MO435 — Probabilistic Machine Learning
INSTITUTO DE COMPUTAÇÃO — UNICAMP

Gaussian Process

Lesson No. 9

Ivan Lima

1 Introduction

A Gaussian process is a generalization of the Gaussian probability distribution. A Gaussian distribution in its general form is a function governed by stochastic properties (mean and covariance). We can understand this function as a infinity long vector, each entry in the vector specifying the function value $f(x)$ at a particular input x. If we 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! [1]

In supervised learning, we observe some inputs x_i and some outputs y_i. First, we assume that $y_i = f(x_i)$, for some unknown function f. Second, we infer a distribution over functions given the data, $p(f|X,y)$. Finally, we use this distribution to make predictions given new inputs, i.e., we compute:

$$p(y_*|\mathbf{x}_*,\mathbf{X},\mathbf{y}) = \int p(y_*|f,\mathbf{x}_*)p(f|\mathbf{X},\mathbf{y})df \tag{1}$$

We can work with the function f in 1 in its parametric representations, thus instead of inferring $p(f|D)$, we infer $p(\theta|D)$. Gaussian Process is a way to perform Bayesian inference over functions themselves,i.e. $p(f|D)$.

Gaussian process ,normally abbreviated to GP, defines a **prior over functions**, which can be converted into a **posterior over functions** once we have seen some data. We only need to be able to define a distribution over the function's values at a finite, but **arbitrary**, set of points, say $x_1,...,x_N$. A GP assumes that $p(f(x_1),...,f(x_N))$ is **jointly Gaussian**, with some mean $\mu(\mathbf{x})$ and covariance $\Sigma(\mathbf{x})$ given by $\Sigma_{ij} = \kappa(x_i,x_j)$, where κ is a positive definite kernel function (see section below for more information on kernels). The key idea is that if $\mathbf{x_i}$ and $\mathbf{x_j}$ are deemed by the kernel to be similar, then we expect the output of the function at those points to be similar. [1] for detailed explanations.

2 GPs for regression

Let the prior on the regression function be a GP, denoted by:

$$f(x) \sim GP(m(x),\kappa(x,\acute{x})) \tag{2}$$

where $m(x)$ is the mean function and $\kappa(x,\acute{x})$ is the kernel or covariance function, defined by:

$$m(x) \quad = \quad \mathbb{E}(f(x)) \tag{3}$$

$$\kappa(x,\acute{x}) = \mathbb{E}[(f(x)-m(x))(f(\acute{x})-m(\acute{x}))^T] \tag{4}$$

We require $\kappa(.)$ to be a positive definite kernel. So, for any finite set of points, this process defines a joint Gaussian:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mu,\mathrm{K}) \tag{5}$$

where $\mathrm{K}_{ij} = \kappa(x_i,x_j)$ and $\mu = (m(x_1),...,m(x_N))$

2.1 Predictions using noise-free observations

Suppose we observe a noise-free training data where $f_i = f(x_i)$ is the noiseless observation of the function evaluated at x_i. Given a test data X_* of size $N_* \times D$, we want to predict the function outputs f_*. GP works as an **interpolator** of the training data. The joint distribution has the following form: (see [2] chapter 4 for detailed explanation on Gaussian properties)

$$\begin{pmatrix} f \\ f_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} \mathrm{K} & \mathrm{K}_* \\ \mathrm{K}_*^T & \mathrm{K}_{**} \end{pmatrix}\right) \tag{6}$$

From the Gaussian conditioning rules ([3] definition 78), the **posterior** has the form:

$$p(f_*|X_*,X,f) = \mathcal{N}(f_*|\mu_* = \mu(X_*) + \mathrm{K}_*^T\mathrm{K}^{-1}(f-\mu(X)), \Sigma_* = \mathrm{K}_{**} - \mathrm{K}_*^T\mathrm{K}^{-1}\mathrm{K}_*) \tag{7}$$

If we use squared exponential kernel, it assumes the form:

$$\kappa(x,\acute{x}) = \sigma_f^2 exp(-\frac{1}{2\ell^2}(x-\acute{x})^2) \tag{8}$$

