TWO APPROACHES FOR COLLECTIVE LEARNING WITH LANGUAGE GAMES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÇAĞLAR GÜLÇEHRE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF COGNITIVE SCIENCE

FEBRUARY 2011

Approval of the Graduate School of Informatics

Prof. Dr. Nazife Baykal
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Deniz Zeyrek Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Cem Bozşahin
Supervisor

Examining Committee Members

Prof. Dr. Deniz Zeyrek	(METU, COGS)	
Assoc. Prof. Dr. Cem Bozşahin	(METU, COGS)	
Assoc. Prof. Dr. Ferda Nur Alpaslan	(METU, CENG)	
Dr. Cengiz Acartürk	(METU, COGS)	
Dr. Murat Perit Çakır	(METU, COGS)	

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ÇAĞLAR GÜLÇEHRE

Signature :

ABSTRACT

TWO APPROACHES FOR COLLECTIVE LEARNING WITH LANGUAGE GAMES

Gülçehre, Çağlar
M.Sc., Department of Cognitive Science
Supervisor : Assoc. Prof. Dr. Cem Bozşahin

February 2011, 89 pages

Recent studies in cognitive science indicate that language has an important social function. The structure and knowledge of language emerges from the processes of human communication together with the domain-general cognitive processes. Each individual of a community interacts socially with a limited number of peers. Nevertheless societies are characterized by their stunning global regularities. By dealing with the language as a complex adaptive system, we are able to analyze how languages change and evolve over time. Multi-agent computational simulations assist scientists from different disciplines to build several language emergence scenarios. In this thesis several simulations are implemented and tested in order to categorize examples in a test data set efficiently and accurately by using a population of agents interacting by playing categorization games inspired by L. Steels's naming game. The emergence of categories throughout interactions between a population of agents in the categorization games are analyzed. The test results of categorization games as a model combination algorithm with various machine learning algorithms on different data sets have shown that categorization games can have a comparable performance with fast convergence.

Keywords: language game, artificial intelligence, emergence, machine learning

ÖZ

DİL OYUNLARI İLE KOLLEKTİF ÖĞRENME İÇİN İKİ YAKLAŞIM

Gülçehre, Çağlar
Yüksek Lisans, Bilişsel Bilimler Bölümü
Tez Yöneticisi : Doç Dr. Cem Bozşahin

Şubat 2011, 89 sayfa

Bilişsel bilimler alanındaki en son çalışmalar, dilin önemli bir sosyal fonksiyonu olduğunu göstermektedir. Dilin yapısı ve bilgi birikimi, alan genelindeki bilişsel işlemler ile birlikte insanlar arasındaki iletişimden doğmaktadır. Bir topluluktaki her birey sınırlı sayıdaki bireyler ile sosyal etkileşim içerisindedir. Buna rağmen toplumlar kendi içlerindeki geniş çaplı düzenlilikleri ile bilinmektedir. Dili karmaşık adaptif bir sistem olarak ele alarak; dilin zamanla nasıl evrimleştiğini ve değiştiğini gözlemleyebiliriz. Ajan-temelli sayısal simülasyonlar, bilim insanlarının gerçekleştirdiği farklı multidisipliner senaryolar, dilin ortaya çıkışını modelleme için yardımcı olmaktadır. Bu tezde, L. Steels'in isimlendirme oyunları ile ajan temelli simülasyonlarla yaptıklarından esinlenilerek; ajan temelli sistemlerde, nesneleri sınıflandırmak için pek çok simülasyon test edilmiştir. Ajanlar arasındaki etkileşimler sonucunda, dil oyunlarının makineli öğrenme için model birleştirme algoritması olarak kullanılması sonucunda, kategorilerin verimli ve doğruluk oranı yüksek bir biçimde ortaya çıkışı değişik veri setleri ile beraber analiz edilip, tezde sunulmuştur.

Anahtar Kelimeler: dil oyunları, yapay zeka, makineli ögrenme, emerjans

To my family . . .

ACKNOWLEDGMENTS

I would like to present my deepest thanks to my thesis supervisor Dr. Cem Bozşahin for his valuable guidance, motivation and support throughout this thesis study.

My special thanks to my colleague Murat Soysal and my manager Serkan Orçan at TÜBİTAK-ULAKBİM for their help, support and patience to complete this work and to all my friends who gave me support whenever I needed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

"*When we study human language, we are approaching what some might call the 'human essence,' the distinctive qualities of mind that are, so far as we know, unique to man and that are inseparable from any critical phase of human existence, personal or social. Hence the fascination of this study, and, no less, its frustration. The frustration arises from the coming to grips with the core problem of human language, which I take to be this: having mastered a language, one is able to understand an indefinite number of expressions that are new to one's experience, that bear no simply physical resemblance and are in no simple way analogous to the expressions that constitute one's linguistic experience; and one is able ... to produce such expressions on an appropriate occasion, despite their novelty... The normal use of language is, in this sense, a creative activity. This creative aspect of normal language use is one fundamental factor that distinguishes human language from any known system of animal communication.*"

–Noam Chomsky

Language is the most complex social norm that the human species are able to learn and acquire. It is an important characteristic of human communication and social organization. Yet there are several open research questions about language and enigmatic nature of language has been engaging human mind since Pāṇini. Investigating how language is formed, emerged or transmitted can give important hints about some essential cognitive aspects[1] of human mind as well. Besides that, Sapir-Whorf hypothesis brings the relation between the language and mind further and it suggests that how human behaves or thinks are shaped by their languages (Kay and Kempton, 1984).

[1] i.e.: language acquisition, language comprehension etc.

Language apparently has an important social function. As a communication tool, it is fundamentally used for interactions between individuals, and in order to adapt to the evolving society, it slightly changes from generation to generation. Although social interactions can sometimes exhibit chaotic characteristics, in reality they are often characterized by shared cooperative activity (Bratman, 1992), or joint actions (Clark, 1996). Joint actions depend on shared cognition, that's a human being's recognition that she can share beliefs and intentions with other humans (Beckner et al., 2009). As a result of these social interactions, languages emerge and change. Human languages known to exhibit diachronic properties such as language evolution, language change, creolization, pidginization and emergence of new languages. "Nicaraguan Sign Language" is a very recent and unique case that linguists had a chance to witness the emergence of a new language. Diachronic properties of human languages have several similarities with evolutionary characteristics of living organisms and it's worthwhile to note that a living organism is stable only when it is dead.

The basic idea of Language is Complex Adaptive System (LCAS) is that a community of language users (or agents) can be viewed as a complex adaptive system which collectively solves the problem of developing a shared communication system. To do so, the community must agree on a repertoire of forms (e.g.: a sound system), a repertoire of meanings (the conceptualizations of reality), and a repertoire of form-meaning pairs (the lexicon and grammar) (Beckner et al., 2009).

Languages change over time according to complex interactions among the individuals of a population. As a result of these complex interactions, individuals establish common conventions which leads to convergence of their lexicons and linguistic coherence. Thus we can claim that language is a dynamically evolving complex system. Besides, it is a self-organizing system that adapts to the changes in ecosystem and society. Despite the ongoing attempts of modeling the language for the sake of learning insights about how a language evolves and how it is emerged, the question of how new languages emerge is still an unanswered question. But the recent advancements in multi-agent simulations of language emergence helped scientists for revealing some mysteries behind this though problem.

Collective and organizational activities between agents in a computational simulation reveal how language organizes and changes in a constrained environment. Hence understanding how language evolves or emerges will help us to understand how the social conventions among the

societies established. This can help us to build better AI systems that use a population of agents which can adapt to the changes while these agents are trying to name objects. The goal of these multi-agent simulations is to arrive to a common convention among the agents. This is very similar to the agents' interactions in the language games for aligning their lexicons in order to establish consensus on a specific word.

Complex adaptive systems (CAS) are a subset of non-linear dynamical systems in which agents (such as ants, cells ... etc) in a dynamic network constantly act and react what the other agents in the network are doing. CAS has just become an important field for the multidisciplinary studies in natural and social sciences. According to mathematicians and physicists, the interest towards complex adaptive systems lies behind the dynamics of how complexity emerges from extremely simple rule systems. For biologists, it is the conception that the natural selection is not the only source of organization in nature. In the social sciences, it is suggested that emergence[2] has a comparable impact on establishing social conventions (Lansing, 2003). To illustrate this concept, consider an immune system which also lacks centralized control and can't decide on a permanent, fixed structure; instead it must be able to adapt to unknown invaders. Yet despite its adaptive nature, a person's immune system is coherent enough to distinguish oneself from anyone else; it will attack cells from any other human. Immune systems, cities, and ecosystems share certain properties that make it useful to consider the instances of a class of phenomena which J. H. Holland calls complex adaptive systems (Holland, 1992). There are strong evidences that human languages are complex adaptive systems (Beckner et al., 2009; Steels, 2000). This characteristic of language, enables it to adapt to the changes in social domains and the environment.

Learning insights of language emergence and evolution will reveal important mysteries about human mind and evolution. Beyond that it will also have important application areas as well. Autonomous artificial agents which need to coordinate their activity in open-ended environments could make use of these mechanisms to develop and continuously adapt their communication systems. On the other hand, understanding how language develops and evolves will enable us to develop technological artifacts that exhibit human-level language understanding and production (Steels, 2000).

Algorithms inspired from nature and biology are commonly used for solving complex and NP-

[2] the idea that how complex global patterns with new properties can emerge from local interactions without any central control

hard problems in computer science. Many natural phenomena are known to exhibit chaotic behaviors. Artificial intelligence and specifically machine-learning tries to solve complex problems that occur as a result of complex processes. Sometimes creating a purely abstract mathematical solution for AI problems is not feasible. Therefore computer scientists frequently use nature inspired algorithms for AI and particularly for optimization problems, such as the traveling salesman problem by using, genetic algorithms, connectionist systems and ant-colony optimizations.

Essentially, categorization is a collective and social activity. There are cultural differences in categories of things between societies.[3] According to prototype theory, categories are prototypes of reality in human mind and it is mode of graded categorization (Rosch, 1999). For instance when somebody ask you to give an example of furniture concept, it is more likely that you will say chair rather than the office desk. But these prototypes change between different cultures. How and why do they change is an important research question that might have an answer by building simulations of categorization process.

In this thesis we have shown that, it is possible to create better AI algorithms by using the CAS models[4] in computational linguistics. We have tested two simulations of categorization of objects by using a specific type of language game, "categorization game". In a nutshell, categorization game is a collective linguistic activity between different language users from different backgrounds. We have also tested these categorization games as model combination algorithms [5] by using with different machine learning algorithms on different data sets.

The remainder of this thesis is organized as follows:

Chapter 2: In this chapter we provide a background related to the topics in different types of language games, language emergence and the perspective that assumes language as a complex adaptive system.

Chapter 3: In this chapter we give a background about various computational learning techniques and discuss about some philosophical problems related to them.

Chapter 4: In this chapter we present the methods and experiments that were conducted and give the empirical results obtained from these experiments.

---

[3] e.g.: colors is one of the example (Baronchelli et al., 2010).

[4] e.g.: language emergence simulations.

[5] Different weak learning algorithms are competing against each other to agree on a category for an object.

Chapter 5: In this chapter we give an overview of contributions of this thesis, discuss about some issues in the current model with possible future improvements.

CHAPTER 2

Background on Language as a Complex Adaptive System

"*A number of blind men came to an elephant. A king told them that it was an elephant. The king asked, 'What is the elephant like?' and they began to touch its body. One of them said: 'It is like a pillar.' This blind man had only touched its leg. Another man said, 'The elephant is like a husking basket.' This person had only touched its ears. Similarly, he who touched its trunk or its belly talked of it differently. Later king explained:*

All of you are right. The reason every one of you is telling it differently is because each one of you touched the different part of the elephant. So, actually the elephant has all the features you mentioned."

–An Eastern Myth[1]

Complex Adaptive Systems (CAS) are the systems that involve many components which adapt or learn as they interact. The study of CAS poses some unique challenges: Some of our most powerful mathematical tools, particularly methods involving fixed points, attractors, and the like, are of limited help in understanding the development of CAS. CAS is at the heart of the several important contemporary problems (Holland, 2006). The state of art technique for analyzing CAS is to build simulations that is modeling the simplified version of the system. In this section we have surveyed different types of language games which is the simplified version of social interactions between individuals.

[1] This story takes place in several different eastern myths such as in Jain, Buddhist, Sufi and Hindu religious philosophy.

2.1 Language Evolution and Language Change from a Computational Perspective

How language emerges, evolves and changes is still an important challenge for linguistics. Language change is a fundamental reality in the language; but how it is evolving is still being questioned. With the advances in computational modeling and simulations, we are able to see the problem from different perspectives. There have been significant amount of increase in the studies about the language evolution. Language evolution is actually a result of cultural evolution which must not be confused with biological evolution. Although cultural evolution and genetic evolution exhibit similar characteristics, cultural evolution seems to exhibit more complex behaviors because of the irrationality of human being (Traulsen et al., 2009).

Variations in language occurs in two levels (Niyogi, 2006):

- **Synchronic Level**: Synchronic variations occur across individuals in space at any fixed point in time. An example of these kinds of variations could be the differences between dialects of a language.
- **Diachronic Level**: Diachronic variations are the variations in the language of spatially localized communities over time.

Evolution of languages seems to be effected by three distinct but interacting adaptive systems (Christiansen and Kirby, 2003):

- Individual Learning
- Cultural transmission
- Biological evolution

Adaptive systems involve in the transformation of information in such a way that it always fits an objective function (Christiansen and Kirby, 2003). This is best illustrated in the case of biological evolution in which natural selection is the mechanism of adaptation par excellence. Variations in the transmitted *genotype* are selected for in a way that the resulting *phenotype* best fits the function of survival and reproduction. Equivalently, individual learning can be

thought of as a process of adaptation of the individual's knowledge. Furthermore Steels and MacIntyre (1998) suggests that language evolves in time in all levels.

The most disputed part of the complex adaptive systems is the notion of adaptation through cultural transmission which is also known as *'glossogeny'*. The knowledge of particular languages persists throughout the time only by being repeatedly used to generate linguistic data, and this data is used as an input to the learner - a type of cultural evolution known as iterated learning. In that sense, one can think of the adaptation of languages themselves to fit the needs of the language user, and more importantly, to the language learner (Christiansen and Kirby, 2003).

When we talk of language evolution, we are usually referring to the evolution in three different timescales:

- The lifetime of an individual.
- The lifetime of a language.
- The lifetime of the species.

What is particularly interesting about language, and why its emergence on earth can be seen as a major transition in evolution, is that there are interactions between "individual learning", "biological evolution" and the "cultural transmission". The learner's mental capacity and structure is effected by the outcome of biological evolution. Likewise, the demands on linguistic transmission are partially determined by the learner's genetically given biases.

2.2 Emergence of Language

Emergence is generally a hard to grasp philosophical conception, and there is no precise definition of complexity and emergence in the literature. There are several different definitions of emergence, and there is a widespread criticism that these definitions frequently are either conflated or abused by the scientists using the term. We clarified our notion of emergence in the following section.

2.2.1 Emergence

Emergence has been an important topic in philosophy for a long time (Chalmers, 2006; Lovejoy, 1927; Klee, 1984; Pepper, 1926; Morgan, 1929) and it is studied extensively from the perspective of complex systems as well. The first notable mention of emergence comes from Aristotle in Metaphysics (Wikipedia, 2010c):

".. the totality is not, as it were, a mere heap, but the whole is something besides the parts ... "

Usually emergence is distinguished and studied in two different fields (Chalmers, 2006):

Strong Emergence : A high-level phenomenon is strongly emergent with respect to a low-level domain when high-level phenomenon stems from the lower-level domain. But the truths concerning higher phenomenon can not be deduced from the lower-level. It is hard to find strong emergence in the nature, because of the implicit downward causation in it. Chalmers claims that the only strongly emergent phenomenon in the nature is consciousness (Chalmers, 2006) and the facts of consciousness are not deducible from physical facts.

Weak Emergence : Weak emergence eventuates when a high-level phenomenon arises from the lower-level domain, and the truths related to the high-level phenomenon would be unexpected given the principles of the low-level domain. This is the kind of emergence that we use when we discuss about the emergence of language throughout this thesis. There are a few core examples of weak emergence (Chalmers, 2006):

- The game of life: From simple low-level rules, complex high-level patterns emerges.
- Connectionist networks: High-level 'cognitive' behavior emerges from simple interactions between simple threshold logic units.
- Evolution: Complex features of biological organisms emerge from simpler lower level features with genetic mutations, recombination and natural selection.

All those cases can be deduced from the lower-level components.

2.2.2 Language Emergence

The exact details of language emergence is problematic, because the system is extremely non-linear. Computational models of language emergence is a plausible approach for learning insights about the emergence of language.

Different components of language are usually studied by different perspectives of emergence framework as discussed by MacWinney (1998):

- **Emergent Syntax:** The belief that the syntax is an autonomous, innate species-specific characteristic is highly questionable argument. Syntax demonstrates the mosaic nature of language change with the use of preexisting neurocognitive components (Schoenemann and Wang, 1996). There are several computational scenarios for emergence of syntax. For example in Talking Heads Experiment, Steels (1998) studied the emergence of syntax with the visually grounded robotic agents.

- **Emergent Semantics:** This is the emergence of semantics from simple observations from bottom-up processes rather than top-down processes that the concepts are imposed to the agents (Staab et al., 2005). Staab et al. (2005) used emergent semantics to refer to a set of techniques and principles for analyzing the evolution of decentralized semantic structures in large scale distributed systems. Typical examples of this kind of emergence is folksonomy and collaborative tagging.

- **Emergence of Auditory Patterns:** This refers to the emergence of auditory patterns during child's development and during first language learning.

- **Emergence of Morphology:** This field tries to answer the question, how inflection marking of English verbs emerged. For example irregular forms of past tense verbs fell, knew, went...etc and regular forms like wanted, tried ...etc.

- **Emergence of Grammar:** According to Hopper (1998), grammar is not source of communication and understanding but by-product of it. In other words grammar is epiphenomenal.

Steels, Baronchelli, Loreto, Vogt and the other scientists from different disciplines who have used the multi-agent simulations for modeling the emergence of language observed the social

dynamical characteristics of language in order to observe how language emerges without any central control in a society. To analyze the characteristics of the emergence, they have designed several computational models and explored the meaning and the form association among population of agents. By studying the simulations of the large number of interactions, we are able to analyze under what conditions language emerges.

Iterated Learning Model for Emergence of Language Iterated Language Model is a tool for investigating the cultural evolution of language. Iterated learning Model is based on the hypothesis that some functional linguistic structure emerges inevitably from the process of iterated learning without the need for natural selection or explicit functional pressure (Smith et al., 2003).

2.3 Language Games

2.3.1 An Overview: Language-game

Wittgenstein (1953) takes on a totally different point of view in Philosophical Investigations (*PI*) from his outlook on language in Tractatus Logico-Philosophicus (*TLP*) (Wittgenstein, 1922). In TLP he insisted on ideal language philosophy, but in PI he changed his position towards an Ordinary Language Philosophy. Language game is a hypothetical game (or a thought experiment) proposed by Ludwig Wittgenstein (Biletzki and Matar, 2010) as a simplified language use of everyday language of individuals (Wittgenstein, 1953). By using language-game Wittgenstein tried to attract philosophers' and linguists' attention to the everyday use of language in order to bring back the language from the ivory towers of analytical philosophers. In Philosophical Investigations (*PI*), Wittgenstein refers the term of language-games repeatedly in several different parts of the book. In PI II he starts describing it using a conversation between the builders:

" The language is meant to serve for communication between a builder A and an assistant B. A is building with building-stones: there are blocks, pillars, slabs and beams. B has to pass the stones, in the order in which A needs them. For this purpose they use a language consisting of the words "block", "pillar", "slab", "beam". A calls them out; B brings the stone which he has learnt to bring at such-and-such a call."

On this conception of the philosophical enterprise, the vagueness of ordinary usage is not a problem to be eliminated but rather the source of linguistic richness. It is misleading even to attempt to fix the meaning of particular expressions by linking them referentially to things in the world. The semantics of a word or phrase or proposition is nothing other than the set of (informal) rules governing the use of the expression in actual life.

Like the rules of a game, Wittgenstein suggested that, these rules for the use of ordinary language are neither right nor wrong, neither true nor false: they are merely useful for the particular applications in which we employ them. The individuals of any community develop ways of speaking that meet their needs as a group, and these constitute the language-game that they employ. Human beings at large constitute a greater community within which similar, though more widely-shared, language-games are played. Although there is little to be said in general about language as a whole, thereof, it may often be fruitful to consider in detail the ways in which particular portions of the language are used (Kemerling, 2001).

Even the fundamental truths of arithmetic, Wittgenstein now supposed, are nothing more than relatively stable ways of playing a particular language game. This reasoning rejects both logical and intuitionist views of mathematics in favor of a normative conception of its use. $2 + 3 = 5$ is nothing other than a way we have collectively decided to speak and write, a handy, shared language-game (Kemerling, 2001).

