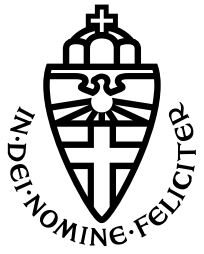
RADBOUD UNIVERSITY NIJMEGEN



FACULTY OF SOCIAL SCIENCE

Nature versus Nurture:

EVOLUTIONARY AND DEVELOPMENTAL ORIGINS OF HYPERPARAMETERS IN INFANT AGENTS

THESIS BSC ARTIFICIAL INTELLIGENCE

Author:
Ellen SCHRADER
(s4693949)

Supervisor:
Danaja RUTAR

Second Supervisor:
Johan KWISTHOUT (PHD)

January 28, 2020

Contents

Abstract

The increasing evidence for an organisation of the brain based on the principles of Bayesian inference has led to a rise in popularity of predictive processing (Kwisthout, Bekkering, & van Rooij, 2017). Most research within the predictive processing framework focuses on explaining cognitive phenomena within the adult brain, missing to explain how these phenomena arise and develop in infant agents. We propose a shift of focus in the predictive processing literature towards development. Before we can investigate development, the initial state of generative models needs to be understood. This research attempts to make a starting step by investigating developmental and evolutionary origins of hyperparameters within infant agents.

Keywords— predictive processing, infant agents, hyperparameters, development, evolution, bayesian modelling

1 Introduction

The predictive processing (PP) framework is commonly recognised as one of the leading theories of cognition. The popularity of PP stems not only from its unifying character but also from its claim to explain both lower- and higher-level cognitive processes throughout the entire cortex (Clark, 2013). Even though there is already a great deal of research on how PP accounts for many lower-level processes, there is plenty more research needed on higher cognitive phenomena such as language and thought (Kwisthout et al., 2017) and specifically on how these phenomena arise and develop in infant agents. Predictive processing is currently faced with a new conundrum: *development*.

Up until now, predictive processing was mainly concerned with explaining cognitive phenomena within the adult brain, not taking into account how these phenomena come into being and develop during the agent's life span. For example, while the predictive processing framework can explain lower-level vision in adult agents (Kwisthout et al., 2017), it does not yet account for the development from poor to adult-like colour vision during the first few months of infancy. This presents PP with the question of how the agent's generative model can capture the increase in complexity of cognitive phenomena such as vision over time. Therefore, the next challenge for predictive processing is to explain the emergence and development of cognitive processes within an infant's brain.

However, before we can address the question of *how* generative models develop from infancy on, the initial state of generative models needs to be investigated. This research focuses on the initial state of hyperparameters within the generative models and explores developmental and evolutionary origins of hyperparameters.

1.1 Predictive Processing

The predictive processing framework is gaining more and more popularity as a unified theory of cognition, perception, and action (Clark, 2013; Kwisthout et al., 2017). The growing popularity of PP has its roots in the increasing body of evidence supporting an organisation of the brain based on the principles of predictive processing (Kwisthout & van Rooij, 2019) and the possibility of describing many cognitive phenomena at different levels of cognition as a form of Bayesian inference (Kwisthout & van Rooij, 2019).

The key idea behind predictive processing is that cognitive phenomena can be explained by and come about through prediction error minimisation. The idea that human behaviour and cognition arise from error minimisation is not new. Already half a century ago, the British psychiatrist and cyberneticist W. Ross Ashby stated that "[t]he whole function of the brain is summed up in: error correction"(Clark, 2013, p.1; from W.R. Ashby, 1930-1972). Humans constantly make predictions about the world based on prior hypotheses with different levels of accuracy. For example, given the sensory information of seeing an animal with stripes, one might predict a *tiger*, a *zebra* or a *cat* with some probability. How the probability mass is distributed over the different hypotheses depends on the agent's prior hypotheses. A person living in Europe will put most of the probability mass on the hypotheses *cat*, whereas somebody from Africa might predict a *zebra* with higher probability. To minimise the discrepancy between the prediction made by the agent and the actual observation, a prediction error is fed back through a hierarchy of increasingly abstract hypotheses, where hypotheses on one level of the hierarchy are described by the predictions of the superordinate levels (Kwisthout et al., 2017) (see Figure 1). The agent then tries to improve its predictions and minimise future prediction errors by either *revising its hypotheses* through changes in the probability distribution, *revising the model*, gathering *new observations*, *intervening in the environment* or as recently proposed by Kwisthout et al. *modulating the level of detail* (Kwisthout et al., 2017). This research will mainly focus on prediction error minimisation by gathering new observations.

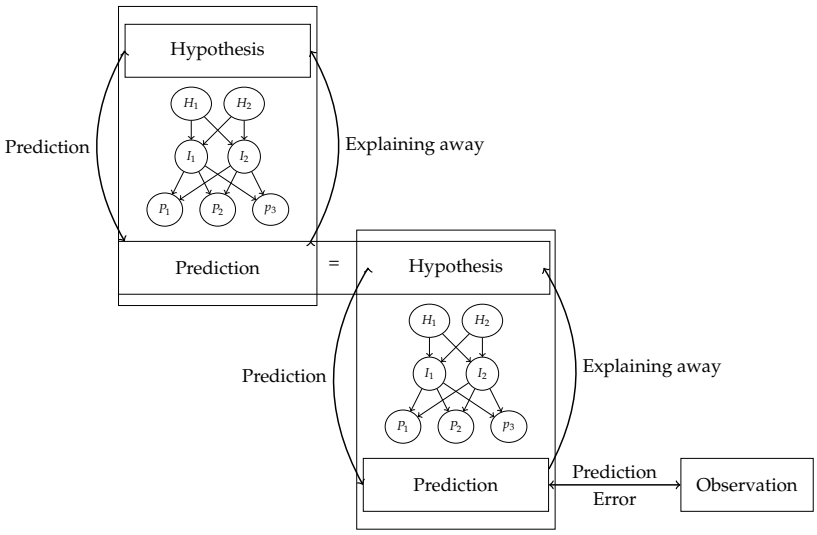


Figure 1: This figure is greatly inspired by Kwisthout et al. (Kwisthout et al., 2017). In predictive processing the brain is seen as a *hierarchical hypothesis-testing mechanism*. When making a prediction the agent starts at the highest level of the hierarchy and makes a prediction. This prediction is then taken as the hypothesis on the next level of the hierarchy and a new prediction is made based on that hypothesis. This process continues until the last level of the hierarchy where the final prediction is made. The final prediction is compared to the agent's observation. The prediction error is calculated and fed back through the hierarchy of hypotheses. The agent then tries to explain away the prediction error by, for example, updating its hypotheses.

To illustrate the process of prediction error minimisation, consider the striped animal example again. If the prediction made by the agent differs from the actual observation, a high prediction error is fed back through the hierarchy. The prediction error indicates the amount of wrongly predicted information in the hierarchy. A high prediction error does not have to be an indication of an unexpected outcome but can also be due to the inherent stochasticity of the environment. The prediction error consists of *reducible* and *irreducible uncertainty*. Reducible uncertainty is caused by "misrepresentations" or uncertainty in the generative model and can be eliminated through learning, for example, by gathering new observations. Irreducible uncertainty, on the other hand, is caused by the stochastic nature of the environment and can thus not be eliminated through learning (Kwisthout et al., 2017). By gathering increasingly more observations, the reducible uncertainty of an agent declines and the prediction error approximates the irreducible uncertainty. This implies that even if the generative model is identical to the model of the environment, the prediction error is not zero due to the stochastic nature of the environment.

The Kullback-Leibler (KL) divergence is often used to compute how well the agent approximates the true distribution. The degree to which the prediction approximates the true distribution can be computed as follows:

$$D_{KL}(P_{obs}|P_{pre}) = \sum_{x \in Obs} P_{obs}(x) log\Big(\frac{P_{obs}(x)}{P_{pre}(x)}\Big), \tag{1}$$

where P_{obs} corresponds to the actual probability distribution over all possible outcomes and P_{pre} corresponds to the predicted probability distribution over the outcomes[1].
For instance, the agent might predict a *cat* with 90% probability and a *zebra* and a *tiger* with 5% probability, whereas the actual distribution of the environment over the hypotheses is 80% for *cat*, 15% *zebra* and 5% for *tiger*. This prediction would result in a KL divergence of ≈ 0.071. The agent can decrease this distance by trying to explain away its prediction error, for example, by updating its prior expectation over the hypotheses, i.e. distributing more probability mass on the hypothesis *zebra*. The next time the agent sees a striped animal, it is more likely to predict a *zebra*. This process of prediction error minimisation enables the agent to learn changes in the stochastic nature of the environment, which is of high importance especially in highly volatile environments.

1.1.1 (Hyper)priors and Hyperparameters

Which hypotheses an agent assigns most of its probability mass to, given some sensory information, is highly dependent on its prior beliefs. These beliefs are represented in the generative model by priors and hyperpriors.

[1]Note that the agent does not have access to the actual distribution of the environment. The prediction error described here is a measure of the quality of the model and not a measure of the signal used by the agent to update its generative model.

Priors are first-order information an agent has about the world. To illustrate this, suppose an agent tries to predict the source of the sensory information originating from a striped animal. Here, the agent's belief that the prediction can be modelled by a categorical distribution with the category probability distribution θ is called a *prior.* The agent might also have some additional belief over the shape of θ. For example θ could be sampled from a Dirichlet distribution with concentration vector α. The belief the agent has over the shape of θ is called a *hyperprior* and the parameters of the hyperprior e.g. α are called *hyperparameters* (see Figure 2). Hyperpriors represent, as the name suggests, knowledge or beliefs an agent has concerning the shape of its priors (second-order information).

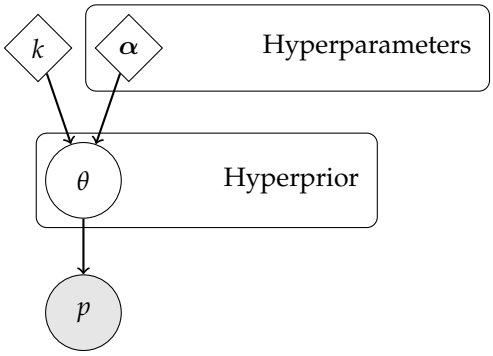


Figure 2: Illustration hyperpriors and hyperparameters

Hyperparameters can be learned throughout the life of an agent, for example by gathering new observations. To illustrate this, consider the hyperparameter α with the form $\alpha = \begin{pmatrix} pc_{cat} & pc_{zebra} & pc_{tiger} \end{pmatrix}$, where pc represents the number of pseudo-observations over one hypothesis also called pseudo-counts. If all hypotheses have a pseudo-count of one $\alpha = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$, the prior is uninformed; the agent does not have any prior belief about the likelihood of each hypothesis and the probability mass is distributed equally over all hypotheses (see Figure 3c). The number of total pseudo-observations indicates how certain the agent is in the distribution over the hypotheses. For example, a concentration vector with the shape $\alpha = \begin{pmatrix} 10 & 1 & 1 \end{pmatrix}$ implies high certainty in the hypothesis *cat* being more likely than the hypotheses *zebra* or *tiger*, whereas $\alpha = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix}$ indicates that the agent is not very certain that *cat* is more likely than *zebra* or *tiger*. Figure 3 exemplifies the proportional relationship between the total number of pseudo-observations and the certainty of an agent. For $\alpha = \begin{pmatrix} 10 & 1 & 1 \end{pmatrix}$ most of the probability mass is focused on the left corner displaying high certainty in the hypothesis *cat* (Figure 3a), whereas for $\alpha = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix}$ the probability mass is not focused on one point but is spread around the left corner of the triangle indicating some uncertainty (Figure 3b).

