Gaussian Processes

January 19, 2015

1 Introduction

In supervised learning, we try to learn an unknown function f for some inputs $\mathbf{x}_i$ and outputs y_i, such that $f(\mathbf{x}_i) = y_i$, possibly corrupted by noise. Then, we work with the distribution of function over given data: $p(f|\mathbf{X}, \mathbf{y})$. Usually, we assume a specific form of f, so we learn a parameter set θ and work with the distribution $p(\theta|\mathbf{X}, \mathbf{y})$. Nonparametric regression techniques, such as LOWESS (locally weighted scatterplot smoother), overcome this problem by using the given data as our knowledge representation directly. Gaussian Process regression is a Bayesian equivalent of such technique.

How is GP different from LOWESS? We make an important assumption: any vector $[f(\mathbf{x}_i)]$ for any $\mathbf{x}_i$ is jointly Gaussian. Also, the covariance between two function values $f(\mathbf{x}_1)$ and $f(\mathbf{x}_2)$ relies on the user-provided kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2)$.

2 GP Regression

2.1 Noise-free Observations

We usually assume the mean function $\mathrm{E}[f(x)] = 0$ because GPs are usually flexible enough that we don't need an intercept. (However, μ is not omitted from below discussion because there are some modeling techniques which employ a parametric representation of μ)

Say we have the training examples $\mathbf{X}$ and $\mathbf{y}$. Since observations are noiseless, we assume $y_i = f(\mathbf{x}_i)$ and write $\mathbf{y} = \mathbf{f}$. Now we can take a test example $\mathbf{X}_*$ and predict the distribution of the corresponding output $\mathbf{f}_*$. How do we do it? By our assumption of f, the vector of $\mathbf{f}$s are jointly Gaussian, and follows the distribution:

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

where $\mathbf{K}$, $\mathbf{K}_*$ and $\mathbf{K}_*^T$ are covariance matrices calculated from user-specified κ. Then, by standard rules for conditioning Gaussians (of course I don't know how to do this, but see section 4.3), we can get a distribution for $\mathbf{f}_*$ conditioned on the rest of the observed variables:

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f}) = \mathcal{N}(\mathbf{f}_*|\mu_*, \Sigma_*)$$

where $\mu_* = \mu(\mathbf{X}_*) + \mathbf{K}_*^T\mathbf{K}^{-1}(\mathbf{f} - \mu(\mathbf{X}))$ and $\Sigma_* = \mathbf{K}_{**} - \mathbf{K}^T\mathbf{K}^{-1}\mathbf{K}_*$. We now got a posterior distribution of output values for $\mathbf{X}_*$.

2.2 Noisy Observations

We now have:

$$\text{cov}\left[y_p, y_q\right] = \kappa\left(\mathbf{x}_p, \mathbf{x}_q\right) + \sigma_y^2 \delta_{pq}$$

where δ_{pq} is the Kronecker delta.

2.3 Kernel Parameters

Different kernel parameters will result in different functions. Say we use a squared-exponential kernel:

$$\kappa\left(x_p, x_q\right) = \sigma_f^2 \exp\left[-\frac{1}{2\ell^2}\left(x_p - x_q\right)^2\right] + \sigma_y^2 \delta_{pq}$$

- Increasing ℓ will decrease covariance between any two points by discounting distances; the function will be more wiggly as a result.
- Increasing σ_f will increase the vertical scale of the function.
- Increasing σ_y will increase the uncertainty around near points.

This basically means we encode our belief on the function's behavior in the kernel.

2.4 Estimating Kernel Parameters

Kernel parameters are chosen by maximizing the marginal likelihood $p(\mathbf{y}|\mathbf{X})$.