Predictive Processing
Exploring Multivalent Variables

Author:
Dennis Verheijden [a]
s4455770

[a]Department of Artificial Intelligence

Supervisor:
Johan Kwisthout [a] [b]

[a]Department of Artificial Intelligence
[b]Donders Institute for Brain, Cognition and Behaviour

Radboud University Nijmegen

Bachelor's Thesis in Artificial Intelligence

Artificial Intelligence

June 15, 2017

Abstract

Our research extends the causal Bayesian network implementation of the Predictive Processing Theory to account for multivalent variables. We also propose a framework for solving the exploration-exploitation trade-off in the Bayesian Predictive Processing implementation. Here we use a Q-Learning approach with Dirichlet distributions as hyperpriors and the free-energy principle as a base for learning. The latter links the proposed methods to neural mechanisms in the brain which have been linked to the exploration/exploitation trade-off. We tested our methods via behavioural studies where a robot had to learn an environment from scratch to navigate to a light source.

1 Introduction

A recent view on cognition is that the brain is a prediction machine, which is constantly matching incoming sensory inputs with top-down expectations or predictions (Clark, 2013). This view is used in the Predictive Processing Theory, which states that the entire operation of the brain can be summarized by a simple, unifying principle (Kwisthout, Bekkering, & van Rooij, 2017). This theory proposes that the brain only processes the part of the input that is inconsistent with the predictions of the brain, the prediction error. In the causal Bayesian network implementation of Predictive Processing, the predictions of its inputs are made in a hierarchical manner by generative (causal) models (Kwisthout et al., 2017). These models are updated based on the prediction error, this allows the brain to achieve the concept of learning.

This concept of learning in Bayesian Predictive Processing implementation has been tested via simulation studies where the agent was taught to correctly predict the outcome of a coin-flip (de Wolff, 2017). Predicting the outcome of a coin-flip is a binary decision problem, either the result is heads or tails. However, in real world situations, such as in navigating environments to reach your goal, there are more than two possible outcomes. The causal Bayesian network implementation of Predictive Processing is currently silent about how to deal with these multivalent variables.

That is what we aim to solve: How to extend the computational implementation proposed by Kwisthout et al. to be able to deal with multivalent variables. As an example, we will be using a Markov Decision Problem setting where an agent has to find a light source. The agent is able to detect multiple levels of light intensities, resulting in the light intensity being a multivalent variable. We will also propose a framework for solving the exploration-exploitation trade-off in the Bayesian Predictive Processing implementation.

We aim to achieve this by using a model-free reinforcement learning method, Q-learning. This allows us to navigate the world to achieve our goals without the need of prior information about how the world is shaped. We will extend this

Q-learning framework with Dirichlet distributions, which is a continuous multivariate probability distribution, and with the free-energy principle (Friston, 2010). The Dirichlet distributions are used as hyperpriors, which provide a way of making predictions and learning about the world. The free-energy principle is used to learn an action-value function per state. The latter two will contribute to solving the exploration-exploitation trade-off in the causal Bayesian network implementation of Predictive Processing.

First, we will introduce the Exploration-Exploitation dilemma and how we aim to solve it. Second, we will introduce hyperparameters and their usage. After, we will explain our framework through pseudocode, highlighting the essential parts in subsections. Subsequently, the experiment will be described followed by the results. We will then proceed with the discussion and recommendations for future study. Finally, we will give a conclusion of this thesis.

2 Exploration-Exploitation Dilemma

We are constantly making decisions. Large decisions, like what career to pursue, and smaller decisions, like what we are going to eat for dinner. For example, when choosing what to eat for dinner: do we go for the food that we know we like, like fish and chips, or do we want to explore new options, like Peking Duck? These decisions are all driven by a higher-level process: whether we want to exploit known certainties, or explore new options. These new options may initially seem worse but could also be more profitable after exploring them. This is known as the Exploration-Exploitation Dilemma.

It is still unknown how the trade-off between exploration and exploitation is made (Cohen, McClure, & Angela, 2007). However, most evidence suggests a trade-off between two components: a reward for performing an action and the amount of information gained after performing an action (Friston, 2010). But how this trade-off is made is yet unknown. This raises the question: How does an agent make the trade-off between exploration and exploitation?

A famous example is the multi-armed bandit problem. Here we have a multiple-armed slot machine, a so-called multi-armed bandit. The goal of the agent is to maximize its profits, i.e. subsequently pulling the arm which yields the highest reward. The solution to this problem is found through exploration and exploitation. One strategy for this problem was proposed by Gittins (1979). In this paper, Gittins reduced the multi-armed slot machine to multiple one-armed slot machines. Every one-armed slot machine provides a random reward from a probability distribution that is specific to that arm. Every pull is then rewarded by a straightforward reward function, i.e. 1 for a successful pull and 0 for an unsuccessful pull. Each reward is exponentially discounted as a function over time. The solution to which arm to pull would then be to choose the machine which has the highest expected reward (Gittins, 1979). This value is

known as the Gittins index.

However, this strategy cannot be applied to all real-world exploration-exploitation problems. This is due to the problem space being non-static. When properties of the problem space change, the Gittins index of a previous non-optimal action would still be low, due to new experiences being heavily discounted. When the previously non-optimal action may be the most optimal action in the new problem space.

Since the Predictive Processing Theory aims to explain cognitive processes of the brain, we propose a solution to the exploration-exploitation trade-off that can be linked back to known neurological systems. This way, our framework could serve as a way of modelling the underlying mechanisms of the brain being a prediction machine.

Studies to the exploration-exploitation trade-off in the human brain have proposed several systems which are thought to be involved in this process. These systems are the midbrain dopamine system, the locus coeruleus-norepinephrine (LC-NE) system and the basal forebrain cholinergic and adrenergic systems (Cohen et al., 2007).

The midbrain dopamine system is thought to be involved in reward and prediction errors (Cohen et al., 2007). Rewards and prediction errors contribute to learning about one's environment, as stated in the Predictive Processing theory (Clark, 2013), where the brain only processes that part of the sensory input that is inconsistent with the predictions of the brain.

The LC-NE system is thought to be specifically involved in the exploration-exploitation trade-off (Aston-Jones & Cohen, 2005). The LC neurons exhibit two modes of activity: phasic and tonic. The phasic mode is proposed to be involved in exploitation and the tonic mode in exploration. The LC-NE system could mediate between the necessity of the two components mentioned earlier.

And finally, the basal forebrain cholinergic and adrenergic systems are thought to be involved in monitoring uncertainty (Cohen et al., 2007). Which could play a role in action selection, such that when there is a high uncertainty, one could deviate from choosing the current optimal action and explore potentially better actions.

We aim to solve the exploration-exploitation dilemma by letting the agent learn an action-value function inspired by components from the neural systems mentioned earlier.

Actions that yield high rewards should have high action values and actions that would lead to a gain in information should also have an increased action value. This corresponds to the midbrain dopamine system, where a higher reward would lead to more dopamine release.

Actions that lead to potential worse rewards should not be discounted in situations with high uncertainty. This corresponds to the basal forebrain cholinergic and adrenergic systems and the LC-NE system, where LC neurons exhibit more tonic activity with high uncertainty (Aston-Jones & Cohen, 2005).

