Gaussian Processes

Advanced Topics in High-Performance Computing

Faisal Qureshi

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Gaussian Processes

These slides are based on Ch. 15 of the Kevin P. Murphy book titled, "Machine Learning: a Probabilistic Approach."

Gaussian Processes

Consider the regression tasks. Given data $(\mathbf{x}_i, y_i)$, $i = 1, 2, \cdots, N$. We assume $y_i = f(\mathbf{x}_i)$ for some unknown function f, possibly corrupted by noise. Our goal is to discover f.

The optimal approach is to infer a *distribution over functions* given data, i.e., we need to find $p(f|\mathbf{X}, \mathbf{y})$.

Once we have $p(f|\mathbf{X}, \mathbf{y})$, we can use this to make *predictions* of the following form: "what is the y_* given a $\mathbf{x}_*$?" as follows

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

Aside

To date, we have modelled $p(\theta|\mathcal{D})$, we now endeavour to model $p(f, \mathcal{D})$.

Guassian Process

- A GP defines a *prior* over functions.
- This prior can be converted to a *posterior* over functions once we have been presented with some data.
- Define a distribution over the function's values at a finite, but arbitrary, set of points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$ to represent distribution over functions.
- A GP assumes that $p(f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_N))$ is jointly Gaussian with mean $\mu_(\mathbf{x})$ and covariance $\mathbf{\Sigma}(\mathbf{x})$ given by $\Sigma_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$. $\mathcal{K}$ is a positive definite kernel function.
- 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 these points to be similar too.

GP for regression

- Consider a prior on the regression function of GP

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), \mathcal{K}(\mathbf{x}, \mathbf{x}')),$$

where $m(\mathbf{x})$ is the mean function and $\mathcal{K}(\mathbf{x}, \mathbf{x}')$ is the covariance function. Specifically

$$m(\mathbf{x}) = \mathtt{E}[f(\mathbf{x})]$$

and

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \mathtt{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}')]$$

- This process defines a joint Gaussian

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

Predictions using noise-free observations

- Given training data $\mathcal{D} = \{(\mathbf{x}_i, f_i), i = 1 : N\}$, where $f_i = f(\mathbf{x}_i)$ is the noise-free observation of the function evaluated at $\mathbf{x}_i$.
- Given a test set $\mathbf{X}_*$ of size $N_* \times D$, we want to predict the function outputs $\mathbf{f}_*$.
- By definition GP joint distribution has the following form

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

  where $\mathbf{K} = \mathcal{K}(\mathbf{X}, \mathbf{X})$ is $N \times N$, $\mathbf{K}_* = \mathcal{K}(\mathbf{X}, \mathbf{X}_*)$ is $N \times N_*$ and $\mathbf{K}_{**} = \mathcal{K}(\mathbf{X}_*, \mathbf{X}_*)$ is $N_* \times N_*$.
- We can get the posterior using the conditioning rules for Gaussians

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f}) &= \mathcal{N}(\mathbf{f}_*|\mu_*, \mathbf{\Sigma}_*) \\ \mu_* &= \mu(\mathbf{X}_*) + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{f} - \mu(\mathbf{X})) \\ \Sigma_* &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \end{aligned}$$

Predictions using noisy observations

- Consider the case where we observe a noisy version of the underlying functions, i.e., $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$.
- The covariance of the observed noisy responses is

$$\mathrm{cov}[y_p, y_q] = \mathcal{K}(\mathbf{x}_p, \mathbf{x}_q) + \sigma_y^2 \delta_{pq},$$

where $\delta_{pq} = \mathbb{I}(p = q)$. So we can write

$$\mathrm{cov}[y|\mathbf{K}] = \mathbf{K} + \sigma_y^2 \mathbf{I}_N \triangleq \mathbf{K}_y.$$

This assumes that the noise terms were independently added to each observation.

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

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_y & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right),$$

where we assume a zero mean.

