Bayesian Statistics

Group exercise 2

Read before you start

For these group exercises, you are expected to deliver the following:

- R & JAGS code for each sub-exercise.
- A report (PDF) containing the answers to the questions, including plots/figures where necessary, and referring to the code you wrote. Make sure figures etc. are properly visible and have informative labels.

Apart from the book by Kruschke, you may also need the JAGS manual[1] to look up the details of the distributions you will need to use.

1 Linear regression

In this exercise you'll implement and experiment with a few linear regression models, as well as compare different variants.

1.1 Theft in Manhattan

Let's take look at New York City crime data from 1966 to 1967. The outcome variable y_i is the increase or decrease in the number of thefts in each i of 23 Manhattan police precincts, compared to a baseline period earlier. The predictor x_i is the percentage increase or decrease in the number of police officers in a precinct. Together with some hyperparameter settings, this forms the following model:

$$\begin{aligned} \tau &\sim \text{Gamma}(0.01, 0.01) & \\ w_j &\sim \text{Gaussian}(0, 1.0) & j = 0, 1 \\ \mu_i | w_0, w_1, x_i &= w_0 + w_1 x_i & i = 1 \dots n \\ y_i | \mu_i, \tau &\sim \text{Gaussian}(\mu_i, \tau) & i = 1 \dots n \end{aligned}$$

Note: we use the JAGS convention to specify the variance of the Gaussian distribution, with $\tau = 1/\sigma^2$. This means that large values of τ generate data points very close to μ_i, while small values generate data points that may deviate further. This way, our notation here should match the notation in JAGS.

1. (20pts) Implement this linear regression model in JAGS, using the template file `crimemanhattan_template.R`.
2. (10pts) The data (the predictors x_i and the to-be-predicted values y_i) are available in the template file. Use this data to learn the posterior distributions over both weights. Plot histograms of both. Does adding more police staff increase or decrease the amount of thefts?
3. (20pts) Make a new plot in which you show the data points as individual points. On top of that, use the R `abline` function to plot the line that w_0 and w_1 define. Use the expectation of the approximated posterior for these weights, i.e. the mean of the sampled values.

[1] http://www.stat.yale.edu/ jtc5/238/materials/jags-2.1.0-user-manual.pdf

4. (30pts) Classical statistical inference would stop here. However, in Bayesian inference we don't actually have *just* the expectation of the posterior, but the whole distribution! This allows us to characterize the *uncertainty* in our estimates of this linear relationship. To visualize this copy the plot you made previously, and superimpose 500 *sampled* lines from the posterior. To do so, you need to randomly select 500 samples from the ones that you have collected using JAGS, then from each sample obtain w_0 and w_1, and then plot the line using `abline` again. To improve visibility, pass `col=rgb(0.8, 0.2, 0.2, max = 1.0, alpha = 0.1)` to the `abline` function, which will make the sample lines transparent[2]. Multiple transparent lines will create opaque parts where the estimate is most certain (as all samples go through that area). Using this visualization, where is the model most certain about its predictions? And where is it most uncertain? Why is it more uncertain there, compared to the other parts of the plot?

1.2 One or two lines? A regression mixture model

Consider again a seemingly simple linear regression task with only one predictor x_i and one prediction y_i for every observation (= data point) i.

1. (10pts) Apply your model from the previous exercise to the data that you the template file `twolines_template.R` loads for you, and learn the posterior distributions of w_0 and w_1. Plot the data points, the expectation of the predicted line (using the expectations of w_0 and w_1), and 500 samples from the posterior. Where in the figure does the fitted line fit the data best, i.e. where is the uncertainty lowest?

2. (20pts) The previous question asked you to blindly apply a simple linear model to data. But as you can see now that you have plotted the data and the estimated line, this data actually appears to come from *two* lines instead of one (that is, each data point belongs to one of the two lines). Extend the generative model to describe that each data point (x_i, y_i) is described by one of two lines (each with its own intercept and slope). You will need an additional variable z_i that indicates from which line data point i comes. You may assume that each line is equally likely in your prior on z_i[3].

3. (20pts) Implement the extended model in JAGS and run it using the data. Plot the data points like you did before, and superimpose 500 samples from this new model. Note that a each sample now contains *two* lines!

