G. Cowan
Royal Holloway Physics
December 2022, Version 1.0

Exercise on Bayesian Parameter Estimation

This exercise provides an introduction to Bayesian parameter estimation. The program `bayesFit.py` generates a data sample $\mathbf{x} = (x_1, \ldots, x_n)$ of $n = 400$ independent values sampled from a pdf that is a mixture of an exponential and a Gaussian,

$$f(x|\boldsymbol{\lambda}) = \theta \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} + (1-\theta)\frac{1}{\xi} e^{-x/\xi} \; . \tag{1}$$

Here $\boldsymbol{\lambda} = (\theta, \mu, \sigma, \xi)$ represents the vector of parameters. Below we will regard the amplitude of the Gaussian θ as the parameter of interest an the rest as nuisance parameters. The program finds the posterior probability of the parameters using Bayes' theorem, finds its maximum to obtain the MAP (maximum a posteriori probability) estimators, and then marginalizes over the nuisance parameters using Markov Chain Monte Carlo (MCMC).

Generating the data and maximizing the posterior pdf are done in `bayesFit.py` in essentially the same way as in the maximum-likelihood fitting program `mlFit.py`. Please refer to the material for the exercise with `mlFit.py` for more information on those steps, including installation of the `iminuit` package.

The program first finds the posterior pdf $p(\boldsymbol{\lambda}|\mathbf{x})$ for the parameters using Bayes' theorem,

$$p(\boldsymbol{\lambda}|\mathbf{x}) \propto p(\mathbf{x}|\boldsymbol{\lambda})\pi(\boldsymbol{\lambda}) \; , \tag{2}$$

where

$$p(\mathbf{x}|\boldsymbol{\lambda}) = \prod_{i=1}^{n} f(x_i|\boldsymbol{\lambda}) \tag{3}$$

is the likelihood (probability for the data given the parameters) and $\pi(\boldsymbol{\lambda})$ is the prior pdf for the parameters. By initial default, the joint prior pdf for all the parameters $\boldsymbol{\lambda} = (\theta, \mu, \sigma, \xi)$ is taken to be a constant.

The program first finds the maximum of the posterior pdf using the program `iminuit`. The resulting parameter values are the MAP estimators, which coincide with the maximum likelihood estimators (MLEs) for the special case where the prior is constant.

Next, the program marginalizes the posterior over the nuisance parameters with the Metropolis-Hastings MCMC algorithm. The algorithm samples the full parameter space according to the posterior pdf (here called the target density) using an iterative rule to proceed from a point $\boldsymbol{\lambda}_i$ to a new value $\boldsymbol{\lambda}_{i+1}$:

1. Generate a proposed point $\boldsymbol{\lambda} \sim q(\boldsymbol{\lambda}|\boldsymbol{\lambda}_i)$, where the proposal density q is a multivariate Gaussian with mean $\boldsymbol{\lambda}_i$ and covariance matrix U, described below.

2. Calculate the ratio $\alpha = \min\left[1, q(\boldsymbol{\lambda}|\boldsymbol{\lambda}_i)/q(\boldsymbol{\lambda}_i|\boldsymbol{\lambda})\right]$.

3. Generate $u \sim \text{Uniform}[0, 1]$.

4. If $u < \alpha$, accept the proposed point, i.e., $\boldsymbol{\lambda}_{i+1} = \boldsymbol{\lambda}$, otherwise repeat the old point, i.e., $\boldsymbol{\lambda}_{i+1} = \boldsymbol{\lambda}_i$.

As the starting point $\boldsymbol{\lambda}_0$ one can use any point that is reasonably well contained within the bulk of the target density's probability; by default the program `bayesFit.py` uses the MAP estimate.

The covariance matrix U of the proposal density is taken here to be a scaled version of the covariance matrix V from the peak of the posterior pdf, i.e.,

$$U = sV \; . \tag{4}$$

The value of s can be adjusted to minimize the number of computing steps needed to accurately determine the marginal pdfs of the parameters. The autocorrelation function described below provides a quantitative measure of performance. As an approximate guide (see, e.g., Gareth O. Roberts and Jeffrey S. Rosenthal, Statistical Science 2001, Vol. 16, No. 4, 351–367) one can take

$$s = \frac{(2.38)^2}{N_{\rm dim}} \; , \tag{5}$$

where $N_{\rm dim}$ is the number of dimensions of the parameter space. This rule provides the optimal convergence if the parameters are independent, and corresponds to the fraction A of proposed values that are accepted in step (4) above of $A = 0.234$.

