Sareh Kouchaki

SDS 383: Modeling II – Section 4

4.1 Non-linear functions

4.1.1 Regression view

So far, we've assumed our latent function is a linear function of our data – which is obviously limiting. One way of circumventing this is to project our inputs into some high-dimensional space using a set of basis functions $\phi : \mathbb{R}^d \to \mathbb{R}^N$, and then performing linear regression in that space, so that

$$y_i = \phi(x)^T \beta + \epsilon_i$$

For example, we could project x into the space of powers of x, i.e. $\phi(x) = (1, x, x^2, x^3 \ldots)$ to obtain polynomial regression.

Exercise 4.1 *Let* $\mathbf{y}$ *and* $\mathbf{X}$ *be set of observations and corresponding covariates, and* y_* *be the unknown value we wish to predict at covariate* $\mathbf{x}_*$. *Assume that*

$$\begin{aligned} \beta &\sim N(0, \Sigma) \\ \begin{bmatrix} f_* \\ \mathbf{f} \end{bmatrix} &= \begin{bmatrix} \phi_*^T \\ \mathbf{\Phi}^T \end{bmatrix}^T \beta \\ \begin{bmatrix} y_* \\ \mathbf{y} \end{bmatrix} &\sim N\left(\begin{bmatrix} f_* \\ \mathbf{f} \end{bmatrix}, \sigma^2 \mathbf{I} \right) \end{aligned}$$

where $\phi := \phi(\mathbf{x})$ *and* $\mathbf{\Phi} := \phi(\mathbf{X})$.

What is the predictive distribution $p(f_*|\mathbf{y}, \mathbf{x}_*, \mathbf{X})$? *Note: this is very similar to questions we did in Section 1.*

Solution:

The procedure to solve this question is similar to what we did in section 1. The result is:

We assume $\epsilon \sim N(0, \sigma_n^2)$.

$$p(f_*|y, x_*, X) = N\left(\frac{1}{\sigma_n^2} \phi(x_*)^T A^{-1} \Phi y, \phi(x_*)^T A^{-1} \phi(x_*)\right)$$

where $A = \sigma_n^{-2} \Phi \Phi^T + \Sigma^{-1}$

Note that, in the solution to Exercise 1, we only ever see ϕ or Φ in a form such as $\Phi^T \Sigma \Phi$. We will define $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \Sigma \phi(\mathbf{x}')$. Since Σ is positive definite, we can write:

$$k(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x})^T \psi(\mathbf{x})$$

where $\psi(\mathbf{x}) = \phi(\mathbf{x}) \Sigma^{1/2}$

If (as here) we only ever access ψ via this inner product, we can choose to work instead with $k(\cdot, \cdot)$. This may be very convenient if the dimensionality of $\psi(x)$ is very high (or even infinite... see later). $k(\cdot, \cdot)$ is often refered to as the kernel, and this replacement is referred to as the kernel trick.

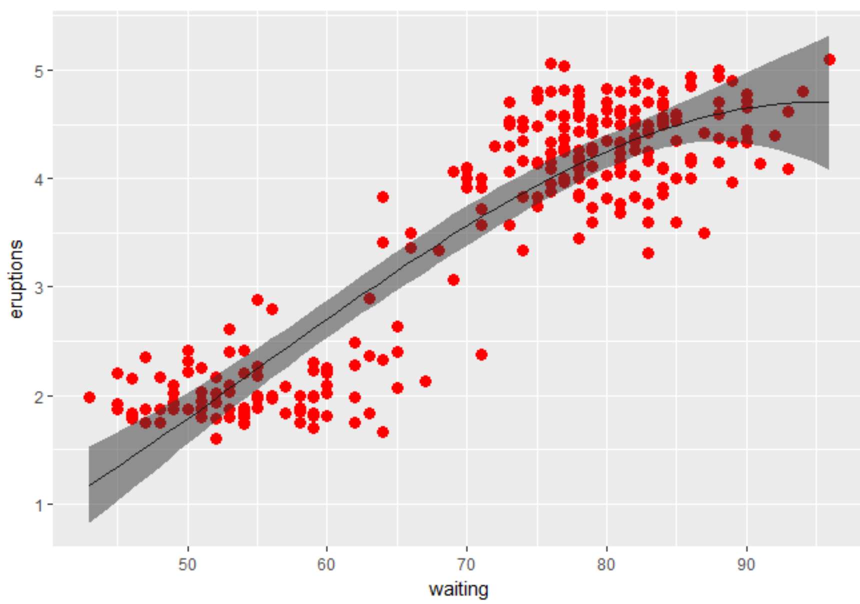
Exercise 4.2 *Let's look at a concrete example, using the old faithful dataset on R*

