Model fitting Juan's data to the John Mikhael model

Beren Millidge

September 24, 2022

Procedure

- Juan's initial data consisted of choice and reward behaviour for 17 mice
- Task was a two-armed bandit with 0.9 and 0.1 probability of reward on each side respectively
- We fit a three-parameter value of the John Mikhael model using maximum likelihood
- This just involves computing the log-likelihood of the data sequence under the model, then fitting using nonlinear optimization algorithm
- As likelihood function could have local minima, all fitted results come from running 20 optimizations from different initial conditions, and taking the best fit.

John Mikhael Model

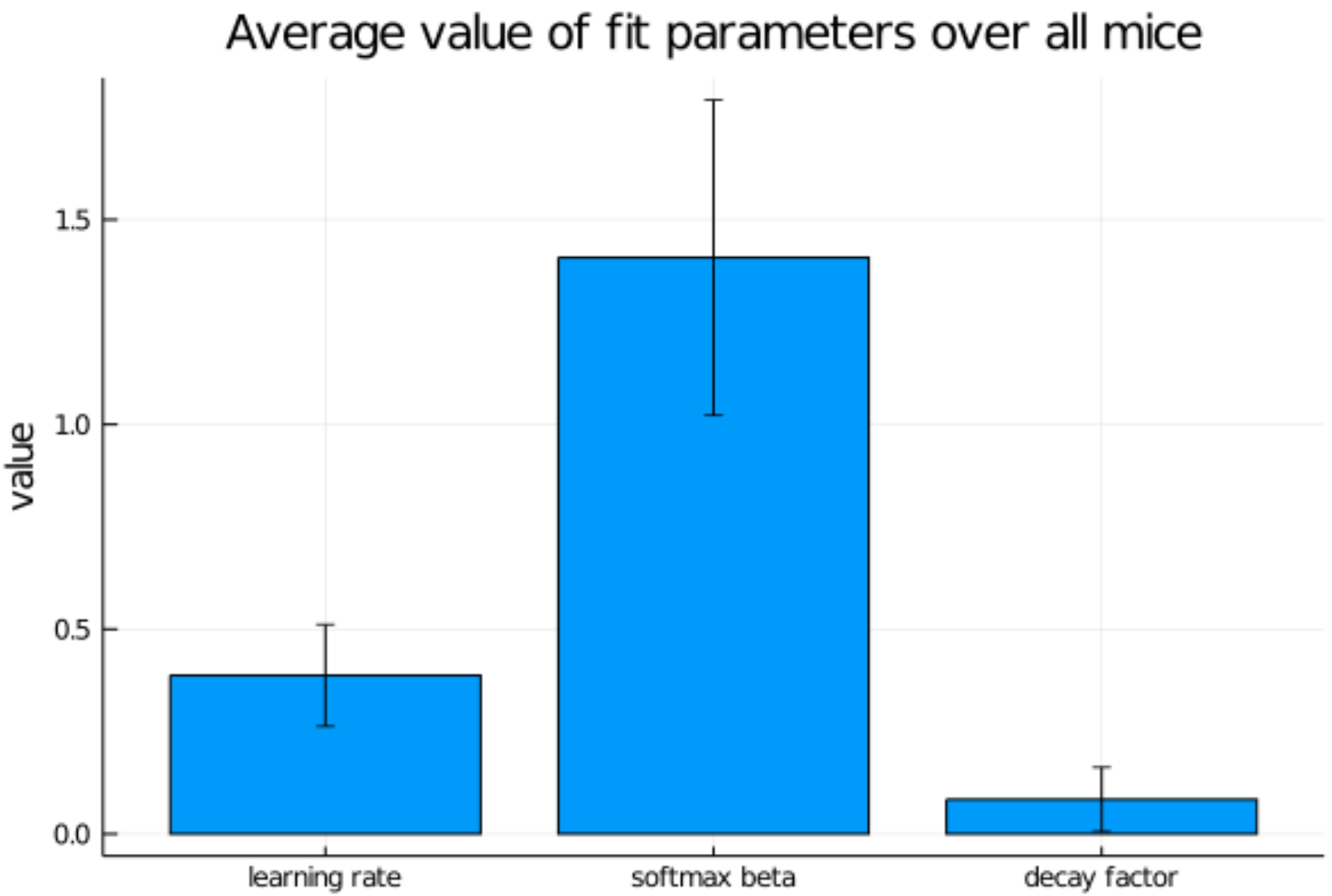
- Core equations of the John Mikhael Model
- Three parameters α (learning rate), β (softmax temperature), γ (decay factor)

$$V = G - N \tag{1}$$
$$G = G + \alpha * max(r - V, 0) - \gamma G \tag{2}$$
$$N = N + \alpha * min(V - r, 0) - \gamma N \tag{3}$$
$$p(a) = \sigma(\beta * (G - N)) \tag{4}$$

