Imperial College London

Department of Computing

The Human Gamma Project

Reverse engineering the human discount factor in Reinforcement Learning tasks.

by

Bernardo Lemos Caldas

Submitted in partial fulfilment of the requirements for the MRes Degree in Advanced Computing of Imperial College London

September 2013

Abstract

Human learning behaviour has been studied by many fields of science, including Psychology, Economics and Neuroscience. Neuroscience, in particular, has provided us with knowledge on the biological basis of learning, which allows us to attempt to quantify it. Being able to do so may provide a way to diagnose neurodegenerative diseases such as Parkinson's, where changes to the learning patterns happen. Previous studies have shown that there is evidence for the applicability of Reinforcement Learning in humans. In this project we present a possible method to quantify one its parameters, the discount factor, a measure of the preference for small immediate rewards over larger delayed rewards. Here, we propose and test a new set of four experiments for inferring this parameter, together with two inference methods. We tested the methods using both synthetic and experimental data and found the methods to be accurate in a wide range of scenarios. We observed that the discount factor that is used depends on the task being performed, having found that one subset of the tasks had significantly higher discount factors than the rest. The reported results show that the proposed method is capable of quantifying one of the parameters of human learning, paving the way for a more complete quantification of the entire model.

Acknowledgments

I would like to express my gratitude to my supervisors, Dr Aldo Faisal and Dr Luke Dickens, for their support, guidance, and valuable ideas throughout the project period. I would like to particularly thank Dr Luke Dickens for our weekly meetings, which were of crucial importance during these months.

I would also like to like to thank Dr Guy-Bart Stan for his help in introducing me to this project and for his constructive criticism.

I wish to address a work of appreciation to all the people who participated in my experiments, particularly for their availability and enthusiasm.

Finally, I would like to thank my family, my friends and my girlfriend Mafalda. Their constant support and feedback together with their high expectations of me have motivated me to do a project of which I am proud.

Contents

Chapter 1

Introduction

1.1 Motivation

The quest for describing human learning has been undertaken by many since the classical age. With the beginnings of psychology as a science, in the 19th century, we began to have more precise information regarding the process of learning, mainly by describing its behavioural aspects. The better we understand the process of learning the better we can learn and this very reason has driven many researchers from many different areas to try to solve one of the most complex questions of our time: how do people learn. Even with the immense computational power in our hands we struggle to make computers learn even the simplest of tasks. However, with the advent of neuroscience, we began to have access to tools that allow us to understand the biological aspects of learning, to understand how learning is implemented in the most powerful processor known, the human brain. Thus, the possibility of not only understanding learning but also being able to quantify it has become a reality. This project aims to contribute in this endeavour.

Every quantification attempt needs to be based on a model, and the quantification of learning is no exception. One such possible model is reinforcement learning (RL), a framework that studies the way that systems can learn to behave optimally in environments where actions lead from a situation to another and may have associated rewards or penalties. We can think of environments that match this description in diverse areas, such as psychology, economics, ethology and control theory. To some extent we can think of this model as an "ideal learner", a computational model of complex animal decision-making processes. However, RL seems to be more than a simple computational model, since some mechanisms, such as the temporal difference learning rule, seem to be directly instantiated in neural structures, such as the activity of dopamine neurons [14]. Studies [29] have shown the presence of reward prediction error signals in animal decision making tasks. Even in humans there is evidence of the presence of these signals, specifically in the ventral striatum and orbitofrontal cortex [19]. This makes RL a very good candidate to be a transparent model of human learning, since there is the possibility of mapping its processes to neural mechanisms. For this reason, RL was chosen as the main framework for our work.

Having a model, we are then faced with the task of choosing what to quantify. The multitude of possible parameters to address learning using RL makes it impossible to attempt to quantify all of them. Therefore, we have chosen one of the most common parameters in reinforcement learning algorithms, the discount factor. The discount factor is a measure of how much the value of a reward decreases with the delay that precedes it. It is crucial to RL algorithms because they are often based on calculating the expected reward of a certain situation or state, and that involves comparing rewards that come in different points in time. In many real world situations waiting for a reward means missing other rewards, and that is why the value of the reward decreases with the delay that is associated with it. Our work will attempt to quantify the discount factor (usually associated with the greek letter γ) of humans while performing different tasks.

Being able to numerically describe human learning has innumerous advantages. The one that motivates this work is being able to use this knowledge to devise a diagnostic test for neurodegenerative diseases. 37 million people worldwide suffer from neurodegenerative diseases, with many

of these patients being at an advanced age. These diseases get progressively worse and more expensive to treat with time before first treatment, so there is an overwhelming urge for a low-cost, non-invasive diagnostic technique that can work at an early stage. Being able to quantify the learning performance, we could test how individuals with and without certain neurodegenerative diseases learn. By analysing the breakdown of learning techniques for both sets of subjects, we could use algorithmic differences and neurological understanding to diagnose the disease.

The potential to use the way people learn as a diagnostic tool can be illustrated by considering Parkinson's disease. Parkinson's, a relatively prevalent neurodegenerative disease, is caused by the apoptosis of dopamine producing cells. As was previously described, human learning algorithm seems to use dopamine (amongst other factors) to control learning [14]. By isolating the aspects of human learning algorithm that are affected by the lowered dopamine outputs, we can create tests to show if the patient has Parkinson's. This argument can be extended to other neurodegenerative diseases. These methods would be able to chart how the mind is degenerating, and therefore are likely to provide early-stage diagnosis.

Therefore, this project will present two main contributions: a set of experiments and a set of algorithms that allow the deduction of the discount factor that is used by humans in reinforcement learning tasks.

1.2 Literature Survey

1.2.1 Neuroscientific evidence for RL

Model-based vs. model-free It is a common idea in neuroscience that the brain contains multiple systems for decision making [6, 21]. Two long standing ideas include the "law of effect" or "habits" [35], in which successful actions are more likely to be repeated in the future, and "goal directed planning" or "cognitive maps" [36], where the choices are made based on the likely outcomes of their actions. In the neuroscientific perspective, these ideas are thought to coexist in different structures in the brain, controlling the behaviour under different circumstances [3]. Computationally, these ideas can be mapped to model-free (habits) and model-based (goal directed) RL. The difference between these two methods is explained in detail in Chapter 2, but, in short, a model-free approach learns the value of an action directly, while a model-based approach learns separately the likely outcomes for an action and the value of those outcomes. Evidence for both model-free [19, 29] and model-based [38, 12] have been found in the brain, by finding correlations with signals in different regions of the brain. A recent study [7] has found evidence, for the first time, for these two methods working in parallel in a specific task. The great majority of studies regarding RL in the brain, however, focus on the model-free methods, based on the correlations of brain activity (specifically dopamine neurons) with reward prediction errors, found in temporal difference (TD) algorithms [8, 13, 18]. Quantitatively, one study [4] has conducted a trial-by-trial regression analysis of dopamine responses in a task with varying reward magnitudes that showed that the response dependence on the magnitude history has the same form as that expected from TD learning.

Policy and agent types Given value estimates for all action candidates, a policy choice has to be made. To avoid missing high-valued states and actions that were not yet explored, some randomness is included in the policy choice step, avoiding pure greedy policies (always choosing the action with the highest value). In RL, this is often accomplished by a 'softmax' decision rule [34] that chooses randomly but with a bias towards the highest valued estimate; in behavioural psychology, the 'matching law' achieves a similar effect. The softmax policy has been shown to fit the behaviour experimental data, even better than the ubiquitous matching law [27]. With regard to the agent types, a study with primates [15] seems to support a TD method called SARSA, while a study with rodents [25] supports a second kind called Q-Learning, both described in chapter 2.

Neural mapping The action choice and therefore the behaviour of a RL agent depends on the value of different states. There has been some discussion on where these values are stored

and processed in the brain. Given the hypothesis [29] that the reward prediction signal is highly correlated with the activity of dopamine neurons, then one of the possible brain regions is the striatum [32], associated with habitual tasks. This is consistent to the fMRI experiments that correlate the prediction errors with blood-oxygenation level dependent (BOLD) in the striatum [19]. The cortex has been presented as another brain area associated with RL [19, 20, 37, 11]. It was also hypothesized that these different brain regions could account for different RL methods, particularly having the cortex associated with model-based RL [38] and the striatum with model-free [6, 5]. Another possible organization structure is suggested by RL methods that subdivide choice into parallel prediction and decision subtasks, known as the Actor-Critic methods, that have been shown to be consistent with the experimental data [26].

1.2.2 Delay discounting

Inside the framework of instrumental conditioning, we can say that animals learn which actions should be taken in order to achieve a goal. In reinforcement learning, this is formulated as finding the optimal policy to achieve some goal. Various goals are possible, but the most common one is maximizing the expected sum of rewards [33]. The question becomes more sensitive when, in some cases, a choice has to be made between receiving a smaller but immediate reward and a larger but delayed reward. In order to compare them it is natural to discount the latter, since choosing a delayed reward implies missing other more immediate rewards. Thus, to determine the most valuable long-term action, it is necessary to define the discount function. The two most common proposals for discount functions are the exponential [28] and the hyperbolic [1]. There has been a recurrent controversy in knowing which of them is used by humans. Psychological studies [23] usually report hyperbolic discount factors, while the neuroscientific community tends to use exponential discount factors [22], mainly because of the simple theoretical properties that are associated with it. One example of such property is the constant decay rate: the value decay between today and tomorrow is the same as the one between 10 and 11 days from now; the same doesn't happen in hyperbolic discount, which explains why the former is also known as "rational". The hyperbolic discounting function can be defined as $Value = \frac{A}{(1+kD)}$ and the exponential as $Value = Ae^{-kD} = A\gamma^D$, where $Value$ is the value of the delayed reinforcer, A is the amount of the reinforcer, D is the delay of the reinforcer and k is a parameter that defines the steepness of the discount curve. Higher values of k imply a more rapid discounting. In this project we will be using the exponential (or geometric) discount factor.

In humans, most measures of the discount factor are done via questionnaires, in which the subjects are asked to choose between different combinations of small and immediate rewards and large rewards given in the future (days, weeks, months, and even years), after being asked to think about the consequences of each choice [10]. However, this kind of experiments doesn't expose the subject to the actual delay, which is a serious limitation. There is also evidence that these questionnaire-based tests may not be robust to moment-to-moment changes in the discount factor [17]. Finally, a psychological review [17] has shown that the results in questionnaire-based methods are less sensitive than operant procedures. To the best of our knowledge, there are two procedural tests to detect the discount factor in humans [22, 30]. However, none of the tests are applicable to our project, because they assume that the discount factor is the same for every task. In these tests, the task is divided in episodes, and in each episode the task structure changes, by changing the delay that the subject experiences to get the larger reward. This choice has been done to require constant relearning of the best strategy. However, in doing this, the authors didn't account for the possibility that the difference in the structure of the task could influence the discount factor that is being used. Therefore, since we want to test this possibility, we will use the set of experiments devised in our lab by Murthy [16], described in Chapter 3.

In terms of neuroscientific studies regarding delay discounting, it should be noted that studies confirm that dopaminergic prediction errors show the expected effects of discounting [25]. The same authors have also found that the encoding of the time-discounted rewards is independent of the encoding of the absolute value of the reward [24]. Finally, using the procedural experiment referred previously, a study has shown that serotonin has a role in defining the discount factor, as

a dietary reduction of serotonin levels caused an increase in impulsivity [31].

1.3 Report outline

The remaining of this report is divided into four sections. We begin by giving, in Chapter 2, the background in the areas of Reinforcement Learning and Statistics needed to understand the methods and results of this project. We then proceed to the description of the experiments and the statistical methods to extract the value of the discount factor, in Chapter 3. In Chapter 4 we present the simulation and experimental results of the methods described in the previous section. We then discuss the implications and limitations of these results, together with a brief insight on the possible future steps, in Chapter 5. At the end of the report, in the Appendices section, we present some extra materials that are needed in order to understand the full extent of this project.

Chapter 2

Methodological Background

2.1 Reinforcement Learning

Reinforcement learning is a particular area of Machine Learning that differs substantially from other subsets of Machine Learning. In Reinforcement Learning, the agent learns by interacting with the environment, by trial and error, to achieve a goal or more precisely, to maximize a reward. Intuitively, the answer that we obtain through RL is how to behave optimally given a current state of an environment, In most problems the reward isn't immediate, so we must take into account that an optimal action can have a reward later on only. Trial-and-error and delayed rewards are the most important features of reinforcement learning. The reinforcement learning agent then should then have a goal, a way to sense the state of the environment and a way to affect its state. In order to do so, the agent must explore new states and actions to discover better strategies while exploiting its current knowledge of the best actions to maximize the reward. This balance between exploration and exploitation is also characteristic of RL problems. The four main sub-elements of a Reinforcement Learning system are [33]:

- a policy. The policy determines what action to take from each state. The aim of Reinforcement Learning is to find the policy that has the highest expected reward.
- a reward function. This is used to compare different actions from different states. Each state-action pair is associated with a reward and is independent of the agent (rather it is a property of the model). This gives the immediate reward of an action.
- a value function. This is what agents use to find their policy. The value function gives the expected total reward of taking a certain action from a certain state and then continuing (a long-term reward, rather than instantaneous).
- a model. The model contains transition functions (likelihood of going from one state to another by taking a certain action), reward functions, a list of states, and the actions available from each state.

This description of RL shows clearly why Markov Decision Processes (MDP) are the most common way of describing a RL problem. A MDP is a control problem that can be described by a set of possible states S, a set of actions A, a transition function t where $t(s, a, s') = p(s'|s, a)$ (for $s, s' \in S$ and $a \in A$), a reward function r where $r(s, a, s')$ is the reward for moving from$s$ to s' under a and a policy π where $\pi(s, a) = p(a|s, \pi)$.

For a particular MDP, choosing the best course of action (and therefore the best policy) depends on having a way to evaluate performance. The first step in order to arrive at such performance measures is determining the return of a trace (which is a sequence of states, actions and rewards starting in a starting state and ending in an absorbing one). We define the return of trace τ as $R(\tau) = \sum_{k=0}^{N} \gamma_k r_{k+1}$, where N is the total number of rewards (or states or actions). The parameter γ is the so-called discount factor, which weighs the importance of rewards that come at later steps. The most common discount factor is the geometric, where $\gamma_k = \gamma^k$. With this definition, we can

define the value V of a particular state as the expected return of a policy π given a state s, given by $V^\pi(s) = E(\sum_{k=0}^{\infty} \gamma_k r_{t+k+1} | \pi, s_t = s)$ and the Q-function which is the expected return of a given state-action pair under policy π, given by $Q^\pi(s,a) = E(\sum_{k=0}^{\infty} \gamma_k r_{t+k+1} | \pi, s_t = s, a_t = a)$. Since the purpose of RL algorithms is to find the best policy, the way the value of each state (or state-action) is calculated is one of the main characteristics of the algorithms. RL algorithms can be divided into three main categories [39]:

1. Dynamic Programming: we can calculate the value of a particular state by knowing the values of the possible next states, according to the Bellman equation

$$\hat{V}'(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} t(s,a,s')(r(s,a,s') + \gamma \hat{V}(s'))$$

   Knowing *a priori* the model of the problem, we can approximate the true value function by iterating through each state and computing the Bellman equation. This approach has the problem of requiring knowledge of the transition probabilities and of the reward function, which are in many cases unrealistic assumptions.

2. Monte-Carlo: Instead of using an internal model of the MDP to calculate the values, we can sample through experience the rewards obtained starting from a particular state and then average them to obtain the value of a particular step. This approach has the advantage of not requiring access to the internal model of the MDP. However, two problems arise in this approach: the first is guaranteeing that every possible state and action is visited while striving for the highest rewards; the second is that we can't learn during an episode, only at the end, when the total reward is obtained. To solve the former, non-greedy policies may be used, such as ϵ-greedy policies, which ensure that the action that leads to a highest reward (according to current estimates) is chosen with a high proabability $(1-\epsilon)$, while least valued actions are still chosen with a small probability ϵ. The way to proceed is then doing an estimation of the values of each state-action followed by the improvement of the policy (for example by recomputing the ϵ-greedy policy with the new estimates of each state-action values). Repeating this process leads to an approximation of the optimal policy. If the policy that is being followed to choose actions and the policy that is being improved are the same, we call the method on-policy. If we follow a different policy than the one we are improving, we call the method off-policy. Another important distinction is batch optimization vs. online learning. In batch optimization, each time the policy is improved we reset the values of each state. In online learning, the previous estimates are updated according to the difference between the current estimate of the return and the previous ones.

3. Temporal-difference (TD) [34]: we can think of a method which uses bootstrapping to update estimates based on other estimates and sampling, allowing us to work without knowledge of the model. Temporal-Difference is one such method. In a TD method, the update is made after each time step, according to $\hat{V}'(s) \leftarrow \hat{V}(s_t) + \alpha[r_{t+1} + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)]$. Two of the most used TD algorithms are SARSA and Q-Learning, where SARSA is an on-policy method and Q-learning is an off-policy one. These will be explained in more detail later on.

The higher efficiency of Temporal Difference Learning over Monte Carlo Methods has been noted in multiple papers, and proved in depth [33]. Much of the current work in Reinforcement Learning is based on Temporal Difference Learning.

Another important distinction has to be made. A model-free method ignores the structure of the problem and aims to learn the best action for every state, or to learn the best state-action pairs. The model-based methods, on the other hand, try to capture the transition probabilities between different states as well as the rewards, using that information to calculate the value of each state and action (instead of calculating them directly).

Finally, we make the distinction between stationary and non-stationary problems. In stationary problems, the properties and dynamics of the environment do not change with time; in non-stationary problems, some of the properties may change. This is an important distinction because

some methods stabilize after some time. If the learning rate decreases with time, then changes that happen after the initial period will not be detected and the system will not learn them. In this project we will be dealing with non-stationary problems, so our methods will always be robust to non-stationary problems.

Temporal difference methods, which are the main focus of this work, commonly include two very import parameters. The first one is the memory of the learner α, i.e. how important are previous estimates of the value of a given state-action pair relatively to new estimates. The second is the discount factor γ, i.e. the weight of future rewards relatively to immediate rewards. Despite the indisputable relevance of both parameters, this project aims to quantify the reward discount factor, which means we are interested in knowing how are future rewards valued compared to the valuation of immediate rewards.

Since our work is based on temporal-difference methods, we describe them in more detail in the next section as well as the equations that govern each of the particular methods.

2.1.1 Temporal-difference methods

Temporal Difference learning is a combination of Monte Carlo (MC) ideas and dynamic programming ideas. TD methods update their estimates based on previously learned estimates of the state-action pairs. Given some experience follow a policy π, both TD and MC methods update the value estimates for the states that occurred in that experience. A MC method would wait until the end of an episode to update all the estimates of the states that occurred in that episode. This can be problematic if, for instance, the problem isn't divided in episodes, as is the case in this project. This may be also problematic due to the fact that no learning is happening during a trace, which may delay improvements in the performance of the learner. In TD, learning happens after every step, which solves these problems. At time $t+1$, a TD method has an estimate of the value of the state, so it can update its previous estimate according to the following equation

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2.1}$$

Intuitively this can be understood in the following way: $V(S_t)$ is the previous estimate of the value of S_t. After executing an action, the agent receives a reward of R_{t+1} and transitions to state S_{t+1}. Thus, $R_{t+1} + \gamma V(S_{t+1})$ is the new estimate of the value of state S_t, since it was the value that the agent received when starting from that state. $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is the difference between the new estimate and the previous one, which is usually called a prediction error, because it reflects by how much the previous estimate (the prediction) is different from the actual value experienced. This error, multiplied by a learning rate α, is added to the previous estimate to correct it. The higher the learning rate the more weight is given to more recent experience. The limit value of $\alpha = 1$ means that only the last estimate is used as the estimate of the value of the state.

Instead of learning the state values, we can learn the state-action pair values, and that leads to an updated version of eq. 2.1

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

The next few paragraphs are dedicated to specific implementations of this general rule, specifically those which are used in this project.

SARSA SARSA is an on-policy method that derives its name from the fact that it uses the quintuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$. As all on-policy methods, we continually estimate the value of the policy π, q_π, while updating π greedily based on q_π. The update process can be described as follows:

1. Choose an action A from S using the defined policy (e.g. ϵ-greedy), derived from Q.
2. Observe R and S'.
3. Choose A' from S' using the defined policy.

4. $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
5. Repeat.

Q-Learning Q-Learning is an off-policy TD method that directly approximates the optimal action-value function independently of the policy being followed. The behavior policy still influences what state-actions pairs are visited. In this case, the update is not made with respect to the chosen action A', but instead to the maximum of the Q values for that state. The update process can be described as follows:

1. Choose an action A from S using the defined policy (e.g. ϵ-greedy), derived from Q.
2. Observe R and S'.
3. $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \arg\max_a Q(S',a) - Q(S,A)]$
4. Repeat.

2.1.2 Other methods

Model based In this method, instead of keeping a state-action value table, we learn separately the transition function and the state value function. The transition function $t(s,a,s')$ can be calculated as the average of the number of times that we transitioned from state s to state s' via action a. The reward function $r(s,a,s')$ is simply the average of the rewards obtained when transitioning from state s to state s' via action a. We can then use dynamic programming to estimate the V table (or the Q table, as that is useful to build the hybrid agent).

1. Choose an action A from S using the defined policy (e.g. ϵ-greedy), derived from Q .
2. Observe R and S'.
3. Update $t(s,a,s')$ and $r(s,a,s')$
4. Recalculate V (or Q) using dynamic programming
(repeatedly running $V(s) \leftarrow \sum_{s'} t(s,a,s')(r(s,a,s') + \gamma V(s'))$ until convergence)
5. Repeat.