4. (10pts) Plot, in the same figure, the *expectation* of the two estimated lines. You'll see a surprising and probably undesired effect. Can you think of a reason the expectations don't nicely match the samples you showed in the previous exercise[4]?

5. (30pts) Given your plots, it (hopefully) seems that the model with two regression lines explains the data much better than the model with one line. But we want to make sure! Implement a model comparison model in JAGS, using an additional categorical variable m. When $m = 1$, the model should explain the data using just one line, while when $m = 2$, the two regression lines should be used. The two models may use the same precision parameter τ. To make sure you implemented everything correctly, plot again the data, and then 250 samples for $m = 1$ and 250 samples for $m = 2$. Note that the latter samples again contain two lines. Use different colors for samples from each of the two models.

6. (20pts) Run this model on the data, and compute (and show) the posterior *model* probabilities $p(m_1|\mathbf{x}, \mathbf{y})$ and $p(m_2|\mathbf{x}, \mathbf{y})$, as well as the Bayes factor. Look up the Bayes factor interpretation table on Wikipedia. Do the outcomes match your intuition? According to this table, can you conclude that the model with 2 lines is better?

[2] You may of course change the coloring to your own preference, but keep the transparency.

[3] Hint: you have seen similar mixture models in the Game of Thrones exercise as well as the Gaussian mixture model of week 04.

[4] This question is a good illustration of why it is important to look at the posterior distribution of your analysis, rather than only at summary measures like the expectation.

1.3 Game of Regression

Recall that in an earlier exercise, you learned the probability of survival of the characters in Game of Thrones. Instead of predicting the probability of survival, we can now try to predict the age (at death) of our favorite characters. We'll do so using Bayesian linear regression, where y_i is the age that character i dies at, and:

- $x_{i,1}$ is the number of pages in the books on which i is mentioned,
- $x_{i,2}$ is the number of weddings i attends,
- $x_{i,3}$ is the size of household i is from,
- $x_{i,4}$ is the length of i in centimeters and finally
- $x_{i,5}$ is the number of times i has said 'winter is coming'.

We will use the same model as previously, with one change: we now have multiple predictors. That means the model becomes

$$\begin{aligned} \tau &\sim \text{Gamma}(0.01, 0.01) \\ w_j &\sim \text{Gaussian}(0, 1.0) && j = 0, 1 \\ \mu_i|\mathbf{w} = w_0 + \sum_{j=1}^{p} w_j x_{ij} &= \mathbf{w}^T \mathbf{x}_i && i = 1 \dots n, j = 1 \dots p \\ y_i|\mu_i, \tau &\sim \text{Gaussian}(\mu_i, \tau) && i = 1 \dots n \ . \end{aligned}$$

Data is available for this exercise in a separate file for $\mathbf{x}$ and y. Load the data using the code in the template `gameofregression_template.R`. Note that $\mathbf{x}$ is a *matrix* where each *row* contains the values for the $p = 5$ predictors for one character. There are data available for $n = 100$ characters (so essentially we have 100 observations to base our estimates of $\mathbf{w}$ on). The ages y are stored in a vector, with each element being the age at which the corresponding character died.

1. (20pts) Implement the linear regression model in JAGS and learn the posterior distributions for each element of $\mathbf{w}$, as well as for w_0[5]. Try using the linear algebra notation $\mathbf{w}^T\mathbf{x}_i$ (i.e. the inner product) instead of summing over each of the predictors manually. Plot the histograms of the approximated posterior distribution of each of these weights. One of these predictors hardly seems to influence the age of death. Which one is that?

2. (20pts) Imagine: In the first episode of season 8, George R.R. Martin introduces a new character. The character has the following predictor vector $\mathbf{x}_{n+1} = (1, 0, 5, 184, 20)$. What is the predicted age of death? Had the character been a giant instead, 380cm tall, what would the predicted age of death be?

2 Comparing models for small images data using logistic regression

In this exercise you will examine model comparison for a small, but conceptually realistic case, where it is possible to enumerate all possible data sets. By computing the evidence for each data set, you will see that some models fit more possible data sets than other models would, but that there is not a single model that always 'wins'. For this exercise, there will be a bit more programming in R, but we will not use JAGS.

