol{x}}{2l^2}\right]$ and $k_1(\boldsymbol{x}, \boldsymbol{x}^{'}) = \exp\left[\frac{\boldsymbol{x}^T\boldsymbol{x}^{'}}{l^2}\right]$.
  3. $k_2(\boldsymbol{x}, \boldsymbol{x}^{'}) = \boldsymbol{x}^T\boldsymbol{x}^{'}$, i.e., the linear kernel is a valid kernel, and so is $k_2(\boldsymbol{x}, \boldsymbol{x}^{'})/l^2$.
  4. Apply the properties shown beforehand.

- Simple task II: Show that the SE kernel corresponds to infinite number of basis function, i.e., the dimension of $\phi(\boldsymbol{x})$ is infinite.

  Answer: Use Taylor expansion, details can be found in appendix.

Elementary Kernels: RQ Kernel

Rational quadratic (RQ) kernel is of the form:

$$k_{RQ}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \left(1 + \frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\alpha l^2}\right)^{-\alpha}. \tag{14}$$

This kernel can be seen as a scale mixture (an infinite sum) of SE kernel functions with different characteristic length-scales. When $\alpha \to \infty$, RQ kernel $\to$ SE kernel, see Appendix.

The set of hyper-parameters to be tuned is $\boldsymbol{\theta}_h = [\sigma^2, l, \alpha]^T$.

Elementary Kernels: Periodic Kernel

Periodic kernel is of the form:

$$k_{PER}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{\sin^2\left(\pi||\boldsymbol{x} - \boldsymbol{x}'||/p\right)}{l^2}\right), \tag{15}$$

which allows one to model functions which repeat themselves exactly.

The set of hyper-parameters to be tuned is $\boldsymbol{\theta}_h = [\sigma^2, p, l]^T$, where

- The period p simply determines the distance between repetitions of the function.

Elementary Kernels: Local Periodic Kernel

Locally periodic kernel is of the form:

$$k_{LP}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{\sin^2\left(\pi||\boldsymbol{x}-\boldsymbol{x}'||/p\right)}{l_1^2}\right) \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{x}'||^2}{2l_2^2}\right). \tag{16}$$

An SE kernel times a periodic kernel results in functions which are periodic, but can slowly vary over time.

The set of hyper-parameters to be tuned is $\boldsymbol{\theta}_h = [\sigma^2, p, l_1, l_2]^T$, where

- The length scales l_1 and l_2 together determine the length of the 'wiggles' in the function.

Stationary and Isotropic Kernels

Before we move on, let us quickly recap some background knowledge:

- Stationary kernel function to be a function of $\boldsymbol{\tau} = \boldsymbol{x} - \boldsymbol{x}'$.
- Isotropic kernel function to be a function of $r = \|\boldsymbol{x} - \boldsymbol{x}'\|$.

It is obvious that the above four examples are not only stationary but also isotropic.

Stationary kernel and its spectral density are Fourier duals:

$$k(\boldsymbol{\tau}) = \int S(s) e^{j2\pi \boldsymbol{s}^T \boldsymbol{\tau}} d\boldsymbol{\tau} \tag{17}$$

$$S(\boldsymbol{s}) = \int k(\boldsymbol{\tau}) e^{-j2\pi \boldsymbol{s}^T \boldsymbol{\tau}} d\boldsymbol{s} \tag{18}$$

Example: $\tau = t$ representing time and $s = f$ representing frequency.

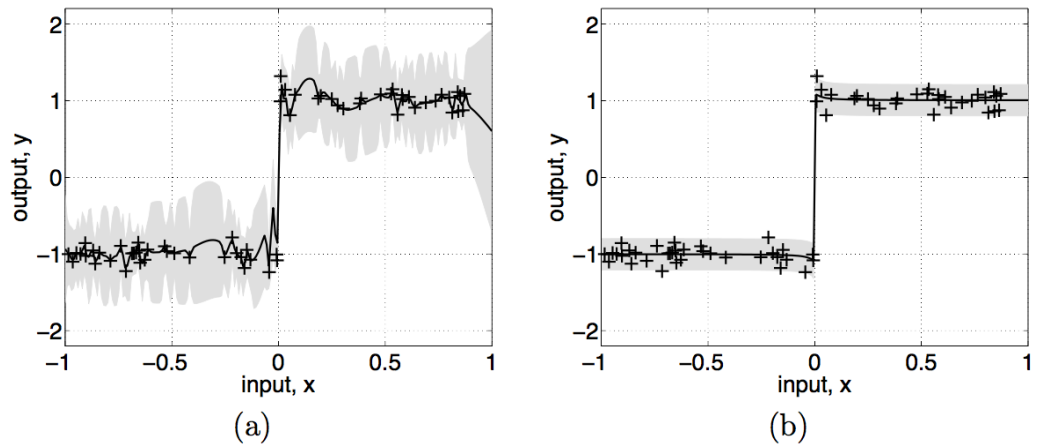
Advanced Kernels: Neural Network Kernel

Lastly, we introduce a famous non-stationary kernel, namely the neural network kernel:

$$k_{NN}(\boldsymbol{x}, \boldsymbol{x}') = \frac{2}{\pi} \sin^{-1}\left(\frac{2\tilde{\boldsymbol{x}}\Sigma\tilde{\boldsymbol{x}}'}{\sqrt{(1+2\tilde{\boldsymbol{x}}\Sigma\tilde{\boldsymbol{x}})(1+2\tilde{\boldsymbol{x}}'\Sigma\tilde{\boldsymbol{x}}')}}\right), \tag{19}$$

where $\tilde{\boldsymbol{x}} \triangleq [1, \boldsymbol{x}^T]^T$ is an augmented input vector. Often $\Sigma = diag(\sigma_1^2, \sigma_2^2)$ is a diagonal matrix and thus $\boldsymbol{\theta}_h = [\sigma_1^2, \sigma_2^2]^T$.

Non-stationary kernel is capable of providing better modeling in terms of abrupt change and saturation effect in the data, see the following example.



Reference

1. C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
2. C. Rasmussen, Gaussian Process for Machine Learning, MIT press, 2006.
3. J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.
4. A. G. Wilson, Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation with Gaussian Processes, Ph.D. Thesis, University of Cambridge, 2014.
5. D. Duvenaud, The Kernel Cookbook. http://www.cs.toronto.edu/ duvenaud/cookbook/

Outline

Parametric vs Nonparametric Models [Z. Ghahramani, 15]

- Parametric models assume some finite set of parameters $\boldsymbol{w}$. Given the parameters, future prediction, y_*, is independent of the observed data, therefore $\boldsymbol{w}$ captures everything there is to know about the data.

- Nonparametric models assume that the data distribution cannot be defined in terms of such a finite set of parameters. But they can be defined by assuming an infinite dimensional $\boldsymbol{w}$. Usually we think of $\boldsymbol{w}$ as a function. The amount of information that $\boldsymbol{w}$ can capture about the observed data can grow as the amount of data grows.
- Nonparametric models: A really large parametric model in practice [Teh, 11].

Bayesian Nonparametric Models

A Bayesian nonparametric model is a Bayesian model introduced on an infinite-dimensional parameter space.

The parameter space is typically chosen as the set of all possible solutions for a given learning problem, e.g.,

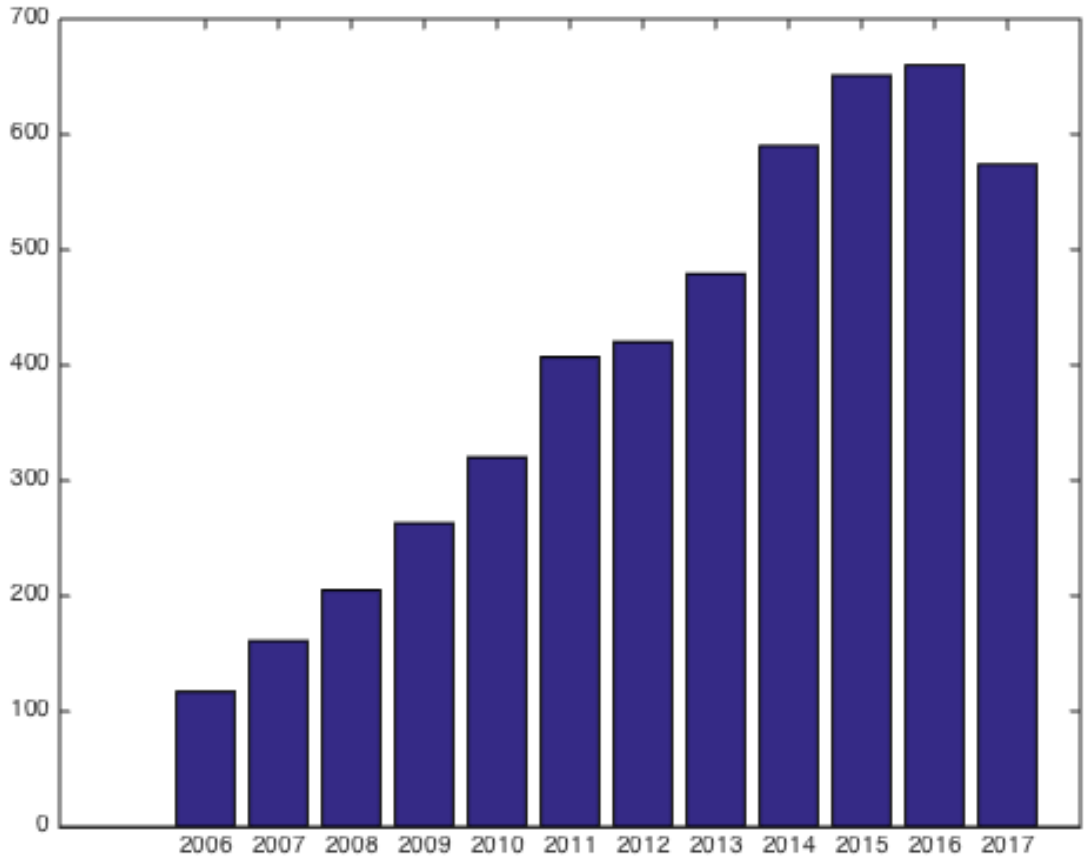
- for regression problem, the parameter space can be the set of continuous functions;
- for density estimation problem, the parameter space can consist of all densities.

Examples:

Parametric	Non-parametric	Application
polynomial regression	Gaussian processes	function approx.
logistic regression	Gaussian process classifiers	classification
mixture models, k-means	Dirichlet process mixtures	clustering

Some examples of Bayesian non-parametric models [Z. Ghahramani, 15].

