University of Warsaw

Faculty of Philosophy and Sociology

Katarzyna Oktaba

333432

Learning to Signal: Socio-environmental Constraints in Category Formation

First cycle degree thesis

Field of study: Cognitive Science

Specialty Linguistics

The thesis written under the supervision of

dr Dariusz Kalociński

Institute of Philosophy

Warsaw 2017

Statement of the Supervisor on Submission of the Thesis

I hereby certify that the thesis submitted was prepared under my supervision and I declare that it satisfies the requirements of submission in the proceedings for the award of a degree.

Date Signature of the Supervisor:

Statement of the Author(s) on Submission of the Thesis

Aware of the legal liability I certify that the thesis submitted was prepared by myself and does not include information gathered contrary to the law.

I also declare that the thesis submitted has not been a subject of proceedings resulting in the award of a university degree.

Furthermore I certify that the submitted version of the thesis is identical with its attached electronic version.

Date Signature of the Author(s) of the Thesis:

Summary

The purpose of this thesis is to analyze the impact of social and environmental factors in the process of learning categories modeled within signaling games' framework. The proposed model is based on Lewis signaling systems, but has been modified to accommodate categories formation. Two reinforcement learning algorithms have been adjusted and implemented to study the game dynamics in different scenarios with varying impact of environmental pressure from pure coordination to a predominant environmental factor. The model addresses the discussion raised in literature regarding the simultaneous impact of social coordination and pressure for expressiveness on the language formation.

Celem niniejszej pracy jest analiza wpływu czynników społecznych i środowiskowych na proces uczenia się kategorii w ramach gier sygnałowych. Prezentowany model opiera się na systemach sygnałowych Lewisa, został jednak zmodyfikowany, aby uwzględnić tworzenie się kategorii. W pracy wykorzystano dwa algorytmy uczenia ze wzmocnieniem, które, po odpowiednich modyfikacjach, pozwoliły na rozważanie scenariuszy z różnym wpływem czynników środowiskowych: od czystej koordynacji po dominujący wpływ środowiska. Model odpowiada na dyskusję poruszoną w literaturze na temat jednoczesnego wpływu presji społecznej i środowiskowej na tworzenie się języka.

Keywords

Signaling games, categories formation, reinforcement learning, language evolution

Gry sygnałowe, kategoryzacja, uczenie ze wzmocnieniem, ewolucja języka

Area of study (codes according to Erasmus Subject Area Codes List)

14.4 Psychology and Behavioural Sciences

The title of the thesis in Polish

Uczenie się sygnałów: ograniczenia społeczne i środowiskowe w tworzeniu się kategorii

Contents

1 Introduction

The learning of categories is a topic overlapping many problems long present in philosophy, linguistics and cognitive science: induction, word meaning, pragmatics, cognitive complexity. This line of research aims at providing an empirically adequate and theoretically sound explanation of how initially unlabeled objects are divided into conventionally named groups. While it can be studied from various angles, in this thesis I focus on a model of category learning in language games. One of the aims of my current approach is to shed some light on the questions raised in a recent discussion concerning the influence of social and environmental pressures on language by Wallentin and Frith (2008).

I propose a model of the coordinate learning of categories, being an extension to Lewis' signaling games first introduced in Lewis (1969). Lewis' game is based on a one-way exchange between the sender and the receiver, who have to come up with a communication system to successfully coordinate information with an action. While the proposed model is expressed within the same formal framework, it introduces some novel concepts to address new research questions.

First modification introduced in this thesis lies in increasing the ratio of the number of stimuli to the number of available signals. In the standard model, there is exactly as many stimuli[1] as there are language terms to name them. This setting allows for separating equilibria, in which there is one and only one term for each stimulus. In the proposed model, there are less language terms than stimuli and separating equilibria are not possible. This modification shows the adequacy of the signaling games' framework to model the emergence of categories in natural language.

Second conceptual modification concerns the factors determining the players' behavior and it directly addresses the considerations raised in Wallentin and Frith (2008) and the following response in Christiansen and Chater (2008*a*), where the varying impact of environmental and social factors in the language formation is considered. In Lewis' model the usage of the language terms is purely conventional; however, the choice is

[1] One may note that in signaling games a word 'stimulus' is rarely used and what is experienced by the receiver is commonly called 'a state of the world' or 'a type'. However, the purpose of this chapter is to offer a coherent overview of the main concepts present in this thesis. Formal terminology will be introduced with due precision in the following chapters.

driven by the pressure for expressiveness, induced by the externally defined correspondence between stimuli and responses, which players aim at conveying in language. In the proposed model, in its basic variant there is no external factor and players solely try to coordinate on a common strategy with all shared strategies being equally good[2]. Next, I add an external factor: natural groupings, which players are rewarded to agree with in their private categorizations. Social (coordinate) and environmental (external) factors are independent, which makes possible both congruent and incongruent scenarios, i.e. ones in which natural and coordinated categorizations overlap, and ones in which they differ and players receive conflicting reinforcements. The weight, which the players put on each factor, can be parametrized to allow for scenarios with marginal pressure for expressiveness and more emphasis put on coordination as well as opposite cases. The intuition is that in the natural language the role of each factor is not uniform as some expressions, e.g. vital for survival, require high expressiveness, while others, e.g. abstract terms, are shaped primarily by social convention.

The proposed approach is implemented and analyzed based on simulations. The experiment tests the model properties using two reinforcement learning algorithms: Herrstein reinforcement learning (HRL) and smoothed reinforcement learning (SRL) with different parameterizations. For each test case at each point in the game I calculate a success rate to evaluate and compare scenarios. Analysis and graphical presentation of the game's dynamics show differences between scenarios, suggesting the impact of the game's protocol (one-way vs two-way communication), the learning algorithms and the presence of external factor on the pace of convergence of private categorizations to a common classification. The main observation is that independent social and environmental factors can be reconciled as long as the players have limited memory. At the beginning, the players most likely receive conflicting reinforcements and those are easier to overcome when only most recent evidence is used for learning as initial contradictory evidence is not slowing down the learning process.

[2]It is important to note that this coordinate approach requires (unlike the standard model) a two-way communication, since one-way exchange would reduce the task to supervised learning of categories rather than the coordination between equal players. One-way scenario is also tested as a benchmark case as it is predictably 'easier' than the coordination task.

At the end of the thesis I summarize the presented study, underlining its limitations as well as potential future research questions. Another summary of the presented results can be found in Oktaba and Kalociński (2018), which will appear in 'The Evolution of Language: Proceedings of the 12th International Conference'.

2 Lewis Signaling Games

Signaling games are a family of models, originating from game theory and economics, which have been extensively used in linguistic modeling. In a basic scenario, it is a two-player game, in which the sender transmits information to the receiver via a signal, the meaning of which is established only through repeated interactions. In this sense, signaling games can be thought of as a formalization of Wittgenstein's design of a language game and have considerably contributed to developments in linguistic modeling.

2.1 Linguistic interpretation

The philosophical concept of learning language through communication dates back to Wittgenstein's idea of a language game, introduced in Wittgenstein (1953). There, he describes a scenario, in which one person (a builder) calls some objects out and the second participant (an assistant) reacts with fetching the objects. The communication is successful, if the builder receives the object he currently needs. Since they are coordinating on their work, their common purpose is to establish a successful communication system (understood as a relation between signals and actions).

While Wittgenstein described the language game as a philosophical concept, game theory, a vast research domain dedicated to studying games in mathematical terms, provides a corresponding formal framework. Signaling games, described in detail e.g. in Sobel (2007), are a class of games of incomplete information, in which some of the players are informed of something that the others are not. The information is transmitted through a signal only. In the two-player setting, there is one sender (the informed player) and one receiver (the player that reacts to the signal). The sender, who is informed about the current state of the world, sends a signal to the receiver, who ignores the state of the world, but must take an action. Players are rewarded, when the action corresponds to the current state.

The scope of applications of signaling games by far exceeds the linguistic domain. It is a flexible framework, which has been used in economics (advertising, bargaining, politics; see Sobel (2007), biology (nestling beggings; see Bergstrom and Lachmann (1998)), and includes costly signaling (see e.g. Gintis et al. (2001)), non-cooperative games (e.g.

Hespanha et al. (2000)) and strategic modeling of honesty (e.g. Polnaszek and Stephens (2013)).

Signaling games gained a linguistic interpretation mainly thanks to the work of Lewis (Lewis (1969)). Lewis' model focuses on the process of the assignment of signals to the states of the world and actions, and in this sense the framework can be used for studies on the emergence of a meaningful signaling system. Also, it is easy to translate Wittgenstein's scenario into this framework as the builder (the sender) is informed about the current state of the building process (the state of the world) and sends a signal to the assistant (the receiver), who must fetch the necessary object (take an action). Lewis signaling games have been further used to explore the dynamics of communicative systems. This line of research has been continued and extended to evolutionary games (see e.g. Huttegger et al. (2010)), scenarios with multiple senders and receivers (see Skyrms (2009)), implementation of various learning algorithms (see Barrett and Zollman (2009) for an extensive study on the role of limited memory), among others.

2.2 Lewis' signaling systems

A signaling game is defined by:

- finite set S of senders (i.e. informed players),
- finite set R of receivers (i.e. uninformed players),
- finite[3] set W of types[4] of a sender, i.e. world states that the sender may be informed of,
- probability distribution $\pi(\cdot)$ over W, defining the probability of a world state $w \in W$ being true,
- finite set C of signals,

[3]Infinite set of all possible world states is formally possible, yet remains beyond the scope of this thesis. A literature review suggests it is a prevalent assumption for linguistic modeling.

[4]The name "type" is frequently used in the economic and game-theoretic literature on signaling games, while philosophical and linguistic researches use the term "a world state" more often. I will follow the convention of "a world state", which should be understood as equivalent to "a type" in standard signaling games' literature.

- finite set A of actions,
- utility function $u : (W \times A) \to \mathbb{R}$.

A basic signaling game is defined for one sender and one receiver ($|S| = |R| = 1$).

A strategy of a player i will be denoted x_i, while a profile of strategies for all players except player i will be denoted $x_{-i} = (x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$ for a game with n players in total. A pure strategy is a function, which defines, for each particular player, her choices given the choices of all other players. Thus, a pure strategy for the sender is a function $x_S : W \to C$ and a pure strategy for the receiver is a function $x_R : C \to A$. A mixed strategy is a probability distribution on pure strategies that are available to the player. A strategy profile is a tuple of the strategies for all players, $\mathbf{x} = (x_1, x_2, ..., x_n)$.

