Lecture 04: Gaussian Process Regression II

[AIX7026] Advanced Machine Learning

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

Contents

Contents

Bayesian linear regression

Let $\mathcal{S} = \{(x^{(i)}, y^{(i)}\}_{i=1}^{n}$ be a training set of i.i.d. examples from some unknown distribution. The standard probabilistic interpretation of linear regression states that

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}, \quad i = 1, ..., n \tag{1}$$

where $\epsilon^{(i)}$ are i.i.d. white noise variables with independent $\mathcal{N}(0, \sigma^2)$ distributions. It follows that $y^{(i)} - \theta^T x^{(i)} \sim \mathcal{N}(0, \sigma^2)$, or equivalently,

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

For notational convenience, we define

$$X = \begin{bmatrix} — \ (x^{(1)})^T \ — \\ — \ (x^{(2)})^T \ — \\ \vdots \\ — \ (x^{(n)})^T \ — \end{bmatrix} \in \mathcal{R}^{n \times d} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} \in \mathcal{R}^n \quad \vec{\epsilon} = \begin{bmatrix} \epsilon^{(1)} \\ \epsilon^{(2)} \\ \vdots \\ \epsilon^{(n)} \end{bmatrix} \in \mathcal{R}^n \tag{3}$$

Bayesian linear regression

In Bayesian linear regression, we assume that a **prior distribution** over parameters is also given; a typical choice, for instance, is $\theta \sim \mathcal{N}(0, \tau^2 I)$. Using Bayes' rule, we obtain the **parameter posterior**,

$$p(\theta \mid \mathcal{S}) = \frac{p(\theta)p(\mathcal{S} \mid \theta)}{\int_{\theta'} p(\theta')p(\mathcal{S} \mid \theta')d\theta'} = \frac{p(\theta)\prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}, \theta)}{\int_{\theta'} p(\theta')\prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}, \theta')d\theta'}. \tag{4}$$

Assuming the same noise model on testing points as on our training points, the "output" of Bayesian linear regression on a new test point x_* is not just a single guess "y_*", but rather an entire probability distribution over possible outputs, known as the **posterior predictive distribution**:

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

For many types of models, the integrals are difficult to compute, and hence, we often resort to approximations, such as MAP estimation.

Bayesian linear regression

In the case of Bayesian linear regression, however, the integrals actually are *tractable*! In particular, for Bayesian linear regression, one can show that

$$\theta \mid \mathcal{S} \sim \mathcal{N}\left(\frac{1}{\sigma^2} A^{-1} X^T \vec{y}, A^{-1}\right) \tag{6}$$

$$y_* \mid x_*, \mathcal{S} \sim \mathcal{N}\left(\frac{1}{\sigma^2} x_*^T A^{-1} X^T \vec{y}, x_*^T A^{-1} x_* + \sigma^2\right) \tag{7}$$

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

Bayesian linear regression

The derivation of these formulas is somewhat involved. Nonetheless, from these equations, we get at least a flavor of what Bayesian methods are all about: the posterior distribution over the test output y_* for a test input x_* is a Gaussian distribution–this distribution reflects the uncertainty in our predictions $y_* = \theta^T x_* + \epsilon_*$ arising from both the randomness in ϵ_* and the uncertainty in our choice of parameters θ. In contrast, classical probabilistic linear regression models estimate parameters θ directly from the training data but provide no estimate of how reliable these learned parameters may be (see Figure 1).