Gaussian Process (GP)

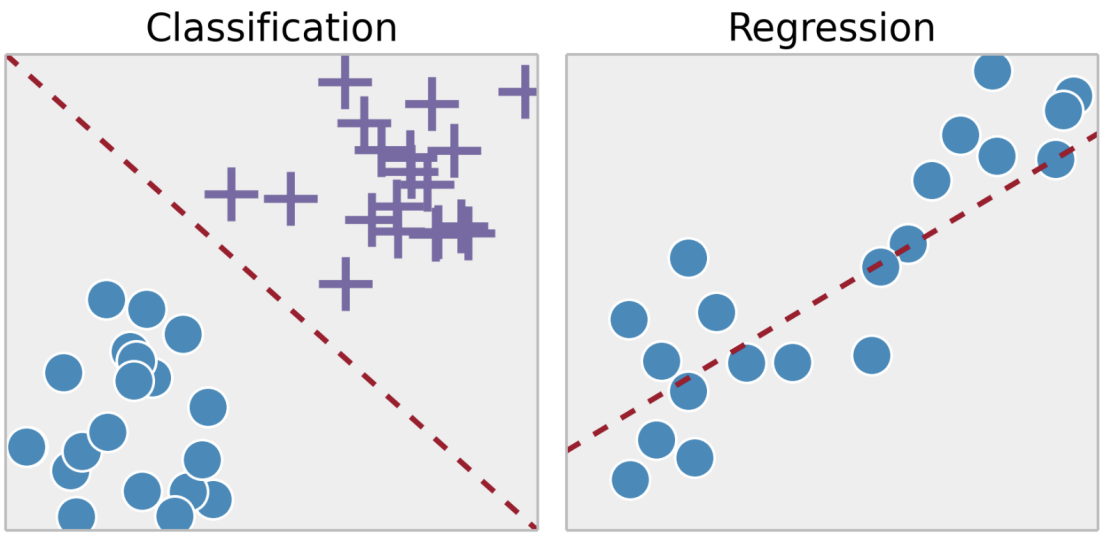


Number of publications on GP in engineering fields, https://www.engineeringvillage.com

Gaussian Process

In the following, Gaussian process will be viewed from a different angle and used as advanced Bayesian tools for

- nonlinear regression
- classification
- clustering (less seen)



Gaussian Process

Definition [Rasmussen and Williams, 06]

A Gaussian process is a collection of random variables, any finite number of which have Gaussian distributions.

We focus on real-valued Gaussian process, which is completely specified by a mean function and a covariance function (a.k.a. kernel function), i.e.,

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_h)), \tag{20}$$

where

- $m(\boldsymbol{x})$ is the mean function, often set to zero in practice, especially when there is no prior knowledge available.
- $k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_h)$ is the covariance/kernel function controlled by the hyper-parameters $\boldsymbol{\theta}_h$.

Gaussian Process for Regression

We consider the general regression model

$$y = f(\boldsymbol{x}) + e, \tag{21}$$

where

- the output y is continuous-valued;
- the underlying regression function $f(\boldsymbol{x})$ is assumed to be a real-valued zero-mean Gaussian process;
- and additive i.i.d. Gaussian noise with variance σ^2 is assumed;
- the noise terms are mutually independent.

The set of all unknown parameters is denoted by $\boldsymbol{\theta} = [\boldsymbol{\theta}_h, \sigma^2]^T$.

Gaussian Process for Regression

Given a training dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ where $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$ contains the training outputs and $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]$ contains the training inputs, we aim to predict $\boldsymbol{y}_* = [y_{*,1}, y_{*,2}, ..., y_{*,n_*}]^T$ given test inputs $\boldsymbol{X}_* = [\boldsymbol{x}_{*,1}, \boldsymbol{x}_{*,2}, ..., \boldsymbol{x}_{*,n_*}]$ with a posterior distribution $p(\boldsymbol{y}_*|\mathcal{D}, \boldsymbol{X}_*; \boldsymbol{\theta})$.

Special note on the notation: sometimes we write $\boldsymbol{X}$ and/or $\boldsymbol{x}_*$ in the conditional probabilities just to clarify which input correspond to which output, but they are essentially deterministic quantities and can be omitted in the conditional probability expressions.

Gaussian Process for Regression

Why GP is a Bayesian Nonparametric Model?

- A Gaussian prior is selected for deriving the posterior distribution of a desired metric, where Bayes theorem naturally comes into play.
- $f(\boldsymbol{x})$ can be seen as an infinite dimensional parameters $\boldsymbol{w}$.
- As the number of the observed data $(\boldsymbol{x}_i, y_i)$ grows, we get to know the model $f(\boldsymbol{x})$ better.
- Consequently, the posterior distribution out of the GP model is refined as the number of observed data grows!

Gaussian Process for Regression

In light of the definition of Gaussian process given beforehand, the joint prior distribution of the training output $\boldsymbol{y}$ and test output $\boldsymbol{y}_*$ can be written explicitly as:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_n & \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) + \sigma^2 \boldsymbol{I}_{n_*} \end{bmatrix}\right), \tag{22}$$

where

- $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*)$ is an $n \times n_*$ matrix of the covariance evaluated at all pairs of training and test points.
- $\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X})$, $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*)$ that are of size $n_* \times n$, $n \times n$, and $n_* \times n_*$, respectively.
- $\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*)^T$.

Gaussian Process for Regression

To derive the posterior distribution $p(\boldsymbol{y}_*|\mathcal{D}, \boldsymbol{X}_*, \boldsymbol{\theta})$, we require:

Conditional Gaussian Distribution

If the two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are jointly multivariate Gaussian as

$$p(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}\right), \tag{23}$$

then it is easy to derive the following conditional probabilities

$$p(\boldsymbol{x}|\boldsymbol{y}) \sim \mathcal{N}(\boldsymbol{\mu}_{x|y}, \Sigma_{x|y}), \qquad p(\boldsymbol{y}|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_{y|x}, \Sigma_{y|x}), \tag{24}$$

where

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \Sigma_{xy}\Sigma_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y), \qquad \boldsymbol{\mu}_{y|x} = \boldsymbol{\mu}_y + \Sigma_{yx}\Sigma_{xx}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x), \tag{25}$$

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}, \qquad \Sigma_{y|x} = \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}. \tag{26}$$

Gaussian Process for Regression

Applying the above result to our GP regression model, we thus obtain

$$p(\boldsymbol{y}_*|\mathcal{D}, \boldsymbol{X}_*; \boldsymbol{\theta}_h) \sim \mathcal{N}\left(\bar{\boldsymbol{m}}, \bar{\boldsymbol{V}}\right), \tag{27}$$

where

$$\bar{\boldsymbol{m}} \triangleq \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X})\left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{y}, \tag{28}$$

$$\bar{\boldsymbol{V}} \triangleq \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) + \sigma^2 \boldsymbol{I}_{n_*} - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X})\left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*). \tag{29}$$

Remark: for ease of our narration on the inference, the parameters $\boldsymbol{\theta}$ is assumed to be known! Later, we will mention how to tune these parameters.

Gaussian Process for Regression

For the simplest case where there is only one test input $\boldsymbol{x}_*$, the posterior distribution is given by

$$p(y_*|\mathcal{D}, \boldsymbol{x}_*; \boldsymbol{\theta}) \sim \mathcal{N}(\bar{m}, \bar{v}). \tag{30}$$

where the posterior mean and variance are respectively

$$\bar{m} \triangleq \boldsymbol{k}_* \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{y}, \tag{31}$$

$$\bar{v} \triangleq k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \boldsymbol{k}_*^T \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{k}_*. \tag{32}$$

Here, $\boldsymbol{k}_* \triangleq [k(\boldsymbol{x}_*, \boldsymbol{x}_1), k(\boldsymbol{x}_*, \boldsymbol{x}_2), ..., k(\boldsymbol{x}_*, \boldsymbol{x}_n)]^T$ is the vector of covariances between the test point and the n training points in $\boldsymbol{X}$.

Gaussian Process for Regression

Some interpretations of the posterior mean and posterior variance (for one test point case) are as follows:

- The posterior mean can be written as $\bar{m} = \boldsymbol{\beta}^T \boldsymbol{y}$, which is a linear combination of the observations, often referred as linear predictor.
- The posterior variance

$$\bar{v} \triangleq k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \boldsymbol{k}_*^T \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{k}_* \tag{33}$$

  is the **difference** between the variance of the prior and the variance explained from the training data $\mathcal{D}$.

Gaussian Process for Classification

- In classification problems, we wish to assign a new input $\boldsymbol{x}_*$ to one of C classes, $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_C$.
- We focus on the probabilistic classification based on Gaussian processes, where test predictions take the form of class probabilities.
- In the sequel, we narrow down our focus to discriminative approach for binary classification, i.e., the case $C = 2$.

Gaussian Process for Classification

We start with a representative parametric binary classification model, namely the linear logistic regression model:

$$p(y = +1|\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(\boldsymbol{x}^T \boldsymbol{w}\right), \tag{34}$$

$$p(y = -1|\boldsymbol{x}, \boldsymbol{w}) = 1 - p(y = +1|\boldsymbol{x}, \boldsymbol{w}), \tag{35}$$

where $\sigma(z) = \frac{1}{1+\exp(-z)}$ is the logistic function. However, $\sigma(z)$ may also take other Sigmoidal type functions.

For symmetric functions satisfying $\sigma(z) = 1 - \sigma(-z)$, the likelihood can be written compactly as

$$p(y_i = +1|\boldsymbol{x}_i, \boldsymbol{w}) = \sigma(y_i f_i), \tag{36}$$

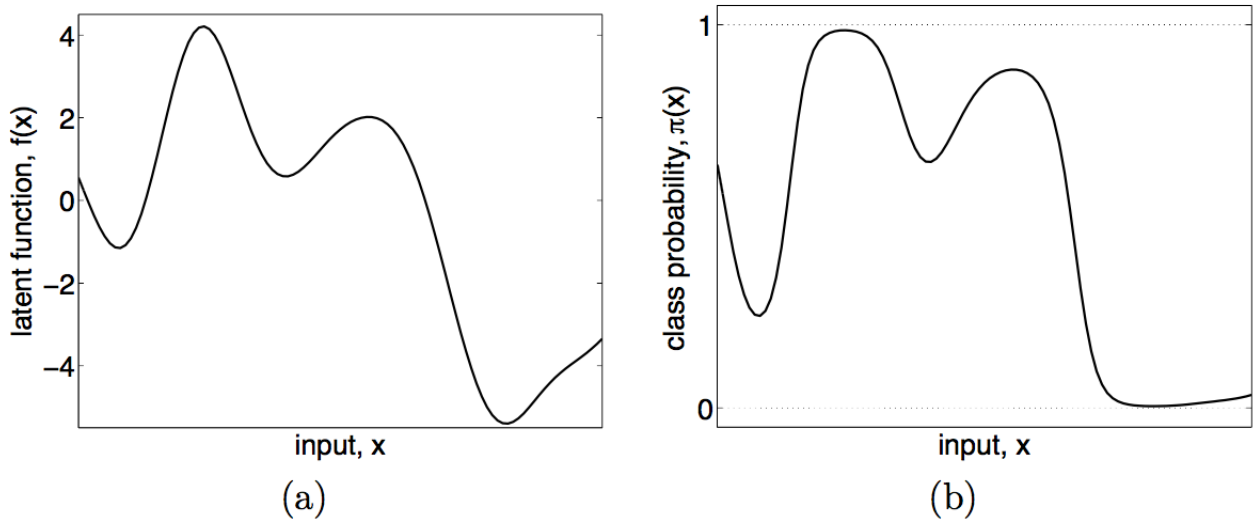
where $f_i \triangleq f(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \boldsymbol{w}$.