For the implementation of this action-value function we will make use of the free-energy principle (Friston, 2010). Here action values are composed of two components, an extrinsic value and an epistemic value. The extrinsic value can be seen as the utility of an action, i.e. the reward of the action. The extrinsic value can be seen as a measure for the information that is gained when the action is executed. Maximizing both the extrinsic and epistemic values leads to minimizing the expected free energy (Friston et al., 2015).

We will let the agent learn the environment through the use of hyperparameters. These capture the cumulative experiences of the agent in the world. This knowledge is embedded in Dirichlet distributions, which store occurrences of events. These distributions can also provide information about the certainty of the predictions an agent makes through a statistical property of the distribution: variance. We will then use the certainty about one's prediction to choose between competing actions. When there is a high uncertainty, the agent should be encouraged to explore rather than to exploit.

3 Hyperparameters

Hyperparameters are parameters of a prior distribution that can be used for a causal Bayesian network. These are the type of networks that are used in the Bayesian implementation of the Predictive Processing theory. The hyperparameters capture the cumulative experiences of the agent in the world. This information is then used to provide knowledge about a prior probability distribution. The parameters are such that the probability density function can be drawn. The probability density function provides information about the predictions of the agent.

In the coin-flip example, this prior distribution could be over the probability $P(X = \text{heads})$. Since there are only two possible outcomes, i.e. heads or tails, we can count the occurrences of heads and tails and use these as parameters for a distribution. In this example, we could use a Beta distribution, denoted as $Beta(\alpha, \beta)$ where α can be the number of heads and β the amount of tails. Examples of these Beta distributions can be seen in figure 1. For example, the Beta distribution *Beta(1,1)* is the initial distribution, here each probability is equally likely. However, in *Beta(15,5)*, the probability on heads is much higher than the probability on tails.

However, Beta distributions can only take two parameters, α and β, while in a real-world situation, this would be insufficient most of the time. For example, when navigating in a dark space and one would want to go to the light source, one would want to be able to distinguish between multiple intensities of light (e.g. no light, some light, prominent light) to locate the light source. A solution to this problem is to use a different distribution that can take multiple parameters: Dirichlet distributions. These distributions are denoted as $Dirichlet([\alpha_1, \alpha_2, \ldots, \alpha_n])$, where α_i is the number of occurrences of the corresponding category. In our experiment, we used three different conditions: α_1: no light,

α_2: some light, α_3: prominent light. The probability density function of a Dirichlet distribution will be plotted as a triangle, where each edge represents one category. The probability density function is plotted as a colour overlay over the triangle. The category with the highest probability density is the most likely category. Examples of Dirichlet distributions can be seen in figure 2. Here the initial distribution is *Dirichlet([1,1,1])* (figure 2a), where every point in the problem space has an equal likelihood. However, in *Dirichlet([15,5,5])* (figure 2d) the probability density is highest near the edge α_1, meaning that the category *No Light* is most likely.

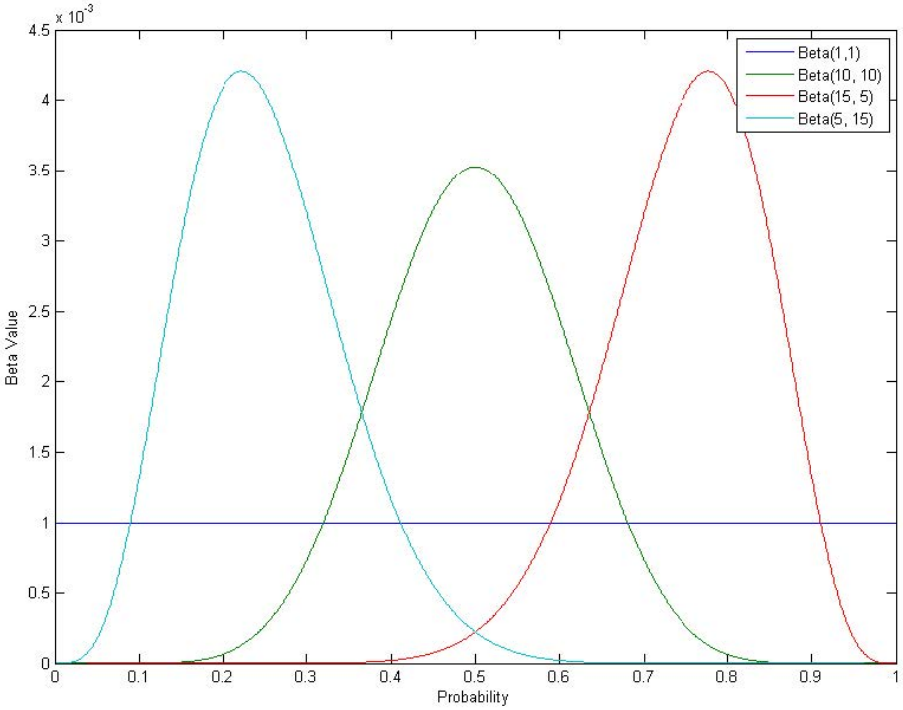
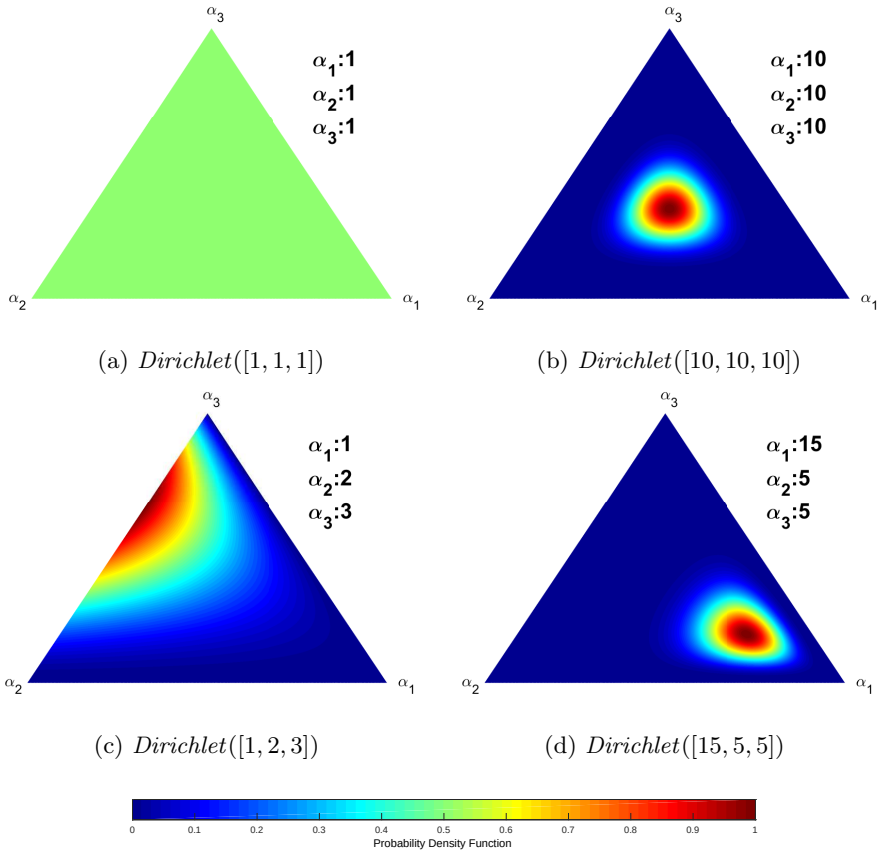


Figure 1: Probability density function of various Beta distributions, from (de Wolff, 2017, figure 1)



(a) *Dirichlet*([1, 1, 1])

(b) *Dirichlet*([10, 10, 10])

(c) *Dirichlet*([1, 2, 3])

(d) *Dirichlet*([15, 5, 5])

Figure 2: Probability density function of various Dirichlet distributions

3.1 Characteristics of Hyperparameters

The parameters that are used to construct the distributions gives us more information than a probability distribution. The parameters can be used to derive statistical properties, which provide us with information about the predictions the agent makes (Friston, 2008). We will be using the following statistical properties: the probability distribution function, the variance and the mean. The probability distribution function will be used to illustrate the distribution, the mean will provide us with the most likely category and its probability, and the variance will be used as a measure of uncertainty over the agent's predictions.

