Radboud University Nijmegen

Department of Psychology and Artificial Intelligence

The Development of Bayesian generative models

Comparing model reduction and model construction

Bachelor Thesis in Artificial Intelligence by

Francesca Drummer

s4770099

Supervised by

Johan Kwisthout[1]
Danaja Rutar[1]

[1]Donders Institute for Brain, Cognition and Behaviour

January 31, 2020

Contents

Abstract

The predictive processing (PP) framework is one of the leading theories to explain cognition. According to PP, the brain continuously predicts sensory inputs given its generative model of the world. An interesting question is how such a generative model of the world is developed. Two approaches for developing a generative model are model updating and model revision. Model updating refers to updating the probabilities over the hypothesis in the model. Compared to that, model revision can take place by constructing or reducing a generative model. While most research focuses on the model updating, in this Bachelor thesis, we will investigate the development of a Bayesian model by model revision. Particularly interesting is the question, how development of a generative model compares in terms of accuracy and causal relations between the two existing model revision processes. Model reduction and model construction are compared by using a computer simulation. In general, neither of the approaches converge to the 'true' model of the environment. However, both approaches developed a model that captures the association rules of the environment. Despite showing that model development can underlie the process of model reduction, as well as model construction, more research in complex areas is necessary to generalize these findings.

1 Introduction

Over the past years, the predictive processing (PP) framework gained an increasing amount of attention. In general, PP is a computational level theory that offers a unifying perspective on action and perception. According to this unification theory, the brain works as a prediction machine, which suggests that actions are both predicted and perceived (Friston, 2003). Because the brain predicts actions in a top-down process, it implies that processing sensory inputs does not exclusively occur in a bottom-up fashion. Rather the brain continuously predicts future sensory evidence based on a generative causal model that represents the world. The mismatch between the predicted future sensory information by the model, and the actual sensory observation is called prediction error (PE) (Kwisthout et al., 2016). After generating the prediction error, it is processed in an upward fashion through the hierarchical generative model. The aim is to minimise future prediction errors by adjusting the model accordingly (Clark, 2013). Minimising the prediction error is coupled with the free-energy principle. The free energy principle defines that any agent must minimise the entropy of its sensory interaction with the world. In the context of predictive processing, entropy defines the uncertainty in predictions. To resolve uncertainty about the world states, the agent must maximise the accuracy of its generative model (Friston et al., 2017). Thus, an agent's generative model must be adjusted such that uncertainty about the environment is resolved and in turn, the accuracy of predictions increases.

The predictive processing literature proposes different ideas about how the brain can minimise prediction errors. Two examples of these propositions are *model updating* and *model revision*. Firstly, *model updating* refers to changing the prior probabilities over the different hypothesis of the model (Kwisthout et al., 2016). Model updating is the more popular approach in the predictive processing field to explain model learning. However, it offers limited possibilities for change, because it can not cause structural revision of the model. Instead, model updating focuses on testing current hypothesis (Perfors, 2012). Nevertheless, in some situations a structural change of the model is required, for example, if the environment is dynamic or the model misrepresents knowledge. In these situations, restricting the change in the model by solely updating the hypothesis is not sufficient (Kwisthout et al., 2016). More specifically, imagine the following example of language learning: Assume an established association between word 'A' and meaning 'M1'. Now consider moving to a different country where the word 'A' relates to a different meaning 'M2'. Integrating this new knowledge

requires adding a variable, connection or value to expand the possible meanings of word 'A'. Thus, a structural change of the model is necessary. The latter approach, *model revision*, can account for such a structural change by adding or removing hypothesis and causal relation between variables to the model (Kwisthout et al., 2016).

Both methods, model updating and model revision, are applied to the generative models of agents. *Generative models* aim to capture the underlying statistical nature of observed inputs. Particular hypothesised causes of the observations help to predict the sensory input. The more accurate the model represents the statistical nature of the input data, i.e. environment, the better the prediction it makes, which increases the posterior probability of the model (Clark, 2013). In this Bachelor thesis, the generative models of the agents are *causal Bayesian networks* modelled using categorical probability distributions as described by Kwisthout, Bekkering and Rooij (Kwisthout et al., 2016). The advantage of causal Bayesian networks is that the stochastic causal relationships between variables can easily be represented (Kwisthout et al., 2016). Besides, that Bayesian models can encode an agents prior belief about the environment and use the probability distribution of the hyperpriors to define how confident the agent is about these beliefs (Otworowska et al., 2018).

In contrast to the commonly used model updating approach, the focus will lie on model revision. Model revision is a umbrella term for *model reduction* and *model construction* which both enable structural changes in the model. However, they differ in their initial models and revision methods. *Model reduction* starts with a high complexity model which integrates the assumption that every variable is *dependent* on each other ('*dependency prior*'). On the other hand, the beginning model for *model construction* is of low complexity where each variable is *independent* of each other ('*independency prior*'). Given these two different initial models, an interesting question to consider is how the states and causal relations of an agent's generative model come about during development. A large research area of development in cognitive science presumes abductive reasoning in the behaviour of children. Problematic is that abductive reasoning assumes that the agent already knows the structure of the model itself (Friston et al., 2017). Therefore, e.g. Friston et al., considers development itself as a process of model reduction over existing accumulated sensory evidence. In contrast to this common assumption, we want to argue that it is also plausible for children to be born with a generative model of low complexity that does not include prior beliefs about the models structure yet. Therefore, instead of only focusing on explaining the process of development with model reduction we hypothesise that the process of model construction is also relevant.

With that in mind, the goal of this paper is to compare the model development of two agents, where one is performing model reduction and the other model construction. The aim is, to investigate how these two approaches that initially assume opposing relations between variables ('dependency' vs 'independency prior') can integrate association rules from a specific environment in their model. We expect to show that via *both* model revision approaches, model construction or model reduction, it is possible to minimise the prediction error of the agent. After all, model revision is one of the different ways to deal with prediction errors as described by Kwisthout et al. (2016). When the prediction error decreases, then the generative model of the agent must represent (some of) the rules underlying the environment. Because in a stochastic environment one can not guarantee that the agent will encounter all possible states, we hypothesise that the generative models do not necessarily need to converge to a 'true' model of the environment. Instead, different models can be accurate representations of an environment, in the sense that they minimise their prediction error. Thus, due to the random nature of the environment, one agent might encounter different states than another agent. Finally, it is likely that their resulting generative models describe parts of the environment equally accurate. However, it is highly unlikely that each model also describes the same environmental components. In addition to the stochastic nature of the environment, the environment

also has some degree of uncertainty in observations. Thus, the brain might not be able to establish a one-on-one mapping between causes and observations (Gładziejewski, 2015). The above-stated hypotheses that both model revision approaches will settle to an accurate but not necessarily 'true' generative model will be tested with the following research question:

How do the final generative models developed under model reduction or model construction compare in terms of dependency relationships and accuracy[1] *?*

The remainder of this section, describes the principles and workings behind model updating and model revision.

1.1 Model Updating

Model updating is one possible approach for adjusting an agent s generative model without the structure of the model itself. Rather, it focuses on updating the hyperpriors of the agent. Hyperpriors represent initial beliefs about the environment. These beliefs are updated over time to include the observations in the environment. In that sense, model updating is the same as updating the probability distributions of the prior beliefs (Kwisthout et al., 2016).

The magnitude of the model updating step depends on the size of the prediction error. Nevertheless, a high prediction error can have different causes, and thus should not always generate a probability updating step. One reason for a non-meaningful prediction error might be caused by a non-deterministic environment. Therefore, it is crucial to distinguish prediction errors caused by misrepresentation of the environment or observation of an outlier. Given the first case, the model does not represent the environment correctly, e.g. because environmental properties changed, and therefore it is appropriate to update the model. On the other hand, updating the model should be avoided if the observed mismatch was random (Kwisthout et al., 2016). For example, consider observing that a person wears a scarf in summer. Such an observation will cause a high prediction error because based on initial beliefs, people do not wear a scarf in summer. In this context, the observation should not induce a high increase of the probability that people wear scarfs in summer because the observation is an outlier. However, if the season changed and now it is winter, then, due to the change in the environment, there should be a model updating step taking place. The new probability distribution should predict that people are more likely to wear scarfs when it is winter.

The example, shows that model updating should ideally only occur under certain conditions. However, this Bachelor thesis does not focus on exploring the conditions for model updating. Instead, the focus will lie on model revision. Therefore, there are no constraints on updating the model in this Bachelor thesis because the goal is to evaluate model revision.

1.2 Model Revision

Model revision describes how a generative model can be changed in structure by adding or removing variables, increasing or decreasing the number of variables in the model or by updating the conditional probability distributions (Kwisthout et al., 2016). The first two examples correspond to a change in the model in terms of its model structure. These aspects of model revision still need to be investigated in the predictive processing context, thus, adding novel variables and values to a model space will not be considered (Kwisthout and Rooij, 2019). In this work, the definition of model revision is restricted to adding or removing causal relationships between variables.

[1] Accuracy refers to the size of the prediction error, e.g. a low prediction error implies high accuracy.

As described in the previous section, model updating should only be performed if it is certain that the prediction error did not occur by chance, but is caused by an environmental change. Similarly, it is essential to only change the structure of the generative model, if there exists a definite causal relation originating from the observed pattern of prediction errors. Therefore, a structural change of the agent's model should only be applied if it would cause an overall increase of prediction accuracy in the model (Smith et al., 2019).

In the context of the PP theory, the specific conditions that should induce a revision of the model remain an open question. The example, mentioned earlier, only focuses on the situation when a model change occurs because of a high prediction error. However, there do exist other possible causes for model change as described by Rutar et al. (ress). The authors for example discuss that multiple small prediction errors could also cause a model change, because when combined they might elicit a high prediction error. Despite the many different possibilities for model change, the focus here lies on model revision caused by a high prediction error.

1.2.1 Model Reduction

As introduced above, a common assumption in developmental cognitive science is that model reduction is considered the main process to describe development. Model reduction is applied to a full generative model that has causal relations between all variables ('dependency prior'). The goal of model reduction is to reduce causal relations between nodes that are not important, to develop a new and less complex model. The reduced model is selected if it represents the evidence in the environment better than the full model. Because the process of model reduction follows a top-down approach that reduces a full model to a simpler model with less ambiguity, it has an abductive reasoning aspect (Friston et al., 2017).

Figure 1 illustrates a simple example of Bayesian model reduction. The figures shows how three different weather conditions ('Rainy', 'Cloudy' and 'Sunny') can help to predict whether a person will use an umbrella or not. Figure 1a shows the full model where each weather node has a causal relation to both prediction nodes ('Umbrella' or 'No umbrella'). After exposing the full model to training data, the model could reduce that, e.g. if it is sunny, a person would take no umbrella. Therefore, the final model does not include causal relations that add no information about the connection between the weather state, and if a person is wearing an umbrella or not. In Figure 1b one possible reduced model is shown. In that reduced model, only the causal relation between 'Sunny' and 'Umbrella' is removed. A possible explanation why both causal relations between 'Cloudy' to 'Umbrella' and 'No umbrella' still exist is because when it is cloudy, the model observed no significant difference between people wearing and umbrella or not. Differently, one would expect that most people do take an umbrella when it is raining. However, the causal relation between 'Rainy' and 'No umbrella' might still exist, because in the last days it has not rained a lot and therefore no relation could be identified.

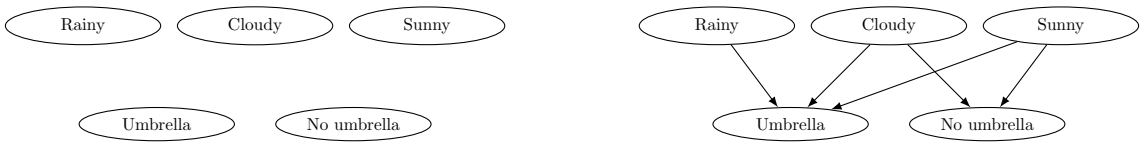
(a) Full model. (b) Possible reduced model.

Figure 1: Full (a) and possible reduced model (b) of a causal Bayesian network.

1.2.2 Model Construction

The process of model construction can be described as a bottom-up approach to develop more complex models. Therefore, model construction contrasts the process of model reduction. Until now, most research on developmental cognitive science evades essential aspects of the model construction process. Thus, challenging questions, such as how it is possible to develop models in a bottom-up fashion by adding novel variables (Friston et al., 2017) remain answered. Moreover, other researches conflict the opinion that adding and creating new variables is a form of realistic learning. For instance, Perfors (2012) argues that every learner (whether human or computer) has an initial generative model that already contains all possible states in the *implicit hypothesis space*. According to Perfors the process of learning does solely involve (a) moving hypothesis from the implicit to *explicit hypothesis space* (*hypothesis generation*) and (b) manipulating hypothesis, e.g. by updating their probabilities, in the explicit hypothesis space (*hypothesis testing*). Thus, Perfors excludes the possibility of adding new states into the implicit hypothesis state altogether (Perfors, 2012). The questions imposed by Perfors are strongly connected to questions about the innateness of one‘s initial model space. Because these questions concentrate on evolutionary aspects instead of model development, we will not focus on them in this Bachelor thesis. However, the until now unanswered question concerning how to add new values to a models state space does restrict the implementation possibilities of model construction.

In order to implement learning as a process of model construction, the definition of construction is limited in this work to adding causal relations. Considering the same weather example from above Figure 2 shows how model construction can occur. This time, the initial model contains all nodes representing the world state, however, without dependencies, or causal relations, between these nodes (Figure 2a). After observation of relevant data, newly discovered dependencies were added to the model by introducing causal relations between nodes (Figure 2b).



(a) Empty model.

(b) Possible expanded model.

Figure 2: Empty (a) and possible expanded model (b) of a causal Bayesian network.

In both model revision approaches, establishing causal relations between nodes can be explained via Hebb's rule: "Cells that fire together, wire together" (Donald O. Hebb) that was proposed by Donald Hebb in 1949 (Hebb, 1949). The principle explains one way of organizing new experiences in the brain in an unsupervised manner. For this, consider the language learning example from above again. In this example word 'A' would be increasingly often observed in combination with meaning 'M2' after moving to a different country. Thus, more neurons encoding the meaning 'M2' get activated when hearing the word 'A'. Therefore, their relationship strengthens and maybe, in the case of model construction, introduces a causal relation between them. On the other hand, in model reduction, two nodes are more likely to stay connected if the nodes are often activated together. If a causal relation between two nodes is less often activated, the relation between those nodes is more likely not to be that relevant. It is important to note that two nodes might only be active together by chance. Therefore, the observation of two nodes to together should not necessarily induce a causal relation between them. Instead, the co-occurrence of these two nodes only specifies that they are more likely to have a causal relation.

