

Gaussian Processes: Applications in Machine Learning

Abhishek Agarwal (05329022),
KReSIT, IIT Bombay

Guide: Prof. Sunita Sarawagi

March 24, 2006

Abstract

Gaussian Processes (\mathcal{GP}) is an approach to generalize the concept of multivariate Gaussian distribution to a distribution over an infinite set of random variables (continuous/ discrete). They provide a whole new dimension to the field of machine learning by defining distributions over functions used for classification. In this seminar we focused on understanding the role of stochastic process and how it is used in supervised learning to define the prior, and posterior distributions over functions. This report gives an introduction to GPs on a fairly elementary level with special emphasis on characteristics relevant in machine learning, like its application in regression and linear classification problems.

In literature, many approaches have been proposed to deal with supervised learning problems like regression (for continuous outputs) and classification (for discrete outputs). Traditionally, parametric models¹ have been used for this purpose. They restrict the class of functions that we consider for learning, like linear functions of input. These have a possible advantage in ease of interpretability, but for complex data sets, simple parametric models may lack expressive power. The advent of kernel machines, such as Gaussian Processes has opened the possibility of flexible models, where instead of restricting the class we actually define a prior on all possible functions, where higher probabilities are given to the function that we consider more likely. In this short report we present the basic idea on how Gaussian Process models can be used to formulate a Bayesian framework for supervised learning. For broader introductions to Gaussian processes, consult [1] [5] [9].

1 Introduction to Gaussian Processes

In this section we will define the Gaussian Processes and show how they define distribution over functions and generalize the concept of multivariate Gaussian distribution. In following sections we will show how we can come up with a prior for this distribution and how we update it in presence of the training data

Traditionally when we talk about classification of data points(X), we come up with a function which assign a single class value to each $x_i \in X$. In case of GPs, instead of a single value, we come up with a whole distribution of class value for each x_i . So in turn we have this collection of infinitely large number of random variables having joint Gaussian distribution. Thus GPs are formally defined as:

¹parametric models: model which during training learns various parameters from the training data, absorbing all the "information"; data is discarded after training

Definition 1 (Gaussian Processes) A Gaussian Process is a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions.

So given the infinite dimensional input domain, and the fact that \mathcal{GP} associate a random vector with each data point, working with such infinite dimensional objects may seem complicated at first, but it turns out that the quantities that we are interested in computing, require only working with finite dimensional objects.

Gaussian Processes are a natural generalizations over Gaussian distribution with two parameters: mean function $m(x)$ and covariance function $k(x, x')$, where former is vector and later is a matrix, which are defined as

$$m(x) = \mathbf{E}[f(x)] \tag{1}$$

$$k(x, x') = \mathbf{E}[(f(x) - m(x))(f(x') - m(x')))] \tag{2}$$

Difference between the two lies in fact that Gaussian distribution is over vectors while \mathcal{GP} is over functions, meaning that in \mathcal{GP} every function f has a distribution with mean function m and covariance function k . Notationally:

$$f \sim \mathcal{GP}(m, k) \tag{3}$$

Also that in case of Gaussian distribution, individual random variables (*r.v.*) are indexed using their respective position in the vector, for example, i^{th} r.v. is specified as $X(i)$. In \mathcal{GP} each r.v. (distribution of function at given data point) is indexed using the argument x (of the random function $f(x)$), for example, r.v. at i^{th} data point is specified as $f(x_i)$, which is the value of the (stochastic) function f at that location.

Definition of \mathcal{GP} implies a *consistency* requirement, which is also some times known as the marginalization property. This property simply means that if the \mathcal{GP} for example specifies $(y_1, y_2) \sim \mathcal{N}(\mu, \Sigma)$, then $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ where Σ_{11} is the relevant submatrix of Σ . In other words, examination of a larger set of variables does not change the distribution of the smaller set, which in terms of classification means that given the joint Gaussian distribution of the r.v.s associated with some data points, their individual distribution is also Gaussian.

Along with the two parameters (μ, k) , there are some hyper-parameters associated with GPs, which actually determines the value of the two parameters. Discussion on these hyper-parameters is out of scope of this report. So in rest of the report we will assume that given a GP, magically we have figure out the value of two parameters. Please refer [1] to learn about hyper-parameters. We will discuss about covariance function in brief in section 3. Now lets see, given a GP, how we come up with various distributions over functions.

