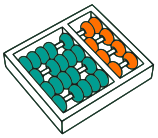

Gaussian Processes [1]

Lesson No. 9

Gustavo de J. Merli - 262948

1 Introduction

In supervised learning, observing some inputs x_i and some outputs y_i. Assume that $y_i = f(x_i)$, for some unknown function f, possibly corrupted by noise. The optimal approach is to infer a distribution over functions given the data, $p(f|X,y)$, and then to use this to make predictions given new inputs, that is, to compute

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

A GP (Gaussian Process) defines a prior over functions, which can be converted into a posterior over functions once it has seen some data.

2 GPs for regression

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

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

where $m(x)$ is the mean function and $\kappa(x,x')$ is the kernel or covariance function.

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

$$\kappa(x,x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))^T] \tag{4}$$

This process defines a joint Gaussian:

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

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

2.1 Predictions using noise-free observations

Suppose a training set $D = \{(x_i, f_i), i = 1:N\}$, where $f_i = f(x_i)$ is the noise-free observation of the function evaluated at x_i. Given a test set X_* of size N_* x D, it is able to predict the function outputs f_*.

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

where $K = \kappa(X,X)$ is N x N, $K_* = \kappa(X,X_*)$ is N x N_* and $K_{**} = \kappa(X_*,X_*)$ is N_* x N_*. The posterior has the following form

$$p(f_*|X_*,X,f) = \mathcal{N}(f_*|\mu_*, \Sigma_*) \tag{7}$$

$$\mu_* = \mu(X_*) + K_*^T K^{-1}(f - \mu(X)) \tag{8}$$

$$\Sigma_* = K_{**} - K_*^T K^{-1} K_* \tag{9}$$

2.2 Predictions using noisy observations

Considering that the observation points are a noisy version of the underlying function, $y = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. In this case, the model is not required to interpolate the data, but it must come "close" to the observed data. The covariance of the observed noisy responses is

$$cov[y|X] = K + \sigma_y^2 I_N = K_y \tag{10}$$

The joint density of the observed data and the latent, noise-free function on the test points is given by

$$\begin{pmatrix} y \\ f_* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K_y & K_* \\ K_*^T & K_{**} \end{pmatrix}\right) \tag{11}$$

where the mean is assumed to be zero, for notational simplicity. The posterior predictive density is

$$p(f_*|X_*, X, y) = \mathcal{N}(f_*|\mu_*, \Sigma_*) \tag{12}$$

$$\mu_* = K_*^T K_y^{-1} y \tag{13}$$

$$\Sigma_* = K_{**} - K_*^T K_y^{-1} K_* \tag{14}$$

2.3 Effect of the kernel parameters

The predictive performance of GPs depends exclusively on the suitability of the chosen kernel. Choosing the following squared-exponential (SE) kernel for the noisy observations

$$\kappa_y(x_p, x_q) = \sigma_f^2 e^{-\frac{1}{2l^2}(x_p - x_q)^2} + \sigma_y^2 \delta_{pq} \tag{15}$$

Here l is the horizonatal scale over which the function changes, σ_f^2 controls the vertical scale of the function, and σ_y^2 is the noise variance.

Extending the SE kernel to multiple dimensions

$$\kappa_y(x_p, x_q) = \sigma_f^2 e^{-\frac{1}{2}(x_p - x_q)^T M (x_p - x_q)} + \sigma_y^2 \delta_{pq} \tag{16}$$

Matrix M can be defined in several ways. The simplest is to use $M_1 = l^{-2} I$. Another possibility is to endow each dimension with its own characteristic length scale, $M_2 = diag(l_i)^{-2}$.

The last way is to create a matrix of the form $M_3 = \Lambda\Lambda^T + diag(l)^{-2}$, where Λ is a D x K matrix, where $K < D$. The columns of Λ correspond to relevant directions in input space.

2.4 Estimating the kernel parameters

Considering and empirical Bayes approach, wich will allow to use continuous optimization methods. Maximizing the marginal likelihood

$$p(y|X) = \int p(y|f, X)\, p(f|X) df \tag{17}$$

Since $p(f|X) = \mathcal{N}(f|0, K)$, and $p(y|f) = \prod_i \mathcal{N}(y_i|f_i, \sigma_y^2)$, the marginal likelihood is given by

$$log\, p(y|X) = log\, \mathcal{N}(y|0, K_y) = -\frac{1}{2} y K_y^{-1} y - \frac{1}{2} log|K_y| - \frac{N}{2} log(2\pi) \tag{18}$$

The first term is a data fit term, the second term is a model complexity term, and the third term is just a constant.