The first statistical property that we will be discussing is the probability density function. This is a function that calculates for every possible value in the sample space, the relative likelihood that the variable would be equal to this value. Informally, the probability distribution function displays the most likely category and its probability. In Beta distributions, this can be plotted as a function of the probability of a variable (see figure 1). However, due to having more

categories in Dirichlet distributions, we plotted the probability density function as a colour overlay in the sample space. Since we used three categories, the sample space is triangle-shaped (see figure 2). A high probability density means that this value is more likely to be drawn out of the distribution. Informally, the category with the highest probability density is the most likely prediction of the agent.

The mean of a distribution is used as the probability of a condition. The condition with the highest probability is the prediction of the agent. The prediction is defined as the category with the highest probability. With Beta distributions, there are two conditions, $X = 1$ or $X = 0$. However, if one knows $P(X = 1)$, then $P(X = 0)$ is also known, since $P(X = 0) = 1 - (P(X = 1)$. The mean of a Beta distribution is given by equation 1. This is illustrated in the probability density plot as the probability where the probability density function is maximized. The agent's prediction is defined as the condition with the highest probability.

$$P(X = 1) = \frac{\alpha}{\alpha + \beta} \tag{1}$$

In Dirichlet distributions you have more than one condition. The probability of a condition is defined by the mean of the corresponding parameter, given by equation 2. The prediction of the agent is the condition that has the highest probability. In the probability density function, this is the condition that corresponds to the edge which minimizes the distance to the maximum probability density.

$$P(X = x_i) = \frac{\alpha_i}{\Sigma_k \alpha_k} \tag{2}$$

The Variance of a distribution is defined as the squared standard deviation of the hyperparameters. It is a measure for statistical dispersion of a set of numbers. The variance of a parameter decreases when parameters are incremented. Thus, with more experience, the variance decreases. To this end, we will use the variance to reason about the certainty of the agent's predictions.

With Beta distributions, the variance of a condition is defined as the variance of α (equation 3). In the probability density function, the variance is illustrated as the width of the peak.

$$\mathrm{Var}(X = 1) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \tag{3}$$

In Dirichlet distributions the variance of a condition is defined as the variance of the corresponding parameter in the distribution, given by equation 4. In the probability density function, the variance is illustrated by the dispersion of the probability density.

$$\begin{aligned} \mathrm{Var}(X = x_i) &= \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0 + 1)} \\ \text{where } \alpha_0 &= \Sigma_k a_k \end{aligned} \tag{4}$$

4 Model Learning

When navigating in an environment, an agent should have a notion of his position in the world, his current state. These states should be an abstract representation of the relevant parts of its environment. Ideally, this state begins in the most basic form, e.g. if there is an obstacle which prevents the agent from moving. After observing high prediction errors, the representation of the states should become more detailed by adding a variable which minimized the prediction error, this is called model revision (Kwisthout et al., 2017; de Wolff, 2017).

However, in this thesis we will be focusing on model updating, rather than model revision, i.e. we will focus on updating the probabilities instead of adding contextual variables. We will be using constant representations of the world that consist of the position of the robot, the orientation the agent is facing and whether there is an obstacle in front of the agent.

We aim to autonomously navigate in a world, potentially without any prior knowledge about the world. We will be needing an online reinforcement learning technique, because we need to learn the environment as we explore and because making actions in a world does not give us any feedback whether that action was optimal or not. That is why we have chosen for a model-free reinforcement learning technique, Q-learning (Watkins & Dayan, 1992).

4.1 Q-Learning

Q-learning is a form of model-free learning which enables users to learn an action-value function by exploring a Markovian domain (Watkins & Dayan, 1992). The action-value function is such, that the utility of an action in a given state is only dependent on the previous experiences when executing that particular action, rather than being dependent on every previous experience leading up to that action. In Q-learning the agent learns the action-values, referred to as Q-Values, in an online fashion, i.e. updating Q-Values after each new experience. In traditional Q-Learning you start with exploring the environment and after the Q-Values converge to a stable value, you start exploiting the gathered knowledge about the world, choosing the action with the highest value. However, this requires a static world. If the world would change, the previously

gathered experience would be outdated resulting in the Q-Values also being outdated. This no longer guarantees choosing the optimal action in a new environment.

To this end we will make some adjustments to the traditional Q-Learning approach. Rather than first exploring the environment and then exploiting the gained knowledge, we will choose our actions based on how certain we are of this action being the most optimal action. We will be using Dirichlet distributions as hyperpriors for our model (section 3). These Dirichlet distributions store our knowledge about the world. If there is a high variance in the probability distribution (section 3.1), we have not gathered enough evidence that our actions are optimal. If we are not certain of our actions being, we should tend to explore more opposed to exploiting potentially sub-optimal actions.

Another adjustment to the tradition Q-Learning approach is in the updating of the Q-Values, where we will be using components of neural systems in the human brain that are proposed to be involved in the exploration-exploitation trade-off as described in section 2.

To further demonstrate the framework, we will be using the pseudocode given in algorithm 1. The algorithm will run for a pre-set number of cycles. It will then run until the agent achieves its goal, reaching the light source location. Every iteration of the algorithm follows a number of steps. First, the algorithm computes valid actions based on the agent's current knowledge of the world and its current state. The agent will then choose one action based on the corresponding Q-Values and the certainty of the action being optimal. This will be further discussed in section 4.2. The agent then proceeds with performing the selected action. After the action has been performed, the agent gathers sensory information about its environment, such as the light intensity and the distance to the nearest object in front of the agent. The distance to the nearest object will then be used to update the internal map of the agent, i.e. whether the agent can move forward. The light intensity will then be used to adjust the Dirichlet distributions, which will be discussed in section 4.3. Finally, the Q-Values of the previous state and selected action will be updated based on the newly gained experience. This will be discussed in section 4.4.

Algorithm 1 Model Learning Pseudocode

```
1: procedure ModelLearning(cycles)
2: Inputs
3:     cycles: amount of cycles
4: Local
5:     QValues: Table containing Q-Values for every possible state-action combination
6:     obstacleField: variable that holds information about where obstacles are
7:     dirichletDistributions: variable holding one Dirichlet distribution per possible state
8:
9: Algorithm
10:    for cycle:=1 to cycles do
11:        currentState ← initialState
12:        repeat
13:            As ← validActions(currentState)
14:            A ← chooseAction(As)
15:            newState ← performAction(A)
16:            obstacleField ← updated based on the newState
17:            dirichletDistributions(newState)
18:                                ← adjustDistribution(newState)
19:            QValues(currentState, A) ← updateQvalues(newState, A)
20:            currentState ← newState
21:        until currentState == light source location
```

4.2 Action Selection

The first thing in every iteration will be to select an action based on our current knowledge of the world. We will be selecting actions based on how certain we are that this action is the most optimal action in our current state.

