Modelling the Learning Trajectories of the Task Switching Paradigm with ANNs

Çam, Sebnem

del Pozo, Isabella

Dymov, Anita

Hovemann, Ilva

Obi, Vanessa

Introduction

What is the phenomenon you want to model? (0.5 points)

Our project wants to model learning trajectories while looking at error rates with congruent and incongruent trials, using a task-switching paradigm. In this task, participants react to the orientation or movement of triangles. At the beginning of each trial, is a display with a fixation cross. Participants must react to the triangles' orientation when the cross is yellow. If it is blue, they must react to the triangles' movement. Pressing the "F" button with their left hand indicates alignment or movement to the left. Pressing the "J" button with their right hand indicates alignment or movement to the right. In our project, we will model learning in the task-switching paradigm. Specifically, we want to examine the error rates with congruent (orientation=movement) vs incongruent (orientation!=movement) conditions. We will compare the models concerning their ability to account for learning trajectories. We predict that with ongoing learning the performance increases and the error rates get lower.

Why is that phenomenon relevant for understanding human cognition? (0.5 points)

Learning is a cognitive process. An individual assimilates knowledge such as behavioral, biomedical, or psycho-motor skills (Falk-Nilsson et al., 2002). Consequently, learning is part of human cognition and therefore a fundamental part of our cognition.

Methods

Why is this modeling method appropriate for understanding the phenomenon? (1 point)

A subfield of Machine Learning algorithms is artificial neural networks (ANNs). They try to imitate the way biological neurons propagate signals. One theory of how ANNs learn is that they change synaptic weights between neurons by learning rules.

In an unsupervised learning space, these weight changes depend solely on input and output. One approach is the Hebbian Learning Rule. It states that a change in a synaptic weight is proportional to the pre-synaptic input and the post-synaptic neuron's output (Oja, 2008).

Formally, this would mean that the weight parameter w_{mn}, connecting neurons m and n, can be changed to optimize the network's performance (Gerstner, Kistler, Naud, & Paninski, 2014)

Therefore, the Hebb rule is an unsupervised learning rule. In particular, it is considered a local unsupervised learning rule because the only dependent variables are the pre-and postsynaptic firing rates and the state of w_{mn}. This information is accessed at the synapse (Gerstner et al., 2014).

In a supervised learning space, weight changes also depend on target outputs. One approach to using supervised learning is backpropagation. The basic idea of the backpropagation algorithm is to minimize an error function in a weight space using gradient descent. Resulting in a specific combination of the network's weights minimizing the function (Rojas & Rojas, 1996).

The initially randomized weights are updated using the gradient of the error function. Therefore, gradients need to be computed recursively (Rojas & Rojas, 1996).

Even though it seems rather unintuitive at first, the backpropagation has two main features that are consistent with biological neural networks (Lillicrap, Santoro, Marris, Akerman, & Hinton, 2020).

1. Backpropagation is based on the synaptic plasticity mechanisms that underlie learning. Specific pre- and postsynaptic spiking behaviors influence "modifications between two neurons" (Lillicrap et al., 2020).

2. Feedback connections give the brain error information that induces synaptic changes (Lillicrap et al., 2020). "These feedback connections can take the form of direct 'top-down' cortico-cortical connections from higher to lower cortical processing areas, like those that exist between V2 and V1 within the visual system" (Lillicrap et al., 2020).

We decided to train our neural networks on two described learning approaches, back-propagation and Hebbian learning, implemented by Oja's Rule. Both approaches seem to have biological plausibility and differ in their implementation.

Consequently, modeling learning with these approaches seems appropriate to give us more insights into learning.

Which hypothesis/hypotheses do you seek to test by contrasting two (or more) models? (1 points)

In our model, we contrast two learning strategies.

Learning: Model 1, using back-propagation, will perform better than Model 2, using Hebbian Learning in terms of error rates.

Error rates will decrease with ongoing trials.

