UMA METODOLOGIA PARA MODELAGEM DE JOGADORES BASEADA EM APRENDIZADO DE MÁQUINA

MARLOS CHOLODOVSKIS MACHADO

UMA METODOLOGIA PARA MODELAGEM DE JOGADORES BASEADA EM APRENDIZADO DE MÁQUINA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: LUIZ CHAIMOWICZ
COORIENTADORA: GISELE LOBO PAPPA

Belo Horizonte
Fevereiro de 2013

MARLOS CHOLODOVSKIS MACHADO

A METHODOLOGY FOR PLAYER MODELING BASED ON MACHINE LEARNING

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

ADVISOR: LUIZ CHAIMOWICZ
CO-ADVISOR: GISELE LOBO PAPPA

Belo Horizonte
February 2013

Machado, Marlos Cholodovskis.
M149m Uma metodologia para modelagem de jogadores baseada em aprendizado de máquina / Marlos Cholodovskis Machado. — Belo Horizonte, 2013.
xxvii, 99 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.
Orientador: Luiz Chaimowicz
Coorientadora: Gisele Lobo Pappa.

1. Computação - Teses. 2. Inteligência artificial - Teses. 3. Jogos eletrônicos. I. Orientador. II Coorientadora. III Título.

CDU 519.6*08 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Uma metodologia para modelagem de jogadores
baseada em aprendizado de máquina

MARLOS CHOLODOVSKIS MACHADO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. LUIZ CHAIMOWICZ - Orientador
Departamento de Ciência da Computação - UFMG

PROFA. GISELE LOBO PAPPA - Coorientadora
Departamento de Ciência da Computação - UFMG

PROF. ANDRÉ MAURÍCIO CUNHA CAMPOS
Departamento de Inf. e Mat. Aplicada - UFRN

PROF. WAGNER MEIRA JÚNIOR
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 18 de fevereiro de 2013.

Dedico este trabalho aos meus pais, Marlos e Michelle. Essa é, de certa forma, também a realização de seus sonhos.

Acknowledgments

Since most of the people I am acknowledging here speak Portuguese, and not necessarily English, I am going to present my acknowledgments in Portuguese, once this text is for them. The unique exception is Pieter Spronck, who I am thankful for kindly sharing his dataset, source code and impressions with me.

Quando eu comecei o mestrado, pensava que não teria muitas pessoas para agradecer ao final. Achava que agradeceria somente aos meus pais, orientadores, irmão e noiva. Eu estava errado, sou grato a muitas outras pessoas.

Primeiro, referente à minha família, eu gostaria de agradecer aos meus pais, que sempre, incondicionalmente, me incentivaram a fazer o mestrado. Além deles, uma pessoa extremamente presente em todo esse período foi meu irmão, que me ouviu, assistiu prévias de palestras, leu pedaços de artigos, sempre disposto a dar opinião. Muito obrigado! Também sou muito grato à minha futura família: Poliana. Obrigado por ter me ouvido, apoiado, e estado presente em cada dia desses dois anos. Por ter atendido minhas ligações, após reuniões, para ouvir o que certamente não a interessava; e, principalmente, por ter acreditado em mim quando disse que casaríamos após o mestrado (já está marcado!) e por ter aceitado viver o meu sonho comigo!

Além da minha família, muitos amigos participaram, de alguma forma, desse mestrado. Agradeço ao Rabelo, por ter escutado, diariamente, relatórios sobre o andamento de tudo. Glívia, Natália, Luiz, Denise, pelas risadas durante as visitas e cafés da tarde. Vitor, pelas longas conversas no *gTalk*, sobre pesquisa, planejamentos, posts e rankings. Não posso também não mencionar meus co-autores, Bruno, Fantini e Renato, sem vocês, tudo teria sido mais difícil.

Ainda relacionado às minhas publicações, gostaria de agradecer a todos os meus revisores "secretos", que certamente melhoraram a qualidade desse trabalho. Além deles, ao Luiz, Gisele, Meira, Capes, CNPq e ao PPGCC, que viabilizaram minhas viagens nesses dois anos.

Não há como não agradecer também ao Synergia, como um todo, pelo excelente ambi-

ente de trabalho e flexibilidade de horário, o que facilitou imensamente o mestrado e permitiu que eu apresentasse meus artigos ao longo desses dois anos.

Gostaria de agradecer também a todos os professores, da graduação e mestrado que, de alguma forma, ajudaram nesse processo. Assim como todas as pessoas que foram parte dos meus experimentos, jogando CIVILIZATION IV com extrema boa vontade.

E por último, mas de forma alguma menos importantes, meus dois orientadores: Luiz e Gisele. Não há como não ser grato a toda a atenção despendida pelo Luiz (70 reuniões registradas e muitas outras "passadas na sala" sem registro!), conversas, paciência e ensinamentos. Assim como a Gisele, que "chegou depois", mas a quem dei bastante trabalho e visitei bastante. Sou realmente muito grato a vocês!

Para concluir, tenho muita dificuldade em diferenciar o processo de aplicação do doutorado com a conclusão do mestrado. Por isso, gostaria de agradecer a todos que se propuseram a me auxiliar, de alguma forma, nesse processo. Luiz Chaimowicz, Gisele Pappa, Wagner Meira, Antonio Loureiro, Nivio Ziviani, Jussara Almeida, Adriano Veloso, Marisa Mancini, Mario Nascimento e toda a Secretaria do DCC, muito obrigado.

Eu certamente sentirei saudades do DCC!

"(...) we can say that Muad'Dib learned rapidly because his first training was in how to learn. And the first lesson of all was the basic trust that he could learn. It is shocking to find how many people do not believe they can learn, and how many more believe learning to be difficult. Muad'Dib knew that every experience carries its lesson."

(Frank Herbert, *Dune*.)

Resumo

A Inteligência Artificial (IA) está recebendo cada vez mais atenção como uma característica fundamental para aumentar a imersão em jogos digitais. Entre as diversas abordagens de IA, uma que está se tornando importante é a modelagem de jogadores. A principal ideia é entender e modelar as características e comportamentos de jogadores para o desenvolvimento de uma IA melhor. A modelagem de diferentes aspectos de jogadores é possível em diferentes níveis de abstração, como ações, posições, preferências e satisfação. Essa modelagem permite que jogos customizem sua IA, dificuldade ou fases, para jogadores específicos, criando uma experiência de jogo mais interessante.

Nesse trabalho, nós discutimos vários aspectos dessa nova área. Primeiramente, uma vez que diversos trabalhos têm abordado esse problema, nós propusemos uma taxonomia para organizar esse campo, discutindo diferentes facetas desse tópico, desde decisões de implementação até o que o modelo tenta descrever. Classificamos então, na nossa taxonomia, alguns dos trabalhos mais importantes nessa área. Além da taxonomia, apresentamos também uma abordagem genérica para a modelagem de jogadores utilizando aprendizado de máquina, e instanciamos essa abordagem no problema de modelagem de preferências de jogadores no jogo CIVILIZATION IV.

A instanciação dessa abordagem passa por diversas etapas. Discutimos uma representação genérica, independentemente do que estiver sendo modelado, e a avaliamos realizando experimentos com o jogo de estratégia CIVILIZATION IV. Resultados mostram a efetividade de caracterização e modelagem dessa abordagem.

Continuando a instanciação da abordagem proposta, avaliamos a aplicabilidade de se utilizar informações de pontuações de jogadores para distinguir diferentes preferências. Para isso apresentamos uma caracterização de agentes virtuais no jogo, comparando seu comportamento com as suas preferências pré-definidas no código-fonte. Uma vez que caracterizamos esses agentes, fomos capazes de observar que diferentes preferências geram diferentes comportamentos. Usando essas informações, atacamos o problema de modelagem de preferências como uma tarefa de classificação binária, com uma abordagem de aprendizado supervisionado. Nós comparamos quatro métodos diferentes, baseados em diferentes paradigmas

(*SVM*, *AdaBoost*, *NaiveBayes* e *JRip*), avaliando-os em um conjunto de partidas jogadas por diferentes agentes virtuais. Atingimos acurácias que superam largamente o estado da arte. Concluindo nosso trabalho, utilizamos os modelos aprendidos para inferir as preferências de jogadores humanos. Utilizando alguns dos classificadores avaliados, obtivemos acurácias acima de 60% para a maioria das preferências avaliadas.

Abstract

Artificial Intelligence (AI) is gradually receiving more attention as a fundamental feature to increase the immersion in digital games. Among the several AI approaches, player modeling is becoming an important one. The main idea is to understand and model the player characteristics and behaviors in order to develop a better AI. It is possible to model player aspects in different levels of abstraction, such as actions, position, preferences, knowledge and satisfaction. This modeling allows games to customize their AI, difficulty or levels to specific players, making the game experience more interesting.

In this work, we discuss several aspects of this new field. Since several works have been tackling this problem, we proposed a taxonomy to organize the area, discussing several facets of this topic, ranging from implementation decisions up to what a model attempts to describe. We then classify, in our taxonomy, some of the most important works in this field. Besides the taxonomy, we also presented a generic approach to deal with player modeling using machine learning, and we instantiated this approach to model players' preferences in the game CIVILIZATION IV.

The instantiation of this approach has several steps. We first discuss a generic representation, regardless of what is being modeled, and evaluate it performing experiments with the strategy game CIVILIZATION IV. Results show the effectiveness of this representation in characterizing and modeling agents.

Continuing the instantiation of the proposed approach we evaluated the applicability of using game score information to distinguish different preferences. To perform this task we presented a characterization of virtual agents in the game, comparing their behavior with their stated preferences. Once we have characterized these agents, we were able to observe that different preferences generate different behaviors, measured by several game indicators. Using this information we tackled the preference modeling problem as a binary classification task, with a supervised learning approach. We compared four different methods, based on different paradigms (*SVM*, *AdaBoost*, *NaiveBayes* and *JRip*), evaluating them on a set of matches played by different virtual agents. We obtained accuracies that improved by far the state of the art. We conclude our work using the learned models to infer human players'

preferences. Using some of the evaluated classifiers we obtained accuracies over 60% for most of the inferred preferences.

List of Figures

List of Tables

List of Acronyms

Acronym	*Description*
AI	*Artificial Intelligence*
CT	*Challenge Tailoring*
DDA	*Dynamic Difficult Adjustment*
FPS	*First-Person Shooter*
ML	*Machine Learning*
MMORPG	*Massively Multiplayer Online Role-Playing Game*
NPC	*Non-Player Character*
R^2	*Coefficient of Determination*
RPG	*Role-Playing Game*
RTS	*Real-Time Strategy*
SMO	*Sequential Minimal Optimization*
SVM	*Support Vector Machines*
TBS	*Turn-Based Strategy*

Contents

Chapter 1

Introduction

> "Would you tell me, please, which way I ought to go from here?"
> "That dependes a good deal on where you want to get to" said the Cat.
> "I don't much care where -" said Alice.
> "Then it doesn't matter which way you go," said the Cat.
>
> Lewis Carroll, Alice's Adventures in Wonderland

This chapter discusses the context of this thesis, as well as the motivation to research *player modeling*. It also presents the thesis objectives, the main contributions of the work, the publications it generated, and an overview of the organization of the rest of the text.

1.1 Context and Motivation

The main goal of most games is entertainment [Nareyek, 2004]. Entertainment is a subjective concept and, in order to know how much a game entertains a player, some general metrics are used. One of the most important metrics is immersion, which is generally related to how absorbing and engaging a game is [Manovich, 2001; Taylor, 2002; Bakkes et al., 2009]. Two common approaches to achieve immersion are the use of stunning graphics and the development of a good Artificial Intelligence (AI) system. While graphics are responsible for initially "seducing" players, AI is responsible for keeping them interested in the game.

For a long time, the game industry has put much of its efforts on the graphics of its AAA games[1]. However, in recent years the focus has started to shift to AI, which has been commonly relegated to a less important role, and new techniques are now constantly being proposed. There are several reasons for this. Maybe the most important is the perception that the immersion achieved with amazing graphics can be spoiled by the behavior of *dummy* non-player characters (NPCs). An example supporting this claim is predictable behaviors that may allow the player to discover a specific opponent weakness and repeatedly explore

[1] A game that has a high budget and is expected to sell a large number of copies.

it during the game. Charles and Black [2004] affirm that "Often this means that the player finds it easier to succeed in the game but their enjoyment of the game is lessened because the challenge that they face is reduced and they are not encouraged to explore the full features of the game".

Besides a greater interest from industry regarding AI techniques, the gap between the game industry and academic AI is being tightened due to the increasing performance of the new computer architectures, which has allowed the use of more sophisticated AI algorithms. At the same time, AI researchers have been considering digital games as an important platform for research. Fairclough et al. [2001] argue that "computer games offer an accessible platform upon which serious cognitive research can be engaged", while Laird and van Lent [2000] suggest that computer games are the perfect platform to pursue research into human level AI. Moreover, the high level of realism achieved by some games has provided us an environment similar to the real world that can be used, for example, to evaluate robotics algorithms without the costs of sensors or real robots.

In this scenario, an AI approach that is gaining attention is *player modeling*, the main topic of this thesis. According to Lucas et al. [2012]:

> "Player modeling concerns the capturing of characteristic features of a game player in a model. Such features may encompass player actions, behaviors, preferences, goals, style, personality, attitudes, and motivations. Player models can be used to let the game adapt automatically to be better able to achieve its goals with respect to the player."

We firmly believe that *player modeling* is a very promising field and many works share this belief. To confirm our claim, Yannakakis [2012], when discussing the current state of game AI in academy and industry, states that *player experience modeling* is one of the "four key game AI research areas that are currently reshaping the research roadmap in the game AI field". Additionally, in 2012, a seminar was held in Dagstuhl, Germany, with the presence of several game AI experts, in order to identify their main research challenges. The report currently available [Lucas et al., 2012] states that *player modeling* is one of these challenges. It also presents a relevant discussion about the main advantages of *player modeling*:

> "(...) creating a player model as an intermediate step has at least two advantages: (1) it creates an understanding of who the player is, and therefore an argument for making specific adaptations; and (2) a player model allows generalization of adaptations to other games."

Despite receiving much more attention recently, *player modeling* has been considered a relevant topic for several years [Carmel and Markovitch, 1993; van den Herik et al., 2005; Laviers et al., 2009]. In this context, we present our research goals followed by our main contributions.

1.2 Problem Definition and Objectives

As discussed above, the term *player modeling* can be used to model several player facets. In this work, we will use *player modeling* referring to modeling player styles. This is called *preference modeling* in the nomenclature defined by Spronck and den Teuling [2010]; den Teuling [2010]. We intend to model player styles automatically, using data extracted from played games. In order to perform this task, it is important to ensure that the data extracted is relevant and allows us to distinguish different players.

Since *player modeling* techniques can be generalized in order to be applied to a set of games, and not in a specific one, we present a generic approach for it and an evaluation of a generic representation for players in different games. To automatically identify styles in the game CIVILIZATION IV we instantiate our generic approach to the problem of *preference modeling*. Finally, due to the huge attention *player modeling* has been receiving, we also organize the field, creating a taxonomy that can be used to better understand current works and ease the discussion about their approaches.

1.3 Contributions

In summary, the main contributions of this work are:

- A *taxonomy* for the *player modeling* field, more specifically:
  - We extracted from the literature several different features and proposed a taxonomy that classifies each work according to six different aspects. These aspects are: *Description, Categories, Goals, Applications, Methods* and *Implementation*;
  - We categorized several important works in the literature using our taxonomy.
- After presenting an organization for the field, we tackle the *player modeling* problem in two phases:
  - We propose a generic approach for *player modeling* as a Machine Learning problem;

  - We discuss a generic representation that can be used across different games, evaluating the possibility of its use in industry, showing that it attends most required features stated by Isla [2005].

- We then instantiate the approach proposed in our problem i.e., modeling *preferences* of CIVILIZATION IV players. In order to do it we performed several tasks:

  - Evaluated the possibility of using the *generic representation* discussed, showing that we are able to infer an agent's representation observing its behavior, in the game CIVILIZATION IV.
  - Evaluated the applicability of CIVILIZATION IV in-game indicators as features for an ML approach. In order to do this, we characterized CIVILIZATION IV agents behaviors with linear regressions:
    * Showing that different agents' preferences do cause an observable impact in several game indicators; and
    * Evaluating the impact of the match result in game indicators, verifying the importance of this information;
  - We used CIVILIZATION IV game indicators to classify, with a supervised learning approach, virtual agents' preferences. We also evaluated the use of the generated models to classify agents that were "not known" by the ML algorithm, since they were not in the training set;
  - We used the models generated for virtual agents to classify self-declared human players preferences also in CIVILIZATION IV.

