Mini-Project: Bayesian linear regression – 2D grid approximation

The previous example goes through the steps of a practical Bayesian linear regression analysis in quite some detail, including the prior check and posterior predictive check. In implementing this analysis on a computer, we used Monte Carlo sampling, to be discussed later, to estimate the posterior distribution.

To help further solidify our understanding of Bayesian analyses, we will go back to the 'primitive' grid approximation but this time in 2D (slope and offset as the two parameters). Along the way, we will pictorially illustrate Bayesian learning in a linear basis function model, as well as the sequential update of a posterior distribution.

Consider a single input variable x, a single target variable y and a linear model of the form $y(x; w) = w_0 + w_1x$. Because this has just two adaptive parameters, we can plot the prior and posterior distributions directly in parameter space. We generate synthetic data from the function $f(x) = a_0 + a_1x$ with parameter values $a_0 = -0.3$ and $a_1 = 0.5$ by first choosing values of x_n from the uniform distribution $U(-1, 1)$, then evaluating $f(x_n)$, and finally adding Gaussian noise with standard deviation of 0.2 to obtain the target values.

The figure below shows the results of Bayesian learning in this model as the size of the data set is increased and demonstrates the sequential nature of Bayesian learning in which the current posterior distribution forms the prior when a new data point is observed. It is worth taking time to study this figure in detail as it illustrates several important aspects of Bayesian inference.

The first row of this figure corresponds to the situation before any data points are observed and shows a plot of the prior distribution (Gaussian) in w space together with six samples of the function $y(x; w)$ in which the values of w are drawn from the prior.

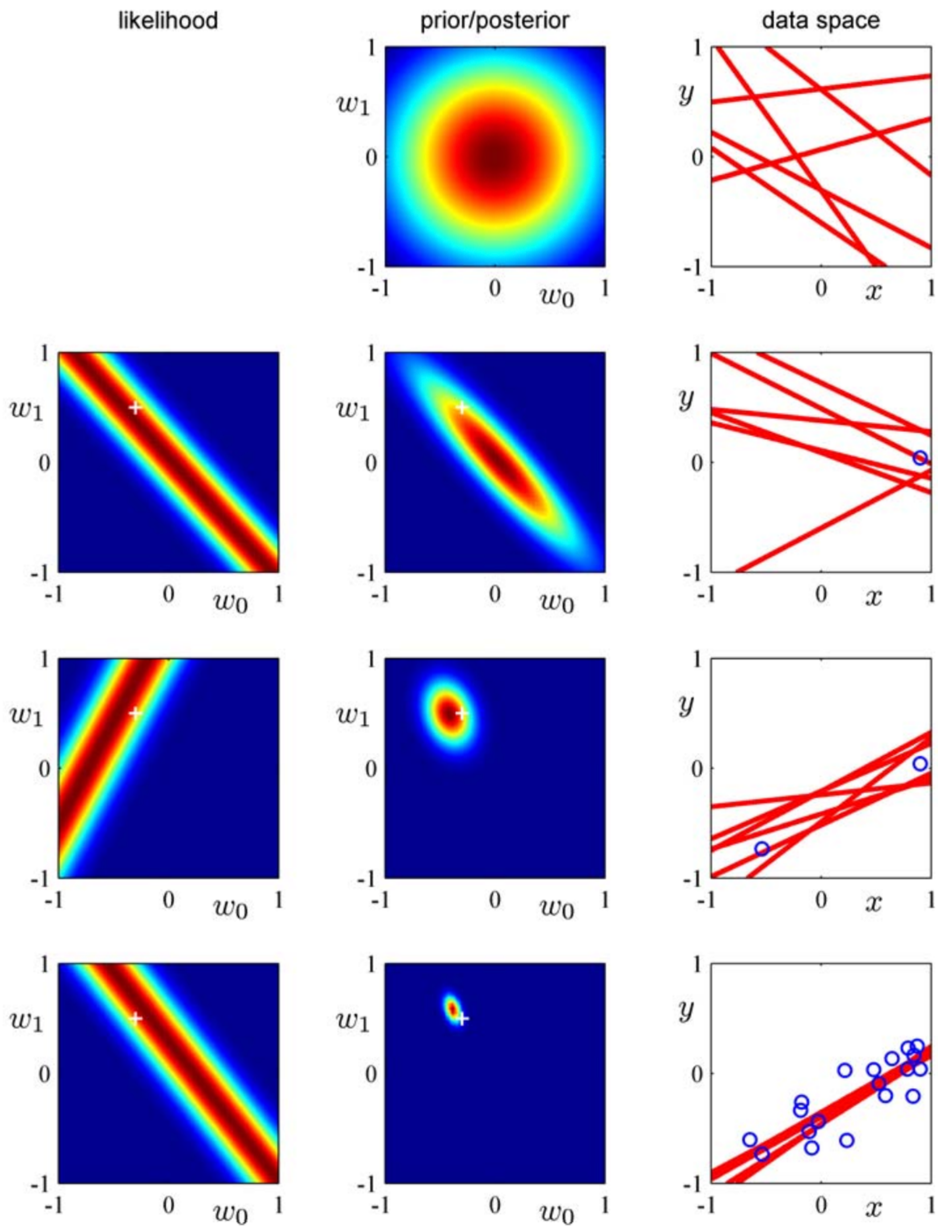
In the second row, we see the situation after observing a single data point. The location (x_1, y_1) of the data point is shown by a blue circle in the right-hand column. In the left-hand column is a plot of the likelihood function $p(y_1|x_1, w)$ for this data point as a function of w. Note that the likelihood function provides a soft constraint that the line must pass close to the data point. For comparison, the true parameter values of the offset (-0.3) and slope (0.5) used to generate the data set are shown by a white cross in the plots in the left column of the figure.