What is a game? To better comprehend the computational models of language games, understanding the game-theoretical notion of game (Normal Form Games) is essential. In our depictions, a game should have the following three aspects (Easley and Kleinberg, 2010):

1. The game should have a population of agents or players.

2. Each agent has a set of options for how to behave; these are usually referred as the player's possible *strategies*.

3. For each strategy, each agent receives a payoff that can depend on the strategies selected by everyone. The payoffs will generally be numbers, with each agent preferring larger payoffs to smaller payoffs.

Formal definition of Normal Form Games: **Formal Definition**: A game in normal form is a structure

$G =< P, \mathbf{S}, \mathbf{F} >$ where:

$P = \{1, 2, \ldots, m\}$ is a set of players,

$\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$ is an m-tuple of pure strategy sets, one for each player, and

$\mathbf{F} = \{F_1, F_2, \ldots, F_m\}$ is an m-tuple of payoff functions.

There are two important challenges while constructing artificial communication systems by using language-games (De Beule et al., 2006):

- Avoiding homonymy: A term can not be associated with more than one category.
- Avoiding synonymy: Categories/names can not be associated with more than one term.

If there is too much homonymy and synonymy in the communication-system, our system can not be used effectively.

Jaeger et al. (2009) suggests that computational models of language games study and examine the role of embodiment, communication, cognition and social interaction in the emergence of language. A typical language game is played between two different agents (usually denoted as speaker and hearer) within a shared world that involves some form of communicative signs. When speaker and hearer shares the same meaning to a particular sign in the world, they will use the existing items in their inventory in a routine way. Otherwise the speaker should be able to create new words, and the hearer should be able to extend its knowledge base with the new item explored by the speaker.

2.3.2 Semiotic Dynamics

Semiotic dynamics is the collective effort of a population of agents or individuals to create a common semiotic system to use for their communication or information organization (Steels, 2006). Large-scale online social communities which use tagging (e.g.: facebook, delicious, flickr etc.) are ubiquitous examples of semiotic dynamics. But semiotic dynamics also occurs in natural language as well. Cattuto et al. (2007) investigated semiotic dynamics with online

collaborative tagging. Steels and Kaplan (1999) have studied semiotic dynamics for emergence of lexicons in robotic agents. Baronchelli et al. (2005) also did an extensive research on statistical mechanics of semiotic dynamics in naming game.

2.3.3 Varieties of Language Games

Wittgenstein's thought experiment influenced many scientists and motivated them to work on different theories of language-games. The investigation of different language games led to different explorations in field.

2.3.3.1 Naming Game

Naming game is a very well studied kind of language game. The theory of naming game is based on the assumption that language evolves and changes from generation to generation. The popularity of naming game emanates from its simplicity and expressive power.

Naming game was conceived to explore the role of self-organization in the evolution of language (Steels, 1995). Steels, in his early studies, such as (Steels, 1995), focused primarily on the formation of vocabularies, i.e. a set of mappings between words and meanings (for instance, physical objects). In this context, each agent develops its own vocabulary in a random private fashion. But agents are forced to align their vocabularies in order to obtain the benefit of cooperating through communication (Baronchelli et al., 2007). Thus, a globally shared vocabulary emerges, or should emerge, as a result of local adjustments of individual word-meaning association.

Agents in naming game has two fundamental roles: speaker and hearer. Steels in (Steels, 1995) called the agent starting the dialog as *initiator* and the agent listening to the initiator as *receiver*. These pairs of agents are drawn from a population of agents randomly. Then the speaker identifies an object by using a name (Steels and MacIntyre, 1998). The game is successful if both agents agree on a particular name and the game is considered as a failure if otherwise. The game is adaptive if both agents are able to change their rules (e.g.: object-name relations) to be more successful in the forthcoming games. There is no global or central control in the game. The game continues until all the agents in the game establish a global consensus with "microscopic" interaction rules on naming the object (Baronchelli et al., 2006;

De Vylder and Tuyls, 2006).

Because different agents can each invent a different name for the previously named object, synonymy is unavoidable (Baronchelli et al., 2008). But we do not consider the case of homonymy in naming game. The probability of getting homonymy at the end of game is arbitrarily small. Since the number of possible words is so large that the probability that two different players will ever invent the same word at two different times for two different meanings is practically negligible (Baronchelli et al., 2008).

Each agent in the game can be described by its inventory –a set of form-meaning pairs– (in our case names are competing to name the object). In the beginning, the inventory is empty ($t = 0$) and evolves dynamically as time passes. At each time-step ($t = 1, 2, 3...$) agents interact with each other (Baronchelli et al., 2008). Interaction rules are as follows:

- The speaker transmits a name to the hearer. If the inventory of the speaker is empty, it invents a new name, otherwise it randomly selects a name from its repository.
- If the hearer has the name in its inventory, then the game will be **successful**. The hearer and speaker will delete all the alternatives names in their inventory and only the winning one will be left.
- If the hearer does not have the name in its inventory, then the game will be **failure**. The hearer will simply add the name to its inventory.

The interaction rules are visualized in Figure 2.1.

The flowchart for generalized model of the naming game is shown in Figure 2.2.

Formal Definition of the Model The mathematical model for naming game we are going to discuss is based on Steels and MacIntyre (1998) and more recently to the study in Lenaerts et al. (2005).

Consider that a population of agents $\mathcal{A}$ with size $\mathcal{N}_{\mathcal{A}}$ where each agent $a_i \in \mathcal{A}$ is surrounded with a set of objects $O_a = \{o_0, ..., o_n\}$ of size $\mathcal{N}_O$. The state of i^{th} agent consists of a set of associations $\mathcal{D}_a$ between objects in the environment and the features of the objects that discriminates it from other objects d^a_j.

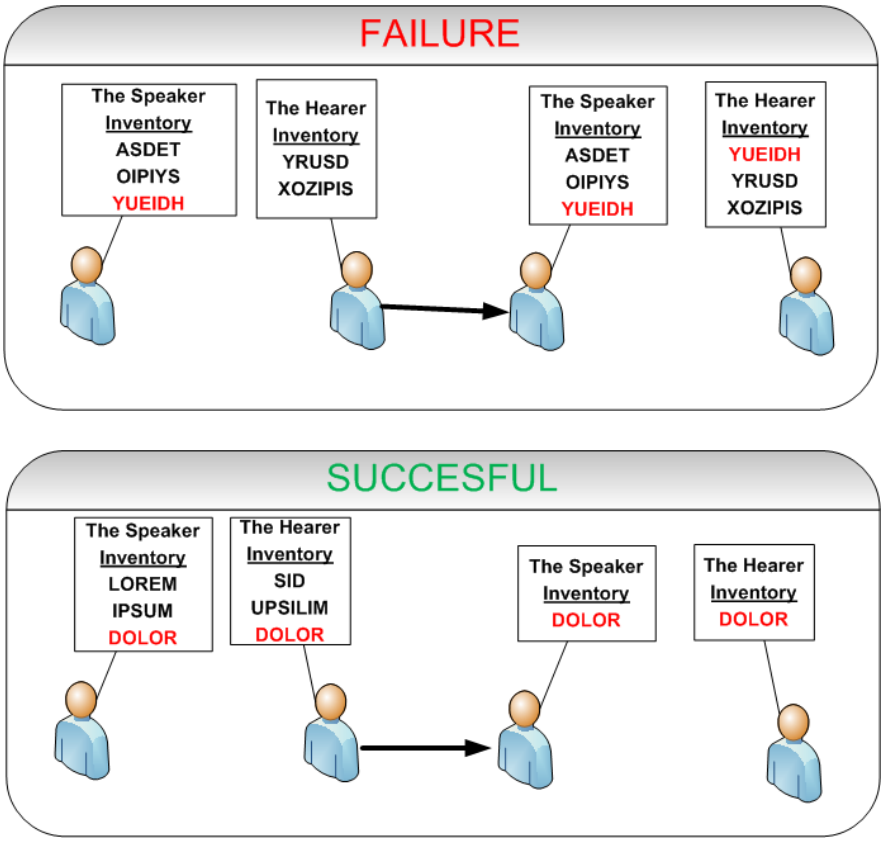

Figure 2.1: **Interaction Rules**: In case of failure the speaker's inventory contains three words: *ASDET*,*OIPIYS* and *YUEIDH*.The speaker utters the name *YUEIDH*, but hearer does not have this name in its inventory. Therefore it removes all the names and add the name that speaker uttered to its inventory. If the hearer has the name that speaker uttered, speaker and hearer will remove all the names in their inventory except the winning one.

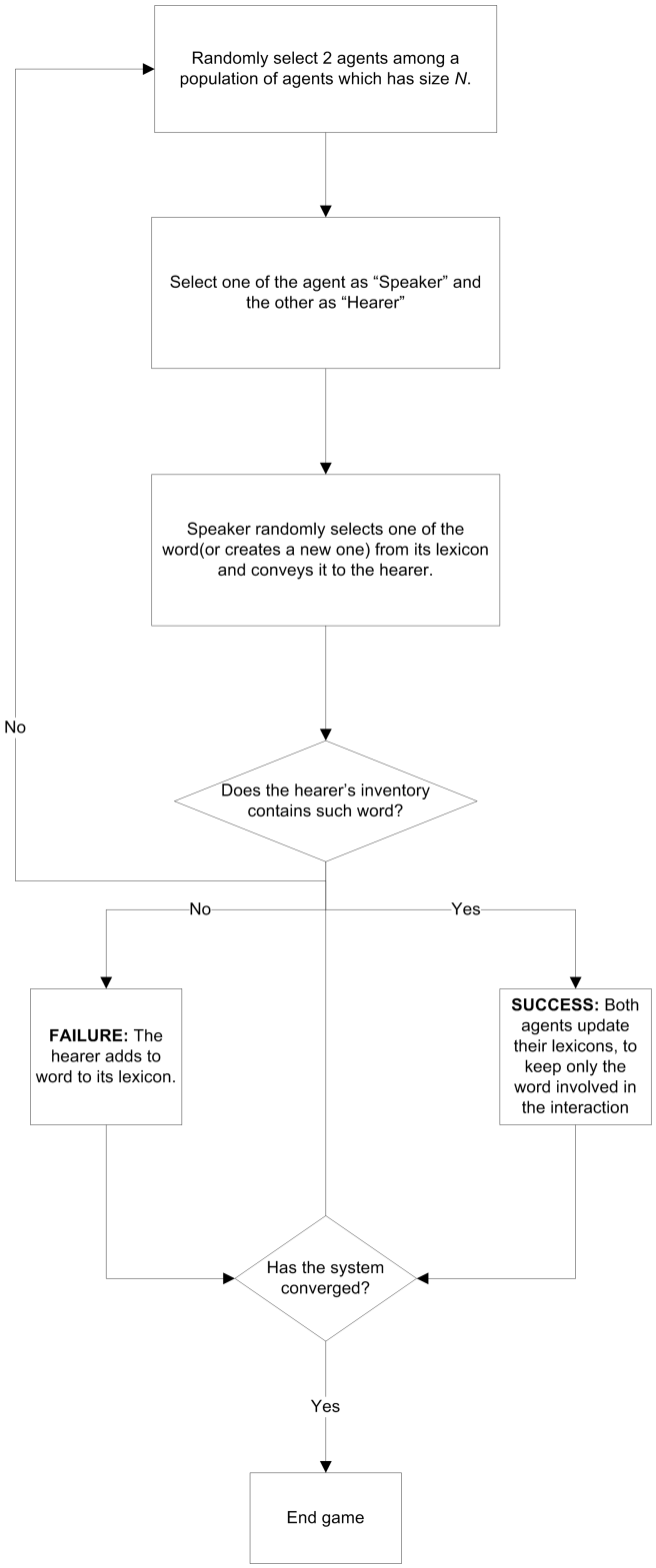

Figure 2.2: **Naming Game Flowchart**

$\mathcal{D}_a = \{(o_0, d_0^a), (o_0, d_1^a), ..., (o_2, d_l^a), ..., (o_j, d_j^a), ...\}$

$\mathcal{D}_a$ includes ambiguous pairings of features and objects. A typical example for the features might be size, shape and the color of different clothes. Each agent $a_i \in \mathcal{A}$ has its own lexicon $\mathcal{L}^a$ which is a set of associations between particular meanings d_j^a and words $w^a \in \mathcal{W}_a$.

$\mathcal{L}_a = \{(d_0^a, w_0^a), (d_0^a, w_1^a), ..., (d_j^a, w_2^a), ..., (d_k^a, w_j^a), ...\}$

$\mathcal{L}_a$ as well as $\mathcal{D}_a$ allows ambiguous pairings but only between words and meanings. By convention each agent $a_i \in \mathcal{A}$ uses the same association between words and meanings. Therefore $(D)_a = (D)$ for all agents. Moreover (D) with size (d) is finite as in (Steels and Kaplan, 1998).

For the sake of simplicity we will assume that the space of all words (w) is $\mathcal{W}$. Hence the number of meaning-word association will be $n = w \times d$. The lexicon is dynamic, therefore meaning and word associations changes in time. Each pair of (d_i^a, w_k^a) has a strength value v_{kl}^a where $v_{kl}^a \in [0, 1]$. Therefore the lexicon $\mathcal{L}_a$ can be represented with a matrix with rows and columns are specifying the strength of associations in $\mathcal{L}_a$:

$$\begin{pmatrix} v_{00}^a & v_{01}^a & \cdots & v_{0w}^a \\ v_{10}^a & v_{11}^a & \cdots & v_{1w}^a \\ \vdots & \vdots & \ddots & \vdots \\ v_{d0}^a & v_{d1}^a & \cdots & v_{dw}^a \end{pmatrix}$$

$\mathcal{L}_a$ is a probability matrix and the state of an agent a is defined as:

$\phi^i = (\mathcal{D}, \mathcal{L}_a)$

(De Vylder and Tuyls, 2006) suggested that naming game always converges using a sampling-response model.

Naming Game Dynamics According to the studies of Baronchelli et al. (2008), the statistical dynamics of naming-games are analyzed in depth. Naming games has the characteristic time required by the system to reach convergence as $N^{1.5}$ where N is the number of agents.

2.3.3.2 Guessing Game

Likewise the naming game and guessing game is being played between a pair of agents; speaker and hearer. The guessing game has been used in Talking Heads experiment that is discussed (Steels and Kaplan, 2002, 1999). Previously similar games were developed in decision theoretical and game theoretical studies as well (Weber, 2003). But the guessing games to be discussed here will be based on Luc Steels' Talking heads experiment (Steels and Kaplan, 2002), (Steels and Kaplan, 1999).

Guessing Games and Talking Heads Experiment The Talking Heads experiment is a conceptual experiment for modeling how the lexicon emerged in the situated environment. Experiments are done at labs scattered around the world, with cameras that are connected to a computer. The cameras can either rotate horizontally and vertically. Agents are software installed on the computer that the camera is connected to. Agents are looking towards a board where several different shapes are located. As part of the game, agents play guessing game with the shapes located on the board and they are trying to arrive to a linguistic consensus. Agents can teleport themselves to another lab after the game is finished.

The game is played between a pair of visually grounded cognitive agents. Agents are capable to do image segmentation and object recognition through the camera. Also each agent can perceive different features of the object such as color, size, shape and location. The set of objects and the related data in the game are called *context* and the object that the speaker chooses is the *topic*. Rest of the objects in the environment form the *background*.

The speaker will give a linguistic cue to the hearer which is related to the object, for example: red square on the higher left corner, blue triangle on the lower left cornet etc. "Talking Heads" use their own language for communication instead of natural language. Exemplary, speaker by uttering "makarena" might intend [UPPER-LEFT CORNER LIGHT-RED].

Based on the speaker's cue, the hearer tries to guess the topic and it communicates its choice of object by pointing it. If both agents agree on the same word for the selected topic, the game succeeds. Otherwise the game fails. In case of failure the speaker points to the correct object that it had in its mind. Both agents repair their own knowledge-base in order to be more successful in upcoming games.

The agent architecture in Guessing Game has two important components:

The conceptualization module: This module is responsible for categorizing the reality to be able to apply categories for finding back the referent on the board or in the perceptual image.

Categories are basically the features of topic like, color, position, shape. Those categories are stored in a discrimination tree.

The Verbalization module: The lexicon of the agents consists of pairs of word-category pairs with weights associated with these pairs. If agents do not have any word for referent, they will invent their own words and when an agent utters a word related to a category, initially it checks its lexicon if it is already in the lexicon. If the category already exists, agent sorts the words referring to the object inside the lexicon according to their weights and then utters the word with the highest score. If the word does not exist in the agent's lexicon, it will create a new word for the object.

When a hearer receives a word for an object, it checks its lexicon and sorts the possible categories associated with the current context and it chooses the highest scored word. If the word is not in the hearer's lexicon it places the word there and the hearer will point to the object.

Guessing games and Cross Situational Learning (*CSL*) Multi-agent computer simulations that are used for language-change and language evolution are very common in scientific literature. The use of these models changed the view on language and directed towards the complex-adaptive system.

The formal studies on word-meaning acquisitions make very strong simplifications in the communication architecture of the agents. Specifically, they consider only single word utterances, and assume a meaning transfer: when a speaker utters a word, the hearer knows what the intended meaning is. These kind of simplifications greatly reduce the complexity of model and therefore the difficulty of the model for understanding the dynamics of language-emergence.

In *CSL* agents (Siskind, 1996), infer the meaning of words by monitoring the co-occurrence of words and their reference (semantics). Based on this assumption Bayesian models are developed in which *CSL* makes the assumption that as you hear a word in different contexts,

in time you will acquire its meaning, from the reference. This model is influenced by Quine's thought experiment regarding to the indeterminacy of translation (Smith, 2003).

De Beule et al. (2006)'s study tries to remove simplifications regarding to meaning-transfer in the language games. They created a guessing game in which $\mathcal{N}$ number of agents try to bootstrap a common lexicon from a set O of objects.

2.3.3.3 Observational Game

Observational game has been first proposed by Vogt (2002) [2] and later Vogt and Coumans (2003) did further investigations and comparisons with other types of language game models. Vogt tried to create a minimal language game for simulating verbal language evolution. Observational games use joint attention and associative Hebbian learning.

Initially two agents are chosen randomly from the population and arbitrarily one of them is selected as speaker. The other one takes the hearer role. The speaker informs the hearer about the referent which is the topic of the game, to establish joint attention. The speaker looks for words that are associated with the subject, and chooses the word-meaning association in which θ has the highest score. If the speaker can not find an appropriate association, it invents a new word and adds the meaning of the word association to the lexicon with an initial association score of $\theta = 0.01$. Then the speaker utters the word. The hearer checks its own lexicon for an association. If the hearer finds the right association, the game succeeds. Otherwise the game fails. According to the result of communication, the lexicon is updated:

a. If the game fails, the listener adds the word-meaning association to its lexicon with an initial association score of $\theta = 0.01$. The speaker reduces the used association score by $\theta = \eta \times \theta$, where $\eta = 0.9$ is a constant learning parameter.

b. If the game succeeds, both robots increase association score θ of $\theta = \eta \times \theta + 1 - \eta$ of the word-meaning association. They apply lateral inhibition on all the other participating associations with $\theta = \eta \times \theta$.

[2] In this study they have worked with autonomous robots for grounding meaning among them.

2.3.3.4 The Selfish Games

In selfish games, there is no non-verbal indication of topic in the game (Vogt and Coumans, 2003). Therefore the agents can not verify whether their communication is successful. They cannot use the association score as an indication of the effectiveness of a word. Hence the meaning of the utterance will be uncertain to the hearer, because there are many possible meanings in the context. Similar to the CSL with the guessing games, as the context changes game to game the cross-section of the contexts in co-occurrence with a particular word will constitute the meaning. Learning is done with *Bayesian Learner*. The association score is:

$\delta = P(m|w) = \frac{P(m)P(w|m)}{P(w)} = \frac{P(w \cap m)}{P(w)}$ where $P(m)$ is the probability of meaning m's occurrence and $P(w)$ is the probability of word w's occurrence.

and δ can be converted to *confidence probability* as done by Smith (2001):

$\delta = \frac{U(w \cap m)}{U(w)}$

where $U(w \cap m)$ is the co-occurrence frequency of meaning, and word and $U(w)$ is the co-occurrence frequency of word. In each game hearer and speaker increments $U(w \cap m)$ and $U(w)$ by 1.

2.3.3.5 Category Game

Category game has been studied extensively in (Baronchelli et al., 2008, 2007), (Baronchelli et al., 2010) and by Puglisi et al. (2008). The goal of the category game is to find out if the categories are in implicit structure of nature or emerges from complex interaction between the individuals in the environment (Puglisi et al., 2008).

The category game phenomena is very similar to applied to the color categorization game in (Baronchelli et al., 2008).

In a category game model, a population of N agents is committed to the categorization of a single analog perceptual channel. Each stimulus is a real number in the interval $[0, 1]$. Categorization is identified with a partition of the interval $[0, 1]$.

Agents have dynamical inventories of form-meaning associations that is linking perceptual

categories to the words representing the linguistic counterparts. The words in the lexicon evolve in time through the language games played.