Although the Hebbian principle underlies both model reduction and construction, it does not follow that both models converge to the same 'true' generative model. For this consider Figure 1 and Figure 2 again, where it is interesting to note that the developed models (Figure 1b and Figure 2b) are not identical. For example, in Figure 2b the nodes 'Sunny' and 'Umbrella' have a causal relation while they do not in Figure 1b. The different final generative models illustrate the idea that models do not necessarily have to converge to the same 'true' model. One reason for the development of different generative models could be the random or stochastic nature of the environment. That means that the established relations depend on observations. Considering that most of our observations originate from a non-deterministic environment, we do not necessarily observe the same. Therefore, the models we establish are tailored to our observations and not automatically to the entire possible observation space of the environment. In respect of the differences between Figure 1b and Figure 2b this could be interpreted as that the first agent observed more people walking around without umbrella when it is raining while the latter agent observed more people taking an umbrella when it is sunny.

2 Explaining away Prediction Error

As mentioned above, the predictive processing aspect is based on a generative models which, in most research, are represented as Bayesian models[2]. Every generative models visualization is called a *graphical* model. Graphical models display a hierarchical structure either in the form of prediction layers [3] or as a result of a hierarchical structure within one 'layer' of the hierarchy, due to variable dependency on hyperparameters. The latter is shown in Figure 3. Particularly, the graphical model consists of three different types of nodes that represent *hyperparameters*, *hidden* and *observed variables*. The top area of the model in Figure 3 shows diamond nodes which represent the *hyperparameters* α and β. In Figure 3 the combination of α and β models our beliefs about the hidden variable θ. More specifically, that means updating θ follows the probability distributions over α and β. Hyperparameters are initially set to a certain probability distribution to represent the prior beliefs about the situation. After data observation, the hyperparameters are updated to capture the nature of the observed data better. In contrast to hyperparameters, *hidden variables* are learned over different trials and not initially set to a specific probability distribution. Hidden variables do most of the time contain the probability distributions of interest. Lastly, the shaded nodes represent *observed variables*. These are the only variables which can be directly observed in the environment. Independent of the type of variable, all nodes are connected by arrows that indicate dependencies. For example, the dependency $\theta \rightarrow y$ ("θ causes y") implies that we specify the following probability dependency: $P(y \mid \theta)$; (Kruschke, 2010).

[2]Research papers on predictive processing that use Bayesian models are, for example, Kwisthout et al. (2016), Friston et al. (2017) and Clark (2013)

[3]Hierarchical models representing layers of causal chains identify predictions made on one level of the hierarchy with hypotheses on the following layer. For more detail on these model, see, i.e. Kwisthout et al. (2016)

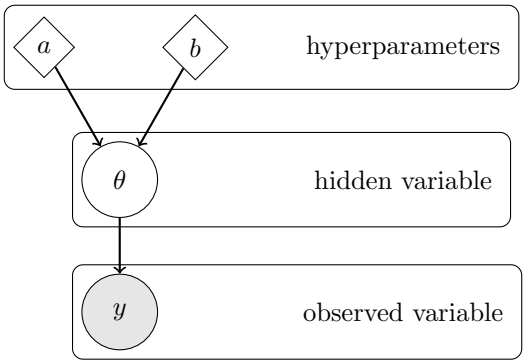


Figure 3: Basic structure of a hierarchical graphical model

Most important for the implementation of model revision are hidden variables. These variables form a so-called hidden state, also referred to as latent state, which defines all possible existing states. Thus, hidden states must contain all variables that can influence the current state of the world. However, since these variables are hidden, the agent can not observe them directly. Instead the agent has to use the observed variables to infer the hidden variables that increase his knowledge about the dependencies in the world (Friston et al., 2017).

A practical example of inferring a hidden state (W) from two specific observed nodes (O_V and O_A) is shown in Figure 4. This example could illustrate a situation in which a person is sitting in a room without windows but would like to know how the weather (W) is outside. The direct observation of the weather is not possible, because there are no windows in the room. However, when a friend comes to visit the visual observation (O_V) about if she is carrying an umbrella or not can provide knowledge about the weather outside. Besides an additional observable auditory cue (O_A), e.g. how much your friend talks, can help you infer about the hidden variable 'Weather' (W). Overall the goal is to infer the hidden node W that independently causes the observed variables O_V and O_A.

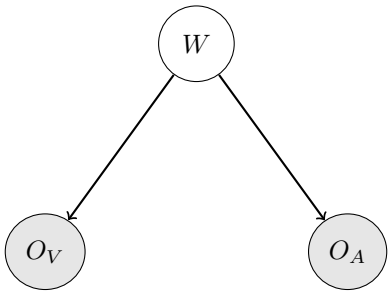


Figure 4: Graphical model with hidden variable W (indicating what the weather conditions are, e.g. rainy, cloudy or sunny) that infleunce the two observed variables: (1) the auditory O_A (how much your friend is talking) and (1) the visual cue O_V (did your friend carry an umbrella or not).

In the following two sections (Section 2.1 and Section 2.2) the methods behind the two techniques *updating* or *revision* are explained. Similarly to the paper by Kwisthout and Rooij (2019) updating will refer to the process of Bayesian updating of the probability distributions over the model. On the other hand, revising a model refers to adding or removing causal relations by using the prediction error in order to accommodate a changing world. This definition of model revision limits its

possibilities to add or remove causal relations. Therefore the definition of model revision here is a simplified version of the common definition of model revision used in other papers, as i.e. Kwisthout et al. (2016). In general model revision would include a structural change in the model by adding or removing variables, or changing the different states a parameter can take. For example, consider a generative model describing the implications of 'Weather' on if people are carrying umbrella's or not. In general, model revision would, for example allow adding a hidden variable 'Wind' to the model. 'Wind' could then help inferring if people will take an umbrella or not. However, as also mentioned in the paper by Kwisthout and Rooij (2019), in the context of predictive processing aspects, such as adding novel variables, still need to be studied.

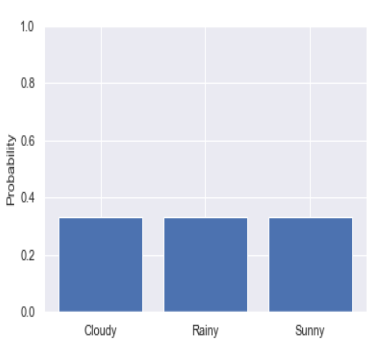
2.1 Updating Hidden States

As stated in Section 2 an increase in the posterior probability of a generative models relates to a more accurate representation of the input data. The posterior probability of the model $p(\theta \mid y)$ stands in relations with the models prior $p(\theta)$ and likelihood $p(y \mid \theta)$ as indicated by the Bayes rule (Equation 1).

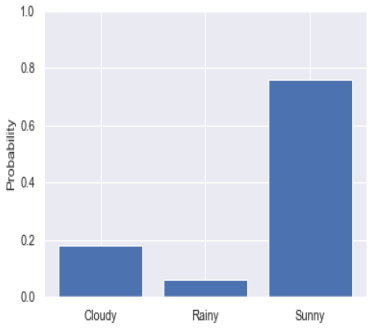
$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{p(y)} \tag{1}$$

The Bayes rule (Equation 1) defines a framework to update beliefs about latent variables of a statistical model with parameters θ given data y using Bayesian inference. Consider the Bayesian network from above again (Figure 1 and 2), where the parameters of the model θ represent concrete states of the hidden node $\theta_i = \{\theta_1 = Cloudy, \theta_2 = Rainy, \theta_3 = Sunny\}$. The Bayes rule (Equation 1) is used to update the probability of one specific state of θ_i given the observed outcome data y, also called the *posterior* ($p(\theta_i \mid y)$). Next to the posterior, the *prior* probability ($p(\theta_i)$) can basically be seen as a summary of past information excluding the information of the current iteration (Vilares and Kording, 2011). Typically the prior starts with a uniform distribution over all possible states, because there are no prior beliefs about the environment yet. That means we initially belief that all possible outcomes of θ_i are equally likely. After observing data y the distribution of θ_i is updated accordingly. Now the prior includes the knowledge about the most current observation. Compared to that the *likelihood* $p(y \mid \theta)$ favours causal relations that increase the probability of co-occurring events.

Figure 5 shows a possible way of updating a discrete categorical probability distribution over the hidden node θ. Initially probabilities are uniformly distributed over all possible states (*Cloudy*, *Rainy* and *Sunny*) of the hidden node θ. After multiple iterations of updates almost all the probability mass is moved to $\theta_3 = Sunny$ and moved away from $\theta_2 = Rainy$ (Figure 5b).



(a) Prior uniform distribution over θ.



(b) Probability distribution over θ after multiple iterations.

Figure 5: Example of updating the categorical probability distribution of the hidden variable θ with the possible states $\theta = [cloudy, rainy, sunny]$. Initially the probability is uniformly distributed over all states (a). After multiple updates most probability mass shifted to 'Sunny' and away from 'Rainy' (b).

After distributing the probabilities over the model parameters, the prediction error between the generative model and the true model of the environment is calculated. A common measure is the *Kullback-Leibler Divergence*, also referred to as *relative entropy*. Equation 2 shows the (*forward*) Kullback-Leibler divergence, which measures the information lost when one (predicted) probability distribution $Pr_{(Pred)}$ is used to approximate the other ('true') probability distribution $Pr_{(Obs)}$. That means a lower value of the KL divergence indicates that two distributions diverge less from each other. Thus, a lower KL value is favourable.

$$D_{KL}(Pr_{(Obs)} \mid\mid Pr_{(Pred)}) = \sum_{\mathbf{p} \in \Omega(Obs)} Pr_{(Obs)}(\mathbf{p}) \log_2 \frac{Pr_{(Obs)}(\mathbf{p})}{Pr_{(Pred)}(\mathbf{p})} \tag{2}$$

Two characteristics that the probability distributions over $Pr_{(Pred)}$ and $Pr_{(Obs)}$ must usually fulfil is that (1) both $Pr_{(Pred)}(x)$ and $Pr_{(Obs)}(x)$ must sum up to 1 and that (2) each of them must be higher than zero ($Pr_{(Pred)}(x) > 0$ and $Pr_{(Obs)}(x) > 0$) for all values $x \in X$ (Han). The first condition is fulfilled in the context of the simulation explained in Section 3. Whereas, when the agent makes a deterministic prediction, instead of returning a probability distribution, the second condition is violated. Allowing deterministic predictions of the agent is important in case there is only one causal relation between the hidden states and outcomes or when the agent randomly guesses an outcome. Because the KL divergence (Equation 2) does only yield an error if the denominator is zero one can use the *reverse* Kullback-Leibler divergence (Equation 3) can instead be used to approximate the distance between the predicted and observed probability distribution.

$$D_{KL}(Pr_{(Pred)} \mid\mid Pr_{(Obs)}) = \sum_{\mathbf{p} \in \Omega(Pred)} Pr_{(Pred)}(\mathbf{p}) \log_2 \frac{Pr_{(Pred)}(\mathbf{p})}{Pr_{(Obs)}(\mathbf{p})} \tag{3}$$

Compared to the forward Kl divergence, the reverse KL divergence measure will not error in case of a deterministic agent prediction, because the environment is surely non-deterministic[4].

[4]Even though the reverse KL divergence does not change the direction of the measure it does, however, change

2.2 Revision of Causal Relations

While the probability distribution over the hidden states is updated after each iteration, as described in the previous Section 2.1, a revision of the causal relations should only occur if it helps to resemble the environment more accurately. Therefore, it is necessary to settle conditions under which model revision should occur. Within the predictive processing framework, the concrete conditions that cause model revision still need to be explored. However, there exists a postulation of different scenarios for which model revision might be favourable to model updating (Rutar et al., ress). Possible causes, such as multiple smaller PE's, are described in the first section. Inspired by the paper of Rutar et al. (ress) model revision here occurs *dependent on the context*. Taking the context into account, one can separate the following causes that generate a high prediction error: (1) random or stochastic causes due to a non-deterministic environment or (2) misrepresentation or unawareness of rules that regulate the environment. This distinction is needed such that only addition of relevant causal relation between two different variables X and Y occurs if their co-occurrence $X \rightarrow Y$ ("X causes Y") is important. In other words, a causal relation between two variables should only exist if the probability for $p(Y \mid X)$ is high enough. Therefore, differently to the updating of the probability distributions, revision will only occur when it adds new information or removes unnecessary knowledge of the agents model.

In the next section describes the methods to simulate the model revision process.

2.2.1 Model Revision Process

The model revision process consists of two phases (Figure 6): At first, if the possibility exists a revised model m_2 is created (Phase 1 in Figure 6). In the second phase of Figure 6, called *Model comparison*, the revised agent's model m_2 is compared to the current agent's generative model m_1. If the revised model is a better representation of the environment than the current model m_1 is replaced by the revised model m_2.