Signaling games as described in Lewis (1969) model the behavior of a meaningful communication system. For this reason, the most basic model[5] is a two-player game, in which there is one sender, $|S| = 1$, one receiver, $|R| = 1$, and exactly N possible states of the world, N available signals and N possible actions, $|W| = |C| = |A| = N$. The pay-off matrix M^*, which defines a pay-off for each pair (w, a) such that $w \in W, a \in A$, is diagonal with

$$\forall i \in \{1...N\}\ m^*_{ii} = 1$$

and

$$\forall i, j \in \{1...N\}^2, i \neq j\ m^*_{ij} = 0.$$

It is worth noticing that matrix M^* is nonnegative, i.e. players do not risk a negative pay-off.

Equivalently, pay-off matrix can be obtained using strategy matrices for both players. Sender's strategy matrix M^S can be defined as a mapping between the world states $w \in W$ and sent signals $c \in C$, for which $m^S_{ij} = 1, i, j = 1, ..., N$ means that the sender sends signal j, when being in the world's state i. Analogously, receiver's strategy matrix M^R can be defined as a mapping between the received signals and possible actions, for which

[5] Skyrms (2009) offers some modifications to this model, e.g. a game with multiple senders or multiple receivers, chains of communicating agents or a dialogue. However, the two-player setting is prevalent in the literature and already provides some valuable insight into the problem.

$m^R_{ij} = 1, i, j = 1, ..., N$ means that the receiver chooses action j, having received the signal i. A pay-off matrix is then defined as $M = M^R M^S$.

Strategy pairs that lead to the pay-off matrix M^* are called signaling systems as they represent the maximal, one-to-one coordination of the players.

2.3 Learning algorithms

Games with multiple rounds or played repeatedly allow the players to adjust their strategies to increase the expected pay-off. Signaling games is a framework compatible with many learning algorithms, each of which is based on different assumptions and results in different game dynamics.

2.3.1 Herrnstein Reinforcement Learning

Herrnstein Reinforcement Learning (HRL for short) is one of the most basic learning models, which often serves as a benchmark for more sophisticated models. The algorithm was originally introduced in Roth and Erev (1995); here described as in Barrett and Zollman (2009). The idea behind the algorithm is that in each round the players act according to their propensities proportional to accumulated awards for each strategy. While propensities are related to probabilities, propensity is not formally a probability (including values larger than 1) and must be converted into a probability measure.

Barrett and Zollman (2009) make a distinction between strategy-based and act-based learning. In act-based learning, players update their propensities only with respect to the state/signal observed, while in strategy-based learning players update complete contingency plans. In this thesis, following the approach in Barrett and Zollman (2009), act-based version is assumed.

Altogether, the algorithm is defined by:

1. the initial propensities,
2. the response rule,
3. the updating rule.

The initial propensities $q_i(0)$ are the weights of each strategy $j \in \{1, ..., J\}$ conditional on the state/signal i at the starting point $t = 0$. It is usually assumed that players begin

with equal propensities for each strategy, that is they have no prior knowledge relevant to the game. **The response rule** converts propensities into probabilities. Herrnstein Reinforcement Learning in its basic form assumes proportional response rule, that is the probability of action j having observed state/signal i at time t is equal to

$$p_{i,j}(t) = \frac{q_{i,j}(t)}{\sum_k q_{i,k}(t)},$$

where k are all possible actions. **The updating rule** defines the dynamics of the game by determining the update of propensities after each round. For Herrnstein Reinforcement Learning the updating rule is defined recursively as

$$q_{i,j}(t+1) = \begin{cases} q_{i,j}(t) + \pi(t) \text{ if action } j \text{ was taken having observed state/signal } i, \\ q_{i,j}(t) \text{ otherwise,} \end{cases}$$

where $\pi(t)$ is a pay-off received at time t, i.e. an element m^*_{ij} of the payoff matrix M^*.

Barrett and Zollman (2009) offer an intuitive and graphic explanation of the idea of Herrnstein Reinforcement Learning, describing it as an urn process. Let us imagine that the sender is given N urns, labeled with the states of the world. In the urns there are balls, each of which is labeled with one signal. At the beginning in each urn there is an equal proportion of balls for each signal. During the game, the sender observes the current state of the world and reaches to the urn labeled with this state. She draws at random one ball, sends the signal represented on the ball and returns the ball to the urn. If the communication is successful, she adds to the urn one additional ball with this signal, thus increasing the probability of choosing it the next time this state of the world occurs. Analogous procedure is carried out by the receiver, for whom the urns are labeled by signals and balls correspond to particular actions. Upon receiving a signal, the receiver reaches to the corresponding urn and chooses an action by drawing randomly a ball. Successful communication adds a copy of the used ball to the urn.

2.3.2 Smoothed Reinforcement Learning

Smoothed Reinforcement Learning (SRL) is one of the models, which limit the memory of the learning agent. However, instead of simply cutting off past observations above some memory limit, SRL gradually reduces the importance of historical observations.

This model has two additional parameters: δ and λ, which affect the updating rule and the response rule respectively.

The initial properties remain equal as in Herrnstein Reinforcement Learning. **The response rule** is defined as

$$p_{i,j}(t) = \frac{e^{\lambda q_{i,j}(t)}}{\sum_k e^{\lambda q_{i,k}(t)}},$$

where $\lambda > 0$ is a parameter, which determines to what extent small changes in propensities impact probabilities. The higher λ, the more small differences in propensities result in greater differences in probabilities. **The updating rule** is defined as

$$q_{i,j}(t+1) = \begin{cases} (1-\delta)q_{i,j}(t) + \delta\pi(t) \text{ if action } j \text{ was taken having observed state/signal } i, \\ q_{i,j}(t) \text{ otherwise,} \end{cases}$$

where δ is a parameter, which determines the impact of latest versus past observations. δ ranges from 0 to 1, where $\delta = 1$ means that the player takes into account only observations from the last round, while $\delta = 0$ implies that she does not update her propensities whatsoever.

All in all, large λ and δ define a player, who takes into account mostly recent observations and each observation has a significant impact on the probability of her action. On the contrary, low δ and λ resemble the basic Herrnstein Reinforcement Learning with more emphasis put on accumulated propensities and close to linear response rule (see e.g. Barrett and Zollman (2009) for a more detailed analysis of the impact of parameters' values on the probabilities).

3 Model

Proposed model uses the signaling games framework to analyze the behavior of the players forced to coordinate their actions. In the basic variant, the players try to establish a common categorization through a two-way communication, where the only factor driving their choices is coordination and social agreement. Then, an external factor is added to represent a pressure for expressiveness as defined in Christiansen and Chater (2008*b*). The main point of interest is the impact of the interaction of two, potentially conflicting, forces on the successfulness of the communication and learning process.

3.1 Description

The model describes the situation of the consecutive one-way communication acts, in which in every exchange there is one player, who is informed and sends the signal, and one player, who receives the signal and takes an appropriate action. The players switch roles within every round. Given a common set of objects and a dictionary, each player privately classifies the objects and labels them with terms in the dictionary. However, there is no shared assignment of objects to the terms and players cannot communicate outside of the game's protocol.

One player starts as the sender and the second one as the receiver. The sender *has something in mind*, which is the information the receiver has no access to[6]. The sender informs the receiver of the kind of the object of interest using a term in the dictionary. Since there are no proper names available, she chooses the name of the corresponding class of objects (the class to which the object belongs to according to her private classification). The receiver is then supposed to react by providing an example of the object of this class. If the sender considers the reply valid (i.e. the example provided belongs to the meaning of the term the sender used), the receiver is awarded a pay-off of 1. She receives no pay-off otherwise. Then players switch roles and repeat the described protocol. A round ends once exchanges in both way have been carried out. After each round players adjust their categorizations, i.e. rearrange the associations between signals and objects.

[6]While "having something in mind" may suggest no external force, it is easy to externalize it by a random assignment of a current object of interest (which corresponds to e.g. externally evoked needs, which direct the agent's thoughts to a particular object).

Intuitively, the described model reflects e.g. the situation, in which one needs *any* example of the particular type of object (e.g. a tool) and has to ask for it, or a discussion is being held and the interlocutor is supposed to provide her own relevant example or argument. In any of those cases there is no uniquely correct reaction, rather a class of equivalent reactions (all are equally good as long as they are relevant and legitimate from the point of view of the sender). If the reaction suggests that the receiver understands a language term differently, the communication has failed.

It is worth underlining that the external assignment of the point of interest does not equal an external influence on the process of categorization. The players are still free to use whatever signals they find relevant and the only reinforcements they receive are the points awarded by the other player (i.e. social coordination). Environmental factors impact the term the player uses by favoring one mappings over the other.

3.1.1 Simplifying assumptions

A few assumptions are made to limit the model's complexity.

1. The signals are given and the players coordinate purely on the semantics, not on the syntax.

2. The classes of objects (both natural groupings and players' private categorizations), which at each moment in time[7] constitute the meaning of the terms, are mutually exclusive and collectively exhaustive. If there are only two terms, then each group is the complement of the other.

3. Players learn only after communication is carried out both ways in one round. It ensures perfect symmetry between players.

3.2 Game's Definition in Signaling Games Framework

The game described in section 3.1 can be defined in terms of a signaling game. Let there be two senders $S = \{s_1, s_2\}$ and two receivers $R = \{r_1, r_2\}$, where $s_1 = r_1$ and $s_2 = r_2$, i.e. each player has two roles. The states of the world W are objects F, $W = F$. The state of the world $f \in F$ means that the sender thinks of an object f. The set of actions is equal

[7] It is important to underline that the players' categorization change probabilistically between rounds.

to the set of states, $A = W = F$, where choosing action $f' \in F$ is understood as "sending back object f' to the sender". The cardinality of the set of signals C (the dictionary) is strictly smaller than the cardinality of set F, that is $|C| < |F|$. The game proceeds as in the standard signaling game model: the sender observes a state of the world (here: thinks of an object $f \in F$), sends a signal $c_S \in C$ and the receiver responds by an action (here: sends back an object f' from the set F). Pay-off matrix is not fixed, but rather changes from one round to another. Additionally, each player has a split (a grouping), that is a function $g : F \to C$, which assigns to each object a label from the dictionary. Pay-off is calculated using the reward function r, which is defined as:

$$r(c_S, g_S, f') = \begin{cases} 1 \text{ if } \ f' \in g_S^{-1}(c_S), \\ 0 \text{ otherwise,} \end{cases}$$

where c_S is a sent signal, g_S is the split of the sender and f' is the object that the receiver sent back in this exchange.