- `data("faithful",package="datasets")` *in R*
- *or available as* `faithful.csv` *on github if you're not using R.*

Let $\phi(x) = (1, x, x^2, x^3)$*. Using appropriate priors on* β *and* σ^2*, obtain a posterior distribution over* $f := \phi(x)^T \beta$*. Plot the function (with a 95% credible interval) by evaluating this on a grid of values.*

Solution:

Code is available on GitHub.



4.1.2 Function space view

Look back at the plot from Exercise 2. We specified a prior distribution over regression parameters, which we can use to obtain a posterior distribution over those regression parameters. But, what we calculated (and plotted) was a posterior distribution over *functions*. Similarly, we can think of our prior on β as specifying a prior distribution on the space of cubic functions. Evaluated at a finite number of input locations – as you did in Exercise 2 – this posterior distribution is multivariate Gaussian. This is in fact the definition of a Gaussian process: A distribution over functions, such that the marginal distribution evaluated at any finite set of points is multivariate Gaussian.

A priori, the covariance of f is given by

$$\text{cov}(x, x') = E[(f(x) - m(x))(f(x^T) - m(x^T))] = k(x, x')$$

. For this reason, our kernel k is often referred to as the covariance function (note, it is a function since we can evaluate it for any pairs x, x'). In the above example, where β had zero mean, the mean of f is zero; more generally, we will assume some mean function $m(x)$.

Rather than putting a prior distribution over β, we can specify a covariance function – remember that our covariance function can be written in terms of the prior covariance of β. For example, we might let

$$k(x, x') = \alpha^2 \exp\left\{-\frac{1}{2\ell^2}|x - x'|^2\right\}$$

– this is known as a squared exponential covariance function, for obvious reasons. This prior encodes the following assumptions:

- The covariance between two datapoints decreases monotonically as the distance between them increases.
- The covariance function is stationary – it only depends on the distance between x and x', not their locations.
- Even more than being stationary, it is isotropic: It depends only on $|x - x'|$.

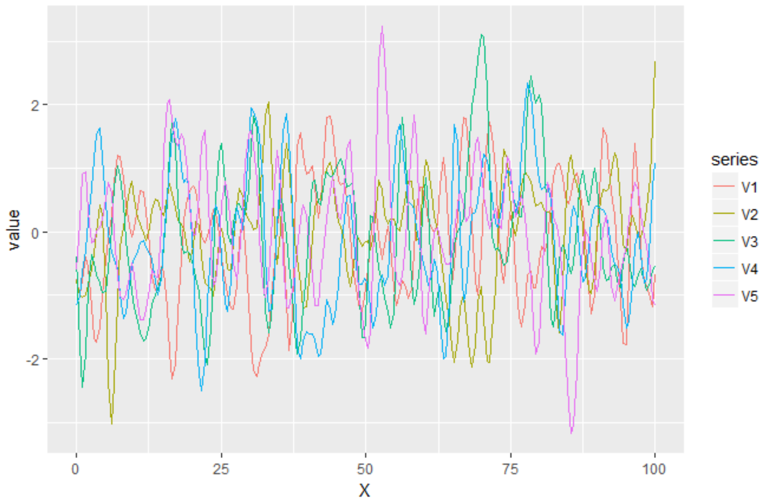
Exercise 4.3 *Let's explore the resulting distribution over functions. Write some code to sample from a Gaussian process prior with squared exponential covariance function, evaluated on a grid of 200 inputs between 0 and 100. For $\ell = 1$, sample 5 functions and plot them on the same plot. Repeat for $\ell = 0.1$ and $\ell = 10$. Why do we call ℓ the lengthscale of the kernel?*

Solution:

