TNM Exercise 2: Predictive Coding & HGF

Saurabh Vaishampayan (svaishampaya@student.ethz.ch)
Linus Rüttimann (rlinus@student.ethz.ch)
Elia Torre (etorre@student.ethz.ch) Group **H**
Timothy Kurer (tkurer@student.ethz.ch) 11.03.2023

1 Code organisation

In the submission folder you will find multiple matlab scripts and live scripts. Ex2_1.mlx deals with code for 2.1, ex2_2.m deals with code for 2.2. ex2_3,mlx deals with code for 2.3. The rest files are for configuration of binary HGF and unit square response function for different subquestions (self evident from naming).

2.1 Speed of Convergence in Predictive Coding

a

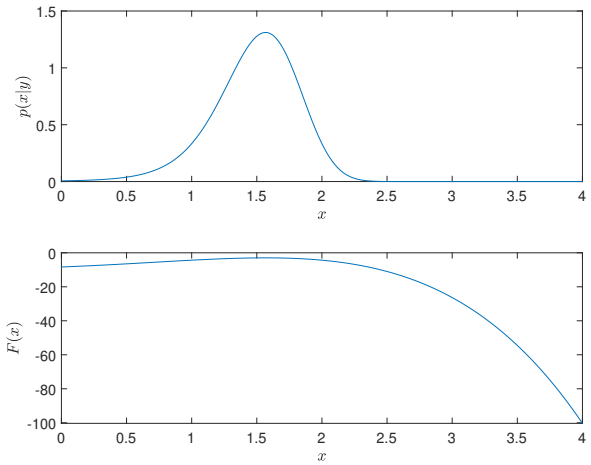


Figure 1: $p(x|y)$ and $F(x)$

The posterior is not normal.

b

The gradient of $F(\phi)$ is given by: $\frac{dF}{d\phi} = \frac{\mu_{pr}-\phi}{\Sigma_{pr}} + \frac{y-g(\phi)}{\Sigma_{gen}}2\phi$. The following code performs a gradient descent on $F(\phi)$:

```
mu_pr = 3;
sigma_pr = 1;
sigma_gen = 1;
y=2;

dF_over_dphi = @(phi) (mu_pr-phi)/sigma_pr + (y-phi.^2)/sigma_gen*2*phi;

dtau = 0.01;
nstep = 600;
phi = zeros(nstep,1);
phi(1) = mu_pr;

for i=2:nstep
```

```
    phi(i) = phi(i-1)+dtau*dF_over_dphi(phi(i-1));
end
tau = 0:dtau:dtau*(nstep-1);

figure(2);
plot(tau, phi);
xlabel('$\tau$','Interpreter','latex');
ylabel('$\phi(\tau)$','Interpreter','latex');

fprintf("phi_0 = %f", phi(end));
```

The gradient descent converts to the value: $\phi_0 = 1.567468$. Fig. 2 shows the trajectory $\phi(\tau)$.

