**Coding up a Bayesian computer participant for the Posner task**

Work in pairs/small groups. Jill, Saad and Tim will help you!

**Version 1**

Open up posner\_learner\_1.m and run it.

The model implemented in this script ‘knows’ that the probability of the target appearing to the left (pL) is either 0.8 or 0.2. It just tries to infer which state is in force on each trial.

First, take a look at Figure 1. Use the script to work out what has been plotted.

Thinking points:

* Why is the model’s estimate of p(Left) determined based on the prior and not the posterior on each trial?
  + HINT: think about what aspect of behaviour I am trying to predict with the model)
* Given that the model knows that p(Left) is only either 0.8 or 0.2 (not, for example, 0.731), why is the model’s estimate of p(Left) not constrained to these values?
  + HINT: Think about how I obtained the estimate for p(Left) on line 66 of the script.

Now take a look at Figure 2. This shows the prior probability of each possible value for p(Left) on three trials (trials 36, 37 and 38), and the model’s estimate of p(Left).

* Note that the model’s estimate of p(Left) (red line) is in between the two possible values, 0.8 and 0.2.
* If I wanted to have the model only pick ‘valid’ values for the estimate of p(Left) - i.e. 0.8 or 0.2 – how would I read out the estimate of p(Left) from these probability distributions?

Try changing the code to output Figure 2 for other series of trials where something interesting may be going on.

* For example, if you look at Figure 1, on trial 188, the model seems to have a ‘moment of doubt’ about the value of p(Left).
* Take a look at the data points (black dots). What caused the model to revise its beliefs?
* Plot the trials in question as in Figure 2.

Finally, what happens if we reduce the difference in probability between the expected and unexpected sides (i.e., reduce cue validity). Try changing the value of cue\_validity (line 6) to 0.55 instead of 0.8 and look at the resulting Figure 1. How is it different from when cue\_validity is 0.8?

* Why?

**Version 2**

Open up posner\_learner\_2.m and run it.

The model implemented in this script ‘knows’ that the target either appears on the left or right, but it doesn’t know that the only possible values for p(Left) are 0.8 and 0.2.

You can see the difference between this version and version 1 if you compare Figure 2 for each model.

* What Is the difference between the probability distribution over p(Left) for the two models?

Take a look at Figure 1 for this model, and compare it to Figure 1 for version1.

* What is the difference between the two models, in terms of how well the estimate of p(Left) tracks its actual value?
* Which model is a better fit to the true state of the world (the blue line)?
* Why do you think this is?

Now look at figure 3. In this plot, I have plotted the probability distribution over p(Left), as in Figure 2, for each trial, as a colourmap. Bright colours indicate probable values of p(Left) on that trial, according to the model.

* Look at how the spread of the distribution changes in each block. How does uncertainty just after a reversal compare to uncertainty a long time after a reversal?
* What, apart from a reversal, makes uncertainty increase?
  + HINT: Try plotting trials 186-190 in Figure 2.

Why is it useful to know how uncertain the model is? What features of behaviour might be predicted by knowing the model’s certainty, in addition to its best guess of p(Left)?

**Version 3**

Open up posner\_learner\_v3.m and run it.

The model implemented in this script is like version 2, with the additional complication that it does not know the probability that a reversal will occur (that p(Left) changes) on any given trial. Instead, it infers this from the data. Note, though, that the model does know something about reversals – that their occurance is given by a fixed probability, p(Jump), that determines whether p(Left) changes on the current trial or not. The model also believes that if a jump does occur, a new value for p(Left) is selected at random.

* Is this actually true?
* What other assumptions could we have used about changes in p(Left)?

Take a look at Figure 1 and compare this to Figure 1 from Versions 1 and 2.

* What do you notice about the fit of the red line to the blue line (fit of p(Left) ) in version 3, compared to version 2?

In version 3, we also have an estimate for p(Jump)

* How does this estimate compare to the true value of p(Jump) which is also plotted? Can this explain the behaviour of estimated p(Left)?

Have a look at the estimate for p(Jump) in the first few trials, up to the first reversal. What do you notice?

Now look at Figure 2, the probability distribution across model parameters on some representative trials (in this case, trials 1-50). This is a similar plot to figure 2 in the previous model version, but now we have a 2D probability distribution, as we are estimating two parameters jointly.

Use Figure 1 to identify the trial number for the first reversal, and find the corresponding plot in Figure 2.

What happens to the peak of the probability distribution over p(Left) ?

What about the distribution over p(Jump) ?

Over the first 50 trials, there seems to be a relationship between the estimate of p(Left) and p(Jump) –

* Which value of p(Left) corresponds to a low value of p(Jump)?
  + Why do you think this is?
  + What would you expect to happen to this relationship in later trials?
  + Try changing the script so figure 2 plots a later set of trials (there are 500 trials in the experiment) to test this hypothesis.

**Version 4 ??**

You could change the transition function so that the model doesn’t assume a jump to totally new values, but a smooth transition instead.