In the following we enumerate the already published works as direct contributions of this dissertation:

- **Machado, M. C.**, Fantini, E. P. C., and Chaimowicz, L. *Player Modeling: Towards a Common Taxonomy*. In Proceedings of the 16th International Conference on Computer Games (CGames), pages 50-57, Louisville, United States of America, 2011.

- **Machado, M. C.**, Fantini, E. P. C., and Chaimowicz, L. *Player Modeling: What is it? How to do it?*. In X Brazilian Symposium on Computer Games and Digital Entertainment (SBGames) - Tutorials, Salvador, Brazil, 2011.

- **Machado, M. C.**, Pappa, G. L., and Chaimowicz, L. *Characterizing and Modeling Agents in Digital Games*. In Proceedings of the XI Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Brasilia, Brazil, 2012.

- **Machado, M. C.**, Rocha, B. S. L., and Chaimowicz, L. *Agents Behavior and Preferences Characterization in Civilization IV*. In Proceedings of the X Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Salvador, Brazil, 2011.

- **Machado, M. C.**, Pappa, G. L., and Chaimowicz, L. *A Binary Classification Approach for Automatic Preference Modeling of Virtual Agents in Civilization IV*. In Proceedings of the 8th International Conference on Computational Intelligence and Games (CIG), Granada, Spain, 2012.

- de Freitas Cunha, R., **Machado, M. C.**, and Chaimowicz, L. *RTSmate: Towards an Advice System for RTS Games*. In ACM Computers in Entertainment (CiE), 2013 (*in press*).

1.4 Roadmap

The remainder of this thesis is organized in seven chapters, as follows.

Chapter 2: In the second chapter we discuss required background for this thesis. We first discuss the game platform used in this work and present its characteristics and programming interfaces. Secondly, we present the machine learning methods that were used to classify players (and virtual agents) and discuss their main differences.

Chapter 3: In this chapter we present the main works related to *player modeling*. We structure the chapter by *player modeling* applications, namely: Game Design, Interactive Storytelling and Opponents Artificial Intelligence. When presenting the related work it became evident the huge amount of published papers in the field, and a lack of organization of these works. Due to that, we also present the taxonomy we proposed for *player modeling*.

Chapter 4: In this chapter we propose a generic approach for *player modeling* as a machine learning problem and present a generic representation for players. This representation can be used for several different goals.

Chapter 5: Once we defined a generic approach for obtaining player models with machine learning techniques, we instantiate it in our problem: preference modeling in the game CIVILIZATION IV. In order to do this, in this chapter we discuss the application of the generic representation proposed in Chapter 4 in the game CIVILIZATION IV, and perform

a characterization of virtual agents behaviors. This characterization was useful to define the features to be used by the ML technique.

Chapter 6: After instantiating the approach proposed in Chapter 4, we classify different agent's preferences with four different ML techniques. All methods use supervised learning and they are: Naive Bayes, JRip, AdaBoost and SVM. In this chapter we evaluate the performance of these techniques to model virtual agents and (human) players' preferences in the game CIVILIZATION IV.

Chapter 7: Finally, in this last chapter we present our conclusions, summarize our results and discuss some future directions.

Chapter 2

Background

> If knowledge can create problems, it is not through ignorance that we can solve them.
>
> Isaac Asimov

This chapter presents background knowledge for the rest of this thesis. We discuss two main topics used in this work: (1) the game platform, its characteristics and programming interfaces; and (2) the main concepts of machine learning, the classifiers used in the experiments and the experimental methodology of our tests.

2.1 Civilization IV

In this thesis, we used CIVILIZATION IV as a game platform to perform our experiments. The platform selection is a very important step when researching digital games because implementation is generally restrained by the game interface. Additionally, it is also important to ensure that the selected game presents the basic characteristics required for the proposed research. We discuss all these topics in sequence. A deeper discussion about several different game platforms and its possibilities is presented in [Machado et al., 2011a].

CIVILIZATION IV is a turn-based strategy game (TBS)[1] released in 2005, developed by the studio *Firaxis Games*. In this game, each player is represented by a leader who controls an empire. Players/Empires compete with each other to reach one of the many game victory conditions.

A high-level description of this game is nicely presented by den Teuling [2010]: "In CIVILIZATION IV a player begins with selecting an empire and an appropriate leader. There are eighteen different empires available and a total of 26 leaders. Once the empire and leader have been selected, the game starts in the year 4000 BC. From here on, the player has to

[1] A turn-based game is a game where each player plays his/her turn while the others wait for his/her move. Its dynamic is very similar to board games, and it differs from real-time games because no actions are taken in parallel.

Figure 2.1: *Screenshot of* CIVILIZATION IV.

compete with rival leaders, manage cities, develop infrastructure, encourage scientific and cultural progress, found religions, etcetera. An original characteristic of CIVILIZATION IV, is that defeating the opponent is not the only way to be victorious. There are six conditions to be victorious as mentioned in [2K Games, 2005]: (1) *Time Victory*, (2) *Conquest Victory*, (3) *Domination Victory*, (4) *Cultural Victory*, (5) *Space Race* and (6) *Diplomatic Victory*. Because of these six different victory conditions the relation between the player and the opponent is different from most strategy games. The main part of the game the player is at peace with his opponents. Therefore it is possible to interact, to negotiate, to trade, to threaten and to make deals with opponents. Only after declaring war or being declared war upon, a player is at war. Any player can declare war any time, unless that player is in an agreement with an opponent which specifically forbids war declaration." An in-game screenshot is presented in Figure 2.1.

These six different victory possibilities make this game very interesting to this research, being one of the main reasons for selecting this platform. The game allows completely different behaviors to succeed, unlike other games in which the unique way to win is to defeat your opponents by attacking them.

In order to encompass different behaviors in its AI, each agent is characterized by a

set of weighted preferences. The preferences are represented by attributes that define the way an agent plays, and are: (1) *Culture*, (2) *Gold*, (3) *Growth*, (4) *Military*, (5) *Religion* and (6) *Science*. The assigned weights represent a "weak" (value 2) or "strong" preference (value 5), besides no preference at all (value 0). Each behavior allows the agent to seek one of the six victory conditions. The main focus of this thesis is to be able to automatically identify suitable weights that represent an observed behavior, both for virtual agents and human players.

2.1.0.1 Programming Interface

In order to access game data and edit agents behaviors, CIVILIZATION IV offers two different possibilities: (1) to edit game resources, such as XMLs; or (2) to edit the game source code (or attach scripts to it).

The XML interface offers the possibility of configuring several game parameters, such as the agents "flavors" (the name they gave to the agents preferences listed above). This XML files set each agent's preferences, allowing us to edit them. This explicit representation is another reason we selected *Civilization IV* as a testbed platform, allowing us to check each agent preference.

Editing game source code is another possibility. Its interface is an SDK that allows people to change the source code of the game and compile it, generating a DLL that replaces the traditional one.

To model agents preferences we need to use indirect observations to infer them. We attached an script to the source code to retrieve game score indicators that we use as evidences for different behaviors generated from different preferences. These indicators are constantly available to all players, and we decided to use them instead of directly evaluating actions because they represent a generalization of actions.

The script we used is called *AiAutoPlay* and it is easily found on the Web. We used a modification of it that was used to generate the dataset presented in [den Teuling, 2010; Spronck and den Teuling, 2010]. We discuss the generated dataset in Chapter 5.

2.2 Machine Learning

Machine Learning (ML) is a common approach for preference modeling, since we want to "learn" a model from a set of available players (examples), and then use this model to classify new players. There are three main approaches of learning: supervised, semi-supervised and unsupervised learning [Alpaydin, 2010].

While in supervised learning a complete set of labeled data is available (we know beforehand the users preferences) in unsupervised learning no classes are known. Taking as an example the task of preference learning, in both supervised and unsupervised learning the data we learn from is a set of matches already played, together with the players preferences. However, in the case of supervised learning, these preferences were previously labeled by an expert, while in unsupervised learning the algorithm learns using distance measures between data examples. Semi-supervised learning, in turn, uses both labeled and unlabeled data during the training process.

Here we model virtual agents' preferences in the game *Civilization IV* with different supervised learning techniques. Each applied technique uses a different paradigm. We discuss the main characteristics of each algorithm used in this thesis in the next sections. For a deeper explanation see [Alpaydin, 2010].

Particularly, we discuss the four classifiers applied to solve our problem: *SVM*, *Naive Bayes*, *JRip* and *AdaBoost*. Each of them produces a different type of model, which might explore different characteristics of the data.

2.2.1 Support Vector Machine

Support Vector Machines (SVM) model classification as an optimization problem, and are considered the state of the art in classification for many different domains. Each training/test instance is modeled as a vector, where each feature represents a different dimension. SVM tries to separate the space into two different subspaces with an hyperplane. It assumes that the data can be linearly separable or that there is a kernel function able to transform the space to achieve this goal.

This separation is done with an hyperplane using a margin[2]. This margin allows noise shifts to exist without changing the class classification, since it has "breathing space". Each class is in one side of the margins.

The SVM problem is an optimization problem that seeks for an *optimal separating hyperplane*, *i.e.* maximizing the margin. This problem can be solved using quadratic optimization methods [Alpaydin, 2010]. We used the LIBSVM [Chang and Lin, 2011] implementation in our experiments.

SVM has two parameters that are very important and directly influence its performance: *cost (c)* and *gamma (g)*. The cost parameter is responsible for evaluating the cost of a misclassification in the training examples. A low cost may imply in a simpler surface, which may misclassify some training examples but avoid overfitting, *i.e.* a situation where the algorithm is adjusted to very specific features of the training data and does not generalize

[2] "the distance from the hyperplane to the instances closest to it on either side" [Alpaydin, 2010].

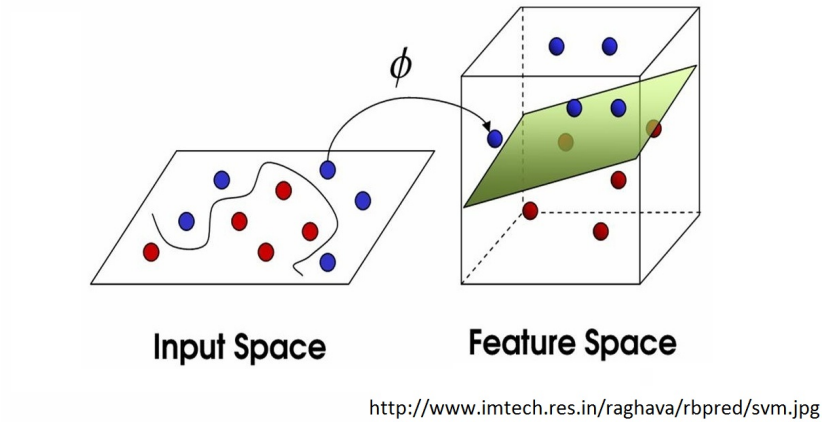


Figure 2.2: *SVM main concepts.*

for additional data. On the other hand, a high value for this parameter may generate very specific surfaces, able to correctly classify all training examples but with limited generalization capability. The gamma parameter defines the influence of a support vector upon its surroundings. A low gamma means a higher influence, leading to a small number of support vectors, while a high gamma transforms each training vector in a support vector, since each vector has a small influence in the whole space.

Figure 2.2 shows the basic *SVM* concepts. Once we have an input space, we apply a kernel function ϕ to map the input space into a feature space, where classification will be held. Classification is then performed finding an hyperplane that maximizes the distance between the instances (blue and red spheres) and the support vectors, that are derived from the hyperplane.

2.2.2 Naive Bayes

Naive Bayes is a probabilistic classifier that assumes that all features (inputs) are independent, generating a classifier that makes its prediction evaluating the probability of each class given an input. Although this assumption is unrealistic, *Naive Bayes* performs well in a wide range of domains, apart from being fast. It can be represented by the following equation:

$$P(\chi|C) = \prod_{j=1}^{d} p(\chi_j|C),$$

where χ is the input and C a multinomial variable taking a class code, as defined by Alpaydin [2010]. Once the algorithm assumes a conditional independence, we can calculate the conditional distribution over the class $\mathbf{C}$ as the product of $p(\chi_j|C)$ for all j, then easily discovering the probability of being a specific class given the features input (χ_j).

2.2.3 JRip

JRip is a Java implementation of RIPPER [Cohen, 1995], and follows a divide-and-conquer strategy that divides the input space into different regions, and finds rules for these regions.

In this approach, rules with IF-THEN statements are learned, one at a time. The algorithm successively executes two different phases: (1) grow and (2) prune. Alpaydin [2010] states that "we start with the case of two classes where we talk of positive and negative examples (...) Rules are added to explain positive examples such that if an instance is not covered by any rule, then it is classified as negative. So a rule when it matches is either correct (true positive), or it causes a false positive. (...) Once a rule is grown, it is pruned back by deleting conditions in reverse order, to find the rule that maximizes" a metric called rule value metric, calculated using the number of true and false positives.

We have used the algorithm implemented in the Weka framework called JRip[3] [Cohen, 1995]. It is important to stress that this algorithm is very interesting because it generates comprehensible knowledge. While the other algorithms generate mathematical models that can be hard to be understood, here the generated rules are quite easy to be understood.

2.2.4 AdaBoost

This algorithm is based on the idea of combining multiple learners[4] that complement each other in order to generate a classifier with higher accuracy.

Using χ_i to represent an arbitrary dimensional input and d_i the prediction of a base learner, Alpaydin [2010] defines a Boosting algorithm, presenting an example, as follows: "Given a large training set, we randomly divide it into three. We use χ_1 and train d_1. We then take χ_2 and feed it to d_1. We take all instances misclassified by d_1 and also as many instances on which d_1 is correct from χ_2, and these together form the training set of d_2. We then take χ_3 and feed it to d_1 and d_2. The instances on which d_1 and d_2 disagree form the training set of d_3. During testing, given an instance, we give it to d_1 and d_2; if they agree, that is the response, otherwise the response of d_3 is taken as the output."

We use the *AdaBoost* algorithm [Freund and Schapire, 1996], an abbreviation for adaptive boosting. It differs from basic Boosting algorithms because it does not require a large training set to work properly, as it does not divide the dataset in disjoint sets. It uses the same dataset successively, giving different weights to each instance as it is misclassified or not.

[3] `http://wiki.pentaho.com/display/DATAMINING/JRip`

[4] In fact, these classifiers are *weak learners*, *i.e.* simple classifiers with an accuracy higher than $\frac{1}{2}$. This means that, for a binary classification problem, they are still better than a random algorithm.

2.2.5 Summary

This section discussed the main techniques used in this thesis. Its main goal was not to deeply discuss each technique, but to show that each algorithm is based in a different paradigm expecting a different characteristic from the dataset.

Our main concern was to highlight these differences between algorithms and show that our choice was not arbitrary. In summary, *Naive Bayes* assumes that the different features that represent a player are independent, with no feature depending on the other. Another simple approach is *JRip*, which requires each class to be distinguished by a set of simple rules. On the other hand, more complex approaches are *SVM* and *AdaBoost*. *SVM* has the premise that different classes can be separated in the space using a kernel function (or linearly), while *AdaBoost* combine different weak classifiers, focused on subparts of the input space, in order to generate a classifier with higher accuracy.

Since each algorithm has a premise that directly impact its performance, we evaluate all of them in our problem. Our results are presented, and further discussed, in Chapter 6.

Chapter 3

Related Work and Player Modeling Taxonomy

You raise up your head
And you ask, "Is this where it is?"
And somebody points to you and says "It's his"
And you say, "What's mine?"
And somebody else says, "Where what is?"
And you say, "Oh my God
Am I here all alone?"

Bob Dylan, *Ballad of a Thin Man*

As previously stated in Chapter 1, *player modeling* is currently a very relevant topic in game AI research. Due to the attention it is receiving, several papers related to this topic are constantly being published. However, *player modeling* is a loose concept and, until recently, the field was not completely structured. Papers used different terms for the same things, generating a lack of precise terminology. In this chapter we present some of the most relevant works in the field and propose a taxonomy to organize them, classifying the works discussed here in several facets.