Hybrid Another possible method is to use a combination of a TD method with a model based method. This can be simply done by evaluating the $Q(s,a)$ table using both methods and using a weighted average of them, according to a mixing parameter. Having the state-action values then the problem is reduced to action selection, which will be discussed in the next section.

2.1.3 Action selection

In order to find a solution to RL problems, we still need a way to convert a state-action value function into a policy. This is called the action selection problem. We will describe two solutions, ϵ-greedy and softmax. If we have N actions available, then the policy $\pi(s,a)$ is defined, for an ϵ-greedy policy, as

$$\pi(s,a) = \begin{cases} 1-\epsilon & if a = \arg\max_{a'} Q(s,a') \\ \frac{\epsilon}{N-1} & otherwise \end{cases}$$

and for a softmax policy as

$$\pi(s,a) = \frac{e^{\frac{Q^\pi(s,a)}{\tau}}}{\sum_{j=1}^{N} e^{\frac{Q^\pi(s,a^j)}{\tau}}}$$

where τ is the Gibbs temperature, in allusion to chemical kinetics. For $\tau \to \infty$, all the actions have the same probability while for $\tau \to 0$ the probability of the dominant action approaches 1.

2.2 Statistics

In this section we will present the basic statistic concepts that are essential to the understanding of this report. This includes a basic summary of hypothesis testing and model comparison methods.

2.2.1 Hypothesis testing

Hypothesis testing can be defined as a method for testing a hypothesis about a parameter given a measured sample [9]. This involves determining the likelihood that a sample statistic could have been selected if the hypothesis were true. The process of hypothesis testing can be described as:

1. Identification of the hypothesis to be tested (null hypothesis).
2. Selection of the test statistic and the significance level, i.e. the criterion upon which we decide that the hypothesis being tested is true or not.
3. Compute the test statistic.
4. Compare the test statistic using the obtained data to what it should be if the null hypothesis were true and make a decision. If the probability of observing that data if the null hypothesis were true is less than a given significance level, then we reject the null hypothesis.

In this project we only use the two-sample Student's t-test, thus deserving a description in this section.

Student's t-test A t-test is a hypothesis test in which the test statistic follows Student's t distribution. We will use this test in two different scenarios: the first to test whether the mean of two samples is significantly different; the second is to test whether a sample rejects the hypothesis of the slope of the linear regression being a certain value.

Student's two-sample paired t-test is usually used (and specifically in this project) to compare the means of two samples. For this test, the null hypothesis is that the means of both samples are equal. The test statistic t_s is [9]

$$t_s = \bar{x}_1 - \bar{x}_2 \sqrt{\frac{n(n-1)}{\sum(x_i - y_i)}} \sim \mathcal{T}_{\mathrm{n}-1}$$

For this test we assume that the paired differences between the two samples are independent and identically distributed. A table of Student's t-distribution confidence intervals can be used to determine the significance level at which two distributions differ.

If we want to test the value of the slope of a fitted model, we can also use the t-test. If we are fitting the model $Y_i = \alpha + \beta x_i + \epsilon_i$ where x_i, Y_i $(i = 1...n)$ are the data points and ϵ_i are independent and identically normally distributed with a mean of 0 and unknown variance, we can test the hypothesis that the slope $\beta = \beta_0$. Then, the test statistic is [9]

$$t_s = \frac{(\hat{\beta} - \beta_0)\sqrt{n-2}}{\sqrt{SSR / \sum_{i=1}^{n}(x_i - \bar{x})^2}} \sim \mathcal{T}_{\mathrm{n}-2}$$

where $SSR = \sum_{i=1}^{n} \hat{\epsilon_i}^2$. Again, a table of Student's t-distribution confidence intervals can be used to determine the the significance level at which two distributions differ.

2.2.2 Model comparison

It is clear that selecting the model that has the highest likelihood in a given set of data is not the best indicator to the quality of the model, primarily due to the problem of overfitting. We desire to have a model as simple as possible that describes the data well enough. One way to deal with this problem is to use information theory, namely using Akaike Information Criterion (AIC)[2].

AIC solves the trade-off between goodness of fit and the complexity of the model by using information theory to give an estimate of the amount of information loss when representing the data using a given model. It should be noted that AIC is a relative criterion, only serving to compare different methods, not giving an estimate of how well the model fits the data. The AIC is given by

$$AIC = 2k - 2\log(L)$$

where k is the number of free parameters and L is the maximized likelihood for the model. A free parameter can be defined as a numerical value in a model that can be adjusted to better fit the data [9]. The best model is the one that has the lowest AIC. Note that having a high likelihood decreases the AIC as expected and having a high number of parameters increases it. Therefore, the term $2k$ serves the purpose of penalizing complex models and therefore overfitting.

Chapter 3

Methods

The global objective of this project is to create a set of experiments and analysis tools to quantify one of the parameters of human learning, the discount factor. This work builds upon a set of experiments previously designed and run by Keshava Murthy [16]. These experiments consist of games devised to enable the extraction of the discount factor the subjects used to solve them, based on the decisions of the subjects. However, a subset of the previously run experiments proved to be ineffective, as well as the proposed analysis. Therefore, we present two main contributions: the first being a new set of experiments that successfully allows the recovery of the discount factor; secondly, we propose a new top-down method to extract the discount factor, together with a set of bottom-up methods to the same purpose.

3.1 The original experiment design

3.1.1 General description

The experiments are composed of games, i.e. models that consist of states and rewards with which the subject interacts via a defined set of possible actions. The games are played on a computer, using a keyboard to record the player's actions and the screen to give the player information regarding the current state and the obtained reward. In all games the main goal is to maximize the obtained reward throughout the whole duration of the game. The structure of the game can be summarised as follows: the subject is shown in what state he is by displaying a texture picture on the screen; the subject chooses one action that triggers a transition to a new state and the delivery of a reward (with the new state and the obtained reward being displayed on the screen). One of these steps is depicted in the figure 3.1. These two events are repeated for a determined number of steps before the game ends. The implementation and design choices of the games are described in Murthy [16].

In all games there are two main possible behaviours. One of them gives small immediate rewards while the other gives larger delayed rewards. The value of the delayed reward that makes both choices equally valuable depends on the discount factor that each subject is using. Thus, by varying the value of the delayed reward, we will observe switches in the strategy that is being followed. Each switch happens at a particular value of the variable reward, which corresponds to an implied discount factor, that can then be extracted.

Two types of experiments were designed: one in which the reward has a deterministic integer value and one in which there were only binary rewards but the probability of obtaining them varied throughout the game. The reasoning behind this choice was to account for the fact that people may not regard the number '2' as being twice as valuable as '1' or may not have the ability to distinguish 70 values of reward. Using a binary reward would avoid this problem. However, distinguishing two different probabilities of reward is more difficult than distinguishing two numbers. We will see that it was this problem that rendered the binary experiments as originally designed useless. The four games, two with real rewards and two with binary rewards, are presented in figure 3.2. Note that games 1 and 2 share the same structure while the reward differs, with the first having a real-valued reward and the second a binary one. The same happens with games 3 and 4. In all games the meaningful choice is made in state 1, which will be referred to as the choice state. Action 1 in

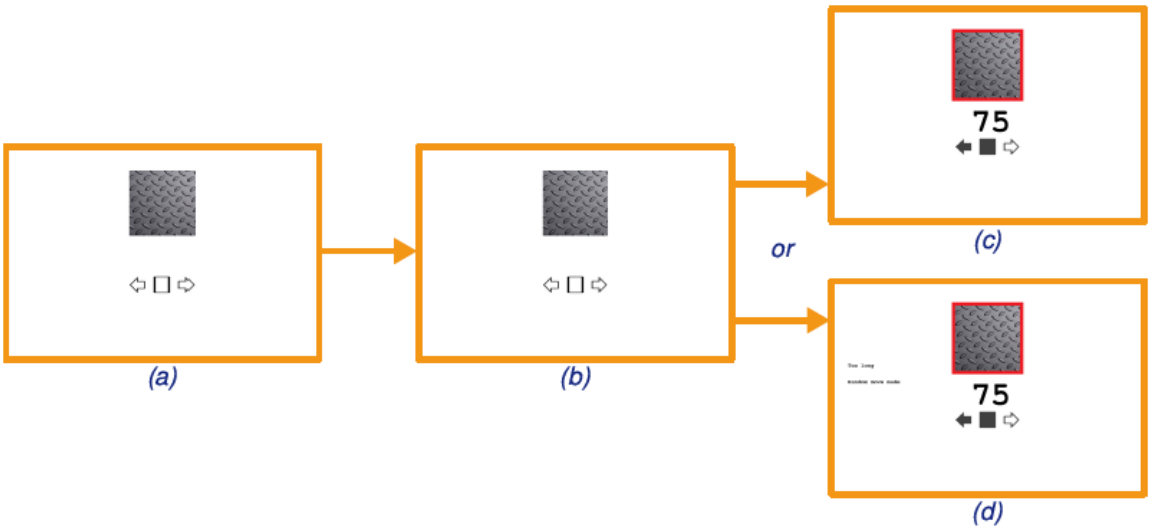


Figure 3.1: The experimental setup. The subject is (a) presented the current state (b) waits for a certain amount of time (c) makes a choice in the allotted time (d) fails to do a choice in the allotted time. After either (c) or (d) a new state is presented (a) after some time.

state 1 corresponds to choosing the long path, while action 2 in state 1 corresponds to choosing the short path. The choice in all other states is trivial, as one of them always gives a reward greater than the other (a fact that can be used to ensure the subject has learnt the game). For example, in game 1 the best choice in state 2 is always action 1, as $x > 25$. We call the non-optimal choice the dummy path. In each game there is also a reward state and a reward action. The reward state is the state in which the subject may obtain the variable reward (or the binary reward with variable probability) by performing the reward action. This is the only property that changes throughout the game. The length of games 1 and 2 is 300 steps, while for games 3 and 4 it is 350 steps.

To help the understanding of this structure of games, we shall describe game 1 as an example. The subject starts the game in state 1. From there, if he selects action 1 he will transition to state 2, receiving a reward of $r = 0$. If he selects action 2, he either transitions to State 1 (with a probability of 80%) while receiving a reward of $r = 25$ or he transitions to State 2 (with a probability of 20%) while receiving a reward of $r = 0$. From state 2 the subject will transition to state 1 with either action, but action 1 results in receiving a reward of $r = x$ and action 2 gives a reward of $r = 25$.

The variation of the variable reward value or probability is presented in figure 3.3. Note that for games 1 and 3 the variable reward ranges from 40 to 160 and for games 2 and 4 the probability of reward ranges from 0.2 to 1.

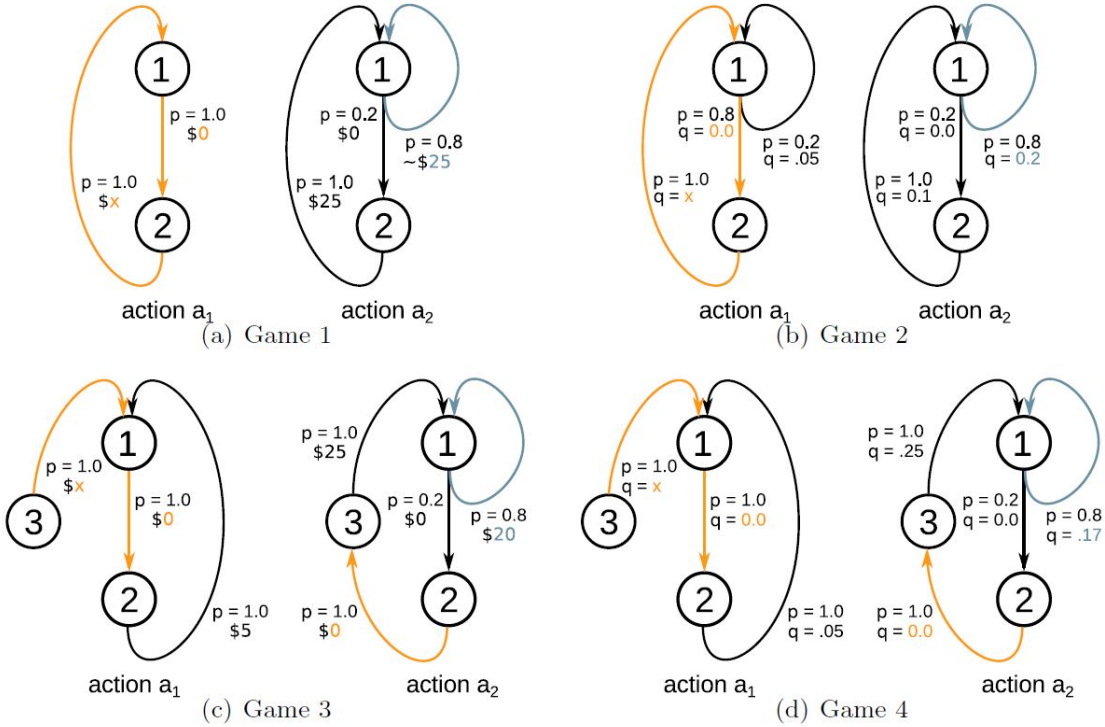
3.1.2 Solving the games

As discussed before, the meaningful action is always chosen in state 1. This means that, in order to find the discount factor corresponding to a given switching value (or vice-versa) we need to find the values of these parameters that make both actions in state 1 equally valuable. Thus, for each game, we will compute each state-action pair value in state 1 and will determine the values that make them equal. We will solve the first game in detail and then a summary of the solution of the other games. For the sake of simplicity, let us denote $\pi(s^2, a^1)$ by π^2 and $\pi(s^3, a^1)$ by π^3.

Game 1

$$Q^\pi(s^1, a^1) = \gamma V^\pi(s^2)$$

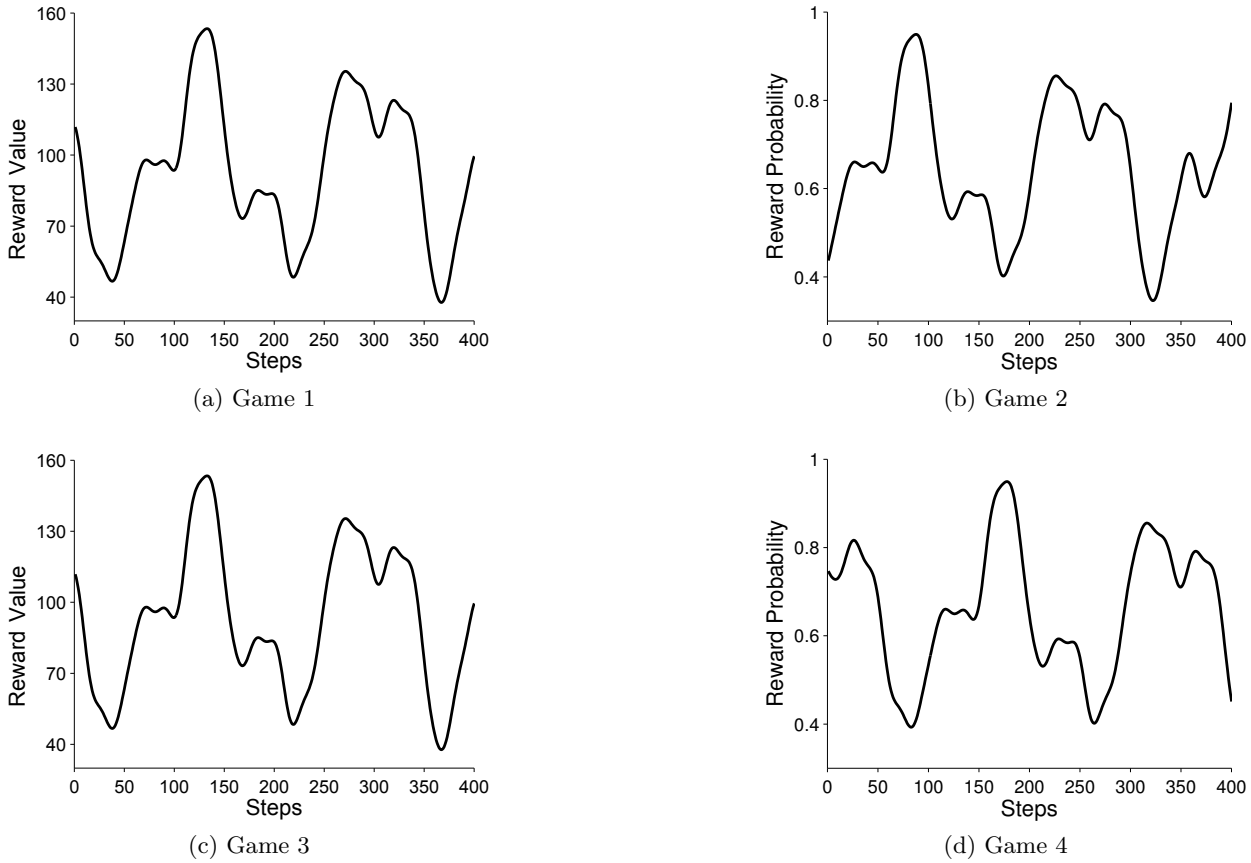
$$Q^\pi(s^1, a^2) = 0.8[25 + \gamma V^\pi(s^1)] + 0.2\gamma V^\pi(s^2)$$



(a) Game 1 (b) Game 2

(c) Game 3 (d) Game 4

Figure 3.2: The transition games described as Markov Decision Processes. The circles represent states and for each game the left (right) picture represents the outcomes of performing action 1 (2). p is the probability of transition while q represents the probability of obtaining a reward in the binary games



(a) Game 1 (b) Game 2

(c) Game 3 (d) Game 4

Figure 3.3: The reward functions of the four games

$$V^\pi(s^2) = \pi^2[x + \gamma V^\pi(s^1)] + [1 - \pi^2][25 + \gamma V^\pi(s^1)]$$

Assuming that the dominant action in s^1 is a^1, then the long path is preferred. We call this the greedy policy π_{g1}, where $V^{\pi_{g1}} = Q^\pi(s^1, a^1)$, simplifying the above equations to

$$V^\pi(s^1) = \gamma V^\pi(s^2) = \gamma\{\pi^2[x + \gamma V^\pi(s^1)] + [1 - \pi^2][25 + \gamma V^\pi(s^1)]\} = \gamma[25 + \pi^2[x - 25]]$$

On the other hand, if we assume that the dominant action in s^1 is a^2, then the short path is preferred. We call this the greedy policy π_{g2}, where $V^{\pi_{g2}} = Q^\pi(s^1, a^2)$, simplifying the above equations to

$$V^\pi(s^1) = 0.8[25 + \gamma V^\pi(s^1)] + 0.2\gamma V^\pi(s^2) = \frac{25\gamma - 25\gamma\pi^2 + \gamma\pi^2 x + 100}{\gamma^2 + 4\gamma - 5}$$

Both paths will be equally valuable when $V^{\pi_{g1}} = V^{\pi_{g2}}$which, for a given γ corresponds to

$$x = \frac{25\gamma\pi^2 + 25}{\gamma\pi^2}$$

Or, for a given switching value x,

$$\gamma = \frac{-25}{25\pi^2 - \pi^2 x}$$

This means that we can infer the discount factor used by the subject if we know the value of x for which both actions are equally valuable (which means that the subject will switch policies when x crosses that value) and the probability of choosing the best path in the non-choice state.

Game 2

$$Q^\pi(s^1, a^1) = 0.01 + 0.2\gamma V^\pi(s^1) + 0.8\gamma V^\pi(s^2)$$

$$Q^\pi(s^1, a^2) = 0.16 + 0.8\gamma V^\pi(s^1) + 0.2\gamma V^\pi(s^2)$$

$$V^\pi(s^2) = \pi^2[x + \gamma V^\pi(s^1)] + [1 - \pi^2][0.1 + \gamma V^\pi(s^1)]$$

Both paths will be equally valuable when $V^{\pi^1} = V^{\pi^2}$which, for a given γ corresponds to

$$x = \frac{11\gamma + 10\gamma\pi^2 + 25}{100\gamma\pi^2}$$

Or, for a given switching value x,

$$\gamma = \frac{-25}{10\pi^2 - 100\pi^2 x + 11}$$

Game 3

$$Q^\pi(s^1, a^1) = \gamma V^\pi(s^2)$$

$$Q^\pi(s^1, a^2) = 16 + 0.8\gamma V^\pi(s^1) + 0.2\gamma V^\pi(s^2)$$

$$V^\pi(s^2) = \pi^2[5 + \gamma V^\pi(s^1)] + [1 - \pi^2][\gamma V^\pi(s^3)]$$

$$V^\pi(s^3) = \pi^3[x + \gamma V^\pi(s^1)] + [1 - \pi^3][25 + \gamma V^\pi(s^1)]$$

Both paths will be equally valuable when $V^{\pi^1} = V^{\pi^2}$which, for a given γ corresponds to

$$x = \frac{20\gamma - 5\gamma\pi^2 + 5\gamma^2\pi^2 + 25\gamma^2\pi^3 - 5\gamma^2 - 25\gamma^2\pi^2\pi^3 + 20}{\gamma^2\pi^3 - \gamma^2\pi^2\pi^3}$$

Or, for a given switching value x,

$$\gamma = -\frac{5\pi^2 - 20 + \sqrt{5(400\pi^2\pi^3 - 400\pi^3 - 120\pi^2 + 16\pi^3 x + 5\pi^2 - 16\pi^2\pi^3 x}}{10\pi^2 + 50\pi^3 - 50\pi^2\pi^3 - 2\pi^3 x - 10}$$

Game 4

$$Q^\pi(s^1, a^1) = \gamma V^\pi(s^2)$$

$$Q^\pi(s^1, a^2) = 0.136 + 0.8\gamma V^\pi(s^1) + 0.2\gamma V^\pi(s^2)$$

$$V^\pi(s^2) = \pi^2[0.05 + \gamma V^\pi(s^1)] + [1 - \pi^2][\gamma V^\pi(s^3)]$$

$$V^\pi(s^3) = \pi^3[x + \gamma V^\pi(s^1)] + [1 - \pi^3][0.25 + \gamma V^\pi(s^1)]$$

Both paths will be equally valuable when $V^{\pi^1} = V^{\pi^2}$which, for a given γ corresponds to

$$x = \frac{17\gamma - 5\gamma\pi^2 + 8\gamma^2\pi^2 + 25\gamma^2\pi^3 - 8\gamma^2 - 25\gamma^2\pi^2\pi^3 + 17}{100\gamma^2\pi^3 - 100\gamma^2\pi^2\pi^3}$$

Or, for a given switching value x,

$$\gamma = -\frac{-5\pi^2 + 17 + \sqrt{1700\pi^2\pi^3 - 1700\pi^3 - 714\pi^2 + 6800\pi^3 x + 25\pi^2 - 6800\pi^2\pi^3 x + 833}}{106\pi^2 + 50\pi^3 - 50\pi^2\pi^3 - 100\pi^3 x + 100\pi^2\pi^3 x - 8}$$

3.1.3 Shortcomings of the binary games

In the deterministic variable reward games you only have to sample the reward function (by choosing the correct action at the right state) once to know its current value. With the binary games this is not the case because the reward is always the same, what varies is the probability of obtaining it. With only one sample (1 or 0) we have very little information, but if we had 10 samples and received a reward in 5 of them, we are more certain that the true value of the reward probability should be around 0.5. The reward action has to be performed multiple times to determine (with a certain level of confidence), what is the current probability of obtaining the reward. The subject has to be able to, at least, to distinguish the current value of the reward probability from the true switching value, i.e. the value that turns both policies (short and long) equally valuable. Thus, it is desirable to have the true switching value near 0.5, since this is the value that is most easilly distinguished from all the other values. This can be easily understood with an example. Suppose the true switching value is 0.8 and that the current reward probability is 0.95. To determine that the long path is more valuable than the short, the subject has to be able to distinguish a probability of 0.95 from 0.8 which, for the same level of confidence, requires a much higher number of visits than distinguishing 0.95 from 0.5.