(a) $\alpha = \begin{pmatrix} 10 & 1 & 1 \end{pmatrix}$ **(b)** $\alpha = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix}$ **(c)** $\alpha = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$

Figure 3: Visualisation of the Dirichlet distributions for different concentration vectors α. The left corner represents the hypothesis *cat*, the centre the hypothesis *zebra*, and the right corner the hypothesis *tiger*.

When making a new observation, the agent updates the hyperparameter α by adding one observation to the pseudo-count of the corresponding hypothesis. For example, when the agent observes a *tiger*, it adds one extra observation to the pseudo-count of the hypotheses *tiger*. The process of updating hyperparameters by increasing the pseudo-count can be seen as a form of knowledge acquisition, where the number of pseudo-observations represent the current state of knowledge.

1.2 Nature vs. Nurture

The possibility of representing knowledge through hyperparameters presents us with the questions whether the initial state of hyperparameters is uninformed or whether it represents some form of innate knowledge.

Scientists have been discussing the question whether nature or nurture is of higher importance for determining human behaviour since centuries. Even ancient Greek philosophers, such as Aristotle and Plato, have already debated whether knowledge is innate or learned through experience (Lewkowicz, 2011). Plato reasoned that humans

are born with innate ideas, implying that knowledge is *a priori* to the start of an individual's life. Aristotle, on the other hand, disagreed with the idea of innate knowledge and argued that humans are born with a *tabula rasa*, a blank slate suggesting that knowledge can only be gained *a posteriori* through experience. Here, Plato embodies nature and Aristotle nurture.

In general, nature refers to some innate knowledge or "pre-wiring" through the genes or some higher being [2]. Nurture, on the other hand, assumes a blank slate and believes that human behaviour is solely influenced by external factors post conception such as experience, exposure and learning of an individual (McLeod, 2018).

Since then the nature vs nurture debate has been going on, continually varying in which of the two is the leading principle, never converging to an answer. However, recently scientists started to recognise the importance of both nature and nurture. The Canadian psychologist Donald Hebb compared the question of nature vs nurture "to asking what contributed more to the area of a rectangle, the length or the width" (Meaney, 2001, p.1). It is impossible to attribute either length OR width full responsibility for the area of the rectangle, nor is it possible to explain the area of a rectangle by looking at length AND width independently. The area of the rectangle can only be described by the interaction between the two. The same holds for nature and nurture. To paraphrase Micheal Meaney: Life does not emerge through nature OR nurture, nor through nature AND nurture, but through the interaction between the two (Meaney, 2001). Take language, for instance, the ability to learn a language is innate, but the language itself is learned through an individual's experience and exposure to the language (Zador, 2019). Any form of development, be it, for example, in the form of learning a language, "emerges from a constant dialogue between gene and environment" (Meaney, 2001, p.2). This constant interplay of nature and nurture needs to be considered when trying to model cognition and specifically development.

1.2.1 Nature vs Nurture in Predictive Processing

After the popularity of symbolic artificial intelligence (AI) declined during the end of the last century, the focus of AI has shifted towards "tabula rasa networks" (Zador, 2019, p.2) such as Artificial Neural Networks. This shift from nature-based systems to systems relying solely on nurture to learn network parameters has yielded considerable advances in the attempt of modelling cognition for example in modelling vision. However, recently nature based approaches have made a comeback. Researchers such as Anthony Zador are advertising to take inspiration from animal brains and introduce innate knowledge to "tabula-rasa networks". This movement is based on the realisation that nurture alone is not enough to imitate the complexity of cognitive processes.

The insufficiency of nurture also applies when modelling cognition under the framework of predictive processing. While attempting to model cognitive processes, the focus should not only lie on knowledge learned from observations but also on the representation of innate knowledge. In general, knowledge acquired by nature can be represented by the initial structure and settings of the generative model. In contrast, knowledge acquired by nurture can be learned from observations and information in the environment (Gruber, 2013).

In the context of hyperparameters, the initial value represents innate knowledge inherited from evolution, and the updating rules represent knowledge acquired during the learning process. For a generative model consisting solely of the beta distribution with hyperparameters α and β, this would imply that the initial values of α and β represent innate knowledge and the updating rules of α and β, e.g. increasing the appropriate hyperparameter by one after every observation, represent knowledge learned throughout development.

This research investigates the effects of nature and nurture in generative models by looking at hyperparameters. The choice to research evolutionary and developmental origins of hyperparameters instead of investigating the structure of the generative model was based on Anthony Zador's suggestion that constraints on wiring and learning rules decoded in the genome can be set by hyperparameters (Zador, 2019). Hyperparameters have additional advantages over other ways of knowledge representation within the generative model. Both the distinction between knowledge acquired from nature and knowledge acquired from nurture and their interplay are straightforward to observe in hyperparameters. The initial state of the hyperparameters that is the number of initial pseudo-counts represents innate knowledge. Learned knowledge is computed by subtracting the innate pseudo-counts from the total pseudo-counts after learning occurred. The interplay between innate and learned knowledge can be evaluated by comparing the total pseudo-counts to the learned and the innate pseudo-counts and by analysing whether the effects of nature and nurture strengthen or weaken each other. Therefore, hyperparameters form a good starting point to investigate evolutionary and developmental origins of generative models.
This research attempts to answer the question *whether infant agents are born with evolved hyperparameters or whether hyperparameters evolve throughout development*. We hypothesise, based on the new insights in the nature vs nurture debate, that not evolution or environment independently influence the state of hyperparameters, but the interplay of evolution and environment together.

[2] often believed by earlier philosophers such as Descartes

2 Methods

2.1 Experimental Set-up

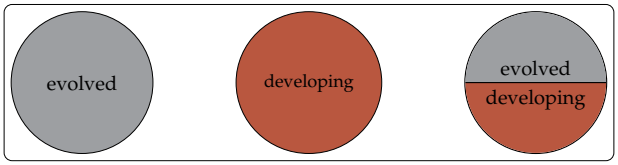
To test this hypothesis, the performance of three agents in a simulated environment will be compared to each other in terms of prediction error minimisation. We chose to investigate the hypotheses utilising a computer simulation rather than an empirical study, as a computer simulation has the advantage that evolution and development can be taken as independent variables. Both the interplay of evolution and development can be studied as well as their individual effects, which would not be possible in an empirical study. We also chose against pure conceptual research, as a computer simulation enforces higher levels of preciseness and clarity when formalising and thereby improves the theory. Furthermore, a computer simulation reveals any incoherencies and gaps in the theory and provides a way to test one's hypotheses before conducting an empirical study [3].
The three agents differ in the initial state of the hyperparameters and in the ability to develop hyperparameters through experience and observations in the environment:

- $Agent_E$: The initial state of the hyperparameters is determined purely by evolution, and the agent is not able to learn hyperparameters a posteriori.
- $Agent_D$: The initial state of the hyperparameters is uninformed, and the hyperparameters evolve throughout development.
- $Agent_{DE}$: The initial state of hyperparameters is set through evolution and hyperparameters can develop a posteriori through experience and observations in the environment.

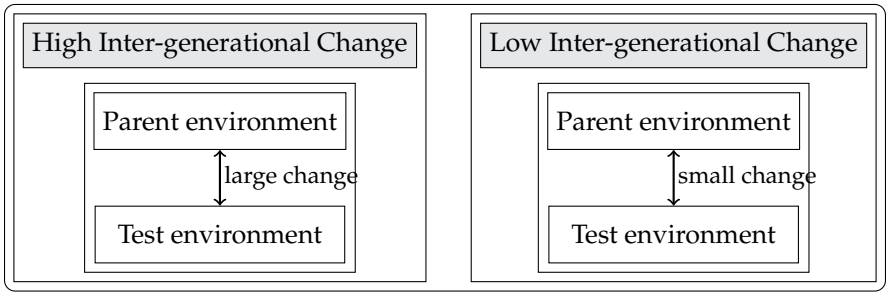
A visualisation of the three different agents can be found in Figure 4a.

The performance of the agents might not only depend on their ability to develop and the initial state of the hyperparameters but also on the amount of *inter-generational change* in the environmental contingencies. That is the amount of change between the environment the agents are tested on (test environment) and the environment the hyperparameters of the evolved agents are based on (parent environment). The decision to also investigate the amount of inter-generational change is based on the hypothesis that in environments with low inter-generational change the evolved agents might have a substantial advantage and that in environments with high inter-generational change the ability to develop might offer the largest advantage. By varying the levels of inter-generational change, we can research whether the characteristics of the interplay between evolution and development differ depending on the amount of change. Additionally, constant levels of inter-generational change do not seem realistic on a long-time scale.
Therefore, we will also investigate whether the performance of the agents is dependent on the amount of inter-generational change. The performance of each agent will be tested in two environments with different levels of inter-generational change -a *low* and a *high* level of change between generations (see Figure 4b). Note that low and high levels of change are measurements of the change between the stochastic nature of the environment in the parent and current generation and not within one generation.

[3] For further explanations regarding the relevance of computer simulation see *The Robo-havioral Methodology* by Maria Otworowska.



(a) Visualisation of the three agents



(b) Visualisation of the two environments the agents are tested in.

Figure 4: The three agents shown in Figure 4a are tested in the two environments (high and low inter-generational change) illustrated in Figure 4b. The initial state of the evolved agents is dependent on the parent environment which differs between the two conditions. The test environment is the same in the two conditions. The only difference between the conditions is the parent environment.

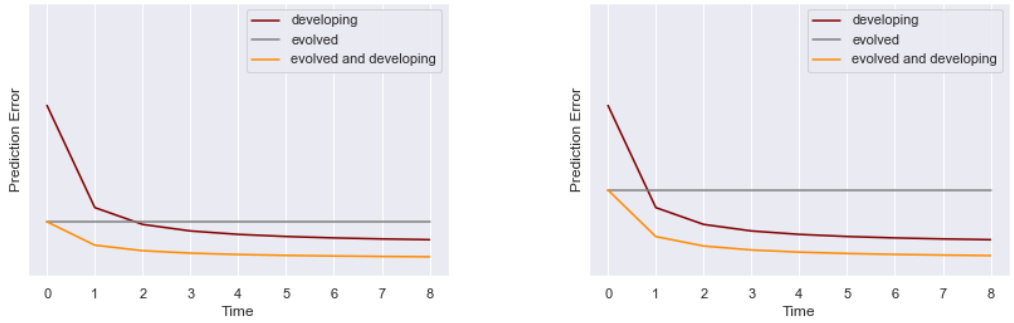
2.1.1 Performance Measurement

The size of the prediction error is calculated using forward Kullback-Leibler divergence. The prediction error was chosen as a performance measure as it is state-of-the-art within the predictive processing community see, for example, (Friston, 2010).
The prediction error is both analysed over time and averaged to evaluate the performance of the agents. The performance of an agent is measured by the decrease in prediction error over time. The trend in prediction error over time is considered as it supplies a good illustration of the differences in performance between the agents, for example, in their ability to learn or their initial prediction error. By looking at the decrease in prediction error over time, one is also able to see whether the information learned from evolution offers some initial advantage and whether the performance ranking of the agents changes over time.
Furthermore, the average prediction errors of the agents are compared as they provide information about an agent's overall performance at all time points. The average prediction error was chosen over the prediction error at the last time point as a measure of performance, as solely looking at the performance of an agent at the last time point would not offer a good representation of an agent's *overall* performance. An agent only observes a few objects for each feature-location combination. One wrong observation at the end of the trial, due to, for example, noise in the agent's perception, could increase the prediction error notably and thus not provide a truthful representation of its general performance. Likewise, the relative change in prediction error would not provide a useful baseline for comparing performance. The relative change in prediction error only provides insights into an agent's ability to learn. However, the extent of learning is not representative of an agent's overall performance. For instance, the purely evolved agent is unable to learn, implying that its relative change in prediction error is close to zero. Consequently, the performance of the purely evolved agent would be assessed as insufficient when looking at the relative change in prediction error. However, even though the purely evolved agent is incapable of learning, it is still possible for the agent to have the lowest prediction error throughout the trial.
The average prediction error is therefore the preferred measurement of performance. It cancels out most of the noise caused by the inaccuracy of an agent's perception and provides an excellent performance measurement.

