Studio 6 Bayesian linear regression
18.05, Spring 2025

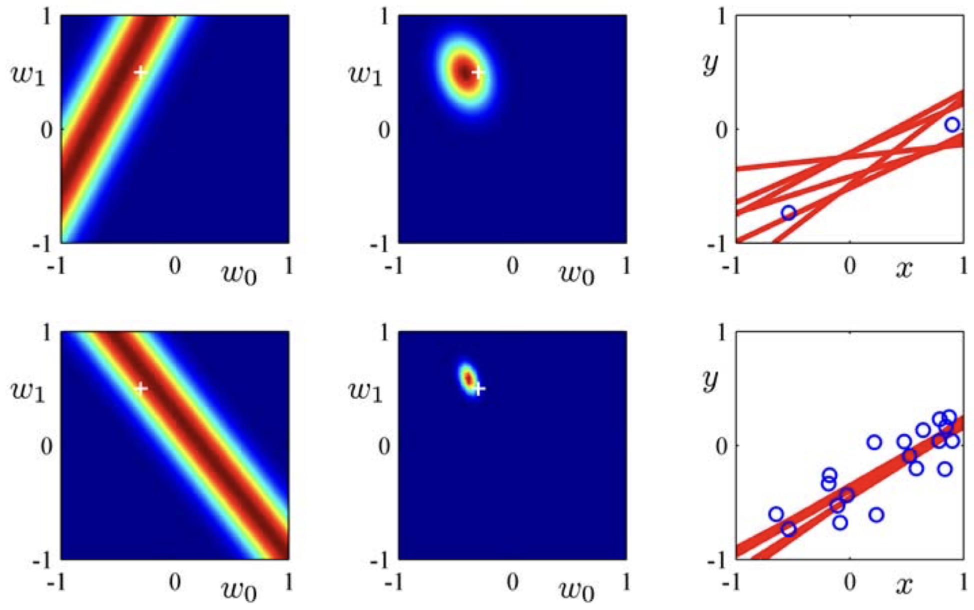

Image: Bishop, Christopher M. "Pattern recognition and machine learning." Springer (2006)

Overview of the studio

This studio explores Bayesian simple linear regression via simulation.

R introduced in this studio

The R needed is introduced in studio6-samplecode.r. We will use heatmaps to draw 2-dimensional functions and

functions: `lm(), image()`

Download the zip file

- You should have downloaded the studio6 zip file from our Canvas site.
- Unzip it in your 18.05 studio folder.
- You should see the following R files
  studio6.r, studio6-samplecode.r, studio6-test.r

  and the following other files

  studio6-instructions (this file), studio6-test-answers.html, companies.csv

Prepping R Studio

- In R studio, open studio6-samplecode.r and studio6.r
- Using the Session menu, set the working directory to source file location. (This is a good habit to develop!)
- Answer the questions in the detailed instructions just below. Your answers should be put in studio6.r
- Solution code will be posted on Saturday at 4 am

Detailed instructions for the studio

• Go through **studio6-samplecode.r** as a tutorial.

Pay special attention to parts about drawing a heat map and fitting and plotting linear regression.

Problem 1. This problem examines estimation of linear functions using MLE and Bayesian approaches to simple linear regression.

Setup

- As a business analyst in a leading marketing company you need to model the relationship between *advertising spending* and *revenue*.
- You work with the (simplistic) assumption that there is a linear relationship between the variables, so that an increase in ad spending corresponds to a proportional increase in expected revenue.
- Mathematically, if x is ad spending and y is revenue, we assume that $E[y \,|\, x] = ax + b$ for some values of a and b.
- So we have a continuous space of hypotheses, which are the real values of a and b that we wish to estimate.
- We will further assume that y given x is normally distributed about its mean with a known variance σ^2. That is $\boxed{y \,|\, x \sim \mathcal{N}(ax+b, \sigma^2).}$
- Below, you will use fixed x values to generate random y values according to the above model for some a and b, and then use the data (x and y values) to estimate the true a and b used.
- **Important:** Ensure your working directory contains all studio files, including `companies.csv`. When you source `studio6.r`, the 100 ad spend values in the csv will be stored in an array named `x` that you should use.

Problem 1a. Here you will finish the code for the function

`studio6_problem_1a(x, a, b, sigma)`

This function should receive a data array, containing the advertisement spending (in millions of dollars) recorded from various companies. Use these values as the x values and generate an array of data for the y values according to the boxed model above for the given a, b, and σ^2. The y values represent the revenue of each company.

Your function should

- Receive an array of x values.
- Generate a sample of y values using the normal model above.
- If `draw_scatter_plot` is set to `TRUE`, create a scatter plot of all (x, y) value pairs (otherwise, skip the plot).
- **Return** the y values using the 'return' statement.

Problem 1b. Here you will finish the code for the function
`studio6_problem_1b()`

Run your code from problem 1a using the supplied data set with $a = 0.5$, $b = -0.3$ and various values of `sigma`. Use the cat statements to briefly explain what changes in the plot when `sigma` increases and all other parameters stay constant.

Your solution should just print out the description. Do not include the calls to `studio6_problem_1a()` in you submitted code.