Informally, the state of the world is the object that the sender has in mind. Based on it, she sends a signal, corresponding to the category to which she classifies this object. The receiver responds by providing the object she finds relevant to the signal. The pay-off is determined by the sender according to her current split.

As can be seen, the payoff and thus the players' choices depend only on the coordination with the interlocutor. The model reflects only the social factors driving the categories formation and does not take into account any external associations, expressiveness, etc. The purpose is to study the learning process *as if* the players' only goal was a shared linguistic convention with no regard to natural factors.

3.3 Modification for external factor

The model described in Chapter 3.1 is based purely on coordination between players and there is nothing "in the world" that would favor any split over another. However, it is not difficult to find numerous, real-world examples to prove that at least in some cases some categorizations make more sense, to put it informally, than others. Sometimes, this external factor is significant (e.g. evolution certainly favors splits that allow us to distinguish between edible and inedible plants), while in the case of more abstract terms

their meaning can be more conventional. To account for such external factor with varying degree of importance, a modification to the model is proposed.

Coordination game with two-way communication and external factor is identical to the game described in Chapter 3.1 except for the reward function. The external factor is modeled as a third player, called Nature, who does not learn nor actively participate in exchanges, but only awards points. Nature has her own split g_N, which she preserves throughout the game. Also, the points awarded by Nature are included with the weight of w_N into the final score whereas the points of a sender are included with weight w_P, $w_P + w_N = 1$, $w_P, w_N \geq 0$. Thus, the modified reward function r^* is defined as:

$$r^*(c_S, g_S, g_N, f, f') = w_P \cdot r(c_S, g_S, f') + w_N \cdot n(f, g_N, f'),$$

where $r(\cdot)$ is defined as in Chapter 3.2 and $n(\cdot)$ is defined as

$$n(f, g_N, f') = \begin{cases} 1 \text{ if } \ g_N(f) = g_N(f'), \\ 0 \text{ otherwise.} \end{cases}$$

Informally, Nature awards a player a point if and only if the object f representing the state of the world corresponds to the action taken by the receiver f' in the sense that they fall into the same Nature's category. This definition retains the notion natural to signaling games that there is some external correspondence between states of the world and actions. It is important to underline that the reward function n does not take into account the signal used, i.e. Nature does not use the dictionary.

3.4 Learning algorithms

Learning algorithms described in Chapter 2.3 require a slight modification to fit the proposed model in its basic form (as introduced in Chapter 3.2). The implementation, described in more detail in Chapter 4.1, follows the urns concept, in which each urn is labeled with an object $f \in F$ and contains propensities for each signal. Unlike in standard HRL model, each player has urns for the sender's role only. Since the set of states of the world and the set of actions are equal, two independent set of urns would lead to incoherent behavior, in which a player e.g. observes a state of the world f and sends a signal c'_S, but reacts with the same object f upon receiving a different signal c''_S within

the same round, providing conflicting evidence. Instead, a player at the beginning of each round draws a signal for each object and thus defines her split g. She preserves this split throughout the round and uses it to both send a signal and respond with an action. The urns are updated after each round and the process repeats.

The extended model (see Chapter 3.3) additionally modifies the updating rule. In each round there are two exchanges and thus two pieces of evidence to learn from. However, in the model with external factor, the pay-off is split into coordination part (sender's points) and external part (Nature's points). Let us focus on one exchange only.[8] The sender observes in the round t the state of the world f, sends a signal c_S according to her split g_S and the receiver responds with object f'. Both sender and receiver at the end of the round update their propensities according to the formula:

$$q_{i,j}(t+1) = \begin{cases} q_{i,j}(t) + w_P \cdot r(c_S, g_S, f') \text{ if } i = c_S,\ j = f', \\ q_{i,j}(t) \text{ otherwise.} \end{cases}$$

That is, they increase the chances of coordinated behavior in the next rounds. Additionally, Nature awards points for the "correct" classification. However, Nature's points take into account the state of the world f, to which the receiver has no access. Therefore, only the sender can use this evidence to update her rules. She does so by using the analogous formula:

$$q_{i,j}(t+1) = \begin{cases} q_{i,j}(t) + w_N \cdot n(f, g_N, f') \text{ if } i = f,\ j = c_S \\ q_i(t) \text{ otherwise.} \end{cases}$$

Given that there are two exchanges in each round, the symmetry between players is preserved.

[8] It is important to note that one round consists of two exchanges and only once two exchanges are carried out the players learn.

4 Simulation

Simulation is a scientific tool, made possible by developments in computing resources, which allow for analysis of phenomena too complex to be feasibly studied formally. It has been used in the modeling of linguistic phenomena since 90. XX (see e.g. Steels (1995)). In this chapter I describe the experimental design and technical details of the implementation.

4.1 Implementation

The simulation environment has been implemented in R programming language. The source code has been attached in Appendix A. The simulation is based primarily on two classes:

1. Figure[9] with three attributes: shape, size and color, all of type *numeric*,
2. Agent with two attributes: split (*list*) and score (*list*).

Agent is what could be called an abstract class, which does not implement any learning algorithm, but hosts functions common to all learning agents. Score attribute stores the points awarded to this player in the last 50 rounds[10], while split attribute stores objects corresponding to each term in the dictionary. It is an implementation of a relation g^{-1}.

Herrnstein reinforcement learning. The simulation is parametrized by:

1. $iter$ – number of iterations of the experiment,
2. n – number of rounds per iteration,
3. $figDims$ – a list of vectors with size, shape and color values, which define the universe of objects (through Cartesian product),
4. $dict$ – a dictionary for the game.

[9]The original motivation behind thinking of objects as figures was to add a complexity factor (e.g. in terms of Boolean complexity, following research in Feldman (2000) and Shepard et al. (1961); more on this in Chapter 6). This line of research has not been eventually pursued in this thesis as the learning algorithms used do not take into account any complexity of the categories. Original terminology has been left to point to possible extensions to the model.

[10]For rounds 1-50, scores for all previous rounds are stored.

The learning algorithm as implemented consists of the following steps:

1. players are initialized as instances of the class hrlLearner (subclass of the class Agent).

2. The initial environment is set, i.e. the objects are defined and all fields are set to their initial values. That is:

   - initial splits for each player are randomly chosen assuming equal propensities (i.e. it is equally likely for each object to be classified under any term). Degenerate splits are allowed, i.e. a player may classify all objects under one term.
   - Urns are initialized for each player as a named list. Each "urn" (element of the list) stores current propensities for each term.

3. Players begin the communication process, which they continue for n rounds:

   (a) Player 1 (P1 for short) assumes the role of the sender and Player 2 (P2) – the role of the receiver.

   (b) An object f is drawn randomly with discrete uniform distribution over all objects in the universe.

   (c) The sender (S for short) sends a communicate c_S from the dictionary to the receiver (or R) based on her current split g_S and the object f drawn in the previous step.

   (d) R receives the communicate c_S and sends an object f' to S, based on her current split g_R. It is possible that at this point of the game R categorizes all objects under the other term and does not know how to respond to term c_S. In this case, she updates her split g_R with no change in propensities (there are no new reinforcements) until any object falls into c_S category and then draws a response f'.

   (e) S compares the received object f', sent communicate c_S and her split g_S and awards R a point if and only if the received object falls into the category named c_S.

(f) The communication process is symmetrically reversed, i.e. P2 assumes the role of the sender (previously held by P1) and P1 assumes the role of the receiver (previously held by P2).

(g) Each player updates her split twice, taking into account information from both exchanges, in which she acted either as the sender or as the receiver; the update of the split consists of update of the urns (according to the algorithm described in Chapter 3.4) and random choice of the term for each object according to a probability based on propensities.

A few comments are worth making. First, it is assumed that one round is a two-way communication, i.e. each player acts as the sender and the receiver, and only then they update their private information. Secondly, players do not store information for their receiver/sender roles separately, as in this case the set of states of the world and the set of possible actions are equal. Thus, players learn from both exchanges and use the same split both times for coherence.

Described algorithm is modified for other learning algorithms. SRL model is implemented analogously, except for the updating rule and the response rule as defined in Chapter 2.3.2.

4.2 External factor

The experiment in its basic form is based purely on coordination, i.e. there is no external factor to restrict the categorization on which the players aim to agree. While this assumption may be indeed justified for some abstract terms, often enough the categorization is influenced by e.g. a functionality of the objects. This section describes the implementation of the modified version of the model as introduced in Chapter 3.3. Corresponding analyses consider the impact of Nature on the performance of learning algorithms.

Herrnstein reinforcement learning with Nature. The simulation is parametrized as before by $iter$, n, $figDims$ and $dict$, and additionally by:

1. w_p – weight of the player's reward p in the final reward function r,

2. w_n – weight of Nature's reward n in the final reward function r.

The game algorithm is identical to the one described in Chapter 4.1 except for the initialization step and the reward function. That is, along with the initialization of the players, a third player Nature N is created and her split g_N is initialized analogously to the splits of other players. The difference, however, lies in the fact that Nature does not update her split during the game and does not learn from other players. Moreover, the reward function r^* is modified to additionally reward the exchanges favored by Nature (i.e. coinciding with Nature's split). The learning algorithms follow the procedure described in Chapter 3.4.