Orientation: Congruent trials have lower error rates than incongruent.

Movement: Congruent and incongruent trials yield the same error rates.

Description of computational model(s)

The two models vary in their learning approach. Therefore, the basic architecture is the same. We are using a Simple Neural Network. The Network consists of one input layer, a hidden layer, and an output layer.

What are the inputs, system properties, and outputs of your model(s)? (1 point)

The system is based on vectors. The input properties are three vectors of the form [0,1] or [1,0]. The first vector describes the orientation input. [0,1] represents an orientation of 180, [1,0] an orientation of 0. Accordingly, the second vector describes the movement. The third input vector denotes the task. [1,0] means task ”name the orientation” and [0,1] ”name the movement”. Between the input layer and the hidden layer are four connecting weights, projecting from the orientation input layer to the orientation hidden layer, from the movement input layer to the movement hidden layer, from the task layer to the orientation hidden layer, and from the task layer to the movement hidden layer.

The hidden layer processes the input through activation functions.

Two weights connect the hidden layers ”movement” and ”orientation” to an output layer.

The output layer uses the softmax activation function and outputs a vector of two probabilities, e.g. [0.57050197, 0.42949803], representing the probabilities of pressing ”F” or ”J”.

Both models apply a learning algorithm. They output the mean squared error, the learned weights, and the learning trajectories of the corresponding weights.

Which assumptions does each model make? (1 point)

Both models assume that learning happens through plasticity and changing synaptic weights.

The first model using backpropagation assumes that learning happens through error feedback and the weights can be changed using the error signal.

The second model using Hebbian Learning assumes that synaptic weight is proportional to the pre-synaptic input and the post-synaptic neuron’s output and the weights need to be changed accordingly.

Describe the computational implementation of each model (e.g., model formulas) (1 point)

Generally, both models share the same feed-forward pass. A feed-forward pass consists of a simple run through the network. In our case, this consists of computing three activation functions. We chose a logistic activation function for the activation of the hidden layer.

(1)

$$y_i = \text{logistic}(net_{y_j}) = \frac{1}{1+e^{-net_{y_j}}}$$

For the output layer activation, we used the softmax operation.

(2)

$$\sigma(x_i) = \frac{exp(x_i)}{\Sigma_k^N \, exp(x_k)}$$

For the backpropagation algorithm, the derivatives of the error function with respect to the corresponding weights are calculated to update the weights.

The following weights are being updated. Consequently, for each of the weights, a derivative of the error function with respect to each weight is calculated.

$$\Delta\mathbf{W}_{\text{output},h_{\text{orientation}}}$$

$$\Delta\mathbf{W}_{h_{\text{orientation}},input_{\text{orientation}}}$$

$$\Delta\mathbf{W}_{\text{output},h_{\text{movement}}}$$

$$\Delta\mathbf{W}_{h_{\text{orientation}},input_{\text{movement}}}$$

Here is one concrete application of the chain rule in the context of calculating the derivative of the error function with respect to the weights of the input to the hidden layer:

(3)

$$-\Delta\mathbf{W} = \frac{\partial\text{Error}}{\partial\mathbf{W}_{h_{\text{orientation}},\text{input}_{\text{orientation}}}}$$

$$= \frac{\partial\text{Error}}{\partial act_{\text{output}}} \cdot \frac{\partial act_{\text{output}}}{\partial net_{\text{orientation}}} \cdot \frac{\partial net_{\text{output}}}{\partial act_{h_{\text{orientation}}}} \cdot \frac{\partial act_{h_{\text{orientation}}}}{\partial net_{h_{\text{orientation}}}} \cdot \frac{\partial\text{net}}{\partial\mathbf{W}_{h_{\text{orientation}},\text{input}_{\text{orientation}}}}$$

To implement the Hebbian Learning Algorithm, we decided to use Oja’s Rule. To bound weights Oja used a normalized Hebb’s Rule, leading us to the following equation:

(4)

$$w = w + rate * y * (input - y * w)$$

w denotes the weight

rate denotes the learning rate

y denotes the firing activity

input denotes the input properties

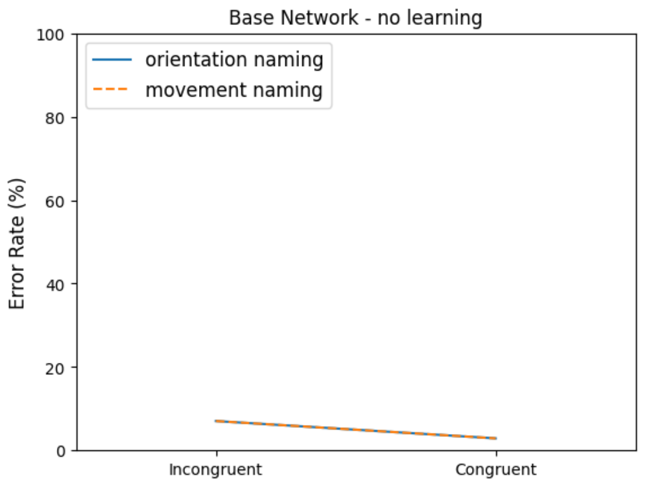

Figure 1: Plot of the simulated data from the base network

Description of the experiment

In the experiment, participants react to the orientation or movement of triangles. At the beginning of each trial, is a display with a fixation cross. Participants must react to the triangles' orientation when the cross is yellow. If it is blue, they must react to the triangles' movement. Pressing the "F" button with their left hand indicates alignment or movement to the left. Pressing the "J" button with their right hand indicates alignment or movement to the right.

Provide an overview of the experiment. What are the independent variables and dependent variables of the experiment? (0.5 points)

Independent variable causes an effect on the dependent variable. Therefore, orientation and movement are independent variables that affect the dependent variable error rates.

How much data was collected (number of participants and trials)? (0.5 points)

Our data consists of 6 participants with each 97 trials, resulting in 582 trials.

Model simulation

Describe the process of simulating data from the model(s). (1 point)

To begin with the experimental conditions were defined. If the triangle points in the same direction as the movement of the triangle, then the trial is congruent (e.g. the triangle points to the right while simultaneously moving to the right). If this is not the case, the trial is incongruent. Additionally, a separation by task type was made. This means we had four conditions: Congruent orientation naming task, incongruent orientation naming task, congruent movement naming task and incongruent movement naming task.

The trials were simulated by a forward pass of the network, which returned the probability with which the model would choose "Left" or "Right" for the given condition. Based on these results the error rates of the model for each condition was calculated and returned.

Before training the network with the different learning algorithms, we simulated data from the base model and plotted the results.

Then we generated training data for the experiment and used it for the training of the models. The network was trained once with Backpropagation and once with Oja's rule, using a learning rate of 0.001. The simulated data was plot once with error rates after each epoch of training, showing the learning trajectory of the model, and once with only the error rates from the completely trained network.

Model fitting

Describe the process of fitting the model(s) to the data. Remember to describe any preprocessing steps of the data. (2 points)

We began by choosing the parameters we wanted to fit. The nature of our model does not leave us a lot of free parameters to pick from, since the weights, as well as biases in the case of backpropagation, are being determined by the given learning algorithm and Oja's Rule assumes that the biases remain at value 1. This left us with the learning rate as our sole parameter needing fitting.

In order to fit the learning rate efficiently, we determined a suitable search space of plausible values. Our search space contains 0.001, 0.01, and 0.1 as possible learning rates as they are the most widely used.

Finally, we used maximum likelihood estimation to evaluate the model fit. The log-likelihood of our model is calculated by adding up the log of the predicted probabilities for the target key presses which we get by subtracting the probability of a false key press from 1.