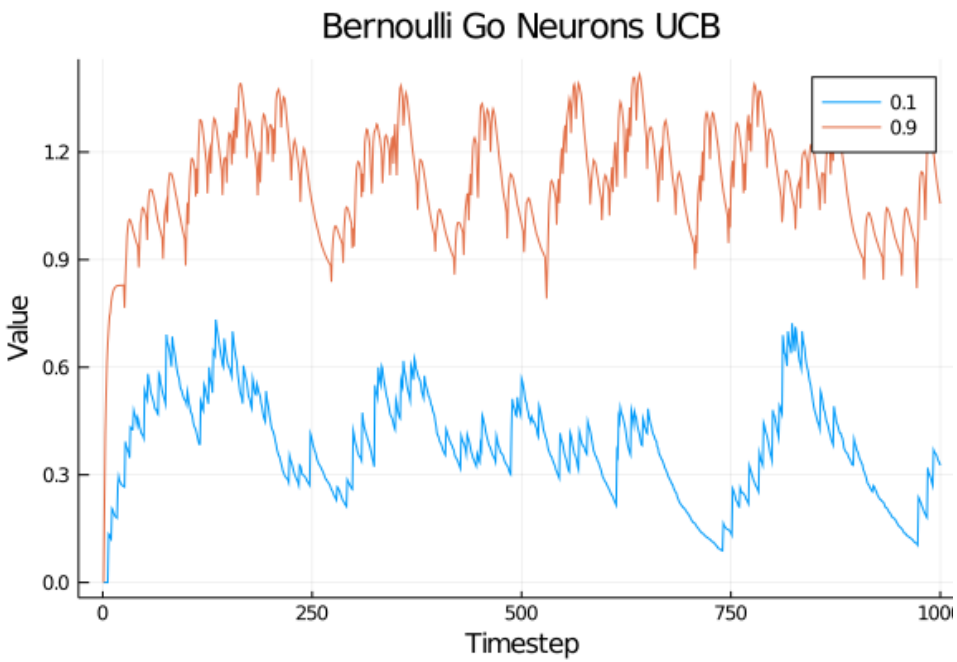
General Results from fitting

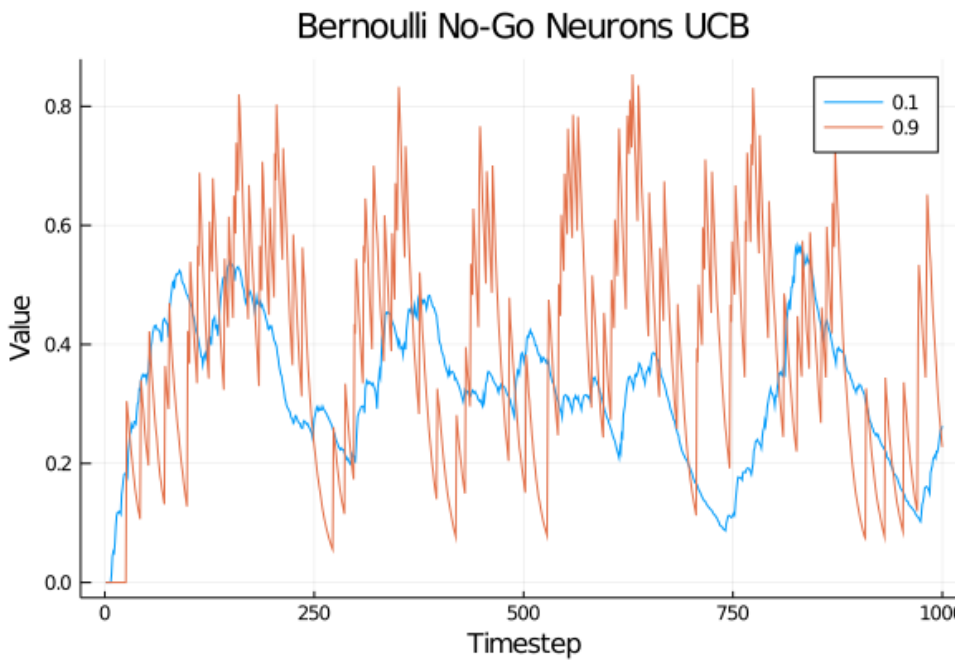
- On average, we get a fairly high learning rate (0.5ish), a fairly low softmax temperature (just about and above 1), and a very low decay factor (0 - 0.1)
- This means that the animals learn fast, don't forget or discount much, and make quite random choices even when they 'know' the right answer.
- There is quite a lot of variability *between* mice, with one or two obvious outliers.
- Fitting the three parameter model was quite stable and resulted in roughly the same parameter values for different initial conditions between 0 and 1
- I also tried fitting a five parameter value with additional parameters a and b which modulate the value of G and N in the softmax – i.e $p(a) = \sigma(\beta * (a * G - b * N))$ but this was highly unstable and gave extreme values (overfitting?)