The autocorrelation function (ACF) is the correlation coefficient between an element of the series of generated parameter values and the value separated from it by a *lag* ℓ,

$$\mathrm{ACF} = \frac{1}{N} \sum_{i=1}^{N} \frac{x_i x_{i+\ell}}{\sigma^2} \tag{6}$$

where N is the number of points generated, $x_i = \theta_i - \langle \theta \rangle$ is the ith value of the parameter (e.g. θ) minus the mean from the whole sequence, and $\sigma^2 = \langle (\theta - \langle \theta \rangle)^2 \rangle$ is the parameter's variance.

By construction ACF $= 1$ for $\ell = 0$. If all of the points in the sequence were independent, the ACF would be zero for $\ell \geq 1$. The points generated by MCMC are not independent, with the ACF falling off as a function of the lag. A faster decrease of the ACF corresponds to better convergence of the generated series.

Exercises:

1(a) Run the program and examine the plots. The first one (Fig. 1 below) shows the data values as ticks on the x axis together with the fitted curve (evaluated with MAP estimators). The uncertainties on the parameters correspond to the covariance $V_{ij} = \mathrm{cov}[\lambda_i, \lambda_j]$ that `iminuit` finds by approximating the posterior as a multivariate Gaussian near its maximum (similar to finding the covariance matrix of the MLEs).

1(b) Look at the output of `bayesFit.py` from the MCMC algorithm: The plots include:

1. Trace plots of each of the parameters (Fig. 2). In some problems it can be useful to discard a subset of the points (called "burn-in") if the starting point $\boldsymbol{\lambda}_0$ is too far from the main concentration of the target density's probability; this is indicated in the trace plots with a vertical yellow bar.

2. Marginal distributions of the individual parameters (Fig. 3). The histograms are normalized to unit area and the MAP estimates are indicated with the vertical bars.

3. The autocorrelation function for the parameters (Fig. 4).

Change the data sample size from $n = 400$ to 200 and 1000 and note the changes in the results.

Using again $n = 400$, fix the parameters μ and σ (by changing the corresponding elements in the array `parfix` from `False` to `True`) and note the changes in the results. When finished, go back to having all four parameters free.

Change the number of MCMC iterations from 10 000 to 100 000 and note the change in the results, particularly in the structures you see in the trace plots. (This probably takes some time to run; for the rest of the exercises it is probably best to change back to 10 000 iterations.

1(c) Change the prior pdfs for ξ and σ to be $\pi(\xi) \propto 1/\xi$ and $\pi(\sigma) \propto 1/\sigma$ and note the change in the results. When finished, go back to constant priors.

1(d) Suppose that one has an independent estimate u of the parameter ξ in addition to the $n = 400$ values of x. Treat u as Gaussian distributed with a mean ξ and standard deviation $\sigma_u = 0.5$ and take the observed value $u = 5$. Find the log-likelihood function that includes both the primary measurements $(x_1, \ldots, x_n)$ and the auxiliary measurement u and modify the fitting program accordingly. Investigate how the results are affected by including u.

1(e) Using the functions `cc_interval` and `HPD_interval` provided in `bayesFit.py`, compute the central credible interval and HPD (highest probability density) interval for the parameter of interest θ using a credibility level of 68.3%. Compare these to the intervals one obtains from a point estimate (the MAP estimate, posterior median or posterior mean) plus or minus one standard deviation. For the standard deviation, try using both the sample standard deviation from the MCMC values and the standard deviation found by `iminuit`, which is based on a Gaussian approximation to the peak of the posterior. Find the estimates and intervales both with and without the auxiliary measurement of ξ as in (d) above and note how this effects the results.

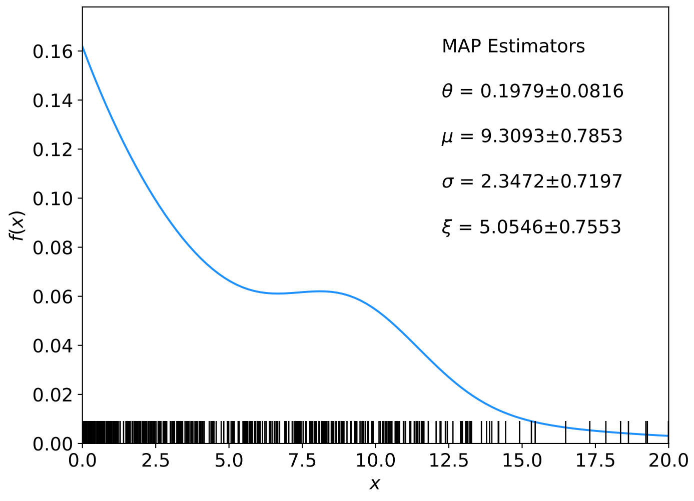

Figure 1: The data values $(x_1, \ldots, x_n)$ as tick marks with the fitted curve with parameter values from the maximum of the posterior pdf.