Since we do not have a static world, we cannot use a simulated annealing approach, where actions are first chosen in a pseudo-random fashion and in later stages in a greedier fashion, i.e. choosing the action with the highest Q-Value. Instead, we will select actions based on certainty. Certainty is provided through previously gathered experience. We will be using the scaled total variance, σ_T, of a Dirichlet distribution (equation 5) as a measure of certainty. The total variance is scaled such that it will produce a high value when there is a high uncertainty and exponentially decays to 1 (see figure 3). We used a γ of 100 in our experiment to achieve this.

$$\sigma_T(M) = 1 + \gamma * \Sigma_{\alpha_i \in M} \mathrm{Var}(\alpha_i) \tag{5}$$

Selecting an action is done through a softmax function, where an action a is chosen with a probability given the Dirichlet distribution M of the current state (equation 6).

$$P(a|M) = \frac{e^{Q(s,a)/\sigma_T(M)}}{\Sigma_a e^{Q(s,a)/\sigma_T(M)}} \tag{6}$$

The total variability of a model M, the Dirichlet distribution, is used as a measure of uncertainty. If σ_T is high, this results in all actions being regarded as equal (equation 7). This is also illustrated in figure 3. Here we have three available actions, each having its own Q-Value. For demonstration purposes, these Q-Values are not updated. Here we see that actions with a higher Q-Value initially have a similar probability of being executed as actions with a lower Q-Value. After being more certain about one's prediction, the probability of the actions being executed diverge. This results in the action with the highest Q-Value being chosen with a higher probability.

$$\lim_{\sigma_T \to \infty} e^{Q(s,a)/\sigma_T} = e^0 = 1 \tag{7}$$

When the agent has no experience the value of σ_T is high, making the three actions almost equivalent to each other. But when the agent gets more experience the value of σ_T drops, making the agent greedier. This results in the agent being more likely to choose the action with the highest Q-Value.

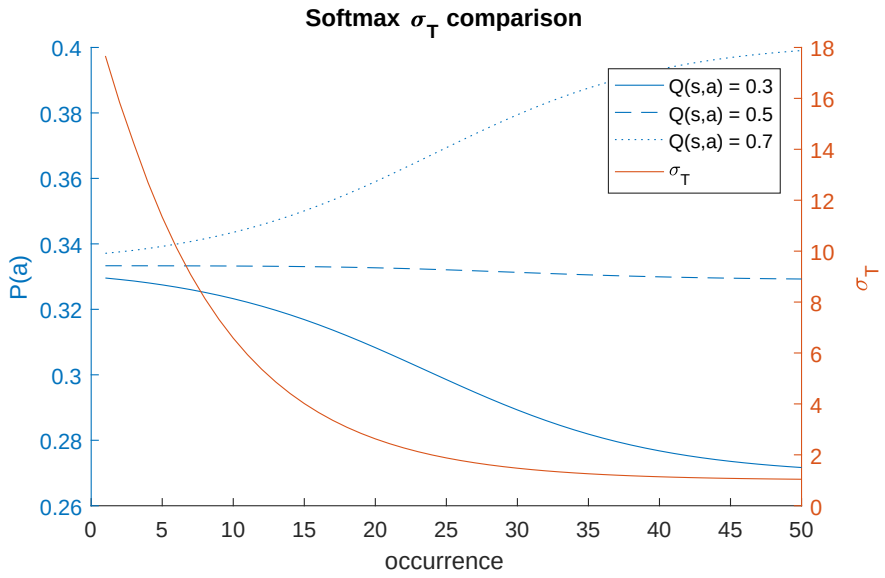


Figure 3: Softmax action selection comparison

4.3 Adjusting Hyperparameters

The hyperparameters are used as the knowledge base of the agent. The hyperparameters store the number of occurrences of the different categories of encountered light intensity per state. The use of Dirichlet distributions enable us to have more than two possible conditions per variable. For illustration purposes, we have chosen to use three categories per distribution: *No Light*, *Some Light* and *Prominent Light*. The respective parameters are α_1, α_2 and α_3 (see section 3).

Updating the hyperparameters is straightforward: after the agent performs an action, it measures the light intensity and decides to what category this light intensity belongs. The corresponding parameter is then incremented by one.

Adjusting the parameters of a Dirichlet distribution changes the characteristics of the distribution, e.g. the variance and mean. This enables the agent to make better predictions and to be more certain about its predictions. Every experience lowers the value of σ_T, making the agent more prone to exploitation, i.e. choose the action with the highest Q-Value, as opposed to exploration.

Adding more experiences to distributions make it a more robust and efficient system. Every experience that is added lowers the variance and increases the certainty of making optimal predictions. However, when examining how predictions are made (equation 1 and 2), one may notice a potentially problematic property: distributions are resistant to change, which is problematic in non-static environments. This is further studied by de Wolff (2017). In this study, de Wolff did simulation studies with coin-flips. In one trial the coin-flips were sorted, i.e. the first 50 flips were heads and the next 50 flips were tails. The results are given in figure 4. Here one can see that the distribution was first slanted all the way to left due to the first 50 trials being heads. After the 50 trials of tails, the distribution was shifted to the middle, indicating a probability of heads being 0.5. With this study, de Wolff pointed out the robustness of the distributions and shows that the prediction error might not be the best metric for error. Instead de Wolff suggests a weighted prediction error where the prediction error is discounted over time.

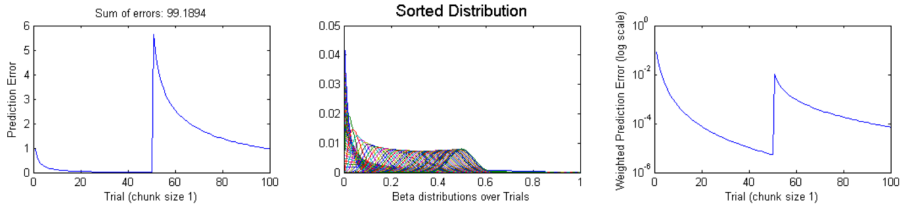


Figure 4: Robustness of distributions, adapted from (de Wolff, 2017, figure 3)

We solve this problem by adjusting the hyperparameters based on the prediction error: when there is a high prediction error, the agent should be inclined to adjust its prior beliefs such that the agent gets more uncertain. This would result in the agent exploring other options. A solution can be found in the equation for variance (equation 4). Here we can see that a solution would be to lower the magnitude of the parameters, while avoiding changing the ratio of these parameters, i.e. the probability distribution should remain the same. To this end we can divide the parameters by a factor and pick the ceiling of these values. This way we avoid making big changes in the probability distribution and still achieve making the agent more uncertain about its predictions (see figure 5). Here we divided the parameters with a factor of 3. The probability density function is more dispersed but the maximum density remained on the same position, meaning that the mean did not change but the variance did change (see section 3.1).

The factor by which the parameters are divided defines how the agent reacts to prediction errors. A higher factor leads to the agent being quick to adapt to prediction errors. However, when a high factor is used, it can also lead to the system never being able to converge to a stable state. This is the case when the perceived light intensity is a border case. After perceiving the light intensity as a different category, the agent would divide the parameters with a high factor. This could result in constantly resetting the distribution.