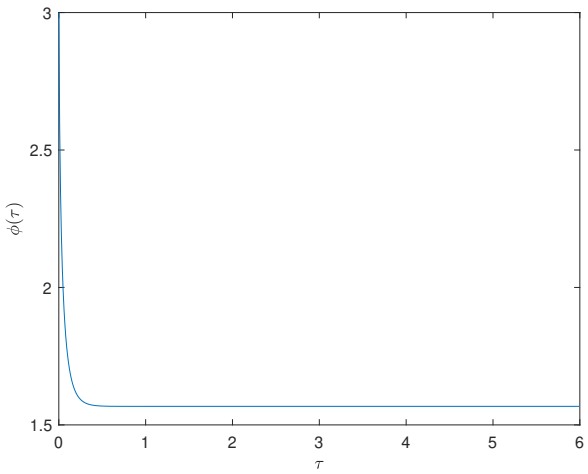


Figure 2: $\phi(\tau)$

c

The following code performs a gradient descent on the coupled differential equations:

```
phi2 = zeros(nstep,1);
error_pr = zeros(nstep,1);
error_gen = zeros(nstep,1);

phi2(1) = mu_pr;
for i=2:nstep
    phi2(i) = phi2(i-1)+dtau*(-error_pr(i-1) + error_gen(i-1)*2*phi2(i
        -1));
    error_pr(i) = error_pr(i-1)+dtau*(phi2(i-1) - mu_pr - sigma_pr *
        error_pr(i-1));
    error_gen(i) = error_gen(i-1)+dtau*(y - phi2(i-1).^2 - sigma_gen *
        error_gen(i-1));
end

figure(3);
plot(tau, phi2);
hold on;
plot(tau, error_pr);
plot(tau, error_gen);

xlabel('$\tau$','Interpreter','latex');
ylabel('Activity','Interpreter','latex');
legend('$\phi(\tau)$', '$\epsilon_{pr}(\tau)$', '$\epsilon_{gen}(\tau)$'
    ,'Interpreter','latex');
```

Fig. 2 shows the trajectory of $\phi(\tau)$,$\epsilon_{pr}(\tau)$, and $\epsilon_{gen}(\tau)$.

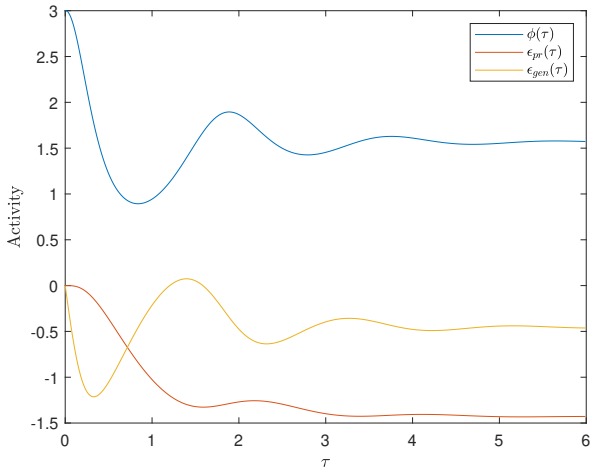


Figure 3: $\phi(\tau)$,$\epsilon_{pr}(\tau)$, and $\epsilon_{gen}(\tau)$

d

The method (b) converges faster, because it directly minimizes $F(\phi)$ without introducing the auxiliary variables ϵ_{pr}, and ϵ_{gen}.

2.2 The HGF toolbox and Experimental Design

a What are the two main functions of the toolbox and what do they do?

The TAPAS Matlab toolbox offers two primary functions - *tapas_fitModel(...)* and *tapas_simModel(...)*.

- *tapas_fitModel(...)* function aids in fitting models to observed responses by giving the option to choose a perceptual model, a response model, and an optimization algorithm. The perceptual model you choose can be a Bayesian generative model or a reinforcement learning algorithm, whereas the response model explains how the states of the perceptual model translate into responses. The optimization algorithm is used to estimate the maximum-a-posteriori (MAP) values of the parameters of both models. Its objective function is the unnormalized log-posterior of all perceptual and response parameters given the data and the perceptual and response models. This corresponds to the log-joint of data and parameters, given the models. It's essential to choose perceptual and response models that work in tandem, while the choice of optimization algorithm is independent.

- *tapas_simModel(...)* function allows the simulation of perceptual states and responses. This is done by choosing a perceptual model and an observation model (to simulate responses). The perceptual model can be a Bayesian generative model or a reinforcement learning algorithm, while the observation model determines how the states of the perceptual model relate to responses. Examples of observation models are the softmax decision rule or the closely related unit-square sigmoid decision model.