Parameter recovery

Describe how you performed parameter recovery for your models. (1 points)

For parameter recovery, we set up a space of potential learning rates we want to recover. We set them up between 0.001 and 0.1, to make them similar to the space of possible values for fitting the model. We then randomly select 20 learning rates. For each of these learning rates we first train the model with surrogate data and then simulate the experiment. We then get responses which we use to fit the respective model trying to recover the true learning rates of the models. We do this for both the Backpropagation and the Hebbian learning model. We then evaluate our recovery by comparing the true learning rates and the fitted learning rate by calculating the Pearson Correlation.

Model comparison (& recovery)

Describe how you compared the models. (1 point)

To compare the models, we followed a systematic process. First, we began by preparing the data, selecting relevant columns which are the correct, task type, coherent orientation and coherent movement direction column and performing one-hot encoding for categorical variables as the task type. We then combined some features into a single column (the task type movement and the task type orientation) and encoded others based on their values.

Next, we split the preprocessed data into training and testing sets. For model fitting, we employed two approaches: backpropagation and Oja's rule. Both models were trained using the same training data with 100 epochs and a learning rate of 0.01.

After training, we made predictions on the test set using both models. The predictions were compared against the actual labels, and accuracy scores were calculated for each model.

To quantify the complexity of each model, we calculated the total number of parameters involved in the neural network architecture.

Finally, to visualize the performance of each model, we generated confusion matrices for both backpropagation and Oja's rule models. These matrices provide insights into the model's ability to correctly classify instances.

By comparing accuracy scores, model complexities, and examining confusion matrices, we assessed the performance and characteristics of each model to determine their relative effectiveness for the given task.

Optional: Describe how you performed model recovery. (0.5 bonus points)

Initially, two models are defined for comparison: backpropagation and Oja's rule. Then, a range of true parameter values (learning rates) is established, utilized to simulate data from both models. Subsequently, a confusion matrix is initialized to store the results of the model comparison, indicating which model performs better for each combination of true parameter values. Nested loops iterate over the true parameter values. For each combination, data is simulated from both models based on the true parameter values. Each model is then trained on the simulated data, and the mean squared error (MSE) of their fits to the data is computed. The fits of each model are compared based on their MSE values, and the confusion matrix is updated accordingly. If the MSE of the backpropagation model is lower, it is deemed better, and the corresponding entry in the confusion matrix is set to 0. Otherwise, the entry is set to 1, indicating that the Oja's rule model is superior. Lastly, the confusion matrix is printed, displaying which model prevails for each combination of true parameter values. Optionally, the confusion matrix can be visually represented as a heatmap using Matplotlib, offering a graphical depiction of the model comparison results. In essence, model recovery facilitates the assessment of different models' performance across various parameter settings and aids in identifying the model that most accurately describes the observed data.

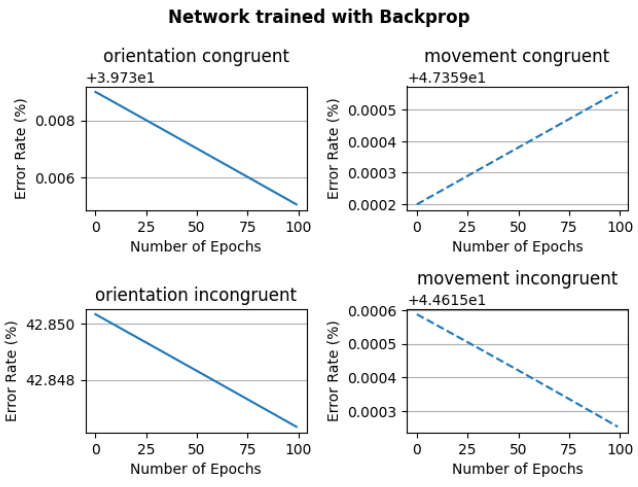

Figure 2: Plot of the simulated data from the network trained with backpropagation showing the learning trajectory of the model over the training epochs.