Results

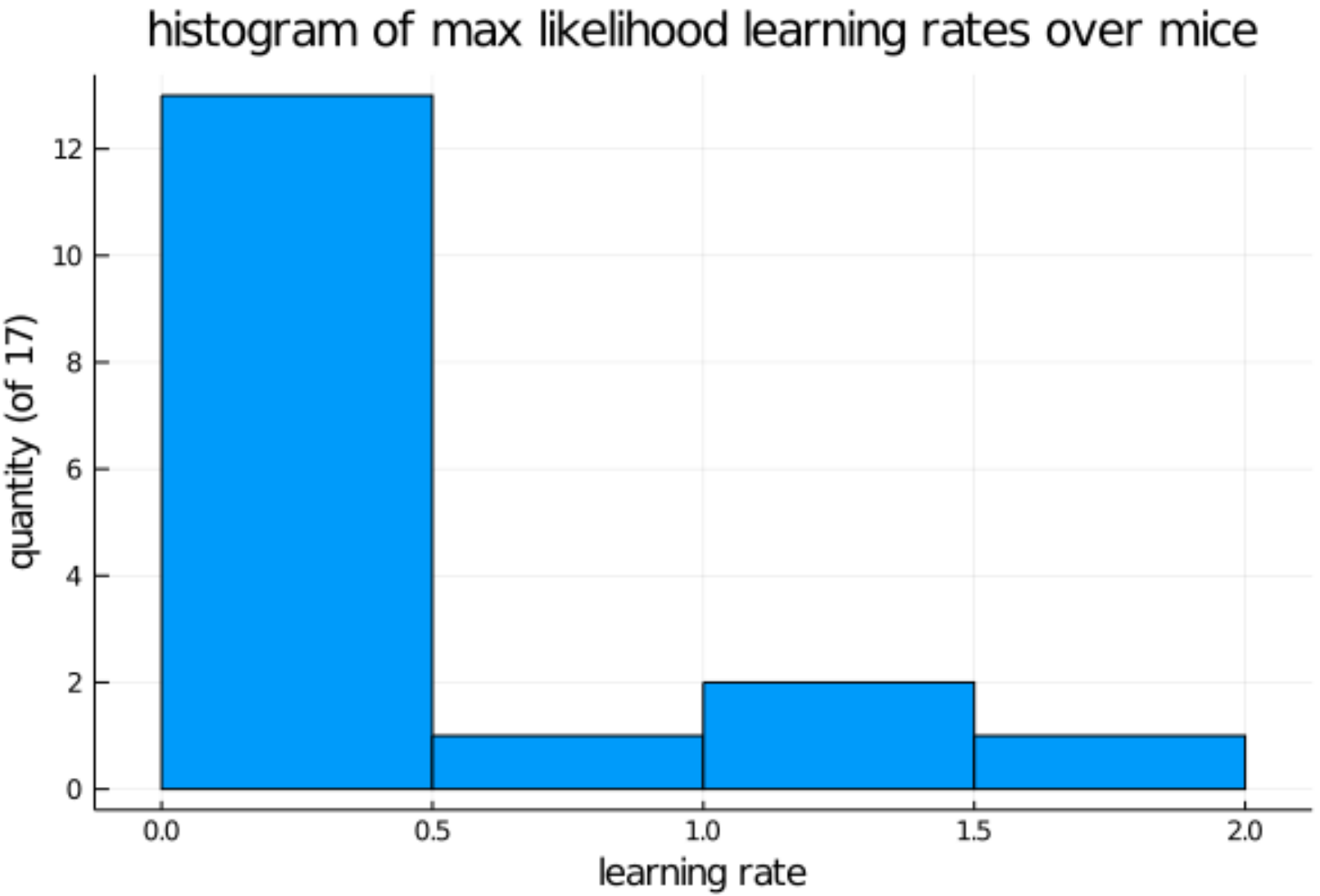

Average value of fit parameters over all mice
value
1.5
1.0
0.5
0.0
learning rate
softmax beta
decay factor