b

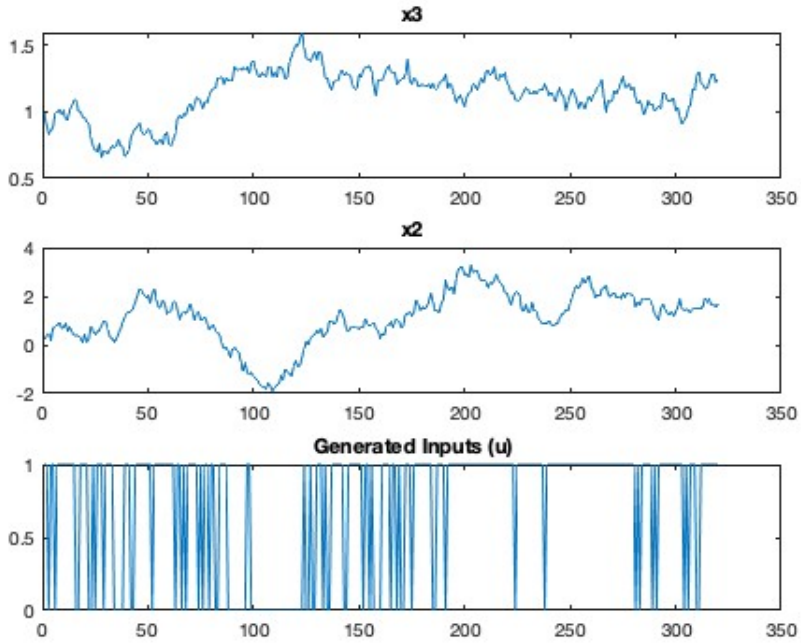
Figure 4: Sensory Inputs u generation using HGF

```matlab
% Main script
iter = 320;
k2 = 1;
omega2 = -4;
omega3 = -6;
theta = exp(omega3);
x2_init = 0;
x3_init = 1;

[u, x2, x3] = generative_model_hgf_binary(iter, k2, omega2, theta, x2_init, x3_init);

figure;
subplot(3, 1, 1);
plot(x3);
title('x3');
subplot(3, 1, 2);
plot(x2);
title('x2');
subplot(3, 1, 3);
plot(u);
title('Generated Inputs (u)');
```

```matlab
function [u, x2, x3] = generative_model_hgf_binary(iter, k2, omega2, theta, x2_init, x3_init)
    % Initialize variables
    u = zeros(1, iter);
    x2 = zeros(1, iter);
    x3 = zeros(1, iter);

    x2_curr = x2_init;
    x3_curr = x3_init;

    % Iterate over trials
    for t = 1:iter
        % Update x3
        x3(t) = x3_curr + sqrt(theta) * randn();

        % Update x2
        x2(t) = x2_curr + sqrt(exp(k2*x3_curr+omega2)) * randn();

        % Generate input u
        u(t) = rand < sigmoid(x2(t));

        % Update current states
        x2_curr = x2(t);
        x3_curr = x3(t);
    end
end

function y = sigmoid(x)
    y = 1.0 ./ (1.0 + exp(-x));
end
```

By running this code several times (Figure 2), we observe different traces for x_3, x_2, and u. When trying out different parameter settings, especially higher values for θ, the dynamics of the model will change. If θ increases, the standard deviation of the Gaussian noise added to x_3 (level 3) increases, resulting in more variation in the trajectories of x_3. This leads to a more volatile representation of the environment at the third level. Due to the increased volatility in x_3, the Gaussian noise added to x_2 (level 2) will also exhibit larger variations, causing more fluctuation in the trajectories of x_2. As a consequence, the generated sensory inputs u will display more variability since they are generated based on the beliefs at the second level (x_2). In summary, varying θ in this modified HGF model will directly affect the volatility of both x3 and x_2, which in turn affects the variability of the generated sensory inputs u.