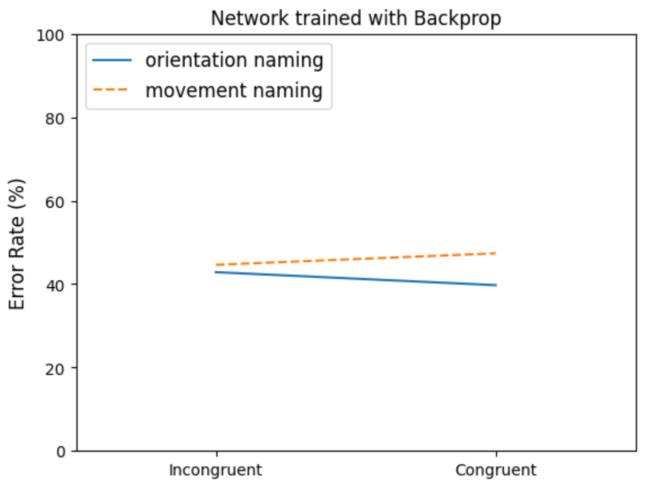

Figure 3: Plot of the simulated data from the network trained with backpropagation showing the error rates of the final weights.

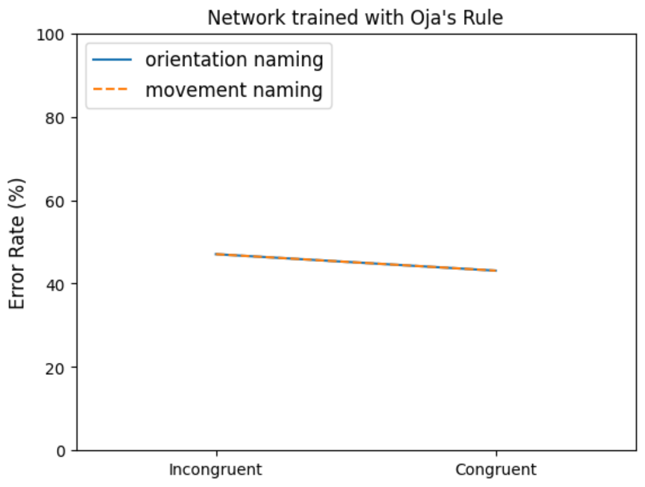

Figure 4: Plot of the simulated data from the network trained with Hebbian Learning (Oja's Rule) showing the error rates of the final weights.

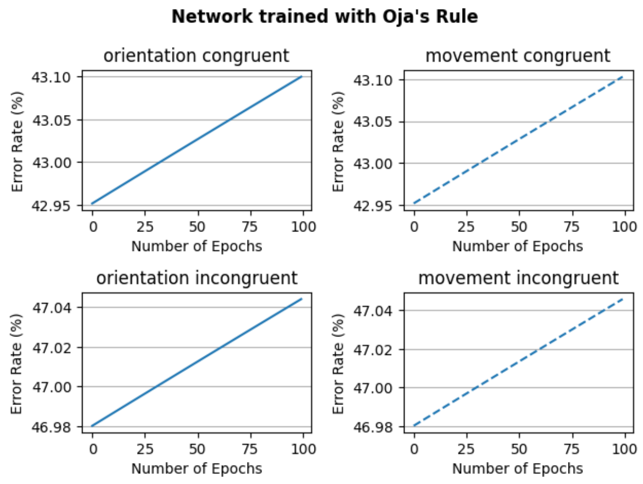

Figure 5: Plot of the simulated data from the network trained with Hebbian Learning (Oja's Rule) showing the learning trajectory of the model over the training epochs.

Results

Simulation results

Which phenomena do the models capture and why? Make sure to support your argument with a plot. (1 point)

The base model (without any learning) showed higher error rates for incongruent trials (7.05%) than for congruent error rates (2.89%) as seen in Figure 1.

