Cathy Zhuang

BIOL 450

2 February 2022

**Concept Questions**

**1. In your own words, describe reward prediction error.**

Reward prediction error occurs when the outcome contradicts the expectation. For example, one receives a reward when not expecting one (a positive prediction error), or one does not receive a reward when in fact expecting one (a negative prediction error). Rescorla and Wagner proposed that we learn only when there are prediction errors.

**2. Describe the primary advantage of the TD algorithm over the Q-Learning algorithm. Why might this be algorithm be important for relating learning to neural signals?**

In the Q-learning algorithm, one can find the expected value of a state action pair, but it does not account for temporality of when the reward arrives. Since there is often a temporal gap between when the cue arrives and when the reward is presented, the TD algorithm accounts for when the reward is presented relative to the cue. This may be important for relating learning to neural signals because when one receives a reward at a specific time step, they may be more inclined to expect a reward when that timestep occurs again, and we need to account for this in the algorithm. In other words, we need the cues themselves to take on a predictive value.

**3. Describe the behavior of a participant whose best-fitting alpha (learning rate) parameter in a Q-learning model is 1. What about someone whose best-fitting alpha is 0?**

An alpha parameter of 1 is useful when one is situated in a volatile environment. In this case, it would not make much sense for a multitude of past trials to influence the current decision if the world is always changing. Thus, an alpha = 1 means that only the last trials have influence on the participant; it is very short-term thinking.

An alpha of 0 means that if there is prediction error, it does not influence the decision. The participant will continue to make the same decision regardless of the outcome. The participant will essentially never learn.

**4. How would you determine whether or not the Q-learning model provides a good fit?**

You can use gradient descent to first find the best fit for your data. You can find a combination of alphas and betas to minimize the difference between the prediction and the actual outcome.

You can then use AIC or BIC to see how well the Q-learning model fits your data. AIC and BIC scale the likelihood and adjust the likelihood to account for aspects of the model, making it legitimate to compare models with different parameters.

**5. Agents are often more responsive to surprising events early in learning than late in learning. How might you modify a model to capture this behavior?**

If we want an event to have more “influence” over the next decision, we will increase the alpha. Thus, we could perhaps start off the model with a high alpha value and decrease the alpha value as there are more and more decisions.

For example, the impact of alpha could be scaled by the nth trial such that the impact of alpha gets progressively smaller (e.g. something like alpha1/n, where alpha decreases as n trials increase). Or, we could separate trials into compartments, such as having a certain alpha when the trial number is less than 20, and having a smaller alpha when the trial number is greater than 20.

**6. Summarize the difference between "model-based" and "model-free" RL models.**

“Model-free” RL models will simply repeat what is rewarded and follow the same path if a decision led to a reward. In contrast, “model-based” RL models also include planning into the mixture as opposed to pure reinforcement. For example, even though a decision may have led to a reward, a participant following a “model-based” RL model may choose another option if they think that there is a higher probability of getting a reward from that option.

**Coding Exercises - To complete these exercises, please download the files in the "RL Code" folder.**

**1. Modify the script QlearningBasic.m to include expected value updating and Softmax decision rule equations (with temperature parameter). Include your code.**

I have also included my code as a separate file in addition to pasting it here.

function fit = QlearningBasic(parm)

global lba uba lbt ubt

global data

global EVMatrix

% scales the parameters based on starting points provided

a1=(uba-lba)/(1+exp(-parm(1)))+lba;

t1=(ubt-lbt)/(1+exp(-parm(2)))+lbt;

% a1 is the alpha parameter from the Q-learning value function

% t1 is the temperature parameter from the softmax choice function

nTrials = length(data);

trialfit = 1;

partialfit = 0;

fit=0;

% these are the expected values for each option

window1 = 0; % the Q value for option 1 on a given trial

window2 = 0; % the Q value for option 2 on a given trial

pr1 = 0; % the probability of selecting option 1 on a given trial

pr2 = 0; % the probability of selecting option 2 on a given trial

for trialnum = 1:length(data)

stimchoice = data(trialnum,2); % the choice made on a given trial

r = data(trialnum,3); % the reward received on that trial

pr1= exp(window1/t1)/(exp(window1/t1)+exp(window2/t1)); % softmax choice function for choosing option1

pr2= exp(window2/t1)/(exp(window1/t1)+exp(window2/t1)); % fill in for choosing option 2

% inverse temp:

% pr1= exp(window1\*t1)/(exp(window1\*t1)+exp(window2\*t1)); % softmax choice function for choosing option1

% pr2= exp(window2\*t1)/(exp(window1\*t1)+exp(window2\*t1)); % fill in for choosing option 2

if stimchoice==1

trialfit(trialnum) = pr1;

window1 = window1 + a1\*(r-window1); %%%%%%%%update EV for option 1

elseif stimchoice==2

trialfit(trialnum) = pr2;

window2 = window2 + a1\*(r-window2); %%%%%%%%update EV for option 2

end

EVMatrix(trialnum,:)=[trialnum stimchoice window1 window2 pr1 pr2];

partialfit=partialfit + log(trialfit(trialnum));

fit = -1\*partialfit; % negative log liklihood

end

end

**2. Fit the Q-learning model to the included dataset (SampleData1). Interpret your output.**

My output was 0.0137, 0.1332, and 59.3731. This means that the best alpha was 0.0137, so it is a rather stable environment, and trials have relatively less weight in decision making (than, say, an alpha of 0.9). The best beta or temperature was 0.1332, meaning that the likelihood one “takes the temperature” of other options is 0.1332. The log likelihood was 59.3731.

**3. Try fitting the model with an inverse temperature parameter. What is different?**

My output was 0.0137, 7.5048, and 59.3731. The log likelihood and alpha values are the same, but the temperature parameter (beta value) is now the inverse of the previous beta value (my previous beta value was 0.1332, and 1/0.1332 = 7.5048, my current beta value). This makes sense because we inverted the temperature parameter.

**4. Create a dataset where the agent responds randomly, independent of feedback. You can create these data files using excel and save as a .csv, or can generate them using Matlab. Keep in mind that the task involves choosing between two options (option 1 and option 2). Option 1 gives a reward ($1) on 70% of trials and otherwise gives nothing ($0). Option 2 gives a reward ($1) on 30% of trials and otherwise gives nothing ($0).**

trialNumber = (1:200)';

optionChose = datasample(1:2, 200)';

rewardOutcomeDollars = zeros(200,1);

table1 = table(trialNumber,optionChose,rewardOutcomeDollars);

for i = 1:length(trialNumber)

if table1.optionChose(i) == 1

table1.rewardOutcomeDollars(i) = randsrc(1,1,[0,1;0.3,0.7]);

elseif table1.optionChose(i) == 2

table1.rewardOutcomeDollars(i) = randsrc(1,1,[0,1;0.7,0.3]);

end

end

writetable(table1,'Zhuang\_BIOL450\_assignment1dataset.csv')

type 'Zhuang\_BIOL450\_assignment1dataset.csv'