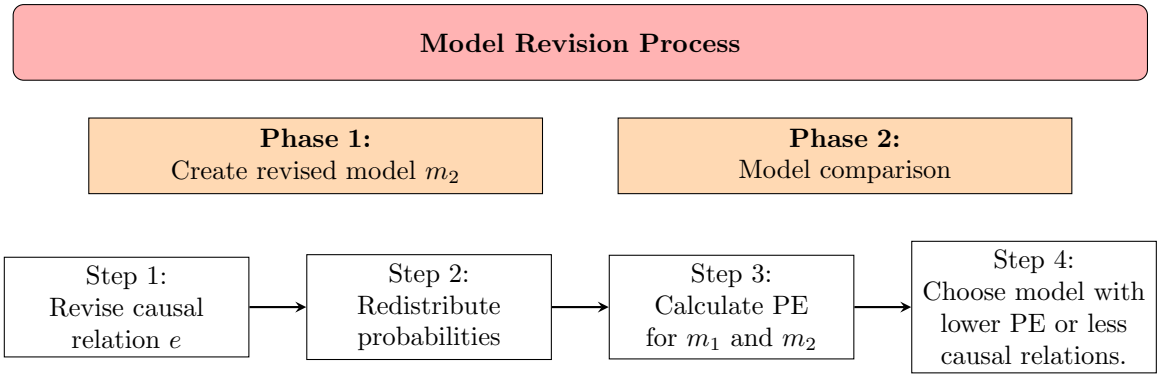


Figure 6: Flowchart of two phase model revision process

(Step 1) The first step in order to create a revised model m_2 requires identifying a causal relation that can be revised (Phase 1 in Figure 6). Which causal relation should be revised depends on which model revision approach the agent is using. If the agent is performing model reduction, then the activated causal relation e between the currently active hidden state used to predict the outcome

the magnitude of the error, because it is not a symmetric distance measure.

and the predicted outcome is revised. Consider a world that has a set of different observed nodes $\mathcal{O} = \{O_1, O_2, ..., O_n\}$ and hidden nodes $\mathcal{H} = \{H_1, H_2, ..., H_m\}$. Every observed O_i or hidden node H_j represents different states of the world. Again, look at Figure 1 from above. In this scenario the world would consist of one hidden node ($H = Weather$) which can take on three different states $H = \{h_1 = \text{Rainy}, h_2 = \text{Cloudy}, h_3 = \text{Sunny}\}$ and one observed node ($O = Umbrella$) with the following states: $O = \{o_1 = \text{True}, o_2 = \text{False}\}$. Now if the agent predicts a specific outcome i based on the hidden node j then the causal relation between $e = h_j \rightarrow o_i$ will be revised That means, the revised model m_2 is a copy of the original model without the causal relation e, such that $m_2 = m_{(Red)} = m_1 - \{e\}$. On the other hand, when the agent performs model construction it chooses a random causal relation e that is added to the original model. Adding the causal relation e creates a revised model m_2, such that $m_2 = m_{(Cons)} = m_1 + \{e\}$. It is necessary to choose a random causal relation initially, because the agent believes that all variables are independent to each other. That implies the agent will not consider the possibility of dependency between variables if it is not explicitly passed as an alternative, revised model.

(Step 2) After removing or adding a causal relation, the number of variables in a specific hidden state H_i changed. Consequently, the sum of the probability over the hidden state does not add up to one anymore. Therefore, a redistribution of the probability is necessary. Redistributing the probability is equal to performing a model updating step, that will be explained later in Section 3.2. For now, it is most important to know that the probabilities are sampled from a Dirichlet distribution. The Dirichlet distribution, also called multivariate Beta-distribution, samples over a vector α (Kruschke, 2010). In this paper α is a concentration vector which keeps track of how often a combinations of hidden states $\mathcal{H}$ and outcomes $\mathcal{O}$ occurred together. When sampling the new probability distributions for the revised model, the Dirichlet distribution will only sample over the states that have a causal relation.

The second phase of the defined model revision process describes *model comparison* (Figure 6). During this phase, the current model m_1 and revised model m_2 are compared via their prediction error to the environment m_E. Overall this phase measures which of the two models (m_1 or m_2) capture the structure of the environment the best. The model comparison step is inspired by the basic idea of Bayesian model comparison (Equation 4). Bayesian model comparison compares the posteriors of each model ($p(m_1 \mid D)$ and $p(m_2 \mid D)$) on the basis of each models evidence ($p(D \mid m_1)$ and $p(D \mid m_2)$) and prior ($p(m_1)$ and $p(m_2)$). Because it is assumes that the two models have the same likelihood they must have the same variable. In other words, model comparison requires the parameters of the different models to be equal and that the only difference between models exists in the underlying probability distribution (Friston et al., 2018).

$$\frac{p(m_1 \mid D)}{p(m_2 \mid D)} = \frac{p(D \mid m_1)}{p(D \mid m_2)} \frac{p(m_1)}{p(m_2)} \tag{4}$$

(Step 3) The last fraction of Equation 4 measures the ratio of the prior beliefs over the two models. When performing model comparison in this work, we do not favour any of the two models *a priori*. Therefore, the last term equals out, and model comparison is only based on the evidence of models (Bayes factor). However, instead of using the ratio of evidence for each model we will immediately compare the posterior odds of the models using the Kullback-Leibler divergence ($PE_{m_1} = D_{KL}(m_1 \mid\mid m_E)$ and $PE_{m_2} = D_{KL}(m_2 \mid\mid m_E)$).

(Step 4) Based on the calculated prediction errors (PE_{m_1} and PE_{m_2}) the agent can decide which model it favours. The following three scenarios can arise

1. $PE_{m_1} > PE_{m_2}$

2. $PE_{m_1} < PE_{m_2}$

3. $PE_{m_1} = PE_{m_2}$

In general, the model selection process should always favour the model with the lowest prediction error. Therefore, in the first two cases, the model with the lower PE is selected by the agent. In the third case, both models encode the same amount of environmental information. If that is the case, the decision of the agent follows the simplicity principle, also referred to as *Occam's razor*. This states, that, if the observation can follow a simpler explanation, it should be chosen over more complex solutions (Feldman, 2016). In other words, only if the data requires it, the agent will keep the more complex model[5]. Note that if the original model m_1 or revised model m_2 should be favoured is dependent on the approach of model revision. The revised model m_2 is simpler for the process of model reduction. However, m_2 is more complex for the process of model construction. Thus, which model the agent should decide to use as its generative model is dependent on the model revision approach.

After the agent performed the model revision process (Figure 6), the agent could have changed its generative model. While the process of model reduction is guided by the assumptions of reducing model complexity, due to the initial dependency between all variables, it might be unclear why an agent should consider a more complex model during the model construction approach. The reason is that adding necessary dependencies between two variables also reduces the free energy of the model, in the sense that the minimization of the overall prediction error causes a reduction in free energy (Friston et al., 2017).

3 Simulation

The previous section focused on the methods for updating and revising a generative model. This section describes the simulation which is used to investigate the development of a generative model. In the following simulation, the agent's goal is to develop a model that integrates the association rules from the environment. The implemented environment is simplified which allows us to observe the interaction of variables and reason about the changes in the generative model of the agent in a step-by-step manner.

The general version of the simulation was developed together with the Bachelor thesis group. Thus, Section 3.1 is written in collaboration with Isabel Burgos, Lennart Geertjes and Ellen Schrader. Following Section 3.1 describes the general simulation environment and Bayesian models used for implementation. Section 3.2 specifies the specific design choices and changes to the Bayesian models. After that, Section 3.3 defines the experimental settings and lastly, Section 3.4 shows the results obtained after running the simulation.

3.1 General Simulation outline[6]

In the following part, the general simulation environment is described as the group built it. The goal was to form a common foundation for the individual research questions of all group members.

[5]The model selection principle, using Occam's razor, is also embodied in Bayesian Inference itself. There is balances the trade off between complexity and structural fit of the model (Tenenbaum et al., 2011).

[6]This section is written in collaboration with the students (Isabel Burgos, Lennart Geertjes and Ellen Schrader) in the Bachelor thesis group.

3.1.1 Simulation Outline

The simulation environment aims to simulate the construction of generative models over time, based on the individual research questions. The performance of the agent is measured by, for instance, the change of prediction error over time.

The simulation environment consists of an agent that is repeatedly shown coloured tiles in different positions. The steps of a single iteration of the simulation can be imagined in the form of an empirical experiment: (Step 1) The agent only sees three empty positions (Figure 7A) (Step 2) then, on one of the positions, a coloured tile is shown (Figure 7B). (Step 3) Depending on the colour and the spatial position, the agent makes a prediction regarding which object it believes to be underneath the tile. (Step 4) Afterwards, the tile will be flipped, showing an object located underneath the tile (Figure 7C). The tile which is flipped is determined by the environment, not the agent (implying a passive agent). (Step 5) Afterwards, the tile flips back, and the agent once again sees three empty positions (Figure 7D). This process is repeated for a given number of iterations together forming one single trial. The goal is that the agent learns to correctly predict the object underneath a tile, based on the feature (in this case colour) and the spatial position of the tile.

The general simulation has certain properties, which will be explained in detail in this paragraph. A tile can have one of three colours: 'blue', 'red' or 'yellow' (Figure 8A). Each of these colours is associated with an object: 'blue' is associated with 'square'; 'red' with 'triangle'; and 'yellow' with 'circle' (Figure 8D). The spatial position is related to where the tile appears; this can be either 'left', 'centre' or 'right' (Figure 8B). The spatial position only influences the object underneath if the tile appears in the centre position. The centre position is always associated with a 'star', independent of the colour. The effect of the spatial position can be seen in Figure 8C, taking Figure 8A as a row of example tiles. The blue and yellow tiles display a 'square' and 'circle', respectively, as expected by their mapping of colour. Although 'red' is mapped to a 'triangle', the red tile in the centre displays a star due to its spatial feature. Do note that the row in Figure 8A is exemplary; in the actual simulation tiles are never shown simultaneous but always sequential, as shown in Figure 7.

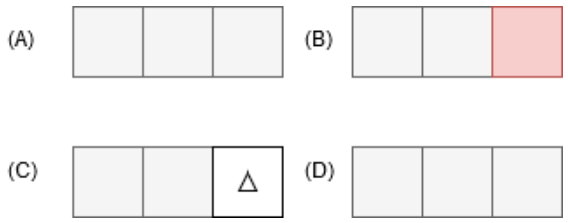


Figure 7: An example of a single iteration in the general simulation. First the agent sees 3 empty positions (A), then, at one of the positions, a tile appears (B). After perceiving the stimuli, the agent makes a prediction, then the tile is flipped revealing the object underneath (C). Lastly the object is covered again (D) and a new iteration starts.

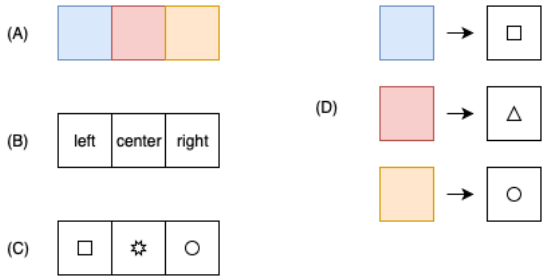


Figure 8: Which object is shown depends on the interaction between the colour of the tile (A) and spatial position (B). The mapping from colour to object is shown in figure (D). However, the mapping does not only depend on colour but also on spatial position so, for example, all centre tiles are mapped to a star (C).

In the following two sections, the model of the environment and the generative model of the agent will be explained. The environment model is used to generate events, such as colour and spatial position of a tile. The agent uses the generated events to predict an object. While the environment model is needed to generate events, the general framework aims to investigate the generative model of the agent.

3.1.2 Environment

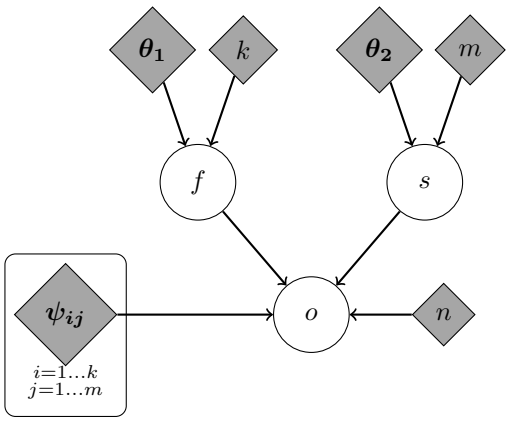


Figure 9: The graphical model of the environment: Here node o represents the object. The object is dependent on the probability distribution $\boldsymbol{\psi_{ij}}$ given the spatial position and feature of the tile, represented by nodes s and f respectively. Nodes k, m and n represent the amount of possible features, spatial positions and objects respectively. $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ are two uniform categorical distributions from which the features and spatial position for each iteration are drawn. Diamond shaped nodes indicate (hyper)parameters which are given to the environment, and the dark grey colour indicates that the (hyper)parameters are constant throughout the trial.

Environment model:

$$f \mid \boldsymbol{\theta_1}, k \sim Categorical(\boldsymbol{\theta_1}, k) \tag{5}$$
$$s \mid \boldsymbol{\theta_2}, m \sim Categorical(\boldsymbol{\theta_2}, m) \tag{6}$$
$$o \mid \boldsymbol{\psi_{ij}}, f, s, n \sim Categorical(\boldsymbol{\psi_{fs}}, n) \tag{7}$$

In Figure 9 the graphical model of the environment is shown. The node o represents the object which can be found underneath a tile. As explained above, the object which is shown when the tile is flipped depends on the feature of the tile f (e.g., red), the spatial position of the tile s (e.g., left), and their corresponding probability distributions, as can be seen in the environment model. For our general environment, the corresponding probability distributions are nearly deterministic. The feature and spatial position of the tile that will be shown are drawn from a categorical distribution with a uniform distribution over all features and spatial positions, respectively. This uniform distribution is represented by the vector of category probabilities $\boldsymbol{\theta_1}$ or $\boldsymbol{\theta_2}$. In general, dark grey diamond-shaped nodes indicate (hyper)parameters that are given to the environment. The hyperparameter k represents the number of possible features, the hyperparameter m the number of spatial positions, and the hyperparameter n the number of possible objects. Note that the number of possible objects and features do not have to be the same, as illustrated in Section 3.1.1. Depending on the environment and conditions needed, these numbers can be adjusted.

3.1.3 Agent

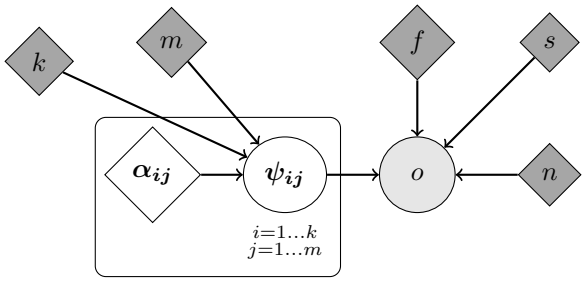


Figure 10: The graphical model of the agent: Here node o represents the object. The object is dependent on the probability distribution $\boldsymbol{\psi_{ij}}$ given the spatial position s and feature of the tile f. Nodes k, m and n represent the amount of possible features, spatial positions and objects respectively. The categorical probability distribution $\boldsymbol{\psi_{ij}}$ is dependent on $\boldsymbol{\alpha_{ij}}$, which represents the concentration vector. Diamond nodes indicate the (hyper)parameters. (Hyper)parameters k, m and n are stable over multiple iterations, while the parameters f and s differ between iterations and hyperparameter $\boldsymbol{\alpha_{ij}}$ is learned throughout the trial.