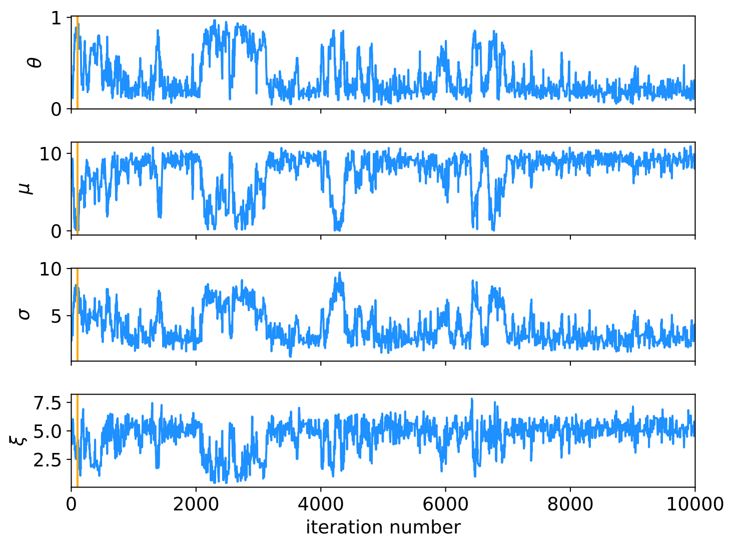

Figure 2: Trace plots of the parameter values sampled by MCMC. The end of the burn-in period is indicated with the vertical bar.

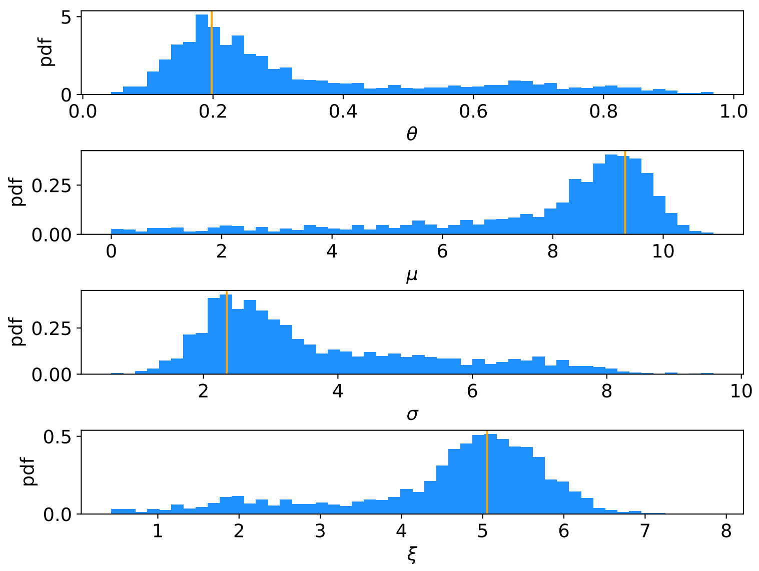

Figure 3: Marginal distributions of the parameters from MCMC. The MAP estimates are indicated with vertical bars.

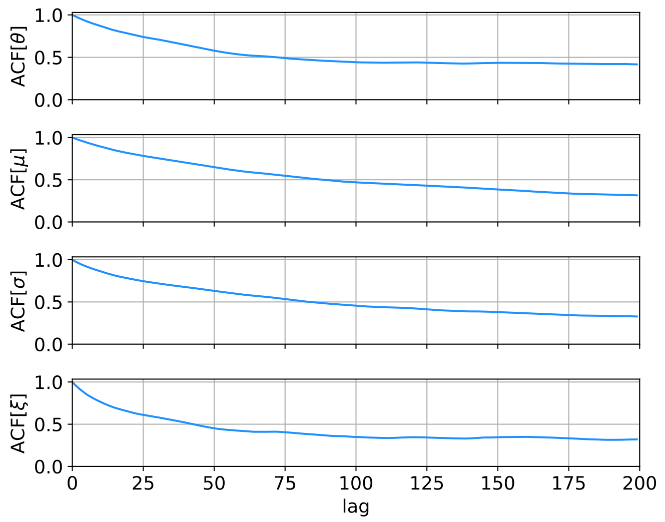

Figure 4: The autocorrelation function (ACF) for the parameters θ, μ, σ and ξ as a function of the lag. The correlation length is the lag at which the ACF falls to 0.5.