Appendix A Reformulating the Hybrid Learning Model Using the State-Space Modeling Framework

Here, we show how the model of Gold et al. (2012) can be implemented using the state-space modeling framework and thus the COMPSS toolbox. Table A.1 shows a line-by-line comparison between two models. Here, we focus on the Hybrid model proposed in the paper. The pure actor-critic or Q-learning processes are special cases of this model with parameter c (see below) set to 0 or 1, respectively.

Table A.1 Hybrid model of actor-critic and Q-learning and its equivalent state-space model

Hybrid Model	Mapping	State-Space Model
Value for each category – $s \in \{s_1, s_2, s_3, s_4\}$ – is updated by $\ddot{a}(k) = r(k) - V(s,k)$ $V(s,k+1) = V(s,k) + w_{v,s}\ddot{a}(k)$	$x_s(k) = V(s,k)$	Latent state variables to represent value dynamics per each category – $s \in \{s_1, s_2, s_3, s_4\}$ $x_s(k+1) = (1 - w_{v,s})x_s(k) + w_{v,s}r(k) + \varepsilon_s$ $\varepsilon_s \sim N(0, \sigma_s^2) \quad \sigma_s^2 \to 0$
Critics' weights are updated by actions – there are two actions in this task (left or right image)– $a \in \{a_1, a_2\}$.+1 $w(s,a,k+1) = w(s,a,k) + w_{c,sa}\ddot{a}(k)$	$x_{as}(k) = w(s,a,k)$	Latent variables to represent critics' weights for taking action a when in state s. There are thus in total $2\times s$ latent variables $x_{as}(k+1) = x_{as}(k) - w_{c,sa}x_s(k) + w_{c,sa}r(k) + \varepsilon_{as}$ $\varepsilon_{as} \sim N(0, \sigma_{as}^2) \quad \sigma_{as}^2 \to 0$
Q-learning – learning the action directly $Q(a,k+1) = Q(a,k) + w_{q,a}(r(k) - Q(a,k))$	$x_a(k) = \mathrm{Q}(a,k)$	Latent variables to represent Q-learning – there are 2 latent variables, each representing one action $x_a(k+1) = (1 - w_{q,a})x_a(k) + w_{q,a}r(k) + \varepsilon_a$ $\varepsilon_a \sim N(0, \sigma_a^2) \quad \sigma_a^2 \to 0$
Action probability in the hybrid model $H(s,a,k) = [(1-c)w(s,a,k) + c\,Q(a,k)]$ $P(a,k) = \exp(H(s,a,k)) \Big/ \sum_a \exp(H(s,a,k))$	z_k: action at time k action 1: $z_k = 1$ action 2: $z_k = 0$	Model of observation process z_k: observation at time k, either action 1 or action 2. $\mathrm{logit}\, p(z_k = 1) = (1-c)x_{as}(k) + cx_a(k)$

Hybrid Model Using the State-Space Modeling Framework. For clarity, we show the case where there is a single trial type being learned, i.e. we represent only variables x_{s_1}, x_{a_1}, and $x_{a_1 s_1}$. For four trial types as in the Gold et al. study, we would add six more state variables representing trial types 2 through 4 and the corresponding action values in those trial types.

State Variable Defintion

$X_k = \left[x_{s_1}(k)\ \ x_{a_1 s_1}(k)\ \ x_{a_1}(k)\right]'$

$U_k = r(k)$

$$W \sim N\left(0, \begin{bmatrix} \sigma_{s_1}^2 & 0 & 0 \\ 0 & \sigma_{a_1 s_1}^2 & 0 \\ 0 & 0 & \sigma_{a_1}^2 \end{bmatrix}\right)$$

$$X_{k+1} = \begin{bmatrix} 1 - w_{v,s_1} & 0 & 0 \\ 1 & -w_{c,s_1 a_1} & 0 \\ 0 & 0 & 1 - w_{q,a_1} \end{bmatrix} X_k + \begin{bmatrix} w_{v,s_1} \\ w_{c,s_1 a_1} \\ w_{q,a_1} \end{bmatrix} U_k + W$$