To this end we have chosen to use a factor of 2. This factor was experimentally defined such that the system is resistant to varying judgement of a perceived light intensity, but flexible enough to account for a non-static environment.

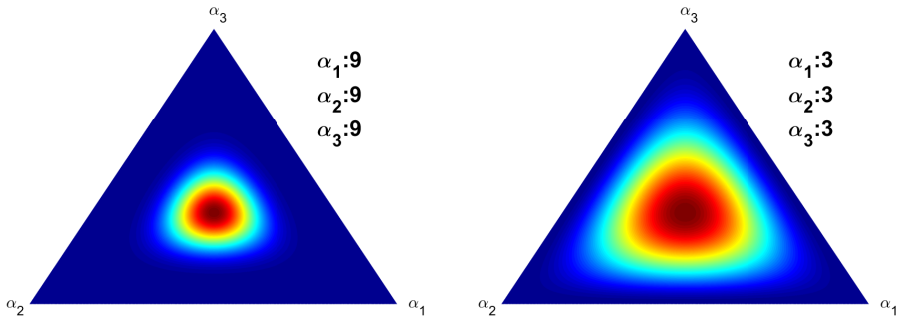


Figure 5: Changing the parameters of a Dirichlet distribution to make an agent more uncertain by dividing the parameters with a factor (3 in this example), without altering the prediction itself.

4.4 Updating Q-Values

Updating Q-Values is vital for the agent when exploring environments. If all actions have equal Q-Values, every action would be regarded as optimal. We want to design the action-value function such that it follows the free-energy principle. The free-energy principle is an information theory where the surprise on sampling some data is minimized (Friston, 2010). The action-value function should thus promote actions which lead to a high information gain.

To this end the Q-Values should be composed of two components: the corresponding reward of performing the action and a reward for performing actions which lead to a high information gain. We implemented this by using the free-energy framework (Friston, 2010). Here it is proposed that the key to the exploration-exploitation trade-off is the minimization of expected free energy of future outcomes. This is done through maximizing extrinsic and epistemic values. Where the extrinsic value corresponds to the expected utility of performing an action and the epistemic value corresponds to the expected information gain (Friston et al., 2015). The idea is that if there is a high information gain when performing an action, there could be some unexplored utility left. This promotes exploring actions with a lower utility that potentially have unexplored utility.

We define the utility of performing an action as the average weighted probability $\overline{p}_w$ of the probability distribution of the resulting state, given by equation 8. The average weighted probability is defined as the weighted sum of the probabilities in the probability distribution, divided by the number of categories in the variable. The weights are defined by equation 9. This equation gives the average weighted probability such that probabilities in higher categories, like *Prominent Light*, are weighted heavier than probabilities in lower categories, like *No Light*.

$$\overline{p}_w(P) = \frac{\Sigma_i P(i) * w_p}{|P|} \tag{8}$$

$$w_p = \begin{cases} \frac{1}{3}, & \text{if category is “no light”} \\ \frac{2}{3}, & \text{if category is “some light”} \\ 1, & \text{if category is “prominent light”} \end{cases} \tag{9}$$

To make the agent less greedy we also added a predicted future utility $\hat{u}$ to the extrinsic value, given by equation 10. The predicted future utility is defined as the maximal Q-Value of the actions in a state. By adding this aspect, we can promote going to states which result in states with high-value actions. This is particularly useful when the agent is in a dark corridor with nothing but darkness surrounding it. The predicted future utility would then promote actions which lead the agent out of this corridor to states with a higher light intensity.

$$\hat{u}(s) = \max_a(Q(s,a)) \tag{10}$$

We defined the expected information gain E as the statistical distance between the probability distribution before experiencing the new information versus after experiencing the new information (equation 12). We used the Kullback-Leibler divergence (Kullback & Leibler, 1951) as a measure for statistical distance, given by equation 11.

$$D_{KL}(P||Q) = \Sigma_i P(i) \log \frac{P(i)}{Q(i)} \tag{11}$$

$$E(s,a) = D_{KL}(P(s)||P(s|a)) \tag{12}$$

However, the range of $\overline{p}_w$ is [0 0.33], $\hat{u}$ is [0 1.0] and E is [0 0.07], so we have to scale these variables such that their maximum values sum up to one and such that the most important factor has the highest weight. To this end we weigh each variable. This results in the update rule of equation 13. Where the new Q-Values are the sum of the average weighted probability of the resulting state, the predicted future utility of the resulting state and the expected information gain in the current state.

$$\begin{aligned} Q(s_i,a) &= \text{extrinsic value} && + \text{epistemic value} \\ &= \gamma_{\overline{p}_w} * \overline{p}_w(P(s_{i+1})) + \gamma_{\hat{u}} * \hat{u}(s_{i+1}) && + \gamma_E * E(s_i,a) \end{aligned} \tag{13}$$

$$\text{where} \quad \gamma_{\overline{p}_w} = 2, \quad \gamma_{\hat{u}} = 0.25, \quad \gamma_E = 2$$

5 The Experiment

We wanted to test if the suggested framework is able to learn the location of a light source. We want to achieve this by making use of a multivalent variable. The multivalent variable that was used is *Light Intensity* and its values are *No Light*, *Some Light* and *Prominent Light*.

To test this, we designed a maze that perfectly utilizes our framework. The designed grid consists of two corridors. At the end of each corridor there was a light source that could illuminate the grid. Every space on the grid would have a light intensity that was a value in the variable *Light Intensity*.

5.1 The Robot

In this experiment, we used a LEGO EV3 robot (figure 6). The robot could perform three actions: turn right, turn left or move forward. The robot is outfitted with two sensors: an ultrasonic sensor and a colour sensor. The ultrasonic sensor was used to detect obstacles in its path, i.e. if the robot could move forward or not. The colour sensor was used as a light sensor, sampling only the intensity of the light, rather than the actual colour. The ultrasonic sensor was placed such that the robot could detect what was directly in front of it. The colour sensor was placed such that when the robot stood in front of the light source, the sensor would be at the same height, directly sampling the light's origin. Every sample from these sensors was an average of 5 samples to reduce the possibility of false samples.

Due to the limited processing capacity of the LEGO EV3 robots, we worked with a master-slave setup. Where the robot would act as a slave, executing all the tasks that were given from the host, a laptop. The host would be the "brains" of the robot, making all the calculations and keeping track of the robot's beliefs about the world.