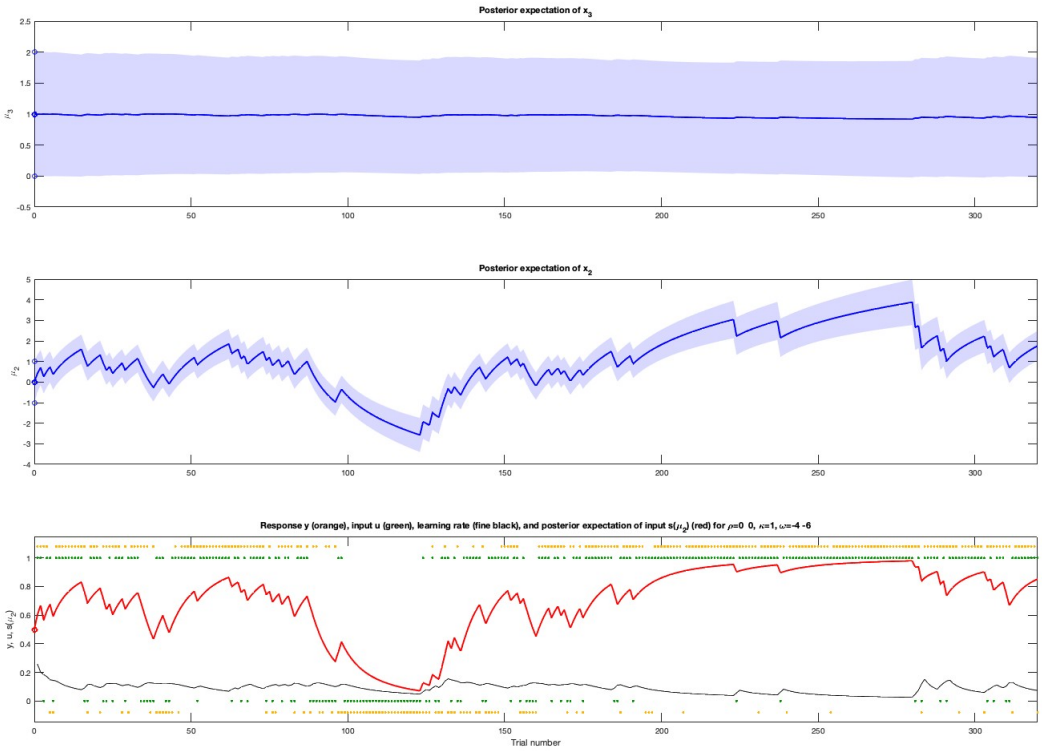
Figure 5: 320 trials trajectories for x_2, x_3 and generated u



c

```
%u = load('example_binary_input.txt')
sim = tapas_simModel(u','tapas_hgf_binary', [NaN 0 1 NaN 1 1 NaN 0 0 1 1 NaN -4 -6], 'tapas_unitsq_sgm', 1);
tapas_hgf_binary_plotTraj(sim);
```

Figure 6: Belief and responses simulation



The model seems to be tracking the evolution of x_2 quite accurately, with the true x_2 always within the range of variance signalled by the shaded region. The same can be said about x_3. Since the true x_3 has smaller variations over the 320 trials, the reference axis are adjusted both in the true and estimated plot, which makes it look, at a first glance, like it does not correctly track the evolution. However, the evolution of the true x_3 is always contained in the variance estimated by the model. Overall, I would say that the model estimates correctly the evolution of the parameters, even if it does so by introducing a high-level of uncertainty signalled by the variance range around each estimate.

d

The HGF represents a valid generative model as it models how the real world behaves by capturing the dynamic nature of the stimuli, also thanks to its ability of capturing hierarchical relationships in the data. On the positive side, the generation of sequences via HGF allows the minimization of the human bias that a manual generation of the stimuli sequence entails, as well as, the analysis of individual subjects using personalized hierarchical structures. In particular, it can help identify patient-specific ability to learn and adapt to environmental uncertainty. Being neurobiologically grounded, the model aims to better grasp the underlying neural mechanisms. On the negative side, HGF is quite a complex model that requires prior knowledge of the environment's hierarchical structure (which might not always be available or accurately known). However, it remains a useful model for the agent to invert during perception, which stands as a foundation of adaptation to a changing environment.