Consider a very simple agent (for example, a robot) that has a camera with a resolution of only 9 pixels. These pixels are binary, i.e. they can only be on or off. They are arranged in a square grid, so that each pixel has the following coordinates (see also Fig. 2(a)):

$$\begin{aligned} \mathbf{x}^{(1)} &= (-1, +1), & \mathbf{x}^{(2)} &= (0, +1), & \mathbf{x}^{(3)} &= (+1, +1), \\ \mathbf{x}^{(4)} &= (-1, 0), & \mathbf{x}^{(5)} &= (0, 0), & \mathbf{x}^{(6)} &= (+1, 0), \\ \mathbf{x}^{(7)} &= (-1, -1), & \mathbf{x}^{(8)} &= (0, -1), & \mathbf{x}^{(9)} &= (+1, -1) \ . \end{aligned}$$

[5] Tip: it will be easier to implement w_0 as a separate parameter than as an element of the vector $\mathbf{w}$.

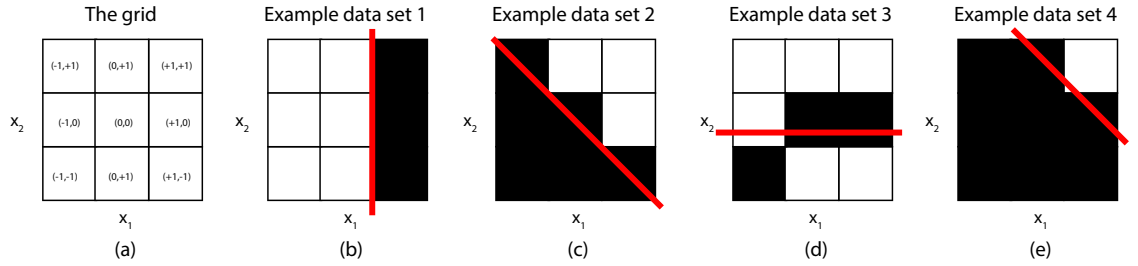

Figure 1: (a) The grid of pixels and their coordinates. (b–d) Examples of data sets, each with different pixels being on or off. Note that it is an arbitrary choice whether we visualize on and off with black or white pixels; it doesn't matter as long as we are consistent. Thick red lines indicate possible decision boundaries.

Each pixel $\mathbf{x}^{(i)}$ is a vector of two coordinates, $x_1^{(i)}$ and $x_2^{(i)}$. The superscript (i) indicates the number of the pixel. Whether a pixel is on (value $+1$) or off (value -1), is indicated by another vector containing the labels $\{y^{(i)}\}_{i=1}^{9}$, $y^{(i)} \in \{-1, 1\}$[6]. A single data set D is a combination of the pixels (which are the same for all data sets, they are just these 9 pixels) and the vector of labels, so we have $D = (\mathbf{x}, \mathbf{y})$. A few example data sets are shown in Fig. 2(b–d).

The task for the robot is to determine a *straight line* (known as a *decision boundary*) that separates the black from the white pixels. It does so by learning the parameters $\mathbf{w}$ that describe this line. Today however, we are not interested in the values of these weights, but rather in which *model* the robot uses to learn these. We'll compare four models, each with the same prior, but with different likelihood functions. Note that these are not the familiar beta, Gaussian or Bernoulli distributions, but don't be intimidated – they are just functions describing the probability of data, given parameters and a choice for a model:

$$p(D|\mathbf{w}, m_0) = \frac{1}{512} \tag{1}$$

$$p(D|\mathbf{w}, m_1) = \prod_{i=1}^{9} \frac{1}{1 + e^{-y^{(i)} w_1 x_1^{(i)}}} \tag{2}$$

$$p(D|\mathbf{w}, m_2) = \prod_{i=1}^{9} \frac{1}{1 + e^{-y^{(i)} \left(w_1 x_1^{(i)} + w_2 x_2^{(i)} \right)}} \tag{3}$$

$$p(D|\mathbf{w}, m_3) = \prod_{i=1}^{9} \frac{1}{1 + e^{-y^{(i)} \left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} \right)}} \quad . \tag{4}$$

As you can see, the models become increasingly complex:

1. The first model generates data uniformly and doesn't care about the pixel information. It's weight vector $\mathbf{w}$ is empty; there are no parameters to learn. This means that all 512 possible 3-by-3 'images' are equally likely.

2. The second model takes into account only the first coordinate (the horizontal direction) for each of the nine pixels. The strength of this coordinate on the probability of the data is modulated by a weight, w_1. In this model, decision boundaries are always vertical lines.