3.1 Related Work

Slagle and Dixon [1970] were the first to present an attempt to model players, but research specifically focused on *player modeling* started in 1993, first aiming at improving search in game trees [Carmel and Markovitch, 1993]. At that time, the processing power available in computers was much smaller than today and, due to this limitation, the authors proposed modeling players as an alternative to better prune game trees in CHESS. Another work, in that same year, which also studied the potential of player models in tree search is [Iida et al., 1993].

From this point, for almost one decade, the few studies in this area were applied to classical games such as CHESS [Carmel and Markovitch, 1993], GO [Ramon et al., 2002], ROSHAMBO[1] [Billings, 2000; Egnor, 2000], the iterated prisoner's dilemma [Kendall, 2005] and POKER [Billings et al., 1998; Davidson et al., 2000]. This scenario started to change with Houlette [2003], who discussed the applicability of player modeling in more complex games, such as those in the FPS genre, and suggested a model to represent these players. Houlette [2003] described this representation as "a collection of numeric attributes, or *traits*, that describe the playing style of an individual player".

After Houlette's work, several other researchers started focusing on non-traditional games, such as FPS, RPG and Strategy games. With a broader scope, *player modeling* started being used for different goals. We now discuss some recent works dealing with different applications where *player modeling* can be used, including the generation of artificial intelligent opponents, game design and interactive storytelling.

3.1.1 Artificial Intelligent Opponents

The great majority of research related to *player modeling* is in this topic, and the first works of the field, discussed at the beginning of this section [Carmel and Markovitch, 1993; Iida et al., 1993; Ramon et al., 2002], can also be classified here. It is also sometimes called *Opponent Modeling*.

A first research branch in *player modeling* is related to games where AI is generally implemented using tree search algorithms such as *MiniMax*. Games in this category are board and card games. Some of the most recent works focus on the game of POKER. Billings [2006]; Aiolli and Palazzi [2008] used a set of weights for each TEXAS HOLD'EM POKER possible player type and predicted one's choice of action as a weighted voting by all player types. On the other hand, Ponsen et al. [2010] created a poker player with *Monte-Carlo Tree Search* algorithms and used *player modeling* to focus on relevant parts of the game tree. Another work, applied to HEADS UP POKER NO LIMIT, is [Southey et al., 2005], in which the authors "present a Bayesian probabilistic model for a broad class of poker games, separating the uncertainty in the game dynamics from the uncertainty of the opponent's strategy".

Besides *player modeling* in tree search algorithms, a second research branch is related to games where tree search algorithms are not able to generate a satisfactory AI. This may be due to several reasons, such as the difficulty of modeling, in a good granularity, different game states and its transitions; or the size of the generated tree. For example, Aha et al. [2005] states that modern strategy games have a branching factor of approximately 1.5×10^3,

[1]The game ROSHAMBO is also known as Rock-Paper-Scissors

while CHESS has a branching factor close to 30. Besides strategy games, other game genres where AI is not generated by search in the state space are action, adventure, RPG and sports.

This second research branch is very generic, encompassing several different works and approaches. In order to better organize this section we have clustered works focusing on what characteristics of the opponent they aim at modeling. We discuss four possibilities here: (1) actions, (2) tactics/preferences, (3) movement/position and, (4) knowledge.

Most of the research in *player modeling* is done with *action models*, which are an attempt to model players' activities in a way that makes it possible to predict the next player's action [Spronck and den Teuling, 2010] . Indeed, many works follow this line and [Rohs, 2007] is a classical example, as they try to predict if an agent will declare war against other in the game CIVILIZATION IV. Another important work is [Laird, 2001] that anticipated actions in QUAKE II.

Player's tactics/preferences are similar to the modeling of player's actions but, while the first models directly observable actions, this second modeling focuses on a goal, that will be reached by a set of small actions. This thesis is mainly focused on this task.

Three works that modeled player's tactics/strategies are [van der Heijden et al., 2008; Laviers et al., 2009; Weber and Mateas, 2009]. In these works, the authors automatically inferred characters goals by observing their actions. Laviers et al. [2009] used *Support Vector Machines (SVM)* to recognize a defensive play "as quickly as possible in order to maximize (...) team's ability to intelligently respond with the best offense". van der Heijden et al. [2008], on the other hand, presented a method that obtains dynamic formations capable of adapting to the formation of the opponent player. This adaptation is performed by a learning algorithm, while the classification of the opponent is done with a set of steps that determine the likelihood of each opponent exhibiting the observed situation. Finally, Weber and Mateas [2009] applied multi-class classification techniques in STARCRAFT game replays to predict players' strategies. Their approach showed to be less susceptible to noise and imperfect information.

A research very related to ours is [den Teuling, 2010] and the paper generated from his master's thesis [Spronck and den Teuling, 2010]. In this research, the authors modeled strategy preferences of virtual agents in the game CIVILIZATION IV, through a supervised learning technique called *Sequential Minimal Optimization (SMO)* as a multi-class classification problem. After learning virtual agents preferences, they tried to identify virtual agents not present in the training and also human preferences. They did not have much success on this last task. The modeled preferences were: *Culture, Gold, Growth, Military, Religion* and *Science*. This is a very important work and we will further discuss it along this thesis. We use their dataset in our research and we use their work as baseline in some parts of this thesis.

Modeling players position/movement is also a common approach in the literature. This

approach may be seen as an attempt to present smart NPCs who do not break game rules (ignoring *fog of war*[2], for example), a common used resource, as Laird and van Lent [2000] already discussed. Two recent works in this topic are [Weber et al., 2011; Tastan et al., 2012]. Weber et al. [2011] modeled players movement/position, in the game STARCRAFT, using a particle based approach, while [Tastan et al., 2012] used "inverse reinforcement learning to learn a player-specific motion model from sets of example traces". Valkenberg [2007] also worked in this problem trying to foresee players position in the game WORLD OF WARCRAFT. Despite the fact that he did not have much success, the problem he worked is an excellent example of the discussed topics. A more successful approach was presented in [Hladky and Bulitko, 2008] for the game COUNTER STRIKE: SOURCE. Other works that also predict the position of opponent players in FPS games are [Laird, 2001; C. J. Darken, 2008].

To conclude, it is also possible to model players' knowledge. A knowledge modeling was proposed by de Freitas Cunha et al. [2013], although the authors did not use this terminology. In this work, we developed an aid system to RTS players and one of its activities is to predict the technological level of an agent based on its units in the game WARGUS. We implemented a reverse path in the dependency tree of the game, deriving what are the technologies known by the enemy once the player has seen a building or unit of his/her adversary.

Several other works were also successful in modeling players in order to improve their AI. We have presented some of the most recent and important papers in the field, and discussed common applications of this technique. Nevertheless, there is a large number of works that we did not cover, such as [Lockett et al., 2007; Schadd et al., 2007; Aiolli and Palazzi, 2009] among others.

3.1.2 Game Design

The main idea of *player modeling* related to *Game Design* is to generate environments that are best suited to each player. This is one of the possibilities of game customization. Once one obtains a model of a player it may generate levels that maximize player's entertainment.

There are several possibilities for maximizing one's entertainment, such as identifying player's gameplay preferences in order to adapt the scenario for his/her style. For instance, once the game notes that a player likes to be a sniper in FPS games, it may generate spots for him/her to stay. The importance of including player models to procedural content generation has been discussed in [Togelius et al., 2011].

[2]Parts of the game world where the player has no units visible to him/her, generating an environment with imperfect information. It is said that these invisible parts are hidden by a fog of war.

An important work in the field of game design is [Drachen et al., 2009]. In this paper, the authors use tools to extract gameplay information from the game TOMB RAIDER: UNDERWORLD and feed neural networks (*emergent self-organizing maps*) with these data in order to obtain playing styles (*Pacifist, Runner, Veteran* and *Solver*). During this process, the authors extract several information to perform classification, e.g., *cause of death, completion time* and *number of deaths*. Using similar features, Mahlmann et al. [2010] predicted when players would stop playing the game or how long they would take to complete it.

Recently, two other methods that cluster gameplay data in order to find players stereotypes in different games were proposed by [Gow et al., 2012; Drachen et al., 2012]. Gow et al. [2012] apply Linear Discriminant Analysis (LDA) in data from two different games: SNAKETRON and ROGUE TROOPER, detecting different gameplay patterns and abilities. They also cluster data from the game ROGUE TROOPER finding four player's stereotypes: *hiperactive, normal, timid* and *naive*. Drachen et al. [2012], on the other hand, uses k-Means and Simplex Volume Maximization to cluster players both from the MMORPG TERA and the FPS game BATTLEFIELD 2: BAD COMPANY 2.

All these papers discussed above, despite of not directly changing the game scenario, are very useful for game designers since they give cues about playing patterns and preferences, which could change game design approaches.

Another work that studies game design improvement based on adaptive analysis is [Pedersen et al., 2010]. Using the game INFINITE MARIO BROS as a test platform, the authors look for correlations between players emotions (*Fun, Challenge, Frustration, Predictability, Anxiety* and *Boredom*) and level characteristics, e.g. *presence of gaps, blocks* and *enemies*. These correlations were evaluated with questionnaires.

There are several other papers in this game design context, such as [Dormans and Bakkes, 2011], which discusses the use of generative grammars to create levels, and [Yannakakis and Togelius, 2011], which presents a framework for procedural content generation driven by computational models of user experience.

Another goal of obtaining an opponent model may be challenge tailoring (CT) or dynamic difficult adjustment (DDA). Zook and Riedl [2012] distinguish these two problems: "CT is similar to *Dynamic Difficulty Adjustment (DDA)*, which only applies to online, real-time changes to game mechanics to balance difficulty. In contrast, CT generalizes DDA to both online and offline optimization of game content and is not limited to adapting game difficulty". In their work, Zook and Riedl [2012] employed tensor factorization techniques for modeling player performance in an action RPG developed by them, also discussing some approaches to tackle the challenge tailoring problem. A work related to dynamic difficult adjustment is [Missura and Gärtner, 2009]. In this paper the authors automatically adjust the difficulty of a game implemented by them by clustering players into different types.

3.1.3 Interactive Storytelling

Besides customizing the game scenario or difficulty, another possibility is to customize the dramatic storyline of a game. This is called interactive storytelling. Another short definition of interactive storytelling was given by Thue et al. [2007]:

> "(...) a story-based experience in which the sequence of events that unfolds is determined while the player plays." [Thue et al., 2007].

This application explicitly demands adaptive behavior since it is interactive. In spite of that it does not necessarily implements a customization for player preferences, although Sharma et al. [2007] showed that "*player modeling* is a key factor for the success of the Drama Management based approaches in interactive games."

One of the first works to use information of specific players to generate a customized story is [Thue et al., 2007]. In this work, the authors propose PaSSAGE, implemented in the game NEVERWINTER NIGHTS. The method models interactive storytelling as a decision process that is influenced by different weights that characterize each player. The authors define their own method as "an interactive storytelling system that uses player modeling to automatically learn a model of the player's preferred style of play, and then uses that model to dynamically select the content of an interactive story".

Another important work regarding interactive storytelling and *player modeling* is [Sharma et al., 2007]. In this work the authors validate the important assumption that "if the current player's actions follow a pattern that closely resembles the playing patterns of previous players, then their interestingness rating for stories would also closely match". Additionally, they present features to differ player types, e.g. "the average time taken by the player to perform an action in the game". They also investigate important features that "can be extracted from the player trace to improve the performance of the player preference modeling".

In the same year, Roberts et al. [2007] also developed a work related to interactive storytelling. The authors used player models to obtain a feature distribution regarding story characteristics. They calculated the importance of each feature for a player and learned a policy to select branches that custom storytelling for each player model.

A recent work regarding storytelling and some kind of player modeling was done by Cardona-Rivera and Young [2012]. The authors discussed the importance of differing between important and unimportant events when modeling a player's story comprehension.

Despite the applicability of *player modeling* in interactive storytelling, there are not many works that have done that. This claim was done by Thue et al. [2007] and it seems to

be still valid.

3.2 Player Modeling Taxonomy

As we already stated at the beginning of this chapter, *player modeling* is currently a hot research topic, with several works being published in this field. Despite that, the field is not completely structured, since no organization is used to name the possible different approaches for this problem. Hence, our first contribution in this thesis is the proposal of a taxonomy, where we discuss research approaches and goals when modeling players. We first presented this taxonomy in [Machado et al., 2011a].

In this section, we first discuss some works related to our taxonomy, i.e. other proposals aiming at organizing the field. We then present our taxonomy, its classes and possible values. To help the reader to distinguish between each class, before its description we present a question that can be seen as a guideline to classify each work. Finally, Table 3.2 shows some of the most relevant works in the field classified in our taxonomy.

3.2.1 Taxonomy's Related Work

A first more general work that presented a rough division organizing the field was [Sharma et al., 2007]. In this paper the authors divided *player modeling* simply by its measurements approach (direct or indirect). They state that direct measurements may use, for example, biometric data, while indirect measurements use in-game data (or *game metrics*) gathered from observation. We focus this thesis only on this second category, *i.e.* our taxonomy classifies works that use indirect measurements approaches.

Recently, at the same time we developed our *player modeling* taxonomy, another research group, at University of California, Santa Cruz, independently developed theirs [Smith et al., 2011a,b]. The authors divided the area in four main categories, which they called *facets*, namely *Domain, Purpose, Scope* and *Source*: "The *Domain* facet of a model answers the question of what it is that the model generates or describes" while "The *Purpose* of a model describes the function of a model in its intended application". "The *Scope* of a model describes to whom the model is intended to be relevant or who is being distinguished in the model" and "Finally, the *Source* facet describes how a player model is motivated or derived" [Smith et al., 2011a].

Finally, [Bakkes et al., 2012] also reviews and organizes the works in the literature. The authors survey the field, classifying each work in one of four categories: *Player Action Modeling*, *Player Tactics Modeling*, *Player Strategies Modeling* and *Player Profiling*.

Table 3.1: *Player Modeling Taxonomy Summary.*

Description	Time Frame	Goals	Applications	Methods	Implementation
Knowledge	Online Tracking	Collaboration	Speculation in Search	Action Modeling	Explicit
Position	Online Strategy Recognition	Adversarial	Tutoring	Preference Modeling	Implicit
Strategy	Off-line Review	Storytelling	Training	Position Modeling	
Satisfaction			Substitution	Knowledge Modeling	
			Game Design		

The unique concept present in [Bakkes et al., 2012] that was not already discussed here is *Player Profiling*. Bakkes et al. [2012] defines it as the attempt "to establish automatically psychologically or sociologically verified player profiles. Such models provide motives or explanations for observed behaviour, regardless whether it concerns strategic behaviour, tactival behaviour, or actions." To clarify this topic, we present a contrast between *player modeling* and *player profiling* presented by van Lankveld et al. [2010]:

> "*Player modeling* is a technique used to learn a player's tendencies through automatic observation in games [Thue et al., 2007] (...) *Player profiling* is the automated approach to personality profiling (...) In player profiling we look for correlations between the player's in game behavior and his scores on a personality test." [van Lankveld et al., 2010]

Since *player profiling* is not the research subject in this work, we are not going to further discuss it. Some of the main researches in this field are [Bohil and Biocca, 2007; Yannakakis and Hallam, 2009; Yannakakis et al., 2009; Yannakakis and Togelius, 2011; van Lankveld et al., 2011; Spronck et al., 2012].

3.2.2 Proposed Taxonomy

Our taxonomy is composed of six different classes that encompass different aspects of a *player modeling* research, ranging from high level concepts, such as what must be modeled, to implementation details. We discuss each of the six classes in the next sections, and we summarize its main components and possible values in Table 3.1.

Note that we present several facets of a *player modeling* work but, some of them have been already discussed by other researchers in different scenarios. Our main contribution was to select and organize different classifications in order to obtain a common taxonomy.

3.2.2.1 Description

What do you want to describe with your model?

Player modeling can be defined as an abstract description of the current state of a player at a moment. This description can be done in several ways like *satisfaction*, *knowledge*, *position* and *strategy* [van den Herik et al., 2005].

The main goal of a game is to entertain its players, which are different from each other and may not enjoy the same challenges or possibilities of the game. When their *satisfaction* is modeled, we may be able to adapt the gameplay to each player. This is called *satisfaction modeling*.

More related to agents' artificial intelligence, we may want to model the player *knowledge*, since this can be useful in several environments with imperfect information. A concrete example is found in games that have *fog of war*, in which answers to the following questions can be very useful: which part of the map the player knows? He/She knows our position? In a game with constant evolution, which evolution level has already been achieved? All these questions may be answered with *knowledge modeling*.