Gaussian Process for Classification

Instead of working with the weighting factors $\boldsymbol{w}$, Gaussian process classification introduces

$$\pi(\boldsymbol{x}) \triangleq p(y = +1|f(\boldsymbol{x})) = \sigma\left(f(\boldsymbol{x})\right), \tag{37}$$

where $f(\boldsymbol{x})$ is assumed to be a Gaussian process.



Panel (a) shows a sample latent function $f(x)$ drawn from a Gaussian process; Panel (b) shows the transformation of this sample function through $\sigma\left(f(x)\right)$.

Gaussian Process for Classification

In GP classification, we are particularly interested in the class probability $\bar{\pi}_*$ for novel test point $\boldsymbol{x}_*$ given the dataset $\mathcal{D}$, concretely,

$$\bar{\pi}_* \triangleq p(y_* = +1|\boldsymbol{x}_*, \mathcal{D}) = \int \sigma(f_*)\, p(f_*|\boldsymbol{x}_*, \mathcal{D})\, df_*, \tag{38}$$

where

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y})\, d\boldsymbol{f}, \tag{39}$$

and $f_* \triangleq f(\boldsymbol{x}_*)$, $\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_n)]^T$.

1. The probability $p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X})$ can be computed using similar technique applied for GP regression.
2. The probability $p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}) / p(\boldsymbol{y}|\boldsymbol{X})$ is the posterior over the latent variables $\boldsymbol{f}$.

Gaussian Process for Classification

Difficulties with GP classification:

- The non-Gaussian distribution $p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y})$ makes the integral in Eq.(39) analytically intractable in general.
- The integral in Eq.(38) can be also analytically intractable for some Sigmoid function.

Solution:

1. Approximate $p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y})$ with a Gaussian distribution using e.g.,
   - Laplace approximation [Williams and Baber, 1998], see Appendix
   - Expectation propagation [Minka, 2001]
   - Variational approximation [Opper, 2009]
2. or Markov chain Monte Carlo (MCMC) [Neal, 1999]

Gaussian Process for Classification

Laplace approximation of $p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})$ is done by Taylor expansion of $\log p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})$ around the maximum of the posterior, yielding

$$q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y}) = \mathcal{N}(\boldsymbol{f}|\hat{\boldsymbol{f}},\boldsymbol{A}^{-1}) \tag{40}$$

where

$$\hat{\boldsymbol{f}} = \arg\max_{\boldsymbol{f}} p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y}), \tag{41}$$

$$\boldsymbol{A} = -\nabla_{\boldsymbol{f}}\nabla_{\boldsymbol{f}} \log p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})|_{\boldsymbol{f}=\hat{\boldsymbol{f}}}, \tag{42}$$

are respectively the maximum of the posterior and the Hessian of the negative log posterior at the maximum. Details see Appendix.

The Hessian $\boldsymbol{A}$ can be further expressed as $\boldsymbol{A} = \boldsymbol{W} + \boldsymbol{K}^{-1}$, where $\boldsymbol{K}$ is short for $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})$ and $\boldsymbol{W}$ is a diagonal matrix with

$$\boldsymbol{W} \triangleq -\nabla_{\boldsymbol{f}}\nabla_{\boldsymbol{f}} \log p(\boldsymbol{y}|\boldsymbol{f})|_{\boldsymbol{f}=\hat{\boldsymbol{f}}}. \tag{43}$$

Gaussian Process for Classification

Now, we come to the inference part.

The exact posterior mean for a novel test point $\boldsymbol{x}_*$ is

$$\mathbb{E}_p\left[f_*|\boldsymbol{x}_*,\mathcal{D}\right] = \int f_* p(f_*|\boldsymbol{x}_*,\mathcal{D})df_* \tag{44}$$

$$= \int \mathbb{E}\left[f_*|\boldsymbol{f},\boldsymbol{X},\boldsymbol{x}_*\right] p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})d\boldsymbol{f} \tag{45}$$

$$= \int \boldsymbol{k}(\boldsymbol{x}_*)^T\boldsymbol{K}^{-1}\boldsymbol{f}\, p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})d\boldsymbol{f} \tag{46}$$

$$= \boldsymbol{k}(\boldsymbol{x}_*)^T\boldsymbol{K}^{-1}\mathbb{E}\left[\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y}\right], \tag{47}$$

where line 1 to 2 is due to $p(f_*|\boldsymbol{x}_*,\boldsymbol{X},\boldsymbol{y}) = \int p(f_*|\boldsymbol{f},\boldsymbol{x}_*,\boldsymbol{X})p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})\,d\boldsymbol{f}$.

We obtain the Laplace approximated posterior mean as

$$\mathbb{E}_q\left[f_*|\boldsymbol{x}_*,\mathcal{D}\right] = \boldsymbol{k}(\boldsymbol{x}_*)^T\boldsymbol{K}^{-1}\hat{\boldsymbol{f}}. \tag{48}$$

Gaussian Process for Classification

The exact posterior variance for a novel test point $\boldsymbol{x}_*$ is

$$Var_p\left[f_*|\boldsymbol{x}_*,\mathcal{D}\right] = \int \left(f_* - \mathbb{E}_p\left[f_*|\boldsymbol{x}_*,\mathcal{D}\right]\right)^2 p(f_*|\boldsymbol{x}_*,\mathcal{D})df_* \tag{49}$$

$$= \mathbb{E}_{p(f_*|\boldsymbol{f},\boldsymbol{x}_*,\boldsymbol{X})}\left[\left(f_* - \mathbb{E}\left[f_*|\boldsymbol{f},\boldsymbol{X},\boldsymbol{x}_*\right]\right)^2\right]$$

$$+ \mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})}\left[\left(\mathbb{E}\left[f_*|\boldsymbol{f},\boldsymbol{X},\boldsymbol{x}_*\right] - \mathbb{E}\left[f_*|\boldsymbol{y},\boldsymbol{X},\boldsymbol{x}_*\right]\right)^2\right] \tag{50}$$

We obtain the Laplace approximated posterior variance as

$$Var_q\left[f_*|\boldsymbol{x}_*,\mathcal{D}\right] = \mathbb{E}_{p(f_*|\boldsymbol{f},\boldsymbol{x}_*,\boldsymbol{X})}\left[\left(f_* - \mathbb{E}\left[f_*|\boldsymbol{f},\boldsymbol{X},\boldsymbol{x}_*\right]\right)^2\right]$$

$$+ \mathbb{E}_{q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})}\left[\left(\mathbb{E}\left[f_*|\boldsymbol{f},\boldsymbol{X},\boldsymbol{x}_*\right] - \mathbb{E}\left[f_*|\boldsymbol{y},\boldsymbol{X},\boldsymbol{x}_*\right]\right)^2\right] \tag{51}$$

$$= k(\boldsymbol{x}_*,\boldsymbol{x}_*) - \boldsymbol{k}(\boldsymbol{x}_*)^T\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{x}_*)$$

$$+ \boldsymbol{k}(\boldsymbol{x}_*)^T\boldsymbol{K}^{-1}\left(\boldsymbol{K}^{-1} + \boldsymbol{W}\right)^{-1}\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{x}_*) \tag{52}$$

$$= k(\boldsymbol{x}_*,\boldsymbol{x}_*) - \boldsymbol{k}(\boldsymbol{x}_*)^T\left(\boldsymbol{K} + \boldsymbol{W}^{-1}\right)^{-1}\boldsymbol{k}(\boldsymbol{x}_*). \tag{53}$$

Detailed derivations of the above steps can be found in the Appendix.

Gaussian Process for Classification

Given an approximation of the posterior distribution $p(f_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*)$, i.e., $q(f_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*)$ which is Gaussian distributed with its mean and variance derived in the previous two slides, we have the probability of having class 1 approximately equal to:

$$\bar{\pi}_* \triangleq p(y_* = +1|\boldsymbol{x}_*, \mathcal{D}) = \int \sigma(f_*)\, p(f_*|\boldsymbol{x}_*, \mathcal{D})\, df_*$$
$$\approx \int \sigma(f_*) q(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) df_*. \tag{54}$$

When $\sigma(z)$ takes the logistic function, the above one-dimensional integral has to be evaluated numerically.

Reference

1. T. P. Minka, A Family of Algorithms for Approximate Bayesian Inference. PhD thesis, Massachusetts Institute of Technology, 2001.
2. C. K. I. Williams and D. Barber, Bayesian Classification with Gaussian Processes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12): 1342–1351, 1998.
3. R. M. Neal, Regression and Classification using Gaussian Process Priors. In Bayesian Statistics 6, pages 475–501. Oxford University Press. 1999.
4. M. Opper and C. Archambeau, The variational Gaussian approximation revisited. Neural computation, 21(3): 786–792, 2009.

Hyper-parameter Optimization: ML Method

The predictive performance of GP regression and classification depends on the goodness of a set of hyper-parameters.

The following methods can be used for tuning the hyper-parameters:

1. deterministic methods including:
   - maximum likelihood (ML)
   - cross-validation
   - alignment score (mainly for GP classification)
2. stochastic methods including:
   - hybrid Monte-Carlo (HMC) [Rasmussen 96, Neal 97].
   - slice sampling

Hyper-parameter Optimization: ML Method

The dominant method for tuning the hyper-parameters is via maximizing the marginal likelihood function $p(\boldsymbol{y};\boldsymbol{\theta})$, where

$$p(\boldsymbol{y};\boldsymbol{\theta}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})d\boldsymbol{f}. \tag{55}$$

Recall that

1. for the GP regression, we have $y_i = f(\boldsymbol{x}_i) + n_i$, where n_i is assumed to be Gaussian noise;
2. and for the GP classification, we have $y_i = \sigma(f(\boldsymbol{x}_i))$, and $\sigma(z)$ is a non-linear mapping,

and $f(\boldsymbol{x})$ is assumed to be a Gaussian process in both cases.