Generative model:

$$\boldsymbol{\psi_{ij}} \mid \boldsymbol{\alpha_{ij}}, n \sim Dirichlet(\boldsymbol{\alpha_{ij}}, n) \quad \text{with } i = 1...k,\ j = 1...m \tag{8}$$
$$o \mid \boldsymbol{\psi_{ij}}, f, s, n \sim Categorical(\boldsymbol{\psi_{fs}}, n) \tag{9}$$

In Figure 10 the graphical model of the agent is shown. As described in Section 3.1.2 (hyper)parameters are represented by diamond nodes. However, in this case, there is a difference between some of the parameters indicated by the colour. Dark grey nodes are given by the environment, where k, m and n are stable over multiple iterations, while the parameters f and s differ between iterations, depending on which feature and spatial location has been selected by the environment. White parameter nodes such as $\boldsymbol{\alpha_{ij}}$ are learned over multiple iterations through observation. The light grey node represents the prediction of the agent.

Each of the feature and spatial specific concentration vectors $\boldsymbol{\alpha_{ij}}$ is initially set to a uniform vector with one pseudo-observation for each object. The probability distribution $\boldsymbol{\psi_{ij}}$ of the objects, given the features and spatial positions, is sampled from a Dirichlet distribution with the concentration vector $\boldsymbol{\alpha_{ij}}$. As for the environment (Figure 9), k represents the number of possible features, m the number of spatial positions, and n the number of possible objects. The prediction of the object is based on a categorical distribution over the probability distribution $\boldsymbol{\psi_{fs}}$ given the observed feature f and spatial position s.

3.1.4 An Adaptation of the Simulation

The environment as it is currently implemented is quite simplistic: there is no volatility in the environment, and it is not very challenging for the agent to understand the environmental contingencies. The group aimed to make a simulation that can serve as a foundation to expand upon for the individual research questions. Therefore everything described above is potentially subject to change in the individual projects.

3.2 Specific Design choices

The general simulation outline is further simplified such that the centre position is associated with object 'square' instead of 'star'. The goal of the agent is to 'understand' and represent the association rules (Figure 8) with the hidden variable $\boldsymbol{\psi_{ij}}$. Other design choices to the simulation are the following:

- *Non-deterministic environment*: As explained in the previous Section 3.1.1, the environment is not deterministic. That means, even though the combination of a feature f and spatial position s are theoretically associated with a specific object o sometimes a different object can follow the co-occurrence of f and s. Co-occurrences that do not follow from the association rules, originate from the uncertainty of the environment.
- *Passive agent*: Instead of letting the agent actively choose its next action (*active inference*), the agent is passive. First of all, that means the agent can not test or confirm its sensory predictions by performing specific actions. However, the agent does have the means to change its predictive model to fit the environment better (*perceptual inference*); (Seth, 2015). Thus, the agent is limited to predict the outcome given a certain policy[7]. Secondly, the agent has no prior belief to minimize free energy in the world. Instead, the only way an agent can resolve uncertainty about the world is by creating a generative model that resembles the environment closely (Friston et al., 2017).
- *No explicit feedback*: The agent can not directly observe the object below a tile in the environment. Thus, updates or revision do rely on unsupervised mechanisms to learn the association

[7] A policy refers to a sampled feature f and spatial position s from the environment.

rules. Instead, the agent uses a unsupervised paradigm to develop its generative model. Therefore, the most probable object is not directly compared with the actual object. However, the agent can observe the prediction error, which gets fed back through the model. In case the prediction error is high, the agent concludes that the prediction was incorrect. However, it is not possible to observe what the correct object below the tile is. That means updating and revision of the model is only based on the prediction error minimization process and not on supervised learning (Smith et al., 2019).

- *Random guessing*: If there is no causal relation between the sampled feature f and spatial position s from the environment, and an object o, then the predicted object is a random guess over the different object possibilities. A random guess is defined as a categorical distribution over a uniform distribution of objects O.
- *Model comparison using the same policy*: When comparing two models, these are compared using the same policy over a specified amount of iterations n (Algorithm 2). Thus, the agent will compare the prediction of the models m_1 and m_2 given the same feature f and spatial position s instead of sampling a new feature f_{new} from the environment every iteration.

Because the design choices did not impact the environment, it was not necessary to change the environment model (Figure 9). However, the design choices and integration of the model revision process (Figure 6) required a change in the agent's generative model of Figure 10.

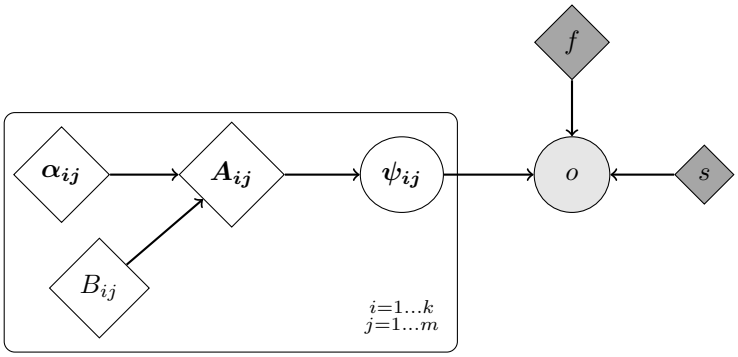


Figure 11: The updated graphical model of the agent: Node o represents the object which is dependent on the probability distribution $\boldsymbol{\psi_{ij}}$ given a feature f and spatial position s of the tile. The categorical probability distributions $\boldsymbol{\psi_{ij}}$ is sampled from the categorical distribution $\boldsymbol{A_{ij}}$. $\boldsymbol{A_{ij}}$ includes all pseudo count for the hidden nodes that have a causal relation to an outcome. In turn $\boldsymbol{A_{ij}}$ is determined given B_{ij}, which represents the causal relations between nodes, and $\boldsymbol{\alpha_{ij}}$, which is the concentration vector over all states.

Generative model:

$$\boldsymbol{A_{ij}} \mid \boldsymbol{\alpha_{ij}}, B_{ij} \sim \begin{cases} \boldsymbol{\alpha_{ij}}, & B_{ij} = 1 \\ 0, & B_{ij} = 0 \end{cases} \quad \text{with } i = 1...k,\ j = 1...m \tag{10}$$

$$\boldsymbol{\psi_{ij}} \mid \boldsymbol{A_{ij}}, n \sim Dirichlet(\boldsymbol{A_{ij}}, n) \quad \text{with } i = 1...k,\ j = 1...m \tag{11}$$

$$o \mid \boldsymbol{\psi_{ij}}, f, s_j, n \sim Categorical(\boldsymbol{\psi_{fs_f}}, n) \tag{12}$$

In Figure 11 the updated graphical model of the agent is shown. As explained in the previous Section, the concentration vector $\boldsymbol{\alpha_{ij}}$ is initially a uniform distribution, where each combination of feature's $f \in F$ and spatial position's $s \in S$ has a pseudo-count of one for every object $o \in O$. The vector B_{ij} encodes the dependency relations by zero's (there *does not* exist a relation between the feature f, spatial position s and object o) and one's (there *does* exist a relation between the feature f, spatial position s and object o). The concentration vector $\boldsymbol{A_{ij}}$ gets updated dependent on $\boldsymbol{\alpha_{ij}}$ and B_{ij}. If B_{ij} has a one at the current policy then the pseudo-counts from $\boldsymbol{\alpha_{ij}}$ are transferred to $\boldsymbol{A_{ij}}$. Otherwise, $\boldsymbol{\alpha_{ij}}$ is set to zero for the current policy (Equation 10). The updating step of $\boldsymbol{\psi_{ij}}$ is equal to the updating explained in Section 3.1.3 for Figure 10, except that the Dirichlet distribution is now sampled over $\boldsymbol{A_{ij}}$ instead of $\boldsymbol{\alpha_{ij}}$ (Equation 11). Lastly, the object is drawn from a Categorical distribution (Equation 12), equal to the process explained in Section 3.1.3.

3.3 Implementation

This section describes the experimental settings of the environment and agents. Overall, the implementation includes two agents, one performing model reduction (agent 1) and the second model construction (agent 2). Both agent's are defined with the same generative model (Figure 11), yet they differ in their variable settings. To be able to compare the different model revision processes and developed final generative models, the basic environment must be the same for both agent's (Figure 9).

3.3.1 Experimental Settings Environment

The environmental settings are equal for both agent's. Every feature f or spatial position s has an associated probability that it can occur with. The probability distributions for each feature f and spatial position s are:

Feature	**Probability**
Blue	$\frac{1}{3}$
Red	$\frac{1}{3}$
Orange	$\frac{1}{3}$

(a) $\boldsymbol{\theta_1}$: Probability for the features $f \in F$.

Spatial position	**Probability**
Left	$\frac{1}{3}$
Center	$\frac{1}{3}$
Right	$\frac{1}{3}$

(b) $\boldsymbol{\theta_2}$: Probability for the spatial positions $s \in S$.

Table 1: Probability distributions over $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ that indicate the probability to choose a certain feature f (a) or spatial position s (b), respectively.

Table 1 defines the probabilities of $\boldsymbol{\theta_1}$ (probability distribution to choose a feature f) and $\boldsymbol{\theta_2}$ (probability distribution to choose a spatial position s) defined in the environment model (Figure 9). As explained above the probability distribution over the features and spatial positions is uniform. That means every feature f or spatial position s is equally likely to occur. The uniform distribution ensures that the agent can theoretically experience all possible combinations equally often.

Besides that Table 2 specifies the probability distribution over the possible objects $o \in O$ given features $f \in F$ and spatial positions $s \in S$. Except for feature 'red' in combination with spatial positions 'left' and 'right' there is little uncertainty in the environment.

Feature	Spatial	Probability
Red	Left	0.5
Red	Center	0.0001
Red	Right	0.5
Blue	Left	0.0001
Blue	Center	0.0001
Blue	Right	0.0001
Orange	Left	0.0001
Orange	Center	0.0001
Orange	Right	0.0001

(a) Probability of object $o_1 = triangle$ combined with feature $f \in F$ and spatial position $s \in S$.

Feature	Spatial	Probability
Red	Left	0.25
Red	Center	0.0001
Red	Right	0.25
Blue	Left	0.0001
Blue	Center	0.0001
Blue	Right	0.0001
Orange	Left	0.9998
Orange	Center	0.0001
Orange	Right	0.9998

(b) Probability of object $o_2 = square$ combined with feature $f \in F$ and spatial position $s \in S$.

Feature	Spatial	Probability
Red	Left	0.25
Red	Center	0.9998
Red	Right	0.25
Blue	Left	0.9998
Blue	Center	0.9998
Blue	Right	0.9998
Orange	Left	0.0001
Orange	Center	0.9998
Orange	Right	0.0001

(c) Probability of object $o_3 = circle$ combined with feature $f \in F$ and spatial position $s \in S$.

Table 2: Probability distributions of $\boldsymbol{\psi_{ij}}$ for the possible objects (o_1 = square (a), o_2 = circle (b), o_3 = triangle (c)) given each combination of feature $f \in F$ and spatial position $s \in S$ of a tile.

3.3.2 Experimental Settings Agents

In contrast to the common environment of both agents, almost all parameter settings for the agents model (Figure 11), are different for the agents performing model reduction or model construction, except for the concentration vector $\boldsymbol{\alpha_{ij}}$.

First consider $\boldsymbol{\alpha_{ij}}$ which has the same setting for both agent's. Assume an environment that consists of the different features $F = \{f_1, f_2, ..., f_k\}$, spatial positions $S = \{s_1, s_2, ..., s_m\}$ and objects $O = \{o_1, o_2, ..., o_n\}$. The concentration vector $\boldsymbol{\alpha_{ij}}$ initially includes a pseudo count of one for each combination of feature $f \in F$, spatial position $s \in S$ and object $o \in O$ (Equation 13).

$$\forall f \in F, \forall s \in S, \forall o \in O, \boldsymbol{\alpha_{fso}} = 1 \tag{13}$$

The initialization of the remaining parameters B_{ij}, A_{ij} and $\boldsymbol{\psi_{ij}}$ is different per agent. However, the values of A_{ij} and $\boldsymbol{\psi_{ij}}$ are sampled according to the Equations 10 and 11. Thus, it is only required to set the values of $\boldsymbol{\alpha_{ij}}$ and B_{ij} before running the model development phase for n iterations (Algorithm 1). Unlike $\boldsymbol{\alpha_{ij}}$, B_{ij} is dependent on the model revision process. The following paragraphs define the settings of B_{ij} for every agent:

Agent 1: Model reduction Agent one should start with a model where each all variables between states are 'dependent' on each other ('dependency prior'). Thus, initially vector B_{ij} defines an existing stochastic causal relation for all possible transitions between hidden states (in this simulation features F and spatial positions S) and outcomes (objects O). In the context of this simulation that means, vector B_{ij} is a uniform distribution where every possible combination of feature $f \in F$, spatial position $s \in S$ and object $o \in O$ is assigned a value of one (Equation 14). As explained earlier, a value of one in vector B_{ij} means that there *does* exist a causal relation.

$$\forall f \in F, \forall s \in S, \forall o \in O, B_{fso}0 = 1 \tag{14}$$

Agent 2: Model construction In contrast to agent one, the condition of the initial model for agent two is that all variables between states must be 'independent' on each other ('independency prior'). Thus, agent two starts with a value of zero for each position in B_{ij} (Equation 15). That means, in the initial model of agent two there *do not* exist any causal relations.

$$\forall f \in F, \forall s \in S, \forall o \in O, B_{fso} = 0 \tag{15}$$

3.3.3 Pseudo Code for Model Revision

Algorithm 1 shows the pseudo-code for training the generative model of the agent using model updating and model revision. Differences between the two model revision approaches are indicated in the Algorithm 1 by different colouring. 'Dark green' colouring stands for the model reduction and 'Dark blue' for the model construction procedure. Both model revision processes are run for the same amount of iterations ($n = 250$) which is sufficient for learning the simple environment presented here.

