Introduction to Gaussian Processes

Filipe P. de Farias
Teleinformatics Engineering Department
Federal University of Ceará
Fortaleza, CE - Brazil
filipepfarias@fisica.ufc.br

Abstract

Roughly speaking a stochastic process is a generalization of a probability distribution (which describes a finite-dimensional random variable) to *functions*. By focussing on processes which are *Gaussian*, it turns out that the computations required for inference and learning become relatively easy. Thus, the supervised learning problems in machine learning which can be thought of as learning a function from examples can be cast directly into the Gaussian process framework.[**rasmussen2006gaussian**]

1 Introduction

The problem of searching for patterns in data is a fundamental one and has a long and successful history. The discovery of regularities in atomic spectra played a key role in the development and verification of quantum physics in the early twentieth century. The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories.

Applications in which the training data comprises examples of the input vectors along with their corresponding target vectors are known as *supervised learning* problems. If the desired output consists of one or more continuous variables, then the task is called *regression*. An example of a regression problem would be the prediction of the yield in a chemical manufacturing process in which the inputs consist of the concentrations of reactants, the temperature, and the pressure.[**bishop2006pattern**]

In general we denote the input as $\mathbf{x}$, and the output (or target) as y. The input is usually represented as a vector $\mathbf{x}$ as there are in general many input variables. We have a dataset $\mathcal{D}$ of n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i) \,|i = 1, \ldots, n\}$. Given this training data we wish to make predictions for new inputs $\mathbf{x}^*$ that we have not seen in the *training set*. Thus it is clear that the problem at hand is inductive; we need to move from the finite training data $\mathcal{D}$ to a function f that makes predictions for all possible input values. To do this we must make assumptions about the characteristics of the underlying function, as otherwise any function which is consistent with the training data would be equally valid.

A wide variety of methods have been proposed to deal with the *supervised learning* problem; here we describe two common approaches. The first is to restrict the class of functions that we consider, for example by only considering linear functions of the input. The second approach is (speaking rather loosely) to give a prior probability to every possible function, where higher probabilities are given to functions that we consider to be *more likely*, for example because they are smoother than other functions.

The first approach has an obvious problem in that we have to decide upon the richness of the class of functions considered; if we are using a model based on a certain class of

functions (e.g. linear functions) and the target function is not well modelled by this class, then the predictions will be poor. One may be tempted to increase the flexibility of the class of functions, but this runs into the danger of *overfitting*, where we can obtain a good fit to the training data, but perform badly when making test predictions.

The second approach appears to have a serious problem, in that surely there are an uncountably infinite set of possible functions, and how are we going to compute with this set in finite time? This is where the Gaussian *process* comes to our rescue. A Gaussian process is a generalization of the Gaussian probability *distribution*. Whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), a stochastic *process* governs the properties of functions. Leaving mathematical sophistication aside, one can loosely think of a function as a very long vector, each entry in the vector specifying the function value $f(x)$ at a particular input x. It turns out, that although this idea is a little naive, it is surprisingly close what we need. Indeed, the question of how we deal computationally with these infinite dimensional objects has the most pleasant resolution imaginable: if you ask only for the properties of the function at a finite number of points, then inference in the Gaussian process will give you the same answer if you ignore the infinitely many other points, as if you would have taken them all into account! And these answers are consistent with answers to any other finite queries you may have. One of the main attractions of the Gaussian process framework is precisely that it unites a sophisticated and consistent view with computational tractability.

2 Regression

We begin by introducing a simple regression problem, which we shall use as a running example throughout this chapter to motivate a number of key concepts. Suppose we observe a real-valued input variable x and we wish to use this observation to predict the value of a real-valued target variable y. For the present purposes, it is instructive to consider an artificial example using synthetically generated data because we then know the precise process that generated the data for comparison against any learned model.

Now suppose that we are given a training set comprising N observations of x, written $\mathbf{x} \equiv [x_1, \ldots, x_N]^{\mathrm{T}}$, together with corresponding observations of the values of y, denoted $\mathbf{y} \equiv [y_1, \ldots, y_N]^{\mathrm{T}}$. Figure ? shows a plot of a training set comprising $N = 10$ data points. The input data set x in Figure ? was generated by choosing values of x_n , for $n = 1, ..., N$, spaced uniformly in range $[0, 1]$, and the target data set y was obtained by first computing the corresponding values of the function $\sin(2\pi x)$ and then adding a small level of random noise having a Gaussian distribution to each such point in order to obtain the corresponding value y_n. By generating data in this way, we are capturing a property of many real data sets, namely that they possess an underlying regularity, which we wish to learn, but that individual observations are corrupted by random noise. This noise might arise from intrinsically stochastic (i.e. random) processes such as radioactive decay but more typically is due to there being sources of variability that are themselves unobserved.