Here arises the first problem. Using the equations in section 3.1.2, we can obtain the true switching values for different values of the discount factor (lower discount factors correspond to higher probabilities of reward for both paths to have the same value). According to the previous work done by Murthy [16], we expect the discount factor to lie within the 0.5-0.9 range, so we should be able to, at least recover those values. However, as you can see in table 3.1, particularly for game 4, these values are extremely high. These values are calculated for the best possible case, where the subject learns perfectly the best strategy for non-choice states (i.e state 2 for games 1 and 2 and states 2,3 for games 3 and 4). We can see that it is impossible to measure a $\gamma = 0.5$ in game 1, because that would require the probability of reward to be 1.19, which is an impossibility. And even to measure a $\gamma = 0.7$ we would need the switching value to lie around 0.8. Not only does

Discount factor (γ)	Switching value
0.5	0.71
0.6	0.63
0.7	0.58
0.8	0.53
0.9	0.49

(a) Game 2

Discount factor (γ)	Switching value
0.5	1.19
0.6	0.93
0.7	0.76
0.8	0.65
0.9	0.57

(b) Game 4

Table 3.1: The true switching value for different discount factors in games 2 and 4. This corresponds to the value of the reward probability that makes both paths equally valuable for a subject with the given gamma. Note that these values are extremely high, in some cases even going beyond 1, which is impossible.

the reward function cross these values few times but also, as we show below, the number of visits required to prove that the current reward probability is greater than 0.8 is very high. We now derive a formal way to calculate how many visits are needed to distinguish between these values of probability. This informs us of the minimum duration for which the reward values should remain constant to allow the subjects to measure it with enough confidence.

Each time the reward action is chosen a Bernoulli trial is performed, with a probability of success $p = x$, where x is the current probability of obtaining a reward. Therefore, a sequence Y of N reward actions is binomially distributed, with $Y \sim Binomial(N, p)$. The goal here is always to distinguish a given probability from the true switching probability that is defined for a given γ. For the sake of simplicity, we'll show the reasoning for calculating the number of visits needed to show that a probability is greater than the threshold, but please note that the procedure is totally symmetric to show that it is lower.

We are trying to reject the null hypothesis that x (the current reward probability) is less than or equal to the threshold t ($H_0 : x <= t \iff x = t$, because the latter is more extreme). We're trying to find the minimum number of successful trials k that, in N trials, would reject the null with confidence of $1 - \epsilon$, i.e.

$$Pr(s > k|N, t) < \epsilon \bigwedge Pr(s > (k-1)|N, t) > \epsilon$$

which can be found using the inverse binomial distribution or by looking up at the distribution tables. Intuitively this means that, in an N length experiment, if we obtained more than k successes then we would reject, with $1-\epsilon$ confidence, the possibility that the probability that generated that trace is less than or equal to t.

The next step, then, is finding the probability of obtaining at least that number k of successes given that the actual probability is x. We want that probability to be as high as possible, but we'll set our minimum to 0.8 which means that, if the true probability is x, then we'll obtain at least the required number of successes k to be certain that $x > t$ in 80% of the cases (given that with $k - 1$ we would not be certain). In the remaining 20% of the cases, we would accept the null hypothesis even though it was false. We'll also assume that we would need, at least, to be 60% certain before making a switch (which is a light assumption). These results are shown in table 3.2. Note that the minimum number of visits is to, at this certainty, distinguish two of this values is 4. And while the maximum we obtained was 12, smaller intervals require more visits. For example, a distinction between 0.58 and 0.53 (which corresponds to $\gamma = 0.7$ and $\gamma = 0.8$ in game 2), requires 25 visits. If we look at the reward function for game 2, we can see that these values vary quickly. In fact, in four steps the reward function can vary as much as 0.09, which can correspond to a totally different discount factor. Therefore, this reward function isn't proper to a binary game since it doesn't have constant values for a sufficiently high number of steps that allow the subject to sample it accurately.

In order to have a working binary game we would need to fulfill the following conditions:

		Threshold value γ				
		0.4	0.5	0.6	0.7	0.8
Current value x	0.5	5	-	-	-	-
	0.6	4	4	-	-	-
	0.7	4	4	8	-	-
	0.8	4	4	4	8	-
	0.9	4	4	4	4	12

Table 3.2: Required number of reward actions in order to successfully detect that the current probability x is greater than the given threshold value γ in 80% of the cases with 60% certainty.

1. For the expected values of gamma, the corresponding switching values should be approximately 0.5.
2. The reward function should be a step-wise function, with step-lengths long enough to allow the subject to accurately sample it.

It should be noted that, given the current set up of the games, the first condition is not possible to satisfy. Particularly for game 3, the only way to reduce the switching values is to decrease the probability of reward of the short path, since if we make the short path less valuable than it currently is then the probability of reward in the long path that makes both paths equally valuable may be lower as well. However, the probability of reward in the short path is already extremely low (0.17). Therefore, this is not an option, and a new set of games had to be created. The game had to include binary rewards (only two possible values) while also allowing the switching values to be approximately 0.5 without reducing the probability of reward on the short path. The solution we designed is presented in the next section.

3.2 New experiments

3.2.1 The games' structure

As described in the previous section, the binary experiments described in section 3.1.2 suffered from several problems that rendered them useless for the objectives of this project. We needed an average switching value around 0.5 and yet we could not lower the reward probability of the short path. This led us to a new kind of experiments, which we call hybrid. Each state-action pair may only result in one of two reward values, but this reward values need not be the same in all states-action pairs. This means that we can adjust the switching values by varying the value of reward that each state-action pair produces, not relying on the probabilities of reward alone. This makes the games easier to learn, as well, since all the transitions and rewards are deterministic except the one at the end of the long path (the reward action) and the transition associated with the short path, which is made probabilistic to ensure that some exploration is made. The resulting games are depicted in figure 3.4. For game A we have three values of reward, 0, 1, and 5. For game B these values are 0, 1, and 8.

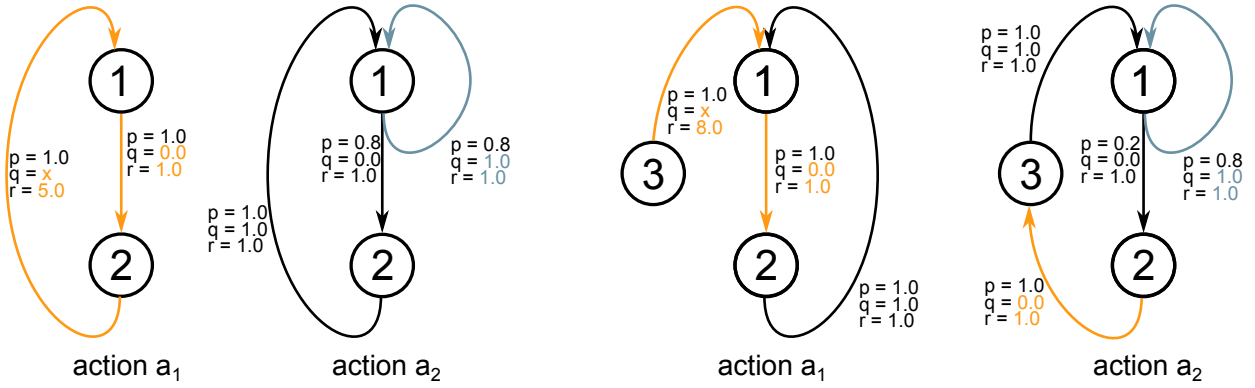
These values were chosen so as to make the switching values near 0.5, as discussed. In order to obtain the needed values to achieve this goal, the analytical solution of these games had to be derived, and is presented below.

Game A

$$Q^{\pi}(s^1, a^1) = \gamma V^{\pi}(s^2)$$

$$Q^{\pi}(s^1, a^2) = 0.8[1 + \gamma V^{\pi}(s^1)] + 0.2\gamma V^{\pi}(s^2)$$

$$V^{\pi}(s^2) = \pi^2[5x + \gamma V^{\pi}(s^1)] + [1 - \pi^2][1 + \gamma V^{\pi}(s^1)]$$



(a) Game A (b) Game B

Figure 3.4: The new hybrid transition games described as Markov Decision Processes. The circles represent states and for each game the left (right) picture represents the outcomes of performing action 1 (2). p is the probability of transition, q represents the probability of obtaining a reward and r represents the value of the reward.

Both paths will be equally valuable when $V^{\pi^1} = V^{\pi^2}$ which, for a given γ corresponds to

$$x = \frac{169\gamma + 150\gamma\pi^2 + 395}{1500\gamma\pi^2}$$

Or, for a given switching value x,

$$\gamma = \frac{-395}{150\pi^2 - 1500\pi^2 x + 169}$$

Game B

$$Q^\pi(s^1, a^1) = \gamma V^\pi(s^2)$$

$$Q^\pi(s^1, a^2) = 0.8[1 + \gamma V^\pi(s^1)] + 0.2\gamma V^\pi(s^2)$$

$$V^\pi(s^2) = \pi^2[1 + \gamma V^\pi(s^1)] + [1 - \pi^2][\gamma V^\pi(s^3)]$$

$$V^\pi(s^3) = \pi^3[8x + \gamma V^\pi(s^1)] + [1 - \pi^3][1 + \gamma V^\pi(s^1)]$$

Both paths will be equally valuable when $V^{\pi^1} = V^{\pi^2}$ which, for a given γ corresponds to

$$x = \frac{17\gamma - 5\gamma\pi^2 + 8\gamma^2\pi^2 + 25\gamma^2\pi^3 - 8\gamma^2 - 25\gamma^2\pi^2\pi^3 + 17}{100\gamma^2\pi^3 - 100\gamma^2\pi^2\pi^3}$$

Or, for a given switching value x,

$$\gamma = \frac{1 + \sqrt{4\pi^2 + 4\pi^3 + 32x - 4\pi^2\pi^3 - 32\pi^2 x - 32\pi^3 x + 32\pi^2\pi^3 x - 3}}{2\pi^2 + 2\pi^3 + 16x - 2\pi^2\pi^3 - 16\pi^2 x - 16\pi^3 x + 16\pi^2\pi^3 x - 2}$$

With these expressions it's straightforward to obtain the corresponding switching values. It should be noted that, in order for the sake of simplicity, only integers were deemed acceptable as rewards, to reduce as much as possible the effect of the non-exact perception of the value of different rewards. The table of switching points for each game is presented in table 3.3.

Discount factor (γ)	Switching value
0.5	0.73
0.6	0.65
0.7	0.59
0.8	0.54
0.9	0.49

(a) Game A

Discount factor (γ)	Switching value
0.5	0.75
0.6	0.68
0.7	0.56
0.8	0.48
0.9	0.42

(b) Game B

Table 3.3: The true switching value for different discount factors in games A and B. This corresponds to the value of the reward probability that makes both paths equally valuable for a subject with the given gamma. These new values are all feasible values of probability, which means that we can theoretically recover the whole range of γ.

3.2.2 The new reward function

The second issue previously described is the shape of the reward function, which was not appropriate for the binary structure of these games. As was described in table 3.2, we need to be able to sample at least 12 times the same reward probability in order to have a reasonable estimate of its value. However, these samples are not obtained in consecutive steps, since the long path length is greater than 1. Therefore, we need to account for the fact that in order to sample the reward function N times, we need to have a step length in the reward function of at least $2N$ ($3N$) steps in game A (B). Moreover, these steps may be interleaved with choices of the short path, so we should account for that as well. However, there has to be a compromise in this choice, since we need to have as many switches of reward probability values, as this would allow us to have a high number of samples of the value for which each subject changes his policy. The solution is to make the step length a sample from a Poisson distribution with a mean of 60. Having a non-deterministic step length has the purpose of stopping subjects from learning the length of the reward steps. At the same time, the length of the experiments is increased to a mean of 650 steps ($600 + Poisson(50)$) to compensate for the slower changes in the reward probability value. The resulting function for both games is depicted in figure 3.5. Our new set of experiments, A and B, has been performed by 11 subjects, with ages between 18 and 25 in our lab, following the same procedure as the one used by Murthy [16]. The experiments were performed in accordance with the institutional and national regulations involving human experiments. The details are explained in Appendix B.

3.3 Bottom-up approach

In this section we propose an ensemble of methods to recover the discount factor used by each subject using a bottom-up approach. As the previously described tasks are designed in such a way that they can be trivially converted to a RL problem (a set of states, each state with a set of actions and each action causing a transition to a new state, possibly resulting in a reward), we can assume that the subjects used a particular RL agent to solve it. Then, we try to compute the set of parameters that would maximize the probability of observing the sequence of states, actions and rewards actually experienced by the player. This results in reducing the problem of finding the discount factor to an optimization problem, where we want to find the maximum of the likelihood function by varying the parameters of the agent. This was implemented in MATLAB using the *fmincon* function, that finds the minimum of a function with constrained parameters. Note that this approach has as many variants as the combinations of the existing RL agents and policies, and so a complete and exhaustive list would be impossible to compile. We are going to present the general procedure of obtaining the objective function, followed by a set of examples for specific RL agents and policies.

The general idea behind this approach is to have a RL agent perform a conditioned simulation. If we were merely performing a simulation, then the agent would be able to perceive the environment

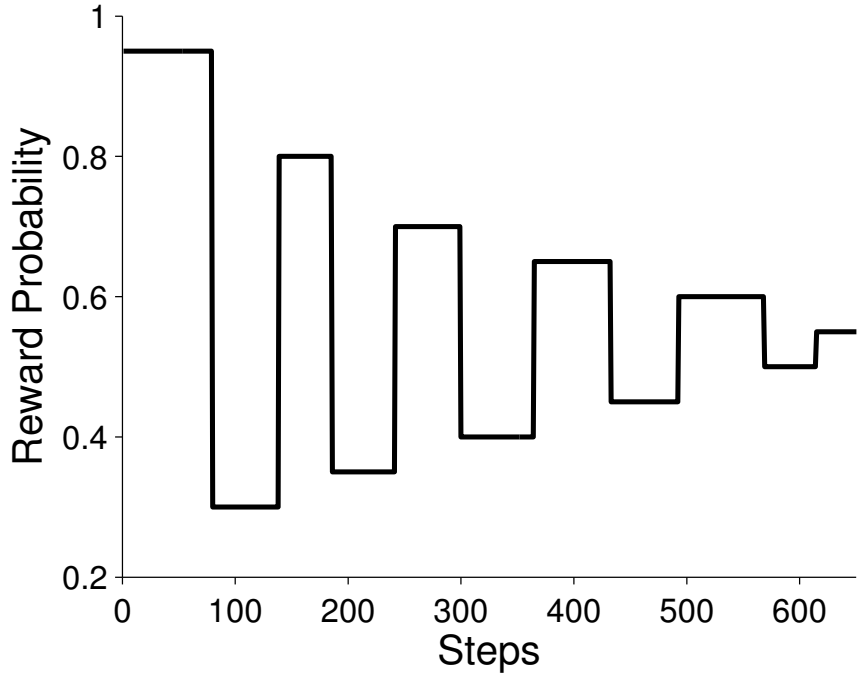


Figure 3.5: The reward function of the hybrid games A and B

(their current state), act upon it (the action) and perceive the obtained reward. Here, however, we allow the agent to perceive the current state and the obtained reward but he may not act upon it freely. Instead of asking the agent to choose an action at each time step we ask him the probability of choosing the action that was actually chosen by the real subject. The agent then, based upon the consequences that the real subject experienced (the state to which he transited and the reward he obtained), updates the probabilities of performing the actions in the next step. If this is done for every step, then the total probability of seeing that sequence of actions given the experience of the subject and the parameters of the agent is given by the product of the probability of seeing each action individually or, more formally,

$$Pr(trace|\mathbf{\Theta}) = \prod_{i=1}^{N} Pr(a_i|q_i)p(q_i|\mathbf{\Theta}) \tag{3.1}$$

where $\mathbf{\Theta}$is the set of parameters of the RL agent,q_i is the probability of choosing action 1 at step i, a_i is the action performed at time step. The general algorithm can then be summarised in three steps:

1. Compute the probability of choosing the action that was chosen given the past experience of the subject and the parameters of the RL agent.

2. Update the probability of choosing each action based on the consequences of last action.

3. Repeat for each step of the trace.

We are going to specify how to perform step 1 assuming different policy types. Note that the update step is exactly the same as if the agent were playing the games freely. The update equations for different RL agents are specified in section 2.1 on page 9. Note also that there is no straightforward way of, given a RL agent with a set of parameters, calculating $p(q_i|\Theta)$. The consequences of this will be shown in section 3.5.

ε-greedy In an ε-greedy policy, the most valuable action has a probability of $1-\epsilon$ and the other action has probability of ϵ. Therefore,

$$\begin{aligned} Pr(a_i|q_i) &= (q_i)^{\mathbb{I}(a_i=a^1)}(1-q_i)^{1-\mathbb{I}(a_i=a^1)} \\ &= (1-\epsilon)^{\mathbb{I}(a_i=A_i)}(\epsilon)^{1-\mathbb{I}(a_i=A_i)} \end{aligned}$$

where $\mathbb{I}(a_i = A_i)$ is an indicator function and A_i is the most valuable action at state i. If the agent uses a Q-table, for example, then $A_i = \arg\max_a Q^\pi(s_i, a)$.

Softmax With a softmax policy, the probability of a given action is given by a softmax activation function. Therefore,

$$\begin{aligned} Pr(a_i|q_i) &= (q_i)^{\mathbb{I}(a_i=a^1)}(1-q_i)^{1-\mathbb{I}(a_i=a^1)} \\ &= \frac{e^{\frac{Q^\pi(s_i,a_i)}{\tau}}}{\sum_{j=1}^{J} e^{\frac{Q^\pi(s_i,a^j)}{\tau}}} \end{aligned}$$

where τ is one of the parameters in $\mathbf{\Theta}$, the Gibbs temperature.

3.4 Top-down approach

In this section we propose a method of extracting the discount factor γ that, unlike the previous methods, does not require the assumption that the subject is operating under a particular RL agent. This method was implemented in MATLAB. We require the lighter assumption that the subject is using a geometric discount factor, which we will try to measure directly. The intuition behind it is simple. If we know the values of the rewards that made the subjects switch policies we can infer the implied γ that was used. Let us look at a specific example. In game 1, as was shown in section 3.1.2, the value of γ is given by

$$\gamma = \frac{-25}{25\pi(s^2, a^1) - \pi(s^2, a^1)x}$$

where $\pi(s^2, a^1)$ is the probability of choosing action 1 in state 2 and x is the current estimate of the reward value. Thus, in order to extract the gamma, we need:

1. The dominant policy that is being undertaken at the choice state in each time step to identify the points where the policy changes.
2. The probabilities of choosing the non-dummy action at each non-choice state.
3. The current estimate of the reward value (or probability).

Note that each of these steps is less trivial than what may appear at first. It is clear that the dominant policy at each time step is not the action that was actually chosen at that time step, because we have to take exploration into account. A subject may be choosing the short path as his policy and yet visit the long path less often in order to sample its current value. Therefore, in order to infer the dominant policy that was being used, we also have to take into account what happened at the neighbouring time steps. The same reasoning is true for the non-choice states. At the same time, the subjects don't have direct access to the reward function, so they don't act according to the knowledge of the current value of the reward function but instead according to their estimate of the current reward value, which depends on their past experience. This problem will also be addressed. Also, since we expect multiple switches to happen in a single trace, there is the possibility that not all the resulting values of γ will be the exactly the same. Since we expect, according to our assumption, that for each game there will be one discount factor, we have to convert this sequence of γ into a single one. With all this information we are able to infer the discount factor used by each subject in each game. We now present the solutions to each of the problems previously described.