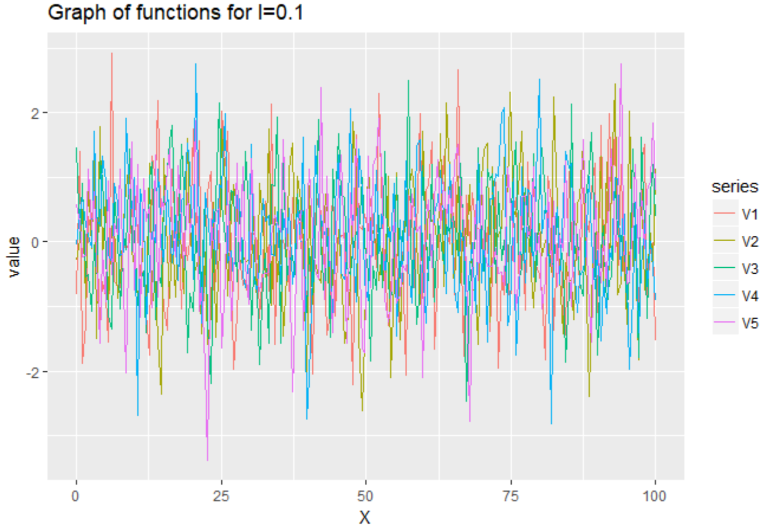
Code is available on GitHub.

The following plots show how modification of l affects the function. By modifying l we can scale the kernel.

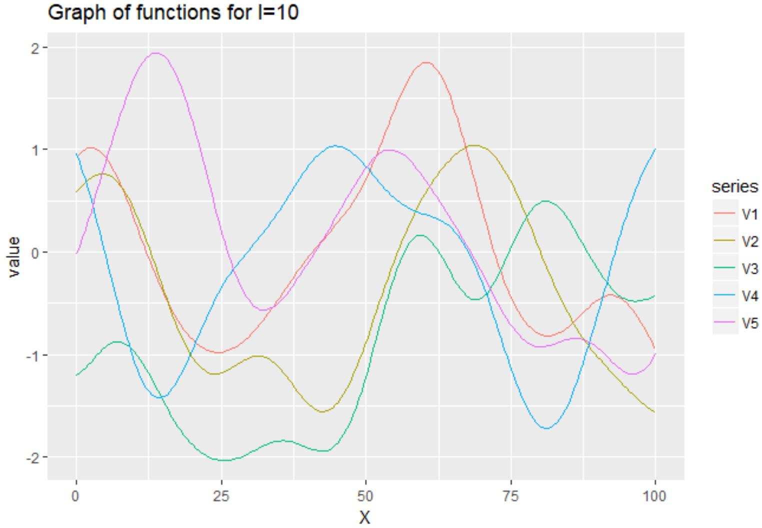
Graph of functions for l=1



Graph of functions for l=0.1



Graph of functions for l=10



Exercise 4.4 *Let* $\mathbf{f}_* := f(\mathbf{X}_*)$ *be the function* f *evaluated at test covariate locations* $\mathbf{X}_*$. *Derive the posterior distribution* $p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y})$, *where* $\mathbf{y}$ *and* $\mathbf{X}$ *comprise our training set. (You can start from the answer to Exercise 1 if you'd like).*

Solution:

According to "Gaussian process for machine learning" book:

$$y = f(X) + \epsilon$$

where $\epsilon \sim N(0, \sigma_n^2)$

So, the covariance of y would be:

$$cov(y) = K(X, X) + \sigma_n^2 I$$

we can write the joint distribution of the observed target values and the function values at the test locations under the prior as:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right)$$

By deriving the conditional distribution, the predictive equations for Gaussian process regression would be:

$$f_*|X, y, X_* \sim N\,(mean(f_*), cov(f_*))$$

Where

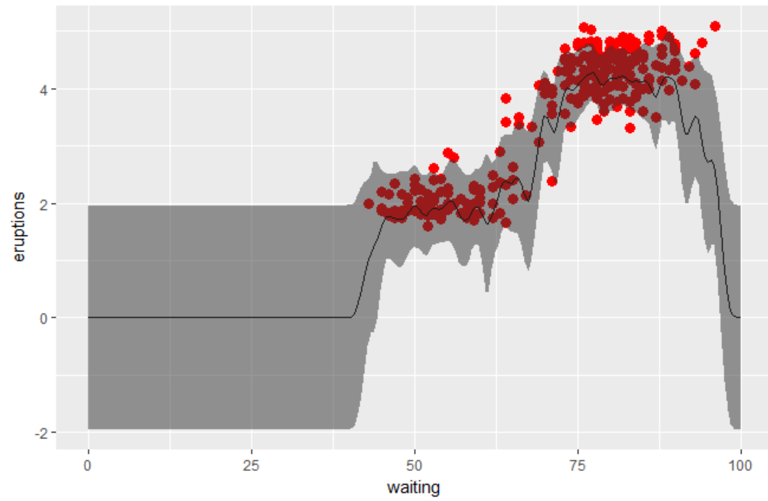
$$mean(f_*) = K(X_*, X)\ [K(X, X) + \sigma_n^2 I]^{-1} y$$

$$cov(f_*) = K(X_*, X_*) - K(X_*, X)\,[K(X, X) + \sigma_n^2 I]^{-1}\, K(X, X_*)$$

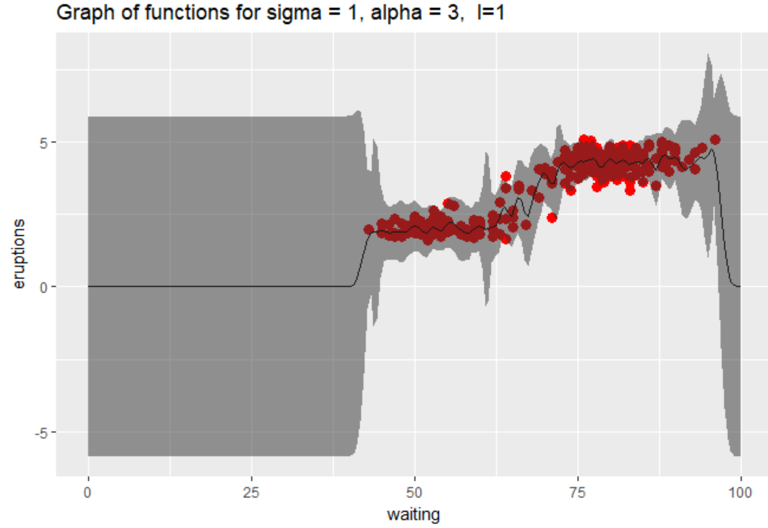
Exercise 4.5 *Return to the faithful dataset. Evaluate the posterior predictive distribution* $p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y})$, *for some reasonable choices of parameters (perhaps explore a few length scales if you're not sure what to pick), and plot the posterior mean plus a 95% credible interval on a grid of 200 inputs between 0 and 100, overlaying the actual data.*

Solution: Code is available on GitHub.

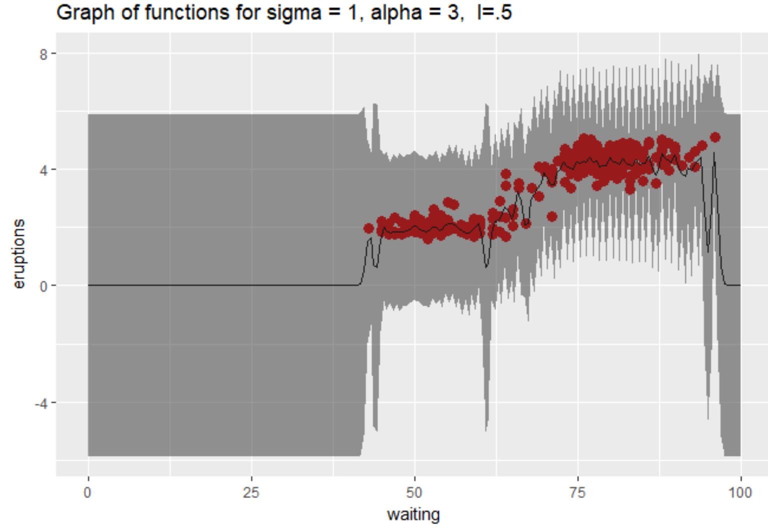
Graph of functions for sigma = 1, alpha =1, l=1



Graph of functions for sigma = 1, alpha = 3, l=1



Graph of functions for sigma = 1, alpha = 3, l=.5



4.2 Parameter Estimation

As we saw in the previous section, the choice of hyperparameters (for the squared exponential case, the length scale ℓ) effects the properties of the resulting function. Rather than pick a specific value for the hyperparameter, we can specify the model in a hierarchical manner—just like we did in the linear case.

