Report of Interacting Adaptive Processes with Different Timescales Underlie Short-Term Motor Learning

Jinshan Yang
University of Southern California
Los Angeles, California, 2134400388
Email: yangjins@usc.edu

***Abstract*—Reproduce the three models that used in simulating progression of motor output introduced in the paper Single State Model, Two State Gain-Specific Model and Two State Gain-Independent, Multi-Rate Model with Python.**

1. Reproduce Figure 1B

I created a file name the reproduce figure1B to implement three models.

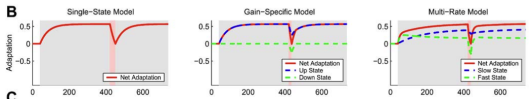

Figure 1: simulations with three models

Here according to Single State Model:

$$x(n+1) = A \cdot x(n) + B \cdot e(n) \ (B_f > B_s, A_s > A_f) \quad (1)$$

And the figure 1 it draws, I set the target position is 0.5, set A is 0.8 and B is 0.5, with 50 iterations, I reproduce the single state model as Figure 2.

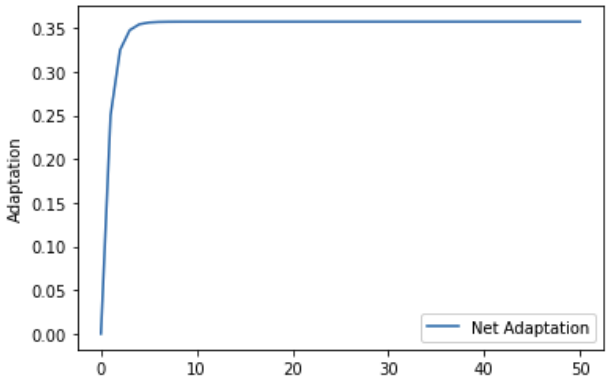

Figure 2: The Single State Model

And according to the Gain-Specific Model:

$$x_1(n+1) = min(0, [A \cdot x_1(n) + B \cdot e(n)]) \quad (2)$$
$$x_2(n+1) = max(0, [A \cdot x_2(n) + B \cdot e(n)]) \quad (3)$$
$$(B_f > B_s, A_s > A_f) \quad (4)$$
$$x = x_1 + x_2 \quad (5)$$

I set the target position is 0.5, with x_1, x_2 initially 0, set A is 0.8 and B is 0.5, with 50 iterations, I reproduce the single state model as Figure 3.

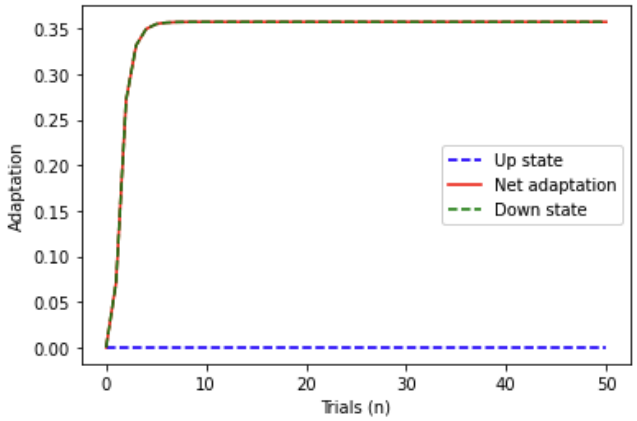

Figure 3: The Gain-Specific Model

And according to the two state, gain-independent, Multi-Rate Model:

$$x_1(n+1) = A_f \cdot x_1(n) + B_f \cdot e(n) \quad (6)$$
$$x_2(n+1) = A_s \cdot x_2(n) + B_s \cdot e(n) \quad (7)$$
$$(B_f > B_s, A_s > A_f) \quad (8)$$
$$x = x_1 + x_2 \quad (9)$$

I set the target position is 0.5, with x_1, x_2 initially 0, set A fast is 0.9 and B fast is 0.5, A slow is 0.7 and B slow is 0.3 with 50 iterations, I reproduce the single state model as Figure 5.