Bayesian linear regression

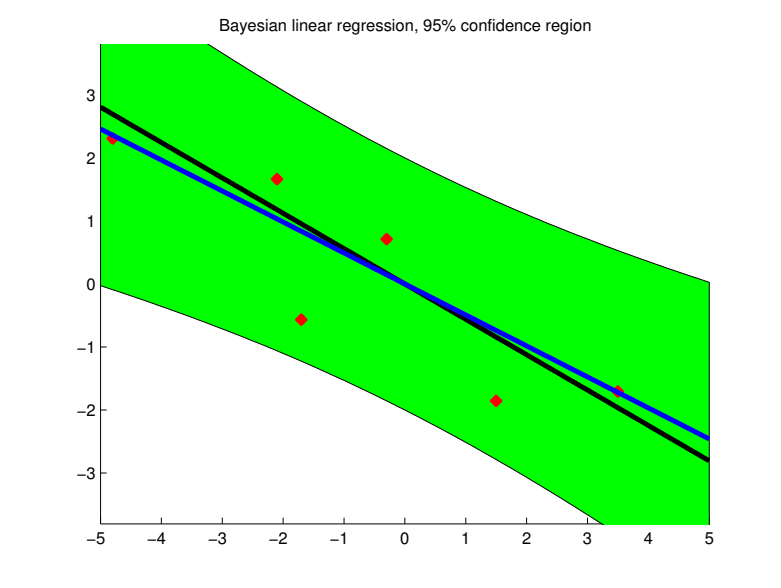


Figure 1: Bayesian linear regression for a one-dimensional linear regression problem, $y^{(i)} = \theta x^{(i)} + \epsilon^{(i)}$, with $\epsilon^{(i)} \sim \mathcal{N}(0, 1)$ i.i.d. noise. The green region denotes the 95% confidence region for predictions of the model. Note that the (vertical) width of the green region is largest at the ends but narrowest in the middle. This region reflects the uncertain in the estimates for the parameter θ. In contrast, a classical linear regression model would display a confidence region of constant width, reflecting only the $\mathcal{N}(0, \sigma^2)$ noise in the outputs.

Learning a distribution over functions: two approaches

Parametric models

- Parameterized model $f_\theta(x)$ with assuming a prior for θ
- Learn a posterior distribution over parameters $p(\theta \mid \mathcal{D})$ to represent a distribution over functions $f_\theta(x)$
- Randomness in θ
- Bayesian linear regression

Nonparametric models

- Stochastic process or random functions $f(x)$
- Learn a distribution over functions directly from data
- Randomness in $f(x)$
- Gaussian process regression

Bayesian regression: parameteric vs. nonparametric

Given a set of training examples, $\mathcal{D} = \{(x_n, y_n) \mid n = 1, ..., N\}$, the goal of Bayesian regression is to make a prediction given new input x_*, computing $p(y_* \mid x_*, \mathcal{D})$.

Parametric approach

- Model $x_n, y_n \mid \theta \sim p(x, y \mid \theta)$, assuming a parametric representation for $f(\cdot) = f_\theta(\cdot)$
- Prior over parameters: $p(\theta)$
- Posterior over parameters:

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})} \tag{8}$$

- Prediction is computed by

$$p(y_* \mid x_*, \mathcal{D}) = \int p(y_* \mid x_*, \theta)p(\theta \mid \mathcal{D})d\theta \tag{9}$$

Bayesian regression: parameteric vs. nonparametric

Given a set of training examples, $\mathcal{D} = \{(x_n, y_n) \mid n = 1, ..., N\}$, the goal of Bayesian regression is to make a prediction given new input x_*, computing $p(y_* \mid x_*, \mathcal{D})$.

Nonparametric approach

- Model $x_n, y_n \sim f$, without parametric representations for $f(\cdot)$
- Prior over function: $f \sim p(f)$
- Posterior over function:

$$p(f \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid f)p(f)}{p(\mathcal{D})} \tag{10}$$

- Prediction is computed by

$$p(y_* \mid x_*, \mathcal{D}) = \int p(y_* \mid x_*, f)p(f \mid \mathcal{D})df \tag{11}$$

Contents

Stochastic process

Stochastic process: finite dimension $\rightarrow$ infinite dimension?

- A single random variable X, specified by its distribution $p(x)$
- Two random variables X_1 and X_2, described by their joint distribution $p(x_1, x_2)$
- N random variables $X_1, ..., X_N$ which correspond to a random vector $\begin{bmatrix} X_1 & \cdots & X_N \end{bmatrix}$, described by their joint distribution $p(x_1, ..., x_N) = p(\mathbf{x})$
- What if $N \rightarrow \infty$?

Gaussian process

Multivariate Gaussian distributions are useful for modeling finite collections of real-valued variables because of their nice analytical properties. Gaussian processes are the extension of multivariate Gaussians to infinite-sized collections of real-valued variables. In particular, this extension will allow us to think of **Gaussian processes** as distributions not just over random vectors but in fact distributions over **random functions**.

Probability distributions over function with finite domains

To understand how one might parameterize probability distribution over functions, consider the following simple example. Let $\mathcal{X} = \{x_1, ..., x_n\}$ be any finite set of elements. Now, consider the set $\mathcal{H}$ of all possible functions mapping from $\mathcal{X}$ to $\mathcal{R}$. For instance, one example of a function $f_0(\cdot) \in \mathcal{H}$ is given by

$$f_0(x_1) = 5, \quad f_0(x_2) = 2.3, \quad f_0(x_3) = -7, \tag{12}$$

$$..., \quad f_0(x_{n-1}) = -\pi, \quad f_0(x_n) = 8. \tag{13}$$

Since the domain of any $f(\cdot) \in \mathcal{H}$ has only n elements, we can always represent $f(\cdot)$ compactly as an n-dimensional vector, $\vec{f} = [\ f(x_1) \quad f(x_2) \quad \cdots \quad f(x_n)\ ]^T$. In order to specify a probability distribution over functions $f(\cdot) \in \mathcal{H}$, we must associate some "probability density" with each function in $\mathcal{H}$. One natural way to do this is to exploit the one-to-one correspondence between functions $f(\cdot) \in \mathcal{H}$ and their vector representations, $\vec{f}$.