Initially all N agents, have only the trivial perceptual category $[0, 1]$ in their lexicon. At each time step two agents are selected, and a scene with $M \leq 2$ stimuli (object/stimuli is denoted as O_i where $i \in [1, M]$) is presented. The speaker must recognize the scene and categorize an object. The hearer will try to guess the categorized object and based on their success or failure they will rearrange their form-meaning associations. The only parameter in the model is d_{min}. It is the **just noticeable difference** of the stimuli. d_{min} is inversely proportional to the perceptive resolution power of the agents. Therefore objects in the same *scene* should satisfy the inequality $|o_i - o_j| > d_{min}$ for all pair of (i, j). The way stimuli are randomly chosen, characterizes the kind of situated environment at the end of the game.

2.3.3.6 Discrimination Games

In discrimination games, the agent tries to distinguish one object or situation from others using sensors and low-level sensory processes. The goal of the discrimination game is to determine if an agent is capable of developing autonomously a repertoire of features to succeed in discrimination and the subsequent adaptation of the feature repertoire Steels (1996).

Formal Definition For the formal definition of the discrimination games we will adapt the terminology of Steels (1996).

There is a set of objects $O = \{o_1, o_2, \ldots, o_m\}$ and a set of sensory channels $S = \{\sigma_1, \ldots, \sigma_n\}$, being real-valued partial functions over O. Each function σ_j defines a value $0.0 \leq \sigma_j(o_i) \leq 1.0$ for each object o_i.

An agent α has a set of feature detectors $D_a = \{d_1^\alpha \ldots d_m^\alpha\}$. A *feature detector* is $d_k^\alpha = \langle p_k^\alpha, V_k^\alpha, \phi_k^\alpha, \sigma_j \rangle$ has an attribute name p_k^α, a set of possible values V_k^α, a partial function phi_k^α and a sensory channel σ_j. The result of applying a feature detector d_k^α to an object o_i is a feature written as a pair $(p_k^\alpha v)$ where p is the attribute name and $v = \phi_k^\alpha(\sigma_j(o_i)) \in V_k^\alpha$.

A discrimination game $d = \langle \alpha, o_t, C \rangle$ contains an agent α, a topic $o_t \in C \subseteq O$. C is the context of the game. If a distinctive set of features are found in the game, the outcome is success. Otherwise the game ends with failure.

2.4 Language as a Complex Adaptive System

Language has fundamentally a social function. Its origin and capacity depends on its role in social life. Social interactions can be uncooperative, and involve conflict, but in the end they are shared cooperative activities or joint actions. There are several mental attitudes for joint actions like planning and goal directed actions, which are commitment to help the others and above all joint beliefs (Beckner et al., 2009). Clark (1996) refers to the language use as a form of joint action, an action that is carried out by an ensemble of people acting in coordination with each other.

2.4.0.7 Language as a Social Software

Social software is an interdisciplinary study that uses formal tools to build social procedures: formal models of knowledge and beliefs, the dynamics of information in a multi-agent setting, the foundations of game theory and logics that may be used to prove correctness of certain social procedures. Parikh (1995) makes an analogy between computer systems and social systems, and compares the natural languages to programming languages. But natural languages are executed inside the minds of individuals. Hence he proposes that formal methods and game theoretical tools that are used for analyzing computer source codes can be used for analyzing natural languages as well (Pacuit and Parikh, 2006).

CHAPTER 3

Background on Computational Learning Techniques and Machine Reasoning

"*... the question of whether Machines Can Think, a question of which we now know that it is about as relevant as the question of whether Submarines Can Swim.*"

–Edsger W. Dijkstra

An important aspect of our study is to investigate the possibilities of using language games to improve classification performance. Therefore we have tested several machine learning algorithms with a variant of language game that we have created -categorization game-. In this chapter we give a brief overview of computational learning techniques and machine reasoning.

3.1 Machine Reasoning

Machine reasoning is the process of "algebraically manipulating previously acquired knowledge in order to answer a new question" (Bottou, 2011). This definition covers both logical and probabilistic inference. But human reasoning doesn't have the limitations of neither probabilistic nor logical inference. Converting the raw data to the logical expressions known to be a hard problem and searching discrete spaces of symbolic formulas can lead combinatorial explosion. Probabilistic reasoning known to have problems with representation. Representing causality with probabilities are challenging (Bottou, 2011).

3.2 Computational Learning and Inference

Learning and planning are important features of intelligent agents. In the following sections, we discuss about popular learning techniques that are suitable for use in multi-agent simulations with language games.

3.2.1 Supervised vs Unsupervised Learning

Unsupervised and Supervised Learning techniques are two of the most popular learning techniques used in computational learning. The difference between the supervised and unsupervised learning algorithms is established with training factor. Crucially the supervised learning algorithm can be stated as learning with teacher and unsupervised learning is the learning without teacher.

3.2.2 Supervised Learning

In supervised setting the learning process is the task of inferring a function from the supervised training data.

Formalization Bousquet et al. (2004) did a detailed analysis and formalization of supervised learning algorithms and my formalization will be based on Bousquet et al.'s work.

Given the features $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ and example d_i will be a set of values of features: $d_i = \{f_0^i, f_1^i, ..., f_n^i\}$. A training data set $\mathcal{D}^t$ consists of examples d_i and their labels t_k where $t_k \in \mathcal{T}$. Thus $\mathcal{D}^t$ will included pairs of examples and their labels $\mathcal{D}^t = \{(d_0, t_j), ..., (d_m, t_j)\}$. A typical supervised learning algorithm seeks a function $g : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space.

$\mathcal{G}$ is the space of all possible functions, g which is also called **"Hypothesis Space"**. Hence $g \in \mathcal{G}$. g can be scoring function such as $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$ where g returns the y that gives the highest score:

$g(x) = argmax\ \mathrm{f}(x, y)$

Most of the probabilistic learning algorithms takes the form of a conditional probability model:

$g(x) = P(x|y)$ or f can take the form of joint probability:

$f(x, y) = P(x, y)$

There are two approaches to be able to choose g and f:

Empirical Risk Minimization: ERM is used to the agreement between a candidate function and the data:

$R_n(g) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{g(\mathcal{X}_i) \neq \mathcal{Y}_i}$ where

- $\mathcal{L}_{g(\mathcal{X}_i) \neq \mathcal{Y}_i}$ is the loss function
- $\mathcal{R}_n$ is the risk function.

ERM tries to minimize the Empirical Risk function:

$\mathcal{R}^{erm} = argmin_{g \in \mathcal{G}} \mathcal{R}_n(g)$

Structural Risk Minimization: SRM is used for preventing overfitting a regularization penalty into optimization. SRM by Occam's Razor prefers less complex models, and it is basically ERM with a regularization penalty function:

$\mathcal{R}^{srm} = argmin_{g \in \mathcal{G}_d, d \in \mathcal{N}} \mathcal{R}_n(g) + pen(d, n)$

Concept Learning The idea of supervised learning is heavily influenced by the cognitive psychologist Jerome Bruner's works of concept learning (Quinlan, 1993).

Concepts have been an important topic for psychology, particularly concepts which identify kinds of things. Such concepts are mental representations which enable one to discriminate between objects that satisfy the concept and those which do not. Given their discriminative use, a natural hypothesis is that concepts are simply rules for classifying objects based on features. Indeed, the "classical" theory of concepts takes this viewpoint, suggesting that a

concept can be expressed as a simple feature-based rule: a conjunction of features that are necessary and jointly sufficient for membership (Goodman et al., 2008).

Concept learning also refers to a learning task in which a human or machine learner is trained to classify objects by being shown a set of example objects along with their class labels. The learner will simplify what has been observed in an example. This simplified version of what has been learned will then be applied to future examples. Concept learning ranges in simplicity and complexity because learning takes place over many areas. When a concept is more difficult, it will be less likely that the learner will be able to simplify, and therefore they will be less likely to learn. Colloquially, task is known as learning from examples. Most theories of concept learning are based on the storage of exemplars and avoid summarization or overt abstraction of any kind (Wikipedia, 2010b).

3.2.2.1 Paradox of Concept Learning

The learning paradox (from the view of Socrates) states that a learner cannot search either for what she knows or for what she does not know. Because if she already knows she does not need to relearn and search for this concept, while if she does not know she will not recognize it even when she encounters it. Fodor stated this paradox with the Fregean sense of concepts (Borensztajn, 2006):

- Concept learning have to do hypothesis testing and confirmation. Hypothesis should be formulated in terms of concepts in the conceptual system/mind.

- We can not formulate hypothesis for primitive concepts without using other concepts. But this would be a circular definition.

If primitive concepts are integrated into a conceptual schema, they must originate from both bottom-up (learning that goes from implicit to explicit knowledge) and top-down (learning that goes from implicit to explicit knowledge) learning processes. Briefly Fodor states that you can not learn new concepts with only bottom-up sensory information and hence primitive concepts should be innate. But there are several problems with this account. By using selectionist principles some of those problems can be fixed (Borensztajn, 2006).

3.2.2.2 Problem Of Induction

Induction is, by dictionary definition, is the process of inferring a general law or principle from the small and simple observations per se. Most of the computational learning algorithms assume that humans learn inductively and based on the induction, but there are problems with logical induction. Problem of induction is an epistemological problem that questions whether the inductive inferences lead to the knowledge. Hence philosophers seek answer for the two question (Wikipedia, 2010g):

- Generalizing about the properties of class of objects from some number of observations. Ex: The cows, I've seen are black, hence all the cows are black.
- Presupposing that a sequence of events in the future will occur as it always has in the past (For ex: Dawn of the sun.). David Hume called this as the "Principle of Uniformity of Nature".

3.2.3 Computational Learning Theory

Computational learning theory is the field of machine learning that analyzes the mathematical characteristics of learning algorithms.

3.2.3.1 PAC Learning

PAC Learning refers to the "Probably Approximately Correct" Learning which is first discussed by Leslie Valiant in (Valiant, 1984) and it is a framework for mathematical analysis of machine learning algorithms.

A major problem in statistical learning theory is the learning of functions. The concept class and the hypothesis class are the class of functions such as:

$\mathcal{F} : X \to Y$ where X and Y are distinct sets.

According to *Probably Approximately Correct* learning, given a class C and an unknown but fixed probability distribution in which examples are drawn from, $p(x)$, we want to find the number of examples, N with the probability at least $1 - \delta$, the hypothesis function H has at

most ϵ amount of error, for any ϵ and δ which satisfies, $\frac{1}{2} < \delta$ and $\epsilon > 0$ (Alpaydin, 2010).

The learner of a function is required to converge to the target function in the limit. But this convergence is probabilistic.

The class $\mathcal{F}$ of possible target functions (in PAC literature usually referred as concept, where concept is adopted from the concept learning) and the hypothesis class $\mathcal{H}$ are the classes of functions of Niyogi (2006),

$\mathcal{H} \Longrightarrow \mathcal{F} : X \rightarrow Y$

In the case of language we can evaluate X to be the set Σ^*, the possible set of strings and Y to be the set $\{0, 1\}$. We can write the $1_L(x) : \Sigma^* \rightarrow \{0, 1\}$

where $1_L(x) = 1$ iff $x \in L$ and L is the target language. Therefore learning indicator function is equivalent to the learning the language itself.

Learnability of Languages with PAC Partha Niyogi did an extensive study of language learning and computational learning theory in Niyogi (2006). The following depiction is taken from that book:

The learner receives the pairs of examples (x, y) where $x \in \Sigma^*$ and $y = 1_L(x)$.

The learner hypothesizes functions in $\mathcal{H}$ which is $\mathcal{H} : \Sigma^* \rightarrow \{0, 1\}$ and the learner maps the data sets to the hypothesis classes. Assume that the learner receives the positive and negative examples in a stream which has k number of elements in D_k, the set of all data streams. Hence:

$D_k = \{(z_1, \ldots, z_k) | z_i = (x_i, y_i); x_i \in \Sigma^*, y^i \in \{0, 1\}\}$

and according to the empirical risk minimization an appropriate $\hat{h}_l$ is chosen:

$\hat{h}_l = argmin_{h \in \mathcal{H}} \frac{1}{l} \sum_{i=1}^{l} (y_i - h(x_i))$

The hat on $\hat{h}_l$ represents that it is a random function.

A learning algorithm $\mathcal{A}$ is an effective procedure mapping data sets to hypothesis, i.e.,

$\mathcal{A} : \cup_{i=1}^{\infty} D_k \rightarrow \mathcal{H}$

$\hat{h}_l$ is $\mathcal{A}(d_l)$ where d_l is a random element of D_l. In a successful learning setting, learner's

hypothesis will converge to the target as the number of data points goes to the infinity.

Because $\hat{h}_l$ is a random function, one may consider the convergence as a probability:

$\hat{h}_l$ converges to 1_L [1], iff for every $\epsilon > 0$:

$\lim_{l \to 0} P[d(\hat{h}_l, 1_L) > \epsilon]$

P allows us to define the distance between the language's corresponding distance functions and it provides the distribution which the data is drawn from, then presented to the learner. Therefore it provides a characterization of the probabilistic behavior of random function of $\hat{h}_l$.

The notion of convergence here is the notion of weak convergence of a random variable and $d(\hat{h}_l, 1_L)$ random variable, because $\hat{h}_l = \mathcal{A}(t_k)$ where t_k is a random text. This notion of weak convergence is usually stated as (ϵ, δ) in PAC formulations. If $\hat{h}_l$ weakly converges to the target 1_l, it follows that for every $\epsilon > 0$ and $\delta > 0$, there exists an $m(\epsilon, \delta)$ such that for all $l > m(\epsilon, \delta)$.

$P[d(\hat{h}_l, 1_L) > \epsilon] < \delta$

This implies that, with high probability of $(> 1 - \delta)$, the learner's hypothesis is approximately close to the target language, and $m(\epsilon, \delta)$ refers to the sample complexity of learning. A set of languages $\mathcal{L}$ are said to be learnable if there is an algorithm $\mathcal{A}$ which can learn every language in the set uniformly.

There are two important concepts that arose as a result of PAC Learning Theory:

Strong learners A PAC learning algorithm, also called as strong learner, with probability at least $1 - \delta$ the error rate of hypothesis is at most ϵ. Moreover, the training time and number of training required must be polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and the size of the training sample.

Weak learners Weak learners are type of classifiers that are only slightly better then the random guess. Assume that there are N possible class labels, then classification done by a weak learner will be only slightly better than $\frac{1}{N}$ (Kearns and Valiant, 1994).

[1] 1_L is the indicator function for language L.

3.2.4 Meta-Learning

Meta-learning or learning to learn is a technique in Machine learning that automatically improves the learning method or the algorithm by using the experiences such that the new algorithm is better than the original algorithm (Schaul and Schmidhuber, 2010). Most popular examples of meta-learning are ensemble learning algorithms like boosting and bagging.

Definition: Definition of meta-learning from (Schaul and Schmidhuber, 2010):

"Consider a domain D of possible experiences $s \in D$, each having a probability $p(s)$ associated with it. Let T be the available training experience at any given moment. Training experience is a subset of D, i.e. $T \in D_T \subset \mathcal{P}(D)$, where $\mathcal{P}(D)$ is the power set of D. An agent π_θ is parametrized by $\theta \in \Theta$. A task associates a performance measure $\phi : (\Theta, D) \mapsto \mathfrak{R}$ with the agent's behavior for each experience. We denote by Φ the expected performance of an agent on D:

$$\Phi(\theta) = \mathbb{E}_{s \in D}[\phi(\theta, s)]$$

Now we define a learning algorithm $\mathbf{L}_\mu : (\Theta, D_T) \mapsto \Theta$, parametrized by $\mu \in M$, as a function that changes the agent's parameters θ based on training experience, so that its expected performance Φ increases. (Here it is assumed that the learning algorithm may be rather complex algorithm in general and may incorporate more than one learning method.) More formally, we define the learning algorithm's expected performance gain δ to be:

$$\delta(\mathbf{L}_\mu) = \mathbb{E}_{\theta \in \Theta, T \in D_T}\left[\Phi(\mathbf{L}_\mu(\theta, T)) - \Phi(\theta)\right]$$

Any learning algorithm must satisfy $\delta > 0$ in its domain, that is it must improve expected performance. A learning algorithm's modifiable components μ are called its meta-parameters. We define a meta-learning algorithm $\mathbf{ML} : (M, D_T) \mapsto M$ to be a function that changes the meta-parameters of a learning algorithm, based on training experience, so that its expected performance gain δ increases:

$$\mathbb{E}_{\mu \in M, T \in D_T}\left[\delta(\mathbf{L}_{\mathbf{ML}(\mu,T)}) - \delta(\mathbf{L}_\mu)\right] > 0$$

In other words, using $\mathbf{L}_{\mu'}$ tends to lead to bigger performance increases than using $\mathbf{L}_\mu$, where $\mu' = \mathbf{ML}(\mu, T)$ are the updated meta-parameters. Note the symmetry of the two definitions,

due to meta-learning being a form of learning itself."

3.2.5 Ensemble Learning

Ensemble learning is a machine-learning paradigm in which multiple learners are trained and predictions are integrated in order to get better predictive performance than the base algorithms. *Base learner* is a single learner from the ensemble. The main idea behind the ensemble methods is to weigh several individuals and combine them to obtain a classifier accuracy than any individual in the ensemble (Rokach, 2010).

Ensemble learning has 3 phases:

- Sampling Phase: In the sampling phase the data is split into partitions or weighted.
- Training Phase: Each classifiers will be trained with the data that is split in the sampling phase.
- Classification Phase: Each classifier will try to classify examples.
- Model Selection/Model Aggregation: The decisions will be aggregated or a specific classifier will be chosen as the decision-maker.

To be able to get a wise decision from a crowd following conditions are needed as stated in (Rokach, 2010):

- Diversity of opinion - Each member should have private information even if it is just an unusual interpretation of the known facts.
- Independence - Members' opinions are not determined by the opinions of those around them.
- Decentralization - Members are able to specialize and draw conclusions based on local knowledge.
- Aggregation - Some kind of mechanism exists for turning the private judgments into a collective decision.

Most of the ensemble learning algorithms are also the meta-learning algorithms, such as boosting and bagging described below. They do not modify the base-learner. They just modify the inputs and outputs.

According to the base-learners that constitute an ensemble it can be grouped in two categories:

- **Homogeneous Ensemble**: If the base-learners that constitute an ensemble are of the same kind, then this ensemble is called to be a homogeneous ensemble.

- **Heterogeneous Ensemble**: If the base-learners that constitute an ensemble are different kinds of learners, then this ensemble is called to be a heterogeneous ensemble.

Why and How do Ensemble Techniques work? Learning algorithms that use a single hypothesis suffer from the following problems that ensemble techniques overcome (Dietterich, 2002):

- **The statistical problem:** The amount of data available for training will not be enough to be able to model the whole space with a single classifier. Therefore voting several equally accurate classifiers might work. A classifier that suffers from this problem is said to have a high ”variance”.

- **The computational problem:** Searching the whole hypothesis space to find the best classifier can be computationally intractable, in these cases heuristic techniques such as stochastic gradient is used. But stochastic gradient can stuck into the local minimum. In that case weighted combinations of several local minimum can overcome the problem.

- **The representation problem:** The representational problem arises when the hypothesis space does not contain any useful hypotheses that is a good approximation to the correct function f. Voting several classifiers can expand the hypothesis space. Hence, by doing weighted voting we can establish a better approximation to f.

Since there is no point in combining learners that takes similar decisions, we try to create diverse classifiers in the ensemble methods (Alpaydin, 2010).

Model Combination Schemes There are several ways to aggregates decisions of base-learners (Alpaydin, 2010):

- Multi-expert Methods: These methods can be grouped in two subcategories:
  - Global approach/learner fusion, given an input, all base-learners generate an output and all these outputs are used. Examples are voting and stacking.
  - In the local approach/learner selection, for example, in mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output.
- Multistage combination methods. An example is cascading.

Bagging and *AdaBoost* algorithms that are discussed in Appendix A are very popular ensemble learning techniques and both of them are a multi-expert method whereas bagging is using Majority Voting and *AdaBoost* is using weighted majority voting.

3.3 Deep Learning

Theoretical results in statistical machine learning suggest that in order to learn the complicated functions that can represent high-level abstractions, deep architectures are required. Deep architectures are composed of multiple layers of non-linear operations, such as neural networks with multiple hidden layers. Several studies in cognitive science have shown that cognitive processes are deep and the brain has a deep architecture (Bengio, 2009). For example when people try to solve a problem they organize their ideas and concepts hierarchically. Humans first learn simpler concepts and then compose them to represent more abstract ones. The first successful attempt to use deep architectures with supervised learning for deep learning is deep belief networks (DBN) (Hinton et al., 2006).

CHAPTER 4

Methodology and Empirical Work

And AC said, "LET THERE BE LIGHT!"

And there was light...

–Isaac Asimov[1]

4.1 Introduction -The Wisdom of Crowds

In language game models that we have discussed in Chapter 2, agents interact with each other to agree on a particular name for an object. The purpose of model fusion in "Ensemble Learning" is to aggregate the decision of several learners or models. Particularly objective of both domains is same, and we suggest to use a specific type of language game for model fusion in ML which is an important problem for ensemble learning.