Results

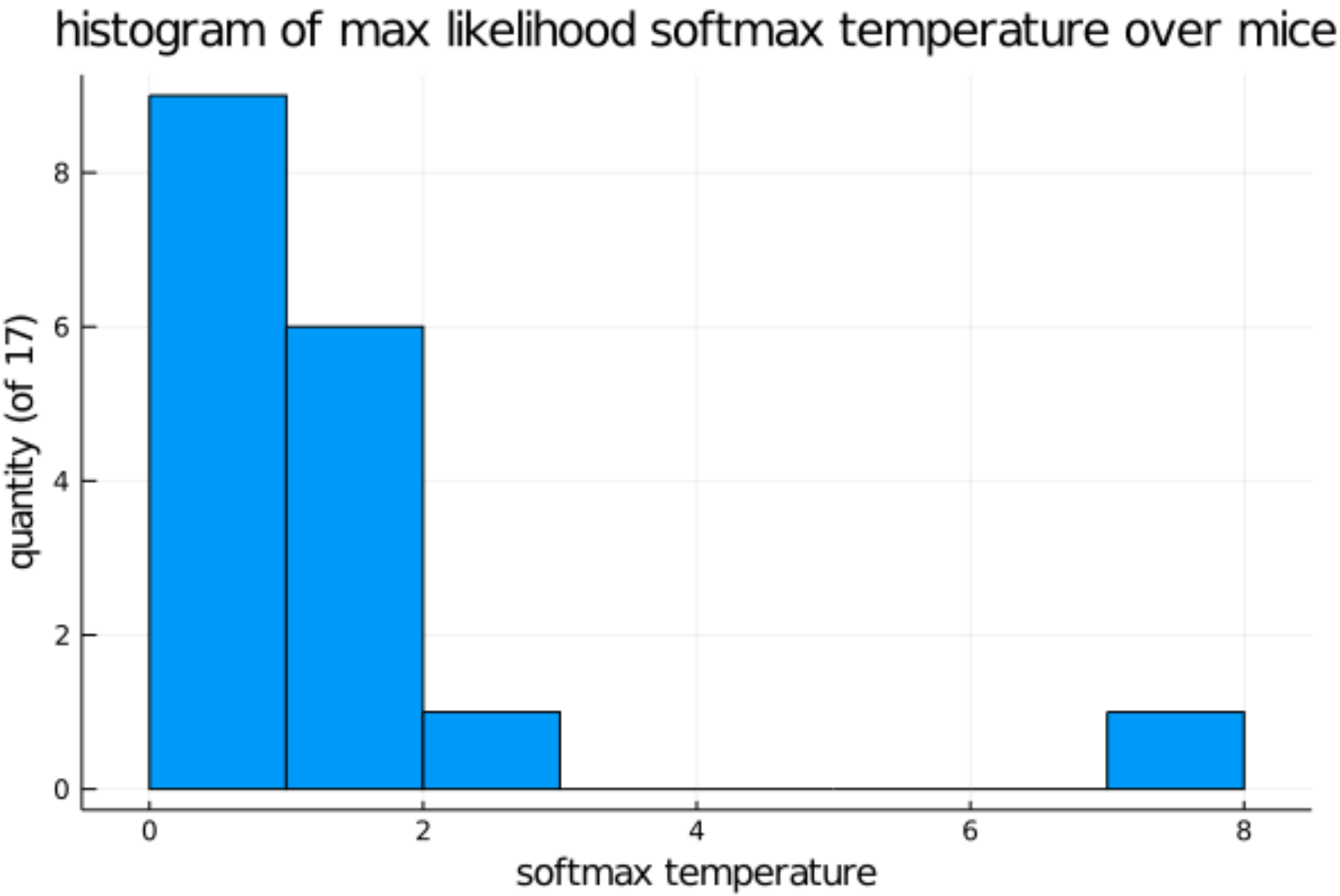

Bernoulli Go Neurons UCB
0.1
0.9
Value
Timestep
0.0
0.3
0.6
0.9
1.2
0
250
500
750