Similarly, we can model the player's movement. Once we are in a partially observable environment, the *position* of other players is generally an important information since it can guide your strategy or actions. This is generally called *position modeling*.

Finally, a higher-level modeling is the *strategy modeling*, which intends to interpret the player actions and relate them with game goals, *i.e.*, we abstract low-level actions seeking a high-level goal. For example, if we want to know our adversary aggressiveness, we will not be concerned with particular actions but with its strategy along the game.

3.2.2.2 Time Frame

When are you going to process the data? Do you have enough time for doing this online?

The different descriptions presented in the previous section lead us to different levels or categories of *player modeling* use, and different moments to process the data. The lowest level of abstraction, with constant processing, is the *Online Tracking*, which is concerned in predicting immediate future actions.

A higher level (*Online Strategy Recognition*) is related to strategy recognition as it involves the identification of a set of actions as a higher level objective or strategy. Finally, the *Off-line Review* is the evaluation of a game log, after its finish. This last level is what many professional players do when they are "studying" their adversaries for a game. Laviers et al. [2009] discuss these three topics and argues that *Online Tracking* is used for single players while *Online Strategy Recognition* can be applied to entire teams. This is true since there is not a definition for a unique team's action. Nevertheless, it is important to note that we can also recognize strategies for individual players.

3.2.2.3 Goals

What are you intending to generate with Player Modeling? What is your main goal?

As previously mentioned, we firmly believe *Player Modeling* is a suitable approach to improve the AI of NPCs in games. This improvement can be seen in different types of goals for the NPCs, that can be divided into three main sets: (1) to collaborate with the human players, (2) to be their opponents, or (3) to be neutral to them, but part of the story.

The first set, related to the collaborative agents, is very difficult because human players have expectations when being aided by NPCs. These expectations are related to their actions: frequently, human players are not able to act properly because the NPCs will not behave accordingly as an unique team. Most of the games implement agents coordination and collaboration through basic orders such as "Attack", "Patrol" and "Hide". The main challenge would be to make these agents act autonomously according to the player behavior, without the need of specific orders.

The second set, the modeling of opponents, has motivation even on literature from centuries ago, as the famous Sun Tzu's quote.

> "Know your enemy and know yourself and you can fight a thousand battles without disaster." *Sun Tzu, The Art of War.*

In addition to the advices of an ancient general, it is very common for human players to study their adversaries before a match. Kasparov, the great chess player, is an example [Carmel and Markovitch, 1993]. Modeling opponents is a fundamental aspect in making games more immersive and challenging, so this is where most of the works in player modeling focuses.

One last possible goal developers can be concerned with is storytelling. In complex games, not all agents are allies or enemies. They can be neutral to players, being part of the scenario and interacting with them to help advancing the plot. Many times this interaction is offered to improve the storytelling of a game (in a medieval world not all people are warriors or mages, there are common people that should make the story more immersive) since once we model the player we may adjust all neutral agents to act properly to him.

3.2.2.4 Applications

What gameplay activity are you going to improve with your model?

In a lower abstraction level we can list, as van den Herik et al. [2005] discussed, four main applications to player modeling: *speculation in heuristic search*, *tutoring and training*, *non-player characters* and *multi-person games*. We renamed and redistributed them in terms of importance and generality. We renamed *speculation in heuristic search* to *speculation in search* and we split *tutoring and training* as two different applications. Finally, we grouped *non-player characters* and *multi-person games* as *substitution* application. We added a fifth application that is *game design*. Several other applications can be listed but we believe that these five cover a satisfactory spectrum of them.

Speculation in search is generally applied to games in which Artificial Intelligence is more related to search in game trees, generally for adversarial goals. Depending on the game complexity, it may be infeasible to check every possibility and even pruning techniques such $\alpha - \beta$ are not sufficient. In these scenarios, we may use player modeling to create a bias that helps the search heuristics.

The collaborative goal can be expressed as the use of *player modeling* to assist players. This can be be done with tutoring, when a non-human agent teaches the player (the *player modeling* is important because this tutoring process can focus on the player preferences) or training, with the presentation of challenges suited to the player characteristics (weakness, style or strategy, for example). Its behavior is important because, if the NPCs do not act properly, the game will no longer be interesting to the player [Scott, 2002].

Another main application to player modeling is in multi-player games. The objective is to allow NPCs to substitute human-players in multi-player games, even mimicking their behavior. It does not matter if the NPCs will be allies or enemies, they must be able to replace the player to keep the previous game balance. Most of the games does not have this approach and its gameplay may be impaired by players that are not able to play the whole game.

Finally, game design is applicable when the game does not want to generate NPCs behaviors, but to change its level or plot, for example.

3.2.2.5 Methods

What models are you intending to use to generate an understanding about the agent?

We can also divide the player modeling field in more specific methods, which are closely related to its description purpose. Spronck and den Teuling [2010] mentioned that most of the research in player modeling is done with *action models*, that are an attempt to model players' activities in a way that makes possible to predict the next player's action (*Online Tracking*). Works that use a set of actions (not concerned in predicting the next

atomic action) are also included here.

Spronck and den Teuling [2010] define *preference modeling* as the modeling of the "player desires to accomplish or experience in the game, and to what extent he is able to do that". This is a very precise definition and, in fact, is concerned to the player's satisfaction.

The last two listed methods are *positioning* and *knowledge modeling*. The first one attempts to obtain relevant information about players location while the second tries to model the player knowledge itself, it means, what he already knows.

Positioning modeling can be better defined as the attempt to predict NPCs positions on games with imperfect information (*fog of war*, for instance). This is a valuable information because, in general, the knowledge of a player position gives a tactical advantage in a game, as previously discussed here. This approach may be seen as an attempt to present smart NPCs who do not break game rules (ignoring *fog of war* for example), a common used resource as Laird and van Lent [2000] already discussed.

Knowledge modeling, on the other hand, tries to "predict" the other players knowledge, humans or not.

It is important to stress the difference between the *Description* and the *Methods* classes, since they have similar possible values. The main difference is that description will use one of the methods to describe a player, e.g. one may model player's actions in order to describe its knowledge; or one may model players' knowledge through questionnaires, for example, aiming at describing players' satisfaction.

3.2.2.6 Implementation

What is the interface between your algorithms and the game in which your model is going to be used?

Once we defined some of the *Player Modeling* subsets related to goals, applications, research areas, among others, we may finish this section with the lower abstraction level of discussion: the implementation. Two approaches can be highlighted: *explicit* and *implicit*.

Spronck [2005] says that "An opponent *[player]* model is explicit in game AI when a specification of the opponent's *[player's]* attributes exists separately from the decision-making process". Thus, an explicit player model is separated from the main source code and it is generally implemented through scripts or XML files. On the other hand, in *implicit* approaches, the attributes are generally embedded and diluted in different parts of the code, which makes the task of identifying and describing these attributes more difficult.

Table 3.2: *Classification, in our taxonomy, of some works in the field.*

Work	Description	Time Frame	Goals	Application	Methods	Implem.
[Bard and Bowling, 2007]	Strategy	Online Strat. Rec.	Adversarial	Spec. in Search	Action Model.	Implicit
[Drachen et al., 2009]	Strategy	Off-line Review	Storyt./Adv.	Game Design	Action Model.	Implicit
[Hladky and Bulitko, 2008]	Position	Online Strat. Rec.	Collab./Adv.	Substitution	Position Model.	Implicit
[Laviers et al., 2009]	Strategy	Online Strat. Rec.	Adversarial	Substitution	Action Model.	Implicit
[Machado et al., 2012a]	Strategy	Off-line Review	Collab./Adv.	Substitution	Preference Model.	Explicit
[Martinez et al., 2010]	Satisfaction	Off-line Review	Adversarial	Game Design	Action/Pref. Model.	Implicit
[Pedersen et al., 2010]	Satisfaction	Off-line Review	Storytelling	Game Design	Preference Model.	Implicit
[Spronck and den Teuling, 2010]	Strategy	Off-line Review	Collab./Adv.	Substitution	Preference Model.	Explicit
[Thue et al., 2007]	Strategy	Online Tracking	Storytelling	Game Design	Preference Model.	Explicit
[Valkenberg, 2007]	Position	Online Strat. Rec.	Adversarial	Spec. in Search	Position Model	Implicit
[Yannakakis et al., 2009]	Satisfaction	Off-line Review	All	Game Design	Action Model.	Implicit

Chapter 4

A Generic Approach for Player Modeling as an ML Problem

> There are two ways of constructing a software design: one way is to make it so simple that there are *obviously* no deficiencies and the other way is to make it so complicated that there are no *obvious* deficiencies.
>
> C.A.R. Hoare The Emperor's Old Clothes" CACM Feb 1981

This chapter discusses how *player modeling* can be tackled as a Machine Learning (ML) problem, presenting a general methodology to apply it. In this methodology, a first step is to define how to represent different players. Hence this chapter also advocates for a generic representation of players proposed by Houlette [2003]. The author introduced it almost as a theoretical model while this chapter discusses its applicability. The next chapter, focused on the instantiation of the methodology proposed here, evaluates the feasibility of the discussed representation.

This methodology was first presented in [Machado et al., 2012a], while most of the discussions regarding the generic player representation are in [Machado et al., 2012b].

4.1 A Methodology for Preference Modeling as a Machine Learning Problem

This section proposes a methodology able to model players following an ML approach. Several player's aspects can be modeled, such as actions, preferences and position. The methodology has six phases:

- Define a representation for the player;
- Define relevant features according to the game;

- Select which relevant examples should be used;
- Model the *player modeling* problem as an ML task, which can be supervised (as done in this thesis) or unsupervised;
- Select the appropriate algorithm(s); and
- Find the best parameters configuration for the selected algorithm(s).

All these topics are generically discussed here, since this approach may be applied to any game in order to define player models.

4.1.1 Representation Definition

As previously stated, a first general concern related to *player modeling* is the representation of different players. It is important to be capable of representing different aspects of a player in the game, being able to obtain models for players with different preferences or knowledge, for instance. This representation is what will be accessed by the game AI in order to generate different behaviors, scenarios or plots, for example.

This step is determinant when defining an ML approach to *player modeling* because the selected algorithm must be capable of generating the player's model, e.g. if one decides to represent a player as a tree, he may struggle to use a neural network to generate it. Hence, this task is required regardless of the approach, using ML or not.

Section 4.2 presents a generic representation that may be used to model players/agents.

4.1.2 Features and Examples Definition

Machine learning algorithms require a set of features as input. For the preference modeling problem, for instance, these features should be able to represent different behaviors of players with different preferences. This is based on the assumption that different behaviors are generated by different preferences. This approach is completely dependent of the game being used as testbed. Once the game platform is defined, a study of the selected data may be useful to assure that the assumptions of the data relevance are correct. We have performed all these steps in Chapter 5 when instantiating this methodology to the game CIVILIZATION IV.

Despite being dependent of the game used as testbed, some general approaches may be common among games of the same genre. For example, strategy games generally present several game indicators along a match. These indicators are related to resources, military and technological characteristics of the game, and they are strong candidate features for an ML technique.

In a First-Person Shooter (FPS) game, indicators derived from players behavior observations, such as life time, the selected weapons and spots, number of kills, among others, may define a player preference. In the case of Action-adventure games, such as the TOMB RAIDER series, player preferences may be defined by other indicators. Among them we can list causes of death, total number of deaths, completion time and demand for help. In fact, all these indicators were used by Drachen et al. [2009] to define models of players in the game TOMB RAIDER: UNDERWORLD.

Although feature definition is essential for any game genre, it is important to stress that the data availability to researchers may vary among different game types. This is mainly because different game developers have different approaches to data extraction. While some allow scripts and even source code modifications, others are extremely reluctant to permit any interaction with the game, other than playing it.

Apart from defining sets of attributes, decisions on how much data to use may also be crucial. For instance, in turn-based games such as CIVILIZATION IV, Spronck and den Teuling [2010] suggested that the first turns should be ignored, because the indicators of different players evolve similarly at the beginning and are not useful to distinguish preferences.

4.1.3 Problem Modeling and Appropriate Algorithms

Once we define relevant features and examples for modeling a player, we need to decide how to represent different players. This representation depends on two different aspects: if we are able to previously identify the players' models and what algorithm we will select for the task.

All three ML approaches discussed in Chapter 2 (supervised, semi-supervised and unsupervised) may be applied to *player modeling*. If the player information is not previously known, unsupervised learning may fit better. In this case, players are clustered according to similar attributes using a distance metric, and the researcher is expected to identify each group by its characteristics. An example of this approach can be found in [Drachen et al., 2009].

On the other hand, if players' models are previously known, we can classify new players based on them. This is the approach presented in this thesis, which uses the preferences of CIVILIZATION IV's virtual agents to identify players' preferences. In general, classifiers are appropriate algorithms for this task.

There are many ways to model this problem using classifiers. Perhaps the most intuitive approach is to model the obtainment of each possible characteristic as a classification problem. Using preference modeling to simplify this discussion, working with concrete concepts, we can say that modeling n preferences requires n different classifiers. For each

classifier, we can predict different types of things. One approach is to predict different levels of preference. For example, the player can be considered to have no preference, weak preference, or strong preference for a specific topic. This is the approach followed in [den Teuling, 2010; Spronck and den Teuling, 2010], called a multi-class problem, in which three classes were considered. We believe that the main problem with this approach is that human players have difficulties in defining their preferences in terms of levels, it is much easier to say simply if they have a preference or not. Furthermore, obtaining datasets detailing different levels can be more complicated than knowing if a player has a preference or not.

Given these drawbacks, an alternative approach that is used in this work, is to model the problem using a binary classification strategy, in which we want to find if the player simply has or has not a preference for a given characteristic. Hence, a binary classifier will predict if a player (example) belongs to the class or not.

Another possible way of modeling this problem is to consider it as a multi-label classification problem. In this case, only one classifier is used, regardless of the number of preferences being modeled. For each example, we could have many different classes (that is the reason for the name multi-label), and a single classifier would be able to predict all of them. For the best of our knowledge, this approach has not been tested for player modeling so far.

Regarding the algorithms used, choosing the best classifier is an open question [Brazdil et al., 2009] and, as discussed in the next chapter, we do not tackle this problem here. This problem is generally studied by the sub-area of meta-learning.

4.1.4 Parameters Configuration

ML algorithms are very sensitive to parameters. Hence, it is very important for researchers to spend some time tuning the algorithm parameters. A common and simple approach for this task, when applied to classifiers, is to perform a grid search on the relevant parameters. We further discuss this topic in the next chapter.

If the researcher does not know the most relevant parameters to be studied, it may be useful to apply some technique that may assist him/her to select them. A common approach is called 2^k Factorial Design, which "is used to determine the effect of k factors, each of which have two alternatives or levels" [Jain, 1991].

An important point when talking about parameter tuning is computational time. The higher the number of examples used for training, the worse the computational time. A simple solution to this problem is to sample the dataset preserving its original features, such as class distribution.

4.2 A Generic's Player Representation

Just as a generic methodology for *player modeling* is a valuable discussion, a generic representation that is able to model different aspects of a player is a useful resource. We advocate here for a representation proposed by Houlette [2003], who did not present any real implementation or evaluation of its feasibility. Thus, our main contribution here is to present a discussion regarding its applicability, theoretically and in the industry, based on several industry requirements listed by Isla [2005].

Besides [Houlette, 2003], as far as we know, one of the few papers that discusses a general methodology for obtaining models through different environments is [Charles and Black, 2004]. In spite of that, it does not validate its assumptions in a real game. A sequence of this work is presented in [Charles et al., 2005], in which the authors also discuss a high-level framework for adaptive game AI. They briefly present an approach for *player modeling* with factorial models but do not investigate further. A more recent work that somehow also discussed a generic representation for agents intentions is [Doirado and Carlos Martinho, 2010]. This work only deals with actions, proposing a framework called *DogMate*. The authors left more general concepts such as preferences out, limiting their model.

4.2.1 Houlette's Representation

The representation discussed here to model players is based on two main components: a set of variables representing specific features of the game and a set of weights that multiply these variables. The set of variables is determined by the game designer and the AI programmer, and it is based on the *domain knowledge* of each game, an inevitable requirement for adaptive AI as Spronck [2005] and Bakkes et al. [2009] previously stated. The weights represent the importance that is given to the feature represented by that variable. They can be manually set by the designer / programmer or learned from experience. Completely different behaviors can be obtained varying the weights of each player, which allows players to adapt to different game conditions.