2.1.2 Hypotheses

Given only a small number of iterations, the following patterns are expected to be observed in the results. $Agent_{DE}$ is expected to outperform the other agents in the low change condition, due to its additional knowledge through evolution, which in this case offers an advantage, and its ability to learn and adapt to changes. In the same condition, $agent_E$ is expected to outperform $agent_D$ as the test environment only changed slightly from the environment its hyperparameters are learned on, and $agent_D$ would not be able to reach the same level of knowledge after only a few iterations (see Figure 5a).
However, in the high change condition, we expect $agent_D$ to outperform $agent_E$ as here the advantage from evolution is a lot smaller than in the low change condition and the developing agent is expected to produce more accurate

predictions than the evolved agent after only a few iterations (see Figure 5b). The evolved and developing agent is expected to outperform the developing agent as long as the knowledge from evolution offers an advantage, and the number of iterations is comparably small. $Agent_D$ is expected to outperform $agent_{DE}$ if the knowledge learned through evolution offers a disadvantage. Moreover, $agent_{DE}$ is expected to outperform $agent_E$ in all cases.



(a) Low inter-generational change

(b) High inter-generational change

Figure 5: Visualisation of the expected performance of the three agents in the two conditions. The performance is measured in terms of prediction error minimisation over time.

2.2 Formalisation of the Agents

The initial states and learning rules of the agents will be illustrated on a simple multi-valued prediction task. The task the agents need to perform in the simulation is slightly more complex. However, this example is complex enough to demonstrate the differences between the three agents. The three agents only vary in the initial states of the hyperparameters and their ability to learn hyperparameters over time. The generative model is the same for all three agents.

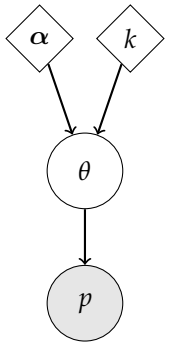


Figure 6: Graphical model of all agents

Figure 6 shows the simplified graphical model of the agents in a multi-valued / non-binary prediction task. Here the vector $\boldsymbol{\alpha}$ represent the concentration parameters and k the number of categories. The vector $\boldsymbol{\theta}$ represents the probability distribution over the categories and is drawn from a Dirichlet distribution using $\boldsymbol{\alpha}$ and k. The prediction p is then drawn from a categorical distribution with the parameter $\boldsymbol{\theta}$.

Generative model:

$$p \mid \boldsymbol{\theta} \sim Categorical(\boldsymbol{\theta}) \qquad \text{with } \boldsymbol{\theta} = \{\theta_1, ..., \theta_k\} \text{ category probabilities} \tag{2}$$

$$\boldsymbol{\theta} \mid \boldsymbol{\alpha}, k \sim Dirichlet(\boldsymbol{\alpha}, k) \qquad \text{with } \boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_k\} \text{ concentration parameters} \tag{3}$$

2.2.1 Purely Evolved Agent

The hyperparameter $\boldsymbol{\alpha}$ is initially set to some concentration vector $\boldsymbol{a}$ through evolution with $\boldsymbol{a} = \{a_1, ..., a_k \mid a_i \in \mathbb{R}^+, i = 1...k\}$. Observations throughout the agent's life do not change the value of $\boldsymbol{a}$. Thus, $\boldsymbol{\alpha}$ has the same value at every time point t.

$$\boldsymbol{\alpha_t} = \boldsymbol{a} \tag{4}$$

2.2.2 Purely Developing Agent

The hyperparameter $\boldsymbol{\alpha}$ is set initially to a uniform and uninformed concentration vector expressed as $\boldsymbol{u} = \{u_1, ..., u_k \mid u_i = 1, i = 1..k\}$. At every time point (here equivalent to observation) the corresponding concentration parameter increases given a sequence of previous observations O_{t-1} at some time point $t \geq 1$:

$$\boldsymbol{\alpha_t} = \{x_1, ..., x_k \mid x_i = u_i + \sum_{o \in O_{t-1}} f(o, i), i = 1...k\} \tag{5}$$

$$f(o, i) = \begin{cases} 1 & \text{if } o = i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

2.2.3 Evolved and Developing Agent

The hyperparameter $\boldsymbol{\alpha}$ is initially set through evolution to a concentration vector $\boldsymbol{a}$ with $\boldsymbol{a} = \{a_1, ..., a_k \mid a_i \in \mathbb{R}^+, i = 1...k\}$ and at every time step (here equivalent to observation) the corresponding concentration parameter increases given a sequence of previous observations O_{t-1} at some time point $t \geq 1$:

$$\boldsymbol{\alpha_t} = \{x_1, ..., x_k \mid x_i = a_i + \sum_{o \in O_{t-1}} f(o, i), i = 1...k\} \tag{7}$$

$$f(o, i) = \begin{cases} 1 & \text{if } o = i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

2.2.4 Modelling an Infant Agent

The goal of this research is to investigate the state of hyperparameters within *infant agents*. However, before focusing on hyperparameters, the question of how infant-like behaviour can be simulated needs to be answered. In the simulation task, the agents do not actively interfere in the world to minimise prediction error but rather reduce prediction error by updating their hyperparameters based on passive observations. This implies that in the context of this experiment only the perception of the agent needs to resemble infancy.

During the first few months of infancy, an infant's perception is generally poor (Granrud, 1993). To imitate the inaccuracy of infant perception, the accuracy parameter ϵ is introduced to the model of the agent. The parameter ϵ indicates how accurate the perception of the agent is, that is how probable it is that the agent's observation is equivalent to the actual event that occurred.
Furthermore, the certainty parameter γ is added to the updating rules to model any uncertainty an agent has in the accuracy of its own perception. Instead of increasing the pseudo-count of the observed element by one after each observation, γ is added to the observed element, and $1 - \gamma$ is distributed over the alternatives.
The accuracy and certainty parameters are added to the model of the agents as follows:

2.2.4.1 Purely Evolved Agent

The formalisation of the evolved agent does not change since it does not take observations into account.

2.2.4.2 Purely Developing Agent

The formalisation of the developing agent changes both in terms of noise and in terms of uncertainty. The hyperparameter $\boldsymbol{\alpha}$ is initially set to a uniform and uninformed concentration vector expressed as $\boldsymbol{u} = \{u_1, ..., u_k \mid u_i = 1, i = 1..k\}$ and at every time point the concentration parameter corresponding to the object observed by the agent increases: Given the sequence of events E_{t-1} and the sequence of observations made by the agent O_{t-1} at some time point t, for which it holds that $O_{t-1} \equiv E_{t-1}$ with accuracy $\epsilon \in [0, 1]$, and given the uncertainty parameter $\gamma \in [0, 1]$, then $\boldsymbol{\alpha}$ is updated as follows:

$$\boldsymbol{\alpha_t} = \left\{ x_1, ..., x_k \mid x_{i=1,...,k} = u_i + \sum_{o \in O_{t-1}} f(o,i) \times \gamma + (1 - f(o,i)) \times \frac{(1-\gamma)}{(k-1)} \right\} \tag{9}$$

$$f(o,i) = \begin{cases} 1 & \text{if } o = i \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

2.2.4.3 Evolved and Developing Agent

The formalisation of the evolved and developing agent changes both in terms of noise and in terms of uncertainty. The hyperparameter $\boldsymbol{\alpha}$ is initially set to some concentration vector $\boldsymbol{a}$ through evolution with $\boldsymbol{a} = \{a_1, ..., a_k \mid a_i \in \mathbb{R}^+, i = 1...k\}$ and at every time step (here equivalent to observation) the corresponding concentration parameter increases: Given the sequence of events E_{t-1} and the sequence of observations made by the agent O_{t-1} at some time point t, for which it holds that $O_{t-1} \equiv E_{t-1}$ with accuracy $\epsilon \in [0,1]$, and given the uncertainty parameter $\gamma \in [0,1]$, then $\boldsymbol{\alpha}$ is updated as follows:

$$\boldsymbol{\alpha_t} = \left\{ x_1, ..., x_k \mid x_{i=1,...,k} = a_i + \sum_{o \in O_{t-1}} f(o,i) \times \gamma + (1 - (f(o,i)) \times \frac{(1-\gamma)}{(k-1)} \right\} \tag{11}$$

$$f(o,i) = \begin{cases} 1 & \text{if } o = i \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

2.2.5 The Role of Evolution

Currently, the concentration vector a of the evolved agents is taken as-is from the parent, giving pseudo-observations in the parent environment as much weight as observations in the test environment. This does not seem realistic, as pseudo-observations inherited through evolution have as much impact as pseudo-observations gathered during the agent's lifetime. Assume that the stochastic nature of the environment changes between generations. The evolved and developing agent would need more than a lifetime to outweigh the pseudo-observations gathered throughout evolution. This becomes even more evident when evolving hyperparameters over multiple generations. The weight of the pseudo-observations learned through evolution would grow with each generation making it impossible for the agent to adapt to changes in the environment. To account for infinitely increasing hyperparameters, the normalisation parameter η is added to the model of the evolved agents.

2.2.5.1 Purely Evolved Agent

The hyperparameter $\boldsymbol{\alpha}$ is initially set to some concentration vector $\boldsymbol{a}$ through evolution with $\boldsymbol{a} = \{a_1, ..., a_k \mid a_i \in \mathbb{R}^+, i = 1...k\}$. Observations throughout the agent's life do not change the value of $\boldsymbol{a}$. Thus, given the normalization parameter η, $\boldsymbol{\alpha}$ has the same value at every time point t.

$$\boldsymbol{\alpha_t} = \eta \boldsymbol{a}$$

2.2.5.2 Purely Developing Agent

The formalisation of the purely developing agent does not change as the initial state of its hyperparameter is not dependent on evolution.

2.2.5.3 Evolved and Developing Agent

The hyperparameter $\boldsymbol{\alpha}$ is initially set to some concentration vector $\boldsymbol{a}$ through evolution with $\boldsymbol{a} = \{a_1, ..., a_k \mid a_i \in \mathbb{R}^+, i = 1...k\}$ and at every time step (here equivalent to observation) the corresponding concentration parameter increases: Given the sequence of events E_{t-1} and the sequence of observations made by the agent O_{t-1} at some time point t, for which it holds that $O_{t-1} \equiv E_{t-1}$ with accuracy $\epsilon \in [0,1]$, and given the uncertainty parameter

$\gamma \in [0, 1]$ and normalization parameter η, then $\boldsymbol{\alpha}$ is updated as follows:

$$\boldsymbol{\alpha_t} = \left\{ x_1, ..., x_k \mid x_{i=1,...,k} = \eta a_i + \sum_{o \in O_{t-1}} f(o,i) \times \gamma + (1 - (f(o,i)) \times \frac{(1-\gamma)}{(k-1)} \right\} \tag{13}$$

$$f(o,i) = \begin{cases} 1 & \text{if } o = i \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

2.3 Simulation

The performance of an agent is evaluated in terms of prediction error minimisation, that is an agent's ability to learn the environmental contingencies of the simulated environment. The simulated environment used to compare the performance of the agents is based on the simulation build in collaboration with Burgos, Drummer, Geertjes, and Rennspies. First, the fundamental simulation will be discussed, followed by a thorough explanation of the extensions specific to this research question.

2.3.1 General Simulation Environment

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

2.3.1.1 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.