For example, in the squared exponential setting, we could specify our model as

$$\begin{aligned}
\ell^2 \sim& \text{Inv-Gamma}(a_\ell, b_\ell) \\
\alpha^2 \sim& \text{Inv-Gamma}(a_\alpha, b_\alpha) \\
\sigma^2 \sim& \text{Inv-Gamma}(a_\sigma, b_\sigma) \\
k(x, x') =& \alpha^2 \exp\left\{-\frac{1}{2\ell^2}|x - x'|^2\right\} + \sigma^2 \delta_{x-x'} \\
y|X \sim& N(0, \tilde{K})
\end{aligned}$$

where K is the covariance function evaluated at the input locations X. Note that we have integrated out f and placed our prior directly on y, incorporating the Gaussian likelihood into the covariance. We can then infer the posterior distribution over ℓ using Bayes' Law:

$$p(\ell|y, X) = \frac{p(y|X, \ell)p(\ell)}{\int_0^\infty p(y|X, \ell)p(\ell)d\ell}$$

Unfortunately, we typically do not have an analytical form for this posterior, so we must resort to either optimization, or MCMC-based inference.

4.2.1 Optimization

In practice, a common approach is to find the ML estimate for the hyperparameters. Let's assume a generic setting, where the log likelihood is parametrized by some vector of parameters θ. The log likelihood is given by

$$logp(y|X, \theta) = -\frac{1}{2}y^T K^{-1} y - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi$$

Taking partial derivatives, we see that

$$\begin{aligned}
\frac{\partial}{d\partial\theta_j}\log p(y|X, \theta) =& \frac{1}{2}y^T K^{-1}\frac{\partial K}{\partial\theta_j}K^{-1}y - \frac{1}{2}\text{tr}\left(K^{-1}\frac{\partial K}{\partial\theta_j}\right) \\
=& \frac{1}{2}\text{tr}\left((\alpha\alpha^T - K^{-1})\frac{\delta K}{\delta\theta_j}\right)
\end{aligned}$$

where $\alpha = K^{-1}y$. We can use these partial derivatives to find the ML estimate of θ, using a gradient-based optimization method

Exercise 4.6 *Calculate the appropriate derivatives for the one-dimensional, squared exponential case used for the* `faithful` *dataset. Use these gradient to find the optimizing value of ℓ^2, α^2 and σ^2. Plot the resulting fit.*

Solution:

$$K(X_i, X_j) = \begin{cases} \alpha^2 + \sigma^2 & X_i = X_j \\ \alpha^2\, exp(-\frac{|X_i - X_j|^2}{2l^2}) & X_i \neq X_j \end{cases}$$

$$\frac{\partial K}{\partial \alpha} = 2\alpha exp(-\frac{|X_i - X_j|^2}{2l^2})$$

$$\frac{\partial K}{\partial l} = \frac{\alpha^2\, |X_i - X_j|^2 exp\,(-\frac{|X_i - X_j|^2}{2l^2}}{l^3}$$