This phenomena could also be observed in the final error rates of the network trained with Oja's Rule (see Figure 4), as well as in the final error rates for the orientation naming task when the network was trained with backpropagation (see Figure 3). Here congruent orientation naming trials had an error rate of 39.74% and incongruent orientation naming trials an error rate of 42.84%.

The models after training with backpropagation and Oja's Rule did capture a learning trajectory, meaning the weights of the model adapted over time. The network trained with backpropagation even decreased its error rate over the training epochs in three out of the four conditions (see Figure 2).

Which phenomena do the models not capture and why? (1 point)

The models show very inconsistent results when running the model multiple times. Sometimes they support our hypothesis, that incongruent trials should yield higher error rates, but often they do not.

Additionally, the error rates did not change very strongly and always stayed close around 50%, meaning the learning trajectory is minimal, within the tenth to hundredth decimal space, and the networks frequently became worse during training, especially when running the code several times (see Figure 5 for an example).

This might be due to the low error rate chosen for the data simulation, which we could test during model fitting through choosing higher learning rates (see Model fitting Section).

Additionally our networks might not be big enough, as too few neurons in a layer can restrict the representation a network learns, which causes under-fitting. Our networks could perhaps show better learning trajectories if a few more hidden layers were added.

Parameter recovery

Which parameters can be recovered more reliably, which less reliably? (1 point)

For both models, the Pearson correlation between true learning rate and fitted learning rate is very close to 0 ($\approx -1 \cdot 10^{-16}$). Thus, there is no strong correlation, which means that for both models the learning rate is not reliably recoverable. Thus, we cannot make any inferences about the underlying dynamics of the experiment and participants' behavior. For instance, we cannot assume that there are different learning rates for different participants.

Optional: Model recovery

Which models can be recovered more reliably, which less reliably? (0.5 bonus points)

Each entry in the matrix denotes which model performed better for a specific combination, with 0 representing backpropagation as the superior model and 1 indicating Oja's rule. Upon analyzing the matrix, several patterns emerge. Backpropagation appears to be favored over Oja's rule for many parameter combinations, as indicated by the prevalence of 0s in the matrix. This suggests that backpropagation tends to provide better fits to the simulated data across a wide range of parameter values. However, there are instances where Oja's rule outperforms backpropagation, as evidenced by the presence of 1s in the matrix. These cases indicate parameter combinations where Oja's rule yields lower mean squared error (MSE) and thus is considered the better model for capturing the simulated data. Overall, backpropagation seems to be recovered more reliably than Oja's rule, given its more frequent appearance as the superior model across different parameter settings. This suggests that backpropagation is generally more robust and versatile in fitting the simulated data compared to Oja's rule. On the other hand, Oja's rule appears to be recovered less reliably, as it is less consistently favored across parameter combinations. This implies that Oja's rule might have specific limitations or dependencies on certain parameter values, leading to its variable performance relative to backpropagation. In summary, while both models demonstrate their strengths in certain scenarios, backpropagation emerges as the more reliable choice overall based on its more consistent performance across a broader range of parameter values. Oja's rule, while capable of providing competitive fits in specific cases, appears to exhibit less robustness and consistency compared to backpropagation in this model recovery analysis.

Model comparison

Which models fit the data better and why? (1 points)

Both backpropagation and Oja's Rule achieved comparable accuracies of 71 percent when fitting the data. The loss values, which indicate the discrepancy between the predicted and actual values, decreased consistently for both models as the epochs progressed. However, the decrease in loss was more pronounced for Oja's Rule, with its average loss decreasing from 0.00065174 to 0.00045147, compared to backpropagation's decrease from 0.00065325 to 0.00045325. This suggests that Oja's Rule may have adapted more efficiently to the data over the course of training.

Considering model complexity, both models have the same complexity level of 16. This implies that they have a similar capacity to capture the underlying patterns in the data.

Overall, both models fit the data well, achieving the same level of accuracy and having comparable complexities. However, Oja's Rule demonstrated a slightly more substantial improvement in loss over the training epochs, indicating potentially more efficient learning.