Prior Gaussian Process

Let us look at an example. Consider the Gaussian process given by:

$$f \sim \mathcal{GP}(m, k), \quad \text{where } m(x) = \frac{1}{4}x^2, \quad \text{and } k(x, x') = \exp(-\frac{1}{2}(x - x')^2). \tag{4}$$

Now given these parameters, which describes the prior distribution over the function, we need to draw some samples from the function f , in order to understand the process. To generate such samples we first pick some data points and given these x -values we can evaluate the vector of means and a covariance matrix at these points using Eq. 4, which defines a regular Gaussian distribution:

$$\begin{aligned} \mu_i &= m(x_i) = \frac{1}{4}x_i^2, \quad i = 1, \dots, n \text{ and} \\ \Sigma_{ij} &= k(x_i, x_j) = \exp(-\frac{1}{2}(x_i - x_j)^2), \quad i, j = 1, \dots, n \end{aligned} \tag{5}$$

where m and k represent the parameter of \mathcal{GP} , while μ and Σ of Gaussian distribution. We can now generate a random vector of function values from this distribution given by:

$$f \sim \mathbf{N}(\mu, \Sigma). \quad (6)$$

Figure 1: *Prior Gaussian Process*

We could now plot the values of vector f as function of x , as shown in Figure 1, in which we show a number of sample function drawn at random from the *prior* distribution over functions specified by a GP in Eq (4). This prior represents our prior beliefs over the kinds of functions we expect to observe, before seeing any data. Primarily its the choice of covariance function, which decides the behavior of the

Posterior Gaussian Processes

In the previous section we saw how to define a prior distribution over functions using GPs and its parameters. Using this prior, we can specify some of the properties of the functions, like its smooth and close to be quadratic. In this section we will see how this distribution gets updated in presence of training data and how we can use the posterior distribution for purpose of prediction for unseen test data. When a dataset is provided to GP, it modifies its parameters to give higher probability to function that pass through these points. This situation is illustrated in Fig 1 where blue circles are the data points, the solid red line is the posterior mean of the distribution, and the black lines gives the 95% confidence region.

One of the primary goals of computing the posterior is that it can be used to make predictions for unseen test cases. Let \mathbf{f} be the known function values of the training cases, and let \mathbf{f}_* be a set of function values corresponding to the test set inputs, X_* . Now as per the definition of GP, \mathbf{f} and \mathbf{f}_* will have a joint Gaussian distribution, which can be represented as:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma_* \\ \Sigma_*^T & \Sigma_{**} \end{bmatrix} \right)$$

where we've introduced following notations: $\mu = m(x_i)$, $i = 1, \dots, n$ for the training means and analogously for the test means μ_* ; for the training set covariances we use *Sigma*, Σ_* for training-test set covariances, and Σ_{**} for test set covariances. Now our aim is to find the conditional distribution of \mathbf{f}_* given \mathbf{f} which is expressed as²:

$$\mathbf{f}_* | \mathbf{f} \sim \mathcal{N}(\mu_* + \Sigma_*^T \Sigma^{-1}(\mathbf{f} - \mu), \Sigma_{**} - \Sigma_*^T \Sigma^{-1} \Sigma_*) \quad (7)$$

This is the posterior for a specific set of test cases. Parameters of corresponding posterior process is:

$$\begin{aligned} f|D &\sim \mathcal{GP}(m_D, k_D), \\ m_D(x) &= m(x) + \Sigma(X, x)^T \Sigma^{-1}(\mathbf{f} - \mathbf{m}) \\ k_D(x, x') &= k(x, x') - \Sigma(X, x)^T \Sigma^{-1} \Sigma(X, x') \end{aligned} \quad (8)$$

²Formulae for conditioning a joint Gaussian distribution is:

$$\begin{aligned} \Rightarrow \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} &\sim \left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B} \end{bmatrix} \right) \\ \mathbf{x} | \mathbf{y} &\sim \mathcal{N}(\mathbf{a} + \mathbf{C} \mathbf{B}^{-1}(\mathbf{y} - \mathbf{b}), \mathbf{A} - \mathbf{C} \mathbf{B}^{-1} \mathbf{C}^T) \end{aligned}$$

Now given these parameters, we have a distribution over function which is having variance less than the variance of prior distribution ($k_D(x, x)$ is equal to the prior variance $k(x, x)$ minus a positive term), since the data has given some additional information.

