Introduction to Gaussian Processes: Regression and Classification

Ruben Martinez-Cantin
Defense University Center
Zaragoza, Spain
rmcantin@unizar.es

Resumen

A Gaussian process is a simple, yet powerful, non-parametric Bayesian model. Originally developed as an stochastic process for time series analysis, it can be also used as a prior distribution over functions, which can be used for inference. In this laboratory we will learn how it can be used for simple examples of nonlinear regression, classification, experimental design and Bayesian optimization.

1. Introduction

During this laboratory we are going to use GPML: The Gaussian process toolbox by Carl Edward Rasmussen and Hannes Nickisch. The toolbox is based on the methods presented in the book also by Carl Rasmussen. Both the toolbox and the book [2] are freely available[1]. Some of the examples that weare going to address here are extracted directly from the book, so you are encouraged to check it during the lab. There is also a short manual in [1]. It is written to work both in MATLAB and GNU OCTAVE.

> **Important:** Before doing anything, run the *gpml/startup.m* scritp. It will make sure all the toolboxes, datasets and other files are in MATLAB path.

2. Part 1: Regression

2.1. Using the GPML code for regression

In order to get some details about how does the toolbox work, we are going to follow the examples that appears in the documentation.

We will open the file *gpml/doc/index.html* and follow the instructions that appear in the **Regression** section. The code that corresponds to that instructions is located in *gpml/doc/demoRegression.m* instructions.

This toolbox allows to use and combine different mean $\mu(x)$ and covariance[2] $k(x, x')$ functions and inference methods.

$$z(x) = GP(\mu(x), K(x, \cdot))$$

The most common approach while working with Gaussian processes is to use a Gaussian likelihood. Thus, the inference can be computed exactly using simple algebraic operations. If that is the case, the optimal solution is always provided by the *infExact* method. However, as we will see during the lab sessions, the Gaussian likelihood is not always the best option. In that case, we need to rely on approximate inference methods.

Once we have selected the functions and inference method, we have to set up the corresponding structure with all the parameters and hyperparameters.

[1] GPML: http://www.gaussianprocess.com
[2] During this labs, we will use the word covariance and kernel interchangeably

First of all, lets take a look at the first code snippet from the demo.

- Can you figure out how are the samples generated from the GP?
- Which terms corresponds to the generative prior?
- Which corresponds to the observation noise?

Continue with the Regression section of the documentation.

Once you have finished, you can compare different covariance or mean functions, different parameters, etc. For example, try to see the behavior of the regression functions for different hyperparameters[3].

For simple comparisons, it is better to use the *usageRegression.m* script that can be found in the same folder. This file allows us to try different methods in parallel over the same input data and view the results together. The index variable is used to select among different likelihoods and inference methods. Modify this file to compare also different covariance functions and hyperparameters.

2.2. Exercise: Step function

As you may have realized, you can control the Gaussian processes smoothness of the regression by adjusting some of the parameters. Which parameter does control the smoothness directly?

In general, the simplest approach, which typically works for many realistic problems, is to take an stationary kernel, that is:

$$k(x, x') = k(|x - x'|)$$

Thus, the smoothness of the function will be preserved in all the function domain. This is a desirable result in many machine learning applications, but sometimes, our function have different frequencies in different intervals. Thus, we need to use non-stationary kernels, or mixtures of them[4].

Check *gpml/covFunctions.m* for a complete description of the different covariance functions that are implemented in the toolbox. There are also some *meta*-covariance functions that can be used to compound simple ones.

Once you are familiar with the properties of the different covariance function, edit the file *gpml/lab/reg_step.m*. Since you can compute the actual value of the test samples, you can obtain the likelihood of your model for this problem. Try to find the GP configuration that gives you the highest likelihood.

Have you seen any other property of the implemented covariance functions? How would you, for example, create an sparse Gaussian process?

2.3. Exercise: Robot kinematics

In this exercise, we are going to study a classical problem in robotics and machine learning: learning the direct kinematics of a robot.