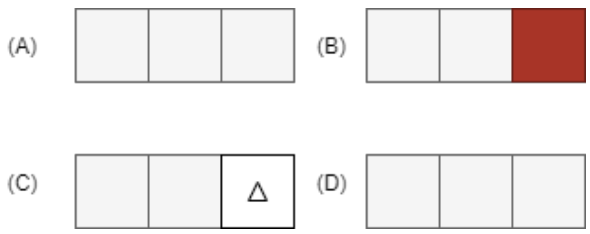


Figure 7: An example of a single iteration in the general simulation. First the agent sees 3 empty positions (A), then, at one of the positions, a tile appears (B). After perceiving the stimuli, the agent makes a prediction, then the tile is flipped revealing the object underneath (C). Lastly the object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

2.3.1.2 Environment

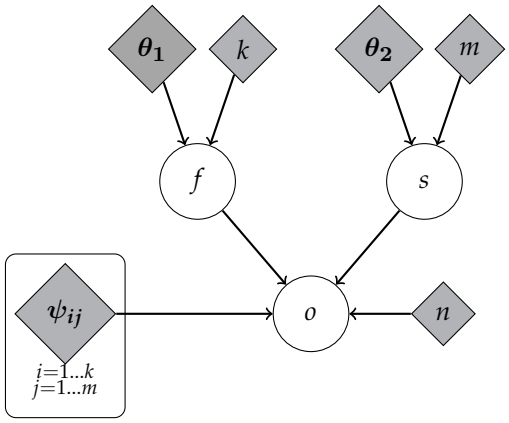


Figure 9: The graphical model of the environment: Here node o represents the object. The object is dependent on the probability distribution ψ_{ij} given the spatial position and feature of the tile, represented by nodes s and f respectively. Nodes k, m and n represent the amount of possible features, 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:

$$f \mid \boldsymbol{\theta_1}, k \sim Categorical(\boldsymbol{\theta_1}, k) \tag{15}$$

$$s \mid \boldsymbol{\theta_2}, m \sim Categorical(\boldsymbol{\theta_2}, m) \tag{16}$$

$$o \mid \psi_{ij}, f, s, n \sim Categorical(\psi_{fs}, n) \tag{17}$$

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 2.3.1. Depending on the environment and conditions needed, these numbers can be adjusted.

2.3.1.3 Agent

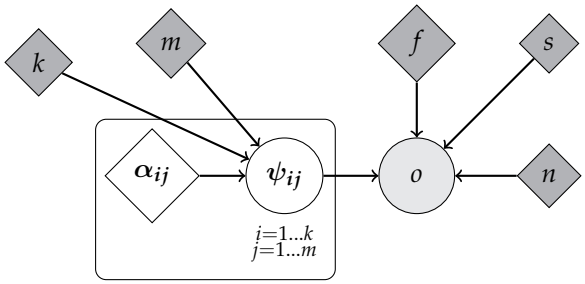


Figure 10: The graphical model of the agent: Here node o represents the object. The object is dependent on the probability distribution ψ_{ij} given the spatial position s and feature of the tile f. Nodes k, m and n represent the amount of possible features, spatial positions and objects respectively. The categorical probability distribution ψ_{ij} is dependent on α_{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 α_{ij} is learned throughout the trial.

Generative model:

$$\psi_{ij} \mid \alpha_{ij}, n \sim Dirichlet(\alpha_{ij}, n) \qquad \text{with } i = 1...k,\ j = 1...m \tag{18}$$

$$o \mid \psi_{ij}, f, s, n \sim Categorical(\psi_{fs}, n) \tag{19}$$

In Figure 10 the graphical model of the agent is shown. As described in Section 2.3.1.2 (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 have been selected by the environment. White parameter nodes such as α_{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 α_{ij} is initially set to a uniform vector with one pseudo-observation for each object. The probability distribution ψ_{ij} of the objects, given the features and spatial positions, is sampled from a Dirichlet distribution with the concentration vector α_{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 ψ_{fs} given the observed feature f and spatial position s.

2.3.1.4 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.

2.3.2 Modifications of the Simulation

The simulation introduced in the previous paragraph has been modified to fit this specific research. The model of the agent has been changed to account for evolution and development by adding the agent-specific updating rules. Furthermore, the accuracy, certainty and normalization parameters as described in section 2.2.4 and 2.2.5 have been added to the model. The value of the parameters, their definition and the reasoning behind the values are shown in table 1.

Parameter	Value	Definition	Justification
$n\_iter_{current}$	100	The number of observations the infant agent observes in the test environment	The chosen number of observations is low enough for an infant agent to observe realistically during its life span. On the other hand, the number of iterations is large enough for the developing agent to learn the stochastic nature of the environment, but small enough to still have some reducible uncertainty.
n_iter_{parent}	1000	The number of observations the agent observes in the parent environment which represent the pseudo-observations inherited through evolution	The number of iterations is large enough for the agent to learn the environmental contingencies and small enough to be realistic.
η	0.1	The normalisation of the pseudo-observations inherited through evolution	A normalisation value of 0.1 gives pseudo-observations gathered during development the same weight as those inherited from evolution.
γ	0.9	The certainty of the agent in its perception	The task the agent needs to perform is comparably simple. Thus, we decided to give the agent high certainty in its perception.
ϵ	0.9	The accuracy of the agent's perception	Perception during infancy is relatively poor compared to adulthood, however, the task the agent needs to perform is relatively simple. Therefore, the accuracy of the agent's perception is not perfect but also not close to chance.

Table 1: An overview of the parameters including their definition, value and justification.

Furthermore, the two environments - with low and high levels of inter-generational change - have been introduced to the simulation environment. The environmental contingencies of the test environment are identical for the two conditions. The amount of inter-generational change is modelled in the level of change between the stochastic nature of the parent environment and the test environment. Implying that for the high inter-generational change condition, the parent environment differs to a greater extent from the test environment than in the low inter-generational change condition.
For the low inter-generational change condition, the likelihood rankings of the test and the parent environment are equivalent. If a specific feature-location combination distributes most probability mass to a *triangle* in the parent condition, then the hypothesis *triangle* must also be most likely for the same feature-location combination in the test environment. The only difference between the test and parent environment in the low inter-generational change condition is the exact probability of sampling a specific object. For example, in the parent condition a *triangle* occurs with 80% probability in the *centre* position independent of the feature. However, in the test condition a *triangle* occurs with almost deterministic probability 99.998% in the *centre* position. There is no difference between parent and test environment regarding which option is most probable; however, there is a difference in the exact amount of probability mass distributed to that option.
The stochastic nature of the parent environment in the high inter-generational change condition is equivalent to the low inter-generational change condition for the spatial positions *left* and *right*. This implies that the only but crucial difference between the low and high inter-generational change condition is in the environmental contingencies of the *centre* position. Here, the likelihood ranking almost completely flipped between the parent and test environment. In the parent environment a *square* is the most probable object ($p = 80\%$) for position *centre* independent of the feature, whereas in the test environment almost all probability mass is distributed to the option *triangle* ($p = 99.998\%$). This characteristic of the parent environment will lead to a large prediction error when trying to predict the outcome for position *centre* in the test environment by utilising knowledge acquired from the parent environment. We chose only to introduce high change for the *centre* position to ensure that the knowledge gained from evolution still offers some advantage. Modelling the high inter-generational change condition in this way ensures that some form of adaptation to the change in environmental contingencies is needed to perform well[4].

[4]The quality of performance is measured in terms of prediction error minimisation.

2.3.3 Experimental Paradigm

The experiment consists of two simulations: one simulation in an environment with low inter-generational change and one in an environment with high inter-generational change. In each simulation, the performance of the three agents is compared in terms of prediction error minimisation. At the beginning of each simulation the three agents and the environment are initialised.

First, the environment corresponding to the experimental condition needs to be initialised. The test environment is independent to the amount of inter-generational change meaning that it stays the same in both conditions, whereas the stochastic nature of the parent environment depends on the amount of inter-generational change. In the low inter-generational change condition the parent environment does not differ from the test environment in terms of the likelihood ranking of the objects. The two environments only differ in how much more likely an object is compared to the others. In the high inter-generational change condition the environment changed in terms of the likelihood ranking of the objects between generations.

To initialise the agents, the initial states of the hyperparameters need to be set. For the purely developing agent, the initial state of the hyperparameter $\boldsymbol{\alpha}$ is an uninformed and symmetric concentration vector. Setting the initial state for the evolved agents is not as easy as for the purely developing agent, since the concentration vector $\boldsymbol{a}$ inherited from evolution needs to be generated first. In this simulation the evolved concentration vector $\boldsymbol{a}$ is generated by letting a purely developing agent run for $\text{n\_iter}_{\text{parent}}$ iterations in the parent environment corresponding to the experimental condition. The learned concentration vector is then taken as the initial state for the inherited concentration vector $\boldsymbol{a}$ of the purely evolved agent, as well as the developing and evolved agent.

After initialising the environments and agents, the main part of the simulation starts. During the span of the simulation, the agents gather $\text{n\_iter}_{\text{current}}$ observations. For each observation, every agent makes a prediction given the feature and the location. Based on that prediction, the prediction error for the specific feature and location is calculated. After the prediction error is calculated, the agents update their hyperparameters following their updating rules, as explained in section 2.2. This procedure is illustrated in the algorithm 1.

Algorithm 1: Simulation

```
env ← environment the agents are tested on;
agents ← developing, evolved, and developing and evolved agent;
n_iter ← number of iterations;
for i ∈ [0, n_iter] do
    stimuli ←environment get stimuli;
    for agent ∈ agents do
        prediction ← agent predicts stimuli ;
        calculate prediction error;
        agent update hyperparameters;
    end
end
```

3 Results

In the following two sections, the performance of the agents both in the low inter-generational change condition and in the high inter-generational change condition is analysed using the two measurement techniques - average prediction error and prediction error minimisation over time - explained in section 2.1.1 [5].

3.1 Low Inter-generational Change

The performance of the agents in the low inter-generational change condition is illustrated in Figure 11. The prediction error for the purely developing agent (red line) starts relatively high and decreases over time. However, it is not able to reach the same performance as the evolved and developing agent at the end of the trial.
The purely evolved agent (grey line) has a lower initial prediction error than the developing agent. In contrast to the developing agent, the prediction error of the evolved agent remains more or less stable over time, except for a few spikes and drops in prediction error. The drop in prediction error at the last time point does not represent learning but merely is due to the evolved agent randomly sampling a distribution which is close to the true distribution.
The evolved and developing agent (orange line) starts similar to the evolved agent with a relatively low prediction

[5] The code used to generate the results can be found in the appendix.

error. In contrast to the evolved agent, the agent is able to learn from observations and decreases its prediction error over time. The decrease in prediction error is only minor since the agent already started with a small prediction error and slowly approximates the amount of irreducible uncertainty.

In general, the evolved agent finishes the trial with the smallest final prediction error and the developing agent with the largest final prediction error (PE). When looking at the final prediction error, the performance ranking is 1. evolved agent, 2. evolved and developing agent, and 3. developing agent sorted decreasingly by performance. However, the final prediction error is not a good measure of an agent's overall performance partially due to the variability in sampling from a distribution. A better way to analyse an agent's performance is by using the average prediction error. The developing agent still performs worst with an average prediction error of ≈ 0.720. However, the evolved and developing agent (PE$\approx$ 0.333) has a smaller average prediction error than the evolved agent (PE≈ 0.412), implying that on average the evolved and developing agent outperforms the evolved agent [6].