Hyper-parameter Optimization: ML Method

For GP regression, due to the Gaussian assumption on the noise, the log-marginal likelihood function can be obtained in closed form as

$$\log p(\boldsymbol{y};\boldsymbol{\theta}) = -\frac{1}{2}\left\{\log\det\left(\boldsymbol{C}(\boldsymbol{\theta})\right) + \boldsymbol{y}^T\boldsymbol{C}^{-1}(\boldsymbol{\theta})\boldsymbol{y} + n\log(2\pi)\right\} \tag{56}$$

where $\boldsymbol{C}(\boldsymbol{\theta}) \triangleq \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X};\boldsymbol{\theta}_h) + \sigma^2\boldsymbol{I}_n$, and $\boldsymbol{\theta} = [\boldsymbol{\theta}_h^T, \sigma^2]^T$.

The hyper-parameter is tuned equivalently by minimizing the negative log-likelihood function. In the GP society, this is often solved using the gradient based algorithms, such as LFGS-Newton, which requires the gradient:

$$\frac{\partial - \log p(\boldsymbol{y};\boldsymbol{\theta})}{\partial\theta_i} = \frac{1}{2}tr\left(\boldsymbol{C}^{-1}(\boldsymbol{\theta})\frac{\partial\boldsymbol{C}(\boldsymbol{\theta})}{\partial\theta_i}\right) - \frac{1}{2}\boldsymbol{y}^T\boldsymbol{C}^{-1}(\boldsymbol{\theta})\frac{\partial\boldsymbol{C}(\boldsymbol{\theta})}{\partial\theta_i}\boldsymbol{C}^{-1}(\boldsymbol{\theta})\boldsymbol{y}. \tag{57}$$

Hyper-parameter Optimization: ML Method

For GP classification, due to the nonlinear mapping $\sigma(y_i f_i)$, the marginal likelihood can not be otained in closed form in general.

Since

$$p(\boldsymbol{y};\boldsymbol{\theta}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})d\boldsymbol{f} = \int \exp\left(\Psi(\boldsymbol{f})\right) d\boldsymbol{f}, \tag{58}$$

where $\Psi(\boldsymbol{f}) \triangleq \log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f})$. Perform Taylor expansion of $\Psi(\boldsymbol{f})$ at the maximum of the posterior $\hat{\boldsymbol{f}}$, an approximated log-likelihood function is then obtained as

$$\log q(\boldsymbol{y};\boldsymbol{\theta}) = -\frac{1}{2}\hat{\boldsymbol{f}}^T \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X};\boldsymbol{\theta}_h)^{-1}\hat{\boldsymbol{f}} + \log p(\boldsymbol{y}|\hat{\boldsymbol{f}}) - \frac{1}{2}\log\det(\boldsymbol{B}), \tag{59}$$

where $\det(\boldsymbol{B}) = \det(\boldsymbol{I}_n + \boldsymbol{W}^{\frac{1}{2}}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X};\boldsymbol{\theta}_h)\boldsymbol{W}^{\frac{1}{2}})$.
Gradient can be obtained similarly as the regression case.

Hyper-parameter Optimization: ML Method

We know that the gradient based algorithms are iterative and at each iteration, in general,

- The computational complexity for evaluating the gradient of the negative marginal likelihood function requires the matrix inverse $\boldsymbol{C}^{-1}(\boldsymbol{\theta})$ or $\boldsymbol{K}^{-1}(\boldsymbol{\theta})$, scales as $\mathcal{O}(n^3)$.
- Once $\boldsymbol{C}^{-1}(\boldsymbol{\theta})$ or $\boldsymbol{K}^{-1}(\boldsymbol{\theta})$ is obtained, the computation of the derivatives scales as $\mathcal{O}(n^2)$ per hyper-parameter.

When the dimension of $\boldsymbol{y}$ is high and the number of hyper-parameters to be optimized is also high, better strategy for parameter optimization is sought after.

Hyper-parameter Optimization: Monte Carlo Method

Another way of dealing with the hyper-parameters is to use Monte Carlo method. Principally, the posterior distribution is calculated by

$$p(\boldsymbol{y}_*|\mathcal{D}) = \int_{\boldsymbol{\theta}\in\Theta} p(\boldsymbol{y}_*|\mathcal{D},\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \tag{60}$$

The Monte Carlo method uses sampling methods to get an approximated posterior distribution via

$$p(\boldsymbol{y}_*|\mathcal{D}) \approx \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} p(\boldsymbol{y}_*|\mathcal{D},\boldsymbol{\theta}_i), \tag{61}$$

where $\boldsymbol{\theta}_i$, $i = 1, 2, ..., N_{MC}$ are samples drawn from the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$.

Hyper-parameter Optimization: Monte Carlo Method

Some remarks are:

- The Monte Carlo approximation of the posterior distribution is a Gaussian mixture;
- The approximation accuracy will in general increase as we take more samples from $p(\boldsymbol{\theta}|\mathcal{D})$;
- The method that we use to sample from $p(\boldsymbol{\theta}|\mathcal{D})$ impacts the approximation accuracy.
- Low complex and efficient sampling method for high-dimensional space is still under development. Some classic work can be found in [Duane 87, Rasmussen 96, Neal 97].

Summary

- Bayesian nonparametric model enables good model flexibility;
- Gaussian noise assumption and factorizing property of the likelihood function largely simplify the derivations;
- Computational complexity for training the hyper-parameters is quite high.

Reference

1. C. E. Rasmussen, Evaluation of Gaussian Processes and Other Methods for Nonlinear Regression. PhD thesis, Dept. of Computer Science, University of Toronto. 1996.
2. R. M. Neal, Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Department of Statistics, University of Toronto. 1997.

GPR Example: Mauna Loa Atmospheric Carbon Dioxide

Review: Given the training dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ and a novel test data point $\boldsymbol{x}_*$, the GPR method does the following to predict y_*:

1. Select a suitable kernel, stationary or non-stationary, according to the prior knowledge about the data.
2. Maximize the log-marginal likelihood function in Eq.(56) to get
   - an optimal set of parameters $\boldsymbol{\theta}$,
   - the inverse of the covariance matrix $\boldsymbol{C}(\boldsymbol{\theta}) \triangleq \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta}_h) + \sigma^2 \boldsymbol{I}_n$.
3. Compute the posterior/predictive mean and variance according to Eq.(31) and Eq.(32).

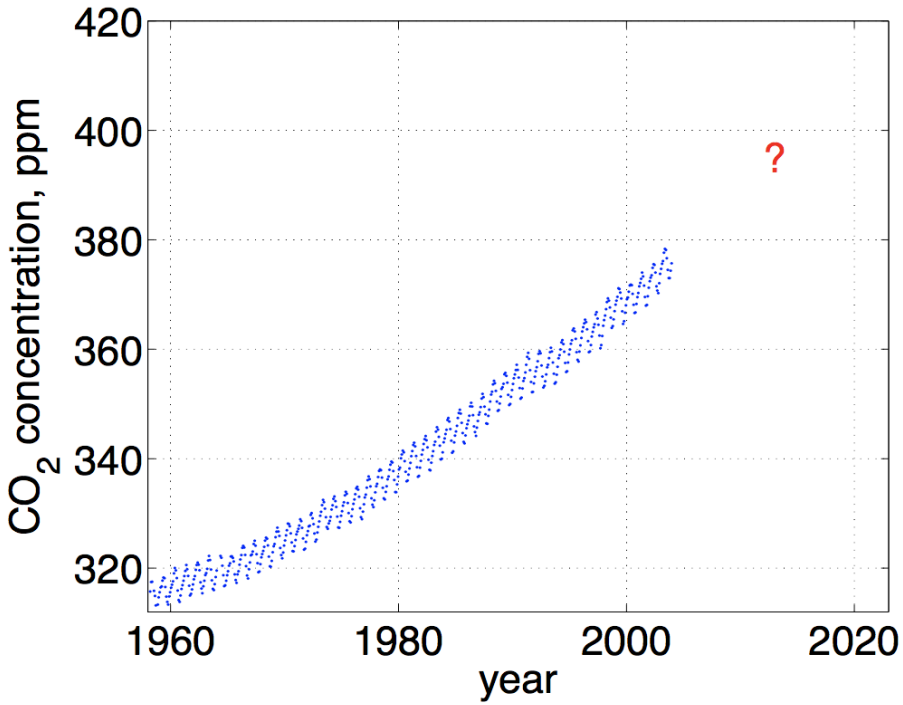
GPR Example: Data Description

The data [Keeling and Whorf, 2004] consists of monthly average atmospheric CO2 concentrations (unit: per million by volume (ppmv)) derived from in-situ air samples collected at the Mauna Loa Observatory, Hawaii, between 1958 and 2003.

Patterns seen from the data:

- long term rising trend
- seasonal variation
- smaller irregularities
- noise

Task: predict beyond 2003.



GPR Example: Kernel Selection

Rasmussen and Williams suggest a composite kernel:

$$k(x, x^{'}; \boldsymbol{\theta}_h) = k_1(x, x^{'}) + k_2(x, x^{'}) + k_3(x, x^{'}) + k_4(x, x^{'}) \tag{62}$$

where

- long-term (smooth) rising trend addressed by the SE kernel

  $$k_1(x, x^{'}) = \theta_1^2 \exp\left[\frac{-(x-x^{'})^2}{\theta_2^2}\right]$$

- seasonal trend addressed by the LP kernel

  $$k_2(x, x^{'}) = \theta_3^2 \exp\left[\frac{-2\sin^2(\pi(x-x^{'}))}{\theta_5^2}\right] \times \exp\left[\frac{-(x-x^{'})^2}{2\theta_4^2}\right]$$

- smaller irregularities/anomaly addressed by the RQ kernel

  $$k_3(x, x^{'}) = \theta_6^2 \left(1 + \frac{(x-x^{'})^2}{2\theta_8\theta_7^2}\right)^{-\theta_8}$$

- and noise addressed by another SE kernel

  $$k_4(x, x^{'}) = \theta_9^2 \exp\left[\frac{-(x-x^{'})^2}{\theta_{10}^2}\right]$$