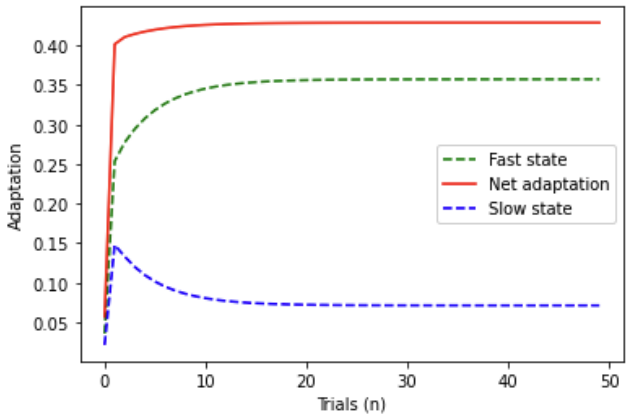

Figure 4: Multi-Rate Model

2. Generate "synthetic data"

Then I used these three models to generate "synthetic data" with the same parameter but adding noise to the output. Since the image of 3D is the experimental data with (mean+/- standard error of the means), I generate 50 data value uniformly from 0 to 1 increasingly. Then with the same parameter as before, I get a similar image from adaptation 0 to 1 as following Figure **??**.

I generate the mean data and set upper bound and lower bound for the generated experimental data in blue line and points with the line, I plot the Multi-rate model with same parameter in green, Single state model with same parameter in red, Gain-specific model with same parameter with the magenta dotted line.

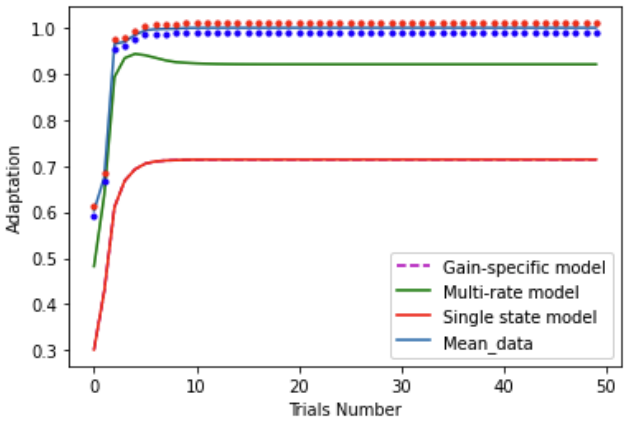

Figure 5: Average Time Course of Adaption

3. Estimate parameters from the synthetic data

First, I created two list, index list and the synthetic data list, the synthetic data list is the list generated by the task 2.

Then, import logging, pyro, torch to use Bayesian sampling with pyro. I define a model function and a guide function. Define a probabilistic model in Pyro that represents the process that generated the synthetic data. This model should include the unknown parameters that we want to estimate. I defined the unknown parameters alpha and beta, defined the likelihood of the data given the parameters in model function with $mean = \alpha + \beta \times x$.

In the guide function, which will be used to perform the Bayesian sampling. The guide function should approximate the posterior distribution over the unknown parameters.

Use the synthetic data to define pyro.infer.SVI, which will perform stochastic gradient descent on the model and guide functions to optimize the parameters of the guide function and approximate the posterior distribution.

Finally, I plot the ELBO loss vs SVI step as Figure 6. And I have get the parameters: alpha is 0.929739, with scale 0.010572155. beta is -0.0015218882 with scale 0.00053218246 7.

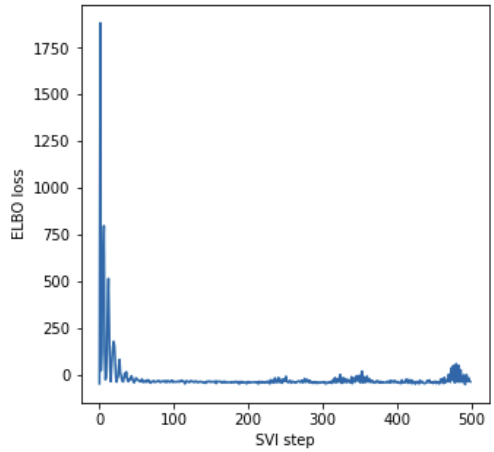

Figure 6: ELBO LOSS

```
AutoNormal.locs.alpha 0.929739
AutoNormal.scales.alpha 0.010572155
AutoNormal.locs.beta -0.0015218882
AutoNormal.scales.beta 0.00053218246
AutoNormal.locs.sigma -4.676358
AutoNormal.scales.sigma 0.10830223
```

Figure 7: Parameter

4. Code Available

The code is available at Github