Bernoulli No-Go Neurons UCB
0.1
0.9
Value
Timestep
0.0
0.2
0.4
0.6
0.8
0
250
500
750

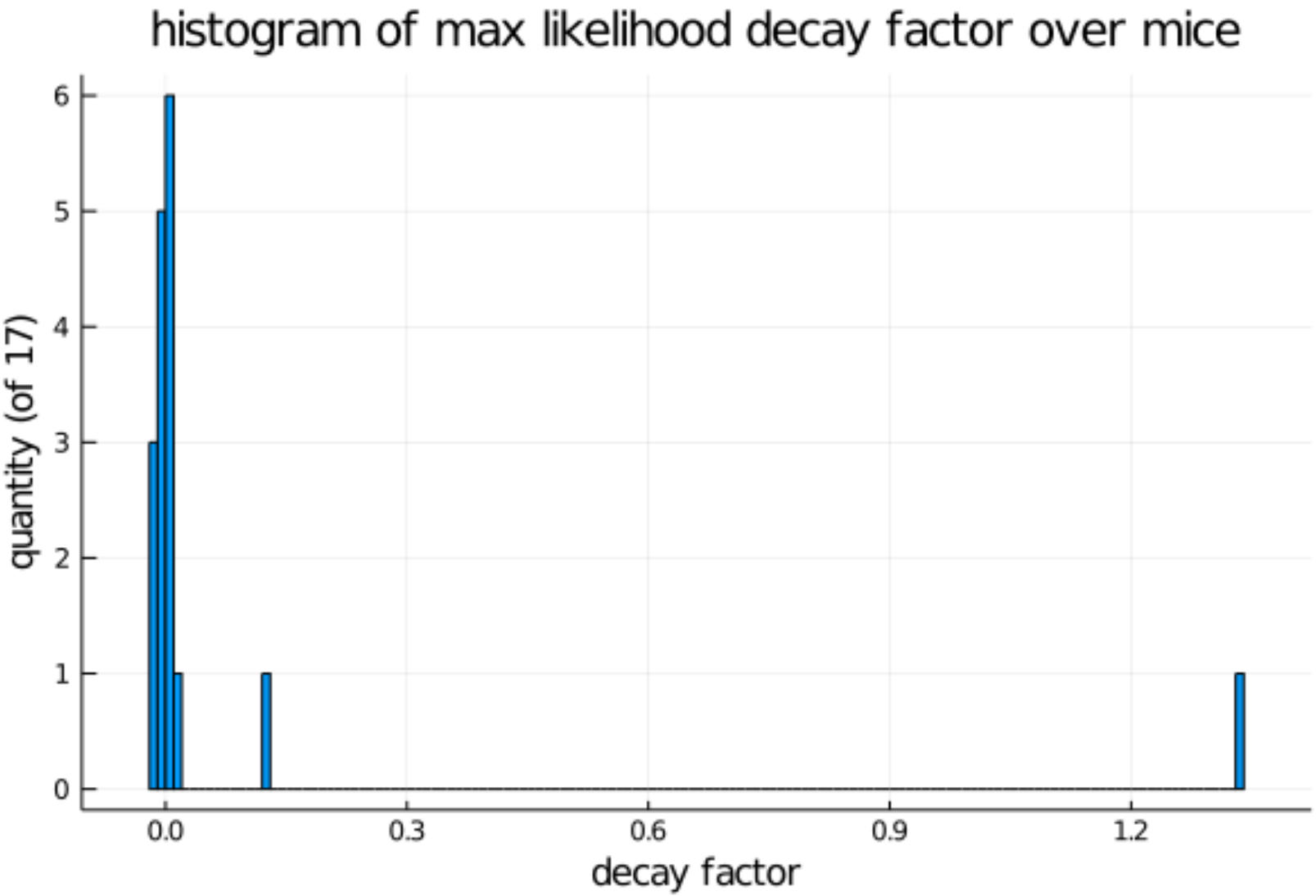
Results


histogram of max likelihood learning rates over mice
quantity (of 17)
0
2
4
6
8
10
12
0.0
0.5
1.0
1.5
2.0
learning rate