More formally, the model is represented as a set of n weights, in which n is the number of characteristics that are going to be modeled:

$$Pm = \langle w_0, w_1, ..., w_i \rangle$$

where w_i is a weight for characteristic c_i of the model Pm.

This representation is generic and can be applied to various games such as CHESS, POKER, FPS and Strategy games. It is very simple, yet powerful enough to represent and model different player behaviors. The main characteristic of this representation is that it is

defined as a vector of weights for different game features. This modeling is generic but the model's features/variables are particular to each game and must be defined by an expert. The main advantage of this approach is that, once techniques are created to infer weights, they can be applied to different games that use this approach.

As an example we may use POKER to illustrate our discussion. POKER is a very hard game for computers to play and, so far, they are not able to defeat the best human players. A promising approach is player modeling and there are many works in this area such as [Billings et al., 1998; Davidson et al., 2000; Southey et al., 2005; Bard and Bowling, 2007].

To apply the proposed representation to model POKER players, it is necessary to represent different player behaviors. A very simplistic model, created just for this example, can be extracted analyzing relevant game features such as agents temerity (T_e), probabilistic capacity (P_c) and bluff tendency (B_t). We could define each of these variables as:

- T_e representing the agents tendency to take risks;
- P_c representing agent's awareness about the probability of winning given a card distribution in a match;
- B_t describing how much does an specific agent bluffs during a game.

Once we created this simple set of variables, we generate the following representation, capable of modeling completely different players:

$$Pm = \langle T_e, P_c, B_t \rangle.$$

To exemplify, defining weights between 0 and 1 and a higher value representing a stronger preference, we could easily generate an expert conservative player, represented as $Pm_1 = \langle 0.1, 0.9, 0.2 \rangle$ or an aggressive player, $Pm_2 = \langle 0.8, 0.6, 0.5 \rangle$.

Thus, this representation implies in two different tasks to generate a player model:

1. To define the model variables: with the *domain knowledge* of the specific genre or game, define what are the relevant features to be modeled and in what level of abstraction it will be done.
2. To define the variables weights: this is what distinguishes behaviors, *i.e.*, different behaviors are set in this second step. It can be done to model non-player characters as well as human players.

The definition of variables is an easier task since we just need to define what we want to model, but the weights setup is a very hard task for several reasons. Maybe the biggest one

is the fact that there is no rule of thumb for doing it. Generating these weights (or other type of modeling) from repeated play, observing players and other virtual agents and tracking game results, often involves the use of machine learning techniques and the most used are those inspired in nature as neural networks or genetic algorithms.

After discussing this representation applicability, our focus will be in instantiating the generic methodology proposed here to model players' preferences in the game CIVILIZATION IV. We will use this representation, then our focus on this thesis will be on setting the weights to represent both virtual agents and human players preferences.

4.2.2 Applicability

A useful model must be capable of satisfactorily representing different agents with different characteristics. This topic includes the capacity to allow *any* behavior just deriving it from the model, what could be seen as a coverage requirement. As we already discussed, once appropriate features are selected, this is completely feasible.

Additionally, it is desirable that it is applicable in industry, what would certify its validity. We discuss both topics below.

4.2.2.1 Representativeness

Regarding the representativeness of a model, we consider that it would be effective if capable of performing two different tasks:

1. The generation of different behaviors by varying the model. In this approach, different weights generate different behaviors; and

2. A model that generates a specific behavior must be "inferable", i.e., one must be able to infer a model that generates an observed behavior. This task, in this modeling, consists in inferring variables weights from observation.

The requirements listed above create a cycle: once we observe a specific behavior we must be able to infer the model which generated it and we must be capable of generating different behaviors from the model.

This evaluation must be performed in the specific game that is being used for modeling players. We validate it for the game CIVILIZATION IV in the next chapter, corroborating our assumption about this representation usefulness, since we were able to perform both tasks.

4.2.2.2 Applicability in Commercial Games

An important topic to discuss before presenting the instantiation of the discussed methodology and representation, is related to the applicability of the representation in commercial games. In general, the game industry is somewhat reluctant about the game AI solutions proposed by academy. As discussed by Fairclough et al. [2001]; Laird and Lent [2001]; Nareyek [2004]; Bakkes et al. [2009], there are several reasons for this: the concern about unpredictable behavior, the necessity of heavy modification and specialization for each game, and the difficulty in understanding the reasons for some observed behaviors and in modifying any configuration.

We argue that due to its simplicity and expressiveness, the representation discussed here avoids most of these problems. We use as a base for this discussion the work of Isla [2005], which presents the AI architecture of a very well-succeeded game, HALO 2[1], and discusses several design principles that should be targeted when developing the AI of a complex game.

First of all, Isla defines some basic AI requirements: *coherence, transparency, runtime, mental-bandwidth, usability, variety* and *variability*. Coherence is related to actions' selection at appropriate times, *i.e.*, how we select actions once we have defined an agent model. Thus, coherence is much more related to the mapping between agent models and actions than to the model itself. On the other hand, transparency is one of the major points of this representation, since we are able to understand agents and even try to predict their behaviors only by observing the weights of each modeled feature. This is also related to mental-bandwidth, since we need to "reason about what's going on in the system" [Isla, 2005].

The representation is also important and useful to level designers since it allows usability (the characters are configurable since we only need to change their weights) and variety (the AI works differently for each different weight combination). Variability, as many of the discussed requirements, is related to the use of the model. It is important to observe that, once this model is defined, it is necessary that game programmers use the weights paradigm to develop their games. It is also important to note that this representation does not limit or hinder any desirable feature as neural network or kernel machine models generally do. Moreover, this approach, due to its simplicity, does not impact game performance.

Once we discussed the basic AI requirements, we are able to show that this representation goes further and also meets the design principles proposed by Isla [2005]. The first one is *"Everything is customizable"*, *i.e.*, the model should be general enough to allow modifi-

[1]Bungie Studios, Microsoft Games Studios, Halo 2 (2004): `http://www.microsoft.com/games/halo2/`

cations in behaviors. This is exactly the main advantage of this representation since it can be seen as an organized set of parameters, a clear definition of customization.

The second design principle, *"Value explicitness above all other things"* is obviously met by the explicit use of weights. Also, the use of weights simplifies the generation and test of specific behaviors and makes easier the process of generating relatively similar behaviors with some small variations in each weight. This is exactly Isla's fourth principle: *"Take something that works and then vary from it"*. (The third principle, *"Hackability is key"*, is not applicable directly to the representation but to other programming levels).

It is interesting to note that, despite the independent development of [Houlette, 2003] and [Isla, 2005], it seems they were developed together because of the similarity in most of the discussed requirements. A final sentence of Isla [2005] evidences its usefulness: "... we are not interested in a scripting system in which the designer specifies EVERYTHING the AI does and where it goes - that would be too complex. We do need, however, the AI to be able to handle high-level direction: direct them to behave generally aggressively, or generally cowardly.".

Hence it is clear that this representation is applicable in the industry. After defining a generic methodology and representation to *player modeling*, we must instantiate it to our problem, the preference modeling in CIVILIZATION IV. This is the subject of the next chapter.

Chapter 5

Player Modeling in Civilization IV

"It's a job that's never started that takes the longest to finish."

J. R. R. Tolkien, The Fellowship of the Ring

This chapter instantiates the generic ML approach proposed in previous chapter to the game CIVILIZATION IV. This instantiation takes into account our final goal, which is to model/predict the CIVILIZATION IV player's preferences, namely *Culture, Gold, Growth, Military, Religion* and *Science*.

Here we present the steps required prior to the execution of the ML algorithm, which final results are presented in Chapter 6. Recall that the proposed methodology had six phases: (1) defining a representation for the player; (2) defining relevant features according to the game; (3) selecting relevant examples that will be used; (4) modeling the problem as a ML task; (5) selecting the appropriate algorithms; and (6) selecting best parameters configuration.

Considering these phases, we first discuss the dataset that is used in our experiments, generated from CIVILIZATION IV's matches played by virtual agents. We then perform an evaluation of the available data, validating its relevance and using it to select the appropriate algorithms to model players' preferences. Finally, we evaluate the representativeness of the generic player representation, presented in Chapter 4, using the game CIVILIZATION IV.

5.1 Dataset

Chapter 2 presented the game CIVILIZATION IV, its mechanics and programming interface, discussing the script *AIAutoPlay* that allows us to log matches between two virtual agents. Based on these topics, we are able to sniffer the game and capture data while virtual agents (or human players) play. Using this feature, den Teuling [2010] was able to generate a dataset to classify players preferences (we will further discuss this classification approach in the next

sections). We present the dataset at this point because, from now on, most of our results are obtained from it.

Three different datasets were used in this thesis, two composed of virtual agents gameplay data, which we discuss here, and other created with human players data. The first two were created by den Teuling [2010] and edited here to our purpose. The third will be discussed in Chapter 6.

The first decision to be made in order to sniffer gameplay is when to capture data. In the case of CIVILIZATION IV, the game structure eases this decision because its turn-based pace clearly defines the data collection moment, at the end of each turn.

As we previously said, den Teuling [2010] used the script *AiAutoPlay* to generate the dataset. This script allows the game to be played by two virtual agents, removing the requirement of human players. This is an important feature because it allows the dataset to have hundreds of matches, what would be impossible if human players had to play these matches. den Teuling [2010] modified this script to collect, for each turn, a set of game indicators. We use this same dataset, and its main features are described next.

The first dataset, called *Traditional Dataset*, was generated by randomly selecting six leaders in the game, and making them play against each other eight times, in a total of 40 games per leader. For each turn, information of each agent was collected. At the end, the shorter game had 240 turns while the longer took 460 turns (the maximum allowed).

From all the data collected, we kept the same 21 features used in [den Teuling, 2010; Spronck and den Teuling, 2010], which are game indicators available to every player during the game. These indicators are scores and counters, modified by players actions, and here we refer to them as features. They all describe some aspect of the game, and a subset is presented in Table 5.1. A complete list is presented in Appendix A.

Table 5.1: *Subset of features and their meanings. The features are related to one player, e.g., the number of units indicator is the number of units that a specific player has.*

Feature	Meaning
Turn	Turn number
War	0 = not in war; 1 = in war
Cities	Number of cities
Units	Number of units
Economy	Overall economic score
Industry	Overall industrial score
Culture	Overall cultural score
Maintenance	Gold needed for maintenance per turn
ResearchRate	Amount of research gained per turn
CultureRate	Amount of culture gained per turn

Notice that the decision to use the same features used by den Teuling [2010]; Spronck and den Teuling [2010] was not made without consideration. In Section 5.2 we show that some indicators are able to distinguish different behaviors, which supports our decision. In fact, these features were used in all three mentioned datasets.

den Teuling [2010] modeled each turn as a vector (example), and since evolution along the match is an important factor, they extended the set of basic features to add this notion of time. The authors name these new features, which are are presented in Table 5.2, *composite features*.

A second dataset, also related to virtual agents, was created to evaluate generalization. We call it *Alternative Dataset*. It was created using a different set of six agents and we will further discuss its use in Chapter 6, where it is used.

Table 5.2: *List of composite features, as in [den Teuling, 2010].*

Modification	Calculation	Meaning
Derivate	$v_t - v_{t-1}$	Increase or decrease in the base feature per turn
Trend	$\frac{(\sum_{i=0}^{4} v_{t-i})}{5}$	Average of base feature over multiple turns
TrendDerivate	$v_t - v_{t-5}$	Derivate of the trend
Diff	$v_t - w_t$	Difference of the base feature with the opponent's
DiffDerivate	$(v_t - w_t) - (v_{t-1} - w_{t-1})$	Derivate of the difference
DiffTrend	$\frac{(\sum_{i=0}^{4} v_{t-i} - w_{t-i})}{5}$	Trend of the difference
DiffTrendDerivate	$(v_t - w_t) - (v_{t-5} - w_{t-5})$	Derivate of the trend of the difference

Each different virtual agent has a specific set of preferences, which are defined by different values representing levels (0 – no preference; 2 – medium preference; 5 – high preference). For our classification experiments, as we will mention when discussing our ML approach, we have removed the first 100 turns of each match to make our results comparable to our baseline.

5.2 Features Definition

A first important step when discussing the available data is to better understand virtual agents behavior and how they are expressed in game data. The evaluation of the data generated from gameplay of different players is what we use to define our features in the proposed ML approach.

In order to perform this task, we model the game CIVILIZATION IV as a set of states, where each state is defined by the data gathered at the end of each player turn. These data consist of several game *information* like, for example, the amount of gold a civilization has or the number of cities. Moreover, each virtual agent in the game may have different

preferences, which are descriptions of the way the agents play, *i.e.* their main priorities during the game such as gold, culture or religion, as discussed in Chapter 2.

To evaluate the usefulness of the available gameplay data, and consequently to define our features, some questions must be answered:

- The information of intermediate states of the game do characterize distinct preferences of different agents?
- What available information distinguish the agents preferences? What is the relation between their predefined attributes and this information?

In this section we answer these questions with a characterization of the behavior of AI controlled agents, looking for relations between the agents predefined preferences and their behavior. The discussions presented here are derived from the results in [Machado et al., 2011b].

5.2.1 Methodology

Our objective is to characterize the behavior of different agents by its gameplay data, and to correlate them with their preferences. This is done by generating linear regressions based on game state indicators, gathered in several matches played between different AI agents. Our intuition was that we would be able to find different functions describing game data for different agents, since they have different preferences, what would justify the selection of this data as a feature.

To perform this evaluation we used a subset of the *Traditional Dataset*, discussed in Section 5.1. We studied three agents preferences: *Culture*, *Gold* and *Growth*. The characterization was performed by observing games between two different agents and analyzing the data generated by these observations. We have carefully chosen these agents in a way that one of them has no interest in a certain preference and the other has high interest in this same preference (values 0 and 5 in the game, respectively). This was done to simplify the comparison between indicators that were supposed to indirectly represent preferences, *i.e.*, we expected a higher value for an agent indicator that has a high interest in the preference related to that indicator.

For example, the agent called *Mansa Musa* has a high interest in *Gold* while the agent *Louis XIV* has no interest in it. Based on this fact, we compared some of their indicators to model their *Gold* preference, looking for different functions to each agent. In fact, we expect *Louis XIV*'s indicators related to gold to be lower than *Mansa Musa*'s indicators since *Mansa Musa* has a higher preference.

We start from the premise that the adversarial agent actions do not impact the state of the player we are analyzing, and we characterize several of its behaviors and preferences. After this first phase, we relax this premise and we observe that it is correct, as we will show in Section 5.2.2.4. This result give us confidence to assume this independence when applying our ML approach.

We have used the leaders in the example above to analyze *Gold* preference (*Louis XIV* and *Mansa Musa*). To analyze the *Growth* and *Culture* preferences, we have used the agents *Alexander* and *Hatshepsut*. The *Growth* preference has a peculiarity: in our dataset there was no agent with a high interest on this preference (value 5), just an average interest (value 2). We used the agent *Alexander* as the one having interest on it while *Hatshepsut* was the one who has no interest.

In all analyses, we have characterized each preference comparing the agents states in each turn, seeking for a function capable of representing this evolution. We did linear regressions in the data and, when the data did not fit in this model, we applied transformations on it to be able to use a linear regression, since the mathematical analysis is simpler and it does not imply in a loss of generality. A liner regression generates functions in the form $y = b_0 + b_1 x$. Our main concern is b_1, since it represents the evolution of the indicators in the game.

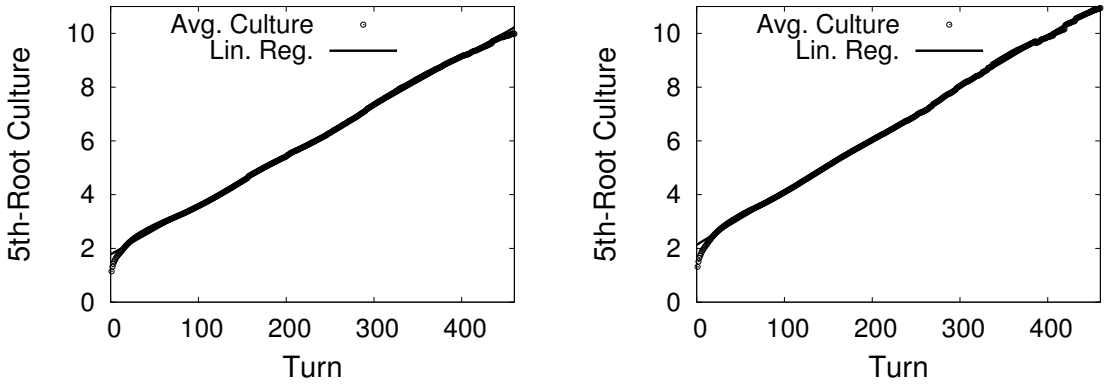
We have summarized the agents states calculating, for each turn and for each indicator, the average of 40 matches (the amount of matches each agent played in the used dataset). At the end, we had 460 points where each point p_i represented the mean of turn i for all agent matches. We have selected some indicators collected during gameplay under the premise that they would be relevant features to distinguish the studied preferences. These indicators were selected intuitively based on our knowledge about the game. All the regression algorithms and evaluation metrics used here are discussed in [Jain, 1991].