Our goal is to exploit this training set in order to make predictions of the value $\mathbf{y}^*$ of the target variable for some new value $\mathbf{x}^*$ of the input variable. As we shall see later, this involves implicitly trying to discover the underlying function $\sin(2\pi x)$. This is intrinsically a difficult problem as we have to generalize from a finite data set. Furthermore the observed data are corrupted with noise, and so for a given $\mathbf{x}^*$ there is uncertainty as to the appropriate value for $\mathbf{y}^*$. Probability theory, provides a framework for expressing such uncertainty in a precise and quantitative manner Section 1.2.

For the moment, however, we shall proceed rather informally and consider a simple approach based on curve fitting. In particular, we shall fit the data using a polynomial function of the form

$$f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{1}$$

where M is the *order* of the polynomial, and x^j denotes x raised to the power of j. The polynomial coefficients $w_0, \ldots, w_M$ are collectively denoted by the vector $\mathbf{w}$. Note that,

although the polynomial function $f(x, \mathbf{w})$ is a nonlinear function of x, it is a linear function of the coefficients w. Functions, such as the polynomial, which are linear in the unknown parameters have important properties and are called *linear models*. In general, we could write this weighted sum with any other function. In other words, we can put this in terms of $\phi_n(x) = x^n$, where ϕ could be other *basis function*, e.g. we could have different functions f for different basis functions.

$$\begin{aligned} f(x, \mathbf{w}) &= w_0\phi_0(x) + w_1\phi_1(x) + w_2\phi_2(x) + ... + w_{M-1}\phi_{M-1}(x); \\ &= w_0 \exp\left\{-\frac{(x-\mu_0)^2}{2\sigma^2}\right\} + w_1 \exp\left\{-\frac{(x-\mu_1)^2}{2\sigma^2}\right\} + \\ &... + w_{M-1} \exp\left\{-\frac{(x-\mu_{M-1})^2}{2\sigma^2}\right\}; \\ &= w_0 \sin(0 \cdot x) + w_1 \cos(1 \cdot x) + \\ &... + w_{M-2} \sin((M-2) \cdot x) + w_{M-1} \cos((M-1) \cdot x); \\ &= \sum_{j=0}^{M-1} w_j \phi_j(x); \end{aligned}$$

Plot each function For simplicity, we'll make use of an more compact notation with matrices (see Appendix **??**). Then being $\boldsymbol{\phi}_m \equiv [\phi_m(x_0) \dots \phi_m(x_N)]^{\mathrm{T}}$, we define the *design matrix* as $\Phi \equiv \left[\boldsymbol{\phi}_0 \cdots \boldsymbol{\phi}_{M-1}\right]$. Some textbooks use the notation X for the design matrix.

The values of the coefficients will be determined by fitting the polynomial to the training data. This can be done by minimizing an *error function* that measures the misfit between the function $f(x, \mathbf{w})$, for any given value of w, and the training set data points. One simple choice of error function, which is widely used, is given by the sum of the squares of the errors between the predictions $f(x_n, \mathbf{w})$ for each data point x_n and the corresponding target values y_n, so that we minimize

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{f(x_n, \mathbf{w}) - y_n\}^2 \tag{2}$$

where the factor of $1/2$ is included for later convenience[1]. We can solve the curve fitting problem by choosing the value of w for which $E(\mathbf{w})$ is as small as possible. Because the error function is a quadratic function of the coefficients $\mathbf{w}$, its derivatives with respect to the coefficients will be linear in the elements of $\mathbf{w}$, and so the minimization of the error function has a unique solution, denoted by $\mathbf{w}^*$, which can be found in closed form (see Appendix **??**)[2]

$$\mathbf{w}^* = \left(\Phi^{\mathrm{T}}\Phi\right)^{-1} \Phi^{\mathrm{T}}\mathbf{y} \tag{3}$$