$$\frac{\partial K}{\partial \sigma} = 2\sigma$$

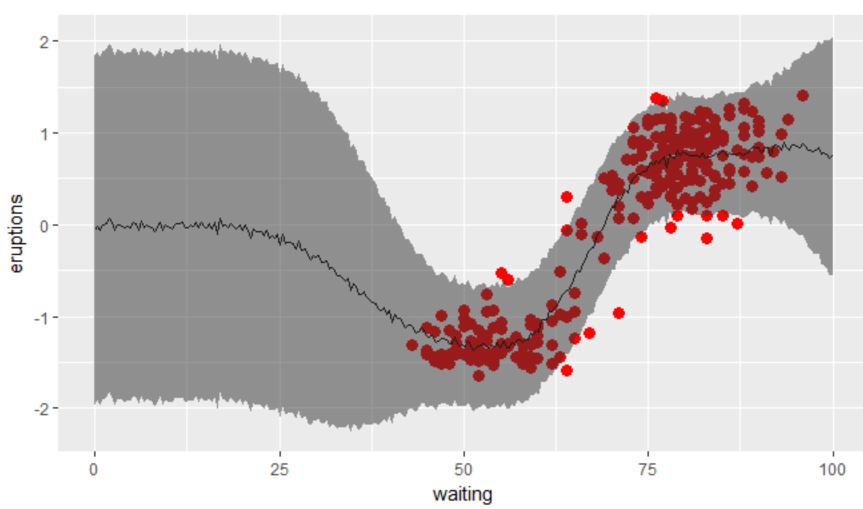
Gradients:

$$\begin{cases} \frac{1}{2} tr((aa^T - K^{-1}) * 2\alpha exp(-\frac{|X_i - X_j|^2}{2l^2})) \\ \frac{1}{2} tr((aa^T - K^{-1}) * \frac{\alpha^2\, |X_i - X_j|^2 exp\,(-\frac{|X_i - X_j|^2}{2l^2}}{l^3}) \\ \frac{1}{2} tr((aa^T - K^{-1}) * 2\sigma) \end{cases}$$

I wrote a code using the above gradient equations to find the optimizing values of parameters. It is available on GitHub under the name of 4.6.R. I checked each part works fine. But, the last part, while statement, ran for 1 hour and I had to stop it without getting any results.

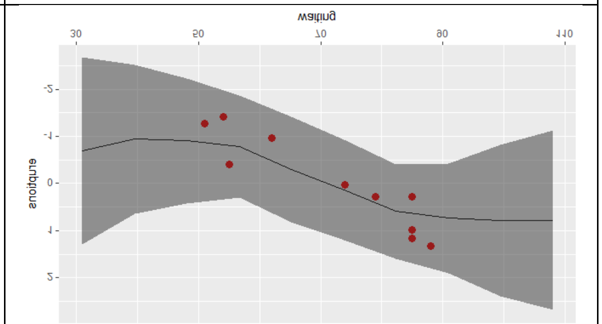
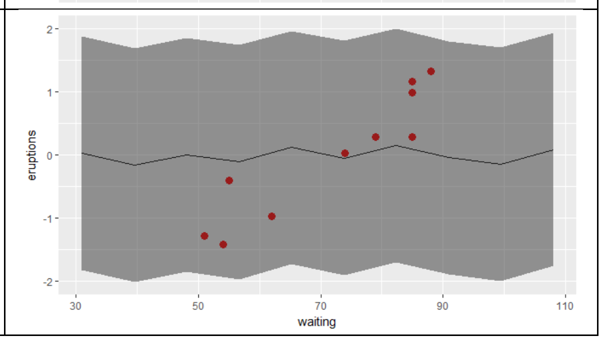
So, I decided to write another code to use the negative log likelihood function and the “optim” command in R to find the optimizing values.

Optimization values: (alpha= 0.919, l= 9.919, sigma =0.324)I tried with several initial values and I got the same results every time.



Exercise 4.7 *Repeat the previous exercise, but this time only use the first 10 data points from the faithful dataset. Repeat the optimization several times, using different initializations/random seeds. You will likely see widely different results – sometimes ℓ is big, sometimes σ^2 is big. Why is this? Discuss why this is a problem here, but wasn't in the previous setting. You may find it helpful to look at the corresponding scatter plot, or plot the log likelihood for certain values of σ^2 and ℓ.*

Solution:

Initial values Alpha, l, sigma	Optimization values Alpha, l, sigma	plot
0.5, 2, 0.4	1.167, 23.91, 0.398	
0.5, 0.08, 0.2	0.834, 0.08, 0.471	

We cold say that since we are using only 10 data points (small part of the data), the optimization function shows high uncertainty.