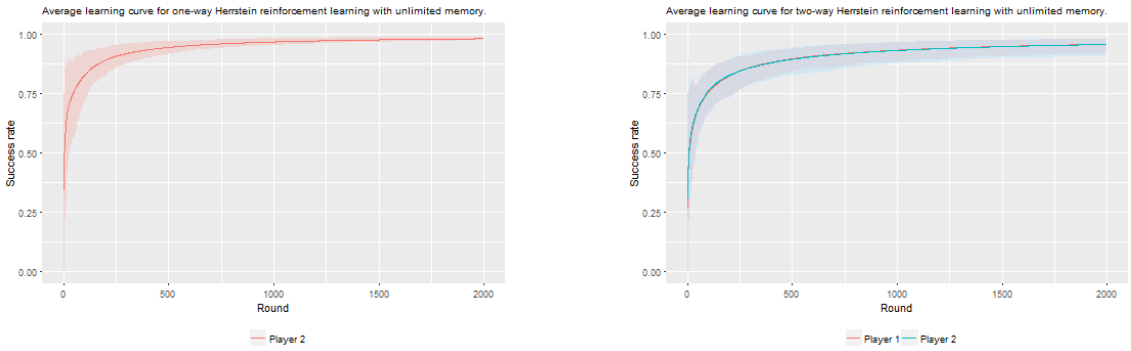
4.3 Experiment and Basic Measures

For each experimental setting, a game of 2000 rounds (iterations) is played 100 times. The measure of performance is a *success rate*, which is calculated as a mean pay-off in the last 50 rounds (following the approach in Zubek et al. (2017)). This measure results in some uneven behavior in the first rounds, but in the long run should converge to one for successful learning process. To assess the impact of coordination, the experiment is also carried out for one-way communication, in which the sender/receiver roles are fixed and the sender acts as a teacher (i.e. she does not update her split or learn from exchanges). Additionally, 2.5% and 97.5% percentiles are plotted to show the dispersion of results.

5 Results

This section presents results of the simulation carried out according to the experimental design described in Chapter 4. The main purpose of the analysis is to give more insight into the problem of category formation and the impact of socio-environmental factors.

5.1 Herrnstein Reinforcement Learning



(a) One-way scenario.

(b) Two-way scenario

Figure 1: Learning curve averaged over 100 repeats of Herrstein Reinforcement Learning. *Source:* own calculations.

Herrnstein Reinforcement Learning is a very basic model, which nonetheless allows for the agents to learn successfully and offers a benchmark for further analyses.

Figure 1a shows that in one-way scenario, the learning agent does successfully assimilate the categorization of her teacher as the learning curve approaches to 1. What is more, the variability of results is low for the final rounds, which suggests that the overall success does not depends on the initial categorization, in spite of some significant fluctuations at the beginning.

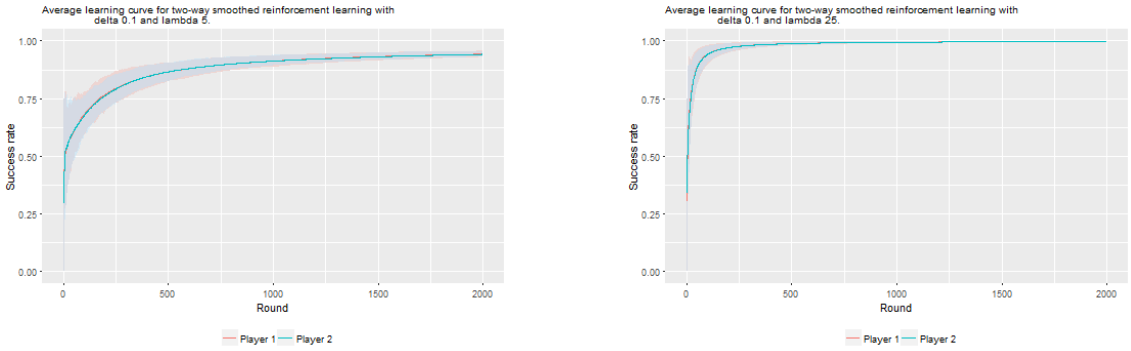
Figure 1b shows that coordinate learning is indeed more difficult. The learning curve retained its shape, but is visibly shifted down, which means that at the same point in the game the average score of each player was lower than the score of the learning player in one-way scenario. Moreover, the lower and upper bound go in parallel throughout the game and do not converge, which suggests that the learning process requires more rounds for convergence or perhaps some categorizations end in loops and players never reach stable solution (shared categorization). Also, it is worth underlining that both average

success rate and confidence intervals perfectly overlap for two players, which is expected for all plots given the perfect symmetry of players.

5.2 Smoothed Reinforcement Learning

Next part of analysis is dedicated to verifying the properties of SRL algorithm. As has been stated in Chapter 2.3.2, the model is parametrized by λ, which determines to what extent small changes in propensities affect the final probabilities (reactivity parameter), and δ, which is the parameter of the forgetfulness of a player.

First scenario is parametrized by $\delta = 0.1$, $\lambda = 5$. It is the case that resembles the most the basic HRL algorithm. Indeed, the learning curve (shown in Figure 2a) is similar to the one in Figure 1b. The convergence to 1 is slower, which implies slower learning pace for this algorithm.



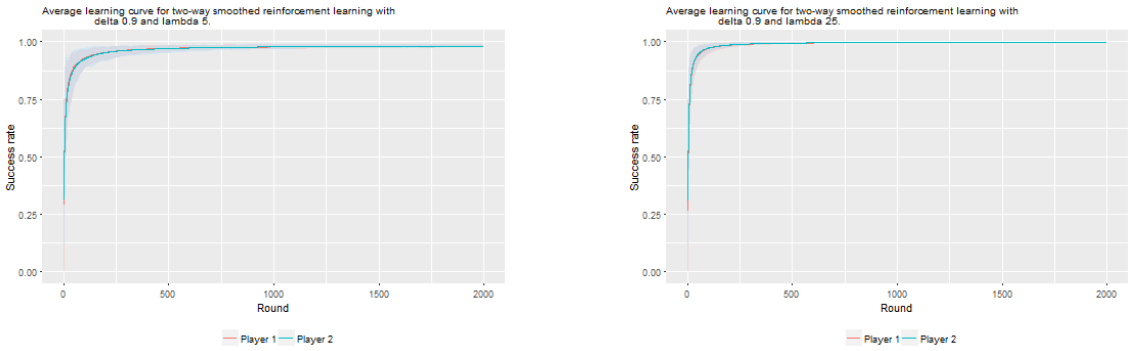
(a) $\delta = 0.1$, $\lambda = 5$

(b) $\delta = 0.1$, $\lambda = 25$

Figure 2: Learning curve averaged over 100 repeats of Smoothed Reinforcement Learning, two-way scenario. *Source:* own calculations.

Figure 2b shows a somewhat intermediate case with $\delta = 0.1, \lambda = 25$, in which the player updates the propensities slowly, but even small changes affect the probabilities. Such player is not really forgetful, but changes her mind quickly upon new evidence. The learning curve shows the fastest convergence to 1 of all algorithms analyzed so far, including the one-way basic HRL model.

Figure 3a shows a reverse case of a player, who puts much emphasis on a new evidence, but reacts slowly to differences in propensities with $\delta = 0.9, \lambda = 5$. The learning curve is shifted slightly down with respect to Figure 2b, but still surpasses other algo-



(a) $\delta = 0.9$, $\lambda = 5$

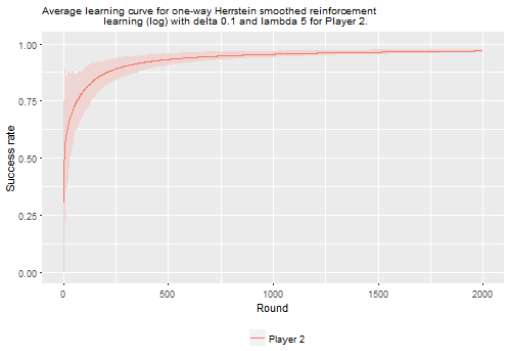
(b) $\delta = 0.9$, $\lambda = 25$

Figure 3: Learning curve averaged over 100 repeats of Smoothed Reinforcement Learning, two-way scenario. *Source:* own calculations.

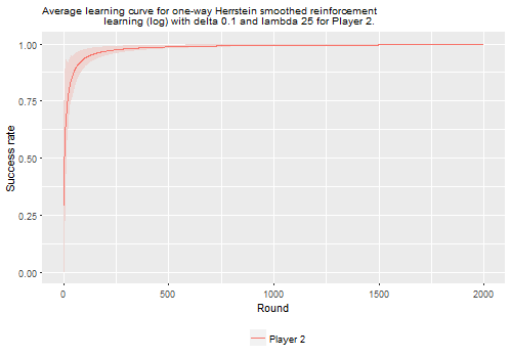
rithms. A comment is worth making at this point. Apparently, λ and δ are to some extent interchangeable, i.e. a player with low forgetfulness and high reactivity behaves similarly to a player with high forgetfulness and low reactivity. While undoubtedly two parameters give more modeling power, for some applications it might be reasonable to retain only one parameter for the ease of analysis and interpretation.

Figure 3b shows the performance of the model, in which the players are extremely forgetful and reactive ($\delta = 0.9, \lambda = 25$). It gives by far the best results, with quick convergence to 1 and very steep learning curve. This outcome is not surprising given the definition of the game. The players change their minds every round, which makes remembering previous exchanges deep in the past unjustified as the other player most likely has changed her categorization since. The contrary is true for a learner-teacher scenario, in which past observations are equally valid throughout the game.

For comparison, the same test cases were run for one-way scenario. Brief analysis shows that in this case it is important that the learning player be either very reactive, forgetful, or both. Low-reactivity, low-forgetfulness scenario stands out when compared with other parametrizations, which feature much faster learning pace. It means that even when the agent learns a constant categorization, it help when she relies more on new observations and reacts quickly to differences in propensities.

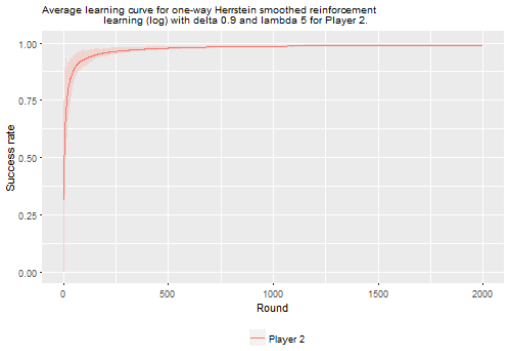


(a) $\delta = 0.1, \lambda = 5$

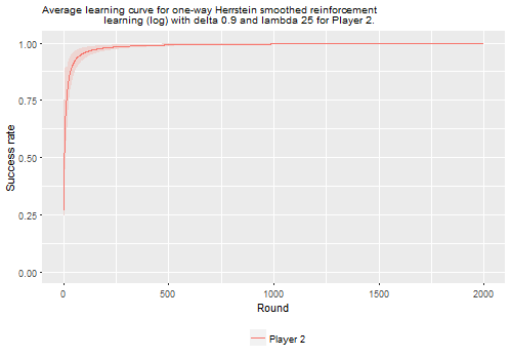


(b) $\delta = 0.1, \lambda = 25$

Figure 4: Learning curve averaged over 100 repeats of Smoothed Reinforcement Learning, one-way scenario. *Source:* own calculations.