Ensemble and collaborative learning algorithms are gaining popularity in the literature. Additionally performance benchmarks against single learner models such as Ruta and Gabrys (2005)'s study showed that, ensemble techniques perform significantly better than the single classifier models in terms of accuracy. They can deal with concept drifts[2] more successfully (Wang et al., 2003) than the single classifier learners. Currently in machine learning, the most popular ensemble learning algorithms are using voting and its variants (e.g.: dynamic voting, weighted majority voting etc) to aggregate several models. In real world, agents do not agree on a topic by averaging, social systems are known with their extreme nonlinearity. Therefore averaging classifiers' decisions will not be a good approximation model for the real

[1] Taken from the Asimov's renowned short story, "The Last Question".

[2] Concept Drift, refers to the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways.

world. Most of the important data related problems are caused by complex social systems. These problems are known to be challenging. Data generated by real-world phenomena usually lack obvious patterns (such as stock market, political conflict resolution data etc). Hence they are not very suitable for general prediction and classification algorithms. Deterministic algorithms do not perform very well with these kind of problems and sometimes adding a bit of entropy may yield better performance.[3] That's why sometimes randomized algorithms perform better on some data sets.

Categories and Categorization

There has been a vast amount of interest on categories in philosophy since Aristotle who, in his tractate Categories, attempts to list the most general kinds into which entities would divide in the world (Thomasson, 2010). Important names of philosophy such as Aristotle, Kant and Husserl studied on the notion of categories but each one of them adopted a different view on categories. Aristotle used language as a clue to ontological categories, and Kant treated concepts as the route to categories of objects of possible cognition, Husserl explicitly distinguished categories of meanings from categories of objects, and attempted to draw out the law-like correlations between categories of each sort. Also Kant and Aristotle lay out a single system of categories whereas Husserl distinguishes two ways of arriving at top-level ontological classifications. According to Husserl categories are entirely a priori matter (Thomasson, 2010). Besides philosophy, there has been great deal of interest on categories in cognitive and computer science as well. In cognitive science debates are more focused on how humans in fact come to group things into categories–whether this involves lists of descriptive (observable or hidden) features, resemblance to prototypes, prominent features weighted probabilistically, etc. But the current literature misses the point that, categorization is in fact a social process. The cultural differences in categories of certain things such as color categories is one of the example of that situation (Roberson et al., 2000; Belpaeme, 2001).

In this chapter we propose two new types of language game (in a late Wittgensteinian sense) for the model combination and simulation of the categorization process in a group of agents:

- Categorization Game (CG)
- Categorization Game with Confidence Rated Belief Updates (CGCRBU)

[3] For example, Littlestone and Warmuth (2002) showed that "randomized weighted majority voting" algorithm outperforms the "typical weighted majority voting".

4.1.1 Majority Voting

Majority voting is one of the most popular and easiest technique for model aggregation. Each member of the ensemble casts its vote for the selected class and the class with the highest number of votes is selected as the candidate class[4].

4.2 List of Proposed Approaches

In this section, we give detailed explanations on the language games we propose for combination of several weak learners to obtain a strong learner as in an ensemble setting.

4.2.1 Categorization Game for Model Aggregation

Categorization game is inspired from the naming game of Steels (1996) but it has several additional functionalities and simplifications in order to adapt to the changes in the domain of the problem.

Broadly speaking, CG can function like a search algorithm that is trying to find an item with specified properties among a set of items. In this thesis several flavors are added to the naming game in order to ensure that it performs like a search algorithm for classifier fusion. In the second game, we have added a fitness function (we choose the speaker according to a belief score.) to make sure of that the communication evolve towards the goal that we wish to reach. In a nutshell, our goal is to combine the models in a reasonable time and find the correct decision. Therefore choosing the fitter agents as speaker will increase the population's bias to a certain target.

4.2.1.1 Categorization Game (CG)

Categorization Game solely performs a basic version of the naming game as introduced by Steels (1995). After classification is performed by each agent, agents start to play the categorization game in order to agree on a specific category. Unlike the naming game, in CG there

[4] Majority voting works like a democratic regime. But the weighted majority voting algorithm, unlike in a democratic regime, each voter's vote has a weight associated with their own decision.

is no invention of new games. We assumed that the categories in the CG is either innate or pre-adopted from the training data set.

The formal definition of CG is shown below:

$\mathcal{A}$ with size $\mathcal{N}_{\mathcal{A}}$ where each agent $a_i \in \mathcal{A}$ is in contact with $O_a = \{o_0, \ldots, o_n\}$ with meanings $\mathcal{M}_a = \{m_0, \ldots, m_n\}$ associated with them $\mathcal{N}_O$ of objects, agents have their own set of categories $C_{a,t}$ and their lexicon will be $L_{a,t} \subset M_a \times C_{a,t} \times \mathcal{N}_O$. Therefore an agent a at time t can be defined as $a_t =< C_{a,t}, L_{a,t}, H_a >$ where H_a is the learning algorithm of agent a at time t.

The algorithm of categorization games is as shown in Algorithm 4.2.1. Categorization game works in a completely stochastic fashion. There is no central control and no objective function and there is no guarantee that the game will always converge to the optimal category as well. But the probability of convergence of majority category is higher than the minority ones:

Let's assume that, c_i are categories and $c_i \in C$ and

$\sharp(c_m) \geq \sharp(c_k) \geq \cdots \geq \sharp(c_j)$

Hence:

$p(c_m) \geq p(c_k) \geq \cdots \geq p(c_j)$

If the class distribution is not uniform, then the probability of c_m -the majority category- will be more likely to be chosen as speaker. Thus it is more likely that the population of agents will agree on c_m for a selected object.

4.2.1.2 Categorization Game with Confidence Rated Belief Updates (CGCRBU)

CGCRBU is based on naming game just like CG. But unlike CG, in the CGCRBU, after randomly two agents are chosen, their roles such as teacher and learner,[5] are not chosen randomly. The agent with the higher belief score is chosen as teacher and the lower one as the learner. The information flow in CG is from teacher to learner and the decisions of teachers determine the outcome of the game. Hence the process of choosing the roles of agents is very important. These belief scores function like the fitness score in genetic algorithms and we can assume that it is a type of objective function that determines the optimality of the agents'

[5] Teacher is equivalent to speaker and the learner is equivalent to the hearer in the naming game.

Algorithm 4.2.1 BasicCategorizationGame

1. **Select** a classification task t among the set of tasks $\mathcal{T}$
2. **Create** a population P of N number of agents $\mathcal{A}$
3. **Sample** training data set $\mathcal{D}^{Tr}$ to obtain a partition D_i^{Tr} of the training data set
4. **Train** each agent $A_i \in \mathcal{A}$ with training data set D_i^{Tr}
5. **for all** $x_i \in \mathcal{D}^{Te}$ **do**
6. **for all** $A_i \in \mathcal{A}$ **do**
7. A_i classifies x_i and obtains label y_i where $y_i \in \mathcal{Y}$
8. **end for**
9. **repeat**
10. Randomly select an agent A_i^S as a teacher and an agent A_i^H as a learner.
11. A_i^S utters the category C_i^S and A_i^H utters the category C_i^H.
12. **if** C_i^S equals C_i^H **then**
13. The game is success
14. **else**
15. The game is failure
16. A_i^H updates its lexicon and replaces C_i^H with C_i^S.
17. **end if**
18. **until** *isgameconverged*
19. **end for**

decision, in which we rank the agents' fitness according to this belief score.

Conceptualization of CG Assume that we have a group of agents A and those agents are trying to categorize an object o. Agent a_i (teacher) initiates a dialogue with agent a_j (learner) -an agent with a lower belief score than the a_i- and it establishes joint attention by pointing the object to be classified. Then a_i asks what the pointed object is. a_j will utter a category (answer) that it believes that the correct category is. If they can't agree, learner will replace the category that she uttered with the category that the teacher uttered and in this case the result will be failure otherwise the result will be success. If the result is success both agents' belief scores will be decreased, otherwise the belief scores will be increased according to Algorithm 4.2.3.

The formal definition of CGCRBU is as below:

- The Teacher and Learner are chosen according to their belief scores β_L and β_T.
- Belief Score of classifier H_a is computed as: $\beta_a = P(x_i|C_k) \times \alpha_a$ where α_a is the confi-

dence score.

- Structure of agent a at time t is: $a_t =< C_{a,t}, L_{a,t}, H_a, \beta_m >$

With the help of belief updates each agent can align their repertoire according to the status of the game. This provides us dynamic and effective communication as a result. Confidence scores are computed by obtaining accuracy on the validation data set, similar to the 1-fold cross validation. Using k-fold cross validation could be very expensive in our simulations. Because each agent would have to be trained k times. Confidence score α_m is multiplied with the $P(x_i|C_k)$ which is the probability of example x_i given that it belongs to the category C_k that the agent A_m utters. So computation of belief score β_i can be formalized as:

$\beta_m = P(x_i|C_k) \times \alpha_m$

Belief scores are updated according to Algorithm 4.2.3 and this algorithm is also illustrated in Figure 4.1. The algorithm of categorization game with confidence rated belief updates is shown in Algorithm 4.2.2.

The algorithm for belief updates are shown in Algorithm 4.2.3. The goal of belief updates is to promote the successful communication and to reduce the number of failed games to speed up the games and increase the accuracy of decisions. Language games are "shared cooperative activities"(SCA) (Bratman, 1992) or "joint action" and in SCA each agent acts according to their expectations and expectations are based on the expectations on the other agents and these expectations are based on the beliefs about the other agents about her. As discussed by Tomasello and Carpenter (2007), social norms, can only be created by the individuals who engage in shared intentionality and collective beliefs, and they play an enormously important role in maintaining the shared values of human cultural groups. Moreover learning is a social and interactive process in which individuals change their beliefs when they receive new information or feedback (Elio and Pelletier, 1997, 1994). These studies show that our belief update algorithm is cognitively plausible as well. The successful belief updates rule is very similar to the perceptron weight update rule.

The algorithm of belief updates are shown on Algorithm 4.2.3, if not stated otherwise, we have chosen η's (update rates) as equal $\eta_s = \eta_h$.

Algorithm 4.2.2 BeliefCategorizationGame($\mathcal{T}$, N, $\mathcal{A}$, $\mathcal{D}^{Tr}$, $\mathcal{D}^{Te}$)

Select a classification task t among the set of tasks $\mathcal{T}$
Create a population P of N number of agents $\mathcal{A}$
Sample training data set $\mathcal{D}^{Tr}$ to obtain a sampled training data set D_i^{Tr}
Train each agent $A_i \in \mathcal{A}$ with sampled training data set D_i^{Tr}
Assign confidence sore T_i for each agent A_i using cross-validation
for all $x_i \in \mathcal{D}^{Te}$ **do**
    for all $A_i \in \mathcal{A}$ **do**
        A_i classifies x_i and obtains label y_i where $y_i \in \mathcal{Y}$
        $\beta_i = F_i * T_i$
    end for
    repeat
        Randomly select an agent A_i as a teacher and an agent A_j as a learner.
        if $\beta_i > \beta_j$ **then**
            A_i utters the category C_i and A_j utters the category C_j.
            if C_i equals C_j **then**
                $S_{success} \leftarrow true$ and $UpdateBeliefs(\beta_i, \beta_j, S_{success})$
            else
                $S_{success} \leftarrow false$ and $UpdateBeliefs(\beta_i, \beta_j, S_{success})$
                A_j updates its lexicon and replaces C_j with C_i.
            end if
        else if $\beta_i < \beta_j$ **then**
            A_i utters the category C_i and A_j utters the category C_j.
            if C_i equals C_j **then**
                $S_{success} \leftarrow true$ and $UpdateBeliefs(\beta_j, \beta_i, S_{success})$
            else
                $S_{success} \leftarrow false$ and $UpdateBeliefs(\beta_j, \beta_i, S_{success})$
                A_j updates its lexicon and replaces C_j with C_i.
            end if
        else
            Randomly select an agent A_t as teacher and A_l as learner.
            A_t utters the category C_t and A_l utters the category C_l.
            if C_i equals C_j **then**
                $S_{success} \leftarrow true$ and $UpdateBeliefs(\beta_t, \beta_l, S_{success})$
            else
                $S_{success} \leftarrow false$ and $UpdateBeliefs(\beta_t, \beta_l, S_{success})$
                A_l updates its lexicon and replaces C_l with C_t.
            end if
        end if
    until *isgameconverged*
end for

Algorithm 4.2.3 UpdateBeliefs($\beta_H, \beta_S, S_{success}$)

1. **if** $S_{success} = True$ **then**
2. $\beta_H \leftarrow \beta_H + \eta_s \times \beta_S$
3. $\beta_S \leftarrow \beta_S + \eta_s \times \beta_H$
4. **else**
5. $\beta_H \leftarrow (\beta_S - \beta_H) \times (1 - \eta_f) + \eta_f \times \beta_S$
6. $\beta_S \leftarrow \beta_S - \eta_f \times \beta_H$
7. **end if**

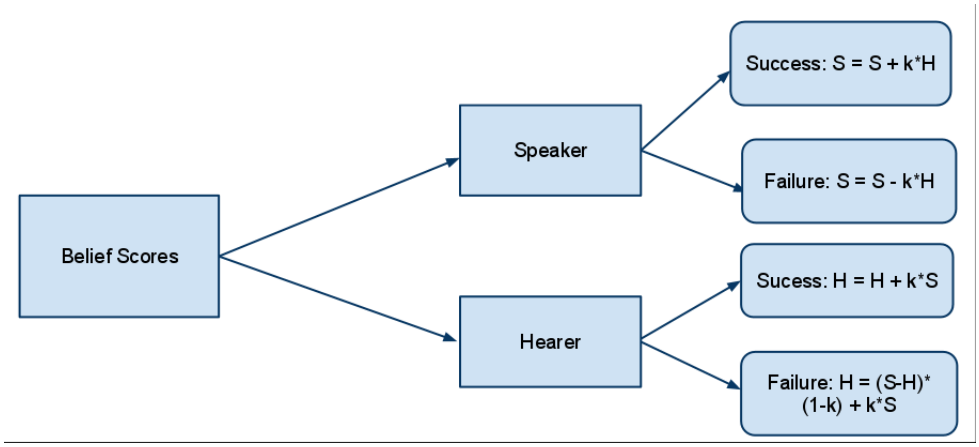

Figure 4.1: **Belief Updates:** This figure basically shows how the belief scores are determined according to the conditions. S is the confidence score of speaker and h is the confidence score of hearer. k is the rate of belief update.

4.2.2 Sampling Techniques for Categorization Games

4.2.2.1 Importance Sampling (IS)

Importance sampling is sometimes referred as biased sampling as well, and it is a variance reducing sampling algorithm. A key issue to achieve small errors (for a given number of samplings) is a suitable strategy of sampling the available multidimensional space. If the volume of data to be sampled is very large, but is characterized by small probabilities over most parts, one can achieve importance sampling by approximating the probability distribution by some function $P(x)$, and generating randomly x according to P, and weighting each result at the same time by $[\frac{dP(x)}{dx}]^{-1}$ (Bock et al., 1998). Importance sampling is usually used for estimating variables.

Weighted Random Sampling (WRS) WRS is essentially a variant of an importance sampling. The weights are assigned according to the distribution of classes among the samples in the data set.

Assume that, $\mathcal{X} = x_0, \ldots, x_n$ and $C = c_0, \ldots, c_m$, where x_i's are our classes and c_k's are our classes that each x_i belongs to. The distributions of classes determined by $D(c_i)$. The core intuition of WRS we have used in these tests is to generate a subsample S_k from $\mathcal{S}$ where $D(c_i^{S_k})$ and $D(c_i^{\mathcal{S}})$ are very close to each other. The reason of using the WRS in our case, is to be able to increase the bias. The *WRS* conducted in this thesis does not use replacements.

4.3 Implementation

We have used java for building the simulation. We used WEKA library[6] (Hall et al., 2009) for using the data-mining learning algorithms like C4.5.[7] The results of simulation can be stored in a NoSQL database MongoDb or a file-based RDBMS, sqlite. The relevant statistics have been drawn with python's *matplotlib* library and *gnuplot* tool. For efficient pseudo random number generation Mersenne Twister algorithm (Matsumoto and Nishimura, 1998) is used. The test data sets should have sufficient number of categories in order to better observe the emergence of categories. I've used libsvm (Chang and Lin, 2001) to test the categorization

[6] WEKA is an open source application developed by University of Waikato.
[7] The WEKA equivalent of C4.5 algorithm is J48

games with SVM algorithm. We have used NetworkX (Hagberg et al., 2008) and Graphviz (Ellson et al., 2003) libraries to visualize the networks of agents.

4.4 Empirical Results

4.4.1 Data Sets

The tests have been conducted on 3 different data sets, and their characteristics can be summarized as below:

MNIST Data Set The MNIST database of handwritten digits, has a training set of 60000 examples, and a test set of 10000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. This data set is specifically suitable for researchers who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting (LeCun et al., 2002). Due to the size of the data set we had to do sampling on the data set in order to do fit the data set into the memory. We have use random sampling on the data set with 40 percent as the sampling percentage. MNIST data set has 785 features (value of each pixel) which are numerical per picture. MNIST is one of the most popular data set in the machine learning community. It has been specifically used for benchmarking deep learning algorithms.

GTVS Data Set GTVS (Ground Truth Verification System) (Canini et al., 2009) is a network traffic classification data set. This data set is constituted with measurements from different places in different times. The data is multivariate and has 13 classes. The classes in the data set are not balanced. Each example in the data set has 12 features and all those features are numerical.

Segmentation Data Set We used image segmentation data set created by Brodley and Friedl (1999) from the outdoor images in University of Massachusetts. The examples of the data set were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel. Each instance is a 3×3 region and has 19 continuous

features. There are 6 classes in the data set, that are either, ”brickface, sky, foliage, cement, window, path or grass“.

Characteristics of Data Sets					
Data Set	# Nominal Attrs.	# Numerical Attrs.	# Classes	# Test Examples	# Training Examples
GTVS	-	12	13	Day1 (324276)	Day2(175662)
MNIST	-	785	10	10000	60000
Segment	-	19	7	1500 (test)	810 (challenge)

Table 4.1: Table showing the basic characteristics of data sets.

Construction of Validation Data Set We used validation data set to obtain a confidence score of a classifier. The training data set is split into N sub sets in which N different classifiers will be trained from N different partitions. Each classifier randomly chooses k (we have used $k = 1$ in our tests.) sub sets of training data set that doesn't include their own training data set and create a validation data set from by aggregating those sub sets.

4.4.2 Basic Phenomenology

In the classical naming game simulations, agents invent their own words in the beginning of the game. But in the categorization game agents have predefined lexicon of categories and they have already have a word for each object (but their belief and confidence might be weak about their knowledge) in their lexicon which originates from their training data set. Thus they do not need to invent new words for each object. In Figure 4.2 we show the changes of number of different words $N_d(t)$ in our game for each round. In contrast to naming game CG does not have fluctuations. The fluctuations in typical naming game is caused by invention of new words. In standard naming game, initially each agent's lexicon is empty and they create new words if they don't have any name.

Plot in Figure 4.2 is monotonically non-increasing. This test is done with 10 agents and a lexicon of 15 words. In the beginning each agent randomly chooses a lexical item from the lexicon and plays the CG.

Plot in Figure 4.2 is of a test that involve 10 agents where each agent select a word from the lexicon of 30 categories.

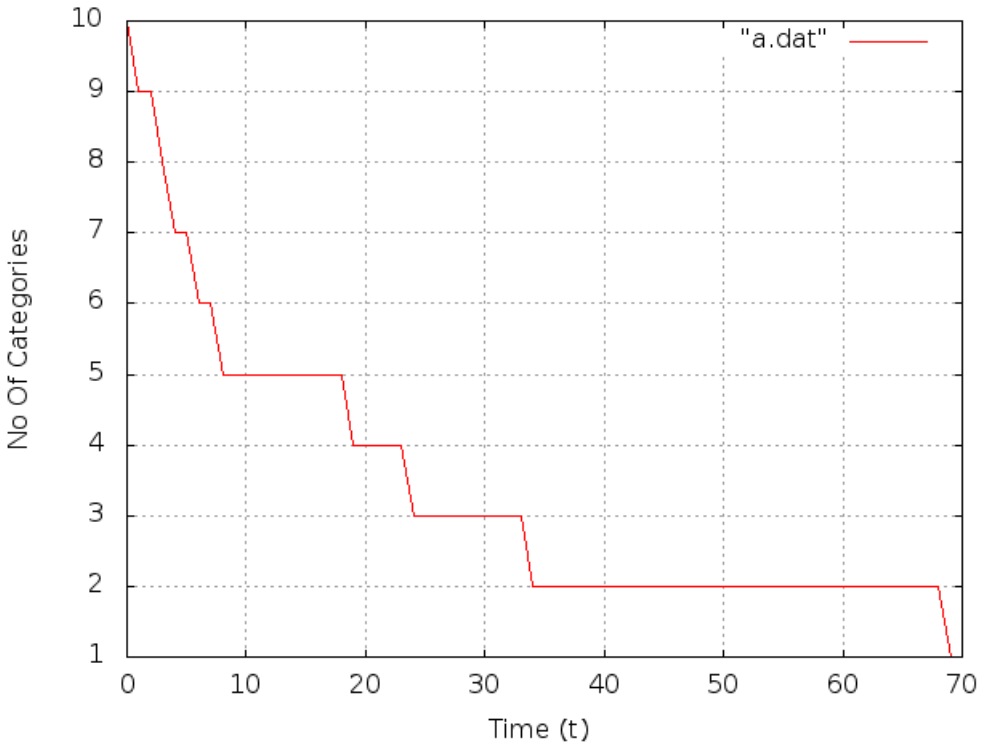

Figure 4.2: (CG) Evolution of lexicon and convergence of categories in categorization game. In the beginning of the game there are 10 different categories and as they agents communicate with each other, the number of different categories decreases. At the end of the game agents agree on a single category. At the time intervals that number of different categories stays same, agents have successful dialogues.