2.2 Predictions using noisy observations

In this case, the function assumes the form $y = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0,\sigma_y^2)$. The covariance of the observed noise response is:

$$cov\left[y|X\right] = \mathrm{K} + \sigma_y^2 \mathrm{I}_N = \mathrm{K}_y \tag{9}$$

where the matrix is diagonal due to the assumption that the noise terms are independently added to each observation. The joint density of the observed data is then:

$$\begin{pmatrix} y \\ f_* \end{pmatrix} \sim N\left(0, \begin{pmatrix} \mathrm{K}_y & \mathrm{K}_* \\ \mathrm{K}_*^T & \mathrm{K}_{**} \end{pmatrix}\right) \tag{10}$$

and the **posterior predictive density** assumes the form:

$$p(f_*|X_*,X,f) = \mathcal{N}(f_*|\mu_* = \mu(X_*) + \mathrm{K}_*^T\mathrm{K}_y^{-1}y), \Sigma_* = \mathrm{K}_{**} - \mathrm{K}_*^T\mathrm{K}_y^{-1}\mathrm{K}_*) \tag{11}$$

We can write the posterior mean in another way, as follows:

$$\bar{f}_* = \mathrm{K}_*^T\mathrm{K}_y^{-1}y = \sum_{i=1}^{N} \alpha_i \kappa(x_i, x_*) \tag{12}$$

where $\alpha = K_y^{-1}y$.

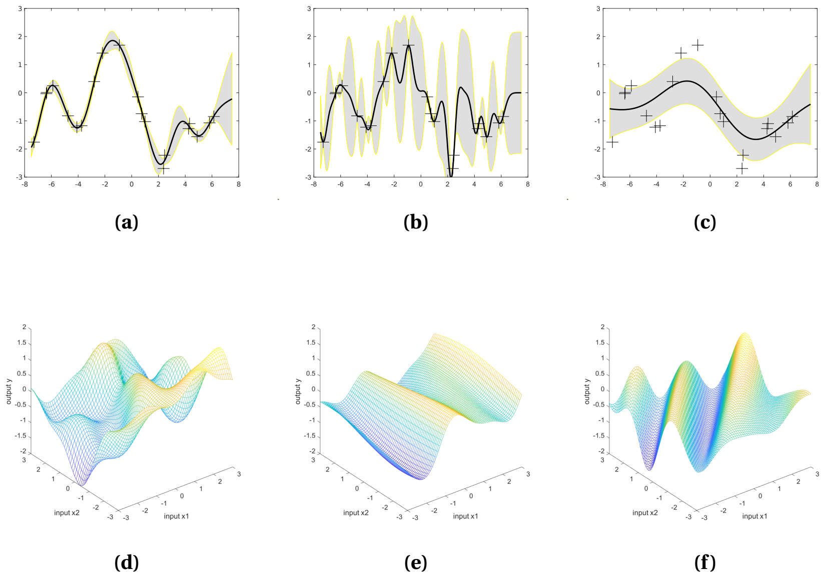
2.3 Effect of the kernel parameters

To illustrate the influence of the kernel parameter on the prediction, suppose we choose the squared-exponential (SE) kernel for the noisy observations, as following:

$$\kappa_y(x_p,x_q) = \sigma_f^2 exp(-\frac{1}{2\ell^2}(x_p - x_q)^2) + \sigma_y^2\delta_{pq} \tag{13}$$

Note that ℓ is the horizontal scale over which the function changes, σ_f^2 controls the vertical scale of the function, and σ_y^2 is the noise variance. Figure 1 generated through [1] gprDemoArd script shows the effects of these parameters.

In figure 1 a, the hyper parameter are chosen $(\ell,\sigma_f,\sigma_y) = (1,1,0.1)$ and the results are a good fit. At experiment b, the hyper parameter are chosen $(\ell,\sigma_f,\sigma_y) = (0.3,1.08,0.00005)$. The ℓ is set to 0.3 and the other two parameters were optimized by maximum (marginal) likelihood, a technique discussed at section 15.2.4 of [2]; in this case the function looks more "wiggly" and the uncertainty goes up faster, since the effective distance from the training points increases more rapidly. At c the hyper parameter are chosen $(\ell,\sigma_f,\sigma_y) = (3,1.16,0.89)$.The ℓ is set to 3.0 and the other two optimized as before. Now the function looks smoother.