Algorithm 1 Training of agent's generative model

procedure MODEL REDUCTION OR MODEL CONSTRUCTION(n)
▷ n, the number of iterations to run the simulation

▷ Initialize $\boldsymbol{\alpha}$ and B
$\boldsymbol{\alpha} \leftarrow$ uniform distribution initialized according to Equation 13
$B \leftarrow$ initialized following Equation 14 or Equation 15
for i in n **do**
▷ Sample from the environment a feature f, a spatial position s and a object o
▷ The agent predicts the object o' given f and s from the environment
$o' \leftarrow \text{AGENT}(f, s)$
▷ Model reduction: Check if there exists a causal relation between f, s and o' in B_{agent}
▷ Model construction: Choose a random causal relation between $f \in F$, $s \in S$ and o' and
▷ check if it does not yet exist in the agents B_{agent}
if $B_{agent}.e(f, s, o')$ **or not** $B_{agent}.e(f, s, o')$ **then**
▷ Create the reduced/expanded model by copying B and $\boldsymbol{\psi_{agent}}$
▷ and removing/adding the causal relation e in the copies
$B`[f, s, o'] \leftarrow 0$ **or** $B'[f, s, o'] \leftarrow 1$
$\boldsymbol{\psi_{agent}}` \leftarrow \text{DIRICHLET}(\boldsymbol{\alpha}[B'])$
if $\text{MODELCOMPARISON}(\boldsymbol{\psi'_{agent}}, f, s)$ **then**
▷ When the revised $\boldsymbol{\psi'_{agent}}$ resembles the environment better,
▷ keep $\boldsymbol{\psi_{agent}}$ with the removed/added causal relation e
$\boldsymbol{\psi_{agent}} \leftarrow \boldsymbol{\psi'_{agent}}$
$B \leftarrow B'$
end if
end if
$PE \leftarrow D_{KL}(\boldsymbol{\psi_{agent}} \,||\, \boldsymbol{\psi_{env}})$
▷ Update the concentration parameter $\boldsymbol{\alpha}$
$\boldsymbol{\alpha(f, s, o)} \leftarrow \alpha(f, s, o) + 1$
end for
end procedure

Algorithm 2 specifies the pseudo-code for the second phase of the model revision process (Figure 6). Again the same colour coding is used to differentiate between model reduction and model construction. During the model comparison the prediction error is calculated for n repetitions. Because children do only need a handful events to learn causal relations the model comparison is limited over $n = 10$ iterations (Tenenbaum et al., 2011).

Algorithm 2 Model Revision process (Phase 2)

function ModelComparison(f, s, $\boldsymbol{\psi'_{agent}}$)
▷ f, feature sampled from the environment
▷ s, spatial position sampled from the environment
▷ $\boldsymbol{\psi'_{agent}}$, revised model of the agent

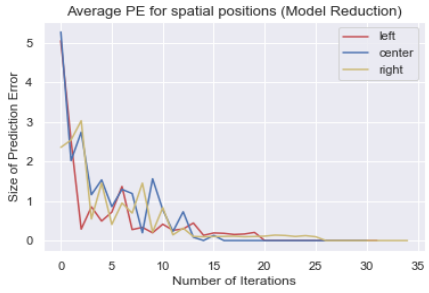
▷ For n repetitions test the performance of the original and revised models on each other
▷ Each time calculate the PE and save it
for i in n **do**
▷ Calculate the PE for the original model
$PE_{original}[i] \leftarrow D_{KL}(\boldsymbol{\psi_{agent}}[f, s] \mid\mid \boldsymbol{\psi_{env}}[f, s])$
▷ Calculate the PE for the revised model
$PE_{revised}[i] \leftarrow D_{KL}(\boldsymbol{\psi'_{agent}}[f, s] \mid\mid \boldsymbol{\psi_{env}}[f, s])$
end for
▷ Compare the average prediction errors of the original and revised model
if $PE_{original}.AVG > PE_{revised}.AVG$ **then**
▷ When the PE of the original model is larger then the revised model is preferred
return 1
else if $PE_{original}.AVG = PE_{revised}.AVG$ **then**
▷ In case the PE is equal for both models, then prefer the simpler model
▷ Model reduction: Prefer the reduced/revised model
▷ Model construction: Prefer the original model
return 1/0
else
▷ The agent keeps its orginal model
return 0
end if
end function

3.4 Results

In this section, the line plots for both model reduction (Figure 12) and model construction (Figure 14) show the development of the prediction error averaged over features F or spatial positions S. Note that features and spatial positions are sampled from the environment. Therefore not every agent observes the same features and spatial positions equally often. Thus, the number of iterations over which the average PE is calculated differs. In addition to the PE line plots, the section also includes Likelihood matrices for each agent. The Likelihood matrices show the established relations and their 'strength' between the hidden states and outcomes (Figure 13 and Figure 15).

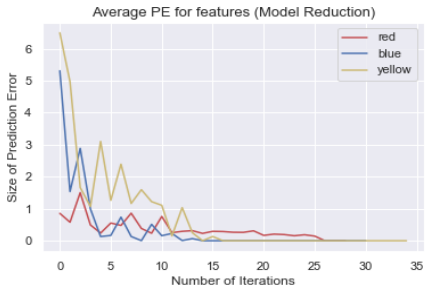
3.4.1 Model reduction

The prediction errors averaged over either the features $f \in F$ or spatial positions $s \in S$ within one model reduction trial can be found in Figure 12.



(a) Prediction error averaged over the spatial positions S for every feature $f \in F$

.



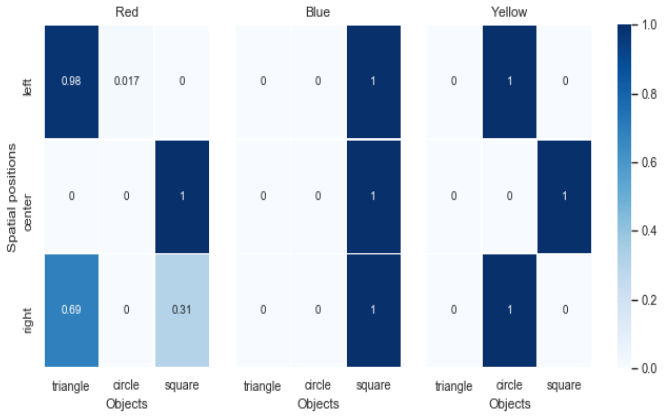
(b) Prediction error averaged over the features F for every spatial position $s \in S$.

Figure 12: Prediction error of one model reduction trial (agent 1). The left Figure (a) shows the PE averaged over the spatial positions S per feature $f \in F$. The right Figure (b) shows the PE averaged over the features F per spatial position $s \in S$.

The prediction error of each feature $f \in F$ averaged over all spatial positions S is plotted in Figure 12a. At the start, the PE is highest for the 'left' and 'centre' and lowest for the 'right' position, with a size of approximately 5 or 2.5, respectively. During the first three iterations, the PE decreases to (more than) half of its previous size for the 'left' and 'centre' positions. After that, up until the 15^{th} iteration, the PE fluctuates from iteration-to-iteration for all spatial positions S. At the end of the trial, the fluctuations descend, and the PE progressively decreases to zero. While the decline of the PE is different for each spatial position, they all reach the same minimum after approximately 25 iterations. Note that the PE is unlikely to settle at zero because it calculates the mismatch between a deterministic prediction by the agent and a non-deterministic environment. If the agent predicts an outcome to occur with a probability of zero, while it theoretically has a low chance of occurring in the environment (i.e. combination of feature 'red' and spatial position 'centre' in Table 2a has the probability of 0.01%) the PE can never converge to zero. Thus, the value of the minimal PE in the line plots of Figure 12 and 14 are most likely not exactly zero, but only close to zero.

Figure 14b shows the averaged PE over the spatial positions S. Both features, 'yellow' and 'blue', have a high PE initially with a size of about 6. Compared to that the PE of feature 'red' is initially lowest, with a size of approximately 1. Within the first iterations, the PE of the features 'yellow' and 'blue' has the steepest decrease. During their decrease, the PE fluctuates over iterations. Later, both PE's of features 'yellow' and 'blue' reach a minimum after approximately 15 iterations. The PE for feature 'red' decreases more progressively with fewer iteration-to-iteration fluctuations. Around the 25^{th} iteration the PE of 'red' reaches the same value as 'yellow' and 'blue'.

(a) Likelihood Matrix of agent 1 *before* Model Reduction trial.



(b) Likelihood Matrix of agent 1 *after* Model Reduction trial.

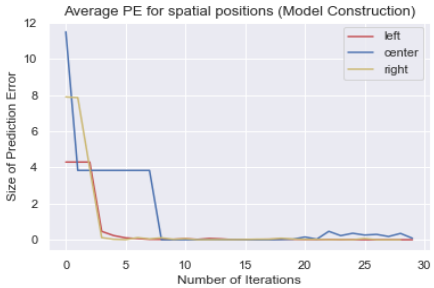
Figure 13: Likelihood matrices of agent 1 (model reduction) *before* (a) and *after* (b) one training trial. The Likelihood matrices show the causal relations and the corresponding probabilities for each transitions between hidden states (combination of feature $f \in F$ and spatial position $s \in S$) and their corresponding outcome (object $o \in O$).

The likelihood matrices in Figure 13 show the causal relations in the model of agent one *before* and *after* the model reduction process. In general, likelihood matrices display the transitions between the hidden states (each features $f \in F$ and spatial positions $s \in S$) to the outcomes (objects $o \in O$). Every likelihood matrix consists of three separate blocks, corresponding to the features 'red', 'blue' and 'yellow'. Each block has a row for each spatial position $s \in S$, 'left', 'centre' and 'right'. Within one block the columns correlate to a object $o \in O$, 'triangle', 'circle' or 'square'. Every row of the likelihood matrix, in a specific block, represents a probability distribution over the specific objects O. Thus, every field indicates how probable the agent is to predict a object o given feature f and spatial position s. The probability of a causal relation is associated with a color according to the probability scale on the right side of the likelihood matrix. Figure 13a shows the likelihood matrix of the agent *before* the model is reduced. Initially, all causal relations of the agent have a equal

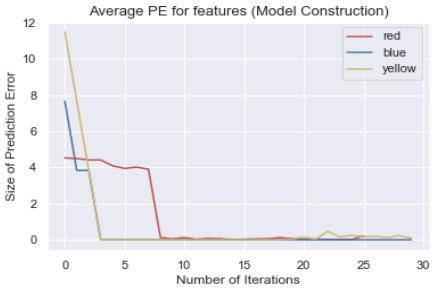
probability of 33%. Figure 13b presents the likelihood matrix of agent one after the model reduction trial. The most causal relations have been reduced for features 'blue' and 'red'. Figure 13b shows that for 'blue' and 'red' the probability mass is completely shifted to one object o. In the left block of the likelihood matrix (Figure 13b) the feature 'red' in combination with the spatial positions 'left' and 'right' has only reduced one causal relation. For the combination 'red' and 'left' there is a higher probability for object 'triangle' to occur with 98% over object 'circle' (1.7%). Finally, the agent predicts the object 'triangle' to occur with a probability of 69% and 'square' with 31%. Within the same block only one causal relation remains for the 'centre' position.

3.4.2 Model construction

The prediction errors averaged over either the features $f \in F$ or spatial positions $s \in S$ within one model construction trial can be found in Figure 14.



(a) Prediction error averaged over the spatial positions S for every feature $f \in F$

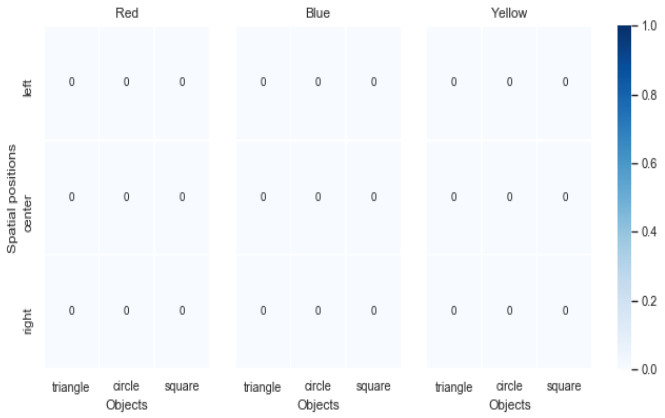


(b) Prediction error averaged over the features F for every spatial position $s \in S$.

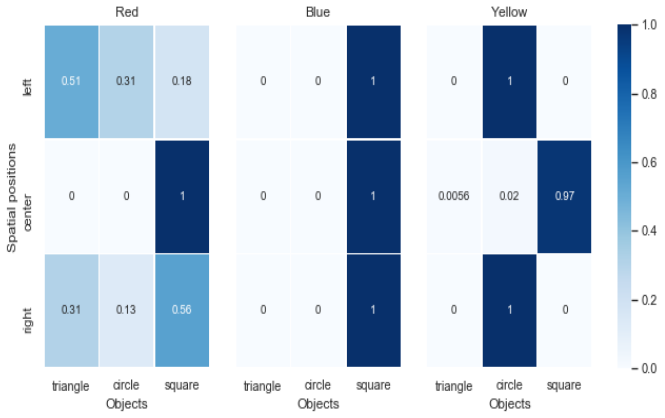
Figure 14: Prediction error of one model construction trial (agent 2). The left Figure (a) shows the PE averaged over the spatial positions S per feature $f \in F$. The right Figure (b) shows the PE averaged over the features F per spatial position $s \in S$.

The PE of each feature $f \in F$ averaged over all spatial positions S is plotted in Figure 12a. At the start, the PE is highest for the spatial position 'left' with a size of 12, then 'right' (size of 8) and lastly 'left' with a size of 4. For all spatial positions, the PE decreases in a step-wise fashion without fluctuations. Both averaged PE's for 'left', and 'right' decrease the most within the first five iterations. The PE for 'centre' reaches its minimum at first during the 7^{th} and 9^{th} iteration. From the 10^{th} iteration on the averaged PE's for all spatial positions are approximately equal with only small sized fluctuations of at most 0.5.