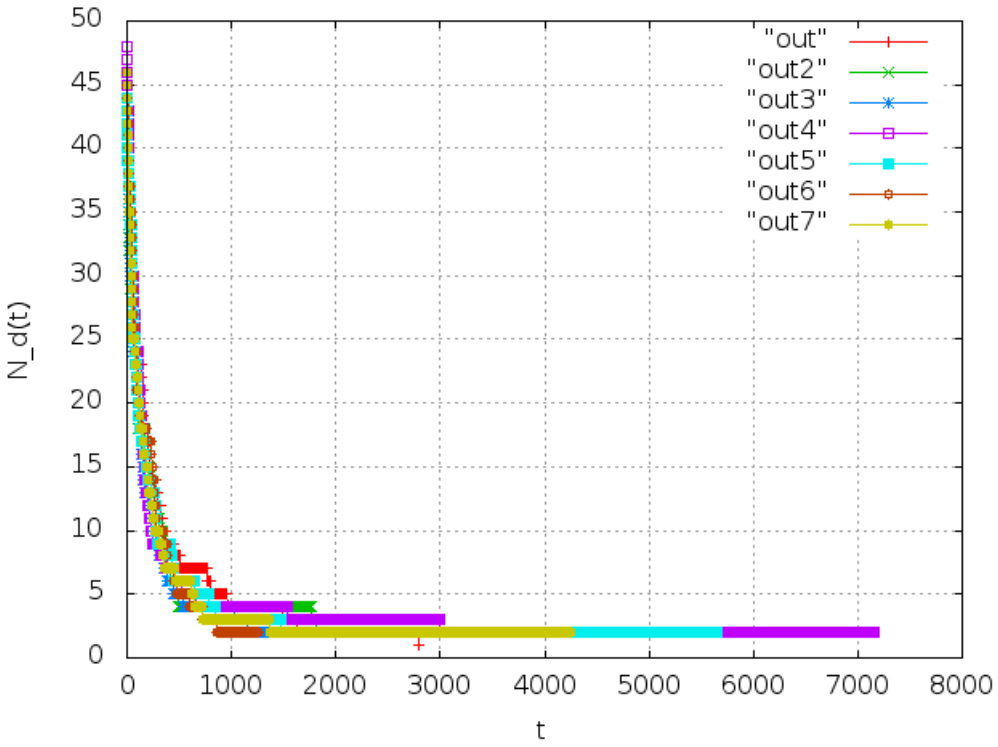

Figure 4.3: The evolution of lexicon and convergence of categories in CG after 7 runs. As seen from this plot $N_d(t)$ initially decreases rapidly and then the decline starts to slow down and in the end they reach linguistic coherence.

In the plot 4.3 60 agents involved in the test and they randomly select a word from the lexicon of 70 categories.

Plot in Figure 4.4: 50 agents involved in that test and each of them randomly selects a category from the lexicon of 60 categories. Success rate $S(t)$ is calculated by dividing the number of successes to the total number of iterations (or communications) in the game.

In the $3D$ plot 4.5 60 agents involved in the test and they randomly select a word from the lexicon of 70 categories.

As seen on 3D interaction networks at Figure 4.9, Figure 4.8 and Figure 4.7 as the number of words in the lexicon and agents increases the interaction network got more complicated and the communication becomes inefficient.

In Figure 4.10 we have shown the interaction network of 10 agents that chose their words from the lexicon of 10 words. The agents that have higher belief score is more likely to be

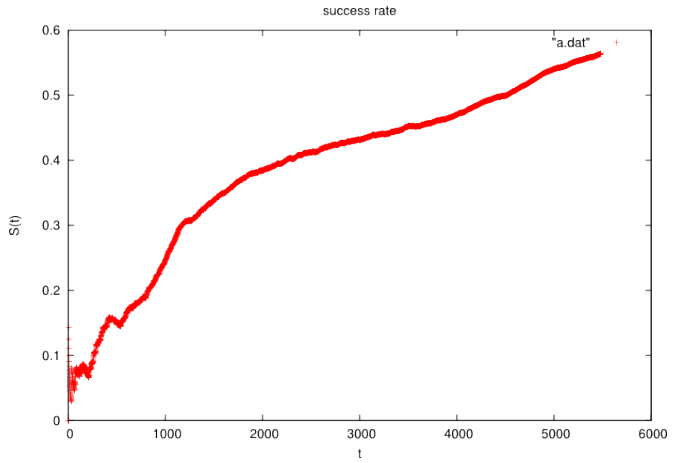

Figure 4.4: This graph is showing the evolution of success rates $S(t)$ with respect to time t.

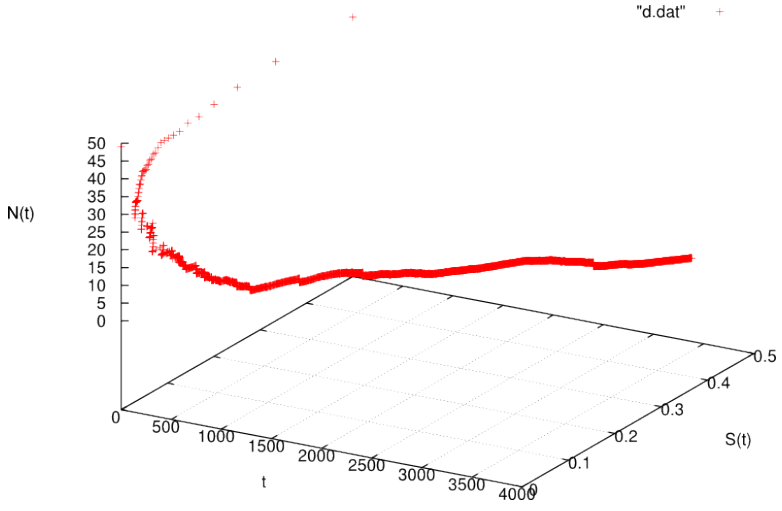

Figure 4.5: This graph is showing the evolution of lexicon and convergence of categories in categorization games. This plot illustrates the change in $S(t)$ and $N_d(t)$ with respect to time.

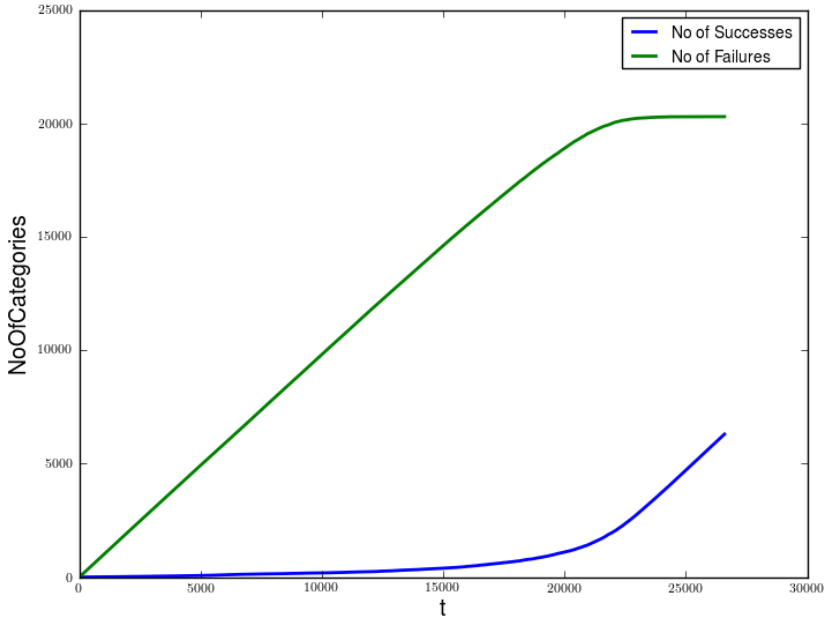

Figure 4.6: The evolution of categories showing number of successes and the number failed communications in CGCRBU with respect to time with 1000 agents and an initial lexicon containing 120 words in which agents chooses randomly in the beginning of the game.

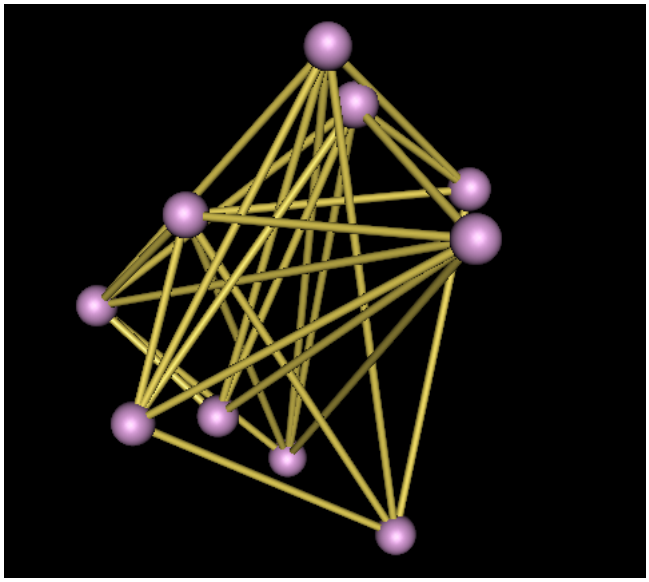

Figure 4.7: 3D interaction network of a CGCRBU with 10 agents and an initial dictionary of 6 words.

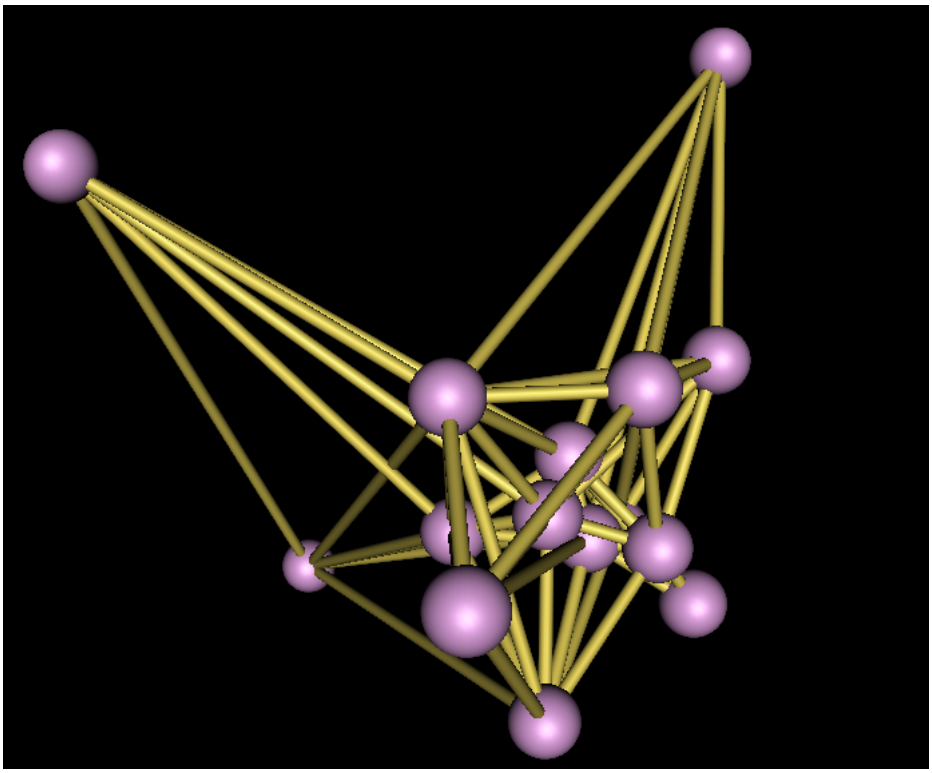

Figure 4.8: The 3D interaction network of CGCRBU with 15 agents and a lexicon of 10 words.

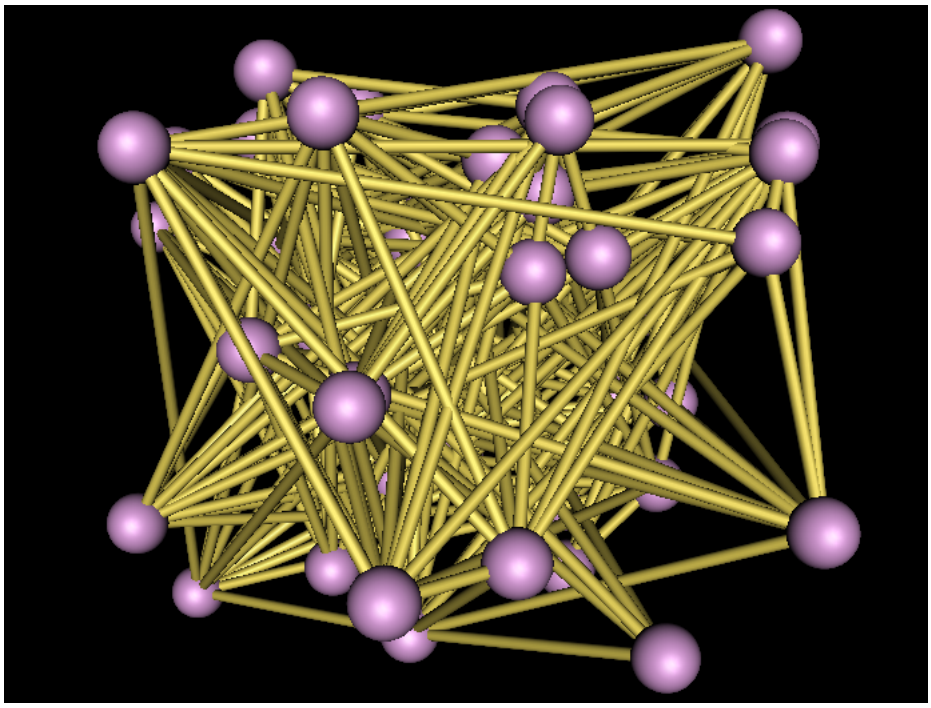

Figure 4.9: The 3D interaction network of CGCRBU with 50 agents and a lexicon of 8 words.

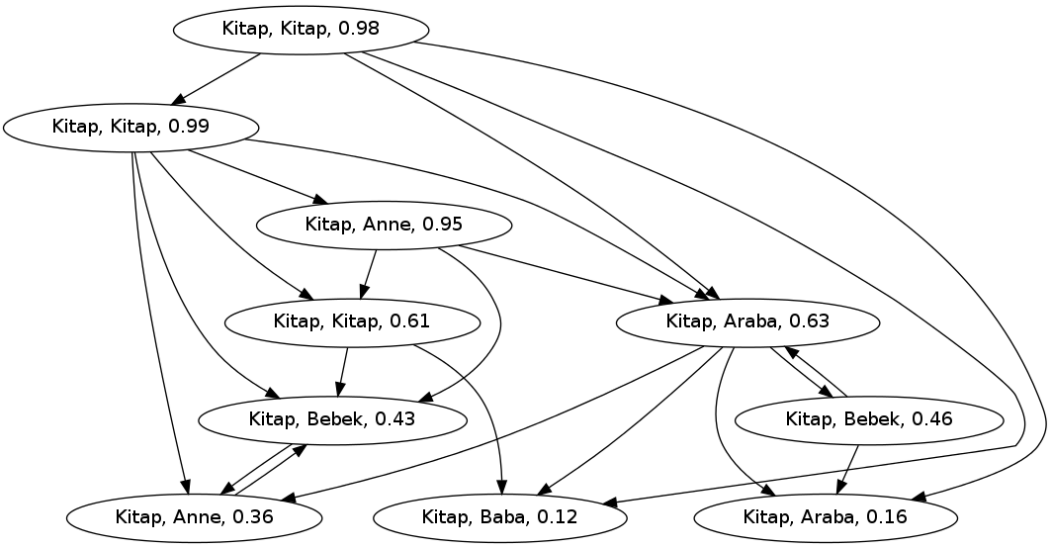

Figure 4.10: Interaction networks with 10 Agents and 10 words in the lexicon. Each eclipse on graph specifies an agent and the first word in each eclipse is the converged word and the second one is the initially selected word by the agent. Numerical value in the eclipses is the belief score of an agent at the end of the game. The direction of arrows specifies the communication flow from teacher to learner.

chosen as the learner.

As seen on Figure 4.12, the success and fail rates are symmetric.

In order to compare the categorization games along with majority voting and other methodologies, we have tested several learning algorithms on three different data sets. Initially we tested them with on MNIST, GTVS and Segmentation data sets and calculated their accuracy. Accuracy is calculated according to the ratio of number of correctly classified examples to the number of examples in the test data set. The size of MNIST and GTVS data sets were too large into fit to memory. Thus MNIST training and test data sets were randomly sampled without replacement with 40 percent and GTVS training data set is randomly sampled without replacement with the ratio 50 percent. In GTVS data set first day's collected data is chosen to be the training data set and the second day's data is chosen as test data set.

In Figure 4.14 and 4.15 we can see the change of accuracies with respect to increasing sampling percentage and number of agents. These two tests are done on segmentation data set with Multinomial Naive Bayes (NB) algorithm as the base learner. **CG** and **CGCRBU** are meta-learning algorithms. Hence they don't depend on type of the base learner. We have

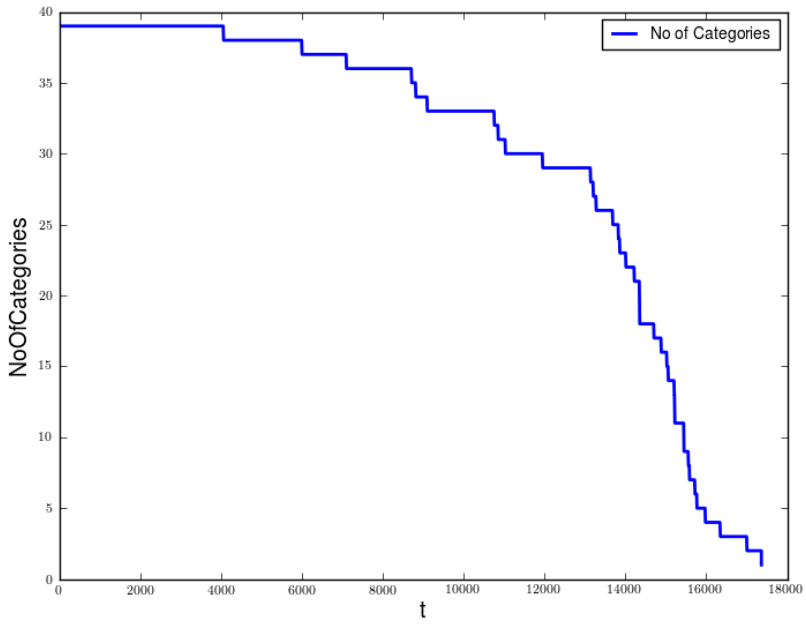

Figure 4.11: The evolution of lexicon and convergence of categories in CGCRBU with 1000 agents and 40 words.

done several independent tests on GTVS, MNIST and Segmentation with NB, C4.5, SVM and Kstar algorithms in order to observe the behavior of the algorithms on different data sets.

As seen from the performance of C4.5 and NB algorithms on MNIST, GTVS and Segmentation data sets. C4.5 performs significantly better than the NB. Because of the NB's conditionally independence assumption, it ignores the conditional dependency between the dimensions of data and does not perform well on our data sets.

The first test is conducted on segmentation data set, and the goal is to observe the change of accuracies with respect to the changes of sampling percentage on the training data set for base learners. The results are shown in Table 3.1.

The second test is conducted on segmentation data set and the goal is to observe the change of accuracies with respect to the changes of number of learners training data set for base learners. The results are shown on Table 4.2.

The third test is about the changes of accuracy on different data sets on different data sets by using C4.5 algorithm as the base learner. The results of the tests are shown on Table 4.3.

