Gaussian Process Regression on Curves

Klas Kronander

January 2, 2016

1 Introduction

Gaussian Process Regression has become one of the most popular tools for nonlinear regression problems. Ease of use and intuitive tuning of the open parameters are likely important reasons for its popularity. GPR is however not free from problems. The biggest drawback of GPR when compared with methods such as LWPR or SVR is computational complexity. GPR involves inversion of an $N \times N$ matrix, with N the number of training points provided to the algorithm. This disqualifies GPR from use in realtime applications such as control problems as soon as the number of data points exceed a few thousand.

In this document, we propose a new form for GPR, which exploits a parameterized, continuous representation of the training inputs to drastically reduce the computation time of GPR. This is done by a particular choice of covariance function, which annihilates influence from all but one point on the input curve, and thereby getting rid of the matrix inversion.

2 Background

Let $\{\mathbf{x}_i\}_{i=1}^N$ be a set of N D-dimensional training input vectors $\mathbf{x}_i \in \mathbb{R}^D$. Let a set of N associated observed outputs $\{y_i\}_{i=1}^N$ with $y_i \in \mathbb{R}$ for all $i = 1 \dots N$. In GPR, it is assumed that any collection of outputs have a joint Gaussian distribution, i.e. it is assumed that the y_i, $i = 1 \dots N$ are samples from a Gaussian Process. The covariance between y_i and y_j is modeled as a function $k(\mathbf{x}_i, \mathbf{x}_j)$ and and the covariance matrix of all observed outputs is denoted $\mathbf{K}$ and defined by:

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \dots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \tag{1}$$

Assuming furthermore a zero mean for the outputs, we can write the joint distribution for the observed output data:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma_n^2 \mathbf{I}) \tag{2}$$

where $\mathbf{y} = [y_1, \ldots, y_N]^T$ has been introduced, and where i.i.d Gaussian observation noise with variance σ_n^2 is assumed. Note that so far all that has been done is pure assumptions. We have made an assumption on how the outputs are correlated that is completely independent on their observed values (the covariance function only depends on the inputs).

We can expand the distribution with a new training point $\mathbf{x}^*$ with an unknown output value y^*:

$$p\left(\begin{bmatrix}\mathbf{y}\\ y^*\end{bmatrix}\right) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K} & \mathbf{k}^*\\ \mathbf{k}^{*T} & k(x^*, x^*)\end{bmatrix} + \sigma_n^2\mathbf{I}\right) \tag{3}$$

where $\mathbf{k}^* = [k(x_1, x^*, \ldots, k(x_N, x^*))]^T$. Gaussian Process Regression predicts the value of y^* by conditioning Eq. (3) to end up with an expression that describes the distribution of y^* given $y_1 \ldots y_N$. Standard Gaussian conditioning yields:

$$p(y^*|\mathbf{y}) = \mathcal{N}(\mu^*, c^*) \tag{4}$$

with known forms of μ^* and c^*. For brevity, we only write the form for μ^*:

$$\mu^* = \mathbf{k}^{*T}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \tag{5}$$

As can be seen, whenever the training set is expanded, a new inversion is necessary. This is the main problem of GPR, limiting its use to data sets of small sizes.

3 Continous Input curve

The main difference of classical GPR and C-GPR proposed in this document is that the former is point-based whereas the latter is based on a continuous representation of the input space. Specifically, we will consider curves as input but the concept generalizes to other manifolds as well.

Assume that the inputs lie on a curve C. Furthermore, assume that we can find the coordinates of any point on C by varying the parameter $\mathbf{c}(t) \in \mathbb{R}^D$. The coordinates of all points on the curve can be found by varying t from 0 to 1. Interpolation techniques such as cubic splines or polynomial fitting are examples of such representations.

4 Curve-based covariance function

The key of our regression technique is a particular choice of covariance function, which is based on the geometry of the input curve. Let $\mathbf{x} \in \mathbb{R}^D$ be an arbitrary input point and let $\mathbf{x}' \in C$ be a point on the input curve. We define the covariance function to as:

$$k(\mathbf{x}', \mathbf{x}) = exp(-(\mathbf{x} - \mathbf{x}')^T\mathbf{\Lambda}(\mathbf{x}')(\mathbf{x} - \mathbf{x}')) \tag{6}$$

where $\mathbf{\Lambda}(\mathbf{x}') \in \mathbb{R}^{D\times D}$ is a metric that depends on the curve C and the point $\mathbf{x}'$. Let $\mathbf{e}'_t$ denote the tangent vector of C at $\mathbf{x}'$. Then, $\mathbf{\Lambda}$ is defined as:

$$\mathbf{\Lambda}(\mathbf{x}') = \begin{bmatrix}\mathbf{e}'_t & \dots & \mathbf{e}_D\end{bmatrix} \text{diag}([\lambda_1, \dots, \lambda_D]) \begin{bmatrix}\mathbf{e}'_t & \dots & \mathbf{e}_D\end{bmatrix}^T \tag{7}$$

where the vectors $\mathbf{e}'_t, \dots, \mathbf{e}_D$ constitute an othonormal basis for $\mathbb{R}^D$.

5 Continuous GPR

With the structure in place, note that if λ_1 increases toward infinity, the covariance function $k(\mathbf{x}', \mathbf{x})$ goes to zero for any x satisfying $(\mathbf{x} - \mathbf{x}')^T \mathbf{e}'_t \neq 0$, i.e. any point which does not lie in the orthogonal plane of C at $\tilde{\mathbf{x}}'$. Specifically, under the assumption that C is reasonably flat[1], the matrix $\mathbf{K}$ in Eq. (1) will become diagonal. This means that Eq. (5) reduces to:

$$\boldsymbol{\mu}^* = \mathbf{k}^{*T} \text{diag}\left(\left[\frac{1}{k(\mathbf{x}_1, \mathbf{x}_1) + \sigma_n^2}, \dots, \frac{1}{k(\mathbf{x}_N, \mathbf{x}_N) + \sigma_n^2}\right]\right) \mathbf{y} \tag{8}$$

Let us now examine what happens with the vector $\mathbf{k}^*$ for large λ_1. Let's assume that a query point x^* lies in a plane orthogonal to the tangent of C at some $\mathbf{x}' \in C$. Since C is sufficiently flat, there is only point along C that has x^* in its orthogonal plane. Hence, *there is one and only one non-zero element in* $\mathbf{k}^*$. With this observation, Eq. (9) can be further simplified to:

$$\boldsymbol{\mu}^* = \frac{k(\mathbf{x}', \mathbf{x}^*)}{k(\mathbf{x}', \mathbf{x}') + \sigma_n^2} y' \tag{9}$$

Hence, there is no matrix inversion but an extremely simple expression to compute.

There are, however, a few caveats. First, our approach requires for each query point $\mathbf{x}^*$ to find the associated $\mathbf{x}' \in C$. In practice, this is done by finding the closest point to x^* on C, which is a relatively quick operation since C is a curve and can be scanned efficiently by varying t in $[0, 1]$. Second, we need a continuous representations of the inputs but also of the outputs, since Eq. (9) requires y'.

A more general remark is that a lot of the flexibility of GPR has been lost, since we restrict ourselves to one specific covariance function. The price we pay is that we can not tune the covariance function to suit the need of the application as flexibly as in the standard GPR framework. We gain instead in terms of ducking the trade-off between a sparse training set and accurate predictions which is typical in GPR[2].

[1] This can be further specified.

[2] In general, few data points are preferable. This in turn requires a large lengthscale to ensure non-zero interpolation between the training data. The long lengthscale in turn decreases the accuracy of the predictions near the training points.