The resulting polynomial is given by the function $f(x, \mathbf{w}^*)$. Plot for different M

There remains the problem of choosing the order M of the polynomial, and as we shall see this will turn out to be an example of an important concept called model comparison or model selection. In Figure 1.4, we show four examples of the results of fitting polynomials having orders M = 0, 1, 3, and 9 to the data set shown in Figure 1.2. We notice that the constant (M = 0) and first order (M = 1) polynomials give rather poor fits to the data and consequently rather poor representations of the function $\sin(2\pi x)$. The third order (M = 3) polynomial seems to give the best fit to the function $\sin(2\pi x)$ of the examples shown in Figure 1.4. When we go to a much higher order polynomial (M = 9), we obtain an excellent fit to the training data. In fact, the polynomial passes exactly through each data point and E(w *) = 0. However, the fitted curve oscillates wildly and gives a very poor representation of the function $\sin(2\pi x)$. This latter behaviour is known as over-fitting. As we have noted earlier, the goal is to achieve good generalization by making accurate predictions for new data. We can obtain some quantitative insight into the dependence of the generalization

[1] Minkowski loss function

[2] The term $\left(\Phi^{\mathrm{T}}\Phi\right)^{-1}\Phi^{\mathrm{T}}$ is sad to be the Moore-Penrose pseudo-inverse matrix, an "inverse" to nonsquare matrices. Indeed, if Φ is square and invertible the term becomes Φ^{-1}.

performance on M by considering a separate test set comprising 100 data points generated using exactly the same procedure used to generate the training set points but with new choices for the random noise values included in the target values. For each choice of M , we can then evaluate the residual value of E(w *) given by (1.2) for the training data, and we can also evaluate E(w *) for the test data set. It is sometimes more convenient to use the root-mean-square (RMS) error defined by

$$E_{\mathrm{RMS}} = \sqrt{2E\left(\mathbf{w}^{\star}\right)/N} \tag{4}$$

in which the division by N allows us to compare different sizes of data sets on an equal footing, and the square root ensures that E RMS is measured on the same scale (and in the same units) as the target variable t. Graphs of the training and test set RMS errors are shown, for various values of M , in Figure 1.5. The test set error is a measure of how well we are doing in predicting the values of t for new data observations of x. We note from Figure 1.5 that small values of M give relatively large values of the test set error, and this can be attributed to the fact that the corresponding polynomials are rather inflexible and are incapable of capturing the oscillations in the function $\sin(2\pi x)$. Values of M in the range 3 ¡ M ¡ 8 give small values for the test set error, and these also give reasonable representations of the generating function $\sin(2\pi x)$, as can be seen, for the case of M = 3, from Figure 1.4.

2.1 Retrieval of style files

The style files for NIPS and other conference information are available on the World Wide Web at

`http://www.nips.cc/`

The file `nips2013.pdf` contains these instructions and illustrates the various formatting requirements your NIPS paper must satisfy. LaTeX users can choose between two style files: `nips11submit_09.sty` (to be used with LaTeX version 2.09) and `nips11submit_e.sty` (to be used with LaTeX2e). The file `nips2013.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own. The file `nips2013.rtf` is provided as a shell for MS Word users.

The formatting instructions contained in these style files are summarized in sections **??**, **??**, and **??** below.

3 General formatting instructions

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing of 11 points. Times New Roman is the preferred typeface throughout. Paragraphs are separated by 1/2 line space, with no indentation.

Paper title is 17 point, initial caps/lower case, bold, centered between 2 horizontal rules. Top rule is 4 points thick and bottom rule is 1 point thick. Allow 1/4 inch space above and below title to rules. All pages should start at 1 inch (6 picas) from the top of the page.

For the final version, authors' names are set in boldface, and each name is centered above the corresponding address. The lead author's name is to be listed first (left-most), and the co-authors' names (if different address) are set to follow. If there is only one co-author, list both author and co-author side by side.

Please pay special attention to the instructions in section **??** regarding figures, tables, acknowledgments, and references.

4 Headings: first level

First level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 12. One line space before the first level heading and 1/2 line space after the first level heading.

4.1 Headings: second level

Second level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the second level heading and 1/2 line space after the second level heading.

4.1.1 Headings: third level

Third level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the third level heading and 1/2 line space after the third level heading.