Till now we did all calculation with an underlying assumption of noise-free training outputs. It is common to many applications that there is noise in the observations. The most common assumption about noise is that of additive i.i.d Gaussian noise in the outputs. In GPs, to take noise into account, every $f(x)$ has an extra covariance with itself, with magnitude equal to the noise variance.

$$\begin{aligned} y(x) &= f(x) + \epsilon, & \epsilon &\sim \mathcal{N}(0, \sigma_n^2) \\ f &\sim \mathcal{GP}(m, k), & y &\sim \mathcal{GP}(m, k + \sigma_n^2 \delta_{ii'}) \end{aligned} \quad (9)$$

where $\delta_{ii'} = 1$ iff $i = i'$. Thus the covariance for a noisy process is the sum of the signal covariance and the noise covariance.

Now when we have learn how to sample function from the \mathcal{GP} distribution, and also how to modify the distribution in presence of specific training data, lets see different \mathcal{GP} models in field of machine learning.

2 GP Models

In this section we will be discussing about application of \mathcal{GP} in traditional approaches for machine learning. Major difference between earlier approach and this one is that in former we used to get single class value for each data point, while in the case of \mathcal{GP} we predict a whole distribution of class values at each data point.

Regression

A simple example of a Gaussian process can be obtained from our Bayesian linear regression model $f(x) = \varphi(x)^T \mathbf{w}$ with prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$. Parameters for this \mathcal{GP} are:

$$\begin{aligned} \mathbf{E}[f(x)] &= \varphi(x)^T \mathbf{E}[w] = 0, \\ \mathbf{E}[f(x)f(x')] &= \varphi(x)^T \mathbf{E}[ww^T] \varphi(x') = \varphi(x)^T \Sigma_p \varphi(x') \end{aligned} \quad (10)$$

Thus $f(x)$ and $f(x')$ are jointly Gaussian, in fact, the function values $f(x_1), \dots, f(x_n)$ corresponding to any number of input points n are jointly Gaussian. Assuming this \mathcal{GP} has *squared exponential*³ (SE) covariance function which specifies the covariance between pair of random variables as:

$$\text{cov}(f(x_p), f(x_q)) = k(x_p, x_q) = \exp\left(-\frac{1}{2}|x_p - x_q|^2\right) \quad (11)$$

The specification of the covariance function implies a distribution (prior) over functions as shown in previous section.

Inference in the Bayesian linear model is based on the posterior distribution over the weights, computed by Baye's rule,

$$\mathbf{posterior} = \frac{\text{likelihood} * \text{prior}}{\text{marginal likelihood}}, \quad p(w|y, X) = \frac{\mathbf{p}(y|\mathbf{X}, \mathbf{w}) * \mathbf{p}(\mathbf{w})}{p(y|X)}$$

where the normalizing constant, also known as the marginal likelihood, is independent of weights and is given by the integral of the likelihood times the prior

$$p(y|X) = \int p(y|\mathbf{f}, X) p(\mathbf{f}|X) d\mathbf{f} \quad (12)$$

³Also called as Radial Basis Function(RBF)

As we know under \mathcal{GP} prior is Gaussian, $\mathbf{f}|X \sim \mathcal{N}(\mathbf{0}, K)$, or

$$\log p(\mathbf{f}|X) = -\frac{1}{2}\mathbf{f}^T K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi \quad (13)$$

and the likelihood is factorized Gaussian $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 \mathcal{I})$, so the log marginal likelihood will be

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T (K + \sigma_n^2 \mathcal{I})^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2 \mathcal{I}| - \frac{n}{2}\log 2\pi \quad (14)$$

So now given a \mathcal{GP} , a prior distribution and a set on data points we can come up with above distributions and in turn posterior distribution, which we can use to make prediction for unseen data.

Binary Classification

In previous section we have considered *regression* problems, where the targets are real valued. Another important class of problems is *classification* problems where we wish to assign an input pattern x to one of the C classes, C_1, \dots, C_C .