2.3 Coordinate Choice and Parameter Identifiability in the HGF

a

First we simulate the responses using tapas_simModel. Then, we have to fit a model to observed responses and perceptual model. We first have to configure the perceptual model and the response model before we can pass these to the fitting function with the relevant priors. These are chosen from tapas_hgf_binary_config_2_3_a1.m and tapas_unitsq_sgm_config_2_3_a1.m respectively. In the assignment, we were not told explicitly about prior distributions so we allowed priors to be uninformative (by setting standard deviations to be large).
In first subpart we are asked to estimate, $\zeta, \mu_3, \kappa_2, \nu$. So we set there prior variances to be large while for others we keep them clamped to the ground truth and set their variance to be small. We observe the following correlation plot (in the typical scenario).

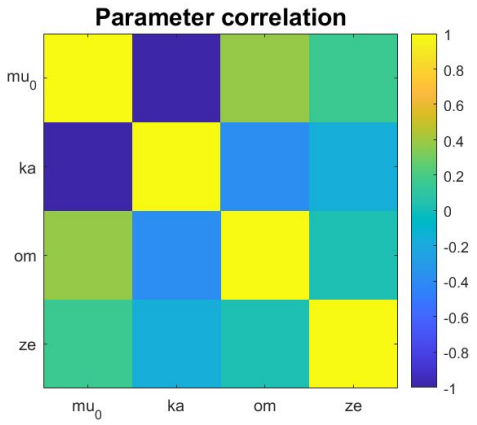


Figure 7: Correlation plots for 2.3a first subpart

In second subpart we are asked to estimate, $\zeta, \mu_3, \omega_2, \nu$. So we set there prior variances to be large while for others we keep them clamped to the ground truth and set their variance to be small. We observe the following correlation plot (in the typical scenario).

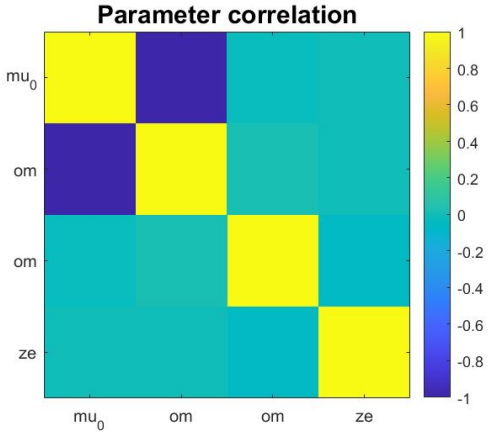


Figure 8: Correlation plots for 2.3a second subpart

The following observation is for above random seed. We had run it for different random seeds and the following observation holds in the typical set. It may happen that for some particular initialisation, you get slightly different correlation plots. But in the typical sense you will get the following result. We had fixed the seed for reproducibility. In general, we observe that in the first method estimates of the parameters are more correlated compared to second method. This is because in the second method kappa2 is fixed while omega2 was estimated. The variance of second level is $e^{\kappa_2 x_3 + \omega_2}$. κ_2 controls how the third level affects the second level while ω_2 controls the overall scaling of the variance (because it is in exponent) independent of what x_3 was. In the first method, we have to estimate both μ_3 and κ_2. Since this is hierarchical, and since κ_2 controls the coupling, setting both as free parameters during estimates

means that estimates become correlated. Whereas when one fixes κ_2, then how μ_3 changes doesn't affect ω_2, so estimates become uncorrelated.

b

We compare belief trajectories for agent simulation with parameters as given in 2.3a and 2.3b