Let the kernel parameters (also called hyper-parameters) be denoted by θ. One can show that

$$log\, p(y|X) = \frac{1}{2} tr\left((\alpha\alpha^T - K_y^{-1}) \frac{\partial K_y}{\partial \theta_j}\right) \tag{19}$$

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

3 GPs meet GLMs

In this section, GPs are extended to the GLM setting, focusing on the classification case. As with Bayesian logistic regression, the main difficulty is that the Gaussian prior is not conjugate to the bernoulli/ multinoulli likelihood. Gaussian approximation will be focused, since it is the simplest and fastest.

3.1 Binary classification

In the binary case, we define the model as $p(y_i|x_i) = \sigma(y_i f(x_i))$, assuming $y_i \in \{-1,+1\}$, and letting $\sigma(z) = sigm(z)$ (logistic regression) or $\sigma(z) = \phi(z)$ (probit regression). As for GP regression, assuming $f \sim GP(0,\kappa)$.

Defining the log of the unnormalized posterior as follows

$$l(f) = logp(y|f) + logp(f|X) = logp(y|f) - \frac{1}{2}f^T K^{-1} f - \frac{1}{2}log|K| - \frac{N}{2}log(2\pi) \tag{20}$$

Let the $J(f) = -l(f)$ be the function to minimize. The gradient and Hessian of this are given by

$$g = -\nabla logp(y|f) + K^{-1} f \tag{21}$$

$$H = -\nabla\nabla logp(y|f) + K^{-1} = W + K^{-1} \tag{22}$$

Using IRLS (Iteratively Reweighted least Squares) to find MAP estimate.

$$f^{new} = f - H^{-1} g = (K^{-1} + W)^{-1}(Wf + \nabla logp(y|f)) \tag{23}$$

At convergencem the Gaussian approximation of the posterior takes the following form

$$p(f|X,y) \approx \mathcal{N}(\widehat{f}, (K^{-1} + W)^{-1}) \tag{24}$$

Now the posterior predictive

$$p(f_*|x_*, X, y) = \mathcal{N}(\mathbb{E}[f_*], var[f_*]) \tag{25}$$

3.2 Multi-class classification

In this section, we consider a model of the form $p(y_i|x_i) = Cat(y_i|S(f_i))$, where $f_i = (f_{i1},...,f_{iC})$, and we assume $f.c \sim GP(0,\kappa_c)$. Thus we have one latent function per class, which are a priori independent, and which may use different kernels. As before, we will use a Gaussian approximation to the posterior.

$$l(f) = -\frac{1}{2}f^T K^{-1} f + y^T f - \sum_{i=1}^{N} log\left(\sum_{c=1}^{C} e^{f_{ic}}\right) - \frac{1}{2}log|K| - \frac{CN}{2}log(2\pi) \tag{26}$$

where

$$f = (f_{11},...,f_{N1},f_{12},...,f_{N2},...,f_{1C},...,f_{NC})^T \tag{27}$$

and y is an encoding of the y_i's which has the same layout as f. Also, K is a block diagonal matrix containing K_c, where $K_c = [\kappa(x_i,x_j)]$ models the correlation of the c'th latent function.

The gradient and Hessian are given by

$$\nabla l = -K^{-1} f + y - \pi \tag{28}$$

$$\nabla\nabla l = -K^{-1} + W \tag{29}$$

where $W = diag(\pi) - \Pi\Pi^T$, where Π is a CN x N matrix obtained stacking $diag(\pi_{:c})$ vertically.

Using IRLS to compute the mode. The Newton step has the form

$$f^{new} = (K^{-1} + W)^{-1}(Wf + y - \pi) \tag{30}$$

To compute the posterior predictive for the visible response, use

$$p(y|x_*, X, y) \approx \int Cat(y|S(f_*))\mathcal{N}(f_*|\mathbb{E}[f_*], cov[f_*])df \tag{31}$$

4 Connection with other methods

There are variety of other methods in statistics and machine learning that are closely related to GP regression/classification.

4.1 Linear models compared to GPs

Consider Bayesian linear regression for D-dimensional features, where the prior on the weights is $p(w) = \mathcal{N}(0, \Sigma)$. The posterior predictive distribution is given by the following

$$p(f_*|x_*, X, y) = \mathcal{N}(\mu, \sigma^2) \tag{32}$$

$$\mu = x_*^T \Sigma X^T (K + \sigma_y^2 I)^{-1} y \tag{33}$$

$$\sigma^2 = x_*^T \Sigma x_* - x_*^T \Sigma X^T (K + \sigma^2 I)^{-1} X \Sigma x_* \tag{34}$$

It is possible to kernelize the above expression by defining $\kappa(x, x') = x^T \Sigma x'$. Thus the Bayesian linear regression is equivalent to a GP with covariance function $\kappa(x, x') = x^T \Sigma x'$. Intuitively this reflects the fact that the model can only represent a limited number of functions. This can result in underfitting, since the model is not flexible enough to capture the data. What is perhaps worse, it can result in overconfidence. So not only the model is wrong, it think it's right.