(a) $\delta = 0.9, \lambda = 5$



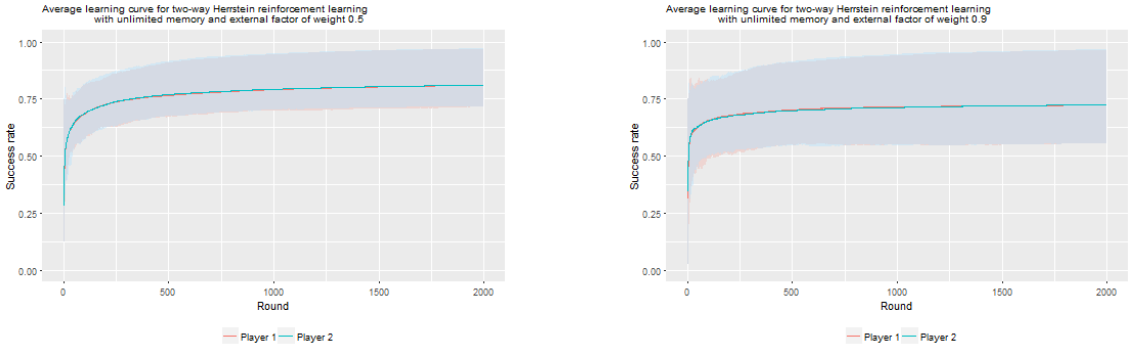
(b) $\delta = 0.9, \lambda = 25$

Figure 5: Learning curve averaged over 100 repeats of Smoothed Reinforcement Learning, one-way scenario. *Source:* own calculations.

5.3 Herrnstein reinforcement learning with Nature

This section presents results of simulation of extended HRL model, which introduces natural groupings. The simulation design has been described in Chapter 4.2.

Figures 6a and 6b show the average learning curves for w_N equal to 0.5 and 0.9 respectively. Let us briefly recall that w_N is the weight of the points granted by Nature in the overall score, which also affects the pace of the learning process. As can be seen, HRL algorithm performs rather poorly in this extended scenario. The learning curve is nowhere near convergence to 1 and the dispersion of the results is visibly larger than in any other test case presented so far, especially for $w_N = 0.9$. One of the characteristics that makes HRL model less effective in coordination games of this type is unlimited memory. Next



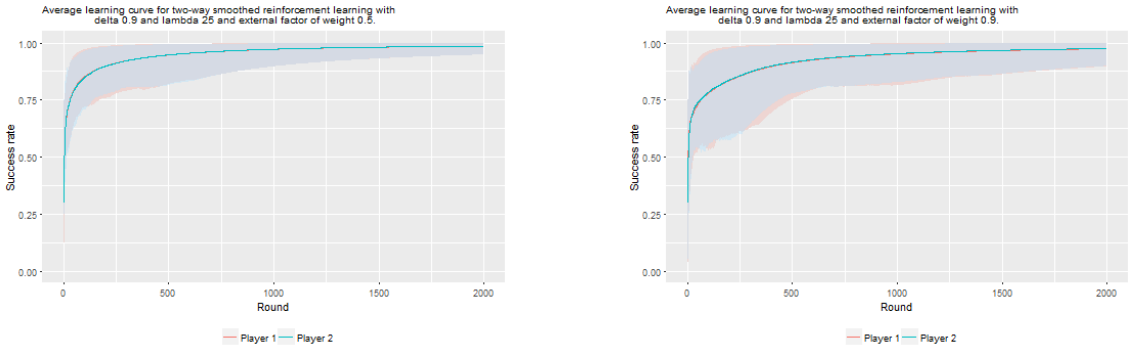
(a) $w_N = 0.5$ (b) $w_N = 0.9$

Figure 6: Learning curve averaged over 100 repeats of Herrstein Reinforcement Learning with Nature, two-way scenario. *Source:* own calculations.

step of the analysis will verify whether SRL model performs better in analogous scenarios.

5.4 Smoothed reinforcement learning with Nature

Previous analyses have shown that SRL model, especially in its most forgetful and reactive settings, yields faster learning pace than basic HRL algorithm. The following test cases aim at verifying to what extent the forgetfulness improves the average success rate when natural groupings are included in the model.



(a) $w_N = 0.5$ (b) $w_N = 0.9$

Figure 7: Learning curve averaged over 100 repeats of Smoothed Reinforcement Learning with Nature, two-way scenario. *Source:* own calculations.

Figures 7a and 7b show average learning curves for w_N equal to 0.5 and 0.9 respectively. As can be seen, the forgetfulness significantly improves the learning efficiency.

The learning curve approaching 1 implies that on average the players agree on the classification, which is also concordant with the natural groupings. Wide confidence intervals show that learning curve is to some extent dependent on initial settings. However, the distribution is visibly not symmetrical around the mean, which suggests that the lower bound reflects the outliers rather than an expected behavior.

5.5 Summary of results

The simulation offers an insight into the dynamics of the game conditional on the implemented learning algorithm and parametrization. Table 1 shows basic statistics after 100th round in each scenario. Table 2 shows analogous figures after 2000th round (at the end of the game).

Table 1: Summary statistics for each test case after 100th round.

Test case	Mean	Standard deviation
HRL	0.7635	0.0527
SRL ($\delta = 0.1, \lambda = 5$)	0.6782	0.0494
SRL ($\delta = 0.1, \lambda = 25$)	0.9414	0.0206
SRL ($\delta = 0.9, \lambda = 5$)	0.9245	0.0254
SRL ($\delta = 0.9, \lambda = 25$)	**0.9727**	0.012
HRL ($w_N = 0.5$)	0.6871	0.0666
HRL ($w_N = 0.9$)	0.6539	0.1054
SRL ($\delta = 0.9, \lambda = 25, w_N = 0.5$)	**0.8487**	0.0756
SRL ($\delta = 0.9, \lambda = 25, w_N = 0.9$)	0.7841	0.1163

There are a few main conclusions to be drawn from these analyses. First, the two-way scenario, in which both players modify and learn categorizations, is indeed more difficult than a corresponding one-way teacher-learner scenario. Secondly, in line with previous research limited memory helps players learn the classifications at faster pace. The forgetfulness is even more important in the extended model, which includes external factor apart from the pure coordination task.

Table 2: Summary statistics for each test case after 2000th round.

Test case	Mean	Standard deviation
HRL	0.9586	0.0171
SRL ($\delta = 0.1, \lambda = 5$)	0.9419	0.0068
SRL ($\delta = 0.1, \lambda = 25$)	0.9970	0.0010
SRL ($\delta = 0.9, \lambda = 5$)	0.9810	0.0039
SRL ($\delta = 0.9, \lambda = 25$)	**0.9986**	0.0006
HRL ($w_N = 0.5$)	0.8108	0.0672
HRL ($w_N = 0.9$)	0.7241	0.1177
SRL ($\delta = 0.9, \lambda = 25, w_N = 0.5$)	**0.9864**	0.0128
SRL ($\delta = 0.9, \lambda = 25, w_N = 0.9$)	0.9759	0.0254

6 Future Work

The study reported in this thesis by no means exhausts the subject introduced in this thesis. There are a few promising lines of research to follow.

6.1 Empirical study

The design of the game described in Chapter 3.1 lends itself, after some modifications, to an experiment with human participants. The purpose of this experiment is to test how human participants behave in the given settings and to what extent the simulation and empirical results overlap.

The proposed experimental task is as follows. Two human subjects are separated with the only communication channel being the computer experimental platform. Each of the subjects (randomly labeled "Player 1" and "Player 2") is displayed a set of figures. Each figure can be described in three dimensions:

1. size (small, medium, large),
2. color (yellow, blue, black),
3. shape (triangle, square, circle).

Each characteristic is emphasized enough as to be unambiguously distinguishable by the participants. Players are then asked to divide the figures into two sets ("A" and "B") and each card must be in one and only one set. Players see only their own cards and know nothing about the other player's split. Then, Player 1 sends a communicate being either "A" or "B" (with the choice of the communicate not being restricted). Player 2 has to point to a card according to the received communicate. If the chosen card is of the right category (is relevant to the communicate) from the perspective of Player 1, Player 2 gets one point. She receives no reward otherwise. Players then switch roles, i.e. Player 2 sends a signal and Player 1 chooses a card. They may rearrange their sets at any time.

6.2 Expanded test cases

The simulation study verified only a fraction of possible test cases. One possible modification concerns initial propensities in both learning algorithms, which have been assumed

to be equal and of the same order of magnitude as reinforcements. As Barrett and Zollman (2009) point out, lifting any of these assumptions influences the game dynamics in standard signaling games models. It is expected that unequal initial propensities may shorten the time needed for convergence as the players will have common tendency for some classifications. However, if each player starts with different propensities, the more they are unbalanced, the more time will be needed for learning evidence to override this initial bias. When it comes to the order of magnitude, given the coordination character of the game, the lower their impact is, the better. Yet, it is interesting to see what is the extent of this impact.

6.3 Authority factor

In the presented model the players are perfectly symmetrical and learn according to the same algorithm. An interesting and significant extension may be introducing the authority factor, i.e. differentiating the players. It may be done either through varying the rewards, limiting the learning process of one players (i.e. making her "stick to her mind" by allowing to update her split only once in some number of rounds) or assuming an altogether different learning algorithm for each player. Such model would be a somewhat intermediate case between one-way and two-way scenarios presented in this thesis.

6.4 Complexity factor

One of the major aspects of learning categories, which is underrepresented in the proposed model, is the cognitive complexity. Shepard et al. (1961) lists a number of studies that strongly suggest that *ceteris paribus* the systematic assignments (arrangements) are easier to learn than random ones, which points at the cognitive complexity of the task as one of the factors. Nonetheless, measuring the complexity of the task is an altogether separate issue with many competing measures and approaches (see e.g. Feldman (2000) and Goodwin and Johnson-Laird (2013)). One of the benchmark concepts is a so called

boolean complexity, which defines complexity as the length of the minimal equivalent expression in propositional logic[11].