Observation Model

$$\mathrm{logit}\, p(z_k = 1) = [0 \quad (1-c) \quad c]\, X_k$$

The model is cast in terms of the trial type – s – and the possible actions within that trial type – a. a can be seen as picking the correct or incorrect choice per each trial.. Note that we only need to characterize one of these two actions per category; this is because not taking an action implies the other action is being taken. We assign two state variables to each task's category to represent the value of being in a state and of taking

action 1 during that state – i.e. we need $x_{s_1}(k)$ and $x_{a_1 s_1}(k)$ for category 1. We assign one state variable to represent the global value of taking action 1 across all possible states/trial types– $x_{a_1}(k)$. Now, we can define the state-transition process for the task by (note action a_1 will be common across states):

$$X_k = \left[x_{s_1}(k)\ \ x_{a_1 s_1}(k)\ \ x_{s_2}(k)\ \ x_{a_1 s_2}(k)\ \ x_{s_3}(k)\ \ x_{a_1 s_3}(k)\ \ x_{s_4}(k)\ \ x_{a_1 s_4}(k)\ \ \ x_{a_1}(k)\right]' \quad \text{(A.1.a)}$$

$$U_k = \left[r(k) * I_{s_1}(k) \quad r(k) * I_{s_2}(k) \quad r(k) * I_{s_3}(k) \quad r(k) * I_{s_4}(k)\right]' \quad \text{(A.1.b)}$$

$$X_{k+1} = \overbrace{\begin{bmatrix} 1-w_{v,s_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -w_{c,s_1 a_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-w_{v,s_2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -w_{c,s_2 a_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1-w_{v,s_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -w_{c,s_3 a_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1-w_{v,s_4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -w_{c,s_4 a_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-w_{q,a_1} \end{bmatrix}}^{\mathbf{Ak}} X_k + \cdots$$

$$\overbrace{\begin{bmatrix} w_{v,s_1} & 0 & 0 & 0 \\ w_{c,s_1 a_1} & 0 & 0 & 0 \\ 0 & w_{v,s_2} & 0 & 0 \\ 0 & w_{c,s_2 a_1} & 0 & 0 \\ 0 & 0 & w_{c,s_3} & 0 \\ 0 & 0 & w_{c,s_3 a_1} & 0 \\ 0 & 0 & 0 & w_{v,s_4} \\ 0 & 0 & 0 & w_{c,s_4 a_1} \\ w_{q,a_1} & w_{q,a_1} & w_{q,a_1} & w_{q,a_1} \end{bmatrix}}^{\mathbf{Bk}} U_k + W_k \quad \text{(A.1.c)}$$

$$\text{logit}\, p(z_k = 1) = \overbrace{\begin{bmatrix} 0 & (1-c)I_{s_1}(k) & 0 & (1-c)I_{s_2}(k) & 0 & (1-c)I_{s_3}(k) & 0 & (1-c)I_{s_4}(k) & c \end{bmatrix}}^{\mathbf{E}} X_k \quad \text{(A.1.d)}$$

where, $I_{s_i}(k)$ is an indicator function which is 1 when the current trial type corresponds to stimulus i. In equation (A.1.c), the probability of taking each action is defined as a function of the learning, captured in the vector X_k. X_k represents both actor-critic and Q-learning values – the first 8 entries describe actor-critic learning for each trial type, while the 9th element represents the global Q-learning process that focuses solely on the action regardless of the trial type. As Gold et al. did, we represent the relative weight of these two learning processes (actor-critic and Q-learning) through a mixing parameter *c*. Using the Logit function defined in (A.1.c), the probability of taking action a_1 is defined by $c\ \ x_{a_1}(k)$ in the absence of the actor-critic model. The probability of taking action a_1 when presented with trial type s_1, in the actor-critic model, is defined by $(1-c)x_{a_1 s_1}(k)$. The overall probability of taking action a_1 when presented with s_1 is thus defined by $c\ \ x_{a_1}(k) + (1-c)x_{a_1 s_1}(k)$. We can similarly define the probability of both actions for the other 3 trial types. Given the model definition, an increase in $x_{a_1}(k)$ implies a global preference for a_1 in all trial types, while an increase in $x_{a_1 s_1}(k)$ implies a specific preference for a_1 when presented with s_1.