5 Citations, figures, tables, references

These instructions apply to everyone, regardless of the formatter being used.

5.1 Citations within the text

Citations within the text should be numbered consecutively. The corresponding number is to appear enclosed in square brackets, such as [1] or [2]-[5]. The corresponding references are to be listed in the same order at the end of the paper, in the **References** section. (Note: the standard BibTeX style `unsrt` produces this.) As to the format of the references themselves, any style is acceptable as long as it is used consistently.

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]", not "In our previous work [4]". If you cite your other papers that are not widely available (e.g. a journal paper under review), use anonymous author names in the citation, e.g. an author of the form "A. Anonymous".

5.2 Footnotes

Indicate footnotes with a number[3] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).[4]

5.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction; art work should not be hand-drawn. The figure number and caption always appear after the figure. Place one line space before the figure caption, and one line space after the figure. The figure caption is lower case (except for first word and proper nouns); figures are numbered consecutively.

Make sure the figure caption does not get separated from the figure. Leave sufficient space to avoid splitting the figure and figure caption.

You may use color figures. However, it is best for the figure captions and the paper body to make sense if the paper is printed either in black/white or in color.

5.4 Tables

All tables must be centered, neat, clean and legible. Do not use hand-drawn tables. The table number and title always appear before the table. See Table **??**.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

[3]Sample of the first footnote

[4]Sample of the second footnote

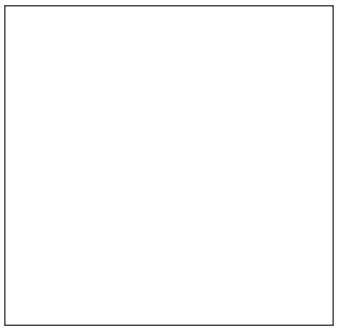

Figure 1: Sample figure caption.

Table 1: Sample table title

PART	DESCRIPTION
Dendrite	Input terminal
Axon	Output terminal
Soma	Cell body (contains cell nucleus)

6 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

7 Preparing PostScript or PDF files

Please prepare PostScript or PDF files with paper size "US Letter", and not, for example, "A4". The -t letter option on dvips will produce US Letter files.

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NIPS. Please see `http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf`
- LaTeX users:
  - Consider directly generating PDF files using `pdflatex` (especially if you are a MiKTeX user). PDF figures must be substituted for EPS figures, however.
  - Otherwise, please generate your PostScript and PDF files with the following commands:

    ```
    dvips mypaper.dvi -t letter -Ppdf -G0 -o mypaper.ps
    ps2pdf mypaper.ps mypaper.pdf
    ```

    Check that the PDF files only contains Type 1 fonts.
  - xfig "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.

  - The `\bbold` package almost always uses bitmap fonts. You can try the equivalent AMS Fonts with command

    ```
    \usepackage[psamsfonts]{amssymb}
    ```

    or use the following workaround for reals, natural and complex:

    ```
    \newcommand{\RR}{I\!\!R} %real numbers
    \newcommand{\Nat}{I\!\!N} %natural numbers
    \newcommand{\CC}{I\!\!\!\!C} %complex numbers
    ```

  - Sometimes the problematic fonts are used in figures included in LaTeX files. The ghostscript program `eps2eps` is the simplest way to clean such figures. For black and white figures, slightly better results can be achieved with program `potrace`.

- MSWord and Windows users (via PDF file):
  - Install the Microsoft Save as PDF Office 2007 Add-in from `http://www.microsoft.com/downloads/details.aspx?displaylang=en&familyid=4d951911-3e7e-4ae6-b059-a2e79ed87041`
  - Select "Save or Publish to PDF" from the Office or File menu
- MSWord and Mac OS X users (via PDF file):
  - From the print menu, click the PDF drop-down box, and select "Save as PDF..."
- MSWord and Windows users (via PS file):
  - To create a new printer on your computer, install the AdobePS printer driver and the Adobe Distiller PPD file from `http://www.adobe.com/support/downloads/detail.jsp?ftpID=204` *Note:* You must reboot your PC after installing the AdobePS driver for it to take effect.
  - To produce the ps file, select "Print" from the MS app, choose the installed AdobePS printer, click on "Properties", click on "Advanced."
  - Set "TrueType Font" to be "Download as Softfont"
  - Open the "PostScript Options" folder
  - Select "PostScript Output Option" to be "Optimize for Portability"
  - Select "TrueType Font Download Option" to be "Outline"
  - Select "Send PostScript Error Handler" to be "No"
  - Click "OK" three times, print your file.
  - Now, use Adobe Acrobat Distiller or ps2pdf to create a PDF file from the PS file. In Acrobat, check the option "Embed all fonts" if applicable.