3.4.1 Dominant policy inference

The aim of inferring the policy is to determine which is the dominant action on visits to the choice state and to determine the probability of each action for non-choice states. As each choice is always between two actions in each state, the problem can be defined in the following manner: there is a sequence of N binary data points $\{X_i\}$, so $X_i \in \{0, 1\}$ and we assume that these are sampled from a Bernoulli distribution i.e. $X_i \sim Bern(q_i)$, for $q_i \in [0, 1]$. Intuitively, q_i is the probability of choosing action 1 at the ith visit to a certain state, with $X_i = 1$ meaning that action 1 was chosen at the ith visit to the state. The dominant policy at visit i, π_i, is π^1 if the dominant action is action 1 and π^2 if the dominant action is action 2. Therefore, $q_i < \frac{1}{2} \implies \pi_i = \pi^1$ and $q_i > \frac{1}{2} \implies \pi_i = \pi^2$. We now present two possible methods for inferring the dominant policy.

Window method It's reasonable to assume that q_i changes slowly, i.e. that consecutive visits to a state follow approximately the same dominant policy. Formally, this means that $q_i \approx q_{i\pm k}$, for small values of k, which is the same as saying that nearby points are generated from approximately the same Bernoulli distribution, $X_j \sim Bern(q_i \pm \delta q)$. Therefore, for a given k, we make the frequentist inference of $q_i$as the average of the binary values inside the window of width $2k + 1$. If $k = 0$, then $\hat{q}_{i,0} = X_i$; if $k = 1$, the window size is 3 $(i - 1, i, i + 1)$ and $\hat{q}_{i,1} = \frac{X_{i-1}+X_i+X_{i+1}}{3}$; generally

$$\hat{q}_{i,k} = \frac{\sum_{j=i-k}^{i+k} X_j}{2k + 1}$$

For the special case where the window size is greater than the number of points available, $\min(i - 1, N - 1) < k \leq max(i - 1, N - i)$ (since $i - 1$ is the number of points to the left of action i and $N - i$ the number of points to the right), then

$$\hat{q}_{i,k} = \frac{\sum_{j=min(1,i-k)}^{max(N,i+k)} X_j}{min(k + i, k + N - i + 1)}$$

Since we do not know the value of k used by each subject, we will take into account the estimates produced by all values of k using a weighted average,

$$\hat{q}_i = \sum_{k=0}^{max(i-1,N-i)} a(k)\hat{q}_{i,k}$$

The function $a(k)$ only has to satisfy the relationship $\sum_k a(k) = 1$ and has to be monotonically decreasing. In this report we consider three hypothesis of weighting: harmonic, $a(k) \propto \frac{1}{k}$, inverse-square $a(k) \propto \frac{1}{k^2}$ and inverse-square root $a(k) \propto (\frac{1}{k})^{1/2}$.

Ratio method Another way to infer the dominant policy is to determine, within a window of width k, whether it is more likely to observe the sequence of X_i inside that window if $q > \frac{1}{2}$ or if $q < \frac{1}{2}$. This is equivalent to say that we infer that the dominant policy is π^1 if $qrat > 1$ and that the dominant policy is π^2 if $qrat < 1$.

$$\begin{aligned} qrat_k &= \frac{p(\{X_j\}_{j=i-k}^{i+k} | q_i > \frac{1}{2})}{p(\{X_j\}_{j=i-k}^{i+k} | q_i < \frac{1}{2})} \\ &= \frac{\prod_{j=i-k}^{i+k} p(X_j | q_i > \frac{1}{2})}{\prod_{j=i-k}^{i+k} p(X_j | q_i < \frac{1}{2})} \end{aligned}$$

For readability we shall drop the index on q_i. If we take the logarithm of $qrat$ we obtain $qind$, defined by

$$qind_k = log(qrat_k) = \sum_{j=i-k}^{i+k} \log p(X_i | q_i > \frac{1}{2}) - \sum_{j=i-k}^{i+k} \log p(X_i | q_i < \frac{1}{2})$$

We infer that the dominant policy is π^1 ($X_i = 1$) if $qind_k > 0$.

Knowing that $p(X_i|q_i > \frac{1}{2}) = \int_{\frac{1}{2}}^{1} p(X_i|q)p(q)dq$, $qind_k$becomes

$$qind_k = \sum_{j=i-k}^{i+k} \log[\int_{\frac{1}{2}}^{1} p(X_i|q)p(q)dq] - \sum_{j=i-k}^{i+k} \log[\int_{0}^{\frac{1}{2}} p(X_i|q)p(q)dq])$$

Assuming a flat prior over q, then

$$\begin{aligned} qind_k &= \sum_{j=i-k}^{i+k} \log[\int_{\frac{1}{2}}^{1} p(X_i|q)dq] - \sum_{j=i-k}^{i+k} \log[\int_{0}^{\frac{1}{2}} p(X_i|q)dq]) \\ &= \sum_{j=i-k}^{i+k} X_j \log(\frac{3}{8}) + (1 - X_j) \log(\frac{1}{8}) - X_j \log(\frac{1}{8}) + (1 - X_j) \log(\frac{3}{8}) \end{aligned}$$

Like the previous methods, if a point lies at one of the boundaries we ignore the indices that extend beyond our limits, obtaining

$$qind_k = \sum_{j=max(1,i-k)}^{min(N,i+k)} X_j \log(\frac{3}{8}) + (1 - X_j) \log(\frac{1}{8}) - X_j \log(\frac{1}{8}) + (1 - X_j) \log(\frac{3}{8})$$

The estimate of $\hat{q}$ will be defined by the average of the observed binary actions in the following way:

$$\begin{cases} \hat{q_k} = \text{average}(\{X_k\}_{\pi_k = \pi^1}) & \text{if } \pi_k = \pi^1 \\ \hat{q_k} = \text{average}(\{X_k\}_{\pi_k = \pi^1}) & \text{if } \pi_k = \pi^2 \end{cases}$$

where $\{X_i\}_{\pi_i = \pi^1}$is the collection of binary actions where the associated dominant policy is $\pi_i = \pi^1$ and $\{X_i\}_{\pi_i = \pi^2}$is the collection of binary actions with the associated dominant policy is π^2.

Since we do not know what k to use, we will use the same weighted average as the one described in the previous paragraph, concerning the window method.

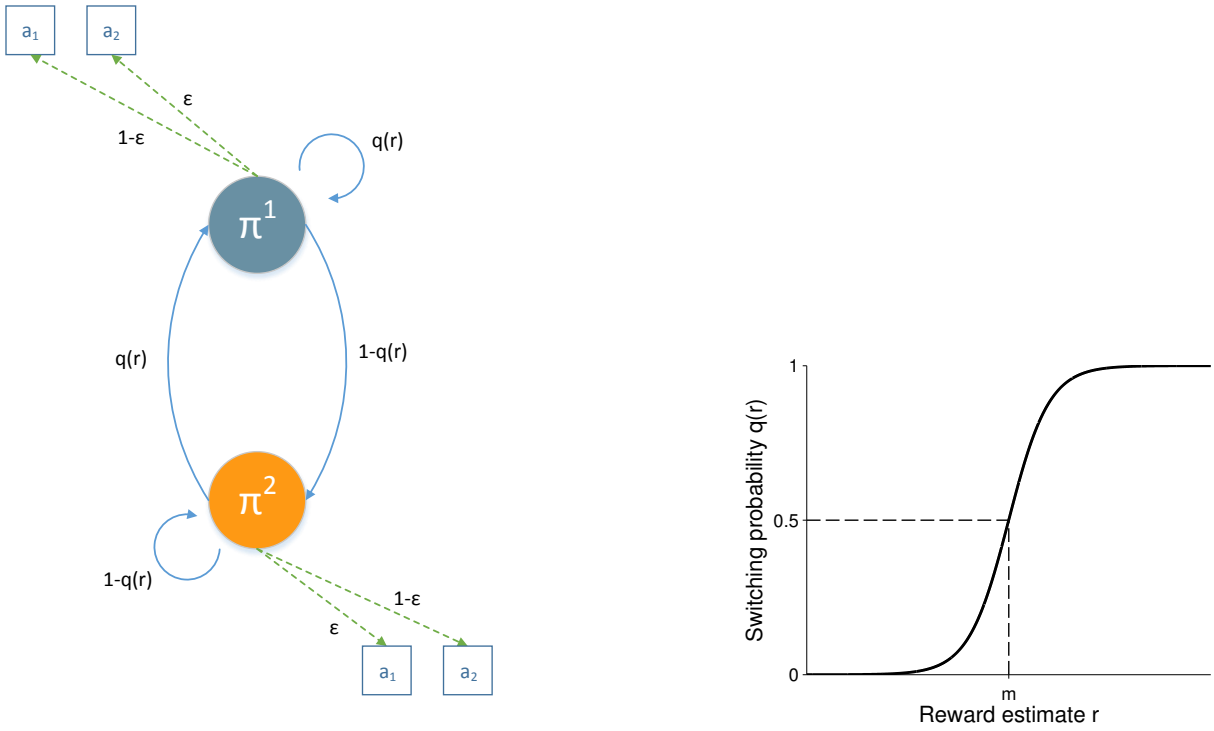
Likelihood It will be useful to know the likelihood of the sequence of obtained dominant actions given the observed actions. This is given by

$$\begin{aligned} Pr(\{a_i\}_{i=1}^{N} | \{\mathbf{q_i}\}_{i=1}^{N}) &= \prod_{s \in S} Pr(\{a_{s,j}\}_{j=1}^{J_s} | \{q_{s,j}\}_{j=1}^{J_s}) \\ &= \prod_{s \in S} \prod_{j=1}^{J_s} q_j^{\mathbb{I}(a_{s,j}=1)} (1 - q_j)^{1 - \mathbb{I}(a_{s,j}=1)} \end{aligned} \tag{3.2}$$

where N is the length of the complete trace,$\mathbf{q_i}$ is the vector of policy probabilities at each state, J_s is the number of visits to state s, S is the collection of states, $\pi_{s,j}$ is the dominant policy at the jth visit to state s, $a_{s,j}$ is the action at the jth visit to state s and $\mathbb{I}(a_{s,j} = 1)$ is the indicator function that has the value 1 if the action at the jth visit to state s was action 1.

3.4.2 Reward value estimate and γ extraction

After the dominant policy inference, we already know the dominant policy at choice states and the probability of each policy at non-choice states. Therefore, only the reward estimation step is missing. We assume that the current estimate of reward value (or probability) is given by a weighted average of the last experienced rewards. We propose two possible weightings, both weighing more the more recent rewards: a gaussian weighting $w(k) \propto e^{\frac{d^2}{2\sigma^2}}$ and a exponential weighting $w(k) \propto e^{\frac{d}{\tau}}$, where d is the distance measured in the number of times the reward action was chosen and σ and τ are parameters of the regarding the width of the weighting function (the higher they are the more rewards are taken into account). Note that $w(k)$ has to obey $\sum_k w(k) = 1$.



(a) The structure of the HMM

(b) The switching function $q(r)$

Figure 3.6: Equivalent Hidden Markov Model. The hidden states are the dominant policies π^1 and π^2, the observables are the actions $a^1$and a^2, $q(r)$ is the switching probability and $1-\epsilon$ is the probability of emitting the action associated with each dominant policy.

For both weighting functions we have to find the best parameter (σor s). Since we are assuming that the subjects have one specific γ when they play each game, then the best width parameter will be the one that gives the most consistent set of γ's (remember that, for each switch in the dominant policy, one γ is extracted, depending on the current estimate of reward and the probabilities of the actions at each state, as described in section 3.1.2). Therefore, we have to solve an optimization problem where the objective function is a function of the standard deviation of the collection of extracted γ's. This was implemented in MATLAB using the *fminunc* function, that finds the minimum of a function with unconstrained parameters. However, if the width parameter is too large, then the averaged reward function will be flat, which will in turn drive the standard deviation of the γ's to 0, while giving no information regarding the true value of the discount factor. Therefore, we introduce a penalty term that prefers lower widths, the standard deviation of the whole averaged reward sequence $\{r_i\}$. Intuitively this means that if the whole average reward sequence has a wide range of values (high standard deviation) and a low standard deviation of the collection of γ's, then we can be confident that the obtained γ is meaningful. If we have a narrow range in the whole average reward sequence and a narrow range in the collection of γ's, nothing can be inferred. Therefore, we select the parameter σ or s that minimizes the quantity $\frac{std(\{\gamma_i\})}{std(\{r_i\})}$. The resulting γ is then the average of the collection of γ's.

Performance evaluation To understand how well this model explains our data, we have to be able to compute the likelihood of seeing the inferred dominant policies at the choice state given that the subject's discount factor was the one inferred in the last step. To do so, it's convenient to present an alternative description of this task as a Hidden Markov Model.

The equivalent description of the tasks is depicted in figure 3.6. At each visit to the choice state, the subject may be following policy π^1 or π^2. However, we have no way to know which directly. We can, however, observe the actions that the subject executes, and this gives us information regarding which of the two policies is more likely to be the current one, since the probability of emitting each

action is different in each policy (for the sake of simplicity we assume that this probability is constant and that is symmetric for both states). The subject may also switch from one policy to the other, and the probability of doing so depends on its current estimate of the reward value, the discount factor and on the current probability of each action at non-choice states.

It's reasonable to assume that the subjects switching behaviour resembles the one presented in figure 3.6b. For a given γ and probability of each action there is a value of the reward function that makes both policies equally valuable (as described in section 3.1.2), which we call m. If the current reward estimate is higher than m, then the probability of switching to (or staying in) policy π^1 (the long path) should increase and be greater than 0.5. The higher the estimate of the current reward, the higher this probability should be. If, on the other hand, the current reward estimate is lower than m, then the probability of switching to (or staying in) policy π^2 should increase and be greater than 0.5. This behaviour can be described by a generalized logistic function with mean m and a certain width s (that we would have to optimize), i.e. the probability $q(r, s, m)$ of transitioning to policy π^1 is given by

$$q(r, s, m) = \frac{1}{1 + e^{-\frac{1}{s}(r-m)}}$$

where r is the current reward estimate.

We can then compute the likelihood of seeing a sequence of dominant policies given the value of $q_1(r)$ and $q_2(r)$ at each time step, that in turn depends on s, on the previously calculated m and on the immediate estimate of reward at each time step. This is given by

$$Pr(\{\pi_i\}_{i=1}^{N} | m, s, \{r_i\}) = \prod_{i=1}^{N-1} t(\pi_i, \pi_{i+1} | r_i, s, m) \tag{3.3}$$

where

$$t(\pi_i, \pi_{i+1}) = \begin{cases} q(r, s, m) & \text{if} \pi_{i+1} = \pi^1 \\ 1 - q(r, s, m) & \text{if} \pi_{i+1} = \pi^2 \end{cases}$$

The value of s is chosen by finding the maximum of the likelihood described above, via an optimization method. Note that the HMM could not be inferred at once (obtaining the dominant policies, the width of the averaging weights, the centre of the logistic function in the same step), since it doesn't capture the probabilities of policies at non-choice states, which are crucial to convert the reward switching value to a value of γ.

Full Hidden Markov Model implementation Having presented the Hidden Markov Model as a way to obtain the likelihood of our complete model, we now present a possible Hidden Markov Model full solution, that would allow us to directly optimize the value of γ. This method isn't implemented in this project, due to time constraints, being included as a possible future step. However, we describe it here since it is a natural extension to the previous model. The main line of reasoning is the same as presented in the previous paragraph, but instead of using the previously inferred γ and policy sequence, we would infer them directly as well. In order to do so, we introduce two changes in the previous model. First, we remove the assumption that the transition function is symmetric. This may account for the possibility that the switch doesn't happen for the same value of the reward when going from π^1 to $\pi^2$and when going from π^2 to π^1 (figure 3.7). Then, we include in the likelihood function the emission probability of each action in each state. The problem is then reduced to finding the set of parameters that maximizes that likelihood. Since this is a non-homogeneous HMM, it's not possible to use the Viterbi algorithm, with the best strategy being using sampling to find that set of parameters.

The new set of equations is given below. The new asymmetric transition probabilities are

$$t(\pi_i, \pi_{i+1}) = \begin{cases} q_1(r, s_1, m_1) & \text{if} \, \pi_i = \pi^1, \pi_{i+1} = \pi^1 \\ 1 - q_1(r, s_1, m_1) & \text{if} \, \pi_i = \pi^1, \pi_{i+1} = \pi^2 \\ q_2(r, s_2, m_2) & \text{if} \, \pi_i = \pi^2, \pi_{i+1} = \pi^1 \\ 1 - q_2(r, s_2, m_2) & \text{if} \, \pi_i = \pi^2, \pi_{i+1} = \pi^2 \end{cases}$$

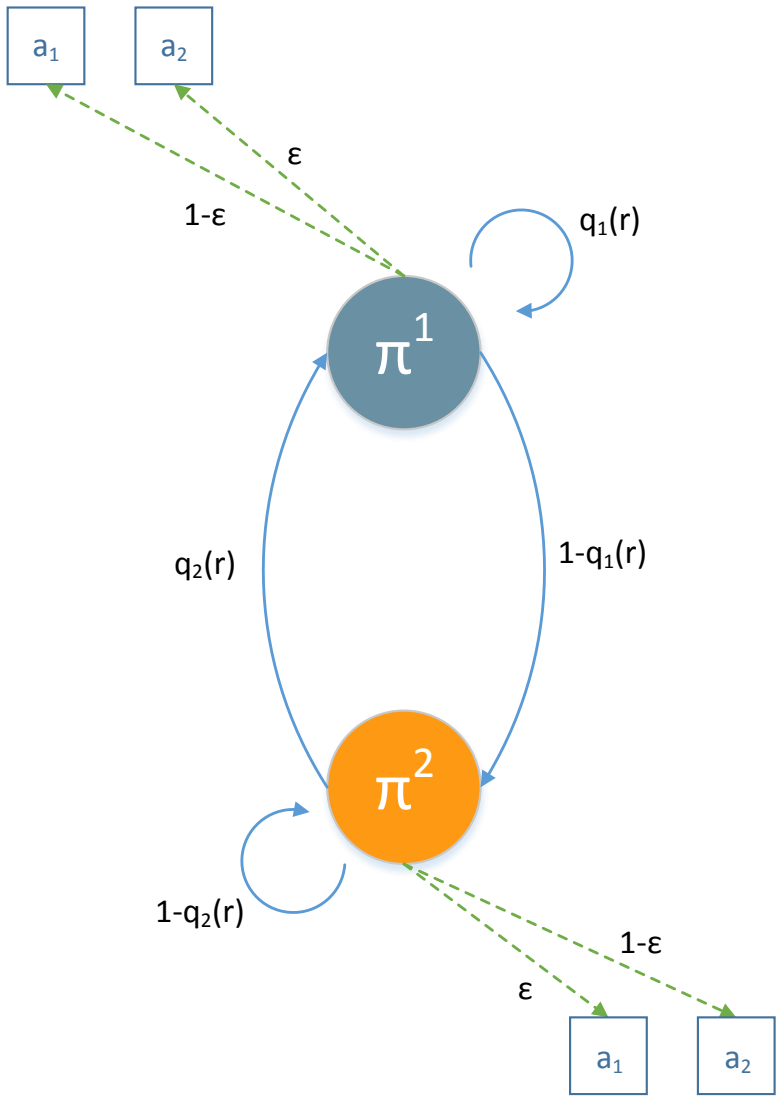


Figure 3.7: HMM with asymmetric transition probabilities

and the new likelihood function is given by

$$Pr(\{a_i\}_{i=1}^N, \{\pi_i\}_{i=1}^N, m, s, \{r_i\}) = \prod_{i=1}^{N-1} t(\pi_i, \pi_{i+1}|r_i, s, m)e(\pi_i, a_i)$$

where $e(\pi_i, a_i)$ is the probability of observing a_i given that we are in policy π_i, i.e.

$$e(\pi^m, a^n) = \begin{cases} 1-\epsilon & \text{if } m = n \\ \epsilon & \text{if } m \neq n \end{cases}$$

Another possible modification is including memory in the model, which is to say that before effectively switching policies we have to transition between more than one hidden state. A possible scheme with 3-state memory is presented in figure 3.8.

3.5 Comparing different methods

In order to compare how well each method captures and describes the observed data, we use the Akaike Information Criterion. Simply comparing the maximized likelihoods of each method

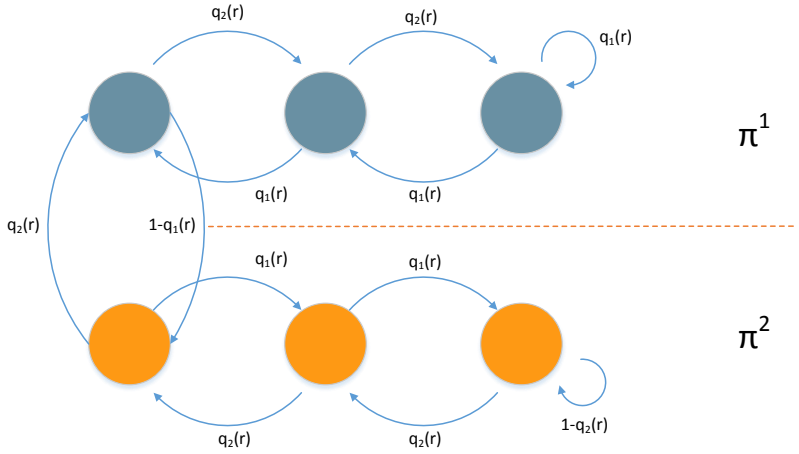


Figure 3.8: HMM with 3-state memory

wouldn't be correct since each method may have a different number of parameters. Therefore, in this section, we specify the likelihood function as well as the number of free parameters for each method.

Bottom-up For the bottom-up methods, the number of free parameters is trivial. For each we have 3 parameters, i.e. α, γ and $\epsilon/\tau/\theta$.