The model of Table A.1 translates directly into MATLAB code, as shown in FIGURES. This close mapping between model specification and computer code makes COMPSS easier to use and interpret.

```
%% Load Reinforcement Learning Data
% data: In, RESP, dGain, Type, List, Gain
% There are 160 trials in total
% In:    160x4, each column corresponds to one of four different states
% ACC:   160x1, it is the accuracy (correct/ Incorrect)
% dGain: 160x1, it is the reward received per each trial
% Type:  160x1, it indicates loss-win trials (1 is for the win trials, and -1 is for the loss trials)
% List:  160x1, it show the trial state - it matches to In
% Gain:  160x1, is the comulative gain
T = csvread('HNS2.csv',1,1);
dGain = T(:,8);
ACC   = T(:,3);
List  = T(:,6);
In    = zeros(160,4);
In(find(List==1),1)=1;
In(find(List==2),2)=1;
In(find(List==3),3)=1;
In(find(List==4),4)=1;
%% Set Behavioral Model
% create model
% - 9 state variables, 2 state variables per stimulus plus one for action
% - 4 input to the state process, reward per stimulus
Param = COMPSS_create_state_space(9,4,0,9,eye(9,9),[],[],[1 2 3 4 5 6 7 8 9],[0 0 0 0 0 0 0 0 0]);
RATE_A = 0.1;
RATE_B = 0.1;
RATE_C = 0.1;
% set initial value for Wv,s1 to RATE_A
Param.Ak(1,1)=  1-RATE_A;
Param.Ak(2,1)=  1;          Param.Ak(2,2)= -RATE_B;
% set initial value for Wv,s2 to RATE_A
Param.Ak(3,3)=  1-RATE_A;
Param.Ak(4,3)=  1;          Param.Ak(4,4) = -RATE_B;
% set initial value for Wv,s3 to RATE_A
Param.Ak(5,5)= 1-RATE_A;
Param.Ak(6,5)= 1;           Param.Ak(6,6)= - RATE_B;
% set initial value for Wv,s4 to RATE_A
Param.Ak(7,7)= 1-RATE_A;
Param.Ak(8,7)= 1;           Param.Ak(8,8)=  - RATE_B;
% set initial value for Wq,a1 to 1
Param.Ak(9,9)= 1-RATE_C;
% set initial value of B
Param.Bk(1,1)   = RATE_A;
Param.Bk(2,1)   = RATE_B;
Param.Bk(3,2)   = RATE_A;
Param.Bk(4,2)   = RATE_B;
Param.Bk(5,3)   = RATE_A;
Param.Bk(6,3)   = RATE_B;
Param.Bk(7,4)   = RATE_A;
Param.Bk(8,4)   = RATE_A;
Param.Bk(9,1:4) = RATE_C;
% set Param.Wk
Param.Wk = 0.01 * eye(9,9);
% Set Param.Ek
Param.Ek(1) = 0;Param.Ek(3) = 0;Param.Ek(5) = 0;Param.Ek(7) = 0;
%% Set Learning Procedure
Iter  = 1000;
Param = COMPSS_set_learning_param(Param,Iter,3,1,1,1,1,1,0,2,0);
%% Format the Data
% Yb
Yb = ACC;
% Ib
Ib = zeros(length(Yb),9);
Ib(find(In(:,1)),2) = 1;
Ib(find(In(:,2)),4) = 1;
Ib(find(In(:,3)),6) = 1;
Ib(find(In(:,4)),8) = 1;
Ib(:,9) = 1;
% all data points are valid
valid = ones(length(Yb),1);
% Uk - is the reward
Uk = zeros(length(Yb),4);
Uk(:,1)= dGain.*Ib(:,2);
Uk(:,2)= dGain.*Ib(:,4);
Uk(:,3)= dGain.*Ib(:,6);
Uk(:,4)= dGain.*Ib(:,8);
%% Run learning with Gamma model
[XSmt,SSmt,Param,XPos,SPos,ML,~,Pb]=COMPSS_em([0 1],Uk,[],Ib,[],Yb,Param,valid);
```

Figure A.1 Analysis script for the hybrid learning task

In the script, RATE_A and RATE_B defines intial values for w_{v,s_i} and $w_{c,s_i a_1}$ $i = 1 \cdots 4$. RATE_C defines

intial value for w_{q,a_1}. The c parameter, defined in (A.1.4), is set to 0 in the script. We define covariance matrix for W_k using Param.Wk; it is set to be a digonal matrix with a small variance. Elements of Ib and Uk are filled by their corresponding values; these values are the stimulus information and returned reward which are the input to the model.

The behavioral signal analyzed in Gold et al. (2012) is solely the action – or decision –taken on each trial. However, there is also a reaction time signal, which might carry extra information about the learning evolution or attribute as in the Prerau et al. papers discussed in section 4. For instance, we might expect to see a shorter reaction time as the correct actions are learned and the subject gains confidence in his/her decisions. It would be straightforward to add reaction time to the model as a function of the learning variables, X_k, and previous decision outcome as a history term. We can define how the reaction time is linked to the model state variables or input – `COMPSS_create_state_space`, and set the type of input being passed to the `COMPSS_em` routine. We can also define how parameters of the reaction time model will be trained using `COMPSS_set_learning_param function`. We even can define the censoring criteria if there is any data point being censored for a long response time, `COMPSS_set_censor_threshold_proc_mode`.