RnD Semester 5 - Gaussian Processes

Guide: Prof. Suyash Awate
Submitted by: Vijaykrishna G

Contents

List of Figures

1 Gaussian Process Regression

1.1 General Theory

Let $x = \{x_i\}$ denote the set of data points and $f = \{f_i\}$ denote the respective function values observed. We wish to recover the original function from which the data was sampled (in the case of my code, it was the sine function). More specifically we would like to obtain a (value, confidence) pair for any query point $x_\star$.

To accomplish this, we take a zero mean gaussian process prior over the space of real valued functions. Given this prior, we can calculate the conditional probabilities $p(f_\star|f)$ using the standard results of gaussian random variables. Some notations:

1. K denotes the covariance matrix with $K(i,j) = exp(-\frac{(x_i-x_j)^T(x_i-x_j)}{2l^2})$
2. $K_\star$ is a column matrix with $K_\star(i) = exp(-\frac{(x_i-x_\star)^T(x_i-x_\star)}{2l^2})$

Now, we want to calculate $p(f_\star|f)$. In GP any finite dimensional distribution is a multivariate gaussian with covariance as given by the squared exponential function. Hence $[f, f_\star]$ is multivariate gaussian and using standard results of conditional distributions in a multivariate gaussian,

$$\boxed{\begin{aligned} f_\star|f &\sim \mathcal{N}(\mu, \sigma^2) \\ \mu &= K_\star^T K^{-1} f \\ \sigma^2 &= 1 - K_\star^T K^{-1} K_\star \end{aligned}}$$

With the above analysis it is easy to draw the recovered function with error bars (as done for the sine function in my code).

1.2 Automatic Parameter Tuning

The parameter l, the kernel bandwidth has to be tuned manually generally. However, we can use MLE estimation to automatically tune l. For convenience, $\theta = 2l^2$. Since $f = \{f_i\}$ came from a MVG, the likelihood

$$L = (2\pi)^{-\frac{n}{2}}|K|^{-\frac{1}{2}}exp(-f^TK^{-1}f)$$

$$LL \propto -\frac{1}{2}log(|K|) - f^TK^{-1}f$$

Note that K is a function of θ i.e $K(i,j) = exp(-\frac{(x_i-x_j)^T(x_i-x_j)}{\theta})$. The derivatives of the quantities in the expression of LL are well known and thus one gets

$$\boxed{\frac{\partial LL}{\partial \theta} = \frac{n}{2} \cdot \frac{f^TK^{-1}K_\theta K^{-1}f}{f^TK^{-1}f} - \frac{1}{2}tr(K^{-1}K_\theta)}$$

where $K_\theta = \frac{\partial K}{\partial \theta}$. After this we use gradient descent to get θ_{MLE} and use that as our automatically tuned kernel bandwidth parameter.