The log-likelihood function is given by (we represent our obtained data, states actions and rewards as $\mathcal{D}$, our parameters set as Θ and the probabilities of executing each action at the choice state as q_i)

$$Pr(\Theta|\mathcal{D}) = Pr(D|\{q_i\})Pr(\{q_i\}|\Theta)Pr(\Theta)$$

where $Pr(D|\{q_i\})$ was defined in eq. 3.1 and we assume a flat prior over all parameters.

As previously described, note that we are only able to calculate $Pr(D|\{q_i\})$, having no clear way to compute $Pr(\{q_i\}|\Theta)$. The natural choice is then to use $Pr(D|\{q_i\})$ as the metric of comparison between methods.

Top-down For the top-down method, the number of free parameters is less trivial. We have the parameters of the policy inference step and the parameters of the γ extraction step. The latter has the mean of the sigmoid m, the width of the sigmoid s and the width of the averaging filter weights, s or σ. We prove that the number of free parameters depends on the smoothing function used.

Let's first consider each window width independently. If we have a trace of length N, we have N possible window lengths w, where $w = 2k + 1$. The trivial cases are when $k = 0$, where we have N free parameters (each time step can have an independent dominant policy) and when $k = N$, where we have 1 free parameter (defining one dominant policy affects all the others). The number of free parameters for a given window width k is the maximum number of independent dominant policies that can be set that don't affect each other. Let's go through an example. If we choose $k = 1$ ($w = 3$), in a trace of length 5, we can set the first dominant policy, and that will influence the second dominant policy, so the second isn't a free parameter. However, the influence of the first dominant policy doesn't reach the third, so the third is again a free parameter. The fourth is influenced by the third, but the fifth is not, so the fifth is a free parameter. This is illustrated in figure 3.9. We can then see by induction that the number of free parameters is given by $f_N(k) = N \div k + N \mod k$.

However, since we are smoothing over all windows, the number of free parameters is not the number of parameters of a single window but a weighted average of the free parameters for each

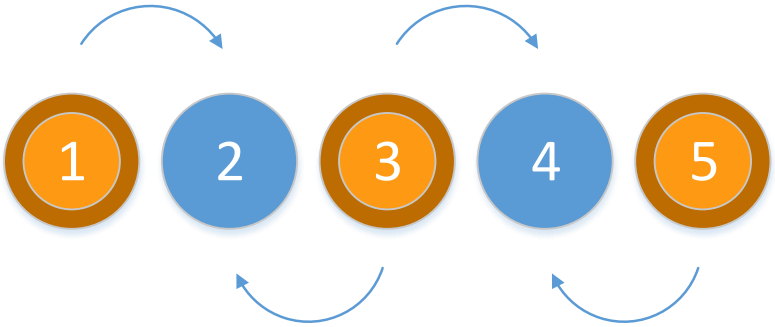


Figure 3.9: Number of free parameters in a sequence of 5 states with $k = 1$. Orange double-circles represent independent states, the arrows represent their influence over neighbouring states and blue states represent dependent states.

used window. Each window will be weighted according to the weight it had in the dominant policy inference. Therefore, the effective number of free parameters is $f_{eff} = \sum_{k=1}^{N} a(k) f_N(k)$, where $a(k)$ is the weighting function used as described on page 26.

The reason for this weighting can be understood using an example. Suppose that our weighting function is $a(k) = 1, k = 0$ and 0 for for all other cases. This is the same as using just a window of width 0, so the number of free parameters should be N. If it was $a(k) = 1, k = N$, then the number of free parameters would be 1. What if we have a linear combination of both windows, where each is weighted by half (i.e. $a(k) = 0.5, k = 0, N$)? Then, the number of free parameters should be $0.5N + 0.5$.

Therefore, for the top-down analysis, the number of free parameters is f_{eff} together with the parameters of the γ extraction step, which are 3.

The likelihood function for the top-down approach is given by

$$Pr(\Theta|\mathcal{D}) = Pr(D|\{q_i\})Pr(\{q_i\}|m, s, \sigma)Pr(m)Pr(s)Pr(\sigma)$$

where $Pr(D|\{q_i\})$ was defined in eq. 3.2 and $Pr(\{q_i\}|m, s, \sigma)$ in eq. 3.3, and we will assume a flat prior over all parameters. For the reason described in the bottom-up method, however, only $Pr(D|\{q_i\})$ will be used to compare each method, and the total number of free parameters is that of the inference step, f_{eff}.

Chapter 4

Results

4.1 Synthetic data

In order to assess the quality of each of our methods, we use different RL agents to generate traces of synthetic data. Having the knowledge of the generating parameters we are able to evaluate how well we can recover them using all of our methods. We use a broad set of generating agents and parameters to evaluate the conditions under which each method performs the best. Ultimately, we want to ascertain whether our top-down and bottom up methods are accurate independently of the what agent generated the trace. Even with a very high accuracy, a method which works only very strict conditions is not fit to recover the parameters in real data, because we do not know if the experimental traces satisfies those conditions, making the results unreliable.

We test each component of each method individually as well as the method as a whole. Our metrics are the error in the inferred discount factor and the likelihood of the inferred model having generated the trace, as described in the Methods section.

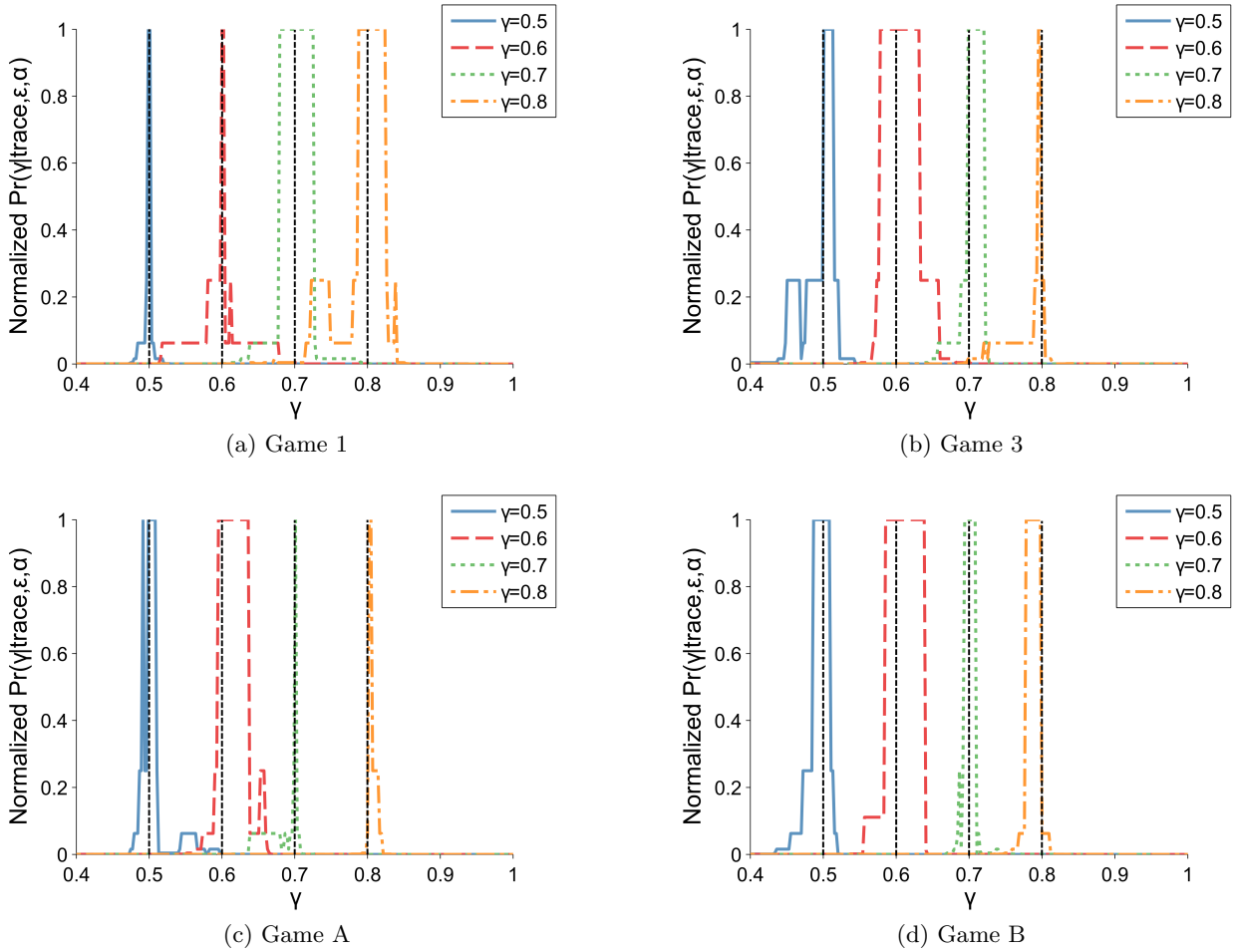
4.1.1 Bottom-up methods

Since the bottom-up approach consists of a direct optimization problem, there are no subsystems to analyse. However, we need to guarantee that, at least in ideal conditions, the parameters are recoverable. By ideal conditions we mean recovering the parameters of the agent that generated the trace using an agent with the same agent type and policy type. This means that, if we generated a trace using a Q-Learning agent with an ϵ-greedy policy, we expect to be able to accurately recover the agent parameters if we our recovery agent is also a Q-Learning agent with an ϵ-greedy policy. We will show the results for SARSA learners as an example, but the results obtained using Q-Learning are similar except where stated otherwise.

We test how well we could predict one of the parameters given that all others were known. The results are depicted in figures 4.1 and in Appendix A, with a particular set of parameters that allowed the recovered parameter to have an absolute error of under 0.05 (for games 1 and 3, the maximum error for the temperature was set at 5). We also investigate the the range of values for each parameter that satisfy the aforementioned criterion. For the cases where multiple values result in the same probability, we require the worst case absolute error to be under 0.05. The results are presented in table 4.1. The limitation on γ is due to the fact that the games were devised to only work for $\gamma > 0.5$, due to the chosen values for the rewards. The limitations on α are more serious, though. An α of less than 0.1 results in severe flat regions for every other parameter, as can be seen in figure 4.2. This means that if a real subject has a learning rate of under 0.1, the bottom-up methods will not be able to recover the generating parameters.

These results remain true when remain true when recovering two parameters at the same time. This is illustrated in figure 4.3, where the low value of α causes the maximum likelihood region to be extremely large, not permitting the recovery of the true value of γ.

Finally, we obtained the absolute error of the predictedγ for each game, averaged over 4 traces generated with different values of γ ($\gamma = \{0.5, 0.6, 0.7, 0.8\}$). For each synthetic trace we infer the most likely value of γ using all the bottom-up methods at our disposal. The results are presented



(a) Game 1

(b) Game 3

(c) Game A

(d) Game B

Figure 4.1: Normalized probability of γ matching the value of γ that generated a given trace. For each game we generated 4 traces with a SARSA learner with an ϵ-greedy policy, with $\epsilon = 0.2$, $\alpha = 0.4$ and $\gamma = \{0.5, 0.6, 0.7, 0.8\}$. Note that the maximum for each value of generating γ lies within less than 0.05 of the true value. The results were normalized so that the maximum for each generating parameter would be 1.

Parameter	Range
γ	[0.5,1]
ϵ	[0,1]
α	[0.1,1]
τ	[5,50] (games 1 and 3) ; [0,1.5] (game A and B)

Table 4.1: Range of parameters that allows for the accurate recovery of every other parameter. Accurate recovery is defined as an absolute error of 0.05 for the most likely parameter. In the case that multiple parameters result in the same probability, the worst-case scenario is used.

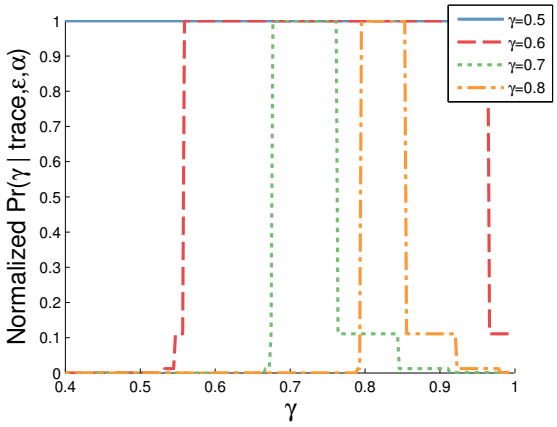
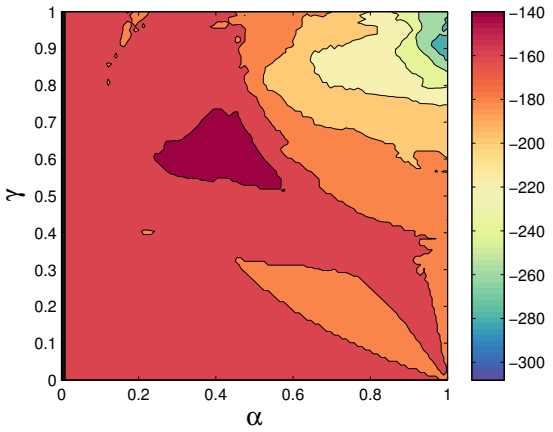
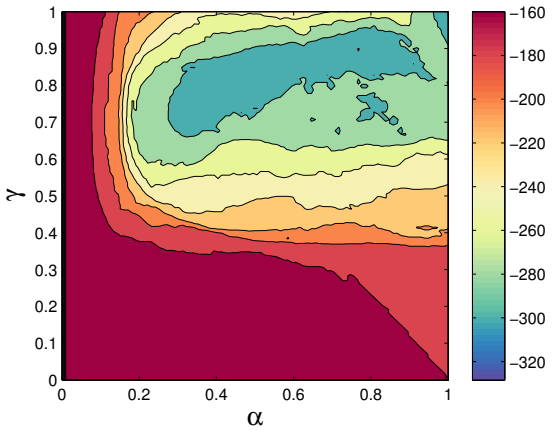


Figure 4.2: Flat regions caused by a generating $\alpha < 0.1$, turning the recovery of the generating parameter impossible.

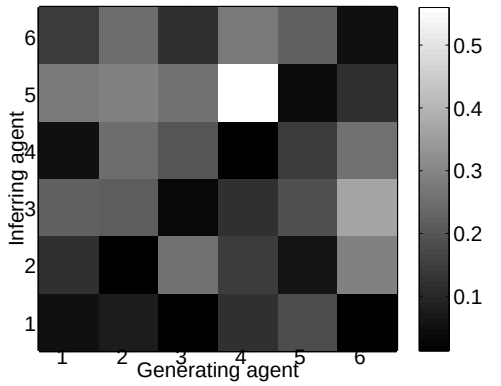


(a) Successful recovery of the parameters with a generating α of 0.4

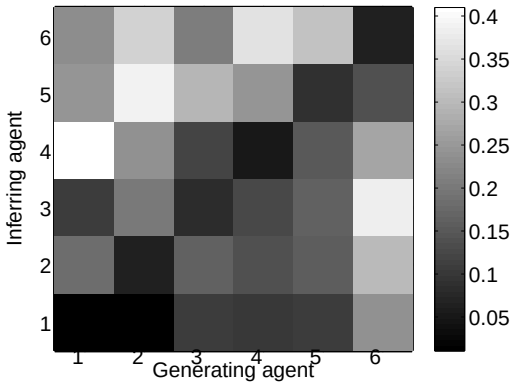


(b) Failure in the recovery of the parameters due to a generating $\alpha$of 0.05

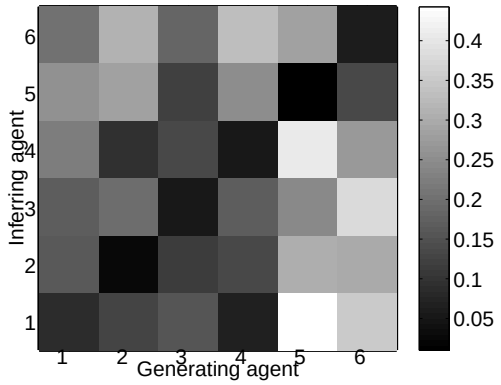
Figure 4.3: Contour plots regarding the likelihood of α and γ being the true generating parameters, with the true $\gamma = 0.6$ and $\epsilon = 0.2$.



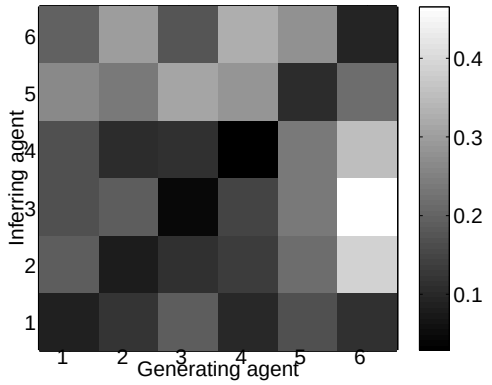
(a) Game 1



(b) Game 3



(c) Game A



(d) Game B

Figure 4.4: The absolute error of the predicted value of γ for all 4 games. The error is averaged for 4 traces generated with different values of γ ($\gamma = \{0.5, 0.6, 0.7, 0.8\}$). Note the low values of error when the type of agent and policy are matched, which happens on the values of the diagonal. In this heatmaps, the labels correspond to: 1- SARSA Softmax; 2- SARSA ϵ-greedy; 3- Q-Learning ϵ-greedy; 4- Q-Learning Softmax; 5- Model-based.

in figure 4.4. Note that, when the generating and inferring agent match in type and policy, the average error is extremely low, which can be seen by observing the diagonal in the heatmap.

4.1.2 Top-down method

To evaluate the performance of the top-down method, we need to compute the accuracy of the two steps of the discount factor inference, i.e. the dominant policy inference and the γ extraction step. Using synthetic data, we have access to the policy trace, i.e. the probability of executing each action at each state for each time step. Thus, by comparing the predictions with the actual policy we can obtain the accuracy of the policy inference method. To compute the accuracy of the γ extraction step, we use the actual policy trace as the dominant policy estimates to isolate the error that could be due to the first step and compare the resulting inferred γ with the γ that generated that particular trace.

The results of the policy inference step are shown in table 4.2. We generate 4 traces per method-weight scheme pair with different policies, two generated with ϵ-greedy policies ($\epsilon = \{0.1, 0.2\}$) and two with a softmax policy ($\tau = \{10, 20\}$) using game 1 as the reference. The values of ϵ are the usual values that ensure the balance between exploration and exploitation and the values of the Gibbs temperatures were chosen to have policy values in the same range as the ϵ-greedy policy. Note that the policy inference step is highly dependent on the characteristics on the generating agent. If we assume that $\epsilon = 0$, then the accuracy of both methods should be close to 1, as the action taken at

Method\Weight Scheme	Harmonic	Inverse-Square	Inverse-Square-Root
Window	0.98	0.91	0.93
Ratio	0.95	0.89	0.93

Table 4.2: Average accuracy of different inference methods over 4 traces, two generated with ϵ-greedy policies ($\epsilon = \{0.1, 0.2\}$) and two with a softmax policy ($\tau = \{10, 20\}$) using game 1 as the reference.

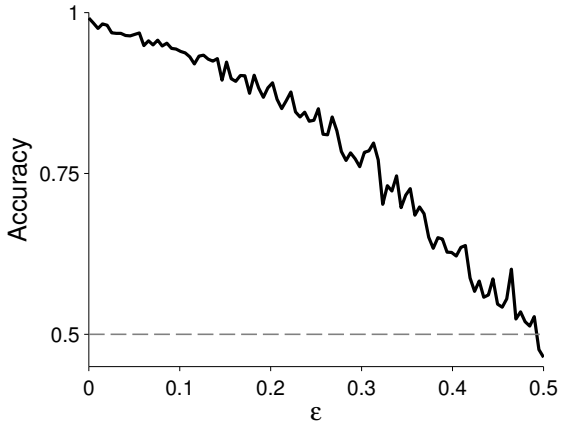


Figure 4.5: Variation of the policy inference accuracy with the greediness of the agent, as measured by the parameter ϵ. We used as an example a SARSA agent with an ϵ-greedy policy and the window method with a harmonic scheme.

each step is always the most valuable one at that point. However, as the exploration increases, it becomes less and less clear what the dominant action is. At the extreme case where $\epsilon = 0.5$, there is total randomness, since independently of the most valuable action, the probability of each action is always 50%. In these cases we expect the accuracy to drop to around 50%. This behaviour can be seen in figure 4.3. Since the window method with harmonic weighting outperforms all the other method-weight schemes pairs, we use it as our primary dominant policy inference scheme.

Using the same synthetic traces described before, but using the actual dominant policies instead of using the inferred ones, we calculate the predicted γ for the exponential and gaussian filters described in section 3.5 and we present the resulting absolute error in table 4.3. As the exponential filter outperforms the Gaussian filter for all games, it is used as our γ extraction method.

4.1.3 Comparing different methods

Since we do not know which (if any) of the RL agents we are considering will be used by real subjects, our best measure of performance with synthetic data must be based on as many different agents as possible. This is easily justifiable by looking at figure 4.4, which clearly shows that the performance of each of the bottom-up methods depends on the agent that was used to generate the data. Therefore, our metrics of performance for each inferring method are both the absolute error of prediction of γ as well as the AIC (loglikelihood penalized by the number of free parameters), both averaged over all generating agents. This eliminates the bias of each inferring agent for its

	Game 1	Game 3	Game A	Game B
Exponential	0.04	0.06	0.06	0.08
Gaussian	0.07	0.08	0.13	0.15

Table 4.3: Accuracy of reward filteres measured in terms of the absolute error of the predicted gamma averaged over four traces, two generated with ϵ-greedy policies ($\epsilon = \{0.1, 0.2\}$) and two with a softmax policy ($\tau = \{10, 20\}$) .