After the characterization of the three listed preferences, we separated matches by their results: victory or defeat. We revisited every analysis, using two different sets to understand the game result impact in our characterization, allowing us to answer the question whether the result influences the analyzed functions.

Next we will present the characterization of each modeled preference. After this first analysis, in Section 5.2.2.4, we analyze the impact of separating games by their result.

5.2.2 Agents Characterization

This section presents the characterization of three different preferences, evaluating the use of seven different features. The preferences characterized using the mentioned features are: *Culture, Growth* and *Gold*.



(a) *Alexander* **(b)** *Hatshepsut*

Figure 5.1: *Linear Regression of the* $\sqrt[5]{Culture}$ *indicator.*

We summarize regressions data in Appendix B, in which we present each obtained coefficient, their confidence interval, and the regression's coefficient of determination[1].

5.2.2.1 *Culture* Preference

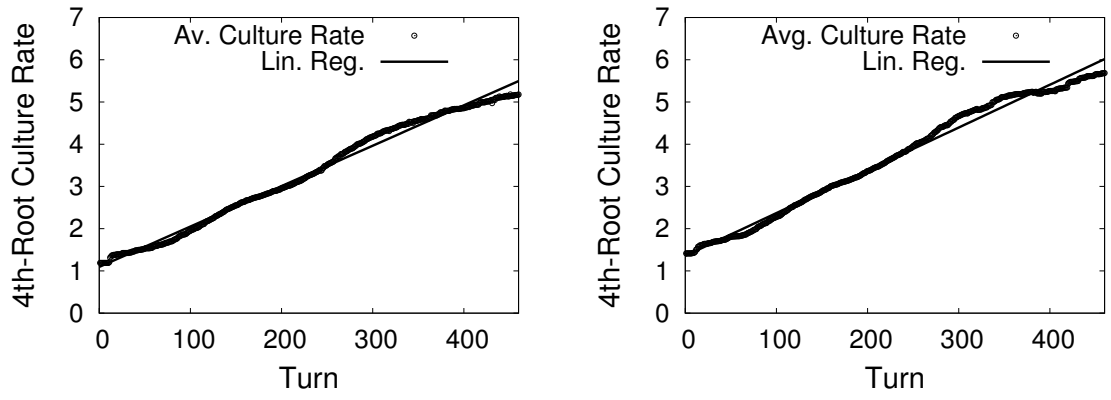
We have selected two indicators (features) to distinguish this preference: *Culture* and *CultureRate*. As previously discussed, the characterization was done using the agents *Alexander* and *Hatshepsut*. The indicators are defined in [den Teuling, 2010] as being the "Overall cultural score" and the "Amount of culture gained per turn", respectively.

We were able to characterize almost perfectly this preference with the two selected indicators. To do it we have modeled the *Culture* indicator as a polynomial of degree five, and *CultureRate* as a polynomial of degree four. This was very satisfying since it is the order of the derivative of the polynomial that represents the *Culture* (as expected, we have tested regressions of these indicators to other functions, we decided to present only the best result). As we discussed in the previous section, a linear regression simplifies this analysis without loss of generality, so we applied the fifth root to all values of *Culture* and the fourth root to all values of *CultureRate*.

We obtained very high coefficients of determination to the *Culture* (99.86% to *Alexander* and 99.85% to *Hatshepsut*) and *CultureRate* indicators (99.11% and 98.93% to *Alexander* and *Hatshepsut*, respectively). Besides this, all obtained coefficients are significant with a confidence of 99%. The graphs with the regressions are presented in Figures 5.1 and 5.2.

Beyond the regression quality, it is important to note that the coefficients b_0 e b_1, of *Hatshepsut*, are greater than those of *Alexander*, with a confidence of 99%. This is what

[1] "The fraction of the variation that is explained determines the goodness of the regression and is called the coefficient of determination, R^2" [Jain, 1991].



(a) *Alexander* **(b)** *Hatshepsut*

Figure 5.2: *Linear Regression of the* $\sqrt[4]{CultureRate}$ *indicator.*

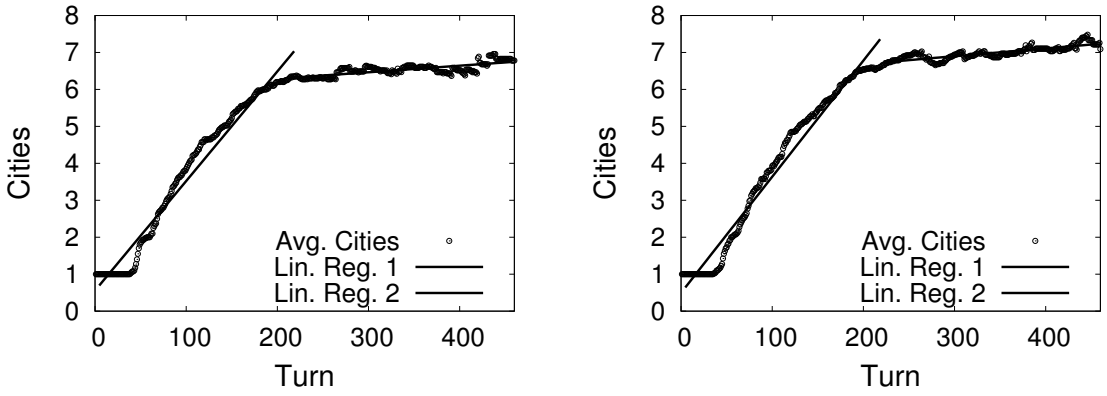
we expected in this situation since *Hatshepsut* has a higher *Culture* preference. This result confirms our hypothesis that some gameplay data are able to distinguish preferences of two different agents, suggesting that they can be used as features when using ML to model players preference.

It is interesting to note that this preference has very few interactions with other game indicators, and maybe *Culture* is the easiest preference to be isolated, since only specific constructions in the game generate culture score. Among the buildings that generate culture are: palaces, educational and religious buildings and wonders. Most of the buildings that generate culture score do not exist at the beginning of the game and they are constructed along the game, explaining why we obtained a polynomial of fifth degree. In fact, we believe that it should be represented by an exponential function, but the limited number of turns does not allow enough growing.

5.2.2.2 *Growth* Preference

We have analyzed the *Growth* preference observing three different indicators (features): *Cities, Land* and *Plots*. The first one is defined as the "Number of cities", the second as "Amount of land tiles" and the third as "Amount of land and water tiles" [den Teuling, 2010]. Recall that *Alexander*, the chosen agent to represent a high preference, did not have this preference on its higher level (value 5), but on average level (value 2).

The analysis of these three indicators presented a recurrent and expected situation: the existence of two distinct intervals in the dataset. Initially, there is a period in which the growth rate (of *Cities, Land* or *Plots*) is high. This *expansionist* period occurs when there are unoccupied lands that are easily dominated. After this initial period, we can observe



(a) *Hatshepsut* **(b)** *Alexander*

Figure 5.3: *Linear Regression of the Cities indicator.*

a *maintenance* phase where there is almost a stabilization of these indicators, since all the world has already been "colonized" by some agent. The turn number we have chosen as turning point is also presented in Appendix B.

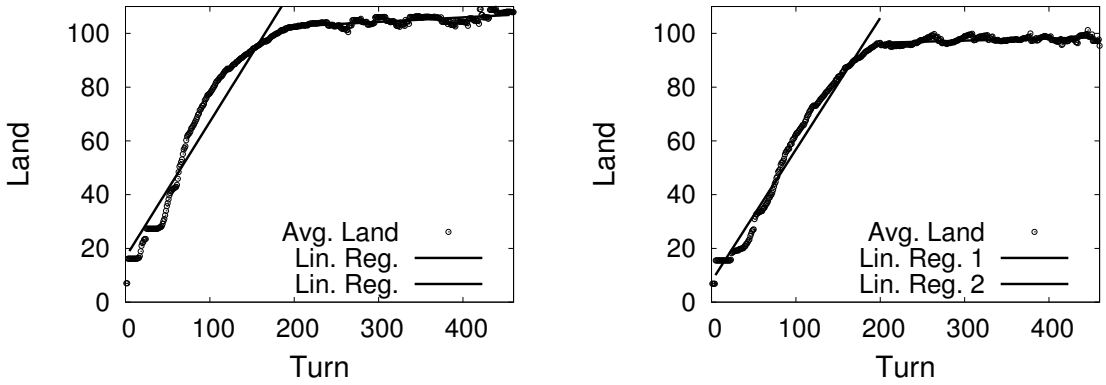
We were able to model mainly the expansionist period, i.e. to obtain functions that fit well in the available data. All these functions were modeled as line segments, one for each period.

The best indicator for *Growth* characterization was *Cities*. For this indicator, we were able to obtain a linear function representing the expansionist period with coefficients different from zero with a confidence of 99%, and a coefficient of determination equals to 97.17% to *Alexander* and 96.80% to *Hatshepsut*. The second line segment, of the maintenance period, was not so successful in modeling the agents behavior. As in the first line segment, all coefficients are different from zero with a confidence of 99%, but we were only able to achieve a coefficient of determination equals to 71.39% to *Alexander* and 56.02% to *Hatshepsut*. The regressions of these indicators are in Figure 5.3.

In the expansionist period, we were able to show that the model coefficients are different between agents. As we said, using the model $y = b_0 + b_1 x$, the coefficient b_1 is larger for *Alexander* with a confidence of 99%. The equality of b_0 is also expected since all agents start with the same number of cities.

We were also able to show that the coefficients in the second line segment of *Alexander* are greater than those of *Hatshepsut*. This result is not so important due to the coefficient of determination of these regressions, but it is still interesting to note that these data corroborate the hypothesis that agents with a higher preference by *Growth* have larger coefficients.

The *Land* indicator allowed us to characterize the expansionist period (coefficient of determination equals to 97.90% to *Alexander* and 93.76% to *Hatshepsut*), with a confidence



(a) *Hatshepsut* **(b)** *Alexander*

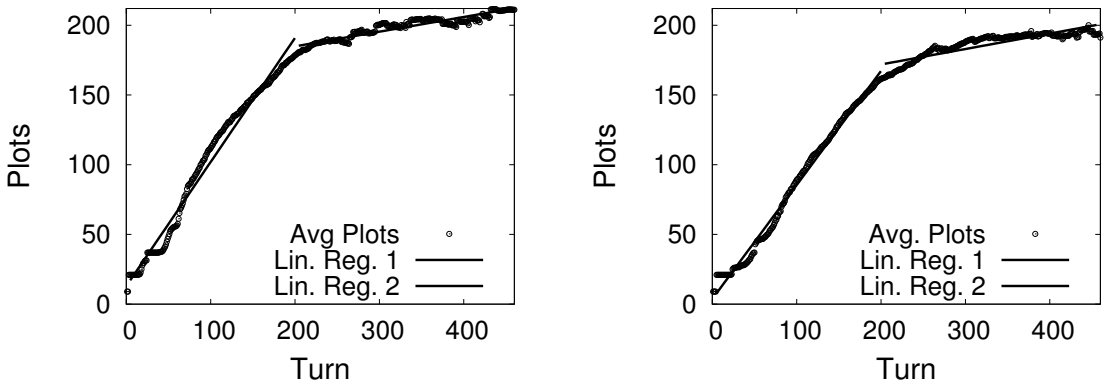
Figure 5.4: *Linear Regressions of the Land indicator.*

of 95% that the coefficients are different from zero – we were able to achieve $b_1 \neq 0$ with a confidence of 99%, but we were not able to distinguish the coefficients between agents. In the maintenance interval, the regression did not explain the data nicely, since *Alexander* and *Hatshepsut* coefficients of determination were, in this interval, 23.90% and 50.04%, respectively. Even being able to show that the coefficients are different from zero with a confidence of 99%, there is no sense in evaluating the intersecction between these two agents. Figure 5.4 presents this regression and, as the others regressions, its coefficients are presented in Appendix B, in which the confidence intervals are presented with a confidence of 90% to show that relaxing the confidence interval still does not allow the coefficient separation.

There is a reason to the *Land* indicator be descriptive but not discriminative. As the *Cities* indicator, initially there are too much to be conquered and this allow us to precisely describe the expansionist period, but not the maintenance period, since there is a natural unpredictability in the game after its stabilization. This unpredictability is smaller to the *Cities* indicator because it is harder to have a decrease in its value, since it is easier to lose territory's tiles than to lose cities. Due to this, the *Land* indicator has a higher variability.

This alteration also depends on other preferences like *Culture*, that also makes it harder to be modeled. We believe the inability to discriminate the generated model coefficients are due to the fact that *Land* can also grow with investment in culture, not necessarily just building cities. A player who privileges cities may evolve its territory just like a player that does not, but invest in culture, what raises its cities borders and maybe imply in a high *Land* value. The non-independent coefficient (b_1) is the growth rate of the agents borders, this implies that the cities creation generates peaks in some curve points but, in general, this is amortized since we generally have a maximum of 10 cities and 460 turns.

Finally, the last indicator we evaluated was *Plots*. This indicator is the sum of the



(a) *Hatshepsut* **(b)** *Alexander*

Figure 5.5: *Linear Regressions of the Plots indicator.*

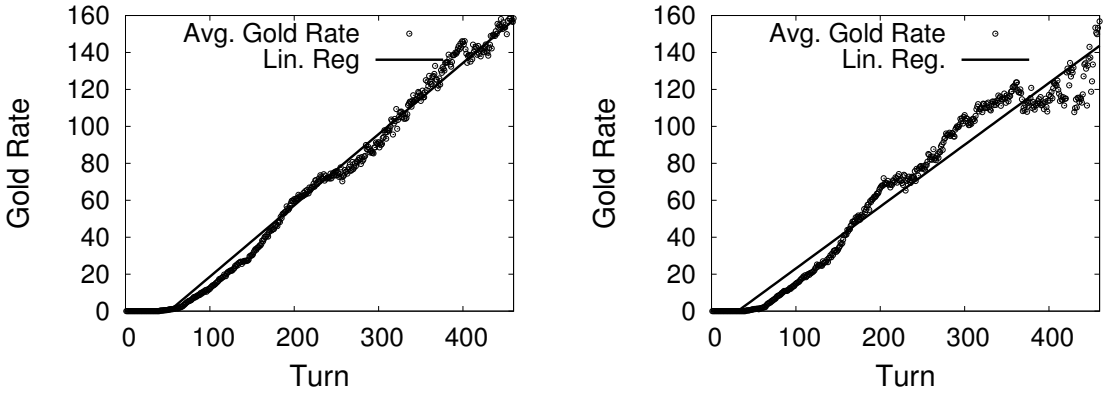
number of land and water tiles in the game. As *Land*, we were able to nicely describe the expansionist period but we were not able to discern the two agents (*Alexander* coefficient of determination is 99.15% and *Hathshepsut* is 95.05%, with b_1 different from zero with a confidence of 99%). All the discussions previously done are also applicable here. The biggest difference between these two indicators is related to the second phase, the maintenance. We were able to achieve better models than those obtained using the *Land* indicator (R^2 equals to 78.73% to *Alexander* and 88.22% to *Hatshepsut*, with coefficients different from zero with a confidence of 99%). The regression of these indicators is in Figure 5.5. We believe this higher "stability" is explained by the water *tiles*, that are harder to be lost by reasons like *Culture*.

In conclusion, as *Plots*, *Land* coefficients overlap. Based on this, the number of cities in each turn is the unique indicator that allows us to discern agents with different preferences, while all indicators successfully describe the agents behaviors. It is interesting to highlight that even not using an agent with a high preference for *Growth*, we were able to distinguish agents with different preferences, showing that even intermediate levels may be useful.