3. The third model takes both coordinates into account, modulated by a weight for each, w_1 and w_2. Lines can now have any orientation.

4. The fourth model also includes an *offset* weight w_0. Now the decision boundaries don't have to cross the (0,0) pixel anymore.

The latter three models are examples of *logistic regression*: the binary pixel status $y^{(i)}$ (on or off) is predicted from the pixel coordinate $\mathbf{x}^{(i)}$.

Figures 2 (b–d) show some examples of different data sets. In Fig. 2(b), you see that a pixel being on or off only depends on coordinate x_1: pixels to the right of the grid are on, all other

[6]Note the slightly different convention of -1 and +1 here, compared to 0 and 1.

pixels are off. The data set in Fig. 2(c) shows a dependency on both coordinates (i.e. the decision boundary would be a diagonal line). The data set in Fig. 2(d) is not easy to separate into black and white pixels using a straight line. None of the models will fit perfectly here. Finally, data set Fig. 2(e) is another example of a diagonal decision boundary. However, this boundary should not go through the middle of the pixel grid (i.e. not through the (0,0) pixel). That means we need an *offset* that shifts this line up or down. Examples of decision boundaries are shown with thick red lines.

Each model uses the same prior distribution on the weights (of course, only for the weights that the model actually uses):

$$p(w_j|m_i) = \text{Gaussian}(0, 100) \quad \forall_{i,j} \ ,$$

i.e. a Gaussian distribution with mean $\mu = 0$ and variance $\sigma = 10$ (note: $\sigma = 10 \leftrightarrow \sigma^2 = 100$, in R you need to supply σ to `rnorm`).

For any given data set D, one might learn the posterior $p(\mathbf{w}|D)$, i.e. which weights (and hence, which decision boundary) best describes this data? Here however, we are interested in which *model* works best — regardless of its parameters! You will do this by computing the (approximate) *evidence* for each of the 512 data sets, using Monte Carlo.

1. (10pts) Step one is to enumerate and store all the 512 data sets. This is already done for you in the template file `pixelgrid.R`. Your task is to compute the model evidences for each model and each data set. Start with computing the evidence $p(D|m_0)$, which you can do exactly using Equation (1) and the fact that there are no weights in m_0.

1. (30pts) Computing the evidence for the other models (m_1, m_2, m_3) cannot be done analytically. But we can do this approximately using Monte Carlo, i.e. by drawing *samples*. Page 275 (top equations) describes how to do this. In short:

   (a) Draw sample weights $\mathbf{w}$ from the prior, i.e. $\mathbf{w}^{(s)} \sim \text{Gaussian}(\mu, \sigma^2)$. Make sure you have the number of weights correct, as the different models have different numbers of weights.

   (b) Using those weights, compute $p(D|\mathbf{w}^{(s)}, m_i)$.

   Repeat this process S times, so you collect S sampled values for the likelihood, given the weights you have taken from the prior. Then

$$p(D|m_i) \approx \frac{1}{S} \sum_{s=1}^{S} p(D|\mathbf{w}^{(s)}, m_i) \tag{5}$$

   approximates the evidence for model m_i. Compute this approximation for every data set, and for each of the models m_1, m_2, m_3, using $S = 10\,000$ samples. Running the computation may take a few minutes, since you are computing the likelihood $3 \times 512 \times 10\,000$ times!

3. (20pts) Now that you have computed the evidence for all data sets and all models, use the available plotting function to visualize the result. Sort the output by m_3. Two data sets are *much* better explained by m_3 than by any of the other models. Which data sets are these? Why can they only be reasonably explained by m_3? What is the Bayes factor between m_0 and m_3 for this data set?

4. (20pts) You can argue that m_3 is a more complex model than m_0. After all, m_0 doesn't have any free parameters, while m_3 has w_0, w_1 and w_2, and hence, much more freedom to fit to complex structure in the data. However, additional complexity comes at a cost (see the slides on Occam's Razor). For how many of the data sets is m_0 actually favored over m_3? Compute the same for all 4×4 model comparisons. Which model seems to best describe the majority of possible data sets?

5. (10pts) Despite the clear dominance of that model, why might the (designers of the) robot still prefer one of the other models if the robot was to, say, use its model to navigate through a forest containing only very tall and straight trees?