Problem 1c. Here you will finish the code for the function
`studio6_problem_1c(x, y, sigma)`

This function will compute and visualize the likelihoods of a and b, for a **single pair** of (x, y) values. As before, `sigma` is given.

We will assume that a and b are between -1 and 1 and discretize the possible a and b values in the first lines of the code (supplied by us).

Your code should create a likelihood table (as an R matrix) over all possible values of a and b in the discretized domain. That is, for every a and b, calculate the density of $\mathcal{N}(ax+b, \sigma^2)$ at y.

Once the likelihood table is ready, if `draw_likelihood_table = TRUE`, visualize it as a heat map.

Before terminating the function should *return* the likelihood table.

Due to space constraints the sample test answers do not contain the values of the returned table.

Problem 1d. Here you will finish the code for the function
`studio6_problem_1d(x, a, b, sigma)`

In this function you will generate y values using a 'true' a and b and then find the maximum likelihood estimate (MLE) for a and b. The function receive the x values of a data set and generates y values **using the function you wrote for problem1a**, with the 'true' parameters a, b and σ.

We do not expect the MLE to recover the true coefficients a and b exactly, but rather to estimate them well.

For this normal model, the MLE gives the line $y = ax+b$ that minimizes the sum of squared errors (see Problem 5 on Pset 6). Use the `lm()` function, as shown in the sample code to find the maximum likelihood estimator for a and b, and print its coefficients.

Calling your function from Problem 1a should produce a scatter plot. Add two lines to the plot: the true line $y = ax + b$ and the estimated line from the MLE. Use different colors for the two lines and add a legend.

Problem 1e. Here you will finish the code for the function
`studio6_problem_1e()`

Run your code from problem 1d using the supplied data set with $a = 0.5$, $b = -0.3$ and various values of `sigma`. Use the cat statements to briefly explain what changes in the plot when `sigma` increases from 0 to 0.5, and all other parameters stay constant.

Next, keep `sigma = 0.5` and gradually reduce the size of the dataset. Use the cat statements to briefly explain what changes in the plot when the size of the data changes.

The following bit of code may prove useful for the second part. If `x` is an array of size n, and $k \leq n$, then `x[1:k]` will be an array with the first k elements of `x`.

Your solution should just print out the descriptions. Do not include the calls to `studio6_problem_1d()` in your submitted code.

Problem 1f. Here you will finish the code for the function
`studio6_problem_1f()`

For the last part we will assume that a and b have a prior distribution of a standard bivariate normal distribution.

The supplied `show_evolving_posteriors()` function takes as input an array of x values and `sigma`, sets $a = 0.5, b = -0.3$, uses your solution of problem 1a to generate y values, and plots a heatmap of the *prior* distribution of a and b.

The function then iteratively preforms Bayesian updating with a single pair of (x, y) values. At each iteration, the function uses your solution to problem 1c to compute the likelihood table and then uses Bayes' law to compute the posterior. Finally, at each iteration, it plots a heat map of the posterior distribution over a and b. Scroll through the plots to animate how the posteriors evolves!

You can see an example of the output of `show_evolving_posteriors()` in the test-answers file. Since the function will produce as many plots as there are elements in the input x, calling it on large arrays can take some time. Start with `x[1:5]`.

Important: Before calling the `show_evolving_posteriors()` function, You must finish both problem 1a and 1c.

In this problem, similar to problem 1e, you will explore how the posteriors evolve when `sigma` is changed, or when the size of available data changes.

Call the function several time while varying `sigma`. Use the first cat statement to briefly explain what changes in the plots when `sigma` increases. Now, fix `sigma` and instead play with the size of the dataset. Use the second cat statement to briefly explain what changes in the plots when the size of the data decreases.

Your solution should just print out the descriptions. Do not include the calls to `show_evolving_posteriors()`

in your submitted code.

If you wish to further explore the evolution of the posteriors, the `show_evolving_posteriors()` can also take as argument alternative values for a, b, and the standard deviations of the prior. Change these values to see how it affects the evolution. You can also set `draw_likelihood_table` to visualize the interleaved likelihoods.

Testing your code

For each problem, we ran the problem function with certain parameters. You can see the function call and the output in studio6-test-answers.html. If you call the same function with the same parameters, you should get the same results as in studio6-test-answers.html – if there is randomness involved the answers should be close but not identical.

For your convenience, the file studio6-test.r contains all the function calls used to make studio6-test-answers.html.

Before uploading your code

1. Make sure all your code is in studio6.r. Also make sure it is all inside the functions for the problems.
2. Clean the environment and plots window.
3. Source the file.
4. Call each of the problem functions with the same parameters as the test file studio6-test-answers.html.
5. Make sure it runs without error and outputs just the answers asked for in the questions.
6. Compare the output to the answers given in studio6-test-answers.html.

Upload your code

Upload your code to Gradescope.

Leave the file name as studio6.r.

You can upload more than once. We will grade the last file you upload.

Due date

Due date: The goal is to upload your work by the end of class. If you need extra time, you can upload your work any time before 6 PM ET on the day of the studio (Friday).

Solutions uploaded: Solution code will be posted on Canvas at 4 AM the day after the studio.