**Cognitive Science Report: Recategorization**

**Introduction:**

This paper is a technical report for a model that uses a specific learning algorithm (hierarchical, precision-weighted prediction error learning) in a non-monotonic learning task (re-categorization). The goal of this paper is to demonstrate 1) the effectiveness of this sort of model in non-monotonic learning, and 2) to demonstrate how this model might be useful for understanding individual differences in learning.

**Task:**

The task used for this exercise was the “re-categorization task”. The re-categorization task is a simple category learning task, in which participants must induce the correct definition of a category by observing a collection of stimuli which may or may not belong to the category in question. Possessing or lacking a specific feature defines category membership. After a set period of time (ideally, long enough for the subjects to have correctly inferred the original category) the category-defining feature is changed.

In this particular version of the task, each stimulus consisted of a vector of 6 binary-valued features. Each stimulus could belong to one of two categories (simply labeled category 1 or category 2). Simulated “participants” viewed each stimulus, and then responded as to which category the stimulus belonged to. After making their guess, participants received feedback indicating the stimulus’ actual category. After 50 such trials the defining feature was switched. Before the switch category membership was defined by having the value of the 1st feature in the stimulus vector equal to “1”, after switching the category was defined by the 2nd value in the stimulus vector equaling “1” (see line 300 in the code for “sample\_recat\_experiment\_run” file).

When using human subjects, this task has a “cover story”, intended to (partially) conceal the true nature of the experiment. In the past this cover story has been that the subject is an interplanetary biologist learning to classify recently discovered martian bacteria as either oxygen resistant or not. Each bacteria has six binary features which may impart oxygen resistance (e.g. headbulbs vs. no headbulbs, bent vs. straight ribosomes, etc.). I mention this because, when describing the model behavior it might be more useful to use terms like “has tail cilia” and “is oxygen resistant” rather than “feature 1 has value “present” “ and “is in category 1”.

*Running the model: define misconception, target, switch- model runs, parameters and learning rate*

*Model output*

**Model**

The model consists of two major components, a performance component, responsible for selecting the best category for a given trial, and a learning component, responsible for learning from feedback after each trial.

Learning

General

*General framework*

Layers: stimulus/input layer, encodes the features/values

Links to the category label

Feedforward during performance

Competition/selection during performance

See Image

*Learning: Hierarchical precision weighted learning*

Link strength corresponds to the estimated co-occurrence frequency between the value and the category label.

Higher order estimates/dynamic parameters

Variability

volatility

See Image

*Note: Comparison to previous models*

Previous versions of this model used a slightly different update method. The general architecture of input/feature nodes linked to category definitions and updated through a Bayesian reward function was the same, but the nodes themselves encoded the stimuli in a slightly different manner. In previous versions of the model, the input nodes corresponded to binary features, whereas in the current model each node corresponded to the *value* of a particular feature. For example, in the earlier model the feature “ribosomes” was encoded by a single node with two (mutually exclusive) values, arbitrarily assigned “1” or “0”; “1” for straight ribosomes, and “0” for bent ribosomes. In this set-up, observing and updating one value of a stimulus has automatic consequences for the weight assigned to the other values at that feature (e.g. decrementing the weight of “bent ribosomes” after a failed categorization instance, implies a relative increase in the strength of straight ribosomes for the category definition) . I

In the current model, since each value had its own node, changes in the weight of one value on a feature did not directly effect the changes in weights for other values at that feature. For example, the feature “ribosomes” was represented by two nodes (“bent node”, “straight node”), each of which updated separately, each node could take a 0 or 1 value indicating whether or not it was present in the stimulus. Note that this set-up does not imply that feature-values are, from the point of view of the model/learner, mutually exclusive (even though the structure of the task is such that any two values of a single feature are, in fact, mutually exclusive). (This change allowed the model to update only the weights of the value that was actually observed, a more psychologically plausible assumption).