4.2 Linear smoothers compared to GPs

A linear smoother is a regression function which is a linear function of the training outputs:

$$\widehat{f}(x_*) = \sum w_i(x_*) y_i \tag{35}$$

where $w_i(x_*)$ is called the weight function.

4.3 SVMs compared to GPs

The SVM objective for binary classification is given by

$$J(f) = \frac{1}{2} f^T f + C \sum_{i=1}^{N} (1 - y_i f_i) \tag{36}$$

4.4 LIVM and RVMs compared to GPs

Sparse kernel machines are just linear models with basis function expansion of the form $\phi(x) = [\kappa(x, x_1), ..., \kappa(x, x_N)]$. This is equivalent to a GP with the following kernel:

$$\kappa(x, x') = \sum_{j=1}^{D} \frac{1}{\alpha_j} \phi_j(x) \phi_j(x') \tag{37}$$

where $p(w) = \mathcal{N}(0, diag(\alpha^{-1}))$. This kernel function has two interesting properties. First, it is degenerate, meaning it has at most N non-zero eigenvalues, so the joint distribution $p(f, f_*)$ will be highly constrained. Second, the kernel depends on the training data. This can cause the model to be overconfident when extrapolating beyond the training data.

4.5 Neural networks compared to GPs

Neural Newtorks are a nonlinear generalization of GLMs. In the binary classification case, a neural network is defined by a logistic regression model applied to a logistic regression model:

$$p(y|x,\theta) = Ber(y|sigm(w^T sigm(Vx))) \tag{38}$$

Consider a neural network for regression with one hidden layer. This has the form

$$p(y|x,\theta) = \mathcal{N}(y|f(x;\theta),\sigma^2) \tag{39}$$

where

$$f(x) = b + \sum_{j=1}^{H} v_j g(x;u_j) \tag{40}$$

where b is the offset of bias term, v_j is the output weight from hidden unid j to the response y, u_j are the inputs weights to unit j from the input x, and $g()$ is the hidden unit activation function. Using as activation / transfer function $g(x;u) = erf(u_0 + \sum_{j=1}^{D} u_j x_j)$, where $erf(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$, and we chosse $u \sim \mathcal{N}(0,\Sigma)$, then the covariance kernel has the form

$$\kappa_{NN}(x,x') = \frac{2}{\pi} sin^{-1}\left(\frac{2\tilde{x}^T \Sigma \tilde{x}'}{\sqrt{(1+2\tilde{x}^T\Sigma\tilde{x})(1+2(\tilde{x}')^T\Sigma\tilde{x}')}}\right) \tag{41}$$

where $\tilde{x} = (1, x_1, ..., x_D)$. This is a true "neural network" kernel, unlike the "sigmoid" kernel $\kappa(x,x') = tanh(a + bx^T x')$, which is not positive definite.

4.6 Smoothing splines compared to GPs

Smoothing splines are a widely used non-parametric method for smoothly interpolating data. They are are a special case of GPs. They are usually used when the input is 1 or 2 dimensional.

4.7 RKHS methods compared to GPs

We can generalize the idea of penalizing derivatives of functions, as used in smoothing splines, to fit functions with a more general notion of smoothness.

$$\widehat{f}(x_*) = \sum_i \widehat{a}_i \kappa(x_*, x_i) = k_*^T (K + \sigma_y^2 I)^{-1} y \tag{42}$$

This is identical to the posterior mean of a GP predictive distribution. Indeed, since the mean and mode of a Gaussian are the same, it is possible to see that a linear regression with an RKHS regularizer is equivalent to MAP estimation with a GP. An analogous statement holds for the GP logistic regression case, which also uses a convex likelihood / loss function.

5 GP latent variable model

In this section, a different way to combine kernels with probabilistic PCA will be discussed. The resulting method is known as the GP-LVM, which stands for “Gaussian process latent variable model”.

Using a linear kernel, the PCA is recovered. Using a more general kernel: $K_z = K + \sigma^2 I$, where K is the Gram matrix for Z. The MLE for $\widehat{Z}$ will no longer be available via eigenvalue methods; instead using gradient-based optimization. THe objective is given by.

$$l = -\frac{D}{2} log|K_z| - \frac{1}{2} tr(K_z^{-1} Y Y^T) \tag{43}$$

and the gradient is given by

$$\frac{\partial l}{\partial Z_{ij}} = \frac{\partial l}{\partial K_z} \frac{\partial K_z}{\partial Z_{ij}} \tag{44}$$

where

$$\frac{\partial l}{\partial K_z} = K_z^{-1} Y Y^T K_z^{-1} - D K_z^{-1} \tag{45}$$

the form $\frac{\partial K_z}{\partial Z_{ij}}$ will of course depend on the kernel used.

References

[1] K. P. Murphy, *Machine Learning : A Probabilistic Perspective*, 1st. Cambridge, Mass. [u.a.]: MIT Press, 2013.