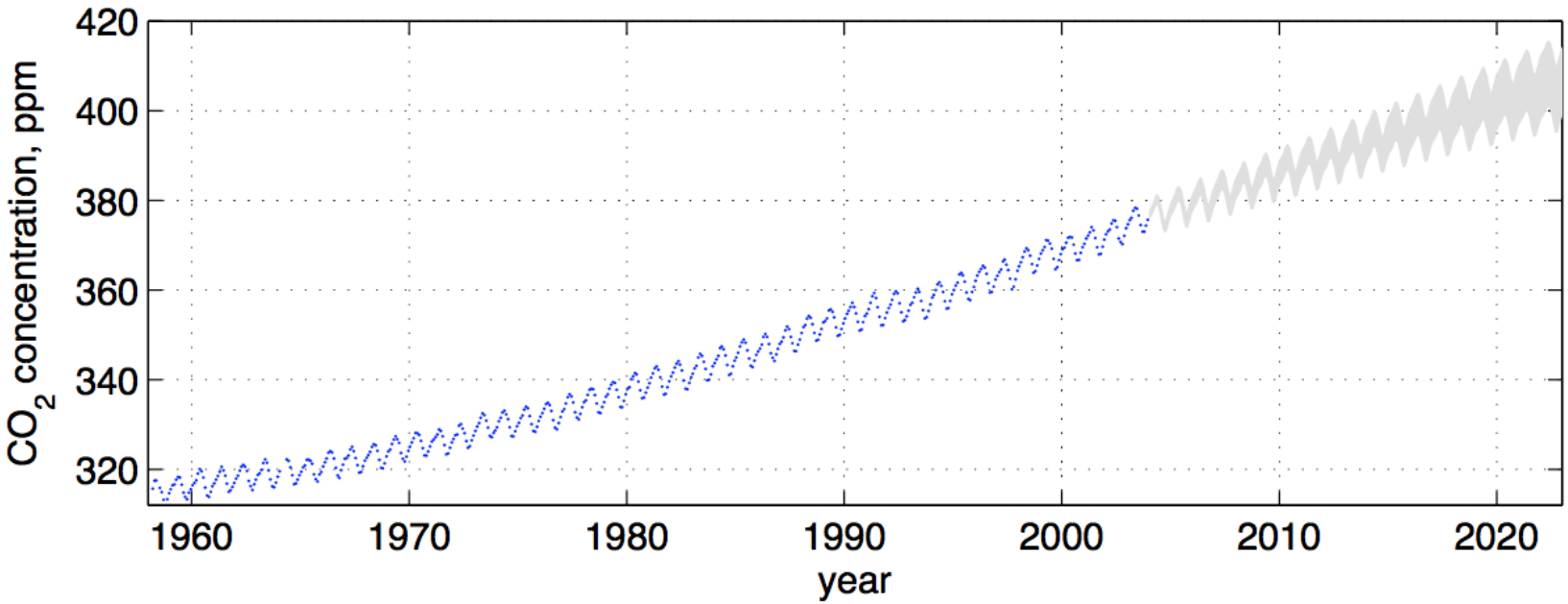
GPR Example: Hyper-parameter Optimization

Hyperparameter optimization:

- Empirical mean of the data (341 ppm) is first subtracted from the data to have a zero-mean Gaussian process.
- The parameters $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_{10}, \sigma^2]^T$ are tuned by maximizing the log-marginal likelihood using conjugate gradient method.
- Random restarts are tried and pick the run with the best log-marginal likelihood, which is $\log p(\boldsymbol{y}|\boldsymbol{\theta}) = -108.5$.

GPR Example: Inference

Prediction performance using the above composite kernel:



The 545 observations of monthly averages of the atmospheric concentration of CO2 made between 1958 and the end of 2003, together with 95% predictive confidence region for a GP regression model, 20 years into the future.

GPC Example: Binary Handwritten Digit Classification

Review: Given the training dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ and a novel test data point $\boldsymbol{x}_*$, the binary GPC method does the following to obtain $p(y_* = 1|\mathcal{D})$ and the final label:

1. Select a suitable kernel, stationary or non-stationary, according to the prior knowledge about the data.
2. Select an initial guess of the hyper-parameters $\boldsymbol{\theta}_h^{(0)}$. Do the following steps until some convergence condition is met:
   - Update the maximum of the posterior $\hat{\boldsymbol{f}}$ at the current set of hyper-parameters $\boldsymbol{\theta}_h^{(i)}$ in the i-th iteration according to Eq.(41).
   - Maximize the approximated log-marginal likelihood function to update $\boldsymbol{\theta}_h^{(i+1)}$ according to (59).
3. Compute the Laplace approximated posterior mean and variance according to Eq.(48) and Eq.(53).
4. If $p(y_* = 1|\mathcal{D}) > 0.5$, we go for label 1; otherwise label 0 for $\boldsymbol{x}_*$. Or alternatively, check the sign of the posterior mean.

GPC Example: Data Description

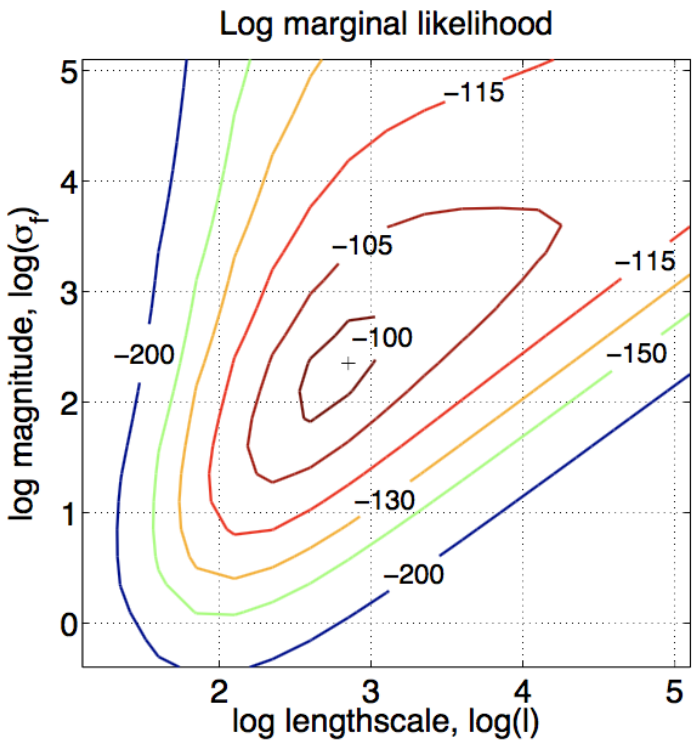
We consider the discrimination of images of the digit 3 from images of the digit 5 as an example of binary classification.

Data description:

- US Postal Service (USPS) database of handwritten digits as segmented 16×16 greyscale images.
- The images are normalized so that the intensity of the pixels lies in $[-1, 1]$.
- The 3s vs. 5s data has 767 training cases, split into 406/361, while the test set has 773 cases split into 418/355.

GPC Example: Hyper-parameter Optimization

- The SE kernel is selected, $k(x, x^{'}; \boldsymbol{\theta}_h) = \theta_f^2 \exp(-(x - x^{'})^2/l^2)$.
- An optimal set of hyper-parameters $\boldsymbol{\theta}_h = [\theta_f, l]^T$ is obtained by maximizing the Laplace approximated log-marginal likelihood function using the conjugate gradient method.

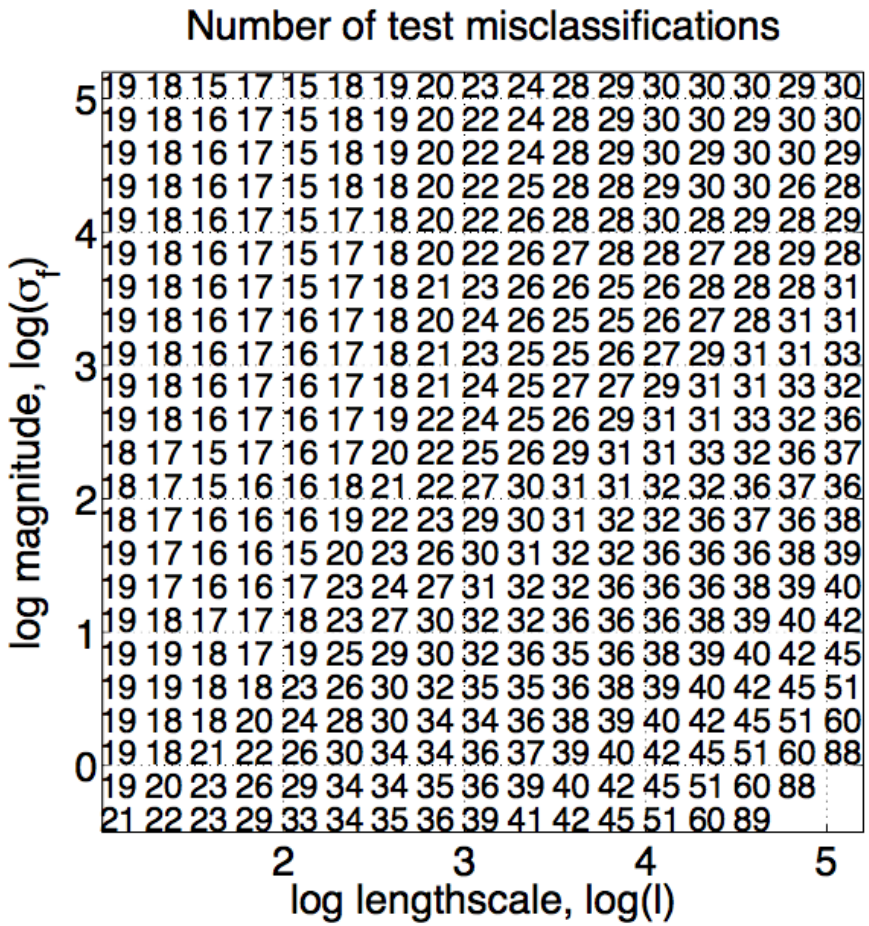


At the optimum:

- $\log(l) = 2.85$ and $\log(\sigma_f) = 2.35$
- $\log q(\boldsymbol{y}; \boldsymbol{\theta}) = -99$

GPC Example: Inference

The predictive performance in terms of the test errors:



the number of test errors (out of 773) when predicting using the sign of the posterior mean.