Both classification and regression can be viewed as *functional approximation* problems. In case of classification using \mathcal{GP} is more complicated than regression, because the assumption of Gaussian likelihood which holds in the case of regression doesnot hold here because targets are discrete class labels. So we do some approxiamte inferencing

Joint probability of class labels and data points canbe expressed using two different approaches:

1. Generative Approach: In this joint distribution is decomposed as:

$$p(x, y) = p(y).(x|y)$$

2. Discriminative Approach: In this joint distribution is decomposed as:

$$p(x, y) = p(x).(y|x)$$

We will be using discriminative approach over here for obvious reason that it models $p(y|x)$ directly. Also for convenience we will be taking example of binary classification. For modelling the binary classification we can use the output of the regression model and turn into a class probabily using *response function* which squashes its argument (with domain $(-\infty, \infty)$) into the range $[0, 1]$. One example is *linear logistic regression* model

$$p(C_1|x) = \lambda(x^T w), \quad \lambda(z) = \frac{1}{1 + \exp(-z)} \quad (15)$$

Linear models for Classification

For \mathcal{GP} classification models, we will be usign linear models as foundation. Following the SVM literature, we use labels $y=+1$ and $y=-1$ to represent two classes. The likelihood is

$$p(y = +1|x, w) = \sigma(x^T w), \quad (16)$$

where w is the weight vector and $\sigma(z)$ can be any sigmoid function, like $\lambda(z)$ used for logistic regression. For symmetric likelihood functions⁴, this can be written more precisely as

$$p(y_i|x_i, w) = \sigma(x_i^T w), \quad (17)$$

⁴symmetric likelihood functions $\Rightarrow \sigma(-z) = 1 - \sigma(z)$, and $P(y_i = -1|x_i, w) = 1 - \sigma(x_i^T w)$

where $f_i \sim f(x_i) = x_i^T w$. Given a dataset $\mathcal{D} = (x_i, y_i) | 1, \dots, n$, and the same Gaussian prior $w \sim \mathcal{N}(0, \Sigma_p)$ as for regression we can obtain the un-normalized log posterior

$$\log p(w|X, y) \sim -\frac{1}{2} w^T \Sigma_p^{-1} w + \sum_{i=1}^n \log \sigma(y_i f^i) \quad (18)$$

Now in the case of regression, posterior was Gaussian while in case of classification, because of the presence of sigmoid function, it does not have a simple analytic form. So we need to do approximate it to some gaussian function as we will see in next section.

Gaussian Process Classification

Basic idea behind binary classification using \mathcal{GP} is to place a GP prior on the latent function $f(x)$ and then "squash" it using logistic function to obtain prior on $\pi(x) \sim p(y = +1|x) = \sigma(f(x))$. This is analogous to linear logistic regression model and equivalent to the development of GP regression from linear regression. In this approach for GP model of classification, inferencing problem is solved by first computing the distribution over the latent function, after seeing the test data

$$p(f_*|X, y, x_*) = \int p(f_*|X, x_*, \mathbf{f}) p(\mathbf{f}|X, y) d\mathbf{f}, \quad (19)$$

where $p(\mathbf{f}|X, y) = p(y|\mathbf{f})p(\mathbf{f}|X)/p(y|X)$ is the posterior over the latent variable, and then use this distribution to produce probabilistic prediction

$$\pi_* = p(y_* = +1|X, y, x_*) = \int \sigma(f_*) p(f_*|X, y, x_*) df_*, \quad (20)$$

Now, as mentioned earlier, in case of regression, prediction was easy because relevant integrals were Gaussian and could be computed analytically, but in case of classification both likelihood as well as posterior are non-Gaussian making the integral analytically intractable. So for this we need to use some analytic approximations of integrals, as discussed in next section.

Laplace Approximation for the Binary GP classifier

We need to find a Gaussian approximate of $p(\mathbf{f}|X, y)$, posterior distribution over latent function. One of the ways is to use Laplace Approximation, in which by doing a second order Taylor expansion of $\log p(\mathbf{f}|X, y)$ around the maximum of the posterior, we obtain a Gaussian approximation

$$q(\mathbf{f}|X, y) = \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, A^{-1}) \propto \exp\left(-\frac{1}{2}(\mathbf{f} - \hat{\mathbf{f}})^T A(\mathbf{f} - \hat{\mathbf{f}})\right) \quad (21)$$

where $\hat{\mathbf{f}} = \operatorname{argmax}_{\mathbf{f}} p(\mathbf{f}|X, y)$ and $A = -\nabla \nabla \log p(\mathbf{f}|X, y)|_{\mathbf{f}=\hat{\mathbf{f}}}$ is the Hessian of the negative log posterior at that point. Now when we differentiate the posterior to find $\hat{\mathbf{f}}$, we observe that $\nabla \log p(\mathbf{f}|X, y)$ is a non-linear function of $\hat{\mathbf{f}}$. So we can use Newton's method to find the values of $\hat{\mathbf{f}}$ and A .