Changing the model structure like this had some unexpected further implications for the updating rules. While the core update function (the Bayesian-reward learning element) remained unchanged, the interpretation of feedback had to change. In the original model, upon seeing what the correct category was, the model would adjust the weights of all values for the *correct* category definition. That is, if the model had guessed “oxygen resistant”, but the correct category was “oxygen intolerant” then the model would adjust the weights for the features mapping to “oxygen intolerant”. In the new model, since only single values (and not total features) were updated, this type of interpretation of feedback would mean that no value would ever be decremented, leading all value+category links to slowly plateau at the maximum weights (this did not happen in the original model because incrementing one value meant decrementing its complement).

In the new model, on a given trial, feedback was interpreted relative to the category “guessed” (selected) by the model for that trial. In this case, the weights of the category-selected+value links were updated (either incremented or decremented, whether the category was correct, or not) not the weights of the actual-category+value links. (This meant that feedback could be used as a generic error signal: If the actual-category matched the expected category then the link strength between the expected category and the values viewed would be increased, while if the actual stimulus category did not match the expected category, then, the links for the expected category would be decreased).

Feature Comparison

**Model Output**

When run, the model outputs a list containing the updated (dynamic) parameters for each value-category link at each trial and a list of the model’s overall success or failure on each trial. These parameters include the estimate as to whether the given value is part of the category definition (category-link strength), the prediction error between the model’s previous estimate of the category-value strength and the most recent feedback received, the precision weighted prediction error, variability, subjective uncertainty on variability, and environmental volatility. In the results section, I focus primarily on interpreting and explaining the model’s overall success (and failure), as well as the meaning of the category link strength, variability, and volatility estimates for the target and misconception values.

[note value types: 0 – 1, natural number, etc.: Greater than \_\_ means blank, less than means \_\_\_\_)

After an experimental run the model produces an output file entitled test\_file.txt, which contains the dynamic parameters for each category+value link. These are formatted as a ‘parameter x trial’ csv. When the model is run using IDLE it outputs a list of trial successes (1 if the model’s prediction matched the trial output, 0 if it did not) through the IDLE interface. To present the results sensibly, I cut and pasted the CSV output and the IDLE output for each of the two different model runs into separate sheets on an excel spreadsheet, and labeled them appropriately. (Note on python code and output files: when running the model, the test\_file.txt opens for “appending” and not “writing” which means that running successive versions of the model will add text to the existing “test\_file” and not overwrite it or save a new version).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Belongs to Cat? (estimate)** | **prediction error first level** | **precision weighted prediction error 1st level** | **variability estimate** | **uncertainty on variability** | **volatility estimate** |

**Running the Model**

**Variations in the model**

The model has three static parameters, in the form of Bayesian priors. These priors roughly correspond to priors on variability (denoted in the model code as omega), volatility (denoted in the model as sigma\_3), and how tightly changes in (estimated) variability and volatility influence one another (denoted as “kappa” in the model). Changing these parameters alters the learning behavior of model. It might be argued that the values of these parameters correspond to (yet-to-be-specified) individual differences. If this is the case, running the model with different parameter settings corresponds to simulating the behavior of different types of learners. (For example, supposing the kappa parameter represented “cognitive flexibility” changing its value would produce different simulated learners who differed in how flexible they were. The model currently remains agnostic as to which specific individual differences these parameters represent. I plan to explore this at a later point using parameter fitting methods, and comparing the best-fit parameters to empirical individual difference data).

In version One of the model, these three parameters were set using empirical estimates from Iglesias et. al (2013)[[1]](#footnote-1).

**Results**

*General*

*Comparing between different learners (models with different initial parameter settings)*

*Comparison to earlier mode*

**# possible error in the category selection function?**

Iglesias, S., Mathys, C., Brodersen, K. H., Kasper, L., Piccirelli, M., den Ouden, H. E., & Stephan, K. E. (2013). Hierarchical prediction errors in midbrain and basal forebrain during sensory learning. *Neuron*, *80*(2), 519-530.

1. The Iglesias experiments used a simple cue-response paradigm ( a single stimulus, varying along a single binary dimension probabilistically cued the presentation of a sing [↑](#footnote-ref-1)