Figure 14b shows the average PE decrease over the spatial positions S for each feature $f \in F$ within one trial. At the beginning, the feature 'yellow' has the highest with a size of almost 12. Features 'blue' and 'red' both have lower PE's of roughly 8 and 4.5, respectively. Both features 'blue' and 'yellow' have a steep decreasing prediction error which reaches a size below 0.5 before the 5^{th} iteration. The PE of feature 'red' is decreasing progressively for the first seven iterations. After that, the PE of 'red' decreases steeply to a size close to zero. For all three features, not many fluctuations from iteration-to-iteration occur. If a fluctuation occurs the size of the prediction error does not change more than 0.5.



(a) Likelihood Matrix of agent 2 *before* Model Construction trial.



(b) Likelihood Matrix of agent 2 *after* Model Construction trial.

Figure 15: Likelihood matrices of agent 2 (model construction) *before* (a) and *after* (b) one training trial. The Likelihood matrices show the causal relations and the corresponding probabilities for each transitions between hidden states (combination of feature $f \in F$ and spatial position $s \in S$) and their corresponding outcome (object $o \in O$).

The likelihood matrices of agent two performing model construction are shown in Figure 15. Before the model construction process starts, the agent has no transition probabilities established between the hidden states and outcomes (Figure 15a). After finishing one trial, the likelihood matrix (Figure 15b) shows that some relations between specific hidden states and outcomes are constructed. Deterministic relations are established for the feature 'blue' in combination with all possible spatial positions to the object 'square'. In the right block (representing feature 'yellow') of Figure 15b deterministic relations exists to the object 'circle' for the 'left' and 'right' spatial positions. In the same block, a causal relation to all objects was constructed given spatial position 'centre'. However, the distributed probabilities over the causal relations indicates that 'square' is predicted with a highest probability of 97% and with the remainder 3% either 'triangle' or 'circle' is predicted. The first block, representing feature 'red', has a deterministic relation to object 'square' given spatial position

'centre'. However, for the other spatial positions 'left' and 'right' the transition probabilities are distributed over all spatial positions $s \in S$. The combination of feature 'red' and spatial position 'left' has the highest probability (51%) associated with object 'triangle', 31% with 'circle' and 18% with 'square'. Lastly, the probability distribution of combination of feature 'yellow' and spatial position 'right' has the following probabilities associated per outcome: 56% with 'square', 31% with 'triangle' and 13% with 'circle'.

4 Discussion

The simulation created for this Bachelor thesis is as a tool to gain insight about the different model revision processes namely model reduction and model construction. In particular, the developed generative models of the two agent's provide a more detailed understanding of the processes. This section discusses the results (Section 3.4) with respect to the initially introduced hypothesis; both models will establish accurate but not necessarily 'true' representations of the environment.

According to the observed results, two generalizations are possible. The first generalization is that *both model reduction and model construction led to accurate agent model.* Figures 12 and 14 support this conclusion by showing that the PE is reduced to almost zero. However, the decline of the PE over time differs per model revision process. During model reduction, the progressive decrease of the PE shows iteration-to iteration fluctuation. Two possible causes for the fluctuation are (a) the stochastic nature of the environment and (b) non-active agent behaviour. Firstly, (a) the stochastic environment introduces randomness to the agent's observations. When the number of observations in a stochastic environment are limited, the different possible observations do not automatically occur equally often. Secondly, (b) the agent can not actively sample from the environment (no active inference) and resolve uncertainty in that way. Missing active inference limits the agent to exclusively update its generative model. In brief, the observations are dependent on the random experiences in the environment because the agent relies on perceptual inference. Fluctuations of the prediction error should then be interpreted as observation of previously not perceived or many unlikely events, and not as poor learning. Unlike model reduction, the PE for model construction decreased step-wise with no sharp fluctuations. The non-existent fluctuations could suggest that model construction is less sensitive to a stochastic and uncertain environment. One possible reason is that the initial model of agent two does not contain causal relations for unlikely observations. On the other hand, in model reduction there would initially exists a causal relation to such an unlikely event and thus many observations of that event would increase its probability in the model. In brief, the results suggest that the missing active inference and stochastic nature as well as uncertainty in the environment might influence the process of model reduction more than model construction. Nevertheless, as summarized in the first generalization, the results also show that both model revision processes can develop an accurate model.

When comparing the average PE development between features and spatial positions (Figures 12 and 14), neither of them seem to influence the agent's learning more. In contrast, the PE seems influenced by higher uncertainty in the environment. In this simulation feature 'red' has the highest uncertainty. Especially for model reduction, the prediction error averaged over 'red' is initially low, because the relation between 'red' and one specific object is less certain. For similar reasons, the PE is initially lower over the 'red' feature when performing model construction. Besides the lower initial PE, the higher uncertainty requires more iterations to reduce the prediction error to almost zero. In the likelihood matrices, the increased uncertainty over feature 'red' causes both agents to establish more causal relation relations for 'red' in combination with all possible spatial positions.

Based on the likelihood matrices, the second generalization is that *the established generative models do not correspond to the 'true' generative model of the environment.* Therefore, even though both agents learn a accurate model of the environment, the established causal relations differ. The causal relations do not only vary *between* the different model revision approaches, but also *within* a specific model revision approach across trials. Most of the differences in causal relations occur over the higher uncertainty feature 'red'. These observations support that the changing causal relations are caused by the stochastic nature of the environment, as explained above.

Summing up, the results are in line with the hypotheses: Although an agent's generative model might not converge to a 'true' model of the environment, the model is still representative. Despite confirming the results, it is essential to note that those conclusions might not be representative of how learning takes place in a complex environment. In any case the results are based on a simplified simulation of an environment. In the preceding discussion, some of the simplifications that potentially influence the results have been mentioned. The next section considers some possible modification of these simplifications such that the findings can be better generalized.

4.1 Possible Modifications

Besides increasing the complexity of the simulation and thus improve its generalization, some other aspects could lead to better representative learning of humans. Two possible adjustments are:

Firstly, a form of active inference can be added to the agent's behaviour. Active inference would allow the agent to sample from the environment actively, instead of being limited to infer the outcomes given a specific policy. The possibility to sample from the environment would improve the psychological plausibility of the simulation concerning how human learning takes place. It is proven that especially learning in children is based on the *exploration* of uncertainty in the environment. When growing older a large amount of exploration is replaced with *exploitation* of high reward states (Gopnik et al., 2015). Explorations require active interference, to allow choosing states that contain the most considerable uncertainty first. Even though exploration is less evident in older humans, successful exploitation of the environment also requires active inference, because it enables choosing the most certain states at first. Besides that, active inference would increase the psychological plausibility; one key aspect of active inference is also that it diminishes the random and stochastic effects in the environment (Friston et al., 2017). Active inference thus limits the possibilities that random observations influence model development. Hence, allowing for active inference could potentially influence model development. That premise is motivated by the results which suggest that model revision processes are influenced in different ways by perceptual inference. Thus, including active inference allows testing how model development changes when the agent can actively determine which states to discover.

Secondly, the current simulation limits agent's to learn the environment either by model reduction or model construction. However, it is unlikely that an agent is limited to reduce or expand its model. Instead, an agent might combine both processes of model revision to develop an accurate model. How model reduction and model construction can be applied together can be illustrated by the language learning example from the beginning. Word 'A' was first associated with meaning 'M1' but then the agent learned that 'A' instead means 'M2'. Changing the word meaning does not succeed by only removing *or* adding one single relation. Instead, a combination of model reduction, removing the current causal connection between 'A' and 'M1', and model constructing, establishing the new causal relation between 'A' and 'M1', is required. Besides language learning, maybe also other processes require a combination of model reduction and model construction to develop an accurate model. For instance, Smith et al. (2019) propose that concept learning requires

a combination of model reduction and model construction. Further Friston et al. (2017) propose that together the model revision processes can resemble the mechanism of sleep. While awake, the process of model construction would add new causal relations. During sleep, model reduction would cut down redundant causal relations (Friston et al., 2017). In conclusion, combining model reduction and model construction could offer new insights about model development and interaction of the two model revision processes.

4.2 Cognitive relevance

Model revision is one approach for prediction error minimisation that is used within the framework of predictive processing. Predictive processing is a cognitive theory explaining the workings of the brain as a prediction machine. Naturally, all processes involved in predictive processing must be useful in a cognitive context to add something to the framework. Earlier mentioned evolutionary and developmental assumptions, e.i. assuming a built-in state space, potentially limit the cognitive relevance of model revision. This section discusses how some of those evolutionary and developmental assumptions could limit approaches of model revision or explains how other assumptions are not plausible.

4.2.1 Evolutionary challenge

In cognitive science a central question regards innateness of knowledge(Perfors, 2012). When working with Bayesian models, this issue does concern the question about: How does creating hypotheses or prior beliefs occur in the first place? Answering this question requires a look at the evolutionary nature of the generative model. Rutar et al. (ress) propose that most researches consider the built-in state space of a model to be evolved through evolution. In their paper, Rutar et al. (ress) describe that the details about how such an evolutionary process can occur currently remain unanswered. However, for now, assume that generative models evolved over generations. Given that assumption, one key question is how such highly complex and increasingly large generative models could be passed on from generation to generation. This question is especially interesting in the context of the model reduction approach, which is based on the assumption that the initial model has an explicit state space. One possibility would be that the explicit state space is passed on over genomes. However, according to the paper by Zador (2019), genomes do not provide enough space to encode a blueprint of the generative model. Thus, unlike in the applied simulation, where the initial model for the model reduction contained all possible states, it seems more likely that the initial model combines multiple states in one node. A different consideration would be that instead of being born with explicit state space, the generative model could encode rules, that make it easier to learn associations faster over time.

4.2.2 Developmental challenge

Assuming that the initial generative model evolved through evolution, the next fundamental question concerns how a model can develop further. Especially debated is the question if learning in the form of adding novel variables can take place. In Section 1 we introduced that this question often gets avoided or receives diverse opinions, such as that learning something new is impossible. The latter statement is for instance supported by Perfors (2012) which argues, that "*any* learner (whether computer or human) must have a built-in hypothesis space" (Perfors, 2012, p.127). Perfors believes that learning anything new is impossible. In her opinion, the only possible way of learning involves *hypothesis testing* or *hypothesis generation*. According to Perfors (2012), hypothesis testing

exists within an already-specified hypothesis space. Within this explicit hypothesis space, hypotheses are tested by deciding which of the assumptions should be preferred. In theory, this process might correspond to model updating because it does not change the structure of the model but 'weighs' hypothesis against each other. The 'weighing' of hypothesis could, therefore, be interpreted as probability updating. Furthermore, a special form of hypothesis testing can involve updating the strength of causal relations. Additionally, the latter learning process, called hypothesis generation, defines moving hypothesis from the implicit hypothesis space to the explicit hypothesis space (Perfors, 2012). If one compares the hidden state with the implicit hypothesis space, then hypothesis generation might correspond to establishing causal relations by moving them to the explicit state space. On the other hand, a *'reverse' hypothesis generation* could be a specific form of model reduction. Thus, hypothesis generation might describe the restricted version of model revision used in this Bachelor thesis, which focuses on revision as adding and removing causal relations. However, the term of hypothesis generation would not be sufficient when considering the original definition of model revision which allows for, i.e., adding novel variables to the hidden or implicit hypothesis space. Perfors (2012) argues that learning as 'learning something novel' would correspond to randomly adding things to the hypothesis space. In her opinion, 'learning something novel' is no form of structural learning, and based on that, Perfors (2012) concludes that nothing can be learned. As discussed by Rutar et al. (ress), this argumentation seems rather unstable because it builds on the following reasoning: "If we cannot establish how X could be learnt, X must be innat" (D'Souza and Karmiloff-Smith, 2016). Thus, Perfors bases her argument on inductive reasoning because it is based on a finite number of particular observations. Inductive reasoning is logically invalid since it is based on a finite amount of examples. Therefore, it provides no good support for her restrictive definition of the learning process. In addition to that, human learning is a complex process which mostly occurs on unsupervised paradigms (Zador, 2019). That means the correct hypothesis is not directly given to the agent but must be inferred from the outcomes. Thus, the agent has to be equipped with a way of searching the implicit hypothesis space while comparing which hypothesis is most likely given the observed outcomes. Searching the hypothesis space would not only be highly computationally expensive but also identifying the correct hypothesis without being provided with a supervised learning technique that provides the correct hypothesis. Therefore, before learning is restricted to moving and testing hypothesis, it is necessary to explain how the correct hypothesis selection can take place given an unsupervised learning paradigm.

5 Conclusion

Neither model reduction or model construction learned the 'true' model of the environment. However, both model revision approaches developed a accurate representation of the environment. When comparing the developed models, the applied model revision approach does not significantly impact the accuracy of the final generative model. Additionally neither of the causal relations show a specific trend depending on what model revision approach was applied. Overall do the results support the hypotheses that even though neither model reduction or model construction learn the 'true' model both can develop an accurate model of the environment. Despite confirming the hypotheses the results provide new insight about how a uncertain and stochastic environment might influence the process of model reduction more than model construction. To determine if the uncertainty and randomness of the environment is the actual cause for the different model developments it is necessary to adjust the simulation. One possible modification to test the origins of the differences between model developments is to e.i. allow for active inference.

6 Future research

In the simulation above, we analysed the model development using a hierarchical generative model. The model used for the simulation has its hierarchical properties in one layer, in the sense of having hyperparameters. Despite hierarchical properties within one layer, a model might consist of multiple layers upon each other. Such hierarchies have a fundamental importance in the predictive processing account because predictions from one level can represent a hypothesis of the lower level (Kwisthout et al., 2016). Thus, an interesting aspect to investigate would be how model revision can develop and maintain a multi-layered hierarchical model. This would require to, e.i. identify conditions when a layer should be removed (model reduction) or when the model should be expanded with an additional layer (model construction), instead of adding information linearly. Particularly challenging would be to determine which variables are related, especially given that human learning follows an unsupervised learning paradigm. Besides classifying necessary conditions that can cause an adjustment of the models hierarchy, another critical point is that hierarchical properties introduce detail in the model (Kwisthout et al., 2016). If the speculation holds that children predict with lower detail, then this assumption imposes additional conditions on the model development. In this situation, one could investigate if during early development model construction should occur more often compared to later in the development stage. Consequently, focusing on these aspects of hierarchical model development could account for higher cognition processes instead of only lower cognition.