Computer Toolboxes

1. MATLAB:
   - built-in *Gaussian Process Regression Models* in Statistic and Machine Learning Toolbox, 2017b.
     `https://cn.mathworks.com/help/stats/gaussian-process-regression-models.html`
   - GPML Matlab Code version 4.0, written by C. Rasmussen,
     `http://www.gaussianprocess.org/gpml/code/matlab/doc/`
2. PYTHON:
   - Gaussian Processes package,
     `http://scikit-learn.org/stable/modules/gaussian_process.html`
3. JULIA:
   - Gaussian Processes package,
     `https://github.com/STOR-i/GaussianProcesses.jl`

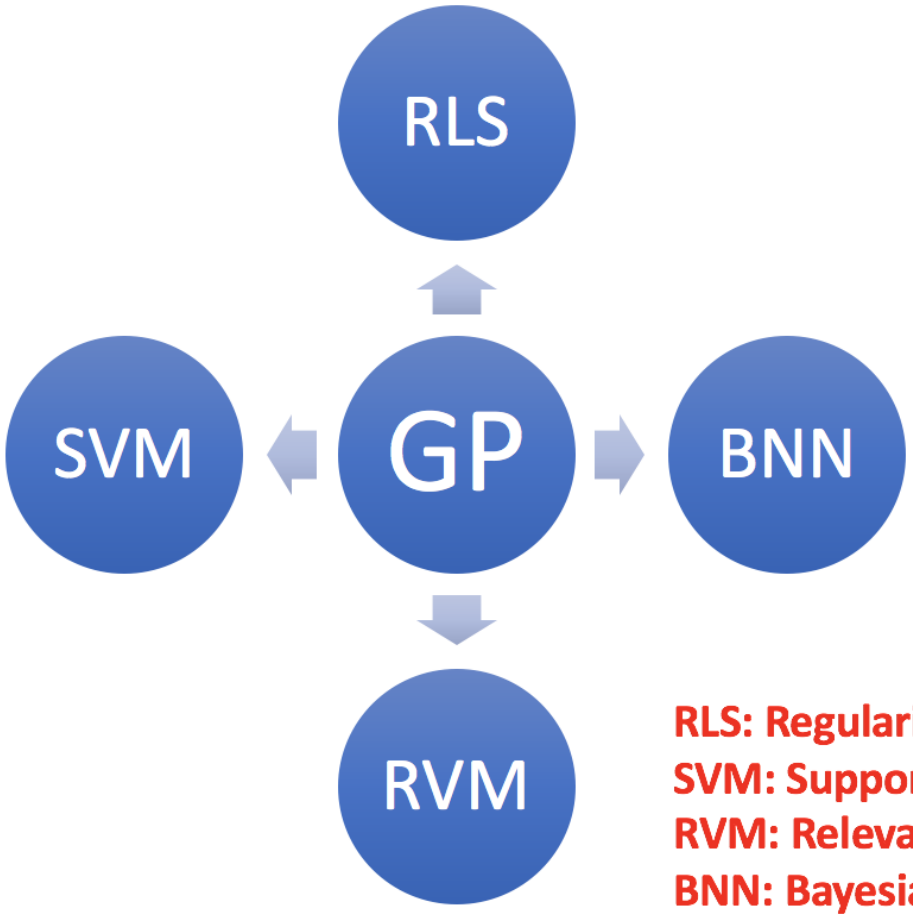
Reference

1. C. D. Keeling and T. P. Whorf, Atmospheric CO2 Records from Sites in the SIO Air Sampling Network. In Trends: A Compendium of Data on Global Change. Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, Oak Ridge, Tenn., U.S.A., 2004.

Outline

Overview

The relationships between the Gaussian Process (GP) models and other models are summarized in the figure below:

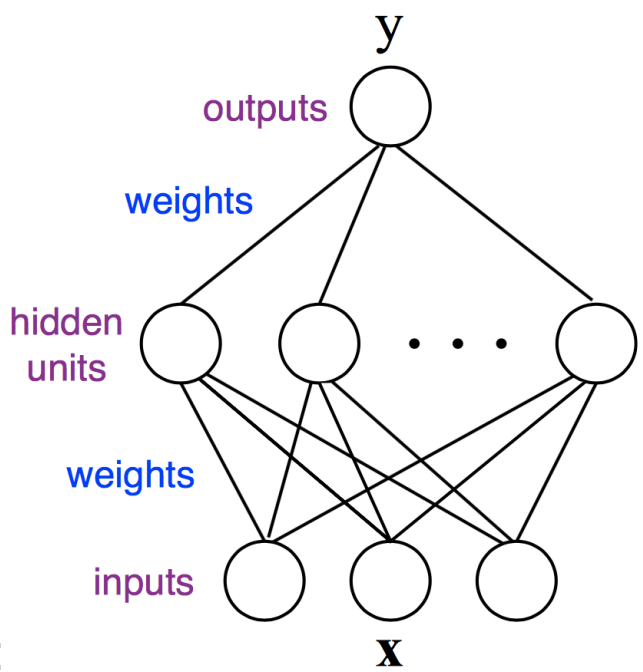


RLS: Regularized Least-Squares
SVM: Support Vector Machine
RVM: Relevance Vector Machine
BNN: Bayesian Neural Network

Relationship with Bayesian Neural Networks

Consider a neural network with one hidden layer of H units that takes an input $\boldsymbol{x} \in \mathbb{R}^p$. The mapping function can be written as:

$$f(\boldsymbol{x}) = b + \sum_{j=1}^{H} v_j h(\boldsymbol{x}; \boldsymbol{u}_j), \quad (63)$$



where

- v_j's are the hidden-to-output weights;
- $h(\boldsymbol{x}; \boldsymbol{u})$ is the hidden unit transfer/activation function (assume to be bounded) which depends on the input-to-hidden weights $\boldsymbol{u}$;
- b is a bias term.

Relationship with Bayesian Neural Networks

Assuming that

- b and v's have independent zero-mean distributions of variance σ_b^2 and σ_v^2, respectively;
- the weights $\boldsymbol{u}_j$ for each hidden unit be i.i.d..

Stacking all weights to obtain $\boldsymbol{w}$, we have

$$\mathbb{E}_{\boldsymbol{w}}\left[f(\boldsymbol{x})\right] = 0 \tag{64}$$

$$\mathbb{E}_{\boldsymbol{w}}\left[f(\boldsymbol{x})f(\boldsymbol{x}')\right] = \sigma_b^2 + \sigma_v^2 \sum_{j=1}^{H} \mathbb{E}_{\boldsymbol{u}}\left[h(\boldsymbol{x};\boldsymbol{u}_j)h(\boldsymbol{x}';\boldsymbol{u}_j)\right]. \tag{65}$$

Remark: The sum is over H i.i.d. RVs. As the transfer function is bounded, all moments of the distribution will be bounded and applying the CLT shows that $f(\boldsymbol{x})$ converges to a Gaussian process as $H \to \infty$.

Relationship with Bayesian Neural Networks

For the following Sigmoid transfer functions, the corresponding kernel function is well known:

- $h(\boldsymbol{x};\boldsymbol{u}) = \Phi\left(u_0 + \sum_{i=1}^{p} u_i x_i\right)$, where $\Phi(z) = \frac{2}{\sqrt{\pi}} \int_0^z \exp(-t^2)dt$ is the error function.
- Assume that $\boldsymbol{u}$ is drawn from a zero-mean Gaussian distribution with covariance matrix Σ. Furthermore, let $\tilde{\boldsymbol{x}} = [1, x_1, x_2, ..., x_p]^T$.
- The neural network kernel is derived then as

$$k_{NN}(\boldsymbol{x}, \boldsymbol{x}') \triangleq \mathbb{E}_{\boldsymbol{u}}\left[h(\boldsymbol{x}, \boldsymbol{u})h(\boldsymbol{x}', \boldsymbol{u})\right] \tag{66}$$

$$= \int \Phi(\boldsymbol{u}^T\tilde{\boldsymbol{x}})\Phi(\boldsymbol{u}^T\tilde{\boldsymbol{x}}')\mathcal{N}(\boldsymbol{u};\boldsymbol{0},\Sigma)d\boldsymbol{u} \tag{67}$$

$$= \frac{2}{\pi}\sin^{-1}\left(\frac{2\tilde{\boldsymbol{x}}^T\Sigma\tilde{\boldsymbol{x}}'}{\sqrt{(1+2(\tilde{\boldsymbol{x}})^T\Sigma\tilde{\boldsymbol{x}})(1+2(\tilde{\boldsymbol{x}}')^T\Sigma\tilde{\boldsymbol{x}}')}}\right) \tag{68}$$

Relationship with Support Vector Machine

This section aims to compare the Gaussian process classification with the support vector machine (SVM).

For the non-separable case, the soft margin SVM computes

$$\boldsymbol{w}_{SVM} = \arg\min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n} \max(1 - y_i f_i, 0) \tag{69}$$

where

- $f_i \triangleq \boldsymbol{w}^T \phi(\boldsymbol{x}_i)$;
- $\max(z, 0)$ return z, if $z > 0$ or zero otherwise;
- $C > 0$ is a regularization term.

Relationship with Support Vector Machine

This convex optimization problem can be solved using QP methods, e.g., the sequential minimal optimization (SMO) algorithm [Platt, 99] and yields a solution of the form $\sum_i^n \alpha_i \phi(\boldsymbol{x}_i)$.

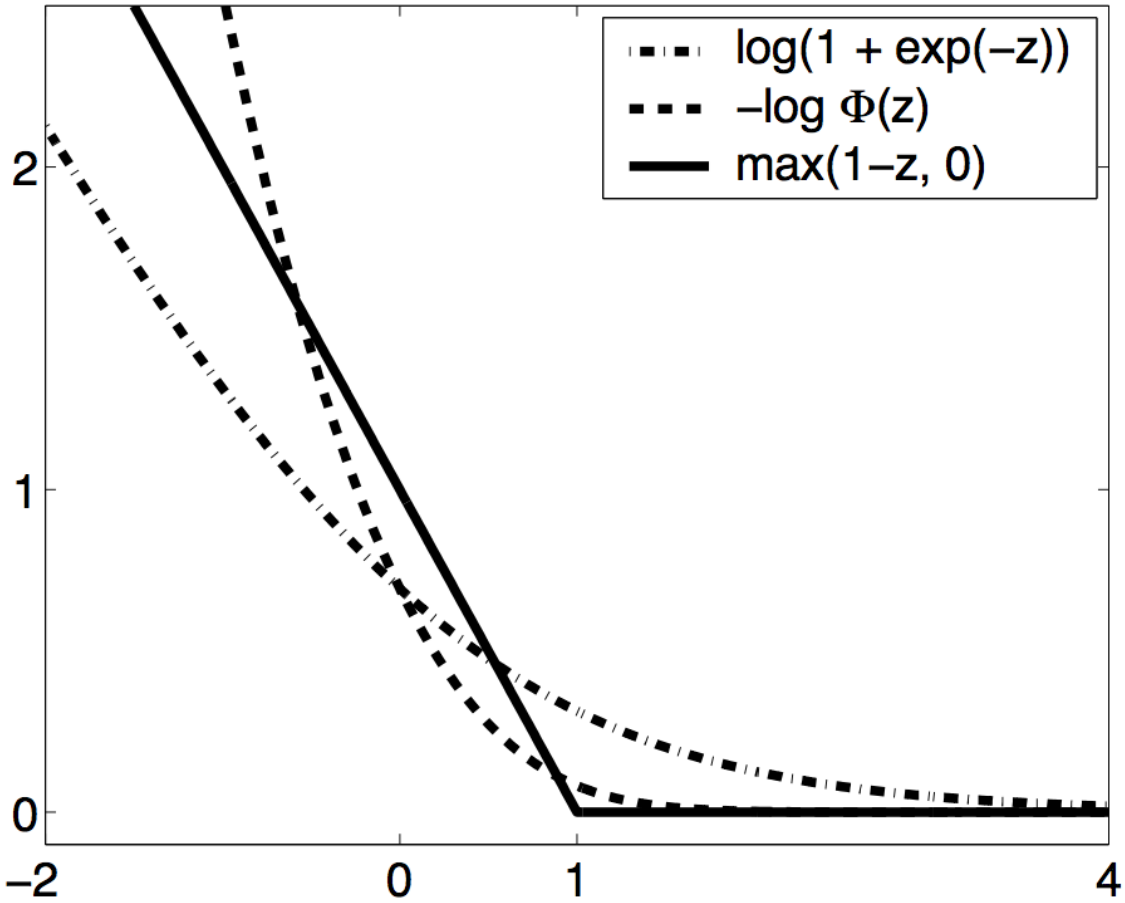
Then, we have $||\boldsymbol{w}||^2 = \boldsymbol{\alpha}^T K \boldsymbol{\alpha} = \boldsymbol{f}^T K^{-1} \boldsymbol{f}$, where we introduced $K\boldsymbol{\alpha} = \boldsymbol{f}$. Thus the soft margin objective function can be re-written as