	Game 1	Game 3	Game A	Game B
Top-down	**0.07**	0.11	**0.10**	**0.15**
SARSA Softmax	**0.07**	0.13	0.11	0.16
SARSA ϵ-greedy	0.09	0.18	0.19	0.26
Q-Learningϵ-greedy	0.16	0.15	0.18	0.19
Q-Learning Softmax	0.09	**0.10**	0.19	0.21
Model-based	0.18	0.25	0.23	0.26
Hybrid (Q-ϵ + MB)	0.16	0.14	0.13	0.15

Table 4.4: Absolute error of the prediction of γ averaged over different values of the generating parameters for all generating agents. The lowest error for each game is highlighted in bold.

	Game 1	Game 3	Game A	Game B
Top-down	**270**	**296**	**669**	**644**
SARSA Softmax	279	318	696	685
SARSA ϵ-greedy	303	330	679	693
Q-Learningϵ-greedy	297	328	692	692
Q-Learning Softmax	280	317	695	690
Model-based	326	347	756	771
Hybrid	311	325	684	689

Table 4.5: AIC of the synthetic trace given the inferred parameters averaged over different values of the generating parameters for all generating agents. The lowest AIC for each game is highlighted in bold. Note that these values cannot be compared across games given that their lengths are different, which affects both the likelihood and the number of free parameters per method.

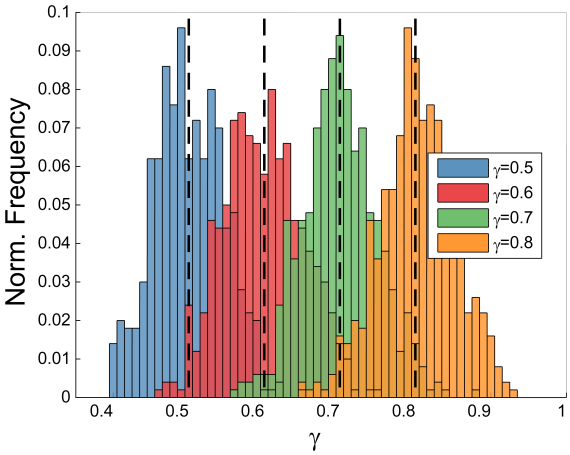
matching generating agent and allows us to evaluate the performance of the method over the whole range of possible agents. The results are presented in tables 4.4 and 4.5.

Notice that, on average, the top-down error and the AIC are lower than any bottom-up approaches, despite the fact that the absolute error for specific agents is lower as we saw before in figure 4.4. This result proves that our top-down method is indeed more flexible and accurate on average than the alternatives. It should also be noted that the particularly high values of the model-based approach are somewhat biased, given that 4 out of 6 generating methods are pure policy-based, rather than transition-based as the model-based method is. Thus, the higher error may arise from there. It should be noted that it is not possible to directly compare the AIC values obtained for different games, as their length and number of free parameters varies. Thus, the comparison should be made between different methods of the same game. Should the need to compare between different games arise, the concept of normalized AIC could be introduced, dividing the AIC by the length of the trace.

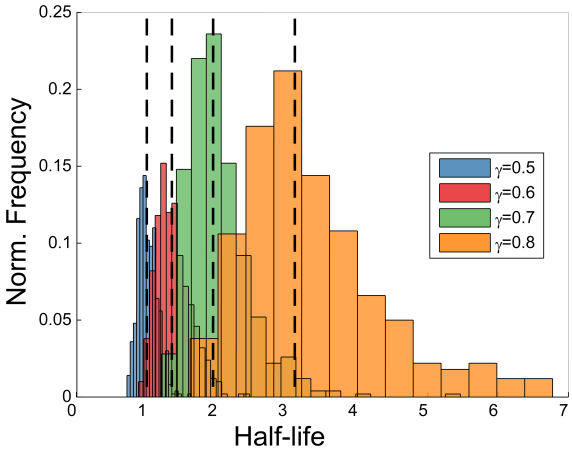
It should be noted that the influence that an error of 0.1 in the prediction of γ depends on the value of γ we are considering, as explained in Murthy [16]. To demonstrate that, it's useful to look at the half-life value, which tells us how many steps are needed before the value of the reward drops to half. The half-life n is computed using the following formula

$$\gamma^n = \frac{1}{2} \Leftrightarrow n\log(\gamma) = \log(\frac{1}{2}) \Leftrightarrow n = \frac{\log(\frac{1}{2})}{\log(\gamma)}$$

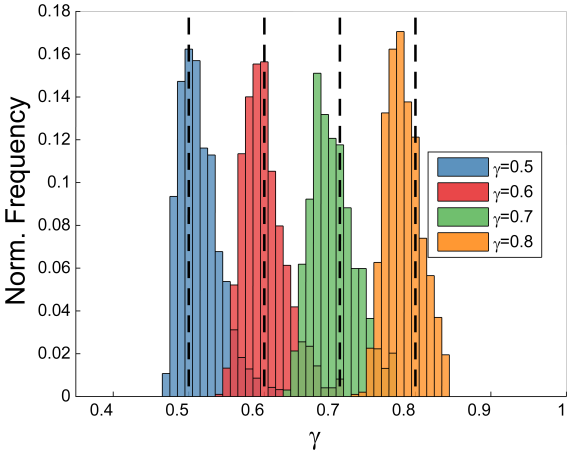
Having $\gamma = 0.5$ results in $n = 1$, while $\gamma = 0.6$ results in $n = 1.35$. The same absolute difference but with different values of γ can result in much higher differences in the half-life. For instance, $\gamma = 0.8$ results in $n = 3.1$, while $\gamma = 0.9$ results in $n = 6.57$. Therefore, if we want to compensate for this difference, then we have to evaluate the accuracy in terms of the half-life. This difference is presented in figure 4.6. Observing the top-down plots, it's quite clear that, while using the inferred γ the errors are very low, the half-life plot shows that the errors for higher values of γ are higher.



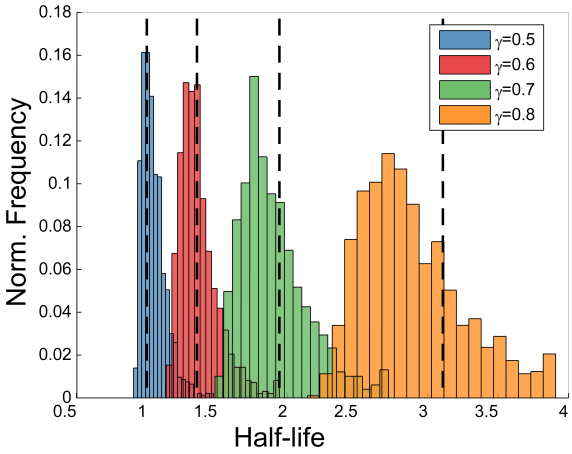
(a) Bottom-up, γ



(b) Bottom-up, half-life



(c) Top-down, γ



(d) Top-down, half-life

Figure 4.6: Frequency of inferred γ or half-life for different values of the generating γ. 1000 traces were generated with a SARSA learner with an ϵ-greedy policy, with $\epsilon = 0.2$, $\alpha = 0.4$ and $\gamma = \{0.5, 0.6, 0.7, 0.8\}$. Figures a) and b) represent the inferred γ/half-life when using a bottom-up method (SARSA ϵ-greedy), while figures c) and d) represent the inferred γ/half-life when using the top-down method to infer it. The dashed lines correspond to the true value of γ/half-life.

	Game 1/3	Game A/B
Top-down	$\mathbf{2.25 \times 10^{-6}}$	$\mathbf{2.41 \times 10^{-5}}$
SARSA Softmax	$\mathbf{3.67 \times 10^{-4}}$	$\mathbf{9.74 \times 10^{-5}}$
SARSA ϵ-greedy	0.43	$\mathbf{6.99 \times 10^{-5}}$
Q-Learningϵ-greedy	**0.01**	0.12
Q-Learning Softmax	$\mathbf{1.56 \times 10^{-5}}$	**0.001**
Model-based	0.52	0.97
Hybrid	0.39	0.28

Table 4.6: P-Values for the double-tailed two sample t-test. The values highlighted represent the cases where the null hypothesis (that both sets of inferred γ's come from distributions with the same mean) was rejected at a 5% significance level.

4.2 Experimental data

The results concerning experimental data are not focused on the accuracy of the inference methods, since we don't have access to the ground truth. Therefore, our analysis is twofold: we analyse the AIC for the different methods as well as the differences in the inferred γ for different games.

The first set of experiments has been conducted by Keshava Murthy (2012), has been performed by 9 subjects. However, one of them didn't play game 1, which is one of the selected games. Therefore, our effective sample size is of 8 subjects. Our new set of experiments, A and B, has been performed by 11 subjects, with ages between 18 and 25 in our lab, following the same procedure as the one used by Murthy.

Hereafter, whenever the Top-down method is mentioned, we will be referring to the top-down method with the window method, using the harmonic scheme and the exponential filter.

4.2.1 Performance comparison

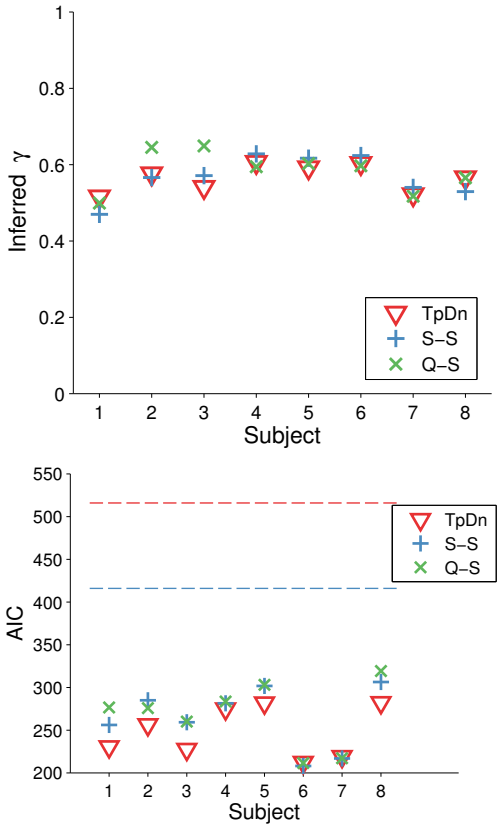
The inferred γ's and the respective AIC for the inference method are presented in figure 4.7. For legibility, only the 3 best performing method (out of 7) are presented, measured by the lowest average AIC over all subjects. For all methods and subjects, all the predictions are better than chance, as shown by the dashed lines. As happened with synthetic data, the top-down method outperforms every bottom-up method. The analysis of the predicted values of γ and their relation to each game is presented in the next subsection.

4.2.2 Comparison of the extracted discount factor in different games

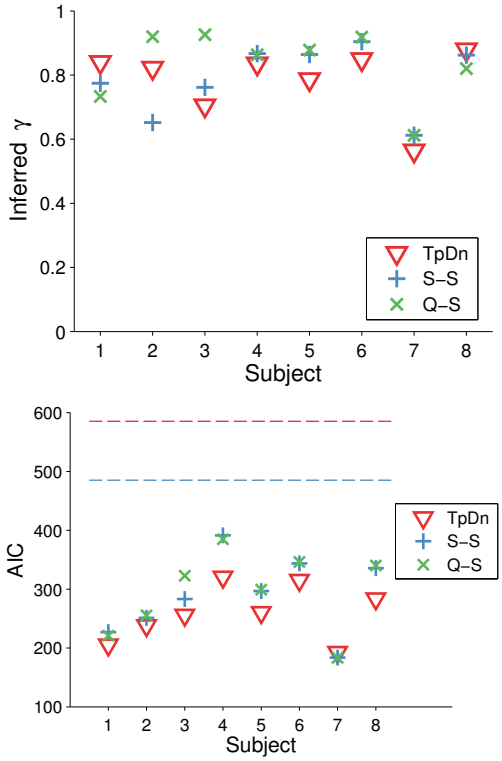
We turn our analysis to the comparison of the values of γ in the different games. Observing the whole population, we want to test the hypothesis that the discount factor that is used by the subject depends on the task he performs. Since we have two groups of experiments with the same structure, we test the hypothesis that the mean γ in games 1 and A differs significantly from the one in games 3 and B, respectively. The first results are depicted in figure 4.8. Note that for all methods the mean γ is higher in games 3 and B than in games 1 and A. To check whether the difference is significant we performed a double-tailed paired t-test and the results are depicted in table 4.6. For the three top-performing methods determined in the last subsection, the hypothesis that both sets of gamma are generated from a distribution with the same mean was rejected at a 5% significance level, allowing us to say that the means for the different games are significantly different.

4.2.3 Assumption evaluation

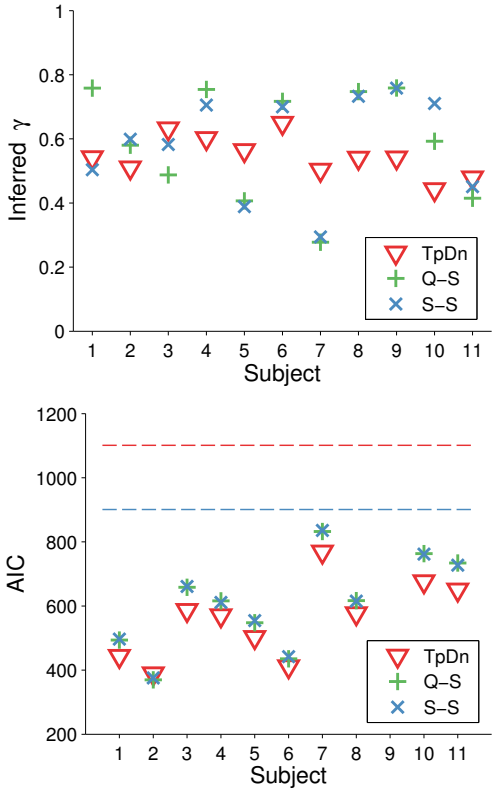
In this section we briefly discuss the evidence we have available to support our initial assumptions, namely that the game is in fact learnt by the players and that it is reasonable to assume that one fixed value of γ is used throughout the game.



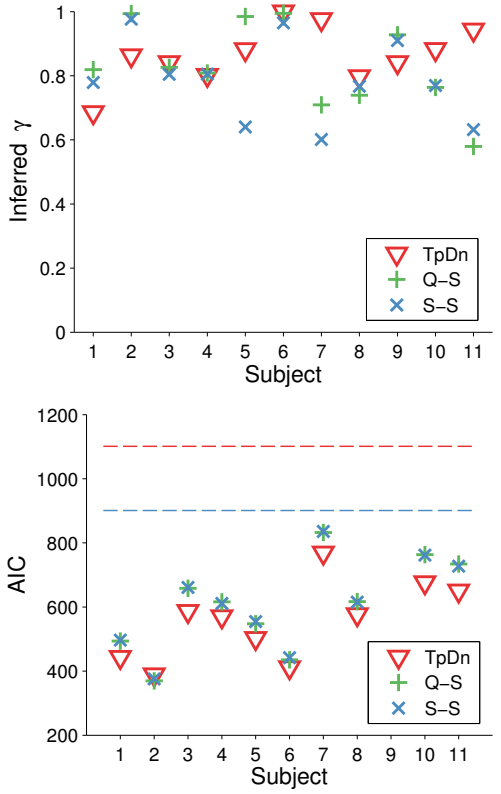
(a) Game 1



(b) Game 3



(c) Game A



(d) Game B

Figure 4.7: Inferred γ and the corresponding AIC for all subjects using the 3 best performing methods (lowest AIC). The dashed lines represent the chance baseline (the top one corresponds to Top-Down methods, the bottom up to Bottom-up). Legend: TD: Top-Down; S-S: SARSA Softmax; Q-S: Q-Learning softmax.

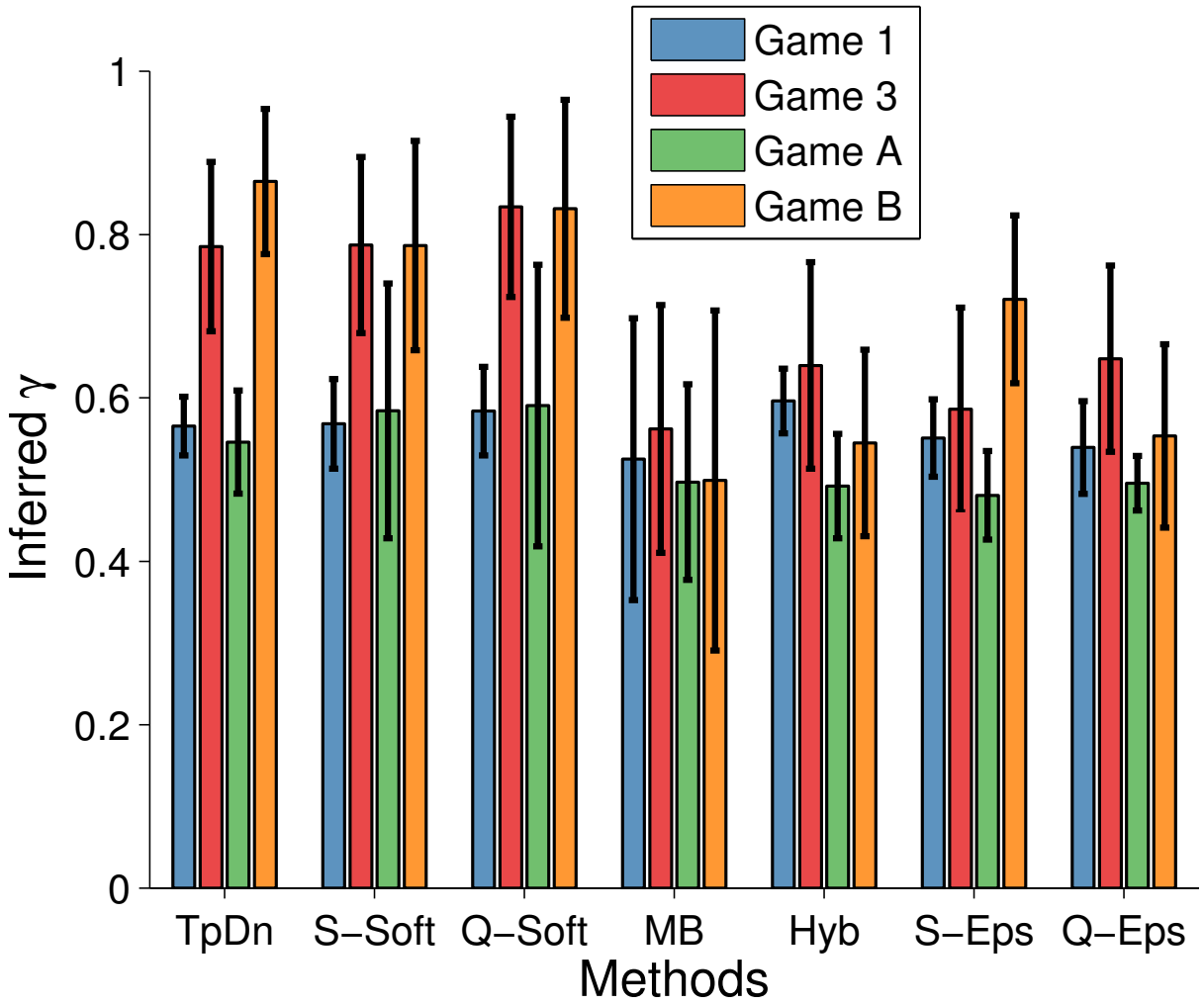


Figure 4.8: The mean γ predicted by the different methods. Games 1/3 - 8 subjects; Games A/B - 11 subjects. The error bars represent 2 standard deviations. Methods: 1:Top-down; 2: SARSA Softmax; 3: Q-Learning Softmax; 4: Model-based; 5:Hybrid; 6: SARSA ϵ-greedy; 7: Q-Learningϵ-greedy

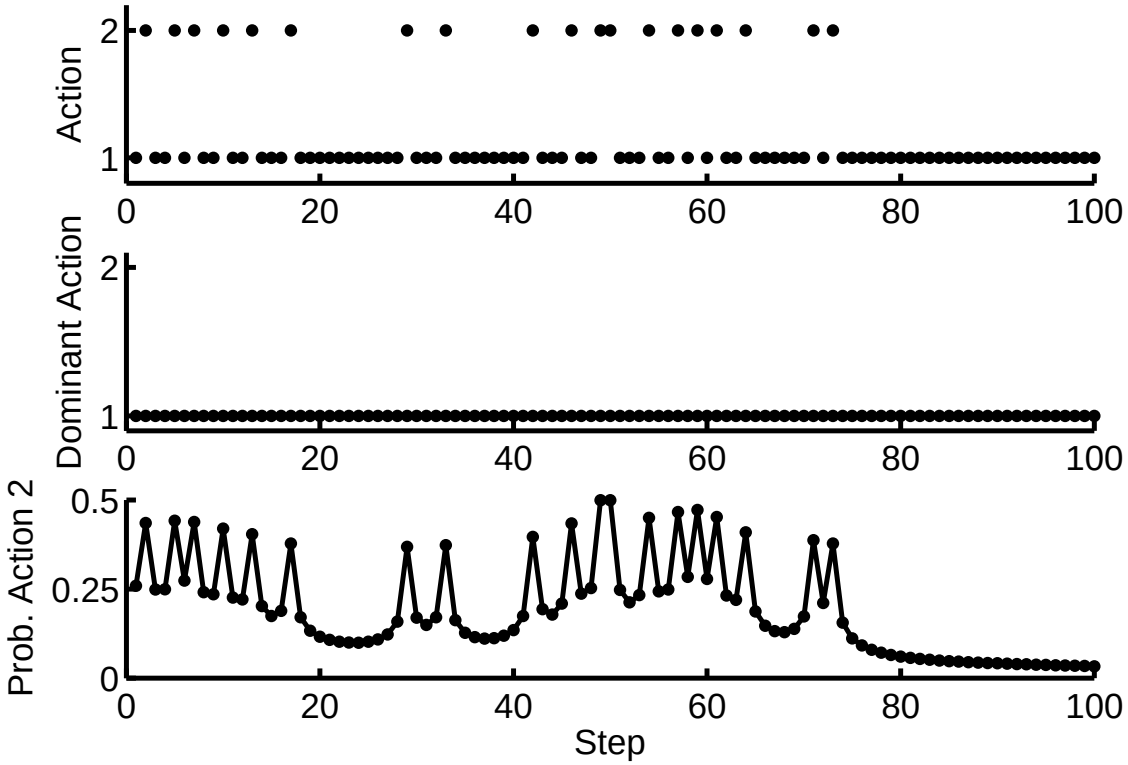


Figure 4.9: Example of policy inference at state 2 of game 1. The exploratory behaviour can be seen in the top plot, but the dominant policy is always the non-dummy action, as shown in the middle plot.