Probability distributions over function with finite domains

In particular, if we specify that $\vec{f} \sim \mathcal{N}(\vec{\mu}, \Sigma^2 I)$, then this in turn implies a probability distribution over functions $f(\cdot)$, whose probability density function is given by

$$p(\vec{f}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(f(x_i) - \mu_i)^2\right). \tag{14}$$

In the example above, we showed that probability distributions over functions with finite domains can be represented using a finite-dimensional multivariate Gaussian distribution over function outputs $f(x_1), ..., f(x_n)$ at a finite number of input points $x_1, ..., x_n$. How can we specify probability distributions over functions when the domain size may be infinite? For this, we turn to a fancier type of probability distribution known as a Gaussian process.

Probability distributions over function with infinite domains

A stochastic process is a collection of random variables, $\{f(x) : x \in \mathcal{X}\}$, indexed by elements from some set $\mathcal{X}$, known as the index set. A **Gaussian process** is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution.

In particular, a collection of random variables $\{f(x) : x \in \mathcal{X}\}$ is said to be drawn from Gausian process with **mean function** $m(\cdot)$ and **covariance function** $k(\cdot, \cdot)$ if for any finite set of elements $x_1, ..., x_n \in \mathcal{X}$, the associated finite set of random variables $f(x_1), ..., f(x_n)$ have distribution,

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}\right). \quad (15)$$

Probability distributions over function with infinite domains

We denote this using the notation,

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)). \tag{16}$$

Observe that the mean function and covariance function are aptly named since the above properties imply that

$$m(x) = \mathbb{E}[f(x)] \tag{17}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{18}$$

for any $x, x' \in \mathcal{X}$.

Probability distributions over function with infinite domains

Intuitively, one can think of a function $f(\cdot)$ drawn from a Gaussian process prior as an extremely high-dimensional vector drawn from an extremely high-dimensional multivariate Gaussian. Here, each dimension of the Gaussian corresponds to an element x from the index set $\mathcal{X}$, and the corresponding component of the random vector represents the value of $f(x)$. Using the marginalization property for multivariate Gaussians, we can obtain the marginal multivariate Gaussian density corresponding to any finite subcollection of variables.

Probability distributions over function with infinite domains

What sort of functions $m(\cdot)$ and $k(\cdot, \cdot)$ give rist to valid Gaussian processes? In general, any real-valued function $m(\cdot)$ is acceptable, but for $k(\cdot, \cdot)$, it must be the case that for any set of elements $x_1, ..., x_n \in \mathcal{X}$, the resulting matrix

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{19}$$

is a valid covariance matrix corresponding to some multivariate Gaussian distribution. A standard result in probability theory states that this is true provided that K is positive semidefinite. Sound familiar?

Probability distributions over function with infinite domains

The positive semidefiniteness requirement for covariance matrices computed based on arbitrary input points is, in fact, identical to Mercer's condition for kernels! A function $k(\cdot, \cdot)$ is a valid kernel provided the resulting kernel matrix K defined as above is always positive semidefinite for any set of input points $x_1, ..., x_n \in \mathcal{X}$. Gaussian processes, therefore, are kernel-based probability distributions in the sense that any valid kernel function can be used as a covariance function!

The squared exponential kernel

In order to get an intuition for how Gaussian processes work, consider a simple zero-mean Gaussian process,

$$f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot)) \tag{20}$$

defined for functions $h : \mathcal{X} \longrightarrow \mathcal{R}$ where we take $\mathcal{X} = \mathcal{R}$. Here, we choose the kernel function $k(\cdot, \cdot)$ to be the **squared exponential** kernel function, defined as

$$k_{\mathrm{SE}}(x, x') = \exp\left(-\frac{1}{2\tau^2}\|x - x'\|^2\right) \tag{21}$$

for some $\tau > 0$. What do random functions sampled from this Gaussian process look like?

The squared exponential kernel

In our example, since we use a zero-mean Gaussian process, we would expect that for the function values from our Gaussian process will tend to be distributed around zero. Furthermore, for any pair of elements, $x, x' \in \mathcal{X}$.