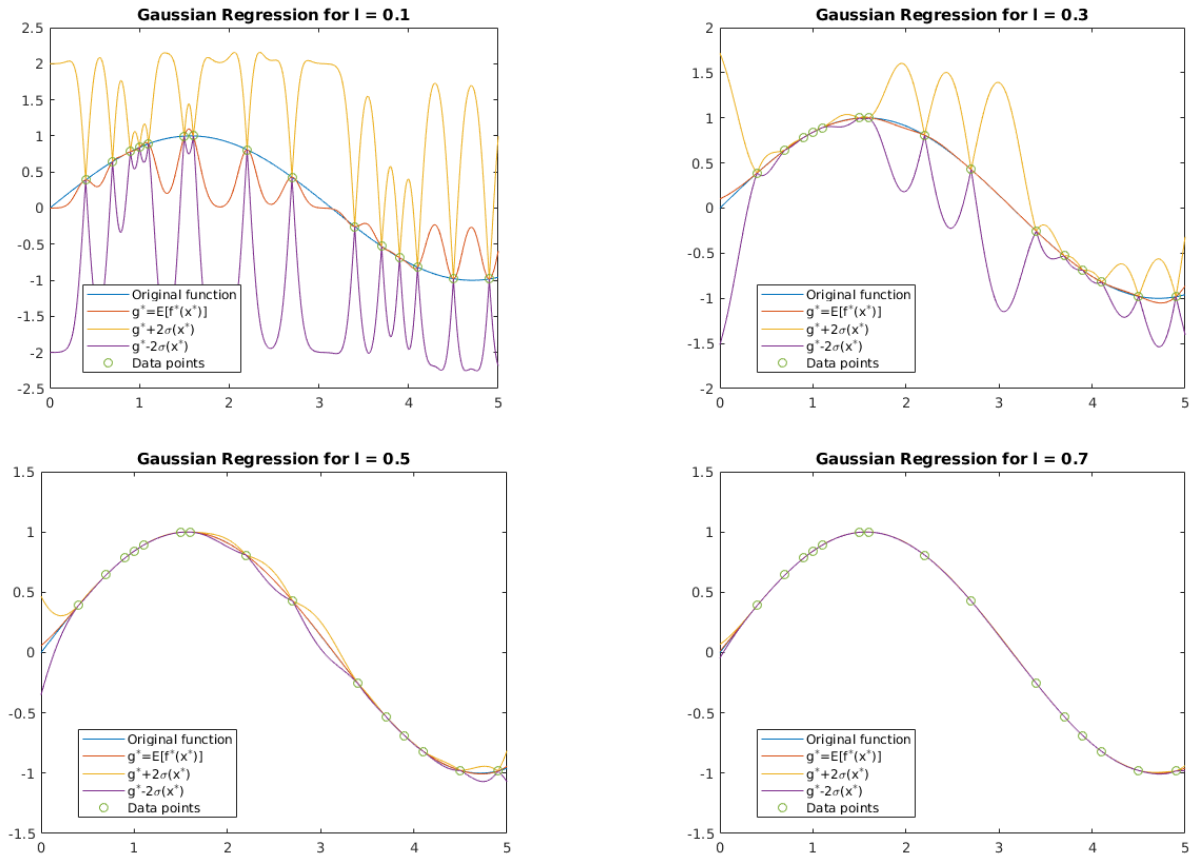


Figure 1: GP Regression performance with varying l (kernel parameter)

2 Gaussian Process Classification

2.1 Model

1. A function (or, infinite dimensional random variable) $f(u)$ gets picked from a Gaussian process distribution over real valued functions. This is the latent variable.

2. For each u, $f(u)$ is transformed to either $y(u) = -1$ or $y(u) = 1$ with $p(y(u) = 1) = \sigma(f(u))$, where $\sigma(u)$ is the sigmoid function.

2.2 Quantities involved

Let $x = \{x_i\}$ and $f = \{f_i\}$ denote the data points and the latent variable at those points respectively. $y = \{y_i\}$ denotes the actual observed entity i.e -1 or 1. In all conditional probabilities that follow we omit x since that is fixed at the start and is not really a random variable. To classify a point $x_\star$, we evaluate $E[f_\star|y]$ and check if it is positive or negative. For this, we require the following integrals. Note that the second uses the first integral.

1. $p(f_\star|y) = \int p(f|y)p(f_\star|f)df$

2. $E[f_\star|y] = \int p(f_\star|y) f_\star df_\star$

Note that:

1. Identical to the explanation in Section 1, $p(f_\star|f)$ is univariate gaussian with $\mu = K_\star^T K^{-1} f$ and $\sigma^2 = 1 - K_\star^T K^{-1} K_\star$
2. $p(f|y) = \frac{p(y|f)p(f)}{p(y)} = \frac{1}{p(y)} \left(\Pi_{i=1}^n \sigma(y_i f_i)\right) \cdot \mathcal{N}_{0,K}(f)$

2.3 Laplace Approximation

The integrals above cannot be solved analytically, hence we will use Laplace approximation (which is essentially a second order Taylor series expansion) for $p(f|y)$ as follows. $q(f|y) = \mathcal{N}(\hat{f}, A^{-1})$, with

1. $\hat{f} = \text{argmax}_f \, p(f|y)$
2. $A = -[\nabla\nabla log(p(f|y)]_{f=\hat{f}}$

The task now is to compute $\hat{f}$. We take $log(p(f|y))$ and optimize. The following are relevant (gradients are all wrt f) :

1. $log(p(f|y)) \propto log(p(y|f)) - \frac{1}{2} f^T K^{-1} f$
2. $\nabla log(p(f|y)) = \nabla log(p(y|f)) - K^{-1} f$
3. $\nabla\nabla log(p(f|y)) = \nabla\nabla log(p(y|f)) - K^{-1} = -(K^{-1} + W)$, where $W = -\nabla\nabla log(p(y|f))$