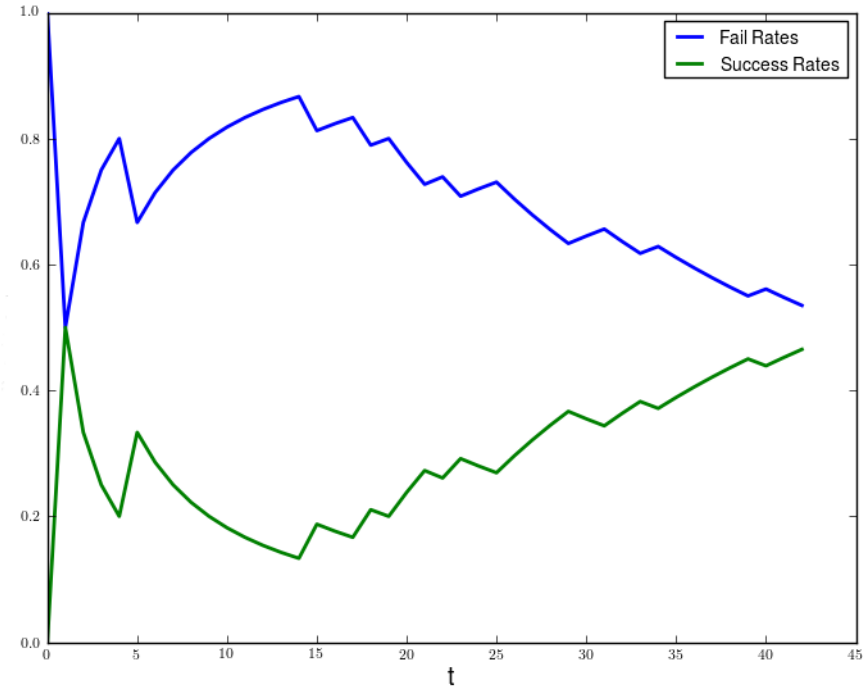

Figure 4.12: The change of success and fail rates with 10 agents and initially 6 words in the lexicon.

Accuracies *wrt* differing sampling percentage					
Algorithm	Sampling Percentage				
	5	10	15	20	25
Majority Voting	**84.814**	**80.123**	**78.518**	75.925	77.160
Categorization Games	76.172	73.827	76.543	76.172	76.790
CGCRBU	76.172	76.419	77.777	**78.888**	**78.518**

Table 4.2: Table of accuracies of model combination algorithms with Naive Bayes algorithm as base learner on segmentation data set. When the sampling percentage is small, majority voting seems to outperform CGCRBU. But as the sampling percentage increases the accuracy of majority voting drops and the CGCRBU's accuracy increases.

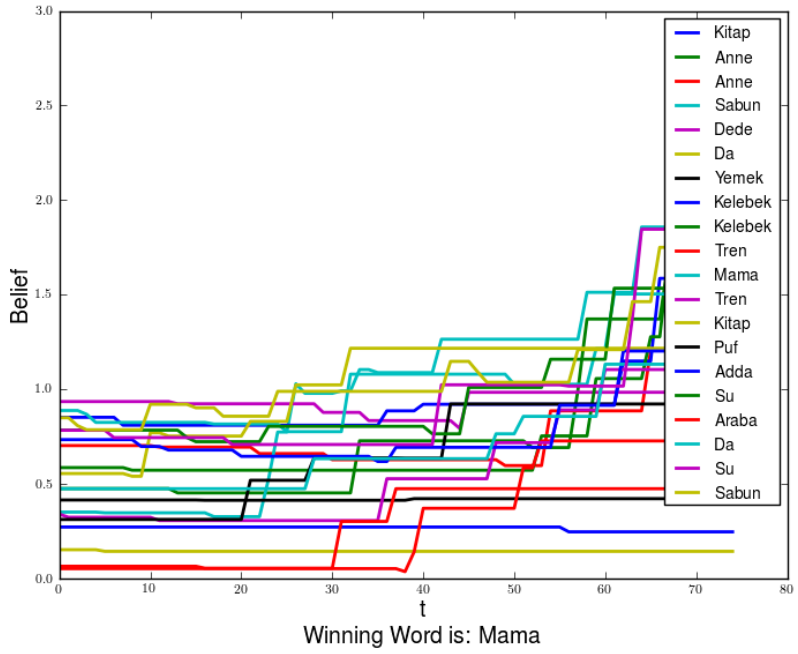

Figure 4.13: The change of belief scores in CGCRBU within a game of 20 agents and 15 words in the lexicon. We have used $\eta_s = 10 \times \eta_h$ in this test. In the beginning as because of the failed dialogues belief scores slightly decreases. Later on as the number of successful games start to increase, the belief scores started to increase as well.

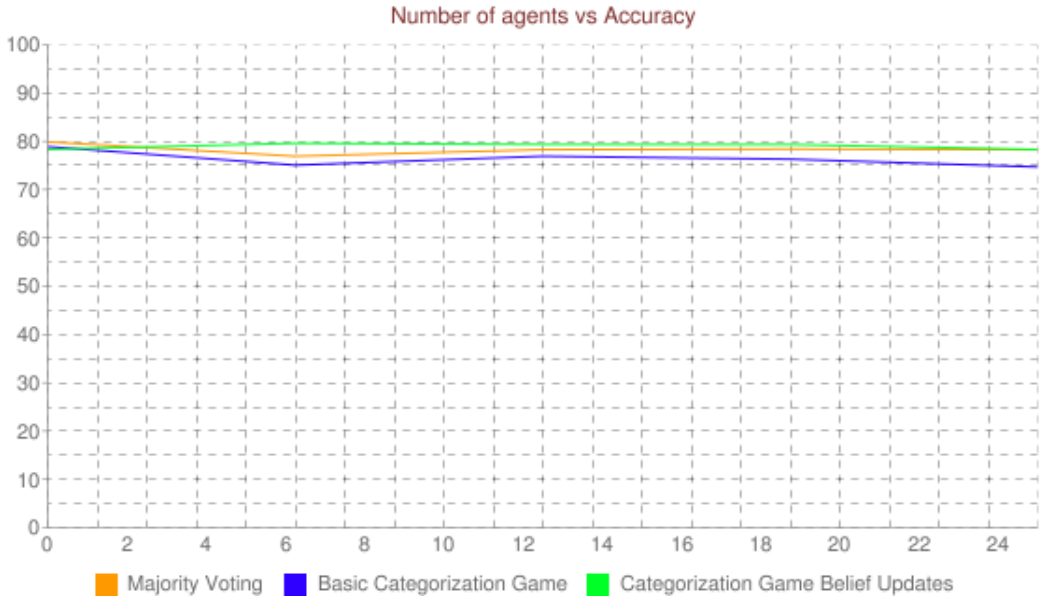

Figure 4.14: The graph of change of accuracies with respect to number of agents. Y axis is the accuracy, X axis is depicting the time.

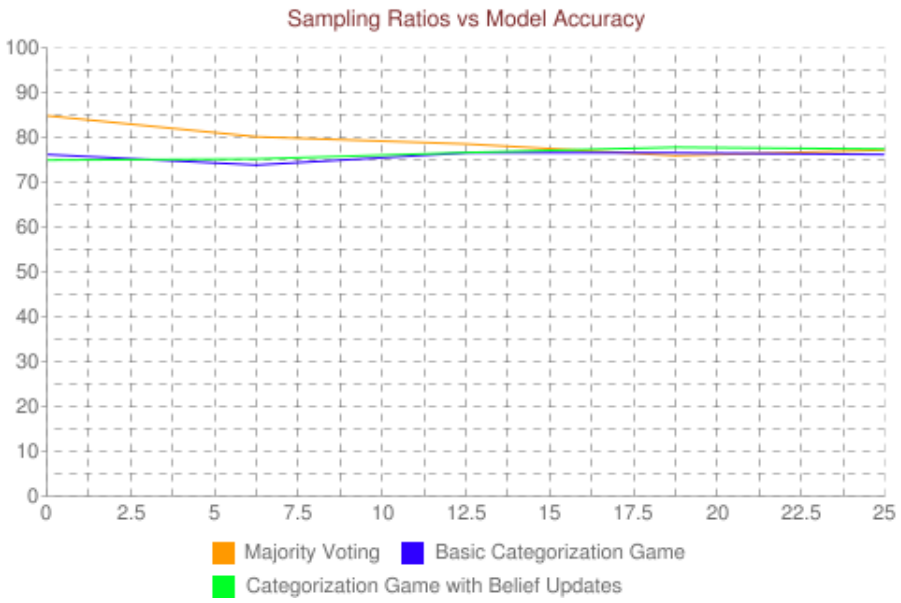

Figure 4.15: The graph of change of accuracies with respect to sampling percentages. X axis is showing the change of sampling ratios, Y axis is for accuracy.

Accuracies of model combination algorithms *wrt* differing number of agents					
Algorithm	Number of agents				
	5	10	15	20	25
Majority Voting	**79.876**	76.913	78.271	78.395	78.271
Categorization Games	78.888	75.0617	76.913	76.296	74.691
CGCRBU	79.382	**80.617**	**80.370**	**80.370**	**79.382**

Table 4.3: Table of accuracies of model combination algorithms with Naive Bayes as base learner on segmentation data set. Majority voting's accuracy drops rapidly as the number of agents gets larger. But the number of agents in the game does not seem to have significant effect on CGCRBU.

Accuracies of C4.5 as base learning algorithm *wrt* type of the data set			
Algorithm	Data Set		
	MNIST	GTVS	Segmentation
Majority voting	**83.625**	97.698	93.45
Categorization Games	68.05	98.025	88.271
CGCRBU	82	**99.0117**	**93.580**

Table 4.4: Table of accuracies of model combination algorithms with C4.5 Decision Tree learning algorithm as the base learner on different data sets. CGCRBU and Majority Voting's performance are very close. But CGCRBU is superior on GTVS and Segmentation data sets. CG has very low accuracy on MNIST.

The fourth test is about the change of accuracy on different data sets on different data sets by using NB algorithm as the base learner. The results of the tests are shown on Table 4.4.

Accuracies *wrt* type of the data set			
Algorithm	Data Set		
	MNIST	GTVS	Segmentation
Majority Voting	69.475	74.431	77.654
Category Games	67.175	**74.903**	72.716
CGCRBU	**69.625**	74.898	**79.626**

Table 4.5: Table of accuracies of model combination algorithms with Naive Bayes learning algorithm as base learner on different data sets.

Accuracies of SVM as base learner on Segmentation Data Set	
Algorithm	Accuracy
Majority Voting	84.426
Weighted Majority Voting	**84.647**
CGCRBU	84.015

Table 4.6: Table of accuracies of model combination algorithms with SVM -CSVC where $\gamma = 0.01$ and $\epsilon = 0.01$- learning algorithm as base learner on segmentation data set. These results are obtained after averaging results of 5 consecutive tests. There is no significant difference among the algorithms. We have used $\eta_s = 10 \times \eta_h$ in this test.

Accuracies of Kstar as base learner on Segmentation Data Set	
Algorithm	Accuracy
Majority Voting	94.6913
Categorization Game	87.6913
CGCRBU	**95.0617**

Table 4.7: Table of accuracies of model combination algorithms with Kstar(Cleary and Trigg, 1995) algorithm -which is a instance based classifier similar to KNN discussed in Appendix A- learning algorithm as base learner on segmentation data set.

The average amount of time for agents in the population to agree on a specific category can be determined by $\chi = \frac{\omega^f + \omega^s}{M}$ where ω^f is the total number of fails, ω^s is the total number of successes and the M is the number of examples in the whole data set. For each example this number is N in majority voting where N is the number of agents. The values of χ with respect to the change in number of agents in the population is shown on table 3.5. The values of *chi* increases exponentially after a certain value. This change is more rapid in CG. When N is lower than 10, the χ value for CGCRBU is smaller than N.

χ values *wrt* the number of agents					
Algorithm	Number of agents				
	5	**10**	**15**	**20**	**25**
CG	2.171	18.406	49.866	92.644	147.744
CGCRBU	2.767	9.739	19.866	80.987	79.135

Table 4.8: Table of values of χ's with CG and CGCRBU on segmentation data set using NB as a base-learner.

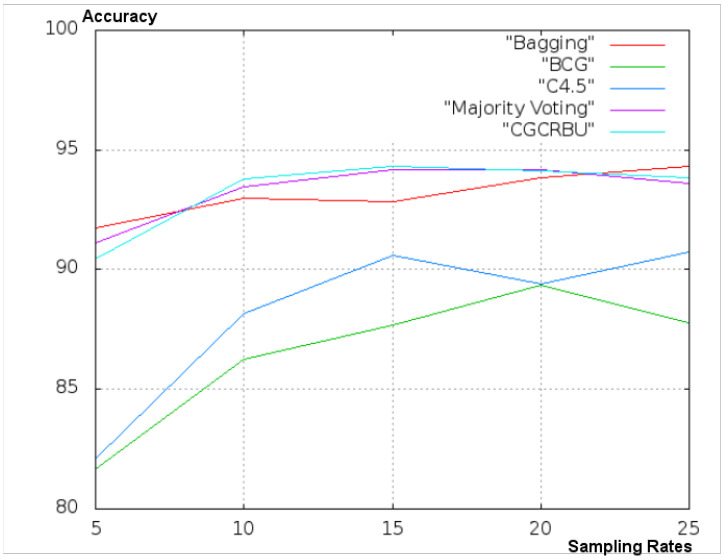

Figure 4.16: The comparison of different model combination algorithms and our proposed algorithms with respect to changing sampling ratio.

As seen in Figure 4.4.2, there is a huge gap between BCG and base learner performed poorly compared to the other algorithms.

Accuracies *wrt* the number of agents						
Algorithm	Accuracies					
	5	10	15	20	25	Avg
Majority Voting	**92.469**	92.710	94.171	93.827	94.691	93.465
Categorization Games	87.001	87.235	87.666	87.04	91.331	88.055
CGCRBU	92.172	**93.795**	**94.182**	**94.102**	**94.975**	**93.845**

Table 4.9: Table of accuracies of model combination algorithms with C4.5 algorithm as base learner with respect to the change in number of agents on segmentation data set. On average CGCRBU performs better than the other algorithms in terms of accuracy.

Accuracies *wrt* the sampling percentages						
Algorithm	Accuracies					
	5	10	15	20	25	Avg
Majority Voting	91.111	93.450	94.171	**94.197**	93.580	93.301
Categorization Games	81.691	86.250	87.666	89.345	87.777	86.546
CGCRBU	90.456	**93.795**	**94.308**	94.112	93.839	**93.302**
Bagging	**91.728**	92.963	92.839	93.827	**94.321**	93.135
C4.5 (Base-learner)	82.098	88.148	90.617	89.382	90.7407	88.197

Table 4.10: Table of accuracies with respect to the change in sampling percentage where C4.5 algorithm is the base learner on segmentation data set.

Because of the randomness involved in the CG and CGCRBU, the accuracy of classification tasks vary in each run. Hence in Table 4.9 and Table 4.10 each test ran 10 times and their accuracies are averaged. On average CGCRBU outperforms other algorithms in both tests.

CHAPTER 5

Conclusion and Discussions

"*Hominid and human evolution took place over millions and not billions of years, but with the emergence of language there was a further acceleration of time and the rate of change.*"
–William Irwin Thompson

Algorithms inspired from the way that nature deals with complex problems are known to perform well for solving challenging problems in computer science. Chaotic behaviors are ubiquitous in nature and social systems. Artificial intelligence and specifically machine-learning tries to solve complex problems occurring as a result of complex processes. Therefore computer scientists frequently use nature inspired algorithms for AI and particularly optimization problems, like solving traveling salesman problem by using algorithms such as, genetic algorithms, connectionist systems and ant-colony optimizations. Emergence and evolution of languages are an optimization process that communicative efficiency provides Darwinian fitness which translates to reproductive success (Niyogi, 2006).

In this thesis, we've proposed two new social collective learning algorithms inspired by late Wittgensteinian language games. This framework is also a conceptualization of the social interactions between individuals for modeling the emergence of language. In that sense we claim that this algorithm can be used as a ideal social learning algorithm that several agents tries to categorize a particular object by discussing with others. Using this framework we have also observed the emergence of categories.

We have done several tests and developed two simple games, categorization game (CG) and categorization game with confidence rated belief updates (CGCRBU) and empirically observed their characteristics and emergence of categories under certain circumstances.

Our tests have shown that CG and CGCRBU converge to an optimal category, in a reasonable time interval. But the running time and accuracy of each game varies in each run[1]. According to the results we obtained from the tests in Chapter 4, the categorization games is a competitive alternative, compared to the majority voting and bagging algorithms[2]. In some tests we obtained better results than the majority voting algorithm without having any performance burdens. On average, CGCRBU converges to an optimal category with less than N iterations where N is the number of agents when the number of agents is small. The increase in number of agents and sampling ratios seem to have a significant effect on the accuracy of the algorithms.

Learning with a language game is a by product of social activities among several agents as in the real world[3]. By using categorization games, we have shown how a population of agents might arrive to a shared language (or to linguistic coherence) through the interactions among the learning agents without any central control at all. In this sense, we have used BCG and CGCRBU as a social learning algorithm.

The major disadvantage of categorization games is that they are not stable.

There are several directions that this thesis can be brought to:

1. A theoretical and mathematical framework for categorization games can be defined, in order to find the lower and upper bounds for the ”Mean Square Error“ of algorithm.

2. The categorization games can be expanded with further ideas, such as using them with AdaBoost (see Appendix C) or LogitBoost algorithm (Friedman et al., 2000) by simply replacing weighted majority voting algorithm.

3. Individuals in real world uses their past experience when they decide on a specific topic and they gain new experiences from the errors they make. Therefore using semi-supervised learning with self-training (Mihalcea, 2004) to train agents with their past decisions that agents made might be a more plausible way to model social learning.

4. The statistical mechanics of emergence of categories in language games should be analyzed to better understand how language games behave in certain situations.

[1] Also consider that our pseudo random number generator uses current time in milliseconds as the seed and so in each run random numbers generated varies.

[2] Bagging uses majority voting with bootstrapping.

[3] Such as second language learning

5. The games we have proposed, should be tested on different data sets and with different learning algorithms, to better understand the behavior of these games. These algorithms can be tested in the heterogeneous ensemble learning context as well.

6. Categories in computer science are usually represented ontologically and hierarchically with relations among them that establishes the associations. Moreover humans seem to have a tendency to relate semantically close categories in order recall back easily. Thus using for example an associated memory might be better way to represent the categories and they may perform better in pattern recognition tasks.

Bibliography

Alpaydin, E. (2010). *Introduction to machine learning 2e*. The MIT Press.

Barber, D. (2010). Bayesian reasoning and machine learning.

Baronchelli, A., C. Cattuto, V. Loreto, and A. Puglisi (2007). Complex systems approach to the emergence of language.

Baronchelli, A., L. Dall'Asta, A. Barrat, and V. Loreto (2005). Strategies for fast convergence in semiotic dynamics. *Arxiv preprint physics/0511201*.

Baronchelli, A., L. Dall'Asta, A. Barrat, and V. Loreto (2006). Topology induced coarsening in language games. *Physical Review E 73*(1), 15102.

Baronchelli, A., T. Gong, A. Puglisi, and V. Loreto (2010). Modeling the emergence of universality in color naming patterns. *Proceedings of the National Academy of Sciences 107*(6), 2403.

Baronchelli, A., V. Loreto, and A. Puglisi (2008). Complex Systems Approach to Natural Categorization. In *The Evolution of Language: Proceedings of the 7Th International Conference*, pp. 399. World Scientific Pub Co Inc.

Baronchelli, A., V. Loreto, and L. Steels (2008). In-depth Analysis of the Naming Game Dynamics: The Homogenous Mixing Case. *International Journal of Modern Physics C 19*(5), 785–812.

Beckner, C., R. Blythe, J. Bybee, M. H. Christiansen, W. Croft, N. C. Ellis, J. Holland, J. Ke, D. Larsen-Freeman, and T. Schoenemann (2009). Language Is a Complex Adaptive System: Position Paper. *Language Learning 59*(s1), 1–26.

Belpaeme, T. (2001). Simulating the formation of color categories. In *International Joint Conference on Artificial Intelligence*, Volume 17, pp. 393–400. Citeseer.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning 2*(1), 1–127. Also published as a book. Now Publishers, 2009.

Biletzki, A. and A. Matar (2010). Ludwig wittgenstein. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Spring 2010 ed.).

Bock, R., R. Bock, and W. Krischer (1998). *The data analysis briefbook*. Springer Verlag.

Borensztajn, G. (2006). Luc Steels, the Talking Heads Experiment and Cognitive Philosophy A tutorial accompanying the presentation in Current Issues PART I-Philosophical Background.

Bottou, L. (2011). From machine learning to machine reasoning. *CoRR abs/1102.1808*.

Bousquet, O., S. Boucheron, and G. Lugosi (2004). Introduction to statistical learning theory. *Advanced Lectures on Machine Learning*, 169–207.

Bratman, M. (1992). Shared cooperative activity. *The Philosophical Review 101*(2), 327–341.

Brodley, C. E. and M. A. Friedl (1999). Identifying mislabeled training data. *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH 11*, 131–167.

Canini, M., W. Li, A. Moore, and R. Bolla (2009). GTVS: Boosting the collection of application traffic ground truth. *Traffic Monitoring and Analysis*, 54–63.

Cattuto, C., V. Loreto, and L. Pietronero (2007). Semiotic dynamics and collaborative tagging. *Proceedings of the National Academy of Sciences 104*(5), 1461.

Chalmers, D. J. (2006). Strong and weak emergence. In *The Re-Emergence of Emergence*. Oxford University Press.

Chang, C.-C. and C.-J. Lin (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Christiansen, M. and S. Kirby (2003). Language evolution: Consensus and controversies. *Trends in Cognitive Sciences 7*(7), 300–307.