Besides that, the simulated model revision process here is based on the assumption that model revision is conditioned to the existence of a high prediction error. However, as pointed out in earlier sections, a high PE is only one possible cause of model revision. As mentioned above, some possible causes of model revision are discussed in the paper by Rutar et al. (ress). In addition to the high PE, the authors suggest that e.i. multiple small successive PE's could combine suggest a misrepresentation of the environment. Thus, in follow up research it might be interesting to compare how different causes influence model revision and how these various conditions impact the development of the model. Besides that, simulating model development under diverse conditions could increase our insight about how external causes influence model revision.

References

Clark, A. (2013). Whatever Next? Predictive Brains, Situated Agents, and the Future of Cognitive Science. *The Behavioral and brain sciences*, 36:1–24.

D'Souza, D. and Karmiloff-Smith, A. (2016). Why a developmental perspective is critical for understanding human cognition. *Behavioral and Brain Sciences*, 39:e122.

Feldman, J. (2016). The simplicity principle in perception and cognition. *Wiley interdisciplinary reviews. Cognitive science*, 7:330–340.

Friston, K. (2003). Learning and inference in the brain. *Neural networks : the official journal of the International Neural Network Society*, 16:1325–52.

Friston, K., Lin, M., Frith, C., Pezzulo, G., Hobson, J., and Ondobaka, S. (2017). Active Inference, Curiosity and Insight. *Neural Computation*, 29:1–51.

Friston, K., Parr, T., and Zeidman, P. (2018). Bayesian model reduction. pages 1–20.

Gładziejewski, P. (2015). Predictive coding and representationalism. *Synthese*, 193:559 – 582.

Gopnik, A., Griffiths, T., and Lucas, C. (2015). When Younger Learners Can Be Better (or at Least More Open-Minded) Than Older Ones. *Current Directions in Psychological Science*, 24:87–92.

Han, J. 2.4.8 kullback-leibler divergence.

Hebb, D. (1949). The Organization of Behavior .A Neuropsychological Theory. *Wiley*, 193.

Kruschke, J. K. (2010). *Doing Bayesian Data Analysis: A tutorial with R and BUGS*. Academic Press.

Kwisthout, J., Bekkering, H., and Rooij, I. (2016). To be precise, the details don't matter: On predictive processing, precision, and level of detail of predictions. *Brain and cognition*, 112.

Kwisthout, J. and Rooij, I. (2019). Computational Resource Demands of a Predictive Bayesian Brain. *Computational Brain and Behavior*.

Otworowska, M., Rooij, I., and Kwisthout, J. (2018). Maximizing entropy of the Predictive Processing framework. pages 1–40.

Perfors, A. (2012). Bayesian Models of Cognition: What's Built in After All? *Philosophy Compass*, 7:127 – 138.

Rutar, D., de Wolff, E., Kwisthout, J., and Rooij, I. (in progress). Learning Generative Models for/from Predictive Processing: Problems and Observations.

Seth, A. (2015). The cybernetic Bayesian brain: From interoceptive inference to sensorimotor contingencies. *Open Mind Project*, pages 1–24.

Smith, R., Schwartenbeck, P., Parr, T., and Friston, K. (2019). An active inference approach to modeling concept learning. pages 1–60.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011). How to Grow a Mind: Statistics, Structure, and Abstraction. *Science*, 331(6022):1279–1285.

Vilares, I. and Kording, K. (2011). Bayesian models: the structure of the world, uncertainty, behavior, and the brain. *Annals of the New York Academy of Sciences*, 1224(1):22–39.

Zador, A. (2019). A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10:1–7.

Appendices

A Simulation code (Python)

The simulation was programmed in `Python` (version 3.7.6). The the following packages were imported:

- `Matplotlib` as `plt`
- `Seaborn` as `sns`
- `Pandas` as `pd`
- `Numpy` as `np`
- `scipy.stats`
- `sklearn.metrics`

In the following subsections all relevant code that is used for the simulation is included. The code is provided with the documentation.

A.1 Environment

```
class Experiment:
    """Class to initialize experimental parameters and basic methods."""

    def __init__(self):
        """Initilization of eperimental parameters."""

        # vector of category probabilities of the objects given the features (almost
            deterministic)
        self.psi_env = np.array(
        [[[0.5,0.25,0.25],[0.00001,0.00001,0.99998],[0.5,0.25,0.25]],
        [[0.00001,0.00001,0.99998],[0.00001,0.00001,0.99998],[0.00001,0.00001,0.99998]],
        [[0.00001,0.99998,0.00001],[0.00001,0.00001,0.99998],[0.00001,0.99998,0.00001]]])

        # vectors with names of features, spatial positions and objects
        self.f_names = ["red", "blue", "yellow"]
        self.l_names = ["left", "center", "right"]
        self.o_names = ["triangle", "circle", "square"]

        self.n_features = len(self.f_names)
        self.n_locations = len(self.l_names)
        self.n_objects = len(self.o_names)

        # Vectors with uniform probability distributions over all objects
        self.objects = np.full(self.n_objects, 1/self.n_objects)
```

```
    # vectors of category probabilities of the features and locations
    self.theta_1 = np.repeat(1/self.n_features,self.n_features)
    self.theta_2 = np.repeat(1/self.n_locations,self.n_locations)

    # Set amount of iterations for model comparison step
    self.size = 10


def environment(self):
    """Sampling from the environment a specific feature, object and location of the
        tile.

    RETURN:
    idx_feature       = index of feature sampled from the environment
    idx_object        = index of object sampled from the environment
    idx_location      = index of spatial position sampled from the environment
    """

    #Draw feature from a categorical distribution
    feature = np.array(multinomial.rvs(p = self.theta_1, n = 1))
    location = np.array(multinomial.rvs(p = self.theta_2, n = 1))

    #Get index of location and feature (starting at 0)
    idx_feature = list(feature).index(1)
    idx_location = list(location).index(1)

    #Draw the object from a categorical distribution given the feature
    obj     = multinomial.rvs(p = self.psi_env[idx_feature, idx_location].flatten(), n
        = 1)

    #Get index of object
    idx_object  = list(obj).index(1)

    return idx_feature, idx_object, idx_location


def agent(self, idx_feature, idx_location):
    """Given a feature sampled from the environment the agent predicts the location of
        a tile and object below that tile
    according to the causal relations that exist. Besides that the agent updates its
        psi and omega.

    INPUT:
    idx_feature       = index of feature sampled from the environment
    idx_location      = index of spatial position sampled from the environment

    OUTPUT:
    idx_predObject    = predicted object below the tile by the agent
```

```
        """

        # The agent predicts the object below the tile given the specific feature and
            predicted location.
        # By either:
        # 1) randomly picking a object or
        if not (self.Bn[idx_feature,idx_location,:].any()):
            obj = multinomial.rvs(p = self.objects, n=1)
        # 2) sampling an object according to the probability distibution of psi agent
        else:
            self.psi_agent[idx_feature, idx_location,:][self.Bn[idx_feature, idx_location
                ,:]==1] = dirichlet.rvs(self.alpha[idx_feature, idx_location,:][self.Bn[
                idx_feature, idx_location,:]==1])[0]
            self.psi_agent[idx_feature, idx_location,:][self.Bn[idx_feature, idx_location
                ,:]==0] = 0.0
            # Normalize values from array float32 to float64 to avoid errors with the
                multinomial distribution
            self.psi_agent[idx_feature, idx_location,:] = np.asarray(self.psi_agent[
                idx_feature, idx_location,:]).astype('float64')
            self.psi_agent[idx_feature, idx_location,:] = self.psi_agent[idx_feature,
                idx_location,:] / np.sum(self.psi_agent[idx_feature, idx_location,:])
            # Choose a object using categoritcal distribution
            obj = multinomial.rvs(p = self.psi_agent[idx_feature, idx_location, :], n=1)
        idx_predObject  = list(obj).index(1)

        return idx_predObject
```

A.2 Analyzation methods

```
class AnalysisModel(Experiment):
    """Methods to analyse the outcomes and generative models created using the experiment.
        """

    def __init__(self):
        """Initialization of the analysis techniques for experiment."""

        # Inherit defined experimental parameters from Class Experiment
        Experiment.__init__(self)

        # True and predicted labels per feature and location combination
        # Rows indicate location and columns the feature y_true[location][feature]
        self.y_trueAll = [[[] for _ in range(self.n_features)] for _ in range(self.
            n_locations)]
        self.y_predAll = [[[] for _ in range(self.n_features)] for _ in range(self.
            n_locations)]

        # True and predicted objects for confusion matrix
```

```
        self.y_trueObject = []
        self.y_predObject = []

        # number of test trials to run the model
        self.n_test = 1

    def average(self, lst):
        """Calculate average of list

        INPUT:
        lst   = list to calculate the average over

        RETURN:
        (float) average over given list
        """

        return sum(lst) / len(lst)

    def graphPredictionErrorAverage_Feature(self, pre_errors, name):
        """Create a plot for each feature which shows prediction error of spatial
            locations in different colours

        INPUT:
        pre_errors = multi-dimensional array with all prediction errors
        name = name of model revision process (used for saving the plots)

        OUTPUT:
        Single graph; line plot for each feature value averaged over the spatial positions
        """

        average_perF = [[] for _ in range(self.n_features)]

        #Average over different features
        k=0
        for i in range(self.n_features):
            # find longest list
            longest = 0
            for j in range(self.n_locations):
                if len(pre_errors[i][j]) > longest:
                    longest = len(pre_errors[i][j])
            # calculate the mean over each feature
            for p in range(longest):
                mean = []
                for j in range(self.n_locations):
                    if len(pre_errors[i][j]) > p:
                        mean.append(pre_errors[i][j][p])
                average_perF[i].append(self.average(mean))
```

```
        plt.plot(average_perF[0], '-r', average_perF[1], '-b', average_perF[2], '-y',
            linewidth=1.5)
        plt.title("Average PE for features ({})".format(name), fontsize=13)
        plt.xlabel("Number of Iterations")
        plt.ylabel("Size of Prediction Error")
        plt.legend(self.f_names)

        # remove the comment on the following line if you wish to save the figure to local
            machine:
        # plt.savefig(name + "PE_AverageFeature.png")

        plt.show()

    def graphPredictionErrorAverage_Location(self, pre_errors, name):
        """Create a plot for each feature which shows prediction error of spatial
            locations in different colours

        INPUT:
        pre_errors = multi-dimensional array with all prediction errors
        name = name of model revision process (used for saving the plots)

        OUTPUT:
        Single graph; line plot for each spatial position value averaged over the features
        """

        average_perS = [[] for _ in range(self.n_locations)]

        #Average over different features
        k=0
        for i in range(self.n_locations):
            # find longest list
            longest = 0
            for j in range(self.n_features):
                if len(pre_errors[j][i]) > longest:
                    longest = len(pre_errors[j][i])
            for p in range(longest):
                mean = []
                for j in range(self.n_features):
                    if len(pre_errors[j][i]) > p:
                        mean.append(pre_errors[j][i][p])
                average_perS[i].append(self.average(mean))

        plt.plot(average_perS[0], '-r', average_perS[1], '-b', average_perS[2], '-y',
            linewidth=1.5)
        plt.title("Average PE for spatial positions ({})".format(name), fontsize=13)
        plt.xlabel("Number of Iterations")
        plt.ylabel("Size of Prediction Error")
        plt.legend(self.l_names)
```

```
    # remove the comment on the following line if you wish to save the figure to local
        machine:
    # plt.savefig(name + "PE_AverageLocation.png")

    plt.show()

def plot_confusion_matrix(self,cm, y, name):
    """Given a python confusion matrix plot its heatmap

    cm   = confusion matrix
    y    = labels of the predicted and actual classes
    name = name of model revision process
    """

    # normalize confusion matrix (2 options)
    # 1) dividing over axis=0 to get precision (fraction of class-k predictions that
        have ground truth label k)
    cm = cm / cm.astype(np.float).sum(axis=0)
    # 2) dividing over axis=1 to get how many samples per class have received their
        correct label
    # cm = cm / cm.astype(np.float).sum(axis=1)

    # create pandas dataframe with corresponding labels
    df_cm = pd.DataFrame(cm, index = [i for i in y], columns = [i for i in y])
    plt.figure()

    # get heatmap using pandas dataframe
    ax = sns.heatmap(df_cm, annot=True, cmap = 'Blues', linewidths=.5)

    #set bottom and top to show full heatmap
    bottom, top = ax.get_ylim()
    ax.set_ylim(bottom + 0.5, top - 0.5)

    # set title and axis labels of matrix
    plt.title('Confusion matrix for ' + name)
    plt.xlabel('Predicted class')
    plt.ylabel('Actual class')

    # remove the comment on the following line if you wish to save the figure to local
        machine:
    # plt.savefig(name + "CM.png")

    # print figure in jupyter notebook
    plt.show()

def likelihoodMatrix(self, psi, name):
```

```
"""Creating likelihood mapping between hidden states and outcomes for each
    location and feature combination
Code based on the following example to generate heatmaps side by side: https://
    stackoverflow.com/questions/42712304/seaborn-heatmap-subplots-keep-axis-ratio-
    consistent


INPUT:
psi = [n_feature * n_location* n_objects]: indicating the likelihood for a objects
    outcome given
     a specific feature and location
name = ['environment', 'Model reduction', 'Model construction']: name of the topic
    for which the likelihood
      mapping is created

OUTPUT:
returning a subfigure 3x1 for each feature a likelihood matrix over all locations
"""

# Create a subplot to save all heatmaps
fig,(ax1,ax2,ax3, axcb) = plt.subplots(1,4, figsize = (10,5), gridspec_kw={'
    width_ratios':[1,1,1,0.08]})
ax1.get_shared_y_axes().join(ax2,ax3)

data_feature1 = psi[0]
df_feature1 = pd.DataFrame(data_feature1, columns = self.o_names, index = self.
    l_names)
g1 = sns.heatmap(df_feature1, annot=True, cmap="Blues", cbar=False,ax=ax1, vmin=0,
     vmax=1, linewidths=.5)
g1.set_ylabel('Spatial positions')
g1.set_xlabel('Objects')
g1.set_title('Red')
#set bottom and top to show full heatmap
bottom, top = ax1.get_ylim()
ax1.set_ylim(bottom + 0.5, top - 0.5)

data_feature2 = psi[1]
df_feature2 = pd.DataFrame(data_feature2, columns = self.o_names, index = self.
    l_names)
g2 = sns.heatmap(df_feature2, annot=True, cmap="Blues", cbar=False,ax=ax2, vmin=0,
     vmax=1, linewidths=.5)
g2.set_ylabel('')
g2.set_xlabel('Objects')
g2.set_title('Blue')
g2.set_yticks([])
#set bottom and top to show full heatmap
bottom, top = ax2.get_ylim()
ax2.set_ylim(bottom + 0.5, top - 0.5)
```

```
        data_feature3 = psi[2]
        df_feature3 = pd.DataFrame(data_feature3, columns = self.o_names, index = self.
            l_names)
        g3 = sns.heatmap(df_feature3, annot=True, cmap="Blues", ax=ax3, cbar_ax=axcb, vmin
            =0, vmax=1, linewidths=.5)
        g3.set_ylabel('')
        g3.set_xlabel('Objects')
        g3.set_title('Yellow')
        g3.set_yticks([])
        #set bottom and top to show full heatmap
        bottom, top = ax3.get_ylim()
        ax3.set_ylim(bottom + 0.5, top - 0.5)

        # remove the comment on the following line if you wish to save the figure to local
            machine:
        # plt.savefig(name + "LM.png")

        fig.suptitle("Features")
        plt.show()
```