2.4 Newton Method

Newton method finds a solution to $\psi(f) = 0$ by the iterative update $f^{new} = f - (\nabla\psi)^{-1}\psi$. Here in our case, $\psi = \nabla log(p(f|y)$. Substitution and an extra simplification step leads to the iterative update

$$\boxed{f^{new} = (K^{-1} + W)^{-1}(Wf + \nabla log(p(y|f)))}$$

2.5 Putting it all together

So, now we have the approximate function $q(f|y)$ for $p(f|y)$, i.e $q(f|y) = \mathcal{N}(\hat{f}, (K^{-1} + W)^{-1})$. If we substitute this back into the integrals,

$$\boxed{\begin{aligned} E[f_\star|y] &= \int_{f_\star} \left(\int_f p(f|y) p(f_\star|f) df \right) f_\star df_\star \\ &= \int_f p(f|y) \left(\int_{f_\star} f_\star p(f_\star|f) df_\star \right) df \\ &= \int_f p(f|y) \left(K_\star^T K^{-1} f \right) df \\ &= K_\star^T K^{-1} \hat{f} \end{aligned}}$$

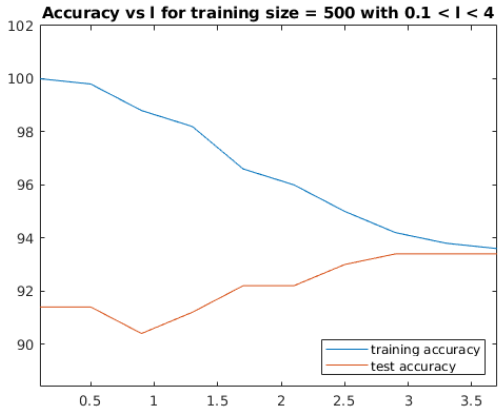
2.6 Monte Carlo

As seen in subsection 2.5,

$$\begin{aligned} E[f_\star|y] &= \int_f p(f|y)\left(K_\star^T K^{-1} f\right) df \\ &= \int_f \frac{1}{p(y)} \left(\Pi_{i=1}^n \sigma(y_i f_i)\right) \left(K_\star^T K^{-1} f\right) \cdot \mathcal{N}_{0,K}(f) df \\ &\propto \int_f \left(\Pi_{i=1}^n \sigma(y_i f_i)\right) \left(K_\star^T K^{-1} f\right) \cdot \mathcal{N}_{0,K}(f) df \end{aligned}$$

This is the expectation of a function under the distribution $\mathcal{N}_{0,K}(f)$ and hence we can use Monte Carlo to check if $E[f_\star|y] > 0$ by sampling from a probability distribution.

(a) 2D dataset



(b) Performance with varying l (kernel parameter)

Figure 2: GP Classification with Laplace approximation

3 Markov Chain Monte Carlo

In the previous section, we saw that we can use monte carlo methods to approximately evaluate the integral. In this section, we will use some low variance MCMC methods like metropolis, HMC.

3.1 Metropolis method

To sample from a pdf $P(x)$, perform the following algorithm.

1. Pick x_0 randomly.
2. For i in 1 to N:
   (a) Pick a point x' using the proposal density $Q(x'; x(i))$, which is usually a symmetric gaussian centred at $x(i)$

(b) Set $x(i+1) = x'$, with probability $min\left(1, \frac{P(x')}{P(x)}\right)$

(c) Else set $x(i+1) = x(i)$

3.2 Hamiltonian Monte Carlo

We define position x, momentum p, potential energy $E(x)$, total energy (hamiltonian) $H(p, x)$ and the relation between all of these:

$$\boxed{\begin{aligned} \dot{x} &= p \\ \dot{p} &= -\nabla E(x) \\ H(p,x) &= E(x) + \frac{p^T p}{2} \end{aligned}}$$

In HMC, we use the potential energy function $E(x) = -log(P(x))$, where $P(x)$ is the distribution we are interested in sampling from. To sample from a pdf $P(x)$, perform the following algorithm:

1. Pick x_0 randomly.
2. For i in 1 to N:
   (a) Pick a random momentum $p(i)$ from a gaussian distribution.
   (b) Run the simulation in (p, x) for τ steps (leapfrog steps) each step being of duration δt.
   (c) Let x', p' be the new values of x, p. Also let $H' = H(p', x')$ and $H(i) = H(p(i), x(i))$. Set $x(i+1) = x'$, with probability $min\left(1, e^{-(H' - H(i))}\right)$
   (d) Else set $x(i+1) = x(i)$
3. When burn-in technique is used, remove some number of samples in the start.