Predictions using noisy observations

- The posterior predictive density is

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f}) = \mathcal{N}(\mathbf{f}_*|\mu_*, \mathbf{\Sigma}_*)$$
$$\mu_* = \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{f}$$
$$\Sigma_* = \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{K}_*$$

Squared-Exponential Kernel

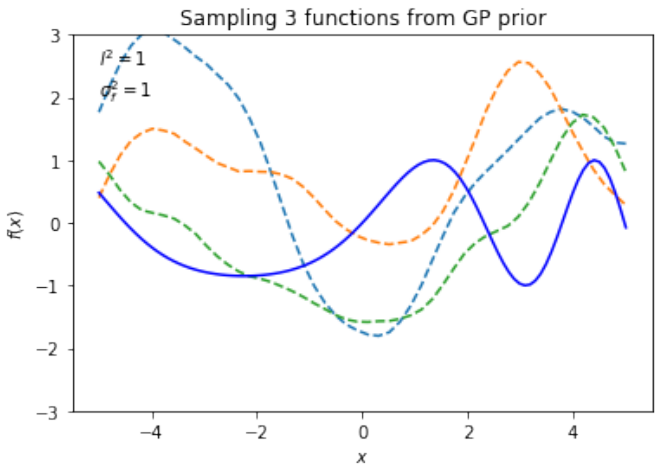
- We can use the squared-exponential kernel, aka Gaussian kernel or RBF kernel, which has the following form in 1D

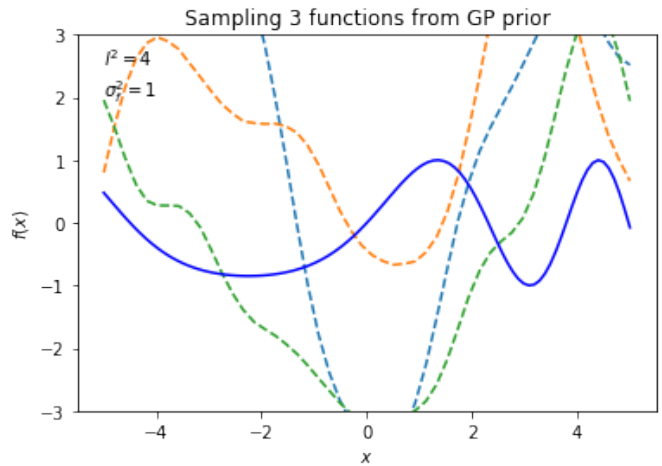
$$\mathcal{K}(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x - x')^2\right),$$

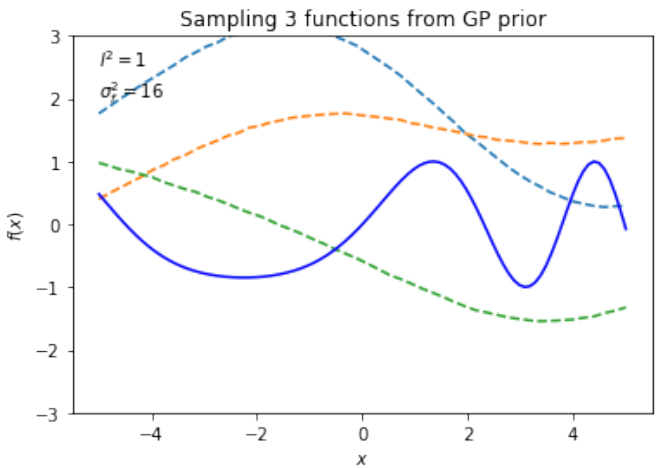
where l controls the horizontal length scale over which the function varies, and σ_f^2 controls the vertical variation.

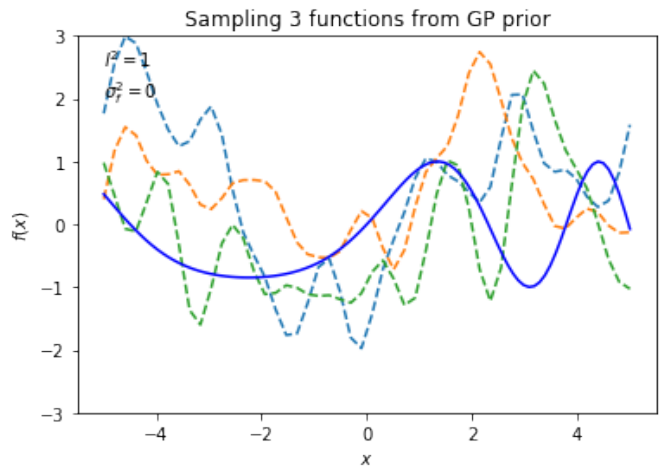
Effect of Kernel Parameters

- The performance of GPs depends upon the suitability of chosen kernel.
- For the squared-exponent kernel in the previous slide, l and σ_f^2 control the horizontal and vertical scale respectively.