Parameter fit

Describe the results of fitting the winning model to the data.

Which parameter values fit the data best? (1 point)

Although there is a difference in the progression of error rates over, the best fitting learning rate value as well as the resulting model accuracy are the same in both of our presented models, namely 0.001 and 71 percent, respectively. While it is inconsistent with the equality of accuracies in both models, the error rates of the hebbian network seem to decrease in for all conditions except congruent movement, while the opposite happens in the model learning through backpropagation. It should be noted that the error rates for backpropagation already started out very small and all changes in error rates are minimal.

Discussion

Which hypothesis does your modeling support and why?. Base your answer on the model comparison (and model recovery) results. (1 point)

Our modeling does not specifically support our hypothesis that a model using Backpropagation will perform better than a model using Hebbian Learning. This might be because our neural network is rather small, as it has only 16 weights and

Which other insights does your model provide? Base your answer on the parameters fits of the winning model. (1 point)

Due to the high inconsistency in results paired with, what we suspect to be, joint runtime issues that we faced with our models, it is hard to determine any more further insights since both models end up performing the exact same.

What are potential weaknesses of your modelling study? (0.5 points)

A potential weakness of our modelling study might be the chosen source code editor. As discussed in the previous section, the models ended up with the exact same error rates for the final weights (after training), which we suspected to be caused by joint runtime issues in google colab. We circumvented this issue during the Model simulation by disconnecting the runtime and restarting the session between the training of the two models. However, this solution could not be used in the other sections of our code due to their specific program structure. This is why the results of the Model simulation and Model comparison are slightly different.

Another potential weakness could be the fact that backpropagation is supervised. Humans do not learn in supervised learning settings where they have true answers. Consequently, Hebbian Learning is closer to human behavior.

This also means that running backpropagation against Hebbian Learning is not fair, since backpropagation will always outperform Hebbian Learning.

However, the biggest weakness might be the underlying structure of both models, due to restrictions posed by our expertise level. Since we could not make our model too big or complex, our networks cannot capture the underlying dynamics of the phenomenon and is rather inconsistent in it's performance. Furthermore, the inability to recover the learning rate might also hint at our model's limitation to describe and explain the system.

What might be another computational modeling approach for gaining a deeper understanding of the phenomenon? (0.5 points)

In order to gain a deeper understanding of the phenomenon of learning, we should look at different ways humans learn. One way we learn is through reinforcement. Therefore, one way to approach this modeling task could be by using reinforcement learning as a learning algorithm. It might be better to use a reinforcement algorithm to train the model, as it might reproduce a closer fit to the learning trajectory of humans performing the task-switching experiment. Such an algorithm might explain learning in the experiment better than supervised or unsupervised learning.

Acknowledgements

List which group members have been responsible for which part of the group projects. E.g.,

- Sebnem Çam: Methods (Parameter Fitting); Results (Parameter Fit); Discussion

- Isabella del Pozo: Methods (Model Simulation); Results (Simulation Results); Discussion

- Anita Dymov: Methods (Model comparison (and recovery)); Results (Model recovery and model comparison); Discussion

- Ilva Hovemann: Introduction; Methods (first two questions); Methods (Description of computational model(s)); Methods (Description of the experiment); Methods (Model Implementation); Discussion

- Vanessa Obi: Methods (Parameter Recovery); Results (Parameter Recovery); Discussion

References

Falk-Nilsson, E., Walmsley, D., Brennan, M., Fournier, D. M., Junfin Glass, B., Haden, K., ... Petersson, K. (2002). 1.2 cognition and learning. *European Journal of Dental Education*, *6*, 27–32.

Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, *21*(6), 335–346.

Oja, E. (2008). Oja learning rule. *Scholarpedia*, *3*(3), 3612.

Rojas, R., & Rojas, R. (1996). The backpropagation algorithm. *Neural networks: a systematic introduction*, 149–182.