If your file contains Type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

7.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the graphicx package. Always specify the figure width as a multiple of the line width as in the example below using .eps graphics

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.eps}
```

or

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

for .pdf graphics. See section 4.4 in the graphics bundle documentation (`http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.ps`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

Appendix

A Derivations

A.1 Matrix Form

- Remembering that

$$f(x, \mathbf{w}) = w_0\phi_0(x) + w_1\phi_1(x) + w_2\phi_2(x) + ... + w_{M-1}\phi_{M-1}(x)$$

- We'll evaluate for all x_n values, and then put $f(x_n, \mathbf{w})$ in the matrix form and get

$$f(x_n, \mathbf{w}) = \begin{bmatrix}\phi_0(x_n) & \phi_1(x_n) & ... & \phi_{M-1}(x_n)\end{bmatrix}\begin{bmatrix}w_0 & w_1 & \cdots & w_{M-1}\end{bmatrix}^\top$$

- And then

$$\underbrace{\begin{bmatrix} f(x_0, \mathbf{w}) \\ f(x_1, \mathbf{w}) \\ \vdots \\ f(x_{N-1}, \mathbf{w}) \end{bmatrix}}_{f(\mathbf{x},\mathbf{w})} = \underbrace{\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & ... & \phi_{M-1}(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & ... & \phi_{M-1}(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_{N-1}) & \phi_1(x_{N-1}) & ... & \phi_{M-1}(x_{N-1}) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}}_{\mathbf{w}}$$

- This represents the system $\mathbf{f} = f(\mathbf{x}) = \Phi\mathbf{w}$.
- We ommited the term $\mathbf{w}$ for simplicity, given that x is always dependent of the parameters.

A.2 Optimizing the parameters

- If $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{f\left(x_n, \mathbf{w}\right) - y_n\right\}^2$
- Putting in the matrix form we have $E(\mathbf{w}) = \frac{1}{2}\left(\Phi\mathbf{w} - \mathbf{y}\right)^{\mathrm{T}}\left(\Phi\mathbf{w} - \mathbf{y}\right)$
- Then we'll have

$$\begin{aligned} E(\mathbf{w}) =& \frac{1}{2}\left((\Phi\mathbf{w})^{\mathrm{T}}(\Phi\mathbf{w}) - \mathbf{y}^{\mathrm{T}}(\Phi\mathbf{w}) - (\Phi\mathbf{w})^{\mathrm{T}}\mathbf{y} + \mathbf{y}^{\mathrm{T}}\mathbf{y}\right) \\ =& \frac{1}{2}\left(\mathbf{w}^{\mathrm{T}}\Phi^{\mathrm{T}}\Phi\mathbf{w} - 2\mathbf{y}^{\mathrm{T}}\Phi\mathbf{w} + \mathbf{y}^{\mathrm{T}}\mathbf{y}\right) \end{aligned}$$

  this by the fact that $\alpha = \mathbf{y}^{\mathrm{T}}(\Phi\mathbf{w}) = (\Phi\mathbf{w})^{\mathrm{T}}\mathbf{y}$, being α a scalar.
- Then too, $E(\mathbf{w})$ remains scalar.
- In sequence, we'll try to minimize it in terms of the weights ($\mathbf{w}$) by $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = 0$
- Then

$$\begin{aligned} \frac{1}{2}\left(2\mathbf{w}^{\mathrm{T}}\Phi^{\mathrm{T}}\Phi - 2\mathbf{y}^{\mathrm{T}}\Phi + 0\right) &= 0 \\ \mathbf{y}^{\mathrm{T}}\Phi\left(\Phi^{\mathrm{T}}\Phi\right)^{-1} &= \mathbf{w}^{\mathrm{T}} \\ \left(\Phi^{\mathrm{T}}\Phi\right)^{-1}\Phi^{\mathrm{T}}\mathbf{y} &= \mathbf{w}^{*} \end{aligned}$$

B
Title of Appendix B

Text of Appendix B is Here