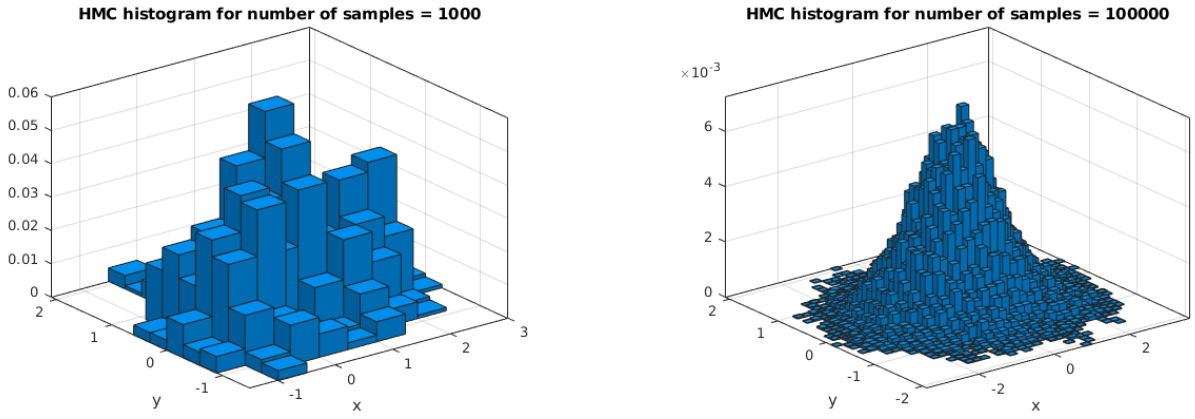


Figure 3: Histogram for a 2D gaussian using HMC for $10^3, 10^5$ samples

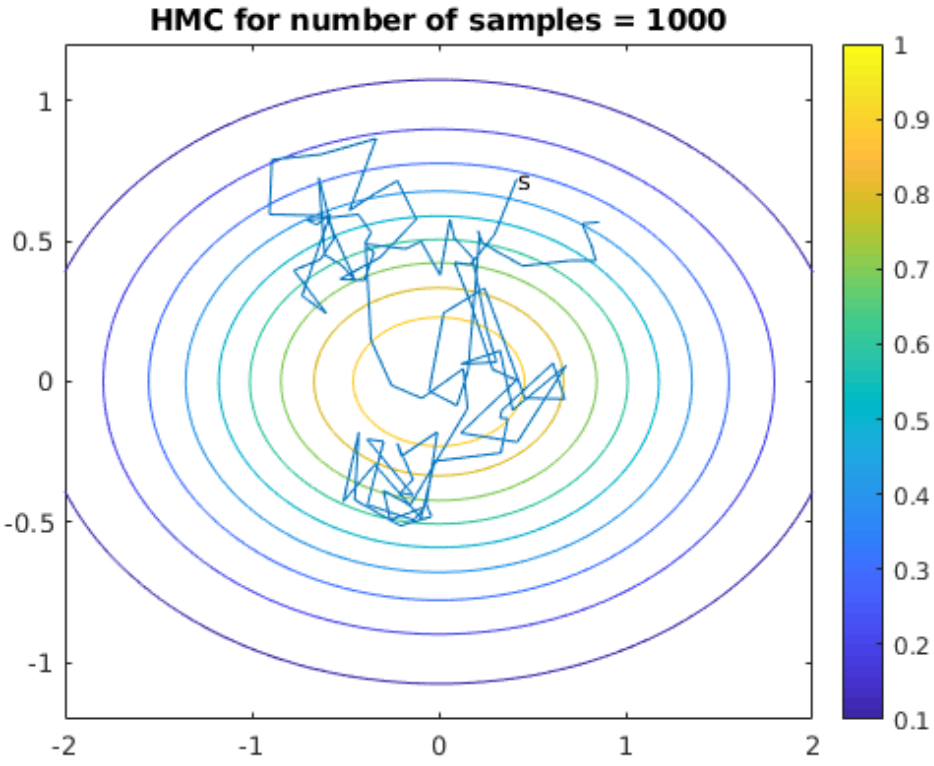
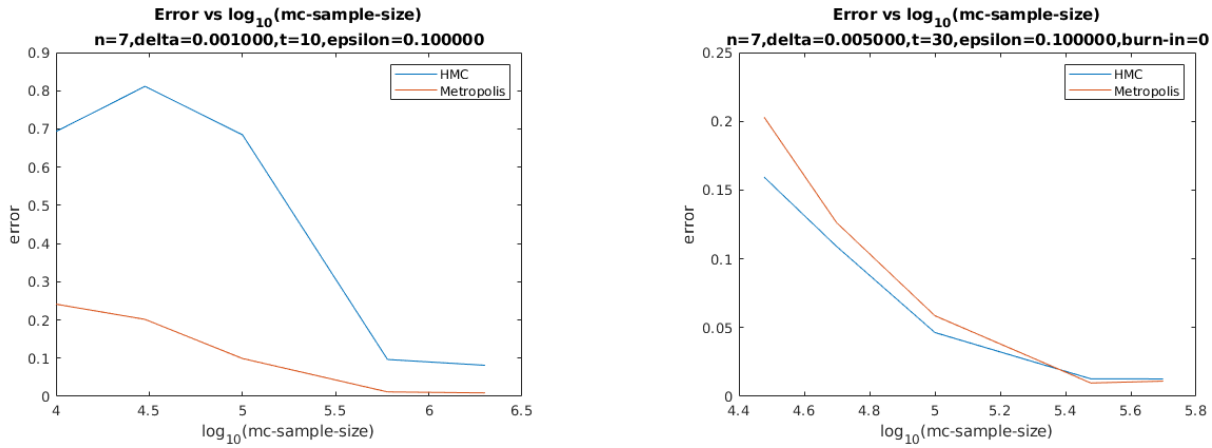


