Bruxelles, November 2018 - Salt Lake City, March 2020

Notes on Gaussian Process Regression

(with Python examples)

Kamila Zdybał

kamilazdybal.github.io, kamila.zdybal@gmail.com

Preface

I would like to give credit to many wonderful resources that let me study GPR with joy. In particular, a great lecture by Professor Anna Scaife [3] is among the ones that gave me a lot of intuition during my first encounter. Figures presented in section 4.2 are reproduction of figures from Professor Scaife's lecture. I also gained deeper mathematical understanding of GPR from the great textbook *Gaussian Process for Machine Learning* by C. E. Rasmussen and C. Williams [1]. Many other resources helped me in understanding GPR and writing this short document. I listed all of them in the References.
I also acknowledge the help that I got from the Cross Validated Stack Exchange community, in particular in response to my question [6].

Please feel free to contact me with any suggestions, corrections or comments.

Keywords

Gausssian Process Regression (GPR), regression, covariance function, covariance kernel

Contents

1 Introduction **1**
1.1 Distribution over functions 1

2 Probability **1**
2.1 Prior and posterior 1

3 Covariance matrix **1**

4 Covariance kernels **2**
4.1 Examples 2
4.2 The meaning of hyperparameters 2
4.3 What is Gaussian in a Gaussian Process? 2

1 Introduction

Gaussian Process Regression (GPR) is a regression technique which does not restrict what functions are used to model the data - unlike linear or polynomial regression for instance. Instead, we allow many possible functions from a certain pre-specified *pool*. To each of these function we attribute a certain probability of how likely we believe it will fit well our data and update this function pool as new data points are observed. This brings us to the first concept worth discussing: *distribution over functions*.

1.1 Distribution over functions

In GPR we allow many types of functions and we attribute a certain likelihood to each of them, quantifying our belief that it can well regress our data. The space of these functions is mapped on a space of likelihoods and such mapping is called a *distribution over functions*. Visually, one may treat this as a probability distribution drawn ”above” the space of all possible functions - some functions are more likely, some are less likely.

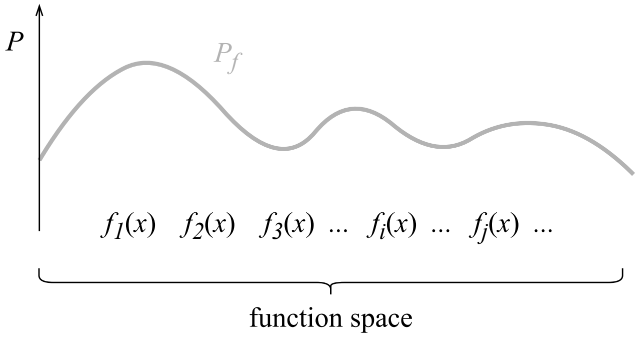

Figure 1: Conceptual visualisation of distribution over functions.

2 Probability

2.1 Prior and posterior

Two words that will be often used in GP are *prior* and *posterior*. The prior and posterior probabilities in GP are often linked to functions and it is important to understand their notion.
The ***prior probability*** is the probability we assign to a specific object (event) before we collect further evidence or make observations that can change it. It can be viewed as an ”initial guess” or ”initial belief” about the given system. The ***posterior probability*** is the updated prior probability that takes into account the new observations or evidence. Conceptually, we may write the following:

$$\text{prior distribution} + \text{data} = \text{posterior distribution} \tag{1}$$

As explained in section 1.1, in GP we encounter probabilities associated with functions. The prior will be the functions that we believe will fit our data well before we actually attempt fitting these functions to new observation points. After the observations have been made, the functions get the probability ”update” to better fit the newly arrived points and these functions that are left will become our posterior.

3 Covariance matrix

Let's think about a dot product between two vectors x_i and x_j:

$$\mathrm{dot}(x_i, x_j) = |x_i||x_j|\cos(\phi) \tag{2}$$

It describes the amount of projection of vector x_i onto x_j (and vice versa) and can be useful when we need to know how much one vector points in the direction of the other.
Now imagine that you have a data set $\mathbf{X}$ with n vectors represented by the columns of this data set, and you would like to know what is the dot product of every possible pair drawn from these vectors. In other words, you would like to know how correlated are all vectors with each other. You can achieve this "global" dot product by multiplying:

$$\mathbf{S} = \mathbf{X}^T\mathbf{X} \tag{3}$$

the result $\mathbf{S}$ is called the *covariance matrix*. Notice that every entry (i, j) in this matrix is a dot product $\mathrm{dot}(x_i, x_j)$ and it either represent the covariance:

$$\mathrm{dot}(x_i, x_j) = \mathrm{cov}(x_i, x_j) \quad \text{for} \quad i \neq j \tag{4}$$

or the variance:

$$\mathrm{dot}(x_i, x_j) = \mathrm{var}(x_i, x_j) \quad \text{for} \quad i = j \tag{5}$$

The covariance matrix is symmetric due to symmetry: $\mathrm{dot}(x_i, x_j) = \mathrm{dot}(x_j, x_i)$.

4 Covariance kernels

The *covariance kernel* is essentialy a function that populates the covariance matrix. This makes our life easier, since first, this matrix might be computationally expensive to evaluate using eq.(3) - notice that for a given size $n \times n$ of the covariance matrix, the data set $\mathbf{X}$ can have arbitrarily many observations (columns of $\mathbf{X}$ can be arbitrarily long). Second, we can easily implement the underlying structure to the covariance.
The only restriction is that the covariance kernel has to be designed such that there is symmetry: $K(x_i, x_j) = K(x_j, x_i)$ (to assure symmetry of the covariance matrix).