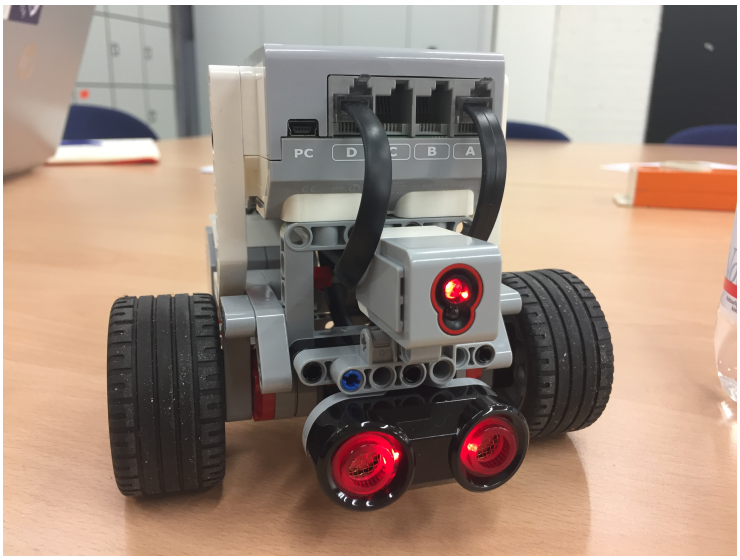


Figure 6: The LEGO EV3 robot used in the experiment, outfitted with a colour sensor (top-sensor) and an ultrasonic sensor (bottom-sensor)

5.2 The World

The world that was used consists of two corridors, the North and East corridor (figure 7). One corridor in front of the robot and one to the right of the robot. The robot sees the world in a grid-like manner. Where one forward-movement corresponds to moving one space forward in the grid. The starting location of the robot was always at the point where the two corridors met and its orientation would always be to the north, directly facing the North corridor. At the end of each corridor was a light source. Only one light source was turned on at a time, with the light source at the east being turned on first. The light sources where both aimed at the initial location of the robot. The light sources were placed such that the light would shine directly into the robot's colour sensor if the robot stood right in front of them. When the robot moves closer to the light source, the light intensity measured by the robot would get higher. But when the robot was not directly looking at the light source, the light intensity would drop to zero. The robot does not initially know where the walls are and learns these as it explores the world. As such, the robot also does not know that the world only consists of two corridors.

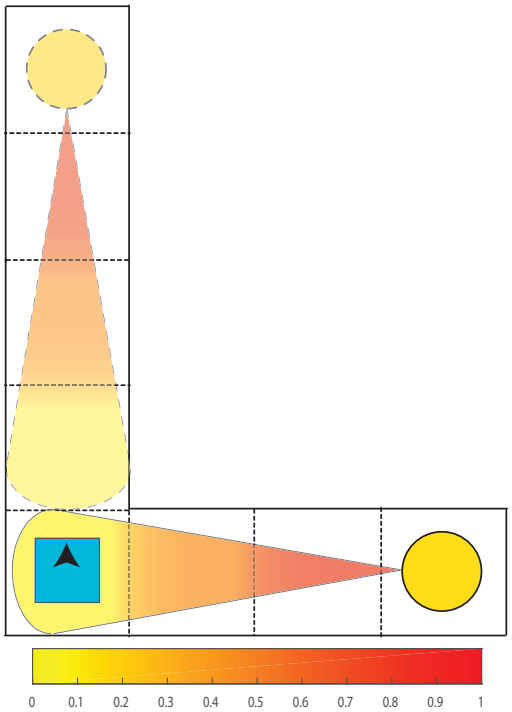


Figure 7: The grid that was used in the experiment: The dashed light source and corresponding bundle is the location of the light source that was initially turned off. The colour in the light bundle corresponds to the light intensity of the light source at that location, where a warmer colour depicts a higher light intensity. The blue box at the left-bottom location is the starting location of the robot, the arrow depicts the orientation the robot is facing

5.3 The Task

The robot knew three different intensities of light: *No Light*, *Some Light* and *Prominent Light*. These three categories are respectively denoted as α_1, α_2 and α_3. In figure 7 these different intensities of light are shown as a colour overlay over the environment, where warmer colours indicate a higher light intensity. The task of the robot was to go to the light source that was turned on. This was achieved by following the framework outlined in section 4. At the beginning of the experiment, the light source in the east corridor was turned on. After finding the light source location, the robot would be moved back to the initial position.

After the robot had learned the initial location of the light source, i.e. that the light source in the east corridor was turned on and the light source in the north corridor was turned off, the light sources were flipped. Now the light source in the north corridor was turned on and the light source in the east corridor was turned off. The robot now had to learn that the light source in the east corridor was turned off and the light source in the north corridor was turned on. Or in terms of the model: The robot had to that there is no light in the east corridor, and learn that there is light in the north corridor.

After the robot had learned that the light source in the north corridor was turned on and the robot could reliably find the optimal path to the light source, the light sources were flipped again. Now the robot had to learn that the light source in the east corridor was turned on again. We did this to study the impact of having to learn a previously known light source location.

6 Results

The model is initialized with the Q-Values for each action being equal. This was done to avoid giving the agent a (dis)advantage. They were initialized at a value of 0.5. This value is chosen because it does not provide the agent with any information nor bias about an action. A value of 0 would discourage exploration due to every unexplored action being worse than an explored action. A value of 1 would cause the opposite effect, it would encourage the agent to immediately explore every action at least once. With a value of 0.5, we do not get such biases.

The Dirichlet distributions are initialized as $Dirichlet([1, 1, 1])$, to give the agent no prior information about the world.

To maintain relevancy and readability, we will only present a subset of all the results. Since every space in the world contains eight states with each three possible actions, the number of plots grow exponentially. We will thus be focussing on the most informative plots.

The most interesting states for the plots containing Q-Values are: in the starting location, turning right to the East corridor and moving forward to the North corridor; in the corridors, moving towards the light sources. The most interesting states for the plots of the Dirichlet distributions are the resulting states after performing the action of the corresponding Q-Value plot. We will not give a plot for every observed distribution through time to maintain relevancy and readability. Instead we will highlight distributions at times where they provide us with the most information: at the end of learning the first and second light source location, and during learning the second light source location.

We will discuss the results in chronological fashion. We will be starting with learning the initial light source. Then proceed with flipping the light sources, discussing what happens in this trial. After, we will discuss how the model converges back to a stable state during the flip. After the model converges we will flip the light sources to their original states and discuss the effects on learning the model. Finally, we will discuss the time it took the agent to learn the environment by using the number of steps required as a measure of the effort made by the agent.

6.1 Learning the Initial Light Source Location

When learning a new light source location, the starting location is particularly interesting because here the agent has to choose which corridor to explore. The corridor where the light source was turned on first was at the end of the East corridor. The hypothesis being that the Q-Value for the initial location and turning right (figure 8) should be higher than the Q-Value for the initial location going forward (figure 10).

When looking at the two plots, we can see that the initial Q-Values for going into the wrong direction are roughly equal to the Q-value for going into the correct direction. But after performing every action at least twice the Q-Value for *Turn Right* is significantly higher. Even though the Q-Value is significantly higher, the action *Forward* is still chosen three times. This is due to the agent still having a high uncertainty, which promotes exploration over exploitation. After having enough occurrences, the agent is more confident over its predictions and the action *Turn Right* is consistently chosen over *Forward*. When having learned the light source location, the Q-Values remain constant. It is worth noting that the agent reported having seen *No Light* once when performing the action *Turn Right*. This is reflected in figure 8 by the sudden increases in epistemic value and in figure 9a having a higher variance and the parameters having non-one values for α_1.

Proceeding to the last locations we want to examine: the corridor positions just before moving to a potential light source location. The initial light source location was in the East corridor, the hypothesis would be that the Q-Values in the East corridor (figure 12) should be higher than the Q-Values in the North corridor (figure 14).

This is indeed the case, when the agent entered the east corridor the Q-Value (figure 12) jumped from the initial value, 0.5, to 1, which is the reward for being at the light source. This Q-Value did not change until the light source location was flipped. This is also reflected in the Dirichlet distributions since the parameters of α_1 and α_2 remained *1* and the parameter α_3 was non-one (figure 13). This means that the agent only saw the highest intensity of light while at the end of the East corridor.