Estimating kernel parameters

- The squared-exponent kernel discussed above has two parameters: l and σ_f^2.
- Exhaustive search over a discrete grid of values, with validation loss as objective.
  - Slow
  - This is how SVM (Support Vector Machines) kernels are sometimes tuned
- Maximize marginal likelihood

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f}.$$

- Recall $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_y)$ and $p(\mathbf{y}|\mathbf{f}) = \prod_i \mathcal{N}(y_i|f_i, \sigma_f^2)$, marginal likelihood is

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{X}) &= \log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_y) \\ &= -\frac{1}{2}\mathbf{y}\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{N}{2}\log(2\pi)\end{aligned}$$

Estimating kernel parameters

- Marginal likelihood:

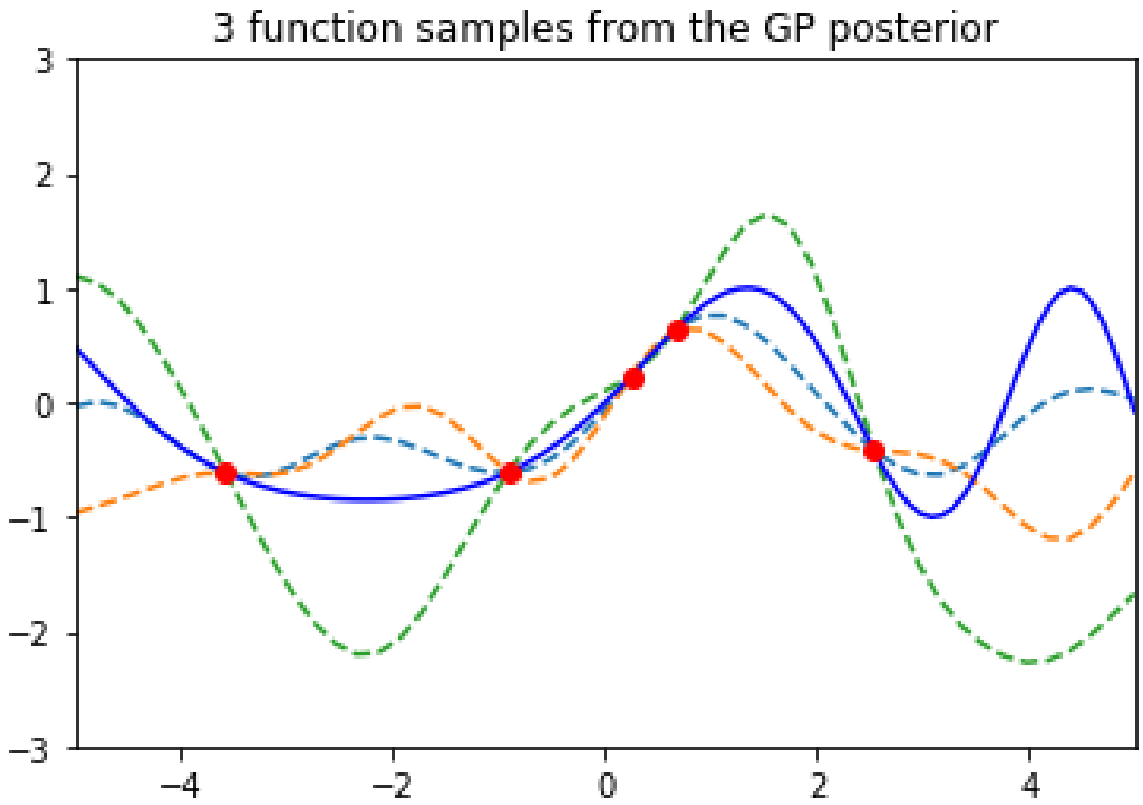
$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{N}{2}\log(2\pi)$$

  - The first term is a data fit term
  - The second term is a model complexity term
  - The last term is a constant
- Kernel parameters are often referred to as *hyper parameters*
  - $\theta = [l, \sigma_f^2]$
- Gradient descent: use