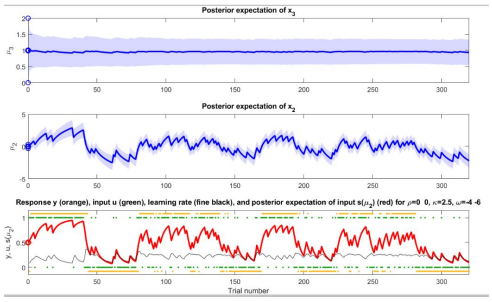


Figure 9: Belief trajectory plots for 2.3a

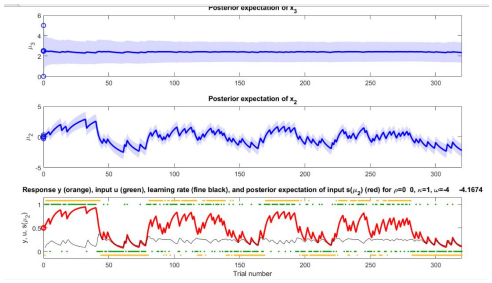


Figure 10: Belief trajectory plots for 2.3b

We observe that the trajectory for x_3 changes: This is obvious because the mean and variance for x_3 changed. However the belief trajectory for x_2 is the same in both plots. If one looks at the equations for x_2 one can see that the variance coupling involves the term $e^{\kappa_2 x_3 + \omega_2}$. If one observes carefully, the product $\kappa_2 \mu_3$ is the same in 2.3a and 2.3b (1 x 2.5 and 2.5 x 1) respectively. Additionally, in this question ω_2 is same in both cases. So $\mu_3 \kappa_2 + \omega_2$ remains same. Hence the trajectory for x_2 remains the same. Since the response function is constant, the response output also remains the same in both.

c

To keep the belief trajectories on lower levels the same, one has to scale μ_3 such that the product of $\mu_3 \kappa_2 + \omega_2$ is the same as new $\mu_3 x \kappa_2 + \omega_2$ as observed in 2.3b. This will keep the belief trajectory at second level the same, and because of hierarchical model, the belief trajectories at lower levels will also be same.

d

We now have to compare the correlation results for fitting settings in 2.3a and 2.3d. For 2.3d we first have to simulate the agent again for a new response function as given in the assignment. After fitting, the following are the correlation plots for the two subparts of 2.3a (where in one part ω_2 was fixed and in another part κ_2 was fixed respectively. We repeat the two methods for 2.3d also. The following observation

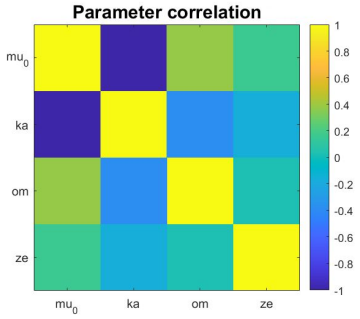
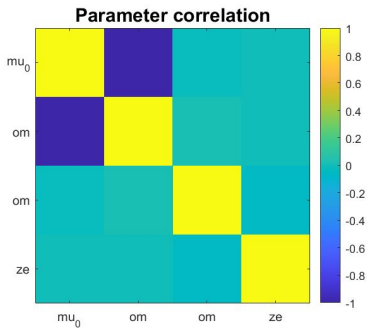
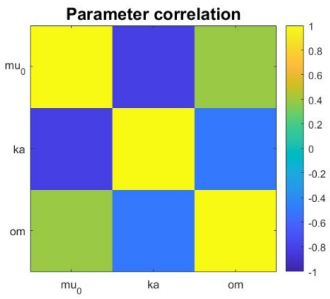
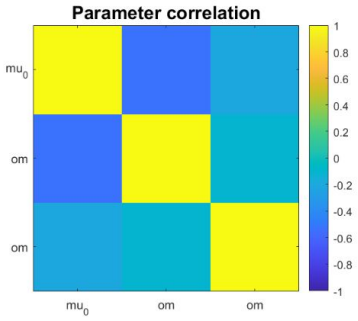
Question	ω_2 was kept fixed	κ_2 was kept fixed
2.3 a		
2.3d		

Table 1: Comparison of 2.3a and 2.3d

is for above random seed. We had run it for different random seeds and the following observation holds in the typical set. It may happen that for some particular initialisation, you get slightly different correlation plots. But in the typical sense you will get the following result. We had fixed the seed for reproducibility. When comparing the outputs of 2.3a and 2.3d , one observes that 2.3d gives that parameters have high posterior cross-correlation compared to 2.3a. This is because since we are basing the response function on the belief about the volatility at 3rd level, the response output becomes more strongly dependent on third level than before. So when doing the inverse HGF fitting, the estimated parameters have high cross correlation as there is more information flow across the levels (by definition of response function).