Prediction

Once we have a Gaussian approximation of posterior, we make predictions by computing

$$\bar{\pi}_* = p(\pi|X, y, x_*) = \int \sigma(f_*) p(f_*|X, y, x_*) df_*, \quad (22)$$

which will give us a distribution over the predictions.

3 Covariance Functions

Covariance functions are used to encode our assumption about the function which we wish to learn. They are analogous to the similarity functions used in supervised learning to define nearness or similarity. Till now in this report we have discussed only about squared exponential covariance functions, but in fact any positive definite function⁵ can be used as covariance function. Some of the important characteristic properties of covariance functions are:

1. Stationary: A *stationary* covariance function is a function of $x - x'$, i.e. they are invariant to translations in input space. For example squared exponential (SE) covariance function given in equation 5.
2. Isotropic: An *isotropic* covariance function is a function of $|x - x'|$, i.e. they are invariant to shift of origin. SE is again an example of such function. These are also called as *radial basis functions* (RBFs).
3. Dot-product Covariance: If a covariance function depends only on x and x' through $x \cdot x'$ we call it *dot product* covariance function. A simple example is $k(x, x') = \sigma_0^2 + x \cdot x'$

Covariance functions play an important role in \mathcal{GP} approach, as they not only decide the prior distribution over the functions, but also control the way data can affect the posterior distribution. Given the properties of various covariance functions, we can choose one which best reflects the prior information, alternatively we can get better understanding of the data, by analyzing the properties of covariance function chosen by maximizing the likelihood.

4 Summary

As we saw Gaussian Processes has provided a whole new dimension to the field of machine learning by introducing the concept of distribution over functions and how conveniently it can be used to specify very flexible non-linear regression and classification problems. We also discussed about the computation needed to make predictions for both the cases. GP models are more powerful and flexible than simple linear parametric models and less complex in comparison to other models like multi-layer perceptrons. In fact in many applications, traditional approach of classification performs badly in comparison to \mathcal{GP} approach [6] [7].

Currently the main short-coming in GP is the computational inefficiency for the case of large datasets, but with evolution of more powerful computers and the development of fast sparse inference approximations, we feel GP models will become applicable. A lot of current work is going into the development of approximate methods for non-Gaussian likelihoods, which are required even in the case of simple binary classification, because of which exact computation of predictions is no longer possible analytically. In this report focus was less on giving the algorithmic details of the approximation algorithm and their optimisation variations as they can be found in references. Instead we aimed to convey the concept of non-parametric algorithm and comparing them with the existing parametric algorithms.

⁵Covariance function should be positive definite to ensure that resulting covariance matrix is positive definite, one of the requirements for it to be a *kernel*

References

- [1] Rasmussen and Williams. Gaussian Process for Machine Learning, The MIT Press, 2006.
- [2] Matthias Seeger. Gaussian Process for Machine Learning, 2004. International Journal of Neural Systems, 14(2):69-106, 2004.
- [3] Christopher Williams, Bayesian Classification with Gaussian Processes, In IEEE Trans. Pattern analysis and Machine Intelligence, 1998
- [4] Rasmussen and Williams, Gaussian Process for Regression. In Proceedings of NIPS' 1996.
- [5] Rasmussen, Evaluation of Gaussian Processes and Other Methods for Non-linear Regression. PhD thesis, Dept. of Computer Science, University of Toronto, 1996. Available from <http://www.cs.utoronto.ca/~carl/>
- [6] Anton Schwaighofer, et. al. GPPS: A Gaussian Process Positioning System for Cellular Networks, In proceedings of NIPS' 2003.
- [7] Murillo-Fuentes, et. al. Gaussian Processes for Multiuser Detection in CDMA receivers, Advances in Neural Information Processing System' 2005
- [8] David Mackay, Introduction to Gaussian Processes
- [9] C. Williams. Gaussian processes. In M. A. Arbib, editor, Handbook of Brain Theory and Neural Networks, pages 466-470. The MIT Press, second edition, 2002.