4.1 Examples

Squared Exponential kernel:

$$K(x_i, x_j) = h^2 \exp\left(\frac{-(x_i - x_j)^2}{\lambda^2}\right) \tag{6}$$

4.2 The meaning of hyperparameters

We will now take a look at the Squared Exponential kernel which was introduced before and explore how the hyperparameters affect the structure of the covariance matrix and the type of priors drawn from the distribution.
Let's increase the kernel width λ first and draw the covariance matrix (the upper three figures in Fig.2). As λ increases, even the variables which are further away are correlated. That is also reflected in the type of prior that is drawn from this distrbution - three bottom figures in Fig.2. For $\lambda = 5$ the prior looks much less random than for $\lambda = 0.1$ - the variable's value at any point is strongly affected by the values of the nearby variables. To put it in yet other words, the derivatives at any point change much less drastically at $\lambda = 5$ than at $\lambda = 0.1$.

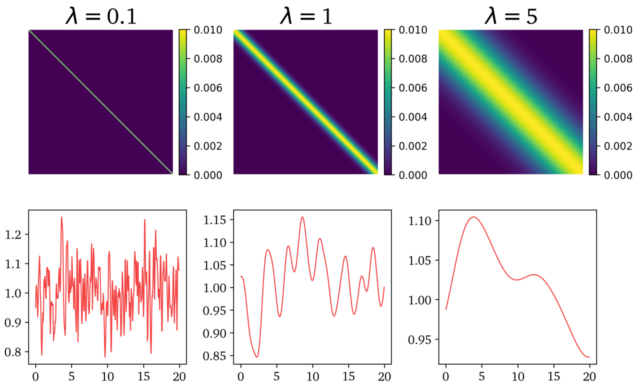

Figure 2: Adjusting the kernel width λ.

Varying the scaling factor h, the priors are now stationary and the covariance has the same width. What is changing is the numerical value of the covariance matrix. For every h the covariance matrix gets scaled and the largest elements in the matrix are always equal to h^2. Thus, varying h will vary the "strength" of the correlation of any variables that are correlated in the first place.

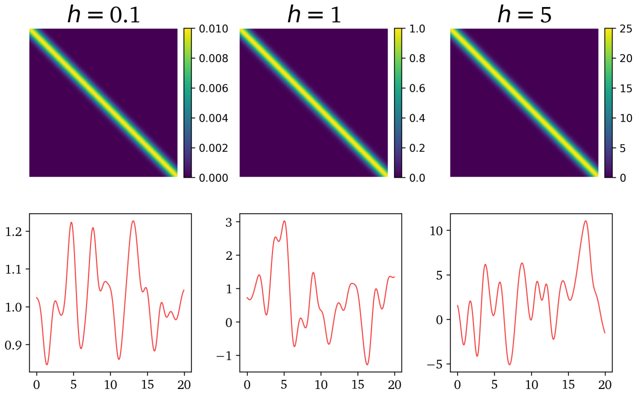

Figure 3: Adjusting the kernel scaling factor h.

4.3 What is Gaussian in a Gaussian Process?

The mean value of the distribution is reflected in the y-axis values in Fig. 2 and 3. It is yet more evident to look at the histograms from more than one prior realisation. In Fig. 4 20 realisations have been drawn and it already becomes visible that the y-value distribution starts to have the bell shape with the prescribed mean.

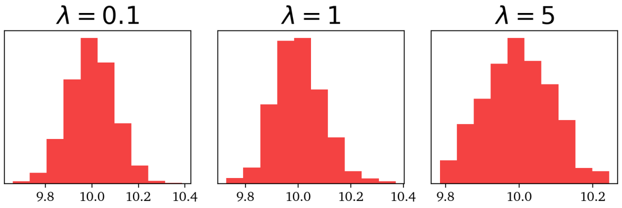

Figure 4: Histogram of 20 realisations with mean = 10.

This brings us to another interesting point which I so far was hiding from view - the one restriction that is placed on the possible regression functions inside the function pool is that **all functions have a Gaussian marginal distribution**. In other words, the histogram of realisations begin to look like a normal distribution as the number of realisations increase.

References

[1] C. E. Rasmussen, C. Williams, *Gaussian Process for Machine Learning*, 2006

[2] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, S. Aigrain *Gaussian Processes for Timeseries Modelling*, 2012

[3] A. Scaife, *Machine Learning: Gaussian Process Modelling in Python*, an online lecture, https://youtu.be/UpsV1y6wMQ8

[4] D. Foreman-Mackey, *Time Series Analysis Using Gaussian Processes in Python & the Search for Earth 2.0*, an online lecture, https://youtu.be/WhoFbu9dBe0

[5] M. R. Malik, B. J. Isaac, A. Coussement, P. J. Smith, A. Parente, *Principal component analysis coupled with nonlinear regression for chemistry reduction*, Combustion and Flame 187 (2018) 30-41

[6] https://stats.stackexchange.com/questions/376141/what-is-a-distribution-over-functions

[7] https://matthewdharris.com/2016/05/16/gaussian-process-hyperparameter-estimation/

[8] D. Duvenaud, *The Kernel Cookbook: Advice on Covariance functions*, https://www.cs.toronto.edu/ duvenaud/cookbook/

[9] D. Duvenaud, *Automatic Model Construction with Gaussian Processes*, PhD thesis, University of Cambridge, 2014

[10] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, M. O'Neil, *Fast Direct Methods for Gaussian Processes*, 2014