The Q-Value in the North corridor (figure 14) first jumped to a value over 0.6 due to the high information gain when the action was performed, but after performing the action three times the value dropped since the information gain

was non-significant and since there was no light detected at the location. The agent never found light in this location, reflected in the Dirichlet Distribution being shifted to α_1, indicating *No Light*.

6.2 First Light Source Location Change

After the agent consistently chose the most optimal path to the light source, the light sources were flipped, the active light source now being in the North corridor instead of the East corridor.
In the starting position the action *Forward* should now become the dominant action instead of *Turn Left*.

The Q-Value for *Turn Right* marginally changed after the first occurrence of the action. This was because lowering the parameters would be the last step of the algorithm, i.e. after the Q-Values were calculated. This resulted in the distribution in figure 9b. After multiple occurrences of *Turn Right*, the Q-Value dropped and the agent had learned that there was a low possibility of light, reflected in figure 9c.

The Q-Values for *Forward* increased faster than the Q-Values for *Turn Right* decreased, this was due to the lower magnitude of the Dirichlet distribution for performing the action *Forward* (figure 9a vs. figure 11a). After more occurrences of the action, it became the dominant action. The Q-Values quickly converged resulting in the agent having learned that there was a high probability that there was light in the North corridor, reflected in the Dirichlet distribution in figure 11c.

The Q-Values in the corridor positions both changed with a significant amount (figures 12-14). With the Q-Values in the North corridor immediately changing to 1.0 and in the East corridor decreasing to a value around 0.4. This rapid change is best illustrated in the plots of the Dirichlet distributions. In the East corridor: The rapid decrease in α_3, learning that there was a low probability of a high light intensity (figure 13b). The increase in α_1, learning that there was a high probability of a low light intensity (figure 13c).

In the North corridor: The rapid decrease of α_1, learning that there was a low probability of a low light intensity (figure 15b). The increase of α_3, learning that there was a high probability of a high light intensity.

6.3 Second Light Source Location Change

After the agent consistently chose the most optimal path to the light source, the light sources were flipped back to their original state. The light source in the North corridor was now turned off and the light source at the East corridor was turned back on. This meant that the dominant action in the initial position should again become *Turn Right* instead of *Forward*.

The learning process was similar to when the agent had to learn the initial light source location. The difference being the speed at which the learning process occurred. The Q-Values of the action *Turn Right* (figure 8) steadily converged back to the Q-Value when the agent had to learn the initial light source location. And the Q-Values of the action *Forward* quickly decreased to a value of around 0.4. This resulted in similar distributions to the distributions when learning the initial light source location (figure 9a-d, figure 11a-d).

For the Q-Values of the actions in corridor positions, the learning process was similar as well, except for the Q-Values in the North corridor (figure 14). These values decreased below the previous minimum value. This was because the surrounding Q-Values leading up to this state also had to decrease to their minimum value, resulting in more occurrences of action leading back to the end of the North Corridor. This resulted in similar distributions as when the agent had to learn the initial light source location (figure 13a-d, figure 15a-d).

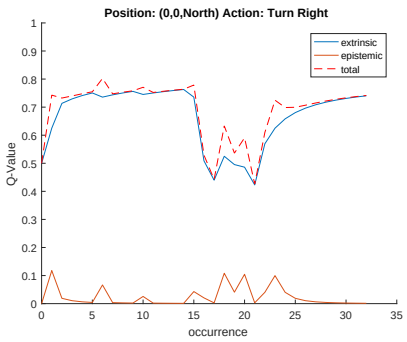
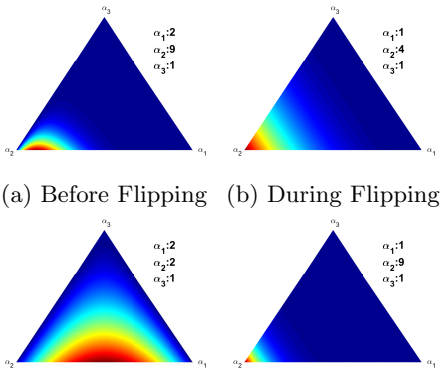


Figure 8: The Q-Value composition for the state with position: (0,0), orientation: North and action: Turn Right



(a) Before Flipping (b) During Flipping

(c) After Flipping (d) End Distribution

Figure 9: Dirichlet distributions for state with position: (0,0) and orientation: East

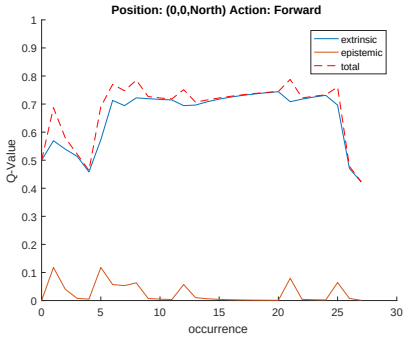
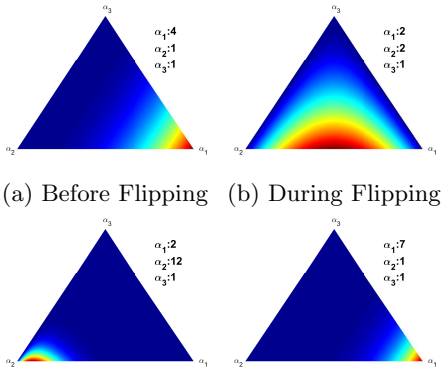


Figure 10: The Q-Value composition for the state with position: (0,0), orientation: North and action: Forward



(a) Before Flipping (b) During Flipping

(c) After Flipping (d) End Distribution

Figure 11: Dirichlet distributions for state with position: (0,1) and orientation: North

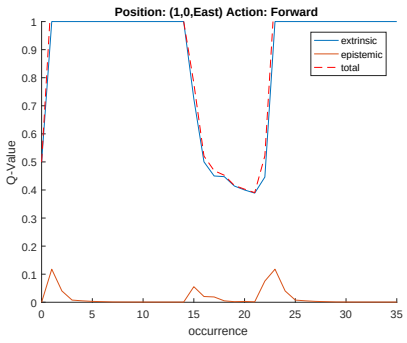
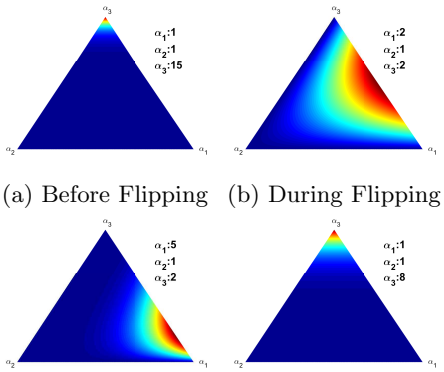


Figure 12: The Q-Value composition for the state with position: (1,0), orientation: East and action: Forward



(a) Before Flipping (b) During Flipping

(c) After Flipping (d) End Distribution

Figure 13: Dirichlet distributions for state with position: (2,0) and orientation: East

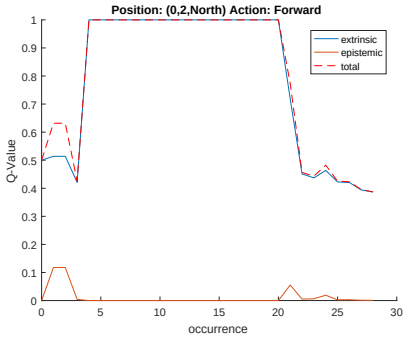
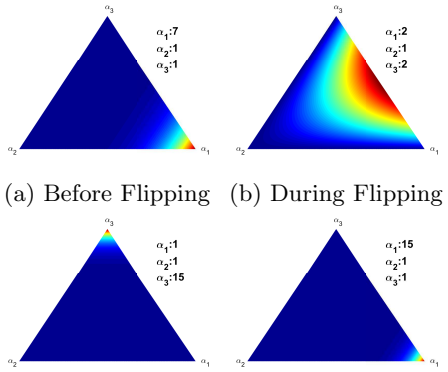


Figure 14: The Q-Value composition for the state with position: (0,2), orientation: North and action: Forward



(a) Before Flipping (b) During Flipping

(c) After Flipping (d) End Distribution

Figure 15: Dirichlet distributions for state with position: (0,3) and orientation: North.