We were able to observe that the characterization/differentiation sometimes is impaired by the interaction of different preferences in the indicator. But these results also gave us hints that the data representing player's state during the game can be used as features by an ML algorithm.

5.2.2.3 *Gold* Preference

The selected indicators for this preference were *Gold* and *GoldRate*. They are respectively: "Amount of gold" and "Amount of gold gained per turn" [den Teuling, 2010] .



(a) *Louis XIV* **(b)** *Mansa Musa*

Figure 5.6: *Linear Regression of the GoldRate indicator.*

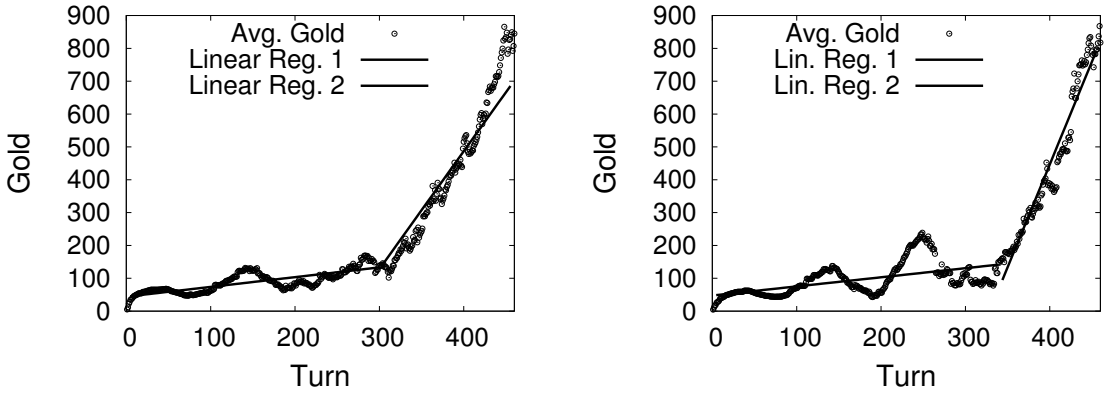
We were able to model the *GoldRate* indicator for the two agents as a straight line. The coefficient of determination of the linear regressions to *Louis XIV* and *Mansa Musa* was, respectively, 98.72% and 96.14%. Besides this, the models coefficient b_1 of each agent are different from zero with a confidence of 99% (we were not able to show b_0 different from zero, what is not a problem since it represents the initial value).

These regressions indicate us that the amount of gold received each turn grows following a linear function but, apparently, the agent preference does not impact in the way it receives gold. This affirmative is valid because even relaxing the confidence of our evaluations we were not able to find intervals that do not overlap. Figure 5.6 presents a visual evaluation of the regressions.

The gold indicator is not modeled as a polynomial of degree two (integral of the gold rate) because it is not the total gold income, but the income decreased by agent's expenses. Still, apparently, the amount of gold received each turn is similar, independently of the player preference. We then raised a second hypothesis that the amount of gold stored by each agent would be different.

The best characterization we achieved for this indicator was using two different line segments. We believe this is due to the gold importance in the game and the several activities that can be done spending it, like donations to other agents, conversion in units upgrades and even receiving it due to the incapacity to continue constructing some buildings, for example. Figure 5.7 exemplifies very well this variable behavior.

Thus, each graph is divided in two line segments. For *Louis XIV*, the first segment goes from turn 1 to turn 300, while for *Mansa Musa*, this first interval goes from 1 to 340. As we can observe in the graph, there is a large variability in this first segment while the second segment is more stable. Despite this analysis, we were not able to assure that the regression



(a) *Louis XIV* **(b)** *Mansa Musa*

Figure 5.7: *Linear Regression of the Gold indicator.*

coefficients are different from zero (the lowest evaluated level of confidence was 90%).

Our premise that the indicators *Gold* and *GoldRate* would describe well the agents behavior was partially satisfied since we were able to characterize the *GoldRate* growth but we were not able to do the same for the *Gold* indicator. We believe this difficulty to describe this preference is due to fact that *Gold* is one of the "most common" and important resources in the game, permeating several possibilities, what "degenerates" the *Gold* evolution during the time.

This last preference being analyzed was harder to be characterized and we were just partially able to model it, since the *GoldRate* describe it but not the *Gold*. The reasons for this are easy to be comprehended after the analysis of the results: creating a city (which we just related to *Growth* preference) implies in a great loss of gold since the cost of new cities is greater than its income. Thus, the variation observed is explained by this. The great "jump" after turn 300 can be explained by the "discovery" of mercantilism, besides the fact that most of the cities became profitable.

In summary, we were unable to distinguish different agents preferences related to *Gold* and we believe this is because gold is an essential resource in the whole game and the preferences are "weaker" when compared to other characteristics not so essential like *Culture*. The reason for this claim is that agents can obtain this resource from different ways and, for a better player experience, a better balancing of this distribution is expected.

Due to our results and discussions during this section, the impact of preferences interaction to distinguish and characterize agents became very evident with this preference. This may be another explanation to our failure in distinguishing different agents.

This last result shows that, for some game variables, the evaluation of a unique feature may be insufficient to obtain an agent preference. These results also gave us confidence

that a set of features containing gameplay data can be used in order to distinguish player preferences using ML algorithms.

5.2.2.4 Victory and Defeat

Prior to continue modeling our problem as an ML approach, it is important to evaluate the influence of the game result in the features' discriminative power. To do this, we separated the data in two disjoint subsets: those originated from matches that were won and those from lost matches. This decision was motivated by the following question: the analysis of all matches as being in the same group, independently of their result, does not generate noises that distort the real agents behavior? Their separation does not make these data more "stable"?

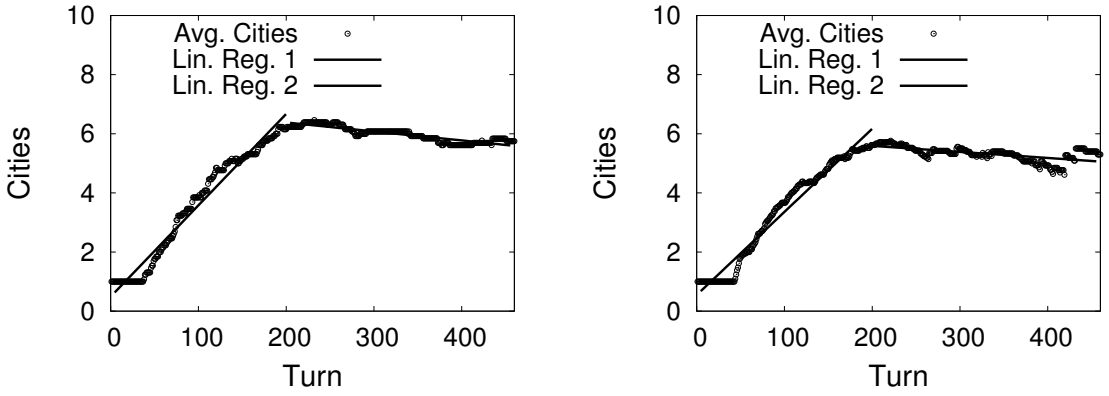
To answer this question, we revisited every generated model recreating it for the two different subsets, *i.e.* each previous model generates two others, using data of victories and defeats. The intuition is that the game result would have impact in the indicators. We present this analysis below.

Regarding culture, another analysis over this preference is useful just to validate the results previously obtained, since they were extremely satisfactorily. Again, we modeled *Culture* as a polynomial of degree five and *CultureRate* as a polynomial of degree four, for both won and lost matches. As in the previous modeling, our regressions were very good for both sets (R^2 greater than 98% to all indicators, for both agents) and all obtained coefficients are not zero with a confidence of 99%.

As obtained in the general analysis, the *Hatshepsut* coefficients were greater than those from *Alexander*, who has no interest for culture, while *Hatshepsut* has.

To model *Growth*, we followed the previous methodology, dividing all indicators in two periods: expansionist and maintenance. In this preference, we were able to observe benefits of the separation between matches that were won or lost. The benefit was a decrease in the data variability and a better understanding of the problem. We were not able to obtain a more distinguishable model for the different agents.

The first evaluated indicator was *Cities*. As in the general evaluation, the expansionist period was easily characterized to *Hatshepsut* and *Alexander* for victory (R^2 equals to 98.01% and 97.91%, respectively) and for defeat (R^2 equals to 96.98% and 96.58%, respectively), with a confidence of 99% that the coefficients are not zero. It is interesting to note that, in the subset of matches won, the coefficients overlap, while they do not in the matches lost. The non-independent coefficient of *Alexander* was greater than the one of *Hatshepsut* with a confidence of 95%.



(a) *Alexander (Defeat)* **(b)** *Hatshepsut (Defeat)*

Figure 5.8: *Linear Regressions for the Growth preference in the lost matches subset.*

Besides this, when *Hatshepsut* presents a "more" expansionist behavior, it increases its victory chances. We believe the statistical difference is achieved only in lost matches because even under adverse situations, *Alexander* still aims to expand his borders while *Hatshepsut* does not.

In the stabilization period, we were able to better characterize the models of won matches due to the lower data variability (R^2 equals to 95.11% to *Alexander* and 81.79% to *Hatshepsut*). This result corroborates our hypothesis that the game result may influence some indicators, since we observed a higher variability in the lost matches, probably due to the different types of victory: an agent can lose a game by score (without a single military conflict), or may have its lands devastated by the enemy. The coefficients of determination to *Alexander* and *Hatshepsut* in this situation were, respectively, 79.84% and 39.59%. This difference between them is probably explained by the military preference of the first, not evaluated here. He is probably better able to defend his lands, even when he loses the game.

Figure 5.8 presents the number of cities in the lost matches. There is an interesting result here: we have observed that, for the first time, the non-independent coefficients (b_1) were lower than zero, i.e. when an agent loses a game, its territory decreases at the ending.

Once we finished this analysis, by the first time we were able to note the variability generated by the game result, precisely in the number of cities. The observed variability is large because in some matches, after a specific turn, the cities may continue increasing or start decreasing.

The analysis of the *Land* indicator for victory and defeat was very similar to the general analysis. We were able to characterize well the expansionist period but we were not able to distinguish the coefficients between agents, not even for matches won or lost.

As in the general analysis, the second interval presents a much higher variability. In

the set of won matches, the coefficient of determination of *Alexander* and *Hatshepsut* was, respectively, 93.65% and 53.95%, with all coefficients different from zero with a confidence of 99%. This difference has already been discussed.

We were able to show in this regression that the growth rate for *Alexander* is greater than *Hatshepsut*'s growth rate, also with a confidence of 99%. The results for lost matches were extremely variable and any discussion comparing them is meaningless. Despite the variability decrease in the won matches subset, we were not able to obtain additional useful information with this separation.

The last indicator related to this preference that needs to be revisited is *Plots*. The division keeps the excellent characterization of the expansionist period in the victory and defeat subsets, but we were still unable to distinguish different agents. The data variability decreased after the division but no additional discussion or analysis is possible.

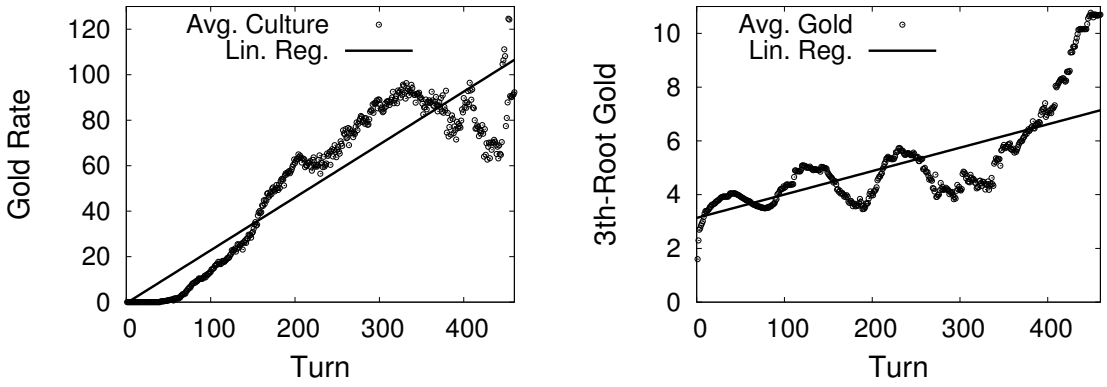
Finally, regarding *Gold*, firstly analyzing the victory subset, we were able to obtain very good regressions to the *GoldRate* indicator, with an R^2 equals to 98.48% for *Louis XIV*, with all coefficients different from zero with a confidence of 99%. We also achieved an R^2 of 97.03% for *Mansa Musa*, with a confidence of 95% that all coefficients are different from zero. Similarly to general characterization, the coefficients of these regressions overlap. The same happened in the lost matches subset. We were able to obtain coefficients different from zero with a confidence of 99% and regressions with R^2 equals to 97.98% to *Alexander* and 84.60% to *Mansa Musa*. Figure 5.9a presents the regression for this last case.

The analysis of the defeat situations shows us that the higher variability observed in the general data of *Mansa Musa* is explained by the matches lost. Probably this occurs because he loses cities at the end of the game and this implies in a smaller gold rate.

This analysis, despite helping us to understand the higher variability in this agent's indicators, did not allow us to distinguish both, and we believe the reasons are the same previously discussed.

We modeled all gold scores as single lines. *Mansa Musa*'s gold indicator in matches lost could have been characterized as two distinct lines, but we decided to keep its regression similar to the others of this indicator, as it can be seen in Figure 5.9b.

As in all other graphs, we can observe a high variability of the indicator values and, as previously discussed, probably due to the cities feature. In the matches won subset, we observed that *Louis XIV*'s non-independent coefficient (b_1) is greater than *Mansa Musa*'s coefficient, that has preference for *Gold*. This relation is inverted on the subset of matches lost. Is interesting to note that these results seem counter intuitive since we would expect a higher coefficient in all cases. It does not happen because *Louis XIV* has a high preference for *Culture* and this influences its amount of gold since the *Culture* generates a territorial expansion that implies in a higher number of resources.



(a) *Mansa Musa (Defeat)* **(b)** *Mansa Musa (Defeat)*

Figure 5.9: *Linear Regressions for the Gold preference in the lost matches subset.*

In summary, at the end of all these analyses we achieve two different conclusions: first of all, for most of the analyzed data, the separation between matches won and lost generally implied in getting a better fit for half of the models and a worse fit for the other half. Despite being able to better characterize the separated models, we are not able to distinguish agents preferences that had not been already distinguished by the general data.

Whether the division is beneficial is not a trivial decision. It depends on the chosen ML technique, since the fusion between victory and defeat may make implicit the average performance of the agent: if it tends to lose more than win, with the time we would be able to better characterize its losing performance since natural "weights" would raise automatically with the results accumulation. This would be more difficult considering separated matches.

5.3 Players Representation

Regarding the generic representation proposed in previous chapter, in this section we instantiate it to the game CIVILIZATION IV. As already discussed, a good model must be able to generate different behaviors by varying the representation of a virtual agent, as well as to be "inferable", i.e. by observing behavior one must be able to represent it. We discuss these two topics in sequence.

In order to evaluate whether it is possible to infer agents' models, we perform an experiment in which we manually infer some of the weights that could model virtual agents, comparing these weights to those in their predefined models. This shows how different behaviors can be explained and expressed by different models.

A second experiment to be performed would be the evaluation of whether the weights variation would generate distinct behaviors. However, this task would be extremely compli-

cated to be performed in the game CIVILIZATION IV. Machado et al. [2012b] further discuss this topic using the game COUNTER STRIKE as test-bed. We show how small changes in the model can generate different behaviors in the game. Here, we decided to not discuss this topic further, to keep the focus on the game CIVILIZATION IV.

5.3.1 Civilization IV: from behaviors to models

As we are interested int inferring models by observing players, we used the game CIVILIZATION IV to perform our tests, since it allows data to be easily extracted and explicitly presents agents characteristics in *XML* files, allowing us to convert it into our representation.

As previously stated in Chapter 2, CIVILIZATION IV provides *XML* files with several game characteristics, such as agents preferences, buildings and units info. This interface allows us to observe and modify the game characteristics. Each agent presents several attributes in the XML defining them. This is quite useful since we are able to correctly generate relevant variables for the model. Six agents preferences are defined in the game: *Culture, Gold, Growth, Military, Religion* and *Science*. Each of these preferences serve as multipliers to agents decisions and cost of specific actions.