Remarkably, as Shepard et al. (1961) reports, the physical representation of the stimuli does matter. Studies with compact representation, i.e. one in which the particular properties (or dimensions) are not easily distinguishable, yield different results than those with distributed stimuli, in which the properties can be easily isolated. One of the hypothetical explanations is that stimuli represented in a compact manner are cognitively (perceptually) complex due to their very representation, something that the boolean complexity fails to take account of.

This brief overview also sets the direction for further extensions to the model. A model of Bayesian learning could be implemented, where prior probabilities of each split would be inversely proportional to their cognitive complexity, consistently with experimental data. Also, it may be interesting to include this factor in the empirical study to compare the two models in terms of how accurately they describe the human behavior both for distributed and compact stimuli.

[11] This approach, extensively described in Chater and Vitanyi (2003), somewhat resembles the concept of Kolmogorov complexity, which defines the complexity of an object x as the length of the shortest program (in any universal programming language) for x.

7 Summary and Conclusions

This thesis is dedicated to learning of categories in language games. In particular, a signaling games model is used and significantly modified to accommodate the categorization process.

A standard signaling game, as introduced into linguistics by Lewis (1969), is a model of emergence of successful communication system in the situation of the asymmetry of information and the lack of a pre-established language. The model proposed in this thesis uses this framework, but shifts the focus from identification to categorization task by reducing the number of available terms in the dictionary. The protocol of the communication is defined as follows. Each player starts with a private, random classification of pre-defined objects into named classes. Player 1 (the sender) thinks of an object and communicates her thoughts using any term in the dictionary. Player 2 (the receiver) responds by providing her own example of an object belonging to the same class. The sender rewards the receiver if she judges the response valid from the point of view of her private classification. They switch roles and eventually aim at converging to the same classification.

Since players cannot communicate outside of the protocol and have no prior knowledge, they must learn from the evidence gathered throughout the game. Then arises the question: what are the factor influencing the learning process? Christiansen and Chater (2008*b*) favors, among others, the hypothesis of environmental pressure, shaping the language for externally optimal expressiveness. This claim is subsequently questioned by Wallentin and Frith (2008), who argue that social aspects might account for some linguistic phenomena, e.g. variations in meaning not explained by environmental constraints. In the thesis this discussion was addressed by considering different models: one in which players' only goal was to coordinate with each other and all shared categorizations are equally good, and one with varying environmental pressure, which forces the players to reflect natural groupings apart from establishing a shared categorization. The impact of each factor (i.e. social vs environmental) is parametrized, which allows for modeling many cases, ranging from pure coordination to a prevalent external pressure.

To computationally verify the game dynamics under different scenarios, two learning algorithms were modified and implemented: Herrnstein reinforcement learning and

smoothed reinforcement learning. They were consequently used for simulation study. First, I considered the coordination-only scenario, in which there was no external factor. The study showed that it is possible to converge to a shared categorization via proposed communication protocol and both learning algorithms. However, smoothed reinforcement learning, which allows for limiting the players' memory, performed significantly better. This observation is consistent with conclusions drawn by Barrett and Zollman (2009).

Next, several test cases with varying weight of environmental factor were considered. In this extended scenario, the forgetfulness of the players proved even more important as Herrnstein reinforcement learning with unlimited memory did not converge on average to a shared classification consistent with natural groupings in the studied range of 2000 rounds. Smoothed reinforcement learning in the studied parametrization on average converged to a shared categorization consistent with natural groupings, which suggests that independent social and environmental reinforcements do not exclude the possibility of establishing a language nor significantly impair the learning process. It must be underlined that those observations are only valid for smoothed reinforcement learning model with high forgetfulness (recent observations have the biggest impact on the player's behavior) and high reactivity (the player quickly "changes her mind" upon new evidence).

The conducted simulation study suggests that for symmetrical players high social pressure increases rather than slows down the learning pace. The limitless memory turned out to be the biggest hindering factor, which made it impossible for the players to reach the common categorization. Overall, the obtained results suggest that the coordination and learning from each other help the players to establish a shared categorization, which is socially-accepted as well as environmentally fit as long as they receive the same environmental reinforcements and do not take into account outdated observations. In this case, the social and environmental constraints, although independent, in the long-term are mutually supportive rather than contradictory. While these findings by no means refute the observation by Wallentin and Frith (2008) of the independence and potential conflict between the two factors, they suggest that in perfectly adaptable, homogeneous language communities, social interaction increases environmental fitness.

A Appendix: Source code

The appendix contains declaration of all classes and functions used for simulations presented in this thesis.

Base functions

```
#####
## import
#####
library(ggplot2)

#####
## declaration of classes
#####

Figure <- setRefClass(
  "Figure",
  fields = list(color = "factor", size = "factor", shape = "factor")
)

Agent <- setRefClass(
  "Agent",

  fields = list(
    split = "list",
    score = "vector"
  ),

  methods = list(

    makeSplit = function(figures, dict){
      split <- list()
      # how many figures are there for splitting
      count <- length(figures)
      # how many labels are left - necessary to avoid degenerate cases
      remain <- length(dict) - 1
      # random permutation
      figures <- sample(figures)
      for(s in dict) {
        if(remain > 0 & count > remain) {
          card <- sample.int(count - remain, 1)
        } else {
          card <- count
        }
        split[[s]] <- figures[1:card]
        figures <- figures[(card+1):length(figures)]
        count <- count - card
        remain <- remain - 1
      }

      return(split)
    },

```

```
    sendCommunicate = function(figure) {
      for(i in 1:length(split)) {
        if(isIn(split[[i]], figure)) {
          return(names(split)[i])
        }
      }
      stop("I do not recognize the figure!")
    },

    sendFigure = function(communicate) {
      while (length(split[[communicate]]) == 0) {
        # choose randomly a figure
        figure <- sample(unlist(split))[[1]]
        # update split taking into account this figure and failure to communicate
        split <<- updateSplit(figure, communicate, 0)
      }
      return(sample(split[[communicate]], 1)[[1]])
    },

    givePoint = function(communicate, figure) {
      return(isIn(split[[communicate]], figure))
    },

    updateScore = function(point) {
      if(length(score < 50)) {
        score <<- c(score, point)
      } else {
        score <<- c(score[2:50], point)
      }
    }
  )
)

#####
## functions
#####

isIn <- function(list, figure) {
  for(f in list) {
    if(identical(f, figure)) {
      return(TRUE)
    }
  }
  return(FALSE)
}


testMeans <- function(res1, res2) {
  pvalues <- c()
  for(i in 1:length(res1)) {
    pvalues[i] <- t.test(res1[[i]], res2[[i]])$p.value
  }
  return(pvalues <= 0.05)
}