- $f(x)$ and $f(x')$ will tend to have high covariance x and x' are "nearby" in the input space, i.e., $\|x - x'\| = |x - x'| \approx 0$, so $\exp\left(-\frac{1}{2\tau^2}\|x - x'\|^2\right) \approx 1$.
- $f(x)$ and $f(x')$ will tend to have low covariance when x and x' are "far apart", i.e., $\|x - x'\| >> 0$, so $\exp\left(-\frac{1}{2\tau^2}\|x - x'\|^2\right) \approx 0$.

More simply stated, functions drawn from a zero-mean Gaussian process prior with the squared exponential kernel will tend to be "locally smooth" with high probability; i.e., nearby function values are highly correlated, and the correlation drops off as a function of distance in the input space (see Figure 2).

The squared exponential kernel

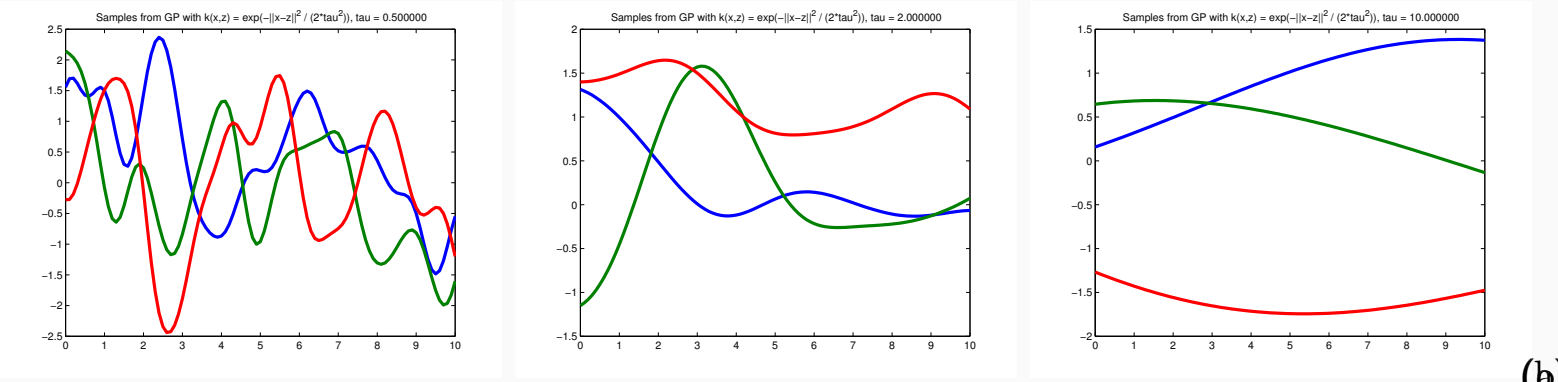
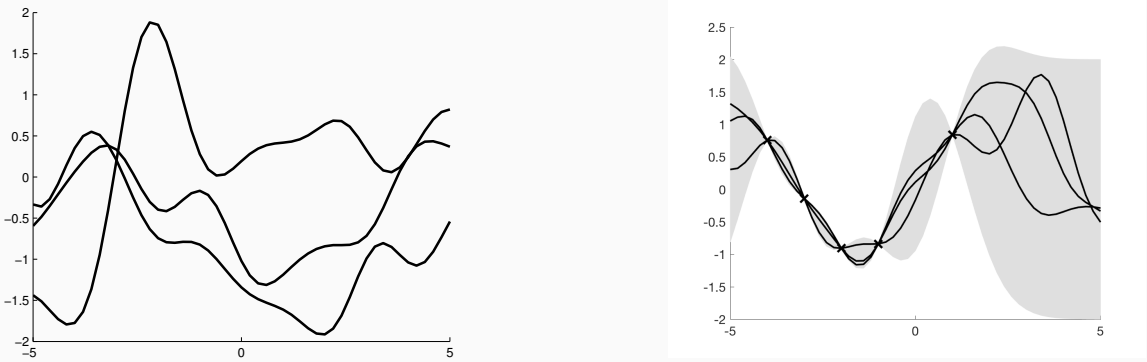