Results


histogram of max likelihood softmax temperature over mice
quantity (of 17)
0
2
4
6
8
0
2
4
6
8
softmax temperature

Results


histogram of max likelihood decay factor over mice
quantity (of 17)
0
1
2
3
4
5
6
0.0
0.3
0.6
0.9
1.2
decay factor

Reward Bases Presentation

- And now for something completely different...
- Have been wondering for a while about whether it is possible to compute instantaneous reward revaluations in a purely model free mode
- Chatted about this with Rafal and he suggested presenting it here to see if anyone has ideas or knows anything related in the literature
- Still at a very early stage and very much theoretical

Reward Revaluation

- Animals need to flexibly and efficiently respond to changes in reward distribution for the same states. I.e. Animal needs to seek out food when hungry, but not when satiated. Etc
- Crucially, these reward revaluations happen *all the time* for homeostatic reasons so this is a core computational problem the brain has to solve
- Classical model-free methods do not solve this and would require the entire model to be re-learnt from scratch every time a reward revaluation occurs which is clearly not feasible for the brain
- Compelling evidence that the brain can actually solve this problem immediately in a zero-shot setting (animals do not have to even experience the reward to compute the reward revaluation (!))

Dead-sea salt experiment

- Famous "dead sea salt" experiment – mice who have been trained to avoid lever which gives out aversive highly salty water.
- when injected with chemicals that simulate severe salt deficiency, the mice immediately start pulling the previously aversive lever, even when it (now) no longer gives out any salty water at all.
- So the mice they never experience salt in its positively valenced state but can still compute good policy with changed rewards.
- This cannot be explained with model-free RL which would at least have to experience the new reward to update value function etc

Related Work

- Dayan claim that these results imply reward revaluation must be due to model-based planning which, given an accurate reward model, can reproduce zero-shot reward revaluation. Problem is model-based planning is cmoputationally expensive, so doing this all the time for minor homeostatic changes seems overkill
- The successor representation of RL (perhaps implemented in hippocampus (Stachenfeld et al)) can do this, but is also computationally expensive.
- Also, since reward revaluation is so common in practice, there *must* be some way to do it in the basal ganglia model-free learning system since otherwise the basal ganglia would be pretty useless as would have to relearn everything from scratch whenever you get hungry (!)
- Zhang et al (2009) have a 'homeostatic' motivation' model where you have an additional homeostatic factor κ which just multiplies the reward in the TD calculation. But this can only work with 1 step tasks as does not change the value function

Reward Bases

- The idea is that instead of simply saying there is a monolithic reward function, we instead treat our reward as a combination of lots of smaller, more specialized reward functions, which can be linearly combined to produce our true 'overall reward'
- This means we model the reward function as $R = \sum_i \alpha_i R_i$ where R_i is a component reward function and α_i is a weighting coefficient.
- We call the R_is "reward bases" since they are effectively basis vectors from linear algebra

Reward Bases

- Crucially we then have that:

$$V = (I + \gamma T + \gamma^2 T^2 \dots) R \tag{5}$$
$$= (I + \gamma T + \gamma^2 T^2 \dots) \alpha_1 R_1 + (I + \gamma T + \gamma^2 T^2 \dots) \alpha_2 R_2 \dots \tag{6}$$
$$= \sum_{i=1}^{N} (I + \gamma T + \gamma^2 T^2 \dots) \alpha_i R_i \tag{7}$$
$$= \sum_{i=1}^{N} \alpha_i V_i(s) \tag{8}$$

- where V is the value function, γ is the discount factor, T is the transition matrix.
- Which means that if the reward can be linearly decomposed into a sum of bases, then so is the value function