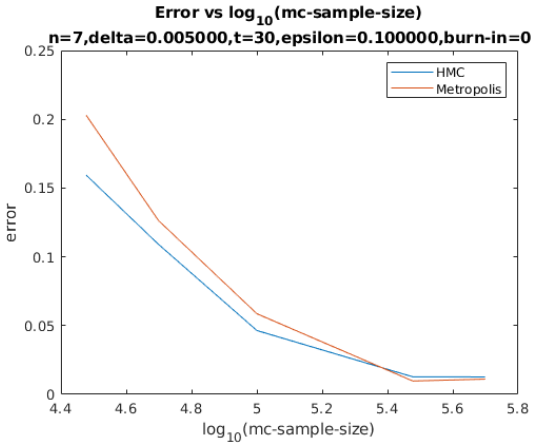
Figure 4: Path for first 100 points in HMC for a 2D gaussian



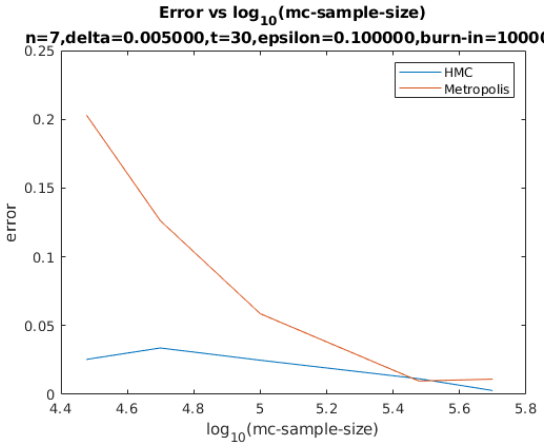
(a) $\delta t = 0.001$, $\tau = 10$

(b) $\delta t = 0.005$, $\tau = 30$

Figure 5: Effect of increasing δt, τ but no burn-in in HMC for a 7D gaussian

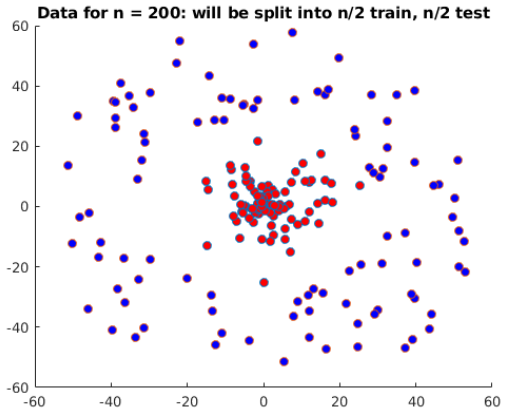


(a) $\delta t = 0.005$, $\tau = 30$, without burn-in

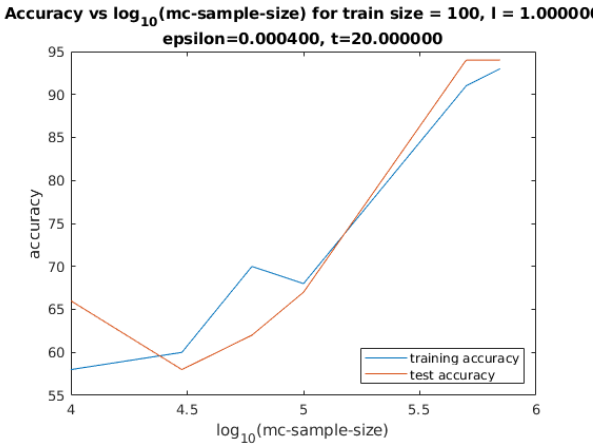


(b) $\delta t = 0.005$, $\tau = 30$ with burn-in

Figure 6: Effect of burn-in at fixed δt, τ in HMC for a 7D gaussian



(a) 2D dataset



(b) Performance variation with number of samples

Figure 7: HMC with burn-in for GP Classification