$$\arg\min_{\boldsymbol{f}} \frac{1}{2} \boldsymbol{f}^T K^{-1} \boldsymbol{f} + C \sum_{i=1}^{n} (1 - y_i f_i)_+ \tag{70}$$

For the binary GP classifier, to obtain the MAP value $\hat{\boldsymbol{f}}$ of $p(\boldsymbol{f}|\boldsymbol{y})$ we solve the following minimization problem:

$$\arg\min_{\boldsymbol{f}} \frac{1}{2} \boldsymbol{f}^T K^{-1} \boldsymbol{f} - \sum_{i=1}^{n} \log p(y_i|f_i). \tag{71}$$

Relationship with Support Vector Machine



A comparison between different error functions.

Reference

1. J. C. Platt, Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In Schölkopf, Burges, and Smola, editors, Advances in Kernel Methods, pages 185–208. MIT Press, 1999.

Outline

Summary

- We introduced Bayesian nonparametric models based on Gaussian processes for nonlinear regression and classification.
- The performance depends on the selection of a kernel function.
- Excellent performance is gained at the cost of high computational complexity, $\mathcal{O}(n^3)$, which prohibits the classic algorithms to be used for big data applications.
- Assumptions on the Gaussian likelihood and factoring property of the likelihood function are unrealistic for many applications.

Selected Recent Advances

- Low-complex Gaussian processes (for big data)
  1. M. Todescato *et.al.*, "Efficient Spatio-Temporal Gaussian Regression via Kalman Filtering", https://arxiv.org/abs/1705.01485, 2017.
  2. J. Hensman and N. Lawrence, "Gaussian Processes for Big Data", http://www.auai.org/uai2013, 2013.
  3. M. Deisenroth, "Distributed Gaussian Processes", https://arxiv.org/pdf/1502.02843, 2015.
  4. S. Ambikasaran *et.al.*, Fast Direct Methods for Gaussian Processes, https://arxiv.org/pdf/1403.6015.pdf, 2015.
- More sophisticated GP model
  1. A. Damianou and N. Lawrence, "Deep Gaussian Process", Proceedings of Machine Learning Research, 2013.
  2. M van der Wilk, "Convolutional Gaussian Processes", https://arxiv.org/abs/1709.01894, 2017.

Some of the above listed papers will be covered in SRIBD reading group.

Outline

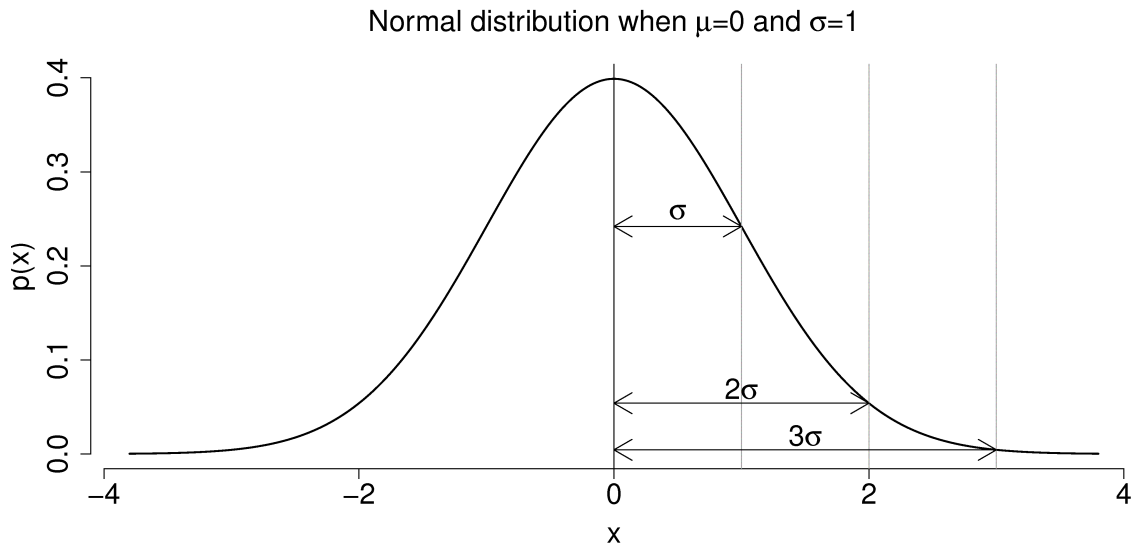
Gaussian Distribution: Univariate Case

The probability density function (pdf) is

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[\frac{-(x-\mu)^2}{2\sigma^2}\right]. \tag{72}$$

The mean and variance are

$$\mathbb{E}(x) = \mu, \quad var(x) = \mathbb{E}\left[(x-\mathbb{E}(x))^2\right] = \sigma^2. \tag{73}$$



Gaussian Distribution: Multivariate Case

The probability density function (pdf) is

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{p/2} \det(\Sigma)^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \quad (74)$$

The mean and covariance matrix are

$$\mathbb{E}(\boldsymbol{x}) = \boldsymbol{\mu}, \quad Cov(\boldsymbol{x}) = \mathbb{E}\left[(\boldsymbol{x}-\mathbb{E}(\boldsymbol{x}))(\boldsymbol{x}-\mathbb{E}(\boldsymbol{x}))^T\right] = \Sigma. \quad (75)$$

Conditional Gaussian Distribution

If the two random variables $\boldsymbol{x} \in \mathbb{R}^{d_x}$ and $\boldsymbol{y} \in \mathbb{R}^{d_y}$ are jointly Gaussian with the following joint distribution:

$$p(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right), \tag{76}$$

then it is easy to derive the following conditional probabilities:

$$p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_{x|y}, \Sigma_{x|y}), \qquad p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_{y|x}, \Sigma_{y|x}), \tag{77}$$

where

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \Sigma_{xy}\Sigma_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y), \qquad \boldsymbol{\mu}_{y|x} = \boldsymbol{\mu}_y + \Sigma_{yx}\Sigma_{xx}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x), \tag{78}$$

and

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}, \qquad \Sigma_{y|x} = \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}. \tag{79}$$

Marginal Gaussian Distribution

Following the previous slide where the joint Gaussian distribution $p(\boldsymbol{x}, \boldsymbol{y})$ was defined.

The marginal distributions out of it are still Gaussian, i.e.,

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_x, \Sigma_{xx}\right), \tag{80}$$

and

$$p(\boldsymbol{y}) = \int p(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{\mu}_y, \Sigma_{yy}\right). \tag{81}$$

An Important Gaussian Identities

The product of two Gaussians gives another (un-normalized) Gaussian

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{a},\boldsymbol{A})\cdot\mathcal{N}(\boldsymbol{x}|\boldsymbol{b},\boldsymbol{B})=\boldsymbol{Z}^{-1}\mathcal{N}(\boldsymbol{x}|\boldsymbol{c},\boldsymbol{C}), \tag{82}$$

where $\boldsymbol{c}=\boldsymbol{C}(\boldsymbol{A}^{-1}\boldsymbol{a}+\boldsymbol{B}^{-1}\boldsymbol{b})$ and $\boldsymbol{C}=(\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1})^{-1}$ and the normalizing factor is

$$\boldsymbol{Z}^{-1}=(2\pi)^{-p/2}\det(\boldsymbol{A}+\boldsymbol{B})^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{a}-\boldsymbol{b})^{T}(\boldsymbol{A}+\boldsymbol{B})^{-1}(\boldsymbol{a}-\boldsymbol{b})\right]. \tag{83}$$

Eigendecomposition of Symmetric, PD Matrices

For real, symmetric, positive definite (PD) covariance matrix Σ, which is of size $n \times n$, its eigenvector equation can be expressed as

$$\Sigma \boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i, \quad i = 1, 2, ..., n$$

where

- the eigenvalues are real;
- and the eigenvectors can be chosen to form an orthonormal set, i.e., $\boldsymbol{u}_i^T \boldsymbol{u}_j = 0$, if $i \neq j$; otherwise $\boldsymbol{u}_i^T \boldsymbol{u}_j = 1$, if $i = j$.

As a consequence, the covariance matrix and its inverse matrix can be re-expressed as

$$\Sigma = \sum_{i=1}^{n} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T, \qquad \Sigma^{-1} = \sum_{i=1}^{n} \frac{1}{\lambda_i} \boldsymbol{u}_i \boldsymbol{u}_i^T. \tag{84}$$

Cholesky Decomposition

A symmetric, positive definite matrix $\boldsymbol{A}$ can be decomposed into a product of a lower triangular matrix $\boldsymbol{L}$ and its transpose (upper triangular)

$$\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{A}, \tag{85}$$

where $\boldsymbol{L}$ is called the Cholesky factor.