	Game 1	Game 3	Game A	Game B
State 2	97%	94%	93%	95%
State 3	-	96%	-	96%

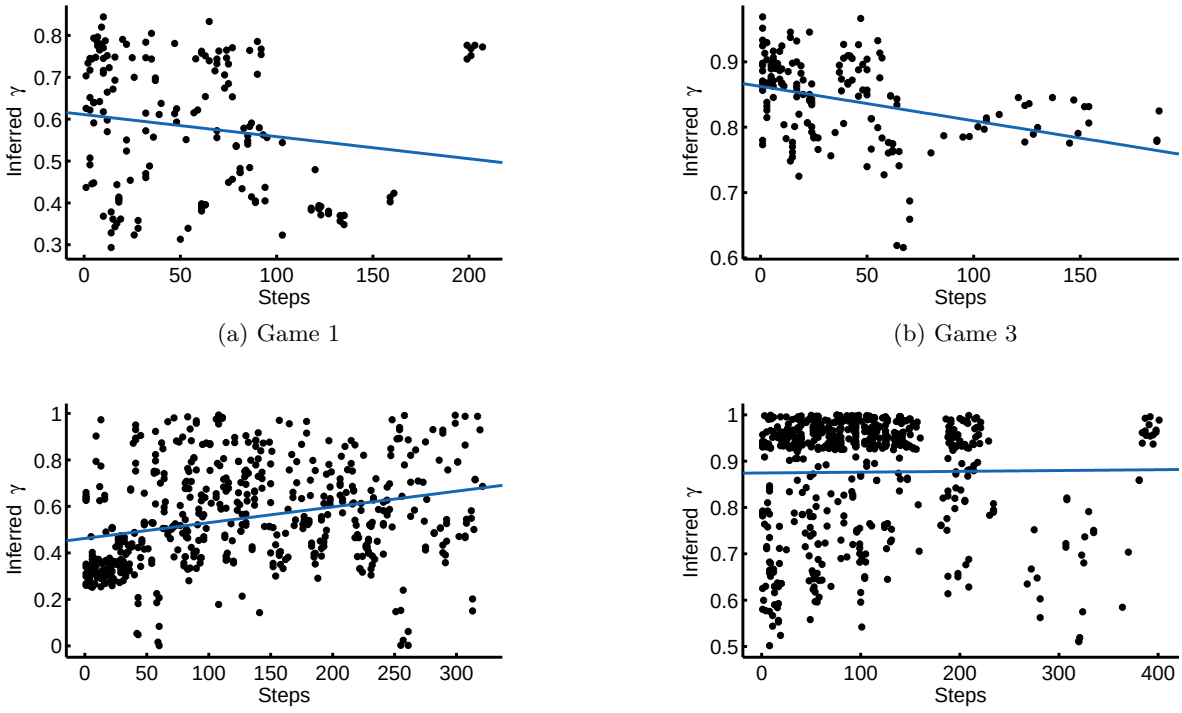
Table 4.7: Percentage of steps in which the followed dominant policy at non-choice states was the non-dummy action for all subjects.

4.2.3.1 Game understanding by subjects

To be able to draw meaningful conclusions from our experiments, it's important to know if the subjects understood the game properly. To assess their understanding, we use two metrics: the first is a direct measure, the percentage of steps in which the dominant action in a non-choice state was the non-dummy action; the second is the perception of the structure of the game as measured by the questionnaire, namely the number of changes that were perceived by the subject and the perception of what changed.

If the subjects captured the games properly, then they should have explored both action at non-visit states and discovered that one of the paths was significantly better than the other, the non-dummy path as described before. An example of this behaviour can be seen in figure 4.9. The top plot, depicting the actual actions, shows that the subject used an exploration-exploitation balance that allowed him to sample the dummy path while keeping the non-dummy action as his dominant policy. The summary of these results in presented in table 4.7. As the percentage of correct path choice is greater than 90% in all games and states, we can reasonably assume that the game was understood by the players.

With respect to the perception of the changes in the games, the data collected in the questionnaires regarding games A and B shows that 8 out of 11 players correctly detected that only the reward values/probabilities were changing, while 10 out of 11 correctly identified the number of



(a) Game 1

(b) Game 3

(c) Game A

(d) Game B

Figure 4.10: Value of γ at each switching point for all subjects grouped by game. The blue line represents the linear regression associated with those switching points.

	Game 1	Game 3	Game A	Game B
p-value	0.17	0.15	0.26	0.44

Table 4.8: P-Values for the double-tailed two sample t-test regarding the slope of the linear regression. The hypothesis that the slope of the linear regression fit is 0 was not rejected at a 5% significance level for any game.

changes as being less than 10. This, again, shows a general understanding of the structure of the game.

4.2.3.2 Fixed discount factor per game

The final verification is to assess how valid the assumption of having a fixed value of γ per game is. In order to do so, we collected the implied discount factor at each switching point for each subject for a particular game and performed a two-tailed t-test to the slope of the linear regression fit. Our null hypothesis is that the slope is 0, i.e. the discount factor is constant for the duration of the game. The results are presented in figure 4.10 and in table 4.8. Note that the null hypothesis was not rejected for any of the games ($p > 0.05$), giving support to our assumption that the value of γ remains the same throughout any particular game.

Chapter 5

Discussion

5.1 Summary outline

In this report we presented two main contributions: an addition to the existing set of behavioural experiments and an ensemble of robust statistical methods that together allow the extraction of the discount factor used in a given task.

We devised a new type of experiments, the hybrid experiments, that use fixed and deterministic rewards and varying probability of obtaining it, in resemblance to the binary experiments previously proposed. We have shown that the previous games were unreliable and we have provided a statistical explanation for that fact, while providing a possible solution.

We created two new types of statistical methods to recover the discount factor, the bottom-up and top-down methods. Using synthetic data, methods of both types proved to be accurate, despite the high disparity of bottom-up methods. In addition to being accurate, we have also shown that the top-down and some of the bottom-up methods were robust to different synthetic generators, being accurate in a wide range of settings. Using real data, we have shown that the top performing methods in terms of AIC predict consistent values of the discount factor. Moreover, we have shown that the games with three states have implied values of the discount factor that are significantly higher than those of the two-state games.

5.2 Results interpretation and limitations

After having presented the methods and the results, some critical analysis is needed, for not only putting results into context but also to acknowledge where the limitations in our work lie. The main assumption of this project is that there is a single discount factor per person per game. To render the results useful, there is the need to further explore the validity of this assumption. Our analysis regarding the variation of the implied discount factors with time for all subjects seems to indicate that there is no evidence to believe that the implied discount factors vary with time. However, a more thorough analysis of this assumption is needed, since our test is not robust to possible complex patterns of the discount factor. Also, in our analysis, we didn't account for the possible existence of a warm-up period in which the subjects are learning the general structure of the experiment, and thus not making meaningful decisions. This hypothesis should be assessed, possibly by comparing different subsets of the whole sequence and looking for significant differences in the implied discount factor or other indicators of warming up behaviour (such as random behaviour). However, it should be noted that this project was the first step in a totally novel hypothesis, that the discount factor has a strong relation with the task being performed. Therefore, the main objective was to look for evidence that would indicate both the validity and the potential of such a hypothesis. In this setting, the results are truly encouraging, since the trend that was observed was statistically significant. Therefore, having established the interest of this particular question, we are now more capable of testing the underlying hypothesis and of refining the process that led to our results.

Regarding the methods, the results are also encouraging. The ability to extract accurately the discount factor in synthetic data increases our confidence on the results of the experimental

data. However, while the top-down approach is theoretically independent of the generating agent, the bottom-up agents are extremely dependent on the agent that generated the data. Our results have shown that, when the generating and the inferring agents match, the performance is good. However, when they do not match, the results vary considerably. Softmax policies proved to be more capable of adjusting to the generating agent, while ϵ-greedy policies were more susceptible to the generating agent. One possibility for this fact is the inability to capture quasi-certain policies in some states, which are needed in some cases, particularly in non-choice states. Having a fixed ϵ for all states doesn't capture the optimal policy for the games, which may lead to poor performance. Due to the impossibility of testing all possible generating agents, we had use a small selection of them. This means that we cannot be sure that our results are valid outside the scope of the tested agents (e.g. our methods may not work if the generating agent was a different one). Therefore, we cannot be certain that the synthetic results obtained using synthetic data can be extrapolated for the experimental data. Thus, in order to increase the validity of our results, it would be helpful to test a wider spectrum of reinforcement learning agents and policy types. In spite of that fact, it is worth noticing that the predictions of discount factor between the top performing bottom-up methods are consistent for most subjects, even with the top-down method. This increases our confidence that the obtained discount factor has meaning and is not being overly influenced by a possible total mismatch between the policy of human subjects and the set of agents that are used to infer the discount factor.

Finally, we turn to the experiments themselves. The need to create the hybrid experiments arose from the fact that, in the previous setting, we were not accounting for the fact that multiple visits to the variable reward state were needed in order to have an accurate estimate of the probability of receiving a reward. This increases the complexity of the game considerably. One indicator of this is the higher inter-subject variability that can be seen in games A and B when compared to deterministic games 1 and 3, even after our modifications. This can also be seen in the higher errors with synthetic data. However, even with more variability, the differences in the implied discount factor in two and three state games are significant, which strongly points toward the validity of our hypothesis: discount factors depend on the task and are higher in three state games than in equivalent two state games. This interesting result raises several questions, one of them being the reason behind this adaptation. A possible explanation is based on the influence that the value of the discount factor has on the learning process. Higher discount factors increase the ability to distinguish similar values of states (or action-state pairs if a Q-table is being used). This means that, if the values of two states are near one another but are different, a high discount factor will increase the ability to detect that difference. On the other hand, a low discount factor increases the speed of convergence of learning. This trade-off may explain the adaptation of the discount factor to the task, and this hypothesis should be tested in future work.

Given the novelty of the hypothesis, there are no similar studies to compare the results with. As stated in the literature survey, the only study [31] that tries to extract the discount factor assumes that the discount factor does not depend on the length of each alternative path, which seems to be an invalid assumption. It's interesting to note, however, that the games used in that study have an average long path length of 4 with a population-wide value of the discount factor of $\gamma = 0.867$ in the control group. In our work, using the top-down method, the average discount factor for experiments with 2 states was $\gamma = 0.58$ and with 3 states of $\gamma = 0.84$, meaning that their obtained value is nearer to our 3-state games than to our 2-state games, which apparently does not contradict our results. These results cannot be compared directly, however, since neither the structure nor the timings of the experiments match.

5.3 Future Work

One of the main achievements of this project is that it has paved the way for a multitude of possible applications that rely on the estimation of the human discount factor. At the same time, we have shown that it reasonable to think that the discount factor that is used by humans depends on the task at hand. These two achievements lead to some very interesting ideas for future work.

Regarding the recovery methods, we can think of two main areas of improvement: the first is the implementation of the complete Hidden Markov Model method that would allow the simultaneous recovery of the dominant policy and the discount factor, possibly allowing higher likelihoods by maximizing all the parameters using a single model; the second consists of using sampling techniques instead of the current maximum likelihood approach we followed in this report. With regard to the possible applications and research on the human discount factor, the possible future steps include: testing the results with games with more than three states or/and with different combinations of lengths of the short and long paths; inclusion of the time factor in the experiments, by varying the time delay between transitions; testing of exponential vs. hyperbolic discount factors; testing the variation of the discount factor in different population groups, such as sleep deprived subjects, Parkinson's patients or even dopamine and serotonin manipulated subjects.

Bibliography

[1] G Ainslie. Specious reward: a behavioral theory of impulsiveness and impulse control. *Psychological bulletin*, 1975.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974. ISSN 0018-9286. doi: 10.1109/TAC.1974.1100705.

[3] Bernard W Balleine and John P O'Doherty. Human and rodent homologies in action control: corticostriatal determinants of goal-directed and habitual action. *Neuropsychopharmacology : official publication of the American College of Neuropsychopharmacology*, 35(1):48–69, January 2010. ISSN 1740-634X. doi: 10.1038/npp.2009.131.

[4] Hannah M Bayer and Paul W Glimcher. Midbrain dopamine neurons encode a quantitative reward prediction error signal. *Neuron*, 47(1):129–41, July 2005. ISSN 0896-6273. doi: 10.1016/j.neuron.2005.05.020.

[5] Kimberlee D'Ardenne, Samuel M McClure, Leigh E Nystrom, and Jonathan D Cohen. BOLD responses reflecting dopaminergic signals in the human ventral tegmental area. *Science (New York, N.Y.)*, 319(5867):1264–7, February 2008. ISSN 1095-9203. doi: 10.1126/science.1150605.

[6] Nathaniel D Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12): 1704–11, December 2005. ISSN 1097-6256. doi: 10.1038/nn1560.

[7] Nathaniel D Daw, Samuel J Gershman, Ben Seymour, Peter Dayan, and Raymond J Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–15, March 2011. ISSN 1097-4199. doi: 10.1016/j.neuron.2011.02.027.

[8] M. R. Delgado, L. E. Nystrom, C. Fissell, D. C. Noll, and J. A. Fiez. Tracking the Hemodynamic Responses to Reward and Punishment in the Striatum. *J Neurophysiol*, 84(6):3072–3077, December 2000.

[9] R. A. Fisher. *Statistical Methods, Experimental Design, and Scientific Inference: A Re-issue of Statistical Methods for Research Workers, The Design of Experiments, and Statistical Methods and Scientific Inference*. Oxford University Press, USA, 1990. ISBN 0198522290.

[10] Shane Frederick, George Loewenstein, and Ted ODonoghue. Time Discounting and Time Preference: A Critical Review. *Journal of Economic Literature*, 40(2):351–401, June 2002. ISSN 0022-0515. doi: 10.1257/002205102320161311.

[11] Jan Gläscher, Nathaniel Daw, Peter Dayan, and John P O'Doherty. States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, 66(4):585–95, May 2010. ISSN 1097-4199. doi: 10.1016/j.neuron.2010.04.016.

[12] Alan N Hampton, Peter Bossaerts, and John P O'Doherty. Neural correlates of mentalizing-related computations during strategic interactions in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 105(18):6741–6, May 2008. ISSN 1091-6490. doi: 10.1073/pnas.0711099105.

[13] Todd A Hare, John O'Doherty, Colin F Camerer, Wolfram Schultz, and Antonio Rangel. Dissociating the role of the orbitofrontal cortex and the striatum in the computation of goal values and prediction errors. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 28(22):5623–30, May 2008. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.1309-08.2008.

[14] P R Montague, P Dayan, and T J Sejnowski. A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 16(5):1936–47, March 1996. ISSN 0270-6474.

[15] Genela Morris, Alon Nevet, David Arkadir, Eilon Vaadia, and Hagai Bergman. Midbrain dopamine neurons encode decisions for future action. *Nature neuroscience*, 9(8):1057–63, August 2006. ISSN 1097-6256. doi: 10.1038/nn1743.

[16] Keshava Murthy. *Reverse Engineering Parameters of Human Reinforcement Learning for the Diagnosis of Neurodegenerative Diseases*. Meng dissertation, Imperial College London, 2012.

[17] Douglas Navarick. Discounting Of Delayed Reinforcers: Measurement By Questionnaires Versus Operant Choice Procedures, 2004.

[18] John P O'Doherty. Reward representations and reward-related learning in the human brain: insights from neuroimaging. *Current opinion in neurobiology*, 14(6):769–76, December 2004. ISSN 0959-4388. doi: 10.1016/j.conb.2004.10.016.

[19] John P O'Doherty, Peter Dayan, Karl Friston, Hugo Critchley, and Raymond J Dolan. Temporal difference models and reward-related learning in the human brain. *Neuron*, 38(2):329–37, April 2003. ISSN 0896-6273.

[20] John P O'Doherty, Alan Hampton, and Hackjin Kim. Model-based fMRI and its application to reward learning and decision making. *Annals of the New York Academy of Sciences*, 1104: 35–53, May 2007. ISSN 0077-8923. doi: 10.1196/annals.1390.022.

[21] Antonio Rangel, Colin Camerer, and P Read Montague. A framework for studying the neurobiology of value-based decision making. *Nature reviews. Neuroscience*, 9(7):545–56, July 2008. ISSN 1471-0048. doi: 10.1038/nrn2357.

[22] Brady Reynolds and Ryan Schiffbauer. Measuring state changes in human delay discounting: an experiential discounting task. *Behavioural processes*, 67(3):343–56, November 2004. ISSN 0376-6357. doi: 10.1016/j.beproc.2004.06.003.

[23] ML Rodriguez and AW Logue. Adjusting delay to reinforcement: Comparing choice in pigeons and humans. *Journal of Experimental Psychology: ...*, 1988.

[24] Matthew R Roesch, Adam R Taylor, and Geoffrey Schoenbaum. Encoding of time-discounted rewards in orbitofrontal cortex is independent of value representation. *Neuron*, 51(4):509–20, August 2006. ISSN 0896-6273. doi: 10.1016/j.neuron.2006.06.027.

[25] Matthew R Roesch, Donna J Calu, and Geoffrey Schoenbaum. Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nature neuroscience*, 10(12):1615–24, December 2007. ISSN 1097-6256. doi: 10.1038/nn2013.

[26] Yutaka Sakai and Tomoki Fukai. The actor-critic learning is behind the matching law: matching versus optimal behaviors. *Neural computation*, 20(1):227–51, January 2008. ISSN 0899-7667. doi: 10.1162/neco.2008.20.1.227.

[27] Kazuyuki Samejima, Yasumasa Ueda, Kenji Doya, and Minoru Kimura. Representation of action-specific reward values in the striatum. *Science (New York, N.Y.)*, 310(5752):1337–40, November 2005. ISSN 1095-9203. doi: 10.1126/science.1115270.

[28] Paul A. Samuelson. A Note on Measurement of Utility. *The Review of Economic Studies*, 4 (2):155, February 1937. ISSN 00346527. doi: 10.2307/2967612.

[29] W Schultz, P Dayan, and P R Montague. A neural substrate of prediction and reward. *Science (New York, N.Y.)*, 275(5306):1593–9, March 1997. ISSN 0036-8075.

[30] N Schweighofer, K Shishida, C E Han, Y Okamoto, S C Tanaka, S Yamawaki, and K Doya. Humans can adopt optimal discounting strategy under real-time constraints. *PLoS computational biology*, 2(11):e152, November 2006. ISSN 1553-7358. doi: 10.1371/journal.pcbi.0020152.

[31] Nicolas Schweighofer, Mathieu Bertin, Kazuhiro Shishida, Yasumasa Okamoto, Saori C Tanaka, Shigeto Yamawaki, and Kenji Doya. Low-serotonin levels increase delayed reward discounting in humans. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 28(17):4528–32, April 2008. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.4982-07.2008.

[32] M Shidara, T G Aigner, and B J Richmond. Neuronal signals in the monkey ventral striatum related to progress through a predictable series of trials. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(7):2613–25, April 1998. ISSN 0270-6474.

[33] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. A Bradford Book, 1998. ISBN 0262193981.

[34] RS Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 1988.

[35] Edward L. Thorndike. *Animal intelligence; experimental studies, by Edward L. Thorndike.* The Macmillan company„ New York„ 1911. doi: 10.5962/bhl.title.1201.

[36] Edward C. Tolman. Cognitive maps in rats and men. 1948.

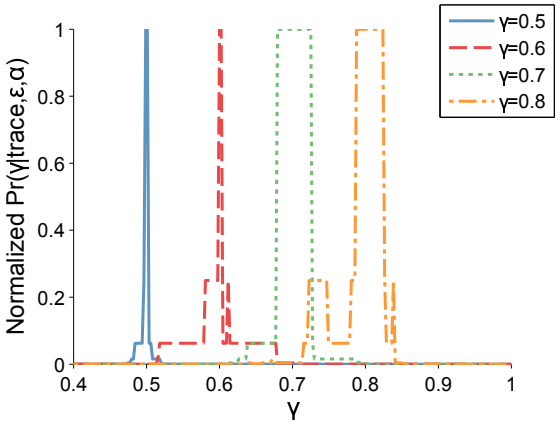
[37] Satoshi Tsujimoto, Aldo Genovesio, and Steven P Wise. Neuronal activity during a cued strategy task: comparison of dorsolateral, orbital, and polar prefrontal cortex. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 32(32):11017–31, August 2012. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.1230-12.2012.

[38] Vivian V Valentin, Anthony Dickinson, and John P O'Doherty. Determining the neural substrates of goal-directed learning in the human brain. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 27(15):4019–26, April 2007. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.0564-07.2007.