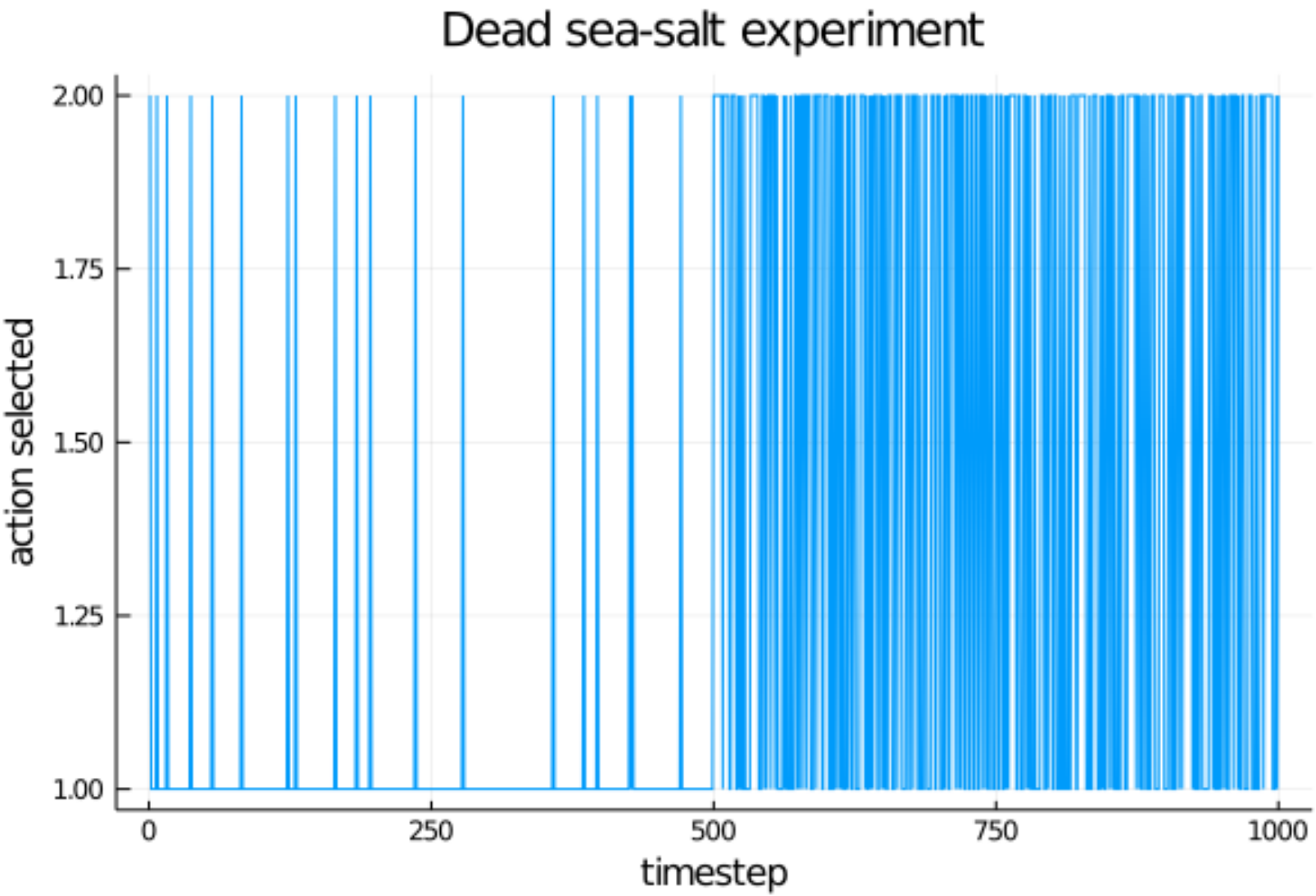
Reward Bases

- Effectively, for each reward basis, we learn a separate value function basis by standard model-free methods (TD learning etc)
- Then, if we want to construct a reward function out of our value function by weighting the rewards with coefficients α, then we know that the optimal value function is simply the weighted sum of all those value function bases, weighted by the same coefficients
- This allows for instant (zero-shot) revaluation of the *value function* for any set of αs – i..e we can instantly do reward-revaluation for any point spanned by the reward basis vectors
- Since we are computing a new *value function*, then this allows for the instant shift of complex multi-step behaviours due to reward revaluation, rather than simply the next time-step.

Reward Bases

- In the brain (basal ganglia) this would look like computing a bunch of different rewards and value functions in parallel – which seems highly doable. Not sure if there is evidence for this though
- Then to compute final value function, we would sum up all the value bases, weighted by the coefficients, which could be provided from elsewhere (i.e. cortex for complex bases, hypothalamus for homeostatic variables etc).
- A key question is how the reward bases are chosen. Most likely solution (to me) seems that animals have a number of fixed bases corresponding to homeostatic variables and other commonly observed cirumstances
- And then perhaps a more flexible cortical system which can create new bases based on repeated experiences

Results


Dead sea-salt experiment
action selected
2.00
1.75
1.50
1.25
1.00
0
250
500
750
1000
timestep