To solve the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, we follow the steps below:

- first solve $\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}$ by forward substitution;
- then $\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{y}$ by back substitution;
- write the solution as $\boldsymbol{x} = \boldsymbol{L}^T \setminus (\boldsymbol{L} \setminus \boldsymbol{b})$, where the backslash notation $\boldsymbol{A} \setminus \boldsymbol{b}$ is the vector $\boldsymbol{x}$ which solves $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

An useful result: the determinant of a symmetric, positive definite (PD) matrix can be calculated by

$$\det(\boldsymbol{A}) = \prod_{i=1}^{n} \boldsymbol{L}_{ii}^2, \quad \textit{or} \quad \log\det(\boldsymbol{A}) = 2\sum_{i=1}^{n} \log \boldsymbol{L}_{ii}. \tag{86}$$

Limits

The limit of a rational quadratic is squared exponential due to

$$\lim_{\alpha\to\infty}\left(1+\frac{x^2}{2\alpha}\right)^{-\alpha}=\exp\left(-\frac{x^2}{2}\right), \tag{87}$$

which explains that the limit of the RQ kernel in Eq.(14) for $\alpha \to \infty$ becomes the SE kernel in Eq.(13).

Measure

Let Ω be the set of all possible outcomes of an experiment and $\mathcal{F}$ be a σ-field of subsets of Ω which contains all the events of interest, then μ is a countably additive measure if it is real and non-negative and for all mutually disjoint sets $A_1, A_2, ... \in \mathcal{F}$, we have

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu\left(A_i\right). \tag{88}$$

1. If $\mu\left(\Omega\right) < \infty$, then μ is called a finite measure;
2. if $\mu\left(\Omega\right) = 1$, then μ is called a probability measure;
3. The Lebesgue measure defines a uniform measure over subsets of Euclidean space.

Integration

We give the integration of a function $f(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}$ with respect to a measure μ by

$$\int f(\boldsymbol{x}) d\mu(\boldsymbol{x}), \tag{89}$$

where it is assumed that $f(\cdot)$ is measurable, i.e., for any Borel-measurable set $A \in \mathbb{R}$, $f^{-1}(A) \in \mathcal{B}^d$.

We classify the following two cases:

1. When μ is a Lebesgue measure, $\int f(\boldsymbol{x}) d\mu(\boldsymbol{x}) = \int f(\boldsymbol{x}) d\boldsymbol{x}$;
2. When μ is a probability measure, $\int f(\boldsymbol{x}) d\mu(\boldsymbol{x}) = \int f(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$, where the non-negative function $p(\boldsymbol{x})$ is called the density of the measure if for all $A \in \mathcal{B}^d$ we have $\mu(A) = \int_A p(\boldsymbol{x}) d\boldsymbol{x}$. Note that not all probability measures have densities.

Kernel and PSD Kernel

A general name for a function $k(\cdot, \cdot)$ of two arguments mapping a pair of inputs $\boldsymbol{x} \in \mathcal{X}, \boldsymbol{x}' \in \mathcal{X}$ into $\mathbb{R}$ is a *kernel*. This term arises in the theory of integral operators, where the operator T_k is defined as

$$(T_k f)(\boldsymbol{x}) = \int_{\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}') \mathrm{d}\mu(\boldsymbol{x}'), \tag{90}$$

where μ denotes a measure.

A *kernel* is said to be positive semidefinite if

$$\int k(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}) f(\boldsymbol{x}') \mathrm{d}\mu(\boldsymbol{x}) \mathrm{d}\mu(\boldsymbol{x}') \geq 0, \tag{91}$$

for all $f \in L_2(\mathcal{X}, \mu)$. Equivalently a kernel function which gives rise to PSD Gram matrices for any choice of $n \in \mathbb{N}$ and $\mathcal{D}$ is positive semi-definite.

Expansion of SE Kernel

For $\boldsymbol{x}, \boldsymbol{x}^{'} \in \mathbb{R}^p$, the polynomial kernel $k(\boldsymbol{x}, \boldsymbol{x}^{'}) = \left(\boldsymbol{x}^T \boldsymbol{x}^{'}\right)^d$, where d is the polynomial order, a positive integer. The kernel function can be expressed as:

$$k(\boldsymbol{x}, \boldsymbol{x}^{'}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}^{'}) \tag{92}$$

where

$$\phi(\boldsymbol{x}) = \left\{ \sqrt{\frac{d!}{n_1! n_2! \cdots n_p!}} x_1^{n_1} x_2^{n_2} \cdots x_p^{n_p} \right\}, \quad n_i \geq 0, n_i \in \mathbb{N}, \sum_{i=1}^{p} n_i = d \tag{93}$$

For example, for $p = d = 2$, $\left(\boldsymbol{x}^T \boldsymbol{x}^{'}\right)^2 = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}^{'})$, where according to the above formula,

$$\phi(\boldsymbol{x}) = \left(x_1^2, x_2^2, \sqrt{2} x_1 x_2\right)^T. \tag{94}$$

Expansion of SE Kernel (Con'd)

For the SE kernel,

$$
\begin{aligned}
k_{SE}(\boldsymbol{x}, \boldsymbol{x}^{'}) &= \sigma^2 \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{x}^{'}||^2}{2l^2}\right) \\
&= \sigma^2 \exp\left(\frac{-||\boldsymbol{x}||^2}{2}\right) \exp\left(\frac{-||\boldsymbol{x}^{'}||^2}{2}\right) \exp\left(\boldsymbol{x}^T \boldsymbol{x}^{'}\right) \\
&= \sigma^2 \exp\left(\frac{-||\boldsymbol{x}||^2}{2}\right) \exp\left(\frac{-||\boldsymbol{x}^{'}||^2}{2}\right) \sum_{j=0}^{\infty} \frac{\left(\boldsymbol{x}^T \boldsymbol{x}^{'}\right)^j}{j!}
\end{aligned} \tag{95}
$$

According to the results shown above for the polynomial kernel, we have

$$
\phi(\boldsymbol{x}) = \sigma \exp\left(\frac{-||\boldsymbol{x}||^2}{2}\right) \left\{ \sqrt{\frac{j!}{n_1! n_2! \cdots n_p!}} x_1^{n_1} x_2^{n_2} \cdots x_p^{n_p} \right\} \tag{96}
$$

for all $j = 0, 1, 2, ..., \infty$, $\sum_{i=1}^{p} n_i = j$.

Expansion of SE Kernel (Con'd)

Example: for k=2, when

- $j = 0$, we have term 1
- $j = 1$, we have two terms x_1, x_2
- $j = 2$, we have three terms $x_1^2, x_2^2, \sqrt{2}x_1x_2$
- $j = 3$, we have four terms $x_1^3, x_2^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_2^2x_1$
- $j = 4$, ...
- $\vdots$

Thus, we get

$$\phi(\boldsymbol{x}) = \left\{1, x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2, x_1^3, x_2^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_2^2x_1, \ldots\right\} \times \sigma \exp\left(\frac{-||\boldsymbol{x}||^2}{2}\right) \tag{97}$$

Laplace Approximation

Laplace approximation of $p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y})$ given by

$$q(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{f}|\hat{\boldsymbol{f}}, \boldsymbol{A}^{-1}) \tag{98}$$

where

$$\hat{\boldsymbol{f}} = \arg\max_{\boldsymbol{f}} p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}), \tag{99}$$

$$\boldsymbol{A} = -\nabla_{\boldsymbol{f}} \nabla_{\boldsymbol{f}} \log p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y})|_{\boldsymbol{f}=\hat{\boldsymbol{f}}}. \tag{100}$$

are respectively the maximum of the posterior and the Hessian of the negative log posterior at the maximum.

Laplace Approximation (Con'd)

Due to the Bayes theorem, the maximum of the posterior $\hat{\boldsymbol{f}}$ can be obtained by

$$\begin{aligned}
\hat{\boldsymbol{f}} &= \arg\max_{\boldsymbol{f}} p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) \\
&\equiv \arg\max_{\boldsymbol{f}} \log p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) \\
&\equiv \arg\max_{\boldsymbol{f}} \log p(\boldsymbol{f}) + \log p(\boldsymbol{y}|\boldsymbol{f}) \\
&\equiv \arg\max_{\boldsymbol{f}} -\frac{1}{2}\boldsymbol{f}^T K(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{f} + \sum_{i=1}^{n} \log \sigma(y_i f_i)
\end{aligned} \tag{101}$$

Since the cost function is concave in terms of the latent variables $\boldsymbol{f}$, the MAP estimate $\hat{\boldsymbol{f}}$ is the unique global maximum.

Laplace Approximation (Con'd)

Due to the Bayes theorem, we know that

$$\log p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y}) = \log p(\boldsymbol{f}) + \log p(\boldsymbol{y}|\boldsymbol{f}) + \text{Const} \tag{102}$$

where the constant terms is irrespective of $\boldsymbol{f}$.

Therefore, the Hessian $\boldsymbol{A}$ can be further expressed as $\boldsymbol{A} = \boldsymbol{W} + \boldsymbol{K}^{-1}$, where $\boldsymbol{K}$ is short for $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})$ and $\boldsymbol{W}$ is a diagonal matrix defined below:

$$\begin{aligned}\boldsymbol{W} &\triangleq -\nabla_{\boldsymbol{f}}\nabla_{\boldsymbol{f}} \log p(\boldsymbol{y}|\boldsymbol{f})|_{\boldsymbol{f}=\hat{\boldsymbol{f}}} \\ &= -\left(\nabla_{\boldsymbol{f}}\nabla_{\boldsymbol{f}} \sum_{i=1}^{n} \log \sigma(y_i f_i)\right)|_{\boldsymbol{f}=\hat{\boldsymbol{f}}}.\end{aligned} \tag{103}$$

Matrix Identities

The matrix inversion lemma states that

1. for the matrix inverse,

$$\left(\boldsymbol{Z}+\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^{T}\right)^{-1}=\boldsymbol{Z}^{-1}-\boldsymbol{Z}^{-1}\boldsymbol{U}\left(\boldsymbol{W}^{-1}+\boldsymbol{V}^{T}\boldsymbol{Z}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{V}^{T}\boldsymbol{Z}^{-1}, \tag{104}$$

2. and for the matrix determinant,

$$\det\left(\boldsymbol{Z}+\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^{T}\right)=\det(\boldsymbol{Z})\det(\boldsymbol{W})\det(\boldsymbol{W}^{-1}+\boldsymbol{V}^{T}\boldsymbol{Z}^{-1}\boldsymbol{U}). \tag{105}$$

where the matrices $\boldsymbol{Z}$ is of size $n \times n$, $\boldsymbol{W}$ is of size $m \times m$, $\boldsymbol{U}$ and $\boldsymbol{V}$ are both of size $n \times m$. It is assumed that the relevant matrix inverses all exist.

Reference

1. C. Rasmussen, Gaussian Process for Machine Learning, MIT press, 2006.
2. A. Shashua, "Introduction to Machine Learning", 2008, available on arXiv