```

```
drawFigure <- function(figures) {
  return(sample(figures, 1)[[1]])
}

oneRound <- function(env, isTwoWay) {
  # player1 starts as a sender
  drawn1 <- drawFigure(env$figures)
  signal1 <- env$player1$sendCommunicate(drawn1)
  action1 <- env$player2$sendFigure(signal1)
  point1 <- env$player1$givePoint(signal1, action1)
  env$player2$updateScore(point1)

  if(isTwoWay) {
    # switch roles
    drawn2 <- drawFigure(env$figures)
    signal2 <- env$player2$sendCommunicate(drawn2)
    action2 <- env$player1$sendFigure(signal2)
    point2 <- env$player2$givePoint(signal2, action2)
    env$player1$updateScore(point2)
  }

  # update splits
  env$player2$split <- env$player2$updateSplit(action1, signal1, point1)
  if(isTwoWay) {
    env$player1$split <- env$player1$updateSplit(action1, signal1, point1)
    env$player1$split <- env$player1$updateSplit(action2, signal2, point2)
    env$player2$split <- env$player2$updateSplit(action2, signal2, point2)
  }
  return(env)
}

playGame <- function(n, figDims, dict, isTwoWay, player1, player2) {
  env <- setEnvironment(figDims, dict, player1, player2)
  avgScores <- list(player1 = c(), player2 = c())
  scores <- list(player1 = c(), player2 = c())

  for(i in 1:n) {
    env <- oneRound(env, isTwoWay)
    avgScores$player1 <- c(avgScores$player1, mean(env$player1$score))
    avgScores$player2 <- c(avgScores$player2, mean(env$player2$score))
  }

  return(avgScores)
}

initFigs <- function(figDims) {
  # check configuration
  stopifnot(c("color", "size", "shape") %in% names(figDims))

  # initialize figures
  combs <- expand.grid(figDims$color, figDims$size, figDims$shape)
  colnames(combs) <- c("color", "size", "shape")
  figures <- c()

  for(i in 1:nrow(combs)) {
```

```
    set <- combs[i,]
    newFig <- Figure$new(color = set$color, size = set$size, shape = set$shape)
    figures <- c(figures, newFig)
  }
  return(figures)
}

plotRes <- function(res, title) {
  res <- as.data.frame(res)
  if(any(res$player1) && any(res$player2)) {
    g <- ggplot(res, aes(1:nrow(res), player1, player2)) +
      geom_line(aes(y = player1, colour = "Player 1")) +
      geom_line(aes(y = player2, colour = "Player 2")) +
      xlab("Iteration") + scale_y_continuous(limits = c(0, 1)) +
      ggtitle(title) +
      theme(plot.title = element_text(lineheight = 0.8, size = '10')) +
      theme(legend.position="bottom", legend.title=element_blank())
  } else if(any(res$player1)) {
    g <- ggplot(res, aes(1:nrow(res), player1)) +
      geom_line(aes(y = player1, colour = "Player 1")) +
      xlab("Iteration") + scale_y_continuous(limits = c(0, 1)) +
      ggtitle(title) +
      theme(plot.title = element_text(lineheight = 0.8, size = '10')) +
      theme(legend.position="bottom", legend.title=element_blank())
  } else {
    g <- ggplot(res, aes(1:nrow(res), player2)) +
      geom_line(aes(y = player2, colour = "Player 2")) +
      xlab("Iteration") + scale_y_continuous(limits = c(0, 1)) +
      ggtitle(title) +
      theme(plot.title = element_text(lineheight = 0.8, size = '10')) +
      theme(legend.position="bottom", legend.title=element_blank())
  }

  return(g)
}

aggRes <- function(playerRes, raw) {
  agg <- list()
  for(i in 1:length(playerRes[[1]])) {
    print(i)
    part <- unlist(lapply(playerRes, function(x) x[i]))
    n <- length(part)
    avg <- mean(part)
    stdev <- sd(part)
    agg$stdev[[i]] <- stdev
    agg$mean[[i]] <- avg
    agg$low[[i]] <- quantile(part, 0.025)
    agg$hi[[i]] <- quantile(part, 0.975)

    if(raw) {
      if(i == 100) {
        agg$raw[[1]] <- part
      }
      if(i == 1000) {
        agg$raw[[2]] <- part
```

```
      }
      if(i == 2000) {
        agg$raw[[3]] <- part
      }
    }
  }

  return(agg)
}

#####
## simulation
#####

runSimulation <- function(iter, n, figDims, dict, isTwoWay, raw = FALSE) {
  res <- list()
  for(i in 1:iter) {
    player1 <- initPlayer()
    player2 <- initPlayer()
    part <- playGame(n, figDims, dict, isTwoWay, player1, player2)
    res$player1[[i]] <- part$player1
    res$player2[[i]] <- part$player2
  }
  agg <- list()
  agg$player1 <- aggRes(res$player1, raw)
  agg$player2 <- aggRes(res$player2, raw)

  return(agg)
}

runSimulationMeans <- function(iter, iterSim, n, figDims, dict, isTwoWay, raw = FALSE) {
  res <- list()
  for(i in 1:iter) {
    part <- runSimulation(iterSim, n, figDims, dict, isTwoWay)
    res$player1[[i]] <- part$player1$mean
    res$player2[[i]] <- part$player2$mean
  }
  agg <- list()
  agg$player1 <- aggRes(res$player1, raw)
  agg$player2 <- aggRes(res$player2, raw)

  return(agg)
}

plotResSim <- function(res, title) {
  res <- as.data.frame(cbind(
    res$player1$mean,
    res$player1$low,
    res$player1$hi,
    res$player2$mean,
    res$player2$low,
    res$player2$hi)
  )
  colnames(res) <- c("player1", "hi1", "low1", "player2", "hi2", "low2")
  if(any(res$player1) && any(res$player2)) {
```

```
    g <- ggplot(res, aes(1:nrow(res), player1, player2)) +
      geom_ribbon(aes(ymin = res$low1, ymax = res$hi1), fill = "mistyrose2") +
      geom_ribbon(aes(ymin = res$low2, ymax = res$hi2), fill = "lightskyblue1", alpha = '0.4') +
      geom_line(aes(y = player1, colour = "Player 1")) +
      geom_line(aes(y = player2, colour = "Player 2"))
  } else if(any(res$player1)) {
    g <- ggplot(res, aes(1:nrow(res), player1)) +
      geom_ribbon(aes(ymin = res$low1, ymax = res$hi1), fill = "lightskyblue1", alpha = '0.4') +
      geom_line(aes(y = player1, colour = "Player 1"))
  } else {
    g <- ggplot(res, aes(1:nrow(res), player2)) +
      geom_ribbon(aes(ymin = res$low2, ymax = res$hi2), fill = "mistyrose2") +
      geom_line(aes(y = player2, colour = "Player 2"))
  }

  g <- g + xlab("Round") + scale_y_continuous(limits = c(0, 1)) + ylab("Success rate") +
    ggtitle(title) + theme(plot.title = element_text(lineheight = 0.8, size = '10')) +
    theme(legend.position="bottom", legend.title=element_blank())
  return(g)
}
```

Herrnstein Reinforcement Learning

```
#####
## learning agents - basic Herrnstein RL, full memory
#####

hrlLearner <- setRefClass(
  "hrlLearner",
  fields = list(
    urns = "list",
    split = "list",
    score = "numeric"
  ),
  contains = "Agent",
  methods = list(
    initUrns = function(figures, dict) {
      urns <- list()

      for(i in 1:length(figures)) {
        # initialize equal propensities
        # figure is stored in the first position of the list (cannot be a name)
        urns[[i]] <- c(figures[[i]], rep(1, length = length(dict)))
        names(urns[[i]]) <- c("figure", dict)
      }
      return(urns)
    },

    findUrn = function(figure) {
      for(i in 1:length(urns)) {
        if(identical(urns[[i]][[1]], figure)) {
          return(i)
        }
      }
    },
```

```
33
34      updateUrns = function(figure, communicate, point) {
35        # no penalty
36        if(point) {
37          urns[[findUrn(figure)]][[communicate]] <<- urns[[findUrn(figure)]][[communicate]] + 1
38        }
39      },
40
41      updateSplit = function(figure, communicate, point) {
42        updateUrns(figure, communicate, point)
43        split <- list()
44
45        for(i in 1:length(urns)) {
46          # for each urn extract propensities
47          urn <- unlist(urns[[i]][-1])
48          # for each urn extract figure it corresponds to
49          figure <- urns[[i]][[1]]
50          # calculate probabilities linearly
51          probs <- urn/sum(urn)
52          # choose a name for this figure
53          drawn <- sample(x = dict, size = 1, prob = probs)[[1]]
54          # add figure to this part of the split
55          split[[drawn]] <- c(split[[drawn]], figure)
56        }
57
58        return(split)
59      }
60    )
61  )
62
63  # create player
64  initPlayer <- function() {
65    player <- hrlLearner$new(
66      split = list(),
67      score = 0,
68      urns = list()
69    )
70    return(player)
71  }
72
73  # set envirnment (agent type specific function)
74  setEnvironment <- function(figDims, dict, player1, player2) {
75
76    # initialize figures
77    figures <- initFigs(figDims)
78
79    # configure players
80    player1$split <- player1$makeSplit(figures, dict)
81    player1$urns <- player1$initUrns(figures, dict)
82
83    player2$split <- player2$makeSplit(figures, dict)
84    player2$urns <- player2$initUrns(figures, dict)
85
86    # set environment
87    env <- list()
```

```
  env$figures <- figures
  env$player1 <- player1
  env$player2 <- player2
  env$dict <- dict

  return(env)
}
```

Smoothed Reinforcement Learning

```
#####
## learning agents - basic Herrnstein RL, smoothed memory
#####

srlLearner <- setRefClass(
  "srlLearner",
  fields = list(
    urns = "list",
    delta = "numeric",
    lambda = "numeric"
  ),
  contains = "Agent",
  methods = list(
    initUrns = function(figures, dict) {
      urns <- list()

      for(i in 1:length(figures)) {
        # initialize equal propensities
        # figure is stored in the first position of the list (cannot be a name)
        urns[[i]] <- c(figures[[i]], rep(1/length(dict), length = length(dict)))
        names(urns[[i]]) <- c("figure", dict)
      }
      return(urns)
    },

    findUrn = function(figure) {
      for(i in 1:length(urns)) {
        if(identical(urns[[i]][[1]], figure)) {
          return(i)
        }
      }
    },

    updateUrns = function(figure, communicate, point) {
      # no penalty
      urns[[findUrn(figure)]][[communicate]] <<- ({
        urns[[findUrn(figure)]][[communicate]]*(1-delta) + delta*point
      })
    },

    calcProbs = function(urn) {
      es <- exp(lambda*urn)
      probs <- es/sum(es)
      return(probs)
    },
```

```
46
47      updateSplit = function(figure, communicate, point) {
48        updateUrns(figure, communicate, point)
49        split <- list()
50
51        for(i in 1:length(urns)) {
52          urn <- unlist(urns[[i]][-1])
53          figure <- urns[[i]][[1]]
54          probs <- urn/sum(urn)
55          drawn <- sample(x = dict, size = 1, prob = calcProbs(urn))[[1]]
56          split[[drawn]] <- c(split[[drawn]], figure)
57        }
58
59        return(split)
60      }
61    )
62  )
63
64  # create player
65  initPlayer <- function(delta, lambda) {
66    player <- srlLearner$new(
67      split = list(),
68      score = 0,
69      urns = list(),
70      delta = delta,
71      lambda = lambda
72    )
73    return(player)
74  }
75
76  # set envirnment
77  setEnvironment <- function(figDims, dict, player1, player2) {
78
79    # initialize figures
80    figures <- initFigs(figDims)
81
82    # configure players
83    player1$split <- player1$makeSplit(figures, dict)
84    player1$urns <- player1$initUrns(figures, dict)
85
86    player2$split <- player2$makeSplit(figures, dict)
87    player2$urns <- player2$initUrns(figures, dict)
88
89    # set environment
90    env <- list()
91    env$figures <- figures
92    env$player1 <- player1
93    env$player2 <- player2
94    env$dict <- dict
95
96    return(env)
97  }
98
99  # override runSimulation
100 runSimulation <- function(iter, n, figDims, dict, isTwoWay, delta, lambda, raw = FALSE) {
```

```
  res <- list()
  for(i in 1:iter) {
    player1 <- initPlayer(delta = delta, lambda = lambda)
    player2 <- initPlayer(delta = delta, lambda = lambda)
    part <- playGame(n, figDims, dict, isTwoWay, player1, player2)
    res$player1[[i]] <- part$player1
    res$player2[[i]] <- part$player2
  }
  agg <- list()
  agg$player1 <- aggRes(res$player1, raw)
  agg$player2 <- aggRes(res$player2, raw)
  return(agg)
}

# override runSimulationMeans
runSimulationMeans <- function(iter, iterSim, n, figDims, dict, isTwoWay, delta, lambda, raw = FALSE) {
  res <- list()
  for(i in 1:iter) {
    part <- runSimulation(iterSim, n, figDims, dict, isTwoWay, delta, lambda)
    res$player1[[i]] <- part$player1$mean
    res$player2[[i]] <- part$player2$mean
  }
  agg <- list()
  agg$player1 <- aggRes(res$player1, raw)
  agg$player2 <- aggRes(res$player2, raw)

  return(agg)
}
```