Clark, H. (1996). Using language. *Computational Linguistics 23*(4).

Cleary, J. and L. Trigg (1995). K^*: An Instance-based Learner Using an Entropic Distance Measure. In *Machine Learning-International Workshop THEN Conference-*, pp. 108–114. Citeseer.

De Beule, J., B. De Vylder, and T. Belpaeme (2006). A cross-situational learning algorithm for damping homonymy in the guessing game. In *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, pp. 466–472. Citeseer.

De Vylder, B. and K. Tuyls (2006). How to reach linguistic consensus: A proof of convergence for the naming game. *Journal of theoretical biology 242*(4), 818–831.

Dietterich, T. (2002). Ensemble learning.

Duda, R., P. Hart, and D. Stork (2001). *Pattern classification*, Volume 2. Citeseer.

Easley, D. and J. Kleinberg (2010). *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge Univ Pr.

Elio, R. and F. Pelletier (1994). The effect of syntactic form on simple belief revisions and updates. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society: Atlanta, Georgia, 1994*, pp. 260. Lawrence Erlbaum Associates.

Elio, R. and F. Pelletier (1997). Belief change as propositional update. *Cognitive Science 21*(4), 419–460.

Ellson, J., E. Gansner, E. Koutsofios, S. North, and G. Woodhull (2003). Graphviz and dynagraph – static and dynamic graph drawing tools. In M. Junger and P. Mutzel (Eds.), *Graph Drawing Software*, pp. 127–148. Springer-Verlag.

Freund, Y. and R. Schapire (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pp. 23–37. Springer.

Freund, Y., R. Schapire, and N. Abe (1999). A short introduction to boosting. *JOURNAL-JAPANESE SOCIETY FOR ARTIFICIAL INTELLIGENCE 14*, 771–780.

Friedman, J., T. Hastie, and R. Tibshirani (2000). Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The annals of statistics 28*(2), 337–407.

Goodman, N., J. Tenenbaum, J. Feldman, and T. Griffiths (2008). A Rational Analysis of Rule-Based Concept Learning. *Cognitive Science 32*(1), 108–154.

Griffiths, T., C. Kemp, and J. Tenenbaum (2008). Bayesian models of cognition. *Cambridge handbook of computational cognitive modeling*, 59–100.

Griffiths, T. and A. Yuille (2008). A primer on probabilistic inference. *The probabilistic mind: Prospects for Bayesian cognitive science*, 33–57.

Hagberg, A. A., D. A. Schult, and P. J. Swart (2008, August). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, pp. 11–15.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter 11*(1), 10–18.

Hastie, T., R. Tibshirani, and J. H. Friedman (2003, July). *The Elements of Statistical Learning* (Corrected ed.). Springer.

Hinton, G., S. Osindero, and Y. Teh (2006). A fast learning algorithm for deep belief nets. *Neural computation 18*(7), 1527–1554.

Holland, J. (1992). Complex adaptive systems. *Daedalus 121*(1), 17–30.

Holland, J. (2006). Studying complex adaptive systems. *Journal of Systems Science and Complexity 19*(1), 1–8.

Hopper, P. (1998). Emergent grammar. *The new psychology of language: Cognitive and functional approaches to language structure 1*, 155–176.

Jaeger, H., A. Baronchelli, T. Brisco, M. H. Christiansen, T. Griffiths, G. Jäger, S. Kirby, N. Komarova, P. J. Richerson, L. Steels, and J. Triesch (2009). *What can mathematical, computational and robotic models tell us about the origins of syntax?* (Strüngmann Forum Reports 3 ed.)., pp. 385–410. MIT Press.

John, G. and P. Langley (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the eleventh conference on uncertainty in artificial intelligence*, Volume 1, pp. 338–345. Citeseer.

Kay, P. and W. Kempton (1984). What Is the Sapir-Whorf Hypothesis? *American Anthropologist 86*(1), 65–79.

Kearns, M. and L. Valiant (1994). Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM (JACM) 41*(1), 67–95.

Kemerling, G. (2001). Ludwig Wittgenstein: Analysis of Language.

Kim, Y., W. Street, and F. Menczer (2002). Meta-evolutionary ensembles. In *IEEE Intl. Joint Conf. on Neural Networks*, pp. 2791–2796. Citeseer.

Klee, R. (1984). Micro-determinism and concepts of emergence. *Philosophy of Science 51*(1), 44–63.

Kohavi, R. and R. Quinlan (1999). Decision tree discovery. In *In Handbook of Data Mining and Knowledge Discovery*, pp. 267–276. University Press.

Lansing, J. (2003). Complex adaptive systems. *Annual Review of Anthropology 32*, 183–204.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (2002). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324.

Lenaerts, T., B. Jansen, K. Tuyls, and B. De Vylder (2005). The evolutionary language game: An orthogonal approach. *Journal of Theoretical Biology 235*(4), 566–582.

Littlestone, N. and M. Warmuth (2002). The weighted majority algorithm. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pp. 256–261. IEEE.

Lovejoy, A. (1927). The Meanings of 'Emergence' and its Modes. *Philosophy 2*(06), 167–181.

MacWinney, B. (1998). Models of the emergence of language. *Annual review of psychology 49*.

Matsumoto, M. and T. Nishimura (1998, January). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul. 8*, 3–30.

Mihalcea, R. (2004). Co-training and self-training for word sense disambiguation. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2004)*.

Morgan, C. (1929). The case for emergent evolution. *Philosophy 4*(13), 23–38.

Niyogi, P. (2006). *The computational nature of language learning and evolution*. MIT press Cambridge, MA.

Pacuit, E. and R. Parikh (2006). Social interaction, knowledge, and social software. *Interactive Computation*, 441–461.

Parikh, R. (1995). Language as social software. In *International Congress on Logic, Methodology and Philosophy of Science*, Volume 415.

Pepper, S. (1926). Emergence. *The Journal of Philosophy 23*(9), 241–245.

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia 4*(2), 1883.

Polikar, R. (2009). Ensemble learning. *Scholarpedia 4*(1), 2776.

Puglisi, A., A. Baronchelli, and V. Loreto (2008). Cultural route to the emergence of linguistic categories. *Proceedings of the National Academy of Sciences 105*(23), 7936.

Quinlan, J. (1986). Induction of decision trees. *Machine learning 1*(1), 81–106.

Quinlan, J. (1993). *C4. 5: programs for machine learning*. Morgan Kaufmann.

Roberson, D., I. Davies, and J. Davidoff (2000). Color categories are not universal: Replications and new evidence from a stone-age culture. *Journal of Experimental Psychology General 129*(3), 369–398.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review 33*(1), 1–39.

Rosch, E. (1999). Principles of categorization. *MIT Press, Cambridge, MA*.

Ruta, D. and B. Gabrys (2005). Classifier selection for majority voting. *Information fusion 6*(1), 63–81.

Schaul, T. and J. Schmidhuber (2010). Metalearning. *Scholarpedia 5*(6), 4650.

Schoenemann, P. and W. Wang (1996). Evolutionary principles and the emergence of syntax. *Behavioral and Brain Sciences 19*(04), 646–647.

Sewell, M. (2008). VC Dimension.

Shannon, C. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review 5*(1), 3–55.

Siskind, J. (1996). A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition 61*(1-2), 39–91.

Smith, A. (2001). Establishing communication systems without explicit meaning transmission. *Advances in artificial life*, 381–390.

Smith, A. (2003). Intelligent meaning creation in a clumpy world helps communication. *Artificial Life 9*(2), 175–190.

Smith, K., S. Kirby, and H. Brighton (2003). Iterated learning: A framework for the emergence of language. *Artificial Life 9*(4), 371–386.

Staab, S., S. Santini, F. Nack, L. Steels, and A. Maedche (2005). Emergent semantics. *Intelligent Systems, IEEE 17*(1), 78–86.

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

Steels, L. (1996). Perceptually grounded meaning creation. In *Proceedings of the International Conference on Multi-Agent Systems*, pp. 338–344.

Steels, L. (1998). The origins of syntax in visually grounded robotic agents. *Artificial Intelligence 103*(1-2), 133–156.

Steels, L. (2000). Language as a complex adaptive system. In *Parallel Problem Solving from Nature PPSN VI*, pp. 17–26. Springer.

Steels, L. (2006). Semiotic dynamics for embodied agents. *IEEE Intelligent Systems*, 32–38.

Steels, L. and F. Kaplan (1998). Stochasticity as a source of innovation in language games. *Artificial life six*, 368.

Steels, L. and F. Kaplan (1999). Collective learning and semiotic dynamics. *Advances in artificial life*, 679–688.

Steels, L. and F. Kaplan (2002). Bootstrapping grounded word semantics. *Linguistic Evolution Through Language Acquisition*, 53.

Steels, L. and A. MacIntyre (1998). Spatially distributed naming games. *Advances in complex systems 1*, 301–324.

Sylvester, J. and N. Chawla (2006). Evolutionary ensemble creation and thinning. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pp. 5148–5155. IEEE.

Tenenbaum, J. and T. Griffiths (2001). Generalization, similarity, and Bayesian inference. *Behavioral and Brain Sciences 24*(04), 629–640.

Tenenbaum, J., T. Griffiths, and C. Kemp (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences 10*(7), 309–318.

Theodoridis, S. and K. Koutroumbas. *Pattern recognition. 2003*. Academic Press.

Thomasson, A. (2010). Categories. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2010 ed.).

Tomasello, M. and M. Carpenter (2007). Shared intentionality. *Developmental Science 10*(1), 121–125.

Traulsen, A., C. Hauert, H. De Silva, M. Nowak, and K. Sigmund (2009). Exploration dynamics in evolutionary games. *Proceedings of the National Academy of Sciences 106*(3), 709.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM 27*(11), 1134–1142.

Vogt, P. (2002). Bootstrapping grounded symbols by minimal autonomous robots. *Evolution of Communication 4*(1), 87–116.

Vogt, P. and H. Coumans (2003). Investigating social interaction strategies for bootstrapping lexicon development. *Journal of Artificial Societies and Social Simulation 6*(1), 1.

Wang, H., W. Fan, P. Yu, and J. Han (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 226–235. ACM.

Weber, R. (2003). Learning with no feedback in a competitive guessing game. *Games and Economic Behavior 44*(1), 134–144.

Wikipedia (2010a). Bayesian inference — Wikipedia, the free encyclopedia. [Online; accessed 11-12-2010].

Wikipedia (2010b). Concept learning — Wikipedia, the free encyclopedia. [Online; accessed 11-12-2010].

Wikipedia (2010c). Emergence — Wikipedia, the free encyclopedia. [Online; accessed 08-12-2010].

Wikipedia (2010d). Information gain in decision trees — Wikipedia, the free encyclopedia. [Online; accessed 11-12-2010].

Wikipedia (2010e). K-means — Wikipedia, the free encyclopedia. [Online; accessed 18-12-2010].

Wikipedia (2010f). Naive bayes classifier — Wikipedia, the free encyclopedia. [Online; accessed 11-12-2010].

Wikipedia (2010g). Problem of induction — Wikipedia, the free encyclopedia. [Online; accessed 08-12-2010].

Wittgenstein, L. (1922). Tractatus logico-philosophicus. *London: Routledge, 1981*.

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

Zhou, Z. (2008). Ensemble learning. *Encyclopedia of Biometrics*, 1–5.

APPENDIX A

On Supervised Machine learning Algorithms

A.1 Decision Trees and C4.5

Trees are one of the most common and powerful data structure in the computer science. Building trees is a cheap procedure; but using them is even cheaper (O(log(N))). Therefore it is an attractive data structure for machine learning studies. An important advantage of using decision trees is that they are transparent unlike Neural Networks or other black box approaches. So they are easy to interpret by the humans as in Figure A.1.

Simply a decision tree has a recursive structure that (Kohavi and Quinlan, 1999):

- a **leaf** node with class value or,
- a **test** node that has more than two outcomes

C4.5 algorithm is a popular decision tree algorithm proposed by Quinlan and it is the successor of ID3 algorithm (Quinlan, 1993).

There are several design decisions that have to be taken in the decision tree construction (Theodoridis and Koutroumbas, Theodoridis and Koutroumbas):

- At each node, a set of questions to be asked has to be decided. Each question corresponds to a set of **children** split nodes and each node t corresponds to a set of examples D_s^t from training data D^t. Splitting a node is equivalent to the splitting the the dataset into **disjoint** descendant subsets $\{D_0^t, \ldots, D_n^t\}$. The root of tree is associated with the whole training dataset D^t. For every split node, following considerations are true:

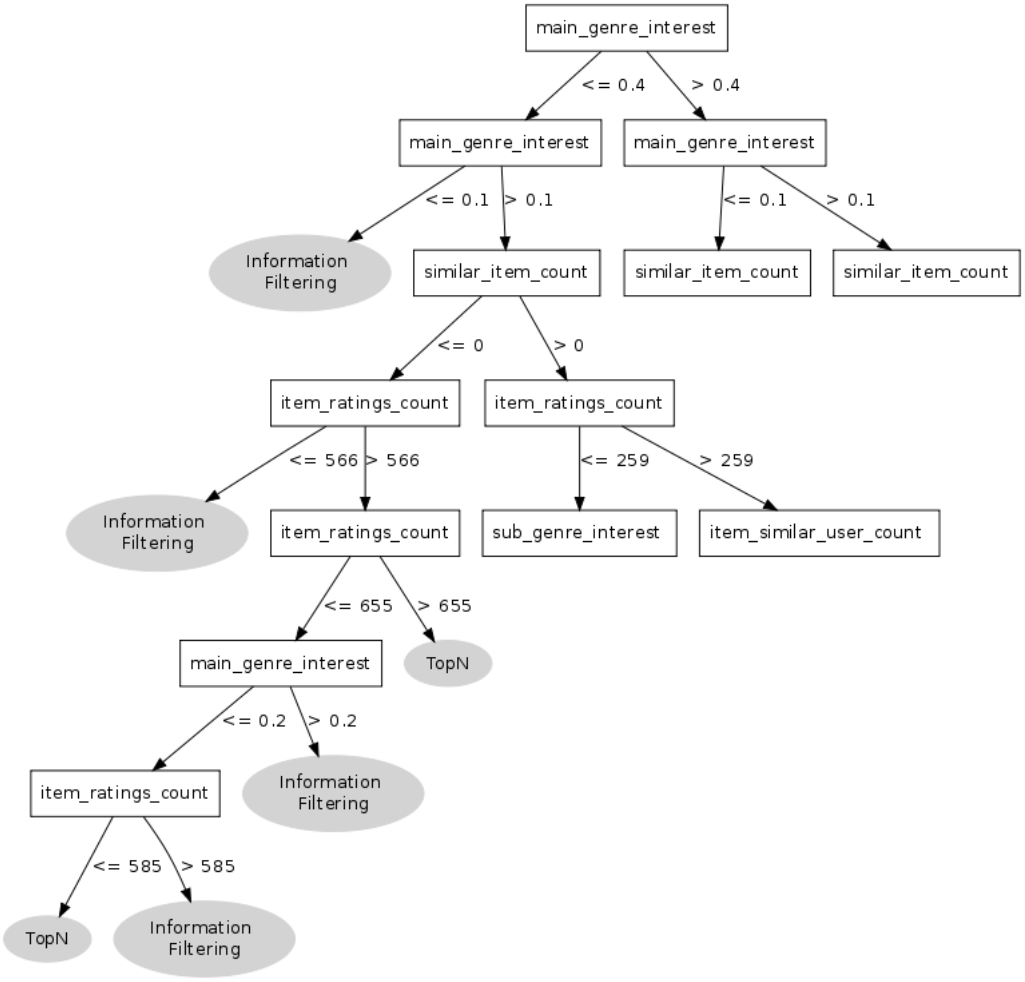

Figure A.1: **An Example Decision Tree:** This figure shows a decison tree with nodes and leaves. As seen from the figure it is possible to classify an example without testing all the features.

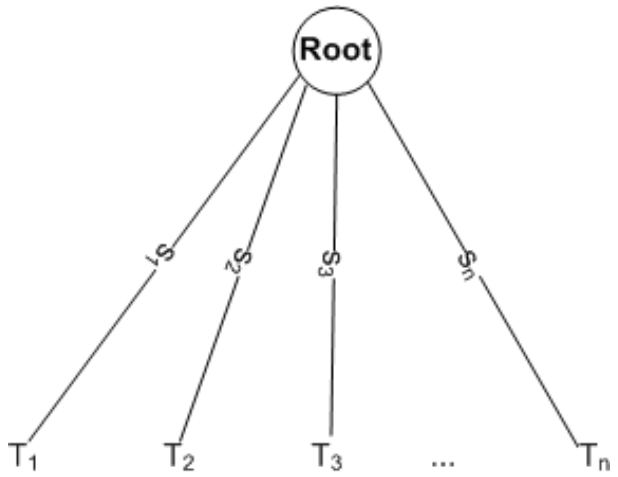

Figure A.2: **Growing Decision Tree:** T_i's are the result of growing tree from the D^t

$$D^t = D_0^t \cup D_1^t \ldots D_n^t$$
$$\emptyset = D_0^t \cap D_1^t \ldots D_n^t$$

- A splitting criterion must be adopted for determining which split criterion is best to use among the set of candidate ones.

- A stop-splitting criterion is required for controlling the growth of the tree and a node is declared as a terminal one (leaf).

A.1.1 Decision tree construction

Most of the decision tree construction algorithms use a top-down construction with a divide and conquer technique:

- If all the examples in D^t belongs to the same class C_j the decision tree is a leaf.

- Otherwise let S be a splitting criterion with outcomes $\{s_1, \ldots, s_n\}$ that produces a non-trivial partition of D^t and let's denote that D_i^t is a nontrivial partition created by S_i. Then the decision tree produced will be as in A.2.

A.1.2 Splitting Criterion for C4.5 Algorithm

C4.5 algorithm uses information gain for splitting nodes as well as ID3 algorithm (Quinlan, 1986). C4.5 uses Information-Gain as the splitting criteria (Kohavi and Quinlan, 1999).

Information gain uses the "Entropy", the measure of disorder in the data. The entropy is calculated according to the relative frequency of class C_j with respect to the training dataset D^t:

$RF(C_j, D^t) = \frac{|C_j|}{|D^t|} = P(C_j|D^t)$

The entropy H of D^t is :

$H(D^t) = E(I(D^t))$

E is the expected value and I is the information content of D^t where in our case $I(D^t) = -log_2(D^t)$.

Entropy H which determines the information content of a message that identifies the class of instance in D^t will be:

$H(D^t) = -\sum_{j=1}^{x} RF(C_j, D^t) log_2(RF(C_j, D^t))$

As a result of assumption that our message is binary encoded we use the log_2 which determines the information content of the message (Shannon, 2001).

By definition the information gain is the change in entropy from a prior state to a state that takes some information as given (Wikipedia, 2010d):

$IG(D^t) = H(D^t) - H(D^t|a)$

The information gain will be as below where D_i^t is the sub-partition of training dataset D^t and Split Test S^t_j:

$IG(D^t, S) = H(D^t) - \sum_{j=1}^{n} \frac{\|S^t_j\|}{\|S\|} H(D^t_j)$

Decision Tree will choose the split test that maximizes the information gain.

The drawback of information gain is that information gain favors the attributes that can take on a large number of distinct values.So for instance, $IG(D^t, S)$ is maximized when the each

S^t_j has one instance. The information gain ratio solves this problem by taking into account the potential information from the partition itself.

$P(D^t, S) = -\sum_{j=1}^{n} \frac{\|S^t_j\|}{\|S\|} log(\frac{\|S^t_j\|}{\|S\|})$

$IG^R(D^t, S) = \frac{IG(D^t,S)}{P(D^t,S)}$

The test S^t_i that maximizes IG^R will be selected as the split node.

A.1.3 Stop Splitting Criterion

There are various methods for stopping splitting criterion; a possibility is to adopt a threshold T and stop splitting if the maximum IG^R is less than T. Other alternatives are to stop splitting either if the cardinality of the subset X_t is small enough or that all points in it belong to a single class (Theodoridis and Koutroumbas, Theodoridis and Koutroumbas).

A.1.4 C4.5 Construction Pseudocode

The general overview of decision tree construction algorithm is shown in Algorithm A.1.1:

A.2 Bayesian Inference and Bayesian Classification

Probabilistic models for human cognition aim to explain cognition by appealing to the principles of probability theory and statistics, which dictate how an agent should act in situations that involve uncertainty. Probabilistic models are very useful for modeling real-world phenomena which includes certain amount of uncertainty.