6.4 Learning time

The number of steps required to reach the light source location are plotted in figure 16. The initial trial had an almost minimal number of required steps to find the light source, this was because the robot chose the correct corridor by chance (see section 4.2). In the second trial, the correct corridor was chosen again, which resulted in the agent finding the most optimal path.

In the third attempt, the agent chose the North corridor. This resulted in the agent exploring the whole corridor, but with no avail since there was no active light source in the North corridor. This is reflected in figure 16 by the peak in trial 3. After the agent learned that there was no light in the North corridor, he backtracked to the starting position. After performing many actions in the starting location, the agent got more certain about its predictions, resulting in consistently choosing the action with the highest Q-Value. This led the agent to the East corridor where the agent already found the most optimal path to the active light source.

After the agent had consistently chosen the optimal path, the light sources were flipped. This happened at trial 15, denoted by the arrow in figure 16. The agent was still certain about its predictions, so the agent kept exploring the East corridor. This was because the Q-Value of actions leading to the end of the East corridor were higher than the Q-Values of actions leading to the North corridor. This is reflected in figure 16 by the peak at trial 15.

After the agent had found the new active light source for the first time, there were still some Q-Values that promoted actions to enter the East corridor. These remnants resulted in the small bumps at trial 17 and 20.

After the agent had consistently chosen the optimal path, the light sources were flipped to the original state. This happened at trial 32. Flipping the light sources to their original state resulted in the same behaviour as in trial 15.

Comparing the peaks at trial 3 and 32 displays that the agent had more trouble learning the light source's location in the 32nd trial than in the initial trials. This was because the agent first had to learn that there was no active light source in the North corridor.

Although the peaks at trial 15 and 32 are arguably of the same size, one cannot argue that it takes the same amount of effort to learn a new light source location without further experiments.

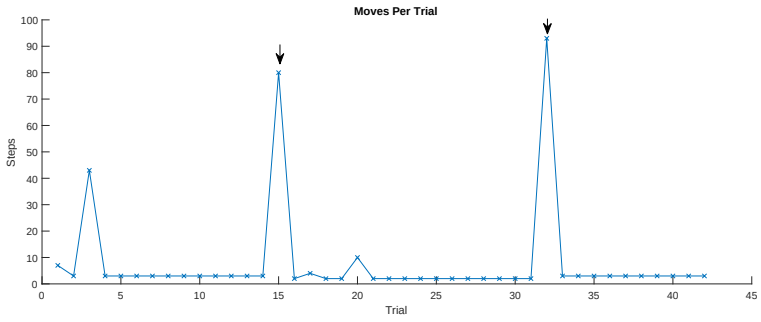


Figure 16: The number of steps per trial, the arrows denote the trial in which the light sources were flipped

7 Discussion & Future Study

One of the shortcomings of this algorithm is displayed in figure 16: When flipping the light sources to their original state, it took about twice as long for the algorithm to converge to a stable state than when the robot initially had to learn the place of the light source. This is because the agent first had to learn that there was no light source at the previous location and relearn that the light source was at the initial location. A possible improvement would be to implement a memory structure such as is present in the human brain: After consolidation of the memory of the light source location in the short-term memory, it may be pushed to the long-term memory. The robot could then switch between known locations if the prediction error is high due to the light source not being where the agent predicted it to be. This could be implemented as another prior for the model.

The way the hyperparameters are updated due to high prediction errors, discussed in section 4.3, could also be improved: Instead of only dividing the parameters by a factor, it may be an improvement to also redistribute the parameter values such that the prediction error is minimized. Another solution could be model revision, where irreducible prediction errors lead to the revision of the structure of the model by adding contextual variables (Oetringer, 2017).

In a pilot study where we could detect seven different categories of light-intensities, we found that the steps that it took for the robot to realise there was no active light source in a corridor, were significantly decreased. We speculate that this was due to the fact that the difference between Q-Values of *No Light* and *Prominent Light* are bigger when there are more categories the agent could detect (see equation 8). This improvement is vital in bigger and more complex grids, since the Q-Values for successive states could become equal due to the inability to detect a higher light intensity.

8 Conclusions

Our research provides a way of how to deal with multivalent variables in the causal Bayesian network implementation of Predictive Processing, through the use of Dirichlet distributions, hereby extending the recent work of (de Wolff, 2017).

We have also provided a framework for how the Bayesian Predictive Processing implementation could solve the exploration-exploitation trade-off. This was done by implementing a Q-Learning framework that used Dirichlet distributions for storing information about the world. The action-selection method incorporates the agent's certainty about its predictions, promoting exploration when there is a high uncertainty. The action-value updating rule uses a free-energy approach incorporating predictions about future states. The effectiveness of this approach is best illustrated in figure 8. When the light sources were flipped for the first time, there is a decrease of extrinsic value, as there is no light in this location. However, since there is a large information gain the total Q-Value increases. This promoted the selection of the action to explore potential information gain.

The action-selection and action-value updating methods have been linked to proposed neurological systems that are involved in the exploration-exploitation trade-off in the brain. This ensures that the proposed framework could serve as a way of modelling the underlying mechanisms of the brain as a prediction machine.

References

Aston-Jones, G., & Cohen, J. D. (2005). An integrative theory of locus coeruleus-norepinephrine function: adaptive gain and optimal performance. *Annu. Rev. Neurosci.*, *28*, 403–450.

Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, *36*(03), 181–204.

Cohen, J. D., McClure, S. M., & Angela, J. Y. (2007). Should i stay or should i go? how the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *362*(1481), 933–942.

de Wolff, E. (2017). *Bursting with error: Dealing with unexpected prediction error in babybots* (Bachelor's Thesis). Radboud University Nijmegen, Comeniuslaan 4, 6525 HP Nijmegen.

Friston, K. (2008). Hierarchical models in the brain. *PLoS Comput Biol*, *4*(11), e1000211.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, *11*(2), 127–138.

Friston, K., Rigoli, F., Ognibene, D., Mathys, C., Fitzgerald, T., & Pezzulo, G. (2015). Active inference and epistemic value. *Cognitive neuroscience*, *6*(4), 187–214.

Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 148–177.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, *22*(1), 79–86.

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*, 84-92.

Oetringer, D. (2017). *Learning to predict with contextual variables: The importance of salience* (Bachelor's Thesis). Radboud University Nijmegen, Comeniuslaan 4, 6525 HP Nijmegen.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.