A.3 Model Reduction

```
class ModelReduction(Experiment):
    """Run experiment using model reduction process."""

    def __init__(self):
        """Initialization of model reduction parameters/variables."""

        # Inherit defined experimental parameters from Class Experiment
        Experiment.__init__(self)

        # Concentration vectors
        self.alpha = np.full((self.n_features, self.n_locations, self.n_objects), 1)

        # List for prediction errors
        self.pre_errorsRule = [[[] for _ in range(self.n_locations)] for _ in range(self.
            n_features)]
        self.pre_errorsTotal = []

        # Probability distributions of hidden states
        self.psi_agent = np.full((self.n_features, self.n_locations, self.n_objects), 1/
            self.n_objects)

        #Connections between the hidden states and the outcomes. All 1 if model reduction
        self.Bn = np.full((self.n_features, self.n_locations, self.n_objects), 1)
```

```
def modelReductionStepBn(self, idx_feature, idx_predLocation, idx_predObject):
    """Perform a model reduction step of the causal relations between the rule and the
        objects below the tile.

    INPUT:
    idx_feature      = feature sampled from the environment
    idx_predLocation = predicted location of feature by the agent
    idx_predObject   = predicted object below the tile by the agent

    OUTPUT:
    Bn_reduced       = reduced matrix indicating causal relations of rule
    psi_reduced      = probabilities over the rules given the reduced causal relations
        of Bn_reduced
    """

    psi_reduced = np.copy(self.psi_agent)
    Bn_reduced = np.copy(self.Bn)

    Bn_reduced[idx_feature, idx_predLocation, idx_predObject] = 0

    # Update the probabilities in the reduced psi matrix
    # 1) If there do exist causal relations given a specific feature and the predicted
        object then use dirichlet distribution
    # 2) Otherwise set all probabilities of psi to zero.
    if Bn_reduced[idx_feature, idx_predLocation,:].any() > 0:
        psi_reduced[idx_feature, idx_predLocation,:][Bn_reduced[idx_feature,
            idx_predLocation,:]==1] = dirichlet.rvs(self.alpha[idx_feature,
            idx_predLocation,:][Bn_reduced[idx_feature, idx_predLocation,:]==1])[0]
        psi_reduced[idx_feature, idx_predLocation,:][Bn_reduced[idx_feature,
            idx_predLocation,:]==0] = 0
    else:
        psi_reduced[idx_feature, idx_predLocation,:] = 0

    return Bn_reduced, psi_reduced

def modelComparisonPsi(self, psi_alternative, idx_feature, idx_location):
    """Compare two probability distributions of psi and determine which represents the
        environment better
    (returns lower prediction error).

    INPUT:
    psi_alternative  = alternative probability distribution of psi
    idx_feature      = feature sampled from the environment
    idx_location     = location of the feature sampled from the environment

    OUTPUT:
```

```
        1               = alternative psi is prefered
        0               = original psi is prefered
        """

        # Prediction error array to store the prediction errors for the original (row 1)
            and alterntive model (row 2)
        preErrorsArray = np.zeros((2,self.size))

        for i in range(0,self.size):
            if not (self.psi_agent[idx_feature,idx_location,:].any()>0):
                psi_random = multinomial.rvs(p=self.objects, n=1)
                preErr_original = entropy(psi_random, self.psi_env[idx_feature,
                    idx_location], base=None)
            else:
                preErr_original = entropy(self.psi_agent[idx_feature, idx_location],self.
                    psi_env[idx_feature, idx_location], base=None)
            if not (psi_alternative[idx_feature,idx_location,:].any()>0):
                psi_random = multinomial.rvs(p=self.objects, n=1)
                preErr_alternative = entropy(psi_random, self.psi_env[idx_feature,
                    idx_location], base=None)
            else:
                preErr_alternative = entropy(psi_alternative[idx_feature, idx_location],
                    self.psi_env[idx_feature, idx_location], base=None)

            preErrorsArray[0,i] = preErr_original
            preErrorsArray[1,i] = preErr_alternative

        # The alternative model is simpler and therefore prefered if the prediction error
            is equal
        if np.mean(preErrorsArray[0,:]) >= np.mean(preErrorsArray[1,:]):
            return 1
        return 0


    def trial(self, n_iter):
        """Updating the agent's generative model.

        INPUT:
        n_iter          = number of times that the agent's model is updated

        OUTPUT:
        pre_errorsRule  = matrix with the predictions error of each combination between
            feature, location and object.
        """

        for i in range(0, n_iter):

            #Get feature and observation from environment
```

```
idx_feature, idx_object, idx_location = Experiment.environment(self)

#Prediction of the agent
idx_predObject =  Experiment.agent(self, idx_feature, idx_location)

if self.Bn[idx_feature, idx_location, idx_predObject] == 1:
    Bn_reduced, psi_reduced = self.modelReductionStepBn(idx_feature,
        idx_location, idx_predObject)
    if self.modelComparisonPsi(psi_reduced, idx_feature, idx_location):
        self.psi_agent = np.copy(psi_reduced)
        self.Bn = np.copy(Bn_reduced)

#Calculate prediction error using relative entropy (Kullback–Leibler
    divergence)
if not (self.Bn[idx_feature,idx_location,:].any()):
    agentPrediction = np.full(self.n_objects, 0)
    agentPrediction[idx_predObject] = 1
    pre_err = entropy(agentPrediction,self.psi_env[idx_feature, idx_location],
         base=None)
else:
    pre_err = entropy(self.psi_agent[idx_feature, idx_location],self.psi_env[
        idx_feature, idx_location], base=None)

#pre_err = Experiment.calculatePredictionErr(self, idx_feature, idx_location,
    idx_predLocation, idx_predObject)
self.pre_errorsRule[idx_feature][idx_location].append(pre_err)
self.pre_errorsTotal.append(pre_err)

#Updating hyperparameters
self.alpha[idx_feature, idx_location, idx_object] += 1
```

A.3.1 Execute Model Reduction Trial

```
# Initialization of classes
Exp = Experiment()
MReduction = ModelReduction()

# Likelihood matrix of psi agent before training
Analysis.likelihoodMatrix(MReduction.psi_agent, "Before_ModelReduction")

# Training of model using model reduction
MReduction.trial(250)

# Plot the prediction errors and liklihood matrices
# ...
```

A.4 Model Construction

```
class ModelConstruction(Experiment):
    """Run experiment using model construction process."""

    def __init__(self):
        """Initialization of model construction parameters/variables."""

        # Inherit defined experimental parameters from Class Experiment
        Experiment.__init__(self)

        # Concentration vectors
        self.alpha = np.full((self.n_features, self.n_locations, self.n_objects), 1)
        self.beta = np.full((self.n_features, self.n_locations), 1)

        # List for prediction errors
        self.pre_errorsRule  = [[[] for _ in range(self.n_locations)] for _ in range(self.
            n_features)]
        self.pre_errorsTotal = []

        # Probability distributions of hidden state
        self.psi_agent = np.full((self.n_features, self.n_locations, self.n_objects), 0.0)

        #Connections between the hidden states and the outcomes. All 1 if model reduction
        self.Bn = np.full((self.n_features, self.n_locations, self.n_objects), 0)

        # Set amount of iterations for model comparison step
        self.size = 10

    def modelConstructionStepBn(self,idx_feature, idx_location, idx_predObject):
        """Perform a model construction step of the causal relations between the rule and
            the objects below the tile.

        INPUT:
        idx_feature         = feature sampled from the environment
        idx_predLocation    = predicted location of feature by the agent
        idx_predObject.     = predicted object below the tile by the agent

        OUTPUT:
        Bn_reduced          = reduced matrix indicating causal relations between features
            and spatial locations
        psi_reduced         = probabilities between features and spatial locations given the
            reduced causal relations of Dm_reduced
        """

        psi_expanded = np.copy(self.psi_agent)
        Bn_expanded = np.copy(self.Bn)
```

```
    # if there is no relation yet for the predicted object, consider to expand a
        relation there
    # otherwise choose a random object index to construct a alternative model
    if Bn_expanded[idx_feature, idx_location, idx_predObject] == 0:
        indexObject = idx_predObject
    else:
        psi_random = multinomial.rvs(p=self.objects, n=1)
        indexObject = list(psi_random).index(1)

    Bn_expanded[idx_feature, idx_location, indexObject] = 1
    psi_expanded[idx_feature, idx_location,:][Bn_expanded[idx_feature, idx_location
        ,:]==1] = dirichlet.rvs(self.alpha[idx_feature, idx_location,:][Bn_expanded[
        idx_feature, idx_location,:]==1])[0]
    psi_expanded[idx_feature, idx_location,:][Bn_expanded[idx_feature, idx_location
        ,:]==0] = 0.0
    return Bn_expanded, psi_expanded

def modelComparisonPsi(self, psi_alternative, idx_feature, idx_location):
    """Compare two probability distributions of psi and determine which represents the
         environment better
    (returns lower prediction error).

    INPUT:
    psi_alternative  = alternative probability distribution of psi
    idx_feature      = feature sampled from the environment
    idx_location     = location of the feature sampled from the environment

    OUTPUT:
    1                = alternative psi is prefered
    0                = original psi is prefered
    """

    # Prediction error array to store the prediction errors for the original (row 1)
        and alterntive model (row 2)
    preErrorsArray = np.zeros((2,self.size))

    for i in range(0,self.size):
        if not (self.psi_agent[idx_feature,idx_location,:].any()>0):
            psi_random = multinomial.rvs(p=self.objects, n=1)
            preErr_original = entropy(psi_random, self.psi_env[idx_feature,
                idx_location], base=None)
        else:
            preErr_original = entropy(self.psi_agent[idx_feature, idx_location],self.
                psi_env[idx_feature, idx_location], base=None)
        if not (psi_alternative[idx_feature,idx_location,:].any()>0):
            psi_random = multinomial.rvs(p=self.objects, n=1)
            preErr_alternative = entropy(psi_random, self.psi_env[idx_feature,
                idx_location], base=None)
```

```
        else:
            preErr_alternative = entropy(psi_alternative[idx_feature, idx_location],
                self.psi_env[idx_feature, idx_location], base=None)

        preErrorsArray[0,i] = preErr_original
        preErrorsArray[1,i] = preErr_alternative

    # The alternative model is more complex and therefore not prefered if the
        prediction error is equal
    if np.mean(preErrorsArray[0,:]) > np.mean(preErrorsArray[1,:]):
        return 1
    return 0

def trial(self, n_iter):
    """Updating the agent's generative model.

    INPUT:
    n_iter          = number of times that the agent's model is updated

    OUTPUT:
    pre_errorsRule  = matrix with the predictions error of each combination between
        feature, location and object.
    """

    for i in range(0,n_iter):

        # Get feature and observation from environment
        idx_feature, idx_object, idx_location = Experiment.environment(self)

        # Prediction of the agent
        idx_predObject =  Experiment.agent(self,idx_feature, idx_location)

        #if self.Bn[idx_feature, idx_location, idx_predObject] == 0:
        if 0 in self.Bn[idx_feature, idx_location, :]:
            Bn_expanded, psi_expanded = self.modelConstructionStepBn(idx_feature,
                idx_location, idx_predObject)
            if self.modelComparisonPsi(psi_expanded, idx_feature, idx_location):
                self.psi_agent = np.copy(psi_expanded)
                self.Bn = np.copy(Bn_expanded)

        #Calculate prediction error using relative entropy (Kullback–Leibler
            divergence)
        if not (self.Bn[idx_feature,idx_location,:].any()):
            agentPrediction = np.full(self.n_objects, 0)
            agentPrediction[idx_predObject] = 1
            agentPrediction[agentPrediction == 0] = 0.0000001
            pre_err = entropy(agentPrediction,self.psi_env[idx_feature, idx_location],
                 base=None)
```

```
        else:
            agent = self.psi_agent[idx_feature, idx_location].copy()
            agent[self.Bn[idx_feature,idx_location,:]==0] = 0.0000001
            agent[self.Bn[idx_feature,idx_location,:]==1] = 0.0000002
            pre_err = entropy(agent,self.psi_env[idx_feature, idx_location], base=None
                )
            #pre_err = entropy(self.psi_agent[idx_feature, idx_location],self.psi_env[
                idx_feature, idx_location], base=None)


        #pre_err = Experiment.calculatePredictionErr(self, idx_feature, idx_location,
            idx_predLocation, idx_predObject)
        self.pre_errorsRule[idx_feature][idx_location].append(pre_err)
        self.pre_errorsTotal.append(pre_err)

        #Updating hyperparameters
        self.alpha[idx_feature, idx_location, idx_object] += 1
```

A.4.1 Execute Model Construction Trial

```
# Initialization of classes
Exp = Experiment()
MConstruction = ModelConstruction()

# Likelihood matrix of psi agent before training
Analysis.likelihoodMatrix(MConstruction.psi_agent, "Before_ModelConstruction")

# Training of model using model reduction
MConstruction.trial(250)

# Plot the prediction errors and likelihood matrices
# ...
```