When we multiply this likelihood function by the prior from the top row, and normalize, we obtain the posterior distribution shown in the middle plot on the second row. Samples of the regression function $y(x; w)$ obtained by drawing samples of w from this posterior distribution are shown in the right-hand plot. Note that these sample lines all pass close to the data point.

The third row of this figure shows the effect of observing a second data point, again shown by a blue circle in the plot in the right-hand column. The corresponding likelihood function for this second data point alone is shown in the left plot. When we multiply this likelihood function by the posterior distribution from the second row, we obtain the posterior distribution shown in the middle plot of the third row. Note that this is exactly the same posterior distribution as would be obtained by combining the original prior with the likelihood function for the two data points.

This posterior has now been influenced by two data points, and because two points are sufficient to define a line this already gives a relatively compact posterior distribution. Samples from this posterior distribution give rise to the functions shown in red in the third column, and we see that these functions pass close to both of the data points.

The fourth row shows the effect of observing a total of 20 data points. The left-hand plot shows the likelihood function for the 20th data point alone, and the middle plot shows the resulting posterior distribution that has now absorbed information from all 20 observations. Note how the posterior is much sharper than in the third row. In the limit of an infinite number of data points, the posterior distribution would become a delta function centered on the true parameter values, shown by the white cross.



Your mini-project is to recreate 2D color maps (don't worry about the third column "data space") similar to the ones above using the 2D grid approximation for the synthetic data described two pages before.

You can use something like the `contourf` or `heatmap` function from Julia Plots.jl package (or another comparable plotting function from plotting packages of your choice) to create color maps.

https://docs.juliaplots.org/latest/gallery/gr/generated/gr-ref022/#gr_ref022

[caution] Consistent with how we represent a matrix, the first index of 2D arrays (your priors, likelihoods, and posteriors) corresponds to a row number (moving vertically) and the second index corresponds to a column number (moving horizontally). To conform with we our usual way of drawing the x-y coordinate axes, you should index your 2D arrays as $z[j,i]$, that is, swap j and i. Here, as is normally the case, the index i runs for your x values and j for your y values.

Your two parameters are the slope w_1 and the intercept w_0. Let's assume the priors for w_1 and the w_0 are both Gaussian with zero mean and standard deviation of 0.25.

For simplicity, let's assume that the standard deviation for y is fixed at the true value (0.2). Otherwise, we would have to do 3D grid approximation (w_0, w_1, and σ).

You would need nested for-loops (for an array of values for w_0 and for an array of values for w_1) to compute the prior, likelihood, and posterior distribution for each step of the Bayesian analysis.

Use (randomly chosen) single data point, two data points, and all 20 data points as in the last row of plots in the figure above. When using the all 20 data points, you may choose to nest another for-loop inside your other two for-loops (for w_0 and w_1).

Check that the MAP values of w_0 and w_1 are close to the "true" values (a_0 and a_1).