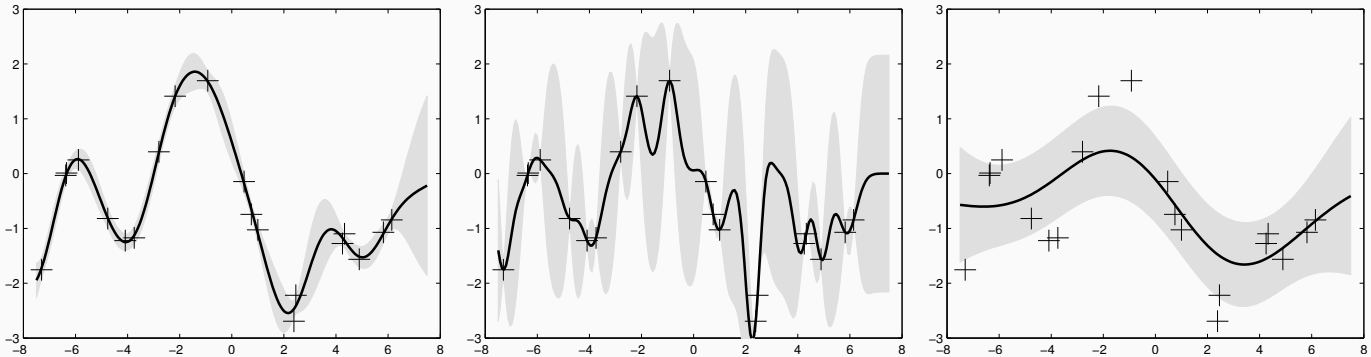
Figure 2: Samples from a zero-mean Gaussian process prior with $k_{SE}(\cdot, \cdot)$ covariance function, using (a) $\tau = 0.5$, (b) $\tau = 2$, and (c) $\tau = 10$. Note that as the bandwidth parameter τ increases, then points which are farther away will have higher correlations than before, and hence the sampled functions tend to be smoother overall.

The squared exponential kernel



(a) (b)

Figure 17.8: Left: some functions sampled from a GP prior with squared exponential kernel. Right: some samples from a GP posterior, after conditioning on 5 noise-free observations. The shaded area represents $\mathbb{E}[f(\mathbf{x})] \pm 2\text{std}[f(\mathbf{x})]$. *Adapted from Figure 2.2 of [RW06]. Generated by code at figures.probml.ai/book1/17.8.*



(a) (b) (c)

Figure 17.9: Some 1d GPs with SE kernels but different hyper-parameters fit to 20 noisy observations. The kernel has the form in Eq. (17.59). The hyper-parameters $(\ell, \sigma_f, \sigma_y)$ *are as follows: (a) (1,1,0.1) (b) (0.3, 1.08, 0.00005), (c) (3.0, 1.16, 0.89). Adapted from Figure 2.5 of [RW06]. Generated by code at figures.probml.ai/book1/17.9.*