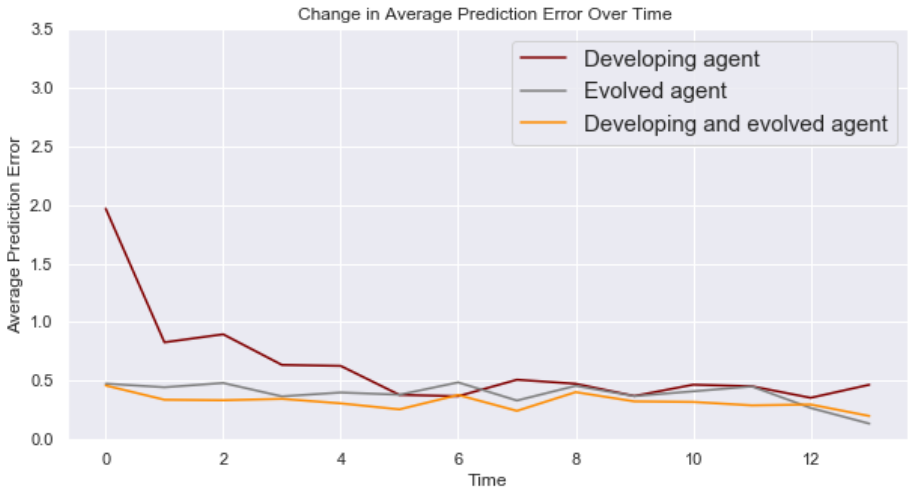


Figure 11: The change in prediction error over time in the low inter-generational change condition averaged over feature-location combinations: The evolved agent (grey line) as well as the evolved and developing agent (orange line) start with a low initial prediction error compared to the developing agent (red line). The prediction error of the evolved agent is more or less stable over time. The prediction error of the evolved and developing agent decreases over time, and the prediction error of the developing agent decreases over time but is not able to reach the same performance as the evolved and developing agent.

3.2 High Inter-generational Change

The only but crucial difference between the high and low inter-generational change condition is the complete change in likelihood ranking in the spatial position *centre* as explained in section 2.3.2. Therefore, additionally to the average performance of the agents across all spatial positions, the performance of the agents in the spatial position *centre* will be analysed.

Figure 12 shows the change in prediction error for the spatial position *centre*. The evolved agent and the evolved and developing agent both have a higher initial prediction error than the developing agent. The prediction error for the evolved agent (grey line) stays high over time. The evolved and developing agent (orange line) decreases its prediction error over time but is not able to reach the same level of prediction error minimisation as the developing agent. The developing agent (red line) starts with a smaller initial error than both evolved agents and decreases the prediction error over time. Important to note is that the change in initial prediction error is not caused by the developing agent performing better, but by the evolved agent performing worse. When comparing the low and high inter-generational change conditions, one can observe that the initial prediction error for the developing agent is roughly equal, but the prediction errors for the evolved agent as well as the evolved and developing agent increased.

This trend can also be observed when comparing the prediction error averaged over time. The developing agent performs best with an average prediction error of ≈ 0.612 [7]. The purely evolved agent is ranked last in the performance ranking with a mean prediction error of ≈ 2.144. The evolved and developing agent outperforms the

[6] The average prediction error of each agent for each feature-location combination can also be found in the appendix.

[7] The value of the prediction error for the developing agent in the high inter-generational change condition is in the same range as the prediction error for the developing agent in the low change condition.

evolved agent greatly with an average prediction error of ≈ 1.147; however, it is not able to reach a level of performance close to the developing agent.

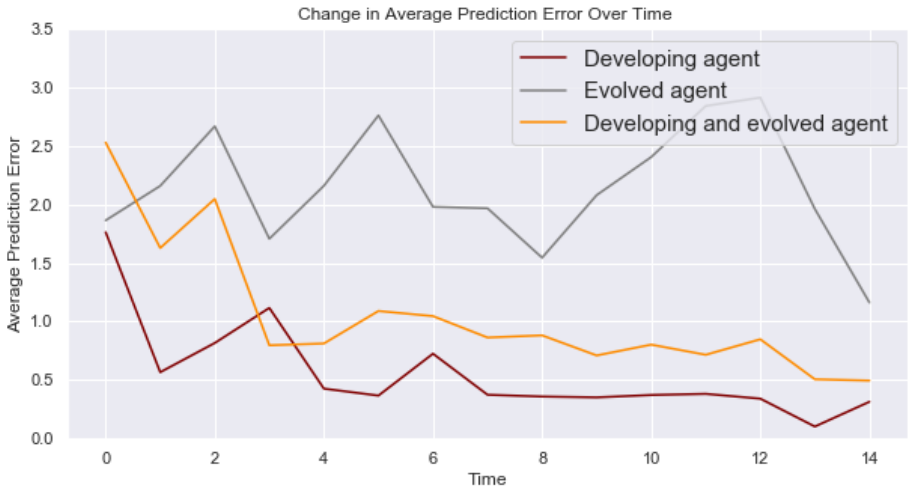


Figure 12: The change in average prediction error over time for position centre: Both evolved agents (grey and orange lines) start with a higher initial prediction error than the developing agent (red line). The prediction error of the evolved agent (grey line) oscillates more or less around a constant average. The evolved and developing agent (orange line) is able to decrease its prediction error over time but does not reach the level of the developing agent. The developing agent (red line) starts with a comparatively smaller prediction error and minimises its prediction error rapidly.

The change in prediction error for the high inter-generational change condition averaged over all feature-location combinations is shown in Figure 13. Similar patterns to the low inter-generational change condition can be observed. The purely evolved agent and the evolved and developing agent have a lower initial prediction error than the purely developing agent. Nonetheless, the initial prediction error of the evolved agents in the high change condition is higher than in the low change condition. The prediction error for the purely evolved agent (grey line) remains relatively stable over time. The fluctuations in the prediction error are not an effect of learning but of the agent making a prediction by randomly sampling from its generative model. In contrast to the evolved agent, the evolved and developing agent reduces its high initial prediction error during the duration of the trial. Similar to the low inter-generational change condition, the developing agent starts with a relatively high prediction error but minimises its prediction error rapidly, causing the developing agent to finish the trial with the lowest prediction error. When looking at the performance ranking, the evolved and developing agent performs best with an average prediction error of ≈ 0.609, closely followed by the developing agent (PE=≈ 0.729). The evolved agent performs worst with an average prediction error of ≈ 1.025.

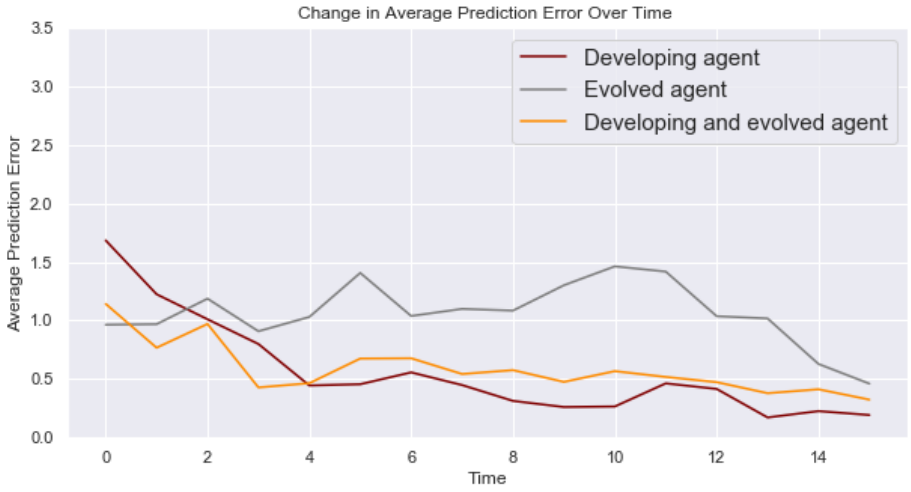


Figure 13: The change in prediction error over time in the high inter-generational change condition averaged over all possible combinations of feature and location: The evolved and developing agent (orange line) and the evolved agent (grey line) both have a smaller initial prediction error than the developing agent (red line). The prediction error of the evolved agent stays relatively high over time. The evolved and developing agent is able to decrease its prediction error over time. The developing agent (red line) starts with a higher initial prediction error but decreases the prediction error rapidly over time.

4 Discussion

4.1 Interpretation of the Results

In general, the results of the experiment are in line with the expected outcomes hypothesised in section 2.1. The dominance of the evolved agents [8] in the low inter-generational change condition suggests that innate knowledge offers an important advantage in environments with minor changes between generations. The difference in initial prediction error between the evolved agents and the developing agent represents the size of the advantage provided by evolution. The evolved and developing agent can outperform the evolved agent due to its ability to learn and further adapt to the environmental contingencies. The observed performance patterns infer that in environments with little changes, innate knowledge offers the most valuable advantage during the first critical moments of infancy.

In the high inter-generational change condition, the small initial prediction error of the evolved agents implies that the knowledge acquired from evolution still offers some advantage. Nevertheless, the comparatively bad performance of the purely evolved agent implies that innate knowledge alone is not enough to perform well. Some form of learning and adaptation to changes in environmental contingencies seems to be required for optimal performance. The dominance of the evolved and developing agent over the purely developing agent suggests that even in the high inter-generational change condition innate knowledge offers some advantage. The observed performance patters infer that in environments with high levels of change between generations, the ability to learn and to adapt is most valuable to decrease prediction error during the start of infancy. However, it is important to note than in some of the trials the developing agent outperforms the evolved and developing agent in the high change condition, which is not in line with our initial hypotheses. However, this observation can be explained by the simplicity of the task, namely that the task is simple enough to learn it in a few observations. For more complex tasks, the developing agent would most likely not be able to outperform the evolved and developing agent.

The dominance of the evolved and developing agent in both conditions suggest that optimal performance requires both the ability to learn and some form of innate knowledge. Opponents of innate knowledge might argue that the evolved and developing agent, of course, outperforms the purely developing agent as both conditions are advantageous to innate knowledge. In other words, in none of the test conditions inherited knowledge is a disadvantage. This is a reasonable concern; however, it misses one crucial consideration. For innate knowledge to be disadvantageous, there needs to be a major flip in environmental contingencies between two generations. Such a flip is extremely unlikely, as such substantial changes in environmental contingencies happen on the time scale of millennia or mega-anna and not between two generations. Changes between two generations are comparatively small such that knowledge learned during the previous generation grants valuable insights into the next generation.

[8] The term evolved agents refers to both the purely evolved agent and the evolved and developing agent.

Even though these changes do not occur often, some individuals might still evolve a maladaptive bias through evolution. In these cases, purely developing agents would outperform agents with innate knowledge. However, these maladaptations only occur rarely as maladaptations are usually caused by mutation or by high changes between generations. High changes in the environmental contingencies such as large changes in climate[9] do not happen rapidly between two generations but steadily over as many as thousand generations and mutations are generally the exception. Furthermore, individuals with maladaptations have a lower chance of survival and sexual reproduction and, thus, a lower chance to pass on disadvantageous knowledge to the next generation. Accordingly, on the scale of generations, the number of individuals flourishing from innate knowledge largely outweighs the number of individuals failing due to maladaptations.

4.2 The Need for Evolution and Development

As noted in the previous section, the modelled task lacks the complexity of real-life tasks such as learning a language. Due to their high complexity, real-life tasks are usually impossible to perform correctly after only a few observations. Thus, to learn a task properly an infant needs to make an abundant amount of observations, which is not realistic given time constraints or additional constraints imposed by the environment. Consequently, the agent needs to find a way to approximate optimal behaviour in its current environment as good as possible. Basing the initial predictions on the previous environment offers a good way for the agent to reduce its prediction error to the greatest extent possible and approximate optimal behaviour. Scientists supporting the autonomy of nurture might argue that even though the developing agent has an initially high prediction error, the agent reaches the same performance as the evolved and developing agent at the end of the trial, thus concluding that innate knowledge is not required. However, these first moments of infancy are crucial. Harmful behaviour during the beginning of an agent's life could have lifetime adverse effects and might even put the agent's survival at risk. For example, the ability to cry is an innate behaviour (Rottenberg & Vingerhoets, 2012), enabling the infant to communicate problems and needs to the outside world. If the agent would have to learn this form of early communication first, some life-threatening problems during the first steps of infancy might be missed due to the agent's inability to communicate its needs. This shows that decreasing the prediction error as much as possible during the beginning of the infant's life is crucial for survival. Innate knowledge offers an excellent and biologically plausible way to minimise the initial prediction error. Signifying the importance of considering innate knowledge when modelling cognition.