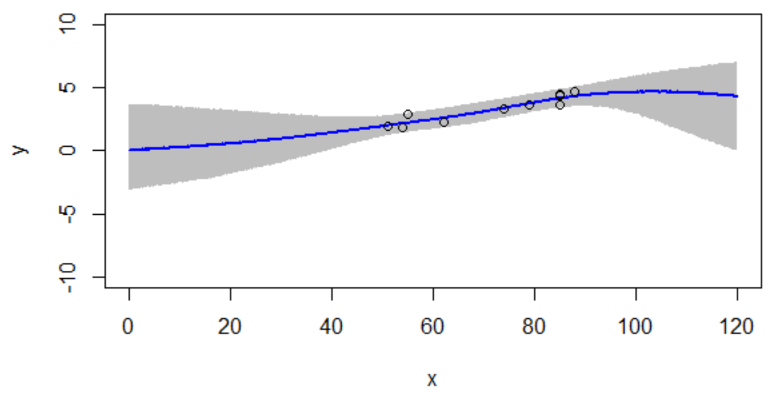
4.2.2 MCMC

Optimization is typically pretty quick, which is why it is commonly used in practice. However, we have no guarantee that our optimization surface is convex. An alternative approach is to sample from the posterior distribution over our hyperparameters.

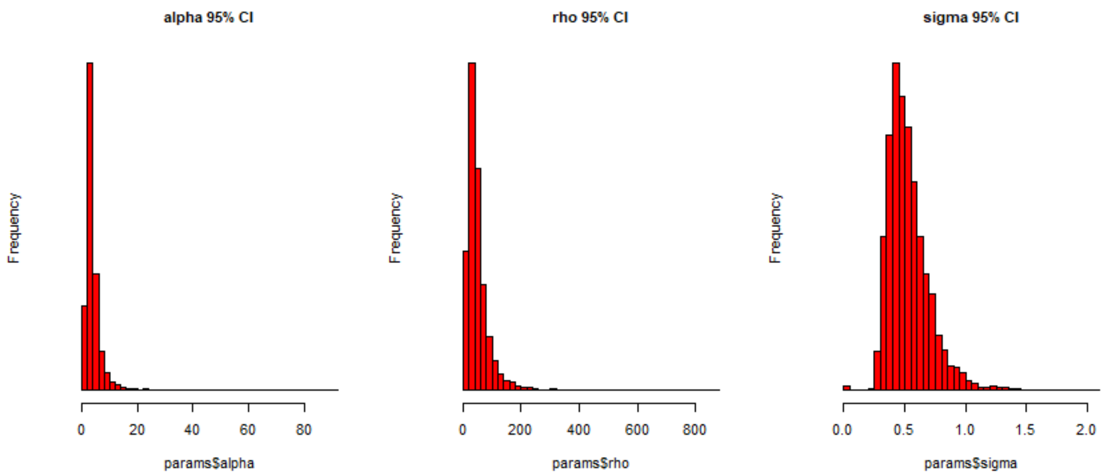
Exercise 4.8 *Since the posterior is non-conjugate, we can't use a Gibbs sampler. We won't go into the details of appropriate sampling methods since this isn't an MCMC course, but we will explore using black-box samplers. In the R folder, there are three files:* `faithful_data.R`, `gp_regression.stan` *and* `run_gp_regression.R`. *Use these to sample from the model and produce 95% credible intervals for α, ℓ and σ, and 95% predictive intervals for t. Go through the code and make sure you understand what is going on.*

Solution:

Plot of gaussian process posterior function and %95 credible interval:



Histogram of parameters (%95 credible intervals):



Exercise 4.9 *Let's now look at a dataset with multiple predictors. Download the dataset weather.csv – this contains latitude and longitude data for 147 weather stations, plus a response "temperature", which is the difference between the forecasted and actual temperature for each station.*

How should we extend our kernel to multiple dimensions? (There is more than one option here). Should we use the same lengthscale for latitude and longitude? Construct an appropriate parametrized kernel, and learn the parameters either via optimization or using MCMC by editing the Stan code (Note: If you go for the stan code, you will need to implement your new kernel).

Using an appropriate visualization tool, plot the mean function (try imshow or contourf in matlab or matplotlib (for python), or image or filled.contour for R).