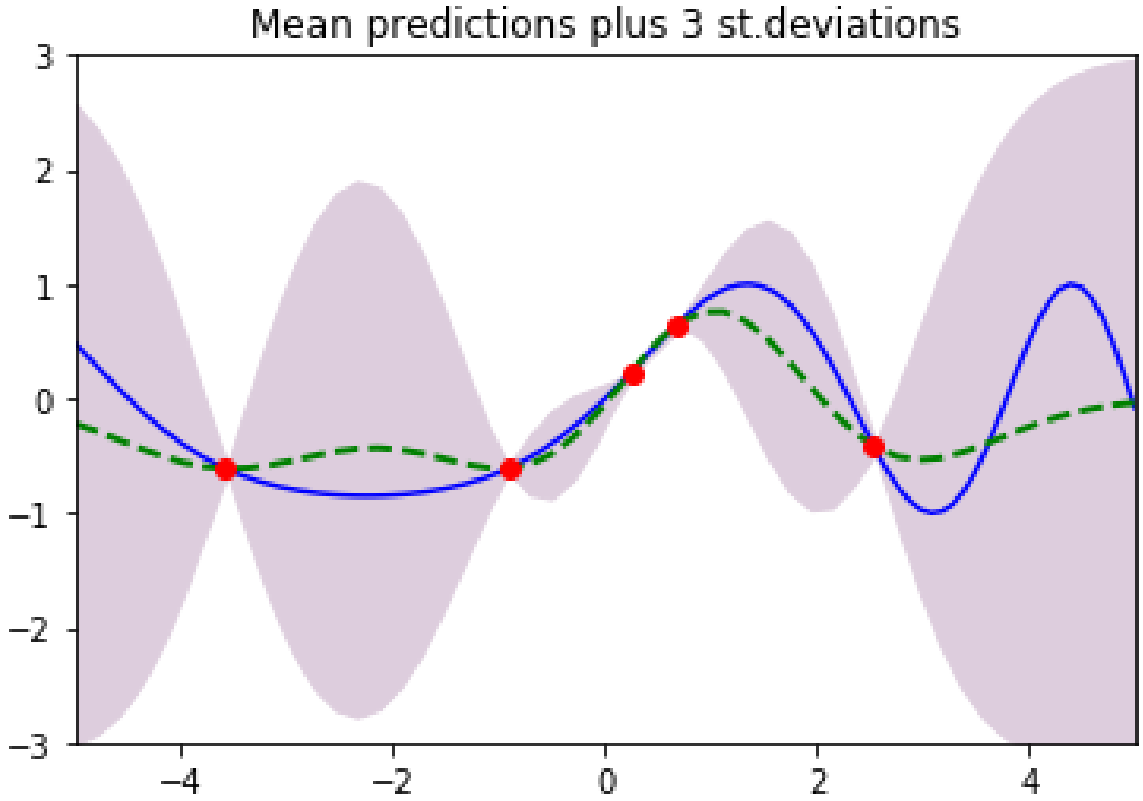
$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|\mathbf{X})$$

to maximize $\log p(\mathbf{y}|\mathbf{X})$ with respect to hyper parameter θ_j. The actual form of $\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|\mathbf{X})$ will depend upon the form of the kernel and the hyper parameter with respect to which we are trying to maximize the kernel.

GP Posterior



GP Regression



Computational concerns

For numerical stability, it is unwise to invert $\mathbf{K}$ or $\mathbf{K_y}$. Rather use the following algorithm for perform GP regression.

Algorithm 15.1: GP regression

1 $\mathbf{L} = \text{cholesky}(\mathbf{K} + \sigma_y^2 \mathbf{I})$;
2 $\boldsymbol{\alpha} = \mathbf{L}^T \setminus (\mathbf{L} \setminus \mathbf{y})$;
3 $\mathbb{E}[f_*] = \mathbf{k}_*^T \boldsymbol{\alpha}$;
4 $\mathbf{v} = \mathbf{L} \setminus \mathbf{k}_*$;
5 $\text{var}[f_*] = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^T \mathbf{v}$;
6 $\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{N}{2}\log(2\pi)$

(See Section 15.2.5.)

GP using Python

- Check out http://scikit-learn.org/stable/modules/gaussian_process.html