Herrnstein Reinforcement Learning with Nature

```
#####
## learning agents - basic Herrnstein RL, full memory
#####

hrlLearnerNat <- setRefClass(
  "hrlLearnerNat",
  fields = list(
    urns = "list",
    split = "list",
    score = "numeric"
  ),
  contains = "hrlLearner",
  methods = list(
    givePointNat = function(figureDrawn, figureRec) {
      for(cat in split) {
        if(isIn(cat, figureDrawn)) {
          return(isIn(cat, figureRec))
        }
      }
      return(FALSE)
    }
  )
)

```

```
# override initPlayer
initPlayer <- function() {
  player <- hrlLearnerNat$new(
    split = list(),
    score = 0,
    urns = list()
  )
  return(player)
}

# create Nature
initNature <- function() {
  nature <- hrlLearnerNat$new(
    split = list(),
    score = 0,
    urns = list()
  )
  return(nature)
}

# override setEnvironment
setEnvironment <- function(figDims, dict, player1, player2, wp, wn) {
  # initialize figures
  figures <- initFigs(figDims)

  # configure players
  player1$split <- player1$makeSplit(figures, dict)
  player1$urns <- player1$initUrns(figures, dict)

  player2$split <- player2$makeSplit(figures, dict)
  player2$urns <- player2$initUrns(figures, dict)

  # set environment
  env <- list()
  env$figures <- figures
  env$player1 <- player1
  env$player2 <- player2
  env$dict <- dict
  env$nature <- initNature()
  env$nature$split <- env$nature$makeSplit(figures, dict)
  env$wp <- wp
  env$wn <- wn

  return(env)
}

# override oneRound
oneRound <- function(env, isTwoWay) {

  # player1 starts as a sender
  drawn1 <- drawFigure(env$figures)
  signal1 <- env$player1$sendCommunicate(drawn1)
  action1 <- env$player2$sendFigure(signal1)
  point1p <- env$player1$givePoint(signal1, action1)
  point1n <- env$nature$givePointNat(drawn1, action1)
```

```
  point1 <- env$wp*point1p + env$wn*point1n
  env$player2$updateScore(point1)

  # switch roles
  drawn2 <- drawFigure(env$figures)
  signal2 <- env$player2$sendCommunicate(drawn2)
  action2 <- env$player1$sendFigure(signal2)
  point2p <- env$player2$givePoint(signal2, action2)
  point2n <- env$nature$givePointNat(drawn2, action2)
  point2 <- env$wp*point2p + env$wn*point2n
  env$player1$updateScore(point2)

  # update splits
  # first round - P1: sender, P2: receiver
  # sender learns by coordination
  env$player1$split <- env$player1$updateSplit(action1, signal1, point1p)
  # sender learns by nature
  env$player1$split <- env$player1$updateSplit(drawn1, signal1, point1n)
  # receiver learns by coordination
  env$player2$split <- env$player2$updateSplit(action1, signal1, point1p)

  # second round - P1: receiver, P2: sender
  # sender learns by coordination
  env$player2$split <- env$player2$updateSplit(action2, signal2, point2p)
  # sender learns by nature
  env$player2$split <- env$player2$updateSplit(drawn2, signal2, point2n)
  # receiver learns by coordination
  env$player1$split <- env$player1$updateSplit(action2, signal2, point2p)

  return(env)
}

# override playGame
playGame <- function(n, figDims, dict, isTwoWay, player1, player2, wp, wn) {
  env <- setEnvironment(figDims, dict, player1, player2, wp, wn)
  avgScores <- list(player1 = c(), player2 = c())

  for(i in 1:n) {
    env <- oneRound(env, isTwoWay)
    avgScores$player1 <- c(avgScores$player1, mean(env$player1$score))
    avgScores$player2 <- c(avgScores$player2, mean(env$player2$score))
  }

  return(avgScores)
}

# override runSimulation
runSimulation <- function(iter, n, figDims, dict, isTwoWay, wp, wn, raw = FALSE) {
  res <- list()
  for(i in 1:iter) {
    print(i)
    player1 <- initPlayer()
    player2 <- initPlayer()
    part <- playGame(n, figDims, dict, isTwoWay, player1, player2, wp, wn)
    res$player1[[i]] <- part$player1
```

```
    res$player2[[i]] <- part$player2
  }
  agg <- list()
  agg$player1 <- aggRes(res$player1, raw)
  agg$player2 <- aggRes(res$player2, raw)
  return(agg)
}

# override runSimulationMeans
runSimulationMeans <- function(iter, iterSim, n, figDims, dict, isTwoWay, wp, wn) {
  res <- list()
  for(i in 1:iter) {
    part <- runSimulation(iterSim, n, figDims, dict, isTwoWay, wp, wn)
    res$player1[[i]] <- part$player1$mean
    res$player2[[i]] <- part$player2$mean
  }
  agg <- list()
  agg$player1 <- aggRes(res$player1)
  agg$player2 <- aggRes(res$player2)

  return(agg)
}
```

Smoothed Reinforcement Learning with Nature

```
#####
## learning agents - smoothed reinforcement learning
#####

srlLearnerNat <- setRefClass(
  "srlLearnerNat",
  fields = list(
    urns = "list",
    delta = "numeric",
    lambda = "numeric"
  ),
  contains = "srlLearner",
  methods = list(
    givePointNat = function(figureDrawn, figureRec) {
      for(cat in split) {
        if(isIn(cat, figureDrawn)) {
          return(isIn(cat, figureRec))
        }
      }
      return(FALSE)
    }
  )
)

# override initPlayer
initPlayer <- function(delta, lambda) {
  player <- srlLearnerNat$new(
    split = list(),
    score = 0,
    urns = list(),
```

```
    delta = delta,
    lambda = lambda
  )
  return(player)
}

# override runSimulation
runSimulation <- function(iter, n, figDims, dict, isTwoWay, delta, lambda, wp, wn, raw = FALSE) {
  res <- list()
  for(i in 1:iter) {
    print(i)
    player1 <- initPlayer(delta = delta, lambda = lambda)
    player2 <- initPlayer(delta = delta, lambda = lambda)
    part <- playGame(n, figDims, dict, isTwoWay, player1, player2, wp, wn)
    res$player1[[i]] <- part$player1
    res$player2[[i]] <- part$player2
  }
  agg <- list()
  agg$player1 <- aggRes(res$player1, raw)
  agg$player2 <- aggRes(res$player2, raw)
  return(agg)
}

# override runSimulationMeans
runSimulationMeans <- function(iter, iterSim, n, figDims, dict, isTwoWay, delta,
                               lambda, wp, wn) {
  res <- list()
  for(i in 1:iter) {
    print(i)
    part <- runSimulation(iterSim, n, figDims, dict, isTwoWay, delta, lambda, wp, wn)
    res$player1[[i]] <- part$player1$mean
    res$player2[[i]] <- part$player2$mean
  }
  agg <- list()
  agg$player1 <- aggRes(res$player1)
  agg$player2 <- aggRes(res$player2)

  return(agg)
}
```

References

Barrett, J. and Zollman, K. (2009), 'The role of forgetting in the evolution and learning of language', *Journal of Experimental & Theoretical Artificial Intelligence* pp. 293–309.

Bergstrom, C. T. and Lachmann, M. (1998), 'Signaling among relatives. iii. talk is cheap', *Proceedings of the National Academy of Sciences* **95**(9), 5100–5105.

Chater, N. and Vitanyi, P. (2003), 'Simplicity: a unifying principle in cognitive science?', *Trends in Cognitive Sciences* **7**, 19–22.

Christiansen, M. H. and Chater, N. (2008*a*), 'Brains, genes, and language evolution: A new synthesis', *Behavioral and Brain Sciences* **31**(05), 537–558.

Christiansen, M. H. and Chater, N. (2008*b*), 'Language as shaped by the brain', *Behavioral and brain sciences* **31**(05), 489–509.

Feldman, J. (2000), 'Minimization of Boolean complexity in human concept learning', *Nature* **407**, 630–633.

Gintis, H., Smith, E. A. and Bowles, S. (2001), 'Costly signaling and cooperation', *Journal of theoretical biology* **213**(1), 103–119.

Goodwin, G. P. and Johnson-Laird, P. N. (2013), 'The acquisition of boolean concepts', *Trends in cognitive sciences* **17**(3), 128–133.

Hespanha, J. P., Ateskan, Y. S. and Kizilocak, H. (2000), Deception in non-cooperative games with partial information, *in* 'Proceedings of the 2nd DARPA-JFACC Symposium on Advances in Enterprise Control', pp. 1–9.

Huttegger, S. M., Skyrms, B., Smead, R. and Zollman, K. J. (2010), 'Evolutionary dynamics of lewis signaling games: signaling systems vs. partial pooling', *Synthese* **172**(1), 177–191.

Lewis, D. (1969), *Convention: A Philosophical Study*, Harvard University Press.

Oktaba, K. and Kalociński, D. (2018), Socio-environmental constraints in category learning, *in* 'The Evolution of Language: Proceedings of the 12th International Conference'. To appear.

Polnaszek, T. J. and Stephens, D. W. (2013), 'Why not lie? costs enforce honesty in an experimental signalling game', *Proceedings of the Royal Society of London B: Biological Sciences* **281**(1774).

Roth, A. E. and Erev, I. (1995), 'Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term', *Games and economic behavior* **8**(1), 164–212.

Shepard, R., Hovland, C. and Jenkins, H. (1961), 'Learning and memorization of classifications', *Psychological Monographs: General and Applied* **75**.

Skyrms, B. (2009), 'Evolution of signalling systems with multiple senders and receivers', *Philosophical Transactions of the Royal Society* **364**, 771–779.

Sobel, J. (2007), 'Signaling games', *Encyclopedia of Complexity and System Science, Springer, forthcoming* .

Steels, L. (1995), 'A self-organizing spatial vocabulary', *Artificial life* **2**(3), 319–332.

Wallentin, M. and Frith, C. D. (2008), 'Language is shaped for social interactions, as well as by the brain', *Behavioral and brain sciences* **31**(05), 536–537.

Wittgenstein, L. (1953), *Philosophical Investigations. (Translated by Anscombe, G.E.M.)*, Basil Blackwell.

Zubek, J., Denkiewicz, M., Barański, J., Wróblewski, P., Rączaszek-Leonardi, J. and Plewczynski, D. (2017), 'Social adaptation in multi-agent model of linguistic categorization is affected by network information flow', *PLOS ONE* **12**(8), 1–25.
URL: *https://doi.org/10.1371/journal.pone.0182490*

List of Tables

List of Figures