Nonetheless, the comparatively bad performance of the evolved agent in the high inter-generational change condition illustrates that innate knowledge is not sufficient for reaching optimal performance, especially in environments with high levels of change between generations. The unsatisfactory performance of the evolved agent suggests that, for optimal performance, an agent needs to be able to develop and adapt to changes in the stochastic nature of the environment.
There are multiple other reasons for the implausibility of a purely evolved agent to model human infant behaviour. Many competencies such as vision and the ability to walk are inadequate or non-existent at birth but improve during infancy implying some form of development[10]. To explain these patterns, a purely evolved agent would need to be born with the knowledge required to perform these behaviours correctly but not employ this knowledge during the first periods of infancy. However, if an agent is already born with the ability to walk correctly, why does the agent not employ this knowledge during the first year of its life to decrease prediction error? Such an agent would not conform to the main principle of predictive processing, i.e. continuous prediction error minimisation, as the agent possesses the knowledge needed to decrease prediction error but does not apply it. Besides a purely evolved agent could not explain the existence of the large interval between infancy and adulthood, if humans are already born with a wholly evolved generative model. Moreover, on a longer time scale, this would mean that an agent cannot improve during its life span, so changes in the generative models between generations could only occur through mutation or selection. However, it is highly unlikely that evolution can lead to rational behaviour if the only ways of improvement are random mutation and selection. Purely evolved agents could also not explain the existence of culture. In other words, how can agents pass on knowledge over multiple generations without a form of development? These incoherencies suggest that the ability to learn is a crucial part of human cognition.

To conclude, both innate knowledge and development are central to explaining how cognitive phenomenons in adult agents arise. Therefore, when modelling cognition, both innate knowledge and learning, and consequently, their representations need to be considered.

[9] Referring to natural changes in climate not caused by human behaviour.

[10] It is interesting to note that even though human infants are not able to walk perfectly only a few seconds after they are born, many animals such as deer and antelopes are able to do so. Implying that the extent of innate knowledge and thus the importance of development and evolution might differ between animals and humans and also across animals.

4.3 Origins of Hyperparameters

As concluded in the earlier paragraphs, the generative model needs to capture the interplay between knowledge acquired from evolution and knowledge acquired from development. The results of the experiment show that hyperparameters offer a possible way to represent said interplay. However, it does not seem likely that hyperparameters are the only way how the generative model represents innate knowledge, as there are some forms of knowledge hyperparameters might not be able to capture.
For instance, the genome does not have sufficient capacity to capture all possible connections between neurons. Consequently, there needs to be another way the genome encodes information on how to wire-up the brain next to specifying connections between neurons. Zador et al. suggest that the genome "specif[ies] a set of rules for wiring up the brain during development" (Zador, 2019, p.5). The incapacity of the genome to capture all possible connections between neurons implies that it would also be implausible for the generative model to capture innate knowledge only in the form of single connections or relations between neurons, i.e. variables. Hyperparameters can capture the strength of connections between neurons. However, rules defining how the generative model constructs or reduces itself during development might be extremely difficult, if not impossible, to capture using hyperparameters. Such wiring rules are more likely to be captured by the causal relations or complexity of the generative model or some other way this research misses to consider. Consequently, it is highly unlikely that innate knowledge in generative models is solely represented through hyperparameters.

To conclude, it seems plausible that the interplay of evolution and development are partially encoded using hyperparameters. However, there is still more research needed investigating different forms of knowledge representation to draw further conclusions.

4.4 Limitations and Future Research

4.4.1 Knowledge Representation

The focus of this research lies on representing evolutionary and developmental knowledge acquisition using hyperparameters. However, as discussed earlier, there are several different ways to represent innate knowledge and learning within the generative model. Two potential ways of representing knowledge would be through hyperpriors or the complexity of the initial generative model.
The initial distribution of a hyperprior could represent innate knowledge, and changes in the type of distribution could be representative for learning. For example, originally the distribution of the hyperprior could be a Gaussian distribution, but throughout the trial the agent learns that a binomial distribution fits the environmental contingencies better. Even though hyperpriors offer a possible way of representing the interplay of nature and nurture, frequent shifts in distribution type do not appear very likely due to their large effect on the outcome of predictions. Likewise, these changes are also quite limited due to the restricted number of probability distributions available to use as hyperpriors. Thus, they are unlikely to be the only way the generative model encodes the interplay of evolution and development.
A second way of representing said interplay could be through the complexity of the generative model. The quantity and character of the initial causal relations could represent knowledge inherited from evolution. This knowledge can then be improved and further adapted to the environmental contingencies by revising the generative model, be it through model construction or model reduction. One interesting point to investigate in this context would be the relationship between the initial complexity of the model and the choice of the agent to apply model construction or model reduction to minimise its prediction error.
The aforementioned methods of representing knowledge are two potential ways of knowledge representation worth investigating; however, there might exist other interesting ways this paper misses to discuss.

4.4.2 Evolutionary Computations and the Role of Evolution

This research chose to simplify the evolution process by modelling the minimum amount of generations needed. Extending the evolutionary computations by, for example, adding selection and random mutation or increasing the number of generations might give additional insights into the role of innate knowledge and how innate knowledge develops over generations.
Furthermore, the parameter η was introduced to model the role of evolution. How much weight the pseudo-observations inherited from evolution have on an agent's predictions is dependent on η. Currently, pseudo-observations inherited from evolution have the same influence on an agent's prediction as pseudo-observations gathered throughout development. However, evolution and development might not have equal weights, which of the two is weighted more might be task-dependent or dependent on the initial prediction error. A high initial prediction error due to high levels of inter-generational change might result in innate knowledge having less influence

on an agent's prediction than knowledge acquired through learning, whereas a small initial prediction error in low change conditions might have the reversed effect.

4.4.3 Volatility

Currently, environmental contingencies only change between generations. However, it might be interesting to investigate the effects of adding intra-generational change based on the suggestion made by Zador et al. that "if an environment is changing rapidly—e.g., on the timescale of a single individual— innate behavioral strategies might not provide a path to as high a level of mature performance as a mixed strategy that relies in part on learning" (Zador, 2019). Adding this form of volatility might give further insights into the effects of evolution and development on the behaviour of the agent.

4.4.4 Quality of Information

It might also be interesting to extend upon development as it is currently defined to make a distinction between different forms of learning, such as supervised and unsupervised learning. Currently, the developing agents are unable to make a qualitative distinction between different forms of learning or information sources.
The form of learning the developing agents currently employ is unsupervised learning. However, in real-life scenarios, there are other forms of learning agents use to acquire knowledge next to unsupervised learning. For example, humans employ supervised learning when passing knowledge on to another human, for instance, through teaching. In this way, humans can learn many complex behaviours relatively fast. To model a form of development closer to reality, one should consider the different forms of learning.
Furthermore, accounting for different levels of quality in information sources might be crucial for minimising prediction error. An agent might trust a specific information source more than another and therefore weighs the knowledge gained from a more trusted information source more. For example, knowledge acquired from an expert in the field might be weighted more than knowledge acquired from an information source known to be often unreliable.

5 Conclusion

In a first attempt to shift the focus of predictive processing towards development, this paper chose to examine the importance of innate knowledge and learning by focusing on the initial state of generative models, particularly the initial state of hyperparameters. We concluded that both innate knowledge and development must be considered when attempting to explain the complexity and emergence of human cognition. Further we proposed a plausible way the generative model could represent evolution and development by utilizing hyperparameters.
However, there are several other unexplored ways the interplay of evolution and development might be represented in the generative model. Consequently, no concrete conclusions can be drawn regarding the origins of hyperparameters before investigating those additional options of knowledge representation. Therefore, we highly suggest to further investigate the interplay of evolution and development in generative models.

References

Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, *36*(3), 181–204. doi: 10.1017/S0140525X12000477

Friston, K. (2010, 02). The free-energy principle: A unified brain theory? *Nature reviews. Neuroscience*, *11*, 127-38. doi: 10.1038/nrn2787

Granrud, C. (1993). *Visual perception and cognition in infancy*. Lawrence Erlbaum Associates.

Gruber, T. R. (2013, February). Nature, nurture, and knowledge acquisition. *Int. J. Hum.-Comput. Stud.*, *71*(2), 191–194. Retrieved from http://dx.doi.org/10.1016/j.ijhcs.2012.10.004 doi: 10.1016/j.ijhcs.2012.10.004

Kwisthout, J., Bekkering, H., & van Rooij, I. (2017). To be precise, the details don't matter: On predictive processing, precision, and level of detail of predictions. *Brain and Cognition*, *112*, 84 - 91. Retrieved from http://www.sciencedirect.com/science/article/pii/S0278262616300161 (Perspectives on Human Probabilistic Inferences and the 'Bayesian Brain') doi: https://doi.org/10.1016/j.bandc.2016.02.008

Kwisthout, J., & van Rooij, I. (2019, Jun 19). Computational resource demands of a predictive bayesian brain. *Computational Brain & Behavior*. Retrieved from https://doi.org/10.1007/s42113-019-00032-3 doi: 10.1007/s42113-019-00032-3

Lewkowicz, D. (2011, 07). The biological implausibility of the nature-nurture dichotomy & what it means for the study of infancy. *Infancy : the official journal of the International Society on Infant Studies*, *16*, 331-367. doi: 10.1111/j.1532-7078.2011.00079.x

McLeod, S. (2018). *Nature vs. nurture in psychology.* https://www.simplypsychology.org/naturevsnurture.html. (Accessed: 2019-11-23)

Meaney, M. (2001, 05). Nature, nurture, and the disunity of knowledge. *Annals of the New York Academy of Sciences*, *935*, 50 - 61. doi: 10.1111/j.1749-6632.2001.tb03470.x

Rottenberg, J., & Vingerhoets, A. J. J. M. (2012). Crying: Call for a lifespan approach. *Social and Personality Psychology Compass*, *6*(3), 217-227. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-9004.2012.00426.x doi: 10.1111/j.1751-9004.2012.00426.x

Zador, A. M. (2019). A critique of pure learning: What artificial neural networks can learn from animal brains. *bioRxiv*. Retrieved from https://www.biorxiv.org/content/early/2019/03/20/582643 doi: 10.1101/582643

Appendix

Source Code of the Simulation

[ ]:
```
#@authors:Ellen Schrader (based on the simulation build with Burgos, Drummer,
Geertjes, and Rennspies)
#for inline plots in jupyter
%matplotlib inline
# import matplotlib
import matplotlib.pyplot as plt
import matplotlib.tri as tri
# import seaborn
import seaborn as sns
# settings for seaborn plot sizes
sns.set(rc={'figure.figsize':(4.5,3)})
# import distributions
from scipy.stats import *
import numpy as np
import functools
from enum import Enum
import math
```