For simplicity we take a planar robot with only two degrees of freedom and we want to predict only the height of the end-effector. This problem can be also solved analytically to test if your solution is valid or not.

The real complexity of this example is that, although there are only two degrees of freedom in the robot, the input data for our Gaussian process is 6D. In fact, only two dimensions are the actual motor commands, two are a linear combination of the motor commands and the last two are just white Gaussian noise (non-informative).

- Can you figure out how can you do feature selection using Gaussian processes?

In previous examples we have been used what we called an *isotropic* kernel, that is, a kernel that have a single set of parameters for every dimension. One simple way to do feature selection is by using a non-isotropic kernel, with Automatic Relevance Determination.

[3] Remember **not** to use the minimize function while setting the hyperparamenters by hand. Directly use the prediction function.

[4] See Bernard Schölkopf's talk about how to combine kernels.

3. Part 2: Classification

During this laboratory, we are going to focus on binary classification. However, Gaussian processes can solve multi-class classification. In you are interested in GP multi-class classification, you are encouraged read chapter 3 of Rasmussen and Williams book [2] and the references therein.

3.1. Introduction

There is no much difference between a regression and a classification problem. Basically, a classification problem has discrete outputs, while a regression problem typically deals with continuous outcomes. As we have seen before, Gaussian processes are intended for continuous outcomes, but we can use a simple trick to deal with that in a Bayesian way. The main idea is to consider the regression function as the *likelihood* of every one of the discrete outcomes (labels). There are some functions that can help us to find those probabilities. The most common functions are the logit and the probit functions.

The logit or logistic function is sigmoid function that takes the form of:

$$f(x) = \frac{1}{1 - e^x}$$

The probit function is the cumulative density function (CDF) of the standard normal distribution:

$$f(x) = \Phi(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

The Bayesian approach for classification using logistic regression or probit regression is to use those functions to represent the likelihood of the classification. That is, the likelihood of observing a label C_1 for an element $\mathbf{x}$ is:

$$p(C_1|\mathbf{x}) = f(\mathbf{x}^T\mathbf{w})$$

- If you have not seen those functions before, you can easily implement and plot those functions in MATLAB/OCTAVE.

3.2. Binary Classification with GPML

Let us go back to the tutorial about GPML and continue with the **Classification** section.

Note how we will use the likelihood functions that we have described (logit and probit). As we have seen in the regression part, only the Gaussian likelihood allows for a closed form of the posterior distribution. Thus, classification problems with GPs always require some approximate inference algorithms. The table of inference methods allows us to check which inference method is valid for which likelihood.

3.3. Handwritten digits recognition

Handwritten character recognition is a well-known problem in machine learning. For this part of the laboratory we are going to use a standard dataset: *the MNIST database* by Yann LeCun and Corinna Cortes. It has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST, collected among Census Bureau employees and high-school students. The digits have been size-normalized and centered in a 28x28 pixels image.

For simplicity, we are going to use the Matlab version made by Sam Roweis[5], that is already sorted for processing. The train or test sets for each digit are stacked in the form of matrices (test0, train3). Each matrix contains one row per digit. In case we want to plot the digits, we can use the *plot_digit* function.

Since we want to focus on binary classification, we are going to work with one class as positive results (e.g.: number $0 \Rightarrow +1$) and the rest as negative (e.g.: numbers $1 - 9 \Rightarrow -1$). You can try different combinations. Also, remember Robert Shapire's talk and how to combine several binary classifiers to get a multi-class classification.

Another problem is the high dimensionality of the feature space, because we use directly the pixels of every image as inputs. However, we know that the actual dimensionality of the problem is much lower. In

[5] http://cs.nyu.edu/ roweis/data.html

this case, we can use a simple PCA projection. You can use the functions *reduceusingpca* and *project2pca*, for that. The first one will compute the principal components based on the training data. The second one is to project the test data to the lower dimensional space.

- Try different subsets of data and different dimension sizes. Do you think you can use all data?
- Compare logit and probit regression.
- Compare different inference methods.

Referencias

[1] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

[2] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Massachusetts, 2006.