The squared exponential kernel

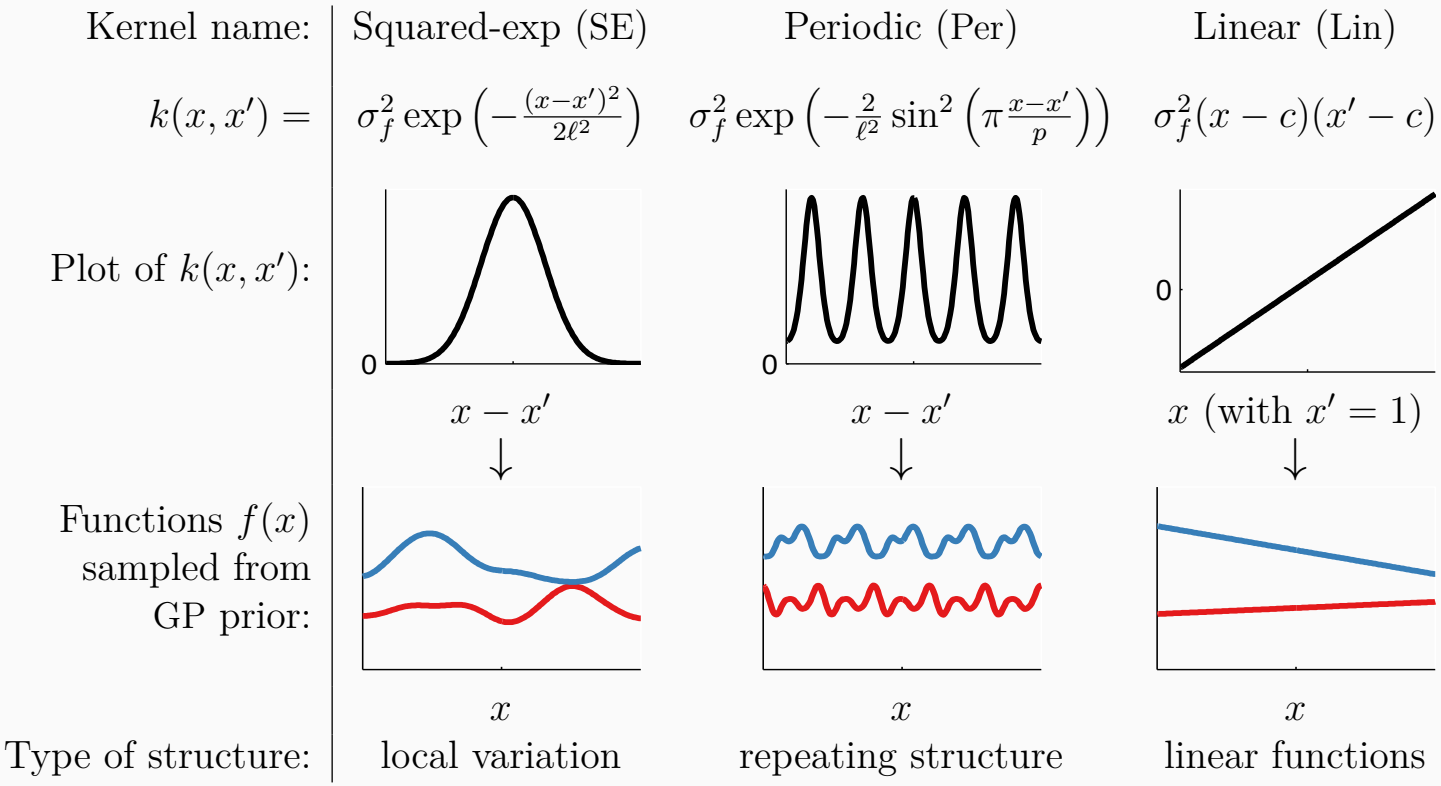


Figure 2.1: Examples of structures expressible by some basic kernels.

Gaussian process regression

Gaussian processes provide a method for modelling probability distributions over functions. Here, we discuss how probability distributions over functions can be used in the framework of Bayesian regression.

Pros

- Theoretically, it can approximate any continuous functions (universality)
- Posterior and predictive distributions are given in closed forms (Gaussians)

Cons

- Heavily depends on the choice of covariance function (kernel)
- $\mathcal{O}(n^3)$ complexity for computing posterior and predictive distributions (can be reduced by various tricks, e.g., inducing point methods)
- Is Gaussian assumption always optimal? –- probably not

Contents

Gaussian process regression model

Let $\mathcal{S} = \{(x^{(i)}, y^{(i)}\}_{i=1}^{n}$ be a training set of i.i.d. examples from some unknown distribution. In the Gaussian process regression model,

$$y^{(i)} = f(x^{(i)}) + \epsilon^{(i)}, \quad i = 1, ..., n \tag{22}$$

where $\epsilon^{(i)}$ are i.i.d. "noise" variables with independent $\mathcal{N}(0, \sigma^2)$ distributions. Like in Bayesian linear regression, we also assume a **prior distribution** over functions $f(\cdot)$; in particular, we assume a zero-mean Gaussian process prior,

$$f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot)) \tag{23}$$

for some valid covariance function $k(\cdot, \cdot)$.

Gaussian process regression model

Now, let $\mathcal{T} = \{(x_*^{(i)}, y_*^{(i)})\}_{i=1}^{n_*}$ be a set of i.i.d. testing points drawn from the same unknown distribution as $\mathcal{S}$. For notational convenience, we define

$$X = \begin{bmatrix} — \left(x^{(1)}\right)^T — \\ — \left(x^{(2)}\right)^T — \\ — \quad \vdots \quad — \\ — \left(x^{(n)}\right)^T — \end{bmatrix} \quad \vec{f} = \begin{bmatrix} f\left(x^{(1)}\right) \\ f\left(x^{(2)}\right) \\ \vdots \\ f\left(x^{(n)}\right) \end{bmatrix} \quad \vec{\epsilon} = \begin{bmatrix} \epsilon^{(1)} \\ \epsilon^{(2)} \\ \vdots \\ \epsilon^{(n)} \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} \tag{24}$$

$$X_* = \begin{bmatrix} — \left(x_*^{(1)}\right)^T — \\ — \left(x_*^{(2)}\right)^T — \\ — \quad \vdots \quad — \\ — \left(x_*^{(n)}\right)^T — \end{bmatrix} \quad \vec{f}_* = \begin{bmatrix} f\left(x_*^{(1)}\right) \\ f\left(x_*^{(2)}\right) \\ \vdots \\ f\left(x_*^{(n)}\right) \end{bmatrix} \quad \vec{\epsilon}_* = \begin{bmatrix} \epsilon_*^{(1)} \\ \epsilon_*^{(2)} \\ \vdots \\ \epsilon_*^{(n)} \end{bmatrix} \quad \vec{y}_* = \begin{bmatrix} y_*^{(1)} \\ y_*^{(2)} \\ \vdots \\ y_*^{(n)} \end{bmatrix} \tag{25}$$