[ ]:
```
#This function returns the feature of the tile flipped and the object below it.
#Theta = vector of category probabilities of the features
#Psi = vector of category probabilities of the objects given the features
(deterministic)
#The number of objects n and features k is included implicitly in the size of the
corresponding arrays
class Environment:
    def __init__(self, theta_f, theta_l, psi):
        self.theta_f = theta_f
        self.theta_l = theta_l
        self.psi = psi

    def create_stimuli(self):
        #Draw feature from a categorical distribution
        feature = np.array(multinomial.rvs(p = self.theta_f, n = 1))
        location = np.array(multinomial.rvs(p = self.theta_l, 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[idx_feature, idx_location].flatten(), n
= 1)

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

        return idx_feature, idx_location, idx_object
```

[ ]:
```
#Model of the agent. The function returns the predicted probability distribution
over the objects given the features
#and the index of the predicted object
#Note: we might not need the predicted object and the index of the object itself
but it is still included as it might
#be useful for some of us
class Agent:
```

```
    def __init__(self, alpha, certainty, acc_perception, developing = False,
evolved = False):
        self.developing = developing
        self.evolved = evolved
        self.psi = None
        self.certainty = certainty
        self.acc_perception = acc_perception
        self.alpha = alpha
        n_locations = len(alpha[0])
        n_features = len(alpha)
        self.pre_errors = [[[] for _ in range(n_locations)] for _ in
range(n_features)]

    def __str__(self):
        if self.developing & self.evolved:
            return "Developing and evolved agent"
        elif self.developing:
            return "Developing agent"
        else:
            return "Evolved agent"

    def predict(self,idx_feature, idx_location):
        self.psi = dirichlet.rvs(self.alpha[idx_feature, idx_location,:])[0]

    def is_developing(self):
        return self.developing

    def set_developing(self,bool):
        self.developing = bool

    #updating of the hyperparameters uncertainty needs to be implemented later on

    def update(self,idx_feature, idx_location, true_object):
        if self.developing:
            n_objects = len(self.alpha[idx_feature, idx_location])

            #The agent does not always perceive the correct object
            if bernoulli.rvs(self.acc_perception)==1:
                perceived_object = true_object
            else:
                perceived_object = np.random.randint(n_objects)
                while perceived_object == true_object:
                    perceived_object = np.random.randint(n_objects)

            #Updating the perceived object
            self.alpha[idx_feature, idx_location, perceived_object]+= self.
 ↪certainty

            #Updating the other hypotheses
            idx = [i for i,_ in enumerate(self.alpha[idx_feature, idx_location])
if i!=perceived_object]
            self.alpha[idx_feature, idx_location][idx] = np.add( self.
 ↪alpha[idx_feature, idx_location][idx]
                                                                ,(1-self.
 ↪certainty)/(n_objects-1), casting="unsafe")

    def calc_pred_error(self, psi_environment,idx_feature, idx_location):
        pre_err=entropy(psi_environment[idx_feature, idx_location],self.psi,
base=None)
        self.pre_errors[idx_feature][idx_location].append(pre_err)
```

[ ]:
```
#The agent learns the probability distribution of the objects given a specific
feature
#n_iter = the number of iterations in this observations the agent sees
#n_objects = the number of objects
#n_features = the number of features and thus also of tiles
class Simulation:
    def __init__(self, environment, agents, n_iter= 1000):
        self.environment = environment
        self.agents = agents
        self.n_iter = n_iter

    def run(self):
        ###Updating the agent's generative model###
        for i in range(0,self.n_iter):

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

            for agent in self.agents:
                #Prediction of the agents
                agent.predict(idx_feature, idx_location)

                #Calculate prediction error using relative entropy
(Kullback-Leibler divergence)
                agent.calc_pred_error(self.environment.
 ↪psi,idx_feature,idx_location)

                #Updating hyperparameters (only for agents with the ability to
develop)
                #The agent does not always perceive the correct object (some noise
in perception infant agent)
                agent.update(idx_feature,idx_location, idx_object)
```

[ ]:
```
#This function returns the concentration vector for the evolved agent.
#The alpha is taken from a solely developing agent which trained on a parent
environment
#This environment can either differ a lot from the test environment or only
slightly
def get_evolved_alpha(psi, normalization,certainty, acc_perception, theta_f,
theta_l, n_iter):

    #initialize soley developing agent and environment
    environment = Environment(theta_f, theta_l, psi)
    n_features  = len(psi)
    n_locations = len(psi[0])
    n_objects   = len(psi[0][0])
    alpha = np.full((n_features,n_locations,n_objects ), 1.0)
    agent = Agent(alpha,certainty=certainty, acc_perception= acc_perception,
developing = True)

    #run simulation given the environment, agent and number of iterations
    simulation = Simulation(environment, [agent] , n_iter)
    simulation.run()

    #return normalized alpha
    return agent.alpha*normalization
```

[ ]:
```
class Change(Enum):
    HIGH = "high"
```

```
    LOW  = "low"

    def __str__(self):
        return "{} inter-generational change".format(self.value)
```

[ ]:
```
###Setting parameters###
n_iter_current = 100
n_iter_parent  = 1000
normalization  = 0.1
certainty      = 0.9
acc_perception = 0.9
change         = Change.HIGH

features = ["red", "orange", "yellow"]
objects  = ["circle", "square", "triangle"]
locations= ["left", "center", "right"]

n_features  = len(features)
n_locations = len(locations)
n_objects = len(objects)

###Initalizing the test environment and parent environment###

psi_test =  np.array([[[0.99998,0.00001,0.00001],[0.00001,0.00001,0.99998],[0.
↪99998,0.00001,0.00001]],
                      [[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]]])

theta_f = np.repeat(1/n_features,n_features)
theta_l = np.repeat(1/n_locations,n_locations)
environment = Environment(theta_f, theta_l, psi_test)

### Choosing the parent environment depending on the amount of inter-generational
change
if change == Change.LOW:
    psi_parent = np.array([[[0.8,0.1,0.1], [0.1,0.1,0.8], [0.8,0.1,0.1]],
                           [[0.1,0.1,0.8], [0.1,0.1,0.8], [0.1,0.1,0.8]],
                           [[0.1,0.8,0.1], [0.1,0.1,0.8], [0.1,0.8,0.1]]])

else:
    psi_parent = np.array([[[0.8,0.1,0.1], [0.1,0.8,0.1], [0.8,0.1,0.1]],
                           [[0.1,0.1,0.8], [0.1,0.8,0.1], [0.1,0.1,0.8]],
                           [[0.1,0.8,0.1], [0.1,0.8,0.1], [0.1,0.8,0.1]]])



###Initializing the agents###

#Initialzing the concentration vector for the developing [d], evolved [e] and
developing and evolved agent [de]
alpha_d  = np.full((n_features, n_locations, n_objects), 1.0)
alpha_e = get_evolved_alpha(psi_parent,normalization, certainty, acc_perception
                            , theta_f, theta_l, n_iter_parent)
alpha_de= np.array(alpha_e) #creating copy due to reference type

#Creating the agents
agent_d  = Agent(alpha_d,  certainty, acc_perception, developing = True)
agent_e  = Agent(alpha_e,  certainty, acc_perception, evolved    = True)
agent_de = Agent(alpha_de, certainty, acc_perception, developing = True, evolved =
True)
```

```
agents = [agent_d, agent_e, agent_de]

###Initializing and running the simulation ###
simulation = Simulation(environment, agents ,n_iter_current)
simulation.run()

### Plotting the prediction errors ###
file = open("Results/{}_{}_{}.txt".format(change, n_iter_current,
normalization),"w")
file.write("Current iterations: {}, \nParent iterations: {},\nNormalization: {},
\nCertainty: {}, \nAccuracy perception: {}\n\n"
           .format(n_iter_current, n_iter_parent, normalization, certainty,
acc_perception))
k=0
means= [[] for _ in range(len(agents))]

colours = ["maroon", "grey", "darkorange"]
fig, ax = plt.subplots(len(features)*len(locations), 1, figsize=(15,15), sharex =
True, sharey=True)
for i in range(len(features)):
    for j in range(len(locations)):
            file.write("Prediction error for feature '{}' and location {}: \n".
 ↪format(features[i],locations[j]))
            for m,agent in enumerate(agents):
                ax[k].plot(agent.pre_errors[i][j], label = agent, color =
colours[m])
                ax[k].set_title("Change in Prediction Error for Feature: '{}' and
Location: '{}'".format(features[i],locations[j]), fontsize=18, loc= 'center')
                #ax[k].set_xlabel("Number of Iterations")
                #ax[k].set_ylabel("Size of Prediction Error")
                ax[k].set_ylim([0,3.5])
                mean =  np.mean(agent.pre_errors[i][j])
                means[m].append(mean)
                file.write("{}: {}\n".format(agent, mean))
            k+=1
            file.
 ↪write("-----------------------------------------------------------------\n\n")
ax[-1].legend(prop={'size': 15}, loc = "lower right")
fig.text(0.5, 0.0, 'Number of Iterations', ha='center', fontsize=15)
fig.text(0.0, 0.5, 'Size of Prediction Error', va='center', rotation='vertical',
fontsize=15)
fig.tight_layout()
for i,agent in enumerate(agents):
    file.write("Mean prediction error for {}: {}\n".format(agent, np.
 ↪mean(means[i])))
file.close()
fig.savefig("Results/{}_{}_{}.png".format(change, n_iter_current, normalization))
```

[ ]:
```
#Calculating the average prediction error across features and locations
def plot_average_pred_err(agents, centre):
    flatten_errors =  []
    for i, agent in enumerate(agents):
        if centre:
            flatten_errors.append([y for x in agent.pre_errors for j,y in
enumerate(x)if j ==1])
        else:
            flatten_errors.append([y for x in agent.pre_errors for j,y in
enumerate(x)])

    sums = np.zeros((len(agents),20))
    occs  = np.zeros((len(agents),20))
```

```
    for a, agent in enumerate(agents):
        for i, array in enumerate(flatten_errors[a]):
            for j, element in enumerate(array):
                sums[a][j]+= element
                occs[a][j]+=1

    average_pre_error =[]
    for a in range(0,len(agents)):
        average_pre_error.append([x/y for x,y in zip(sums[a],occs[a])if y!=0])
    plt.figure(figsize=(10,5))
    colours = ["maroon", "grey", "darkorange"]
    for a, agent in enumerate(agents):
        plt.plot(average_pre_error[a], label =agent, color = colours[a])
        plt.title("Change in Average Prediction Error Over Time ")
        plt.xlabel("Time")
        plt.ylabel("Average Prediction Error")
        plt.legend(prop={'size': 15}, loc = "upper right")
        if centre:
            plt.savefig("Results/Average_prediction_error_centre_{}.png".
↪format(change))
        else:
            plt.savefig("Results/Average_prediction_error_{}.png".format(change))
        plt.ylim((0,3.5))
    plt.show()

#Average prediction error across all locations
plot_average_pred_err(agents, False)
#Average prediction error position centre
plot_average_pred_err(agents, True)
```