(a) (b) (c)

(d) (e) (f)

Figure 1. Kernel's hyper parameters effect.

One can introduce a term M in equation 13.

$$\kappa_y(x_p, x_q) = \sigma_f^2 exp(-\frac{1}{2\ell^2}(x_p - x_q)^T M(x_p - x_q)) + \sigma_y^2 \delta_{pq} \tag{14}$$

In figure 1 d,e and f show 2d functions sampled from a GP with an SE kernel but different hyper-parameters generated by gprDemoArd, written by Carl Ras2006. The kernel has the form in Equation 14, where for each experiment M was set as follows:

d) M = I

e) M = $diag(1,3)^{-2}$

f) M = $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} + diag(6,6)^{-2}$

2.4 Estimating the kernel parameters

Considering an empirical Bayes approach and maximizing the marginal likelihood, which in log has the following form:

$$log(p(y|X)) = log(\mathcal{N}(y|0, \mathrm{K}_y)) = -\frac{1}{2} y\mathrm{K}_y^{-1} y - \frac{1}{2} log|\mathrm{K}_y| - \frac{N}{2} log(2\pi) \tag{15}$$

The first term is a data fit term, the second term is a model complexity term, and the third term is just a constant. Now denoting the kernel hyper parameters by θ, we have:

$$\frac{\partial}{\partial\theta_j} log(p(y|X)) = \frac{1}{2} y^T \mathrm{K}_y^{-1} \frac{\partial \mathrm{K}_y}{\partial\theta_j} \mathrm{K}_y^{-1} y - \frac{1}{2} tr(\mathrm{K}_y^{-1} \frac{\partial \mathrm{K}_y}{\partial\theta_j}) \tag{16}$$

$$\frac{\partial}{\partial\theta_j} log(p(y|X)) = \frac{1}{2} tr((\alpha\alpha^T - \mathrm{K}_y^{-1}) \frac{\partial \mathrm{K}_y}{\partial\theta_j}) \tag{17}$$

where $\alpha = K_y^{-1} y$. One can note from equations 16 and 17 that the partial derivative depends on the form of the kernel. Given an expression for the log **marginal likelihood** and its **derivative**, we can estimate the kernel parameters using any standard **gradient-based optimizer**.

An alternative to computing a point estimate of the hyper-parameters is to compute their posterior via Bayesian inference for the hyper-parameters. More on that in [2], section 15.2.4.2. Also, one can use a quite different approach to optimizing kernel parameters known as **multiple kernel learning**. See [2], section 15.2.4.3 for more details.

2.5 Computational and numerical issues

The predictive mean depends on a matrix inversion. For reasons of numerical stability, it is unwise to directly invert it. A more robust alternative is to compute a Cholesky decomposition.

Summarizing, the six steps sequence for the GP approach is as follows:

1) $\mathbf{L} = \mathbf{cholesky}(\mathbf{K} + \sigma_{\mathbf{y}}^{\mathbf{2}}\mathbf{I})$
2) $\alpha = \mathbf{L}^{\mathbf{T}} \setminus (\mathbf{L} \setminus \mathbf{y})$
3) $\mathbf{E}[\mathbf{f}_*] = \mathbf{K}_*^{\mathbf{T}}\alpha$
4) $\mathbf{V} = \mathbf{L} \setminus \kappa_*$
5) $\mathbf{var}[\mathbf{f}_*] = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^{\mathbf{T}}\mathbf{v}$
6) $\mathbf{logp}(\mathbf{y}|\mathbf{X}) = -\frac{\mathbf{1}}{\mathbf{2}}\mathbf{y}^{\mathbf{T}}\alpha - \Sigma_{\mathbf{i}}\mathbf{logL_{ii}} - \frac{\mathbf{N}}{\mathbf{2}}\mathbf{log}(\mathbf{2}\pi)$

References

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

[2] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press Cambridge, Massachusetts London,England, 2012.

[3] D. Barber, *Bayesian Reasoning and Machine Learning*. Springer, 2010.