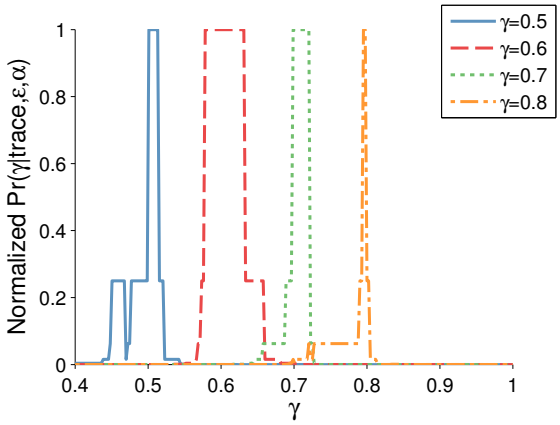
[39] CJCH Watkins. Learning from delayed rewards. 1989.

Appendix A

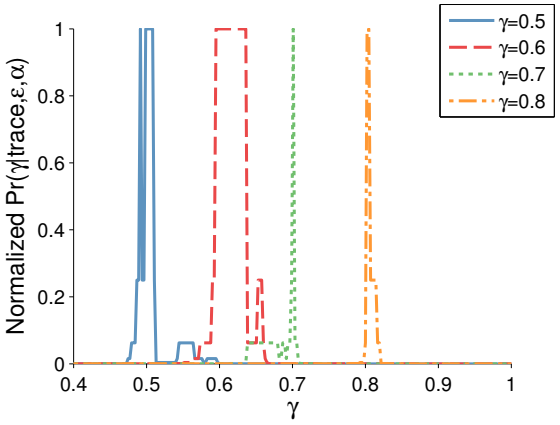
Local parameter analysis



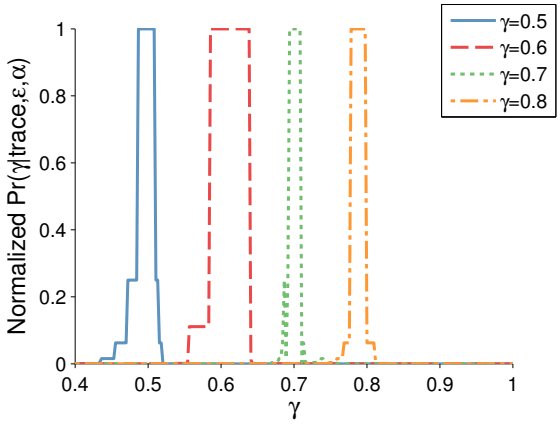
(a) Game 1



(b) Game 3

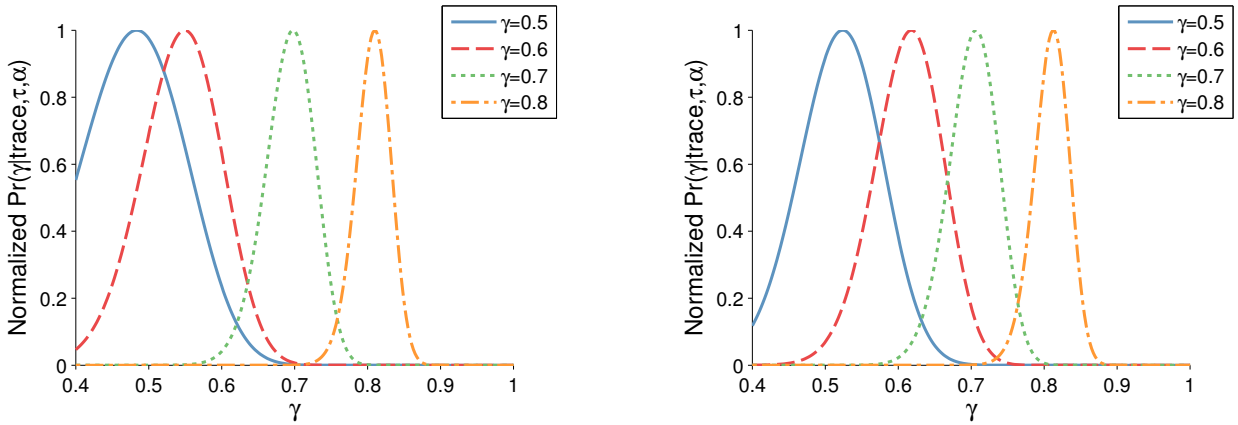


(c) Game A



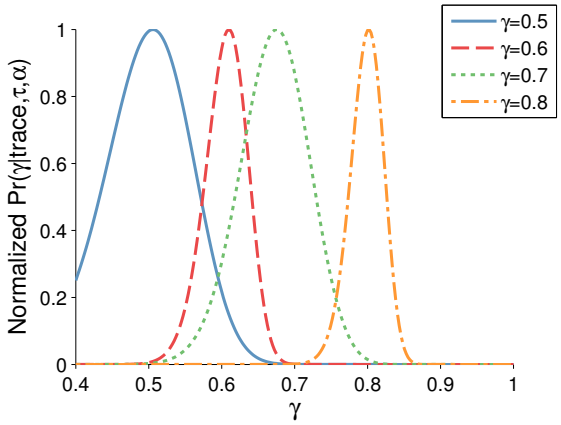
(d) Game B

Figure A.1: Normalized probability of γ matching the value of γ that generated a given trace. For each game we generated 4 traces with a SARSA learner with an ϵ-greedy policy, with $\epsilon = 0.2$, $\alpha = 0.4$ and $\gamma = \{0.5, 0.6, 0.7, 0.8\}$. Note that the maximum for each value of generating γ lies within less than 0.05 of the true value. The results were normalized so that the maximum for each generating parameter would be 1.

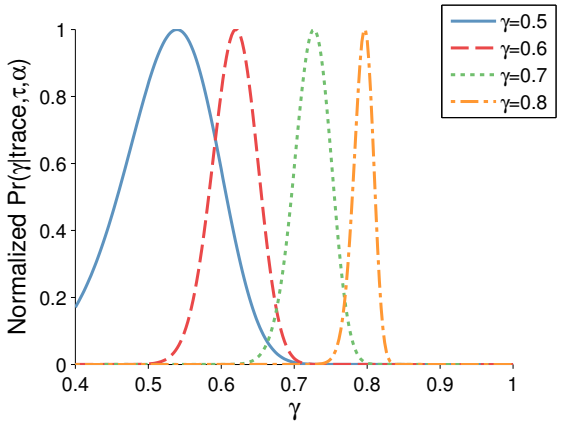


(a) Game 1

(b) Game 3

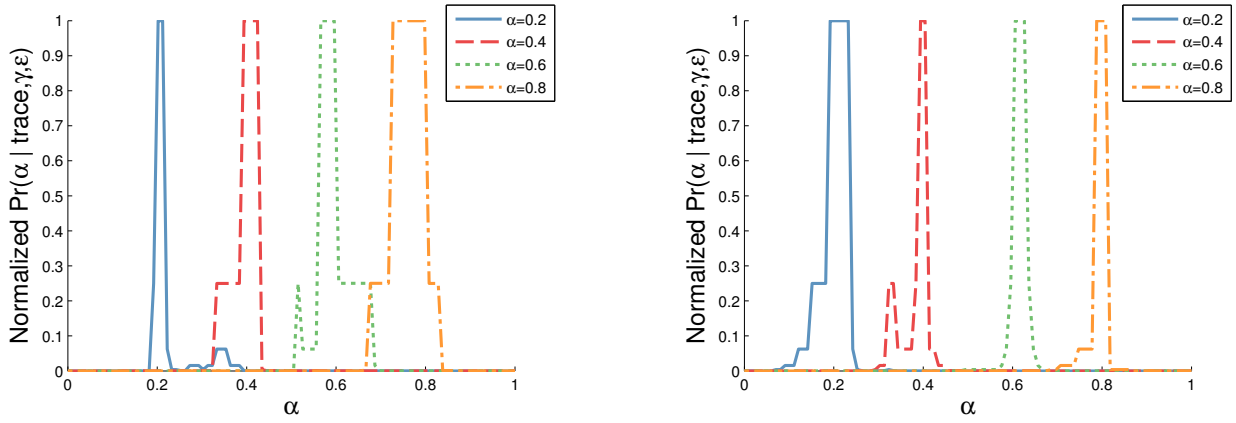


(c) Game A



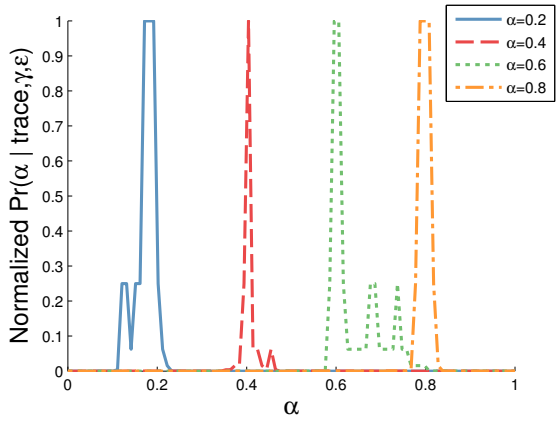
(d) Game B

Figure A.2: Normalized probability of γ matching the value of γ that generated a given trace. For each game we generated 4 traces with a SARSA learner with an softmax policy, with $\tau = 25$ for games 1 and 3 and$\tau = 0.25$ for games A and B, $\alpha = 0.4$ and $\gamma = \{0.5, 0.6, 0.7, 0.8\}$. Note that the maximum for each value of generating γ lies within less than 0.05 of the true value. The results were normalized so that the maximum for each generating parameter would be 1.

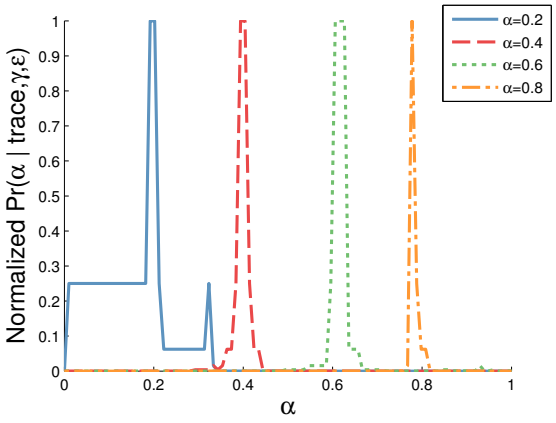


(a) Game 1

(b) Game 3

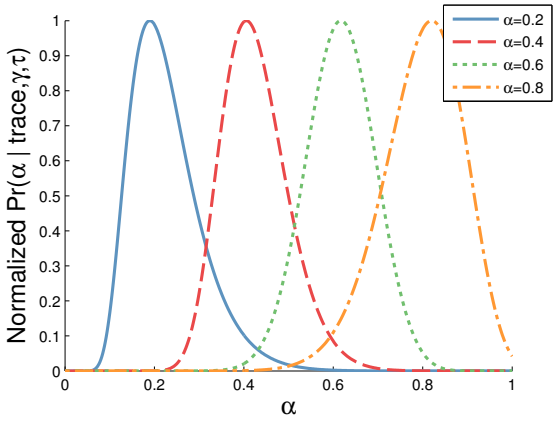


(c) Game A

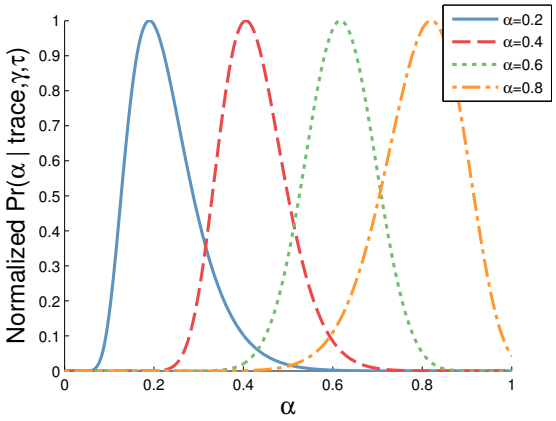


(d) Game B

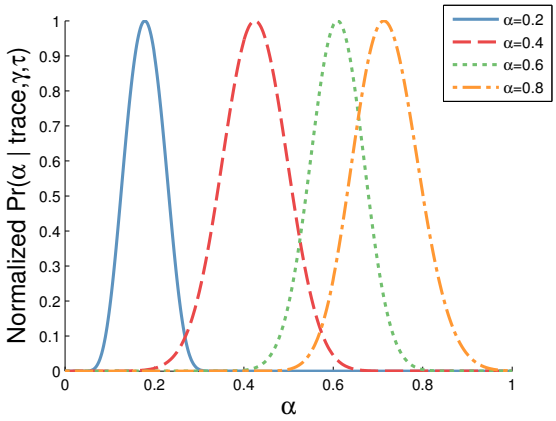
Figure A.3: Normalized probability of α matching the value of α that generated a given trace. For each game we generated 4 traces with a SARSA learner with an ϵ-greedy policy, with $\epsilon = 0.2$, $\alpha = \{0.2, 0.4, 0.6, 0.8\}$ and $\gamma = 0.7$. Note that the maximum for each value of generating α lies within less than 0.05 of the true value. The results were normalized so that the maximum for each generating parameter would be 1.



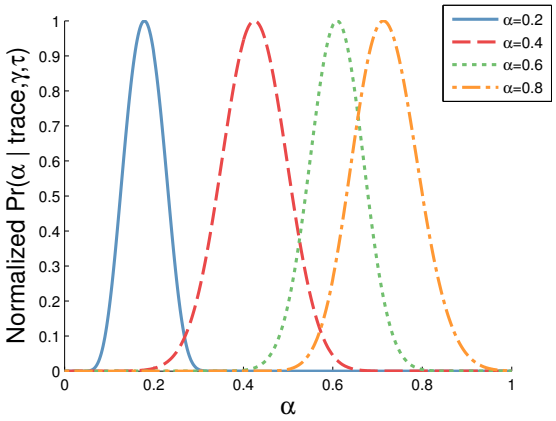
(a) Game 1



(b) Game 3

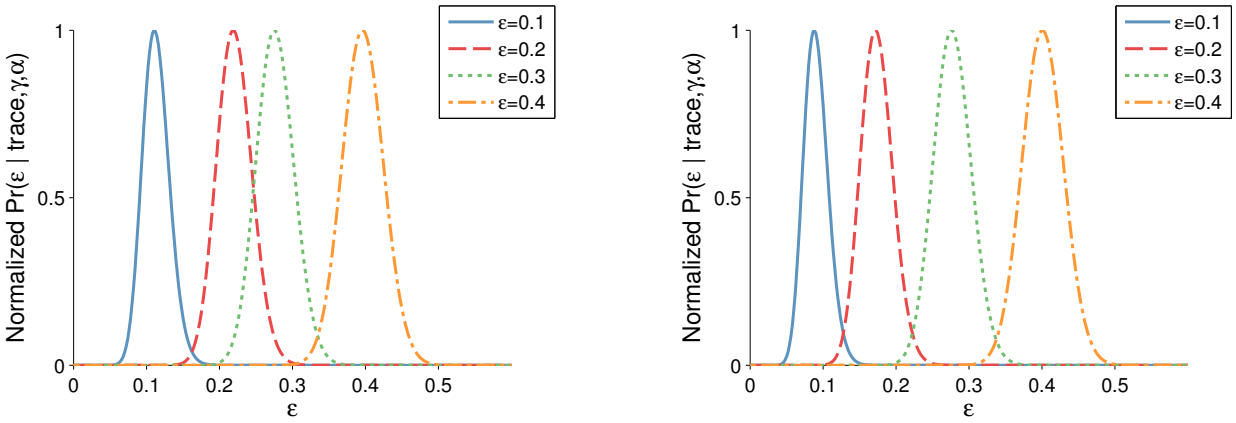


(c) Game A



(d) Game B

Figure A.4: Normalized probability of α matching the value of α that generated a given trace. For each game we generated 4 traces with a SARSA learner with an softmax policy, with $\tau = 25$ for games 1 and 3 and$\tau = 0.25$ for games A and B, $\alpha = \{0.2, 0.4, 0.6, 0.8\}$ and $\gamma = 0.7$. Note that the maximum for each value of generating α lies within less than 0.05 of the true value. The results were normalized so that the maximum for each generating parameter would be 1.



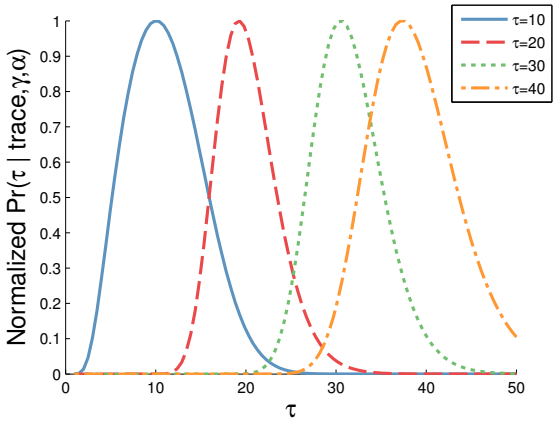
(a) Game 1

(b) Game 3

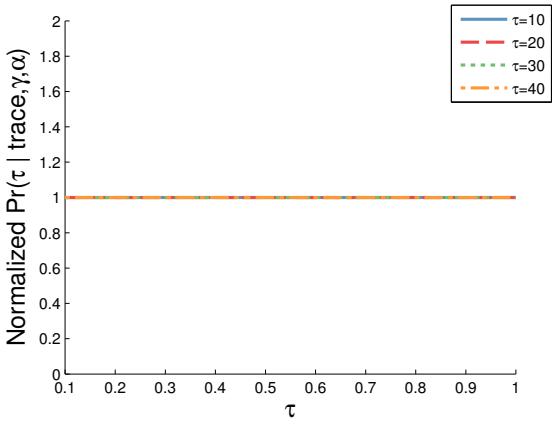
(c) Game A

(d) Game B

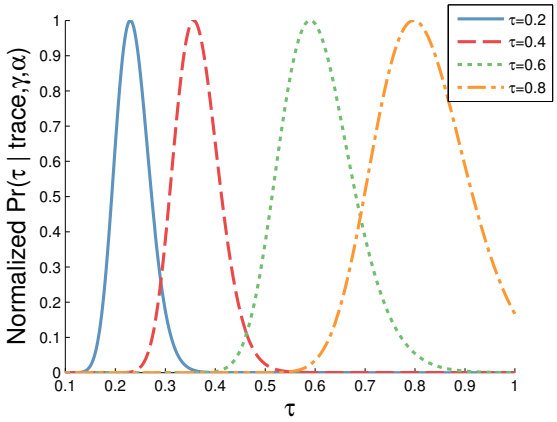
Figure A.5: Normalized probability of ϵ matching the value of ϵ that generated a given trace. For each game we generated 4 traces with a SARSA learner with an ϵ-greedy policy, with $\epsilon = \{0.1, 0.2, 0.3, 0.4\}$, $\alpha = 0.4$ and $\gamma = 0.7$. Note that the maximum for each value of generating ϵ lies within less than 0.05 of the true value. The results were normalized so that the maximum for each generating parameter would be 1.



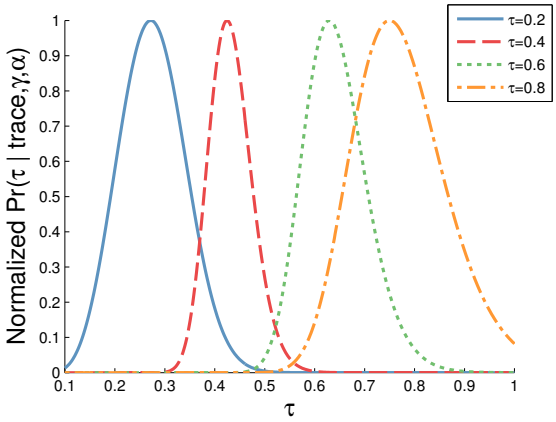
(a) Game 1



(b) Game 3



(c) Game A



(d) Game B

Figure A.6: Normalized probability of τ matching the value of τ that generated a given trace. For each game we generated 4 traces with a SARSA learner with an softmax policy, with $\tau = \{10, 20, 30, 40\}$ for games 1 and 3 and$\tau = \{0.2, 0.4, 0.6, 0.8\}$ for games A and B, $\alpha = 0.4$ and $\gamma = 0.7$. Note that the maximum for each value of generating τ lies within less than 5 of the true value for games 1 and 3 and 0.05 for games A and B. The results were normalized so that the maximum for each generating parameter would be 1.

Appendix B

Experiments

Eleven healthy, right-handed male volunteers, with no history of psychiatric or neurological disorders, gave written informed consent after the nature and possible consequences of the study were explained. The study was done in accordance with the institutional and national regulations involving human experiments.

Before the beginning of the experiments, they were given an instruction sheet that explained the experiments and the rules. The text used is transcribed below.

> "Your aim in this game is to maximise your reward over the entire session of the game, by finding and choosing the best action to take from each state at each time.
> The structure of the game is as follows:
>
> 1. The game is played as a series of rounds.
> 2. You begin each round at a given state.
> 3. You then choose one of two actions, which causes you to transition to the next state, and this may give you a reward (you are told how much).
> 4. You repeat steps 1 to 3 until the game ends.
>
> In each round, the state you are in is represented by a texture on the screen. On choosing an action, the state to which you transition and the consequent reward both depend on the starting state and action. A transition may leave you in the same state. The properties of the game may change slowly/irregularly as you play. You choose an action by pressing the labelled keyboard keys. After every action, there will be a brief period where you cannot make a move. In this time, your last move and the current state will be highlighted and the reward you obtained is displayed. If you fail to make your move within 2 seconds, a random move will be made for you and shown to you in the same manner.
> There are a total of two games for you to play. Each game will last roughly 15 minutes (this will vary independently of you). Between games, you will be given a 5 minute break to relax. "

After the experiments, the subjects were asked to answer a questionnaire verbally, which is also transcribed below.

1. Which of the two games was the hardest?
2. In which game was it possible to get rewards more quickly?
3. How many times did the properties of the game change in each game?
4. What properties of the game changed? (The rewards associated with each action/the states that followed certain actions/both)
5. Did these changes happen in all states or just in some?
6. Did you think your score was worse than average, average or better than the average?
7. Comments