Performance for Feature-Location Combinations

Low Inter-generational Change

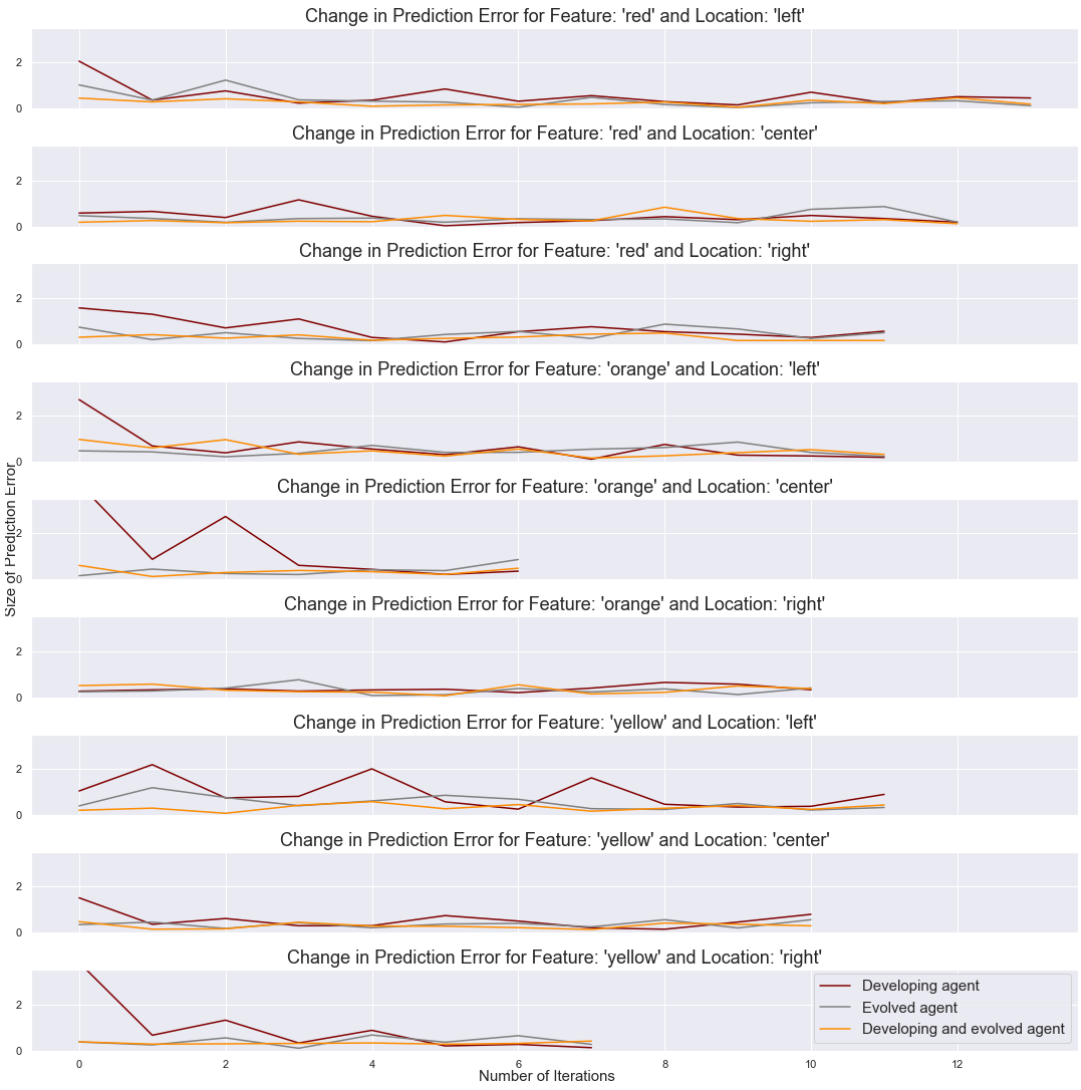


Figure 14: The change in prediction error over time in the low inter-generational change condition for each combination of feature and location: The red lines represent the developing agent; the grey lines the evolved agent and the orange lines the developing and evolved agent. The number of iterations, here observations, differs per feature-location combination as both the feature and location are drawn from a probability distribution by the environment. Even though each feature and location have the same probability, the number of iterations is too small for the observations to be equally distributed over feature and location. The nature of the probability distributions for each feature-location combination that is the rank of the objects based on likelihood is the same as for the test condition. However, the parent environment in the low inter-generational change condition is slightly more variable than the test condition.

High Inter-generational Change

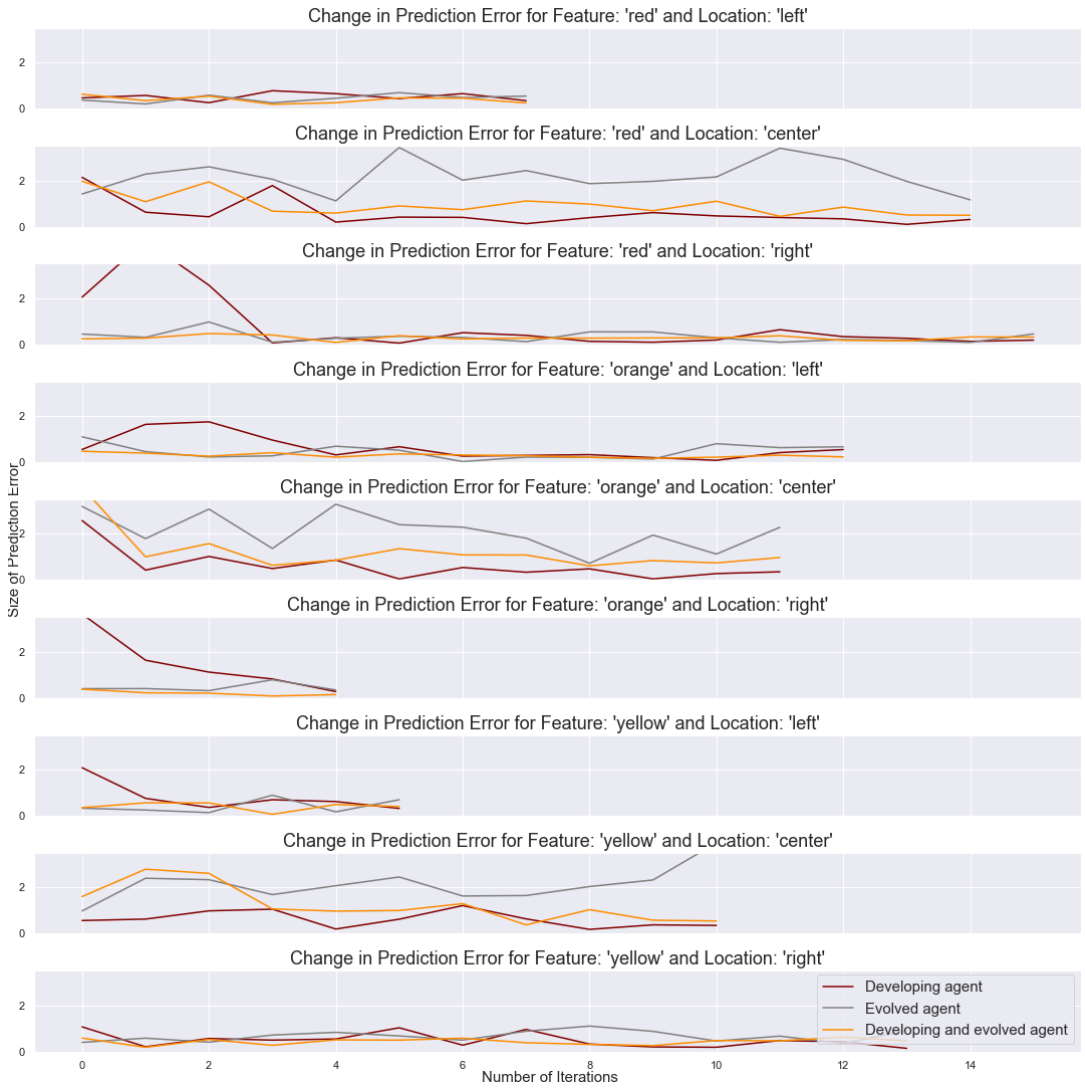


Figure 15: The change in prediction error over time in the high inter-generational change condition for each combination of feature and location: As can be seen in the graph, the red line represents the developing agent, the grey line the evolved agent and the orange line the developing and evolved agent. The number of iterations, here observations, differs per feature location combination as the environment generates the feature and location. Even though each feature and location have the same probability, the number of iterations is too small for the observations to be equally distributed over feature and location. The probability distributions of the locations *right* and *left* did not change compared to the low inter-generational change condition. The probability distribution over the objects for location *center* shows high inter-generational change.

Average Prediction Errors

Low Inter-generational Change

```
Current iterations: 100,
Parent iterations: 1000,
Normalization: 0.1,
Certainty: 0.9,
Accuracy perception: 0.9

Prediction error for feature 'red' and location left:
Developing agent: 0.570350036242214
Evolved agent: 0.39105003666512866
```

```
Developing and evolved agent: 0.26847503009236284
```

```
Prediction error for feature 'red' and location center:
Developing agent: 0.414577114260525
Evolved agent: 0.368650090779198
Developing and evolved agent: 0.2980683531150777
```

```
Prediction error for feature 'red' and location right:
Developing agent: 0.6816358731226647
Evolved agent: 0.44425744330378686
Developing and evolved agent: 0.29220878731154803
```

```
Prediction error for feature 'orange' and location left:
Developing agent: 0.6505922730680568
Evolved agent: 0.4786551675597334
Developing and evolved agent: 0.4886497600998246
```

```
Prediction error for feature 'orange' and location center:
Developing agent: 1.3476078976962944
Evolved agent: 0.40081037459832397
Developing and evolved agent: 0.3607473406981501
```

```
Prediction error for feature 'orange' and location right:
Developing agent: 0.3709937407106791
Evolved agent: 0.3093687311161728
Developing and evolved agent: 0.33935958452372844
```

```
Prediction error for feature 'yellow' and location left:
Developing agent: 0.9455004209735222
Evolved agent: 0.5413994399179806
Developing and evolved agent: 0.32522085693734765
```

```
Prediction error for feature 'yellow' and location center:
Developing agent: 0.5483116569183191
Evolved agent: 0.37092848860684646
Developing and evolved agent: 0.30298947188252756
```

```
Prediction error for feature 'yellow' and location right:
Developing agent: 0.9508237626148097
Evolved agent: 0.4032813654201851
Developing and evolved agent: 0.32455094503773735
```

```
Mean prediction error for Developing agent: 0.7200436417341205
Mean prediction error for Evolved agent: 0.4120445708852618
Mean prediction error for Developing and evolved agent: 0.333363347744256
```

High Inter-generational Change

```
Current iterations: 100,
Parent iterations: 1000,
Normalization: 0.1,
Certainty: 0.9,
Accuracy perception: 0.9

Prediction error for feature 'red' and location left:
Developing agent: 0.5263396034702755
Evolved agent: 0.4548604503967505
Developing and evolved agent: 0.397441943122239
```

```
Prediction error for feature 'red' and location center:
Developing agent: 0.5812897960376431
Evolved agent: 2.1815069546307777
Developing and evolved agent: 0.9369927962519445
```

```
Prediction error for feature 'red' and location right:
Developing agent: 0.7773902595062006
Evolved agent: 0.33191165797386896
Developing and evolved agent: 0.2882270637786911
```

```
Prediction error for feature 'orange' and location left:
Developing agent: 0.6217576843529837
Evolved agent: 0.46482637998099025
Developing and evolved agent: 0.3013910252019348
```

```
Prediction error for feature 'orange' and location center:
Developing agent: 0.6305704288449779
Evolved agent: 2.1119621565902524
Developing and evolved agent: 1.2389488055680034
```

```
Prediction error for feature 'orange' and location right:
Developing agent: 1.499331680004101
Evolved agent: 0.45439264593973216
Developing and evolved agent: 0.21168094035821788
```

```
Prediction error for feature 'yellow' and location left:
Developing agent: 0.8081352929266926
Evolved agent: 0.41630500557711647
Developing and evolved agent: 0.4074579600118089
```

```
Prediction error for feature 'yellow' and location center:
Developing agent: 0.6249270240711584
Evolved agent: 2.1386807797306724
Developing and evolved agent: 1.2658172374062744
```

```
Prediction error for feature 'yellow' and location right:
Developing agent: 0.49290892082318416
Evolved agent: 0.6676395542909039
Developing and evolved agent: 0.4375277969104428
```

```
Mean prediction error for Developing agent: 0.7291834100041352
Mean prediction error for Evolved agent: 1.0246761761234517
Mean prediction error for Developing and evolved agent: 0.609498396512173
```