A.2.1 Bayesian Inference

In practical usage, "Bayesian inference" refers to the use of a prior probability over hypotheses to determine the likelihood of a particular hypothesis given some observed evidence; that is, the likelihood that a particular hypothesis is true given some observed evidence (the so-called posterior probability of the hypothesis) comes from a combination of the inherent likelihood

Algorithm A.1.1 BUILDTREE(*root*)

1. **if** all instances are in the same class **then**
2. add leaf with Class C_j to the *root*
3. **return** *root*
4. **end if**
5. **repeat**
6. **if** $IG^R > maxIgr$ **then**
7. $maxIgr \leftarrow IG^R$
8. **end if**
9. **until** No Attribute left
10. **if** $maxIgr < T$ **then**
11. **return** *root*
12. **end if**
13. $attBest \leftarrow argmax(IG_R)$
14. create decision node *dNode* that splits from *attBest*
15. **return** BUILDTREE(*dNode*)

(or prior probability) of the hypothesis and the compatibility of the observed evidence with the hypothesis (or likelihood of the evidence, in a technical sense). (Tenenbaum et al., 2006)

Bayesian inference is regularly used by the probabilistic models of cognition, showing the central role of Bayesian inference in reasoning under uncertainty (Tenenbaum et al., 2006), (Griffiths and Yuille, 2008), (Tenenbaum and Griffiths, 2001), (Griffiths et al., 2008).

Bayesian inference heavily relies on the Bayes Theory. So it would be nice to see how Bayes Theory is derived:

Probability theory is based on the following two rules:

Sum rule:

- $p(X) = \sum_i p(X, Y_i)$

and the product rule:

- $p(X, Y) = P(X|Y)p(Y) = P(Y|X)p(X)$

where, $\forall X_i \in X \rightarrow p(X_i) \geq 0$ and $\sum_i p(X_i) = 1$

and by using the product rule above we can get the *Bayes Rule*:

- $p(X|Y) = \frac{p(Y|X)p(X)}{P(Y)}$

In this formulation (Wikipedia, 2010a),

- Y is the *evidence* that has been observed
- X is the *hypothesis*
- $p(X)$ is the *prior probability* of X which has already been inferred when the new evidence becomes available
- $p(Y)$ is the *marginal probability* of Y: a priori probability
- $p(Y|X)$ is the *conditional probability* or *likelihood* in which given that you've observed the evidence Y, what is the probability of the hypothesis X being correct.
- $p(X|Y)$ is the *posterior probability* of hypothesis X given that evidence Y is observed.

and we can rewrite the bayes theorem as below:

- $posterior = \frac{prior \times likelihood}{evidence}$

also from the above statements we can infer:

- $p(Y|X) \propto p(X|Y)p(Y)$

Bayesian decision theory(BDT) introduces a loss function $L(X, \alpha Y)$ for the cost of making the decision αY when the input is Y and the true hypothesis is the X. BDT tries to minimize the the risk by selecting an appropriate $\alpha * (.)$ function that minimizes the risk, the expected loss function will be (Tenenbaum et al., 2006):

$R(\alpha) = \sum_{x_i \in \mathcal{X}, y_i \in \mathcal{Y}} L(x_i, \alpha y_i) P(x_i, y_i)$

The loss function is selected so that the same penalty is paid for all wrong decisions: $L(X, \alpha Y)$ if $\alpha Y \neq h$, $L(X, \alpha Y) = 0$ and $\alpha Y \neq h$ $L(X, \alpha Y) = 1$, so that we'll get a hinge loss. Therefore the best decision rule is the maximum a posteriori estimator (MAP) $\alpha^*(Y) = argmax P(X|Y)$.

A.2.2 Naive Bayesian Algorithm

Naive Bayes Classifiers are one of the most fundamental type of classifiers. A bayesian classifier estimates the likelihoods from the training data, but this process requires additional simplifications. One of the important assumption of Naive Bayesian classifier is that the probability model for a classifier is a conditional model Wikipedia (2010f), (Barber, 2010):

$P(C_i|D_i)$

Given that C_i is a the class that an example D_i belongs to. We know that D_i is a set of features, $F_0, F_1, F_2, \ldots F_j$, so we can rewrite the conditional probability as:

$P(C_i|F_0, F_1, \ldots, F_j)$

and according to above formulation, the bayes theorem will be:

$P(C_i|F_0, \ldots, F_j) = \frac{p(F_0,\ldots,F_j|C_i)p(C_i)}{p(F_0,\ldots,F_j)}$

we ignore the denominator in the bayes theorem because it is constant and doesn't depend on C_i, therefore we'll only $P(C_i, F_0, \ldots, F_j)$ is equivalent to:

$P(C_i, F_0, \ldots, F_j)$

$= P(C_i)P(C_i|F_0, F_1, \ldots, F_j)$

$= P(C_i)P(F_0|C_i)P(F_1|C_i, F_0)P(F_2|C_i, F_0, F_1)\ldots P(F_n|C_i, F_1, \ldots F_{j-1})$

The naiveness of naive bayes classifier originates from the conditional independence assumption:

$P(F_j|C_i, F_k) = P(F_j|C_k)$

if $j \neq k$ then the joint model will be,

$P(C_i, F_0, \ldots, F_j) = P(C_i)P(F_0|C_i)P(F_1|C_i)\ldots P(F_j|C_i)$

$= P(C_i)\prod_{j=1}^{n} P(F_j|C_i)$

Adding the $Z(evidence)$ scaling factor of conditional distribution over C_i for the features F_j our conditional model will be:

$P(C_i, F_0, \ldots, F_j) = \frac{1}{Z} P(C_i) \prod_{j=1}^{n} P(F_j|C_i)$

The corresponding classifier for the model can be expressed as:

$classify(F_0, \ldots, F_j) = argmax P(C_i) \prod_{j=1}^{n} P(F_j|C_i)$

Naive bayesian estimation for continuous attributes

The formulation shown above can be used confidently for the discrete attributes but, for the numeric attributes we have to use a parametric model like gaussian distribution John and Langley (1995):

$P(C_i, F_j) = \frac{1}{n} \sum_i g(F_j; \mu, \sigma)$

and the μ is the mean:

$\mu = \frac{1}{N} \sum_{k=1, f_k^j \in F_j} x_i$

The σ where σ^2 is the variance is calculated as,

$\sigma = \sqrt{\frac{1}{(N-1)} \sum_{k=1, f_k^j \in F_j} (x_i - \mu)^2}$

$g(f_j; \mu, \sigma)$ is the gaussian function,

$g(f_j; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(f_j - \mu)^2}{\sigma^2}}$

Naive bayesian document classifier pseudocode

Algorithm A.2.1 TRAINNAIVEBAYES(D^T)

1. **global** C
2. **for all** $c \in C$ **do**
3. $\quad \gamma^c \leftarrow \frac{|D_c^T|}{|D^T|}$
4. $\quad T \leftarrow concat(D_c^T)$
5. $\quad N \leftarrow noOfTokens(T)$
6. $\quad$ **for all** $X \in D^T$ **do**
7. $\quad\quad$ **for all** $f_i \in X$ **do**
8. $\quad\quad\quad P(f_i|c) = \frac{|T^{f_i}|}{|T|}$
9. $\quad\quad$ **end for**
10. $\quad$ **end for**
11. **end for**

Algorithm A.2.2 CLASSIFY(X)

1. **local** p
2. **for all** $c \in C$ **do**
3. $\quad p = P(c)$
4. $\quad$ **for all** $t \in X$ **do**
5. $\quad\quad p = p \times P(t|c)$
6. $\quad$ **end for**
7. **end for**
8. **return** p

A.3 k-NN

k-NN is a nonparametric classification algorithm. Therefore it doesn't do any assumption about the distribution of the data. Furthermore it is lazy-learner. Thus it only does a naive function approximation during the training and most of the computation is left to the classification phase.

k-NN uses distance metric and calculates its closest k neighbor according to this similarity distance. Most used similarity metric for k-NN is *Euclidean Distance*. However euclidean distance works only for numeric attributes. Most popular distance metrics for k-NN:

- Euclidean distance: Euclidean distance is the ordinary distance between two data points in $\mathbf{P}, \mathbf{Q}$ euclid space:
  $d^E(\mathbf{P}, \mathbf{Q}) = \sqrt{\sum_{i=1}^{n}(P_i - Q_i)^2}$
- Mahalanobis distance: Unlike Euclidean distance, Mahalanobis Distance takes into account the correlations of the data set and it is scale-invariant. Mahalanobis distance of a multivariate vector $x = (x_1, x_2, x_3, \ldots, x_N)^T$ from a group of values with mean $\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_N)^T$ and covariance matrix S is defined as:
  $d^M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}$

K-nn is also a non-parametric density estimation that is formalized as (Duda et al., 2001),

$P(x) = \frac{k}{(NV)}$

and where,

- V is the volume surrounding the sample x
- k is the number of samples in V
- N is the total number of samples

Algorithm of k-NN is fairly simple shown in Algorithm A.3.1; but usually for the problems that we don't know anything about the distribution is very efficient.

Pseudocode of k-NN with is as follows (Peterson, 2009):

In order to optimize the classification with k-NN, usually spatial data structures like kd-tree, ball-tree or R-Trees are being used. This diminishes the bottleneck in calculation of distances during the classification.

Some important features of the k-NN are:

- The k-NN can be used to classify data without requiring a specific model for generalization, this is called "instance-based learning" which is a kind of "lazy learning". The new unclassified data is compared to the training data and based on the similarity, it is classified.
- A distance/similarity metric for the data should be available.
- The k-NN classification is based solely on local information (only the k nearest neighboring data points are inspected during the classification process).
- The decision boundaries produced by the k-NN can be in any arbitrary shape.
- The classification is sensitive to the correct selection of k. If k is too small it may lead to "over-fitting" and for some problems you may need to fine-tune the model in order find the correct k for the model.
- For high-dimensional large datasets, classification will be slow if a specific data structure for the data is not chosen.

Algorithm A.3.1 KNN(D^T)

1. initialize the $n \times n$ distance matrix D, initialize the $\Omega \times \Omega$ confusion matrix C, set $t \leftarrow 0, TotAcc \leftarrow 0$, and set NumIterations equal to the desired number of iterations (re-partitions).
2. calculate distances between all the input samples and store in $n \times n$ matrix D. (For a large number of samples, use only the lower or upper triangular of D for storage since it is a square symmetric matrix.)
3. for $t \leftarrow 1$ to $NumIterations$ do
4. set $\mathbf{C} \leftarrow 0$, and $n_{total} \leftarrow 0$.
5. partition the input samples into κ equally-sized groups.
6. for fold $\leftarrow 1 to \kappa$ do
7. assign samples in the foldth partition to testing, and use the remaining samples for training. Set the number of samples used for testing as n_{test}.
8. set $n_{total} \leftarrow n_{total} + n_{test}$.
9. for $i \leftarrow 1 to n_{test}$ do
10. for test sample $\mathbf{x}_i$ determine the k closest training samples based on the calculated distances.
11. determine $\hat{\omega}$, the most frequent class label among the k closest training samples.
12. increment confusion matrix $\mathbf{C}$ by 1 in element $c_{\omega,\hat{\omega}}$, where ω is the true and ω the predicted class label for test sample $\mathbf{x}_i$. If $\omega = \hat{\omega}$ then the increment of +1 will occur on the diagonal of the confusion matrix, otherwise, the increment will occur in an off-diagonal.
13. determine the classification accuracy using $Acc = \frac{\sum_j^{\Omega} c_{jj}}{n_{total}}$ where c_{jj} is a diagonal element of the confusion matrix $\mathbf{C}$.
14. calculate $TotAcc \leftarrow TotAcc + Acc$.
15. calculate $AvgAcc \leftarrow TotAcc/NumIterations$

A.4 Statistical Learning Theory

A.4.1 VC(Vapnik-Chervonenkis) Dimension

VC dimensions proposed by Vapnik and Chervonenkis and its foundation has been established by them between 1971-1990. It measures the capacity of a hypothesis space. Capacity is the measure of the complexity and measures the expressive power or richness of a set of functions (Sewell, 2008).

I'll adapt here Alpaydin's definition of VC dimensions (Alpaydin, 2010):

"Let us say we have a data set containing N points. These N points can be labeled in $2N$ ways as positive and negative. Therefore, $2N$ different learning problems can be defined by N data points. If for any of these problems, we can find a hypothesis $h \in H$ that separates the positive examples from the negative, then we say H shatters N points. That is, any learning problem definable by N examples can be learned with no error by a hypothesis drawn from H. The maximum number of points that can be shattered by H is called the Vapnik-Chervonekis (VC) dimension of H, is denoted as $VC(H)$, and measures the capacity of the hypothesis class H. We see that an axis-aligned rectangle can shatter four points in two dimensions. Then $VC(H)$, when H is the hypothesis class of axis-aligned rectangles in two dimensions, is four. In calculating the VC dimension, it is enough that we find four points that can be shattered; it is not necessary that we be able to shatter any four points in two dimensions. For example, four points placed on a line cannot be shattered by rectangles. However, we cannot place five points in two dimensions anywhere such that a rectangle can separate the positive and negative examples for all possible labellings. VC dimension may seem pessimistic. It tells us that using a rectangle as our hypothesis class, we can learn only data sets containing four points and not more. A learning algorithm that can be learn data sets of four points is not very useful. However, this is because the VC dimension is independent of the probability distribution from which instances are drawn. In real life, the world is smoothly changing, instances close by most of the time have the same labels, and we need not worry about all possible labellings. There are a lot of data sets containing many more data points than four that are learnable by our hypothesis class. So even hypothesis classes with small VC dimensions are applicable and are preferred over those with large VC dimensions, for example, a lookup table that has infinite VC dimension."

APPENDIX B

On Unsupervised Machine Learning Algorithms

B.1 Unsupervised Learning

In unsupervised learning algorithms, natural groupings or clusters are formed without an explicit teacher.

B.1.1 Clustering:

From unlabeled examples, $\{d_1, d_2, \ldots, d_n\}$ where $d_m \in \mathbb{R}^D$

How can we find a mapping such as $d_m \rightarrow \{c_1, \ldots, c_k\}$ where c_i is a cluster which consists of data points in the space $\mathbb{R}^D$. Therefore clustering is a form of vector quantization which allows mapping group of similar vectors to an integer.

B.1.2 K-means Algorithm

K-means algorithm is a type of algorithm for finding clusters and cluster centers in a set of unlabeled data (Hastie et al., 2003). The K refers to the number of clusters to be created in the data. Given the initial set of clusters K-means performs two steps:

- **e-step:** for each center we identify the subset of training points (its cluster) that is closer to it than any other center;
- **m-step:** the means of each feature for the data points in each cluster are computed, and this mean vector becomes the new center for that cluster.

These two steps are iterated until the convergence.

Given a set of data points $\mathbf{X} = x_1, \ldots, x_n$ in d dimensional euclidean space, k-means tries to cluster these n samples into k clusters where $k \leq n$ $\mathbf{S} = s_1 \ldots s_k$ and tries to minimize the within-cluster-sum of squares (WCSS) (Wikipedia, 2010e):

$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x_j} \in S_i} \|\mathbf{x_j} - \boldsymbol{\mu}_i\|^2$

where μ_i is the mean of points in S_i.

Algorithm B.1.1 KMeansCluster(X)

1. **global** k, n
2. **local** $\mu_1, \mu_2, \ldots, \mu_i$
3. **repeat**
4. perform **e-step** and assign each sample to a cluster:
   $S_i^{(t)} \leftarrow \{\mathbf{x}_j : \|\mathbf{x}_j - \mu_i^{(t)}\| \leq \|\mathbf{x_j} - \mu_{\mathbf{i}^*}^{(\mathbf{t})}\| \mathbf{forall} i^* = 1, \ldots, k\}$
5. perform **m-step** and calculate the the means of samples in the cluster and find the new centroid:
   $\mu_{\mathbf{i}}^{(\mathbf{t+1})} \leftarrow \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x_j} \in S_i^{(t)}} \mathbf{x_j}$
6. **until** no change in μ_i

APPENDIX C

On Ensemble Learning Algorithms

C.1 Ensemble Learning Algorithms

C.1.0.1 Bagging

Bagging (Polikar, 2009), (Zhou, 2008) trains a group learners from the bootstrap samples. Say we have a training set D with size N and then we'll create samples D_i from D with size N' where $N' \leq N$. D_i's are our bootstrap samples. Bagging combines the classifiers with weighted majority voting in which most voted prediction becomes output of the classifier. Bagging reduces the variance of the classifier. The algorithm of bagging is fairly simple and shown in Algorithm C.1.1.

Algorithm C.1.1 Bagging Algorithm

1. **Inputs:** Data set $\mathcal{D} = (\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)$;
2. Base Learner $\mathcal{L}$
3. The size of ensemble S_E
4. **Algorithm:**
5. **for** $s_E \leftarrow 1 \ldots S_E$ **do**
6. $\quad \mathcal{D}_{s_E} \leftarrow bootStrap(\mathcal{D})$
7. $\quad h_{s_E} \leftarrow \mathcal{L}(\mathcal{D}_{s_E})$
8. **end for**
9. **Test:**
10. $H(\mathbf{x}) \leftarrow argmax_{y \in \mathcal{Y}} \sum_{s_E=1}^{S_E} w_{s_E}(y_{s_E} = h_{s_E}(\mathbf{x}_{s_E}))$
11. **return** $H(\mathbf{x})$

C.1.0.2 AdaBoost

Boosting is a general technique that attempts the boost the accuracy of the base-learning algorithm. Boosting stems from the "PAC" learning model. It trains "weak-learning" algorithms and boosts them in order to get "strong-learner".

AdaBoost is a specific type of boosting technique proposed by Freund and Schapire in 1995 (Freund and Schapire, 1995). The algorithm of AdaBoost is shown in Algorithm C.1.2 which is also shown in (Polikar, 2009), (Freund et al., 1999), (Freund and Schapire, 1995).

Algorithm C.1.2 AdaBoost Algorithm

1. **Inputs:** $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)$ where $\mathbf{x_i} \in \mathcal{X}, y_i \in \mathcal{Y}$ and $\mathcal{Y} = \{y_0, \ldots, y_i\}$
2. Initialize $D_1 = \frac{1}{n}$
3. Base Learner $\mathcal{L}$
4. The size of ensemble S_E
5. **Algorithm:**
6. **for** $s_E \leftarrow 1 \ldots S_E$ **do**
7. $\quad \mathcal{D}_{s_E} \leftarrow bootStrap(\mathcal{D})$
8. $\quad h_{s_E} \leftarrow \mathcal{L}(\mathcal{D}_{s_E})$
9. $\quad$ Calculate the error ϵ_{s_E} of h_{s_E}: $\epsilon_{s_E} \leftarrow \sum_{i=1}^{n} \mathbf{I}\|y_i \neq h_{s_E}(x_i)\|\mathcal{D}_{s_E}(i) = \sum_{i:y_i \neq h_{s_E}(x_i)} D_{s_E}(i)$
10. $\quad$ Compute the normalized Error: $\alpha_{s_E} \leftarrow ln\frac{1-\epsilon_{s_E}}{\epsilon_{s_E}}$
11. $\quad$ Update Distribution with normalization factor
12. $\quad Z_{s_E}$: $\mathcal{D}_{s_E+1}(i) \leftarrow \frac{\mathcal{D}_{s_E}(i)exp(\alpha^{s_E}\mathbf{I}\|y_i \neq h_{s_E}(x_i)\|)}{Z_{s_E}}$
13. **end for**
14. **Test:**
15. $C(\mathbf{x}) = argmax_k \sum_{m=1}^{M} \alpha^n \mathbf{I}(\mathcal{L}^m(\mathbf{x}) \neq k)$

I is the indicator function, and $\mathcal{D}_{s_E}(i)$ holds the distribution of error of i^th example by the model $\mathcal{L}_{s^E}$. AdaBoost starts by training a base-learner $\mathcal{L}$ and in each iteration it computes the error ratio of the classifier than updates the Distribution and weights of examples that are misclassified by the previous learner will be higher. Therefore we'll obtain diversity between the learners.

C.1.0.3 Evolutionary Ensembles

Evolutionary Ensembles use Genetic algorithms to evolve the best classifier among the ensemble. The most important evolutionary ensemble algorithm is *EVEN* (Kim et al., 2002), (Sylvester and Chawla, 2006). Instead of using voting they employ genetic algorithms and the best decision evolves among the ensemble. EVEN uses accuracy for the fitness function. The algorithm for EVEN is as in Algorithm C.1.3 which is taken from (Sylvester and Chawla, 2006). EVEN starts before the training phase splits the data set into three disjoint sets, a training set, a validation set, and a test set to calculate the accuracy against.

Algorithm C.1.3 Evolution(G, p, C)

1. **Inputs:** Number of Generations G, Population size p, Number of classifiers C
2. Weight vector $\mathcal{W}$
3. Predictions $\mathcal{X}$
4. $P_0 = RandomPopulation(p)$
5. **for** $g \leftarrow 1 \ldots G$ **do**
6. **for** $i \leftarrow 0 \ldots p$ **do**
7. Compute $fitness_i = accuracy(P_g.i)$
8. **end for**
9. Sort P_g by $fitness$
10. $P_{g+1} = interbreed(P_g) + mutations(P_g) + survivors(P_g)$
11. **end for**
12. Select $P_{G-1}.i from P_{G-1}$ such that,
13. $fitness(P_{G-1}.i) = argmax(fitness(P_{G-1}.1), \ldots, fitness(P_{G-1}.i))$
14. **return** $\mathcal{W} = weights(P_{G-1}.i)$
15. **return** $\mathcal{Y} = predictions(P_{G-1}.i)$