Gaussian process regression model

Given the training data $\mathcal{S}$, the prior $p(f)$ and the testing inputs X_*, how can we compute the posterior predictive distribution over the testing outputs $\vec{y}_*$? For Bayesian linear regression, we used Bayes' rule in order to compute the parameter posterior, which we then used to compute posterior predictive distribution $p(y_* \mid x_*, \mathcal{S})$ for a new test point x_*. For Gaussian process regression, however, it turns out that an even simpler solution exists.

Contents

Prediction

Recall that for any function $f(\cdot)$ drawn from our zero-mean Gaussian process prior with covariance function $k(\cdot, \cdot)$, the marginal distribution over any set of input points belonging to $\mathcal{X}$ must have a joint multivariate Gaussian distribution. In particular, this must hold for the training and test points, so we have

$$\left[\begin{array}{c} \vec{f} \\ \vec{f}_* \end{array} \right] \Bigg| \, X, X_* \sim \mathcal{N}\left(\vec{0}, \left[\begin{array}{cc} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{array} \right] \right) \tag{26}$$

where

$$\begin{aligned}
\vec{f} \in \mathcal{R}^n \quad &\text{such that} \quad \vec{f} = \left[\begin{array}{ccc} f\left(x^{(1)}\right) & \cdots & f\left(x^{(n)}\right) \end{array} \right]^T \\
\vec{f}_* \in \mathcal{R}^n \quad &\text{such that} \quad \vec{f}_* = \left[\begin{array}{ccc} f\left(x_*^{(1)}\right) & \cdots & f\left(x_*^{(n)}\right) \end{array} \right]^T \\
K(X,X) \in \mathcal{R}^{n \times n} \quad &\text{such that} \quad K(X,X)_{ij} = k(x^{(i)}, x^{(j)}) \\
K(X,X_*) \in \mathcal{R}^{n \times n_*} \quad &\text{such that} \quad K(X,X_*)_{ij} = k(x^{(i)}, x_*^{(j)}) \\
K(X_*,X) \in \mathcal{R}^{n_* \times n} \quad &\text{such that} \quad K(X_*,X)_{ij} = k(x_*^{(i)}, x^{(j)}) \\
K(X_*,X_*) \in \mathcal{R}^{n_* \times n_*} \quad &\text{such that} \quad K(X_*,X_*)_{ij} = k(x_*^{(i)}, x_*^{(j)})
\end{aligned}$$

Prediction

From our i.i.d. noise assumption, we have that

$$\begin{bmatrix} \vec{\epsilon} \\ \vec{\epsilon}_* \end{bmatrix} \sim \mathcal{N}\left(\vec{0}, \begin{bmatrix} \sigma^2 I & \vec{0} \\ \vec{0}^T & \sigma^2 I \end{bmatrix}\right). \tag{27}$$

The sums of independent Gaussian random variables is also Gaussian, so

$$\begin{bmatrix} \vec{y} \\ \vec{y}_* \end{bmatrix} \Bigg| X, X_* = \begin{bmatrix} \vec{f} \\ \vec{f}_* \end{bmatrix} + \begin{bmatrix} \vec{\epsilon} \\ \vec{\epsilon}_* \end{bmatrix} \tag{28}$$

$$\sim \mathcal{N}\left(\vec{0}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) + \sigma^2 I \end{bmatrix}\right). \tag{29}$$

Prediction

Now, using the rules for conditioning Gaussians, it follows that

$$\vec{y}_* \mid \vec{y}, X, X_* \sim \mathcal{N}(\mu^*, \Sigma^*) \tag{30}$$

where

$$\mu^* = K(X_*, X)(K(X, X) + \sigma^2 I)^{-1}\vec{y} \tag{31}$$

$$\Sigma^* = K(X_*, X_*) + \sigma^2 I - K(X_*, X)(K(X, X) + \sigma^2 I)^{-1}K(X, X_*) \tag{32}$$

Remarkably, performing prediction in a Gaussian process regression model is very simple, despite the fact that Gaussian processes in themselves are fairly complicated.