Solution:

4.3 Beyond regression: non-conjugate likelihoods

So far, we've focused on Gaussian processes in a regression context. We can however use them as the basis of a non-Gaussian regression... using exactly the same techniques as we used for the regression setting! For example, in Section 3, we dealt with non-Gaussian data by transforming our regression output:

$$y_i|\beta, x_i \sim f(g^{-1}(x_i^T\beta))$$

where f is an appropriate likelihood model (e.g. Bernoulli for binary data, Poisson for count data) and g^{-1} was a function that maps the real-valued $x_i^T\beta$ to an appropriate space for that likelihood.

We can do exactly the same here, by letting

$$\begin{aligned} f \sim& \mathrm{GP}(0, k) \\ y_i \sim& f(g^{-1}(f(x_i))) \end{aligned}$$

Let's start by considering a binary example. In the GLM setting, we looked at both probit and logit regression. We can use the same approaches here!

Exercise 4.10 *Describe (including pseudo-code) how we could implement probit Gaussian process regression, using an auxiliary variable method analogous to that used in Exercise 3.1.*

For the logit case, we can again use a Laplace approximation to approximate our posterior. In the GLM setting, we used the Laplace approximation to approximate the posterior over β. Here, we will work directly with our function f evaluated at our training locations, and approximate $p(f|X, y, \theta)$, where θ are the parameters of our covariance function.

Let $P^*(f) \propto p(f|X, y, \theta)$ be our unnormalized posterior, so that $\log P^*(f) = \log p(y|f) + logp(f|X) = \log p(y|f) - \frac{1}{2}f^TK^{-1}f - \frac{1}{2}\log|K| + \text{const}$.

Exercise 4.11 *Derive the Hessian of* $\log P^*(f)$*, when* $y_i \sim$ *Bernoulli* $\left(\frac{1}{1+e^{-f_i}}\right)$

Solution:

$$\log p^*() = log\, p(Y|f) - \frac{1}{2}\, f^TK^{-1}f - \frac{1}{2}\, log|K| + const$$

$$\mathrm{p}(y_i|\mathrm{f}) = \text{Bernouli } (\frac{1}{1+e^{-f}})$$

Let's look at this probability in detail:

$$\text{Bernoulli dist:} \quad \mathrm{p}(y_i|\mathrm{p}) = p^{y_i}(1-p)^{1-y_i}$$

$$p = \frac{1}{1+e^{-f}} = \sigma(\mathrm{f})$$

$$\mathrm{p(Y|f)} = \prod_{i=1}^{n} p^{\mathrm{y}_i}(1-p)^{1-\mathrm{y}_i}$$

$$\log(\mathrm{p(Y|f)} = \sum_{i=1}^{n} \mathrm{y}_i logp + (1-\mathrm{y}_i)log(1-p)$$

So,

$$\log p^*() = \sum_{i=1}^{n} \mathrm{y}_i logp + (1-\mathrm{y}_i)log(1-p) - \frac{1}{2} f^T K^{-1} f - \frac{1}{2} log|K| + const$$

** $\frac{\partial \sigma(f)}{\partial f} = \frac{\partial}{\partial f}(1+e^{-f})^{-1} = \sigma(f)\,(1-\sigma(f))$

**$\frac{\partial log(\sigma(f))}{\partial f} = \frac{1}{\sigma(f)}\frac{\partial \sigma(f)}{\partial f} = 1-\sigma(f)$

** $\frac{\partial log(1-\sigma(f))}{\partial f} = \frac{1}{1-\sigma(f)}\frac{\partial(1-\sigma(f))}{\partial f} = -\sigma(f)$

** $\frac{\partial\,(\frac{1}{2} f^T K^{-1} f\,)}{\partial f} = \frac{1}{2}\,[f^T K^{-1} + f^T (K^{-1})^T] = \frac{1}{2}\,f^T [K^{-1} + (K^{-1})^T]$

So,

$$\frac{\partial log(p^*()\,)}{\partial f} = \sum_i (y_i\,(1-\sigma(f)) - (1 - y_i\,)(\sigma(f))) - \frac{1}{2} f^T [K^{-1} + (K^{-1})^T]$$

$$\frac{\partial^2 log(p^*()\,)}{\partial f^2} = -\sum_i \sigma(f)(1-\sigma(f) - \frac{1}{2}[K^{-1} + (K^{-1})^T]$$

** first part is the sum along diagonal, and the second part, K is symmetric, so we can put K^{-1} in the bracket.