Gaussian process regression model

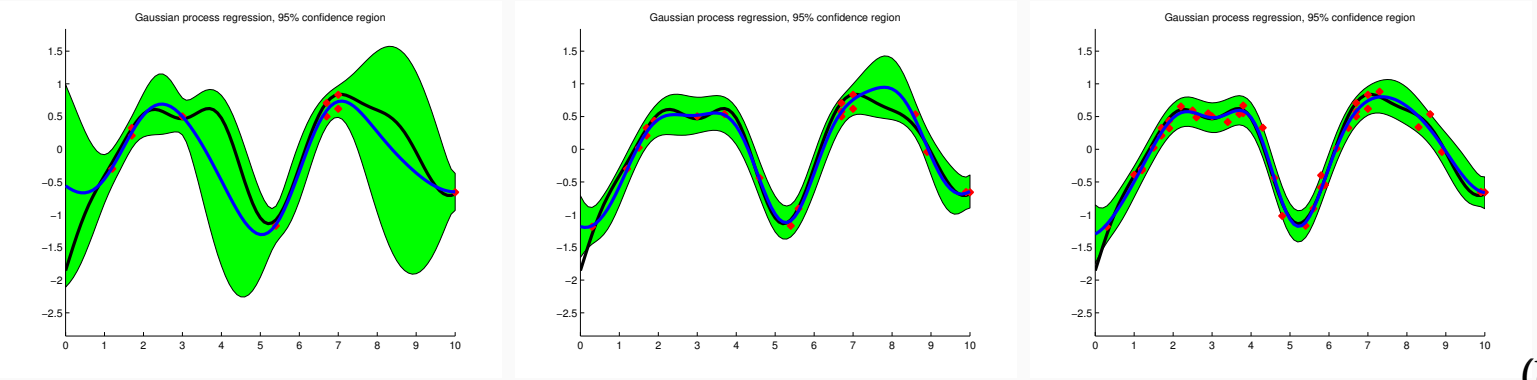


Figure 3: Gaussian process regression using a zero-mean Gaussian process prior with $k_{SE}(\cdot, \cdot)$ covariance function (where $\tau = 0.1$), with noise level $\sigma = 1$, and (a) $m = 10$, (b) $m = 20$, and (c) $m = 40$ training examples. The blue line denotes the mean of the posterior predictive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

Contents

Summary

We close our discussion of our Gaussian processes by pointing out some reasons why Gaussian processes are an attractive model for use in regression problems and in some cases may be preferable to alternative models (such as linear and locally-weighted linear regression):

1. As Bayesian methods, Gaussian process models allow one to quantify uncertainty in predictions resulting not just from intrinsic noise in the problem but also the errors in the parameter estimation procedure. Furthermore, many methods for model selection and hyperparameter selection in Bayesian methods are immediately applicable to Gaussian processes (though we did not address any of these advanced topics here).
2. Like locally-weighted linear regression, Gaussian process regression is non-parametric and hence can model essentially arbitrary functions of the input points.

Summary

We close our discussion of our Gaussian processes by pointing out some reasons why Gaussian processes are an attractive model for use in regression problems and in some cases may be preferable to alternative models (such as linear and locally-weighted linear regression):

3. Gaussian process regression models provide a natural way to introduce kernels into a regression modelling framework. By careful choice of kernels, Gaussian process regression models can sometimes take advantage of structure in the data (though, we also did not examine this issue here).
4. Gaussian process regression models, though perhaps somewhat tricky to understand conceptually, nonetheless lead to simple and straightforward linear algebra implementations.

Contents

Reference and further reading

- “Chap 6.4 | Gaussian Processes” of C. Bishop, Pattern Recognition and Machine Learning
- “Gaussian Processes” of Chuong B. Do (updated by Honglak Lee)
- Gaussian Process Summer School 2019 (Online: http://gpss.cc/gpss19/)
- David MacKay, Gaussian Process Basics (Online: http://videolectures.net/gpip06_mackay_gpb/)
- Carl E. Rasmussen and Christopher K. I. Williams, Gaussian Processes for Machine Learning (Online: http://www.gaussianprocess.org/gpml/)
- Carl E. Rasmussen, Learning with Gaussian Processes (Online: http://videolectures.net/epsrcws08_rasmussen_lgp/)
- “Chap 17 | Kernel methods” of K. Murphy, Probabilistic Machine Learning: An Introduction (Online: https://probml.github.io/pml-book/book1.html)
- “Gaussian Processess” and “Neural Processess” of AI소사이어티, PRML 2021 겨울학교