Since we do not intend to present the best possible model for the game, we decided to use the preferences described and valued in the *XML* to generate an agent representation:

$$Pm = \langle C_u, G_o, G_r, M_i, R_e, S_c \rangle$$

where C_u is preference for culture, G_o for gold, G_r for growth, M_i for military, R_e for religion and S_c for science.

Our goal is to infer different models from different behaviors, and verify if this representation is coherent with observed behaviors. Hence, just like in Section 5.2, we selected two different agents: one with a high preference for a specific feature and other without it. Our goal was to analyze game indicators that would, theoretically, be affected by this preference. We expect higher weights for the agent with a higher preference.

For example, agent *Alexander* has a strong *Military* preference while agent *Hatshepsut* prefers *Culture*. These agents may be represented by us using the information available on the *XML* files. Extracting the respective weights from the configuration files, we obtain the following models (Pm_A for *Alexander* and Pm_H for *Hatshepsut*):

$$Pm_A = \langle 0, 0, 2, 5, 0, 0 \rangle$$

$$Pm_H = \langle 5, 0, 0, 0, 2, 0 \rangle$$

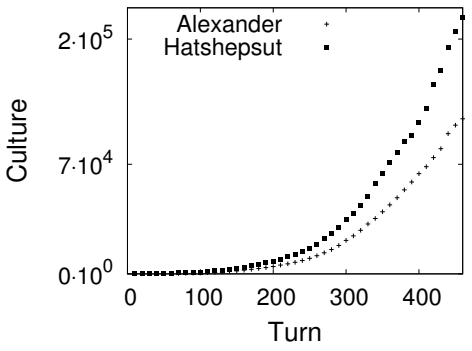


Figure 5.10: *Comparison of Culture between different agents.*

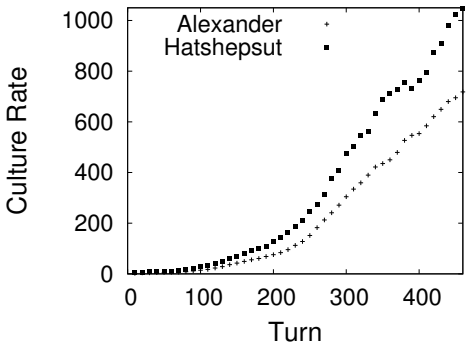


Figure 5.11: *Comparison of Culture Rate between different agents.*

As we can see, the set of variables different from zero of each agent is a disjoint set and it is expected to result in distinct behaviors during the game, assuming that the above representation is useful.

In fact, in Section 5.2, we already showed differences between different agents. We will discuss here some of those results applied to this evaluation. We used the same dataset used to characterize agents with linear regressions.

The plots for *Alexander* and *Hatshepsut* overall culture score along all matches is presented in Figure 5.10. An analogous plot, but for *Culture Rate*, is presented in Figure 5.11. Note that differently from previous section, we did not apply any transformation to the data.

As expected, based on the agents' models, *Hatshepsut*'s curves represent a superior bound for *Alexander*'s curves. This is coherent with *Hatshepsut*'s higher preference for *Culture*.

Performing the linear regressions previously discussed, we obtained the following straight line equations for *Alexander* (y_A) and *Hatshepsut* (y_H), for *Culture*. Between parenthesis we present the coefficient of determination.

$$y_A = 0.0183x + 1.7772\ (99.86\%)$$

$$y_H = 0.0194x + 2.1366\ (99.85\%)$$

Using the same pattern, the line equations for the *Culture Rate* indicator were:

$$y_A = 0.0096x + 1.0939\ (99.93\%)$$

$$y_H = 0.0101x + 1.3567\ (99.11\%)$$

Finally, we were able to show, using *paired t-tests*, that both coefficients are statistically

different with a confidence of 99%.

After this characterization we can clearly see that *Alexander* indicators are bounded below those from *Hatshepsut*. Then we can infer that $C_{uA} < C_{uH}$. Simplifying the assignment, assuming that we have only two different values, 0 and 5, we can say that $Pm_A = \langle 0, ... \rangle$ and $Pm_H = \langle 5, ... \rangle$, which corresponds to the agents original model.

Our objective here was to show that it is possible to generate models in the discussed representation from data collected during play. We performed the same operations for other two randomly selected preferences (*Growth* and *Gold*) in previous section, and its analyses are also applicable here.

5.4 Overview

In this chapter, we have instantiated the three first steps of the generic approach proposed to model the problem of preference prediction. We first distinguished virtual agent's preferences using gameplay data, showing that these different behaviors can be observed along the time and manually inferred for some preferences, what assisted us defining the features to be used by the ML algorithm we define (the gameplay indicators). We also concluded that the impact of the division between matches won and lost is not so useful, assisting us to understand what are relevant examples, as defined in Chapter 4. Then we presented an evaluation regarding the generic representation, showing its feasibility in the game CIVILIZATION IV.

The next chapter discusses the use of ML to automatically define player's preferences, first discussing the last steps of the instantiation of the approach discussed in the previous chapter: the selection of the ML task, the appropriate algorithm and its best parameters configuration.

Chapter 6

Experimental Results

> It doesn't matter how beautiful your theory is, it doesn't matter how smart you are - if it doesn't agree with the experiment, it's wrong.
>
> Richard Feynman

This chapter presents our experimental results regarding the prediction of CIVILIZATION IV player's preferences. Our approach, which follows a supervised learning framework, is detailed together with the used algorithms and the search method for their best parameters configuration.

Three different experiments were performed: we first evaluated the accuracy of the selected algorithms in the *Traditional Dataset*, composed of known virtual agents. After that, we used the generated models to predict the preferences of virtual agents in the *Alternative Dataset*, composed of virtual agents different from those used in the training phase. Finally, after evaluating the generality of the obtained models in virtual agents, these models are used to predict self-reported human players' preferences.

6.1 Experimental Methodology

Before presenting our experimental methodology, we shall define how we are going to approach the preference modeling problem, the fourth step in the methodology proposed in Chapter 4. We decided to model it as a supervised learning task, once we know the preferences of each artificial player. Additionally, it is easy to ask human players to self-classify themselves. In this approach, the algorithm learns a model by finding relations between a set of features that describe the examples (matches) and a class (preference). For the preference learning problem, each example represents a player turn and the features are the game score indicators, as discussed in the previous chapter.

We decided to model the class as the presence or absence of a given preference (binary

classification), allowing us to make a more precise inference. This decision was based on the assumption presented in Section 4.1.3, where we state that human players may face difficulties when mapping their preferences into different levels. Additionally, as we show in Table 6.1, some preferences do not have more than two values in the original dataset. This approach represents one of the main differences between our work and the one presented in [den Teuling, 2010; Spronck and den Teuling, 2010], which tackled player modeling as a multi-class classification problem. As showed by experimental results, our approach achieves better accuracies.

Regarding the classification algorithms used, the fifth step listed in Chapter 4, our first idea was to use the SVM to make our results directly comparable with those reported in [den Teuling, 2010; Spronck and den Teuling, 2010]. We used a Radial Basis Function (RBF) kernel. Note that, as mentioned, the problem of choosing the best classifier for a learning problem is an open issue in the machine learning literature, tackled by the sub-area of meta-learning [Brazdil et al., 2009].

Besides *SVM*, we also evaluated three different classifiers based in different paradigms, namely *NaiveBayes, AdaBoost* and *JRip*, aiming to compare the performance of these different approaches since, as far as we know, no other work had done this properly in the past. Each classifier has different assumptions about the input data, and generate different models that may explore distinct data characteristics, allowing us to evaluate different approaches for tackling preference modeling.

The first experiment we performed was the application of the four classification algorithms in the *Traditional Dataset*, using a 10-fold cross validation. Hence we used this first result to select the best algorithms to evaluate how the models generated by them generalize. In a second experiment, we used the whole *Traditional Dataset* as training set, and generated models that classified six different artificial players that were never seen by the classifiers (*Alternative Dataset*), then we used the same models to predict human players preferences.

6.1.1 K-Fold Cross Validation

A cross-validation is a method traditionally used in the ML literature to give experiments statistical confidence. It divides the dataset in k different folds (in our case, k = 10), where one fold is separated for test, and the other $k - 1$ are used during the training process. This approach generates k different training and test sets, each test set being different from the other nine. The final results reported correspond to the average error over the k folds, in order to guarantee that the results were not found by chance according to the characteristics of the learning data. When this process is stratified, the folds preserve the class distribution

of the original dataset.

6.1.2 Parameters Optimization and Data Sampling

Once we included the *SVM* algorithm as one of the classifiers, we went to parameter tuning, the last step in the approach proposed in Chapter 4. We sought for the best possible parameters only for *SVM* because it is known to be extremely sensitive to parameters variations. The best parameters configuration for each data fold was obtained using a tool called *easy*, present in the *libSVM* implementation [Chang and Lin, 2011].

In practice, we optimized two different parameters: *cost (c)* and *gamma (g)*. Their meanings were already presented in Chapter 2. Algorithm 1 presents the grid search performed by *easy* and its maximum and minimum values (defined in the tool). This algorithm is applied to each fold of the cross-validation. We are looking for $best\_c$ and $best\_g$. The step values used are also those defined in the tool *easy*.

Algorithm 1 Grid Search

Ensure: Best parameters c and g

```
c ← 0, best_c ← 0
g ← 0, best_g ← 0
best_accuracy ← 0
for c = −5 to 15 with step 2 do
  for g = 3 to -15 with step -2 do
    {Evaluate accuracy with these parameters}
    if Evaluate(2^c, 2^g) > best_accuracy then
      best_c ← c
      best_g ← g
    end if
  end for
end for
return {best_c, best_g}
```

The *Evaluate* function is responsible for running the classifier (training and testing) with the parameters passed. The accuracy is calculated as the number of instances correctly classified divided by the number of instances in the dataset.

As observed in the algorithm, this search is computationally expensive, requiring 6,600[1] different experiments with *SVM*. It was not possible to run these experiments in a feasible time. For this reason, a sampling method was used to reduce the 72,653 vectors available for training, speeding up the learning process. To ease the comparison between the learning algorithms, we used the same sampled dataset to evaluate all algorithms.

[1] 11 from *c* search × 10 from *g* search × 10 from 10-fold × 6 from number of preferences.

Since our dataset contained data from complete matches and their evolution, we sampled the data considering not its vectors, but its matches. In other words, we added or removed complete sets of vectors, with each set representing a whole match. It is important to stress that we sampled just the training set, not the test set. In other words, we first generate a set containing 1/10 of the dataset (due to the 10-fold cross validation). This is the test set. After this step we sample the remaining matches to create the training set. We originally had 240 matches and sampled 25% of their original size, obtaining a test set of 24 matches and a training set with 54 matches[2].

All results reported were obtained by the best parameter configuration chosen by *easy* using the sampling process described above. Note that the sampling process is done with data different from those used for testing the models, and used by all learning algorithms.

Tables 6.1 and 6.2 present the number of examples belonging to each class in the test set of the original dataset (Table 6.1) and the sampled dataset (Table 6.2). The sampling algorithm is showed in Algorithm 2. In the new dataset, we mapped the instances with class 0 as those with class -1, and those with class 2 and 5 as 1.

Table 6.1: *Number of samples in each class for the test sets of the original data in the Traditional and Alternative Dataset as in [den Teuling, 2010] (problem modeled with 3 classes).*

Preferences	Known Agents (*Traditional*)			Unknown Agents (*Alternative*)		
	0	2	5	0	2	5
Culture	11,828	-	6,111	16,330	-	-
Gold	11,937	3,023	2,979	11,412	2,242	2,676
Growth	14,756	3,183	-	4,981	11,349	-
Military	6,337	5,396	6,206	5,454	-	10,876
Religion	11,602	6,337	-	13,951	2,739	-
Science	15,296	-	2,643	13,552	-	2,778

[2] 25% of 216 matches (the result of the total set minus the test set).

Table 6.2: *Number of samples in each class of the test set for the sampled Traditional and Alternative Datasets. Standard deviation showed between parenthesis.*

Preferences	Avg. of Known Agents		Unknown Agents	
	-1	1	-1	1
Culture	4,914.7 (158.2)	2,420.2 (161.6)	16,330	-
Gold	5,037.7 (311.5)	2,464.4 (201.8)	11,412	4,918
Growth	5,821.9 (359.3)	1,304.2 (135.0)	4,981	11,349
Military	2,472.8 (129.5)	4,853.2 (574.7)	5,454	10,876
Religion	4,714.6 (241.5)	2,576.2 (322.6)	13,951	2,739
Science	5,788.5 (601.3)	1,262.0 (82.6)	13,552	2,778

Algorithm 2 Sample Dataset

Input: Sample percentage $perc$ $\{0 \leq perc \leq 1\}$
Array $matches$ containing all matches {Each match contains all its turns}
Output: Array $sampledMatches$ with sampled matches

$sampledMatches \leftarrow \emptyset$
$matchesWithPref \leftarrow \emptyset$
$matchesWithoutPref \leftarrow \emptyset$
for $i = 0$ to $matches.size$ **do**
  {Check if the agent of that match has the preference}
  if $matches[i].preference$ **= true then**
    $matchesWithPref.add(matches[i])$
  else
    $matchesWithoutPref.add(matches[i])$
  end if
end for
$shuffle(matchesWithPref)$
$shuffle(matchesWithoutPref)$
for $i = 0$ to $matchesWithPref.size \times perc$ **do**
  $sampledMatches.add(matchesWithPref[i])$
end for
for $i = 0$ to $matchesWithoutPref.size \times perc$ **do**
  $sampledMatches.add(matchesWithoutPref[i])$
end for
return $sampledMatches$

6.2 Classification of Virtual Agents Preferences

The experimental phase with artificial agents (not human players) was divided in two steps. As already mentioned, we first predicted preferences of virtual agents from the *Traditional Dataset*, using a standard 10-fold cross-validation procedure. Then, in a second experiment, we used the entire *Traditional Dataset* to generate a model that classified the agents in the *Alternative Dataset*. This second experiment was designed to evaluate the generalization capabilities of the models when predicting preferences of unknown agents, since generalization is the most important characteristic to enable the model to be used in games in real situations.

The first experiment used 130 features, including two available at the end of each match, called *match result* and *peace*. This allowed us to perform an *off-line review* (as discussed in Chapter 3). Since we do not have these two features in the *Alternative Set*, with unknown agents, we retrained our dataset removing them, using 128 features to simulate an *on-line tracking* in our second experiment. We divided the results presentation in these two sections.

6.2.1 Off-line Review of Known Agents

We compare the four methods using binary classification (*Binary-Class SVM*, *Naive Bayes*, *AdaBoost* and *JRip*) with the Majority Class and the results reported by Spronck and den Teuling [2010], which we named *Multi-Class SMO*. As we previously discussed, this work can be considered our baseline, since the authors tackled the same problem of preference modeling in the game CIVILIZATION IV. The experimental results are reported in Table 6.3.

The *Majority Class* corresponds to the percentage of the most frequent class of the dataset. For example, for the *Culture* preference, 67.0% of the turns were played by agents with no preference for *Culture*. This means that if the classifier learned nothing and generated a model classifying every turn as "without preference", it would obtain the reported accuracy. The column *Multi-Class SMO* presents the results reported in [den Teuling, 2010; Spronck and den Teuling, 2010]. They modeled the problem as a multi-class classification, *i.e.*, instead of modeling preferences as existent or not, they modeled three levels of preference. This result is shown just for a high-level comparison, since the baseline did not performed cross-validation on its experiments, making a more detailed comparison meaningless.

We tried to reproduce the results reported in [den Teuling, 2010; Spronck and den Teuling, 2010] but we obtained a different number than those reported, hence we were not able to run a cross-validation in the baseline's approach to compare theirs with our approach, in this

first experiment. Thus, in Table 6.4, we use the *Majority Class* as baseline, only presenting the improvement of our approach over it. We understand that it is not an ideal baseline and, we only use it to evaluate whether the applied algorithms are learning "something".

First analyzing the results generated by the four algorithms we ran using the binary classification approach, we can see that the unique algorithm that performed better than a possible *Majority Class* approach for all preferences was *JRip*. Surprisingly, *SVM* did not present a good performance when compared to *AdaBoost* and *JRip*, even demanding considerably more time to be trained with good parameters.

It is interesting to observe that the accuracies corroborate most of the discussion presented in Chapter 5, since the worse improvements occurred for the *Gold* and *Growth* preferences. The only improvement obtained for these preferences that was greater than zero was 0.7%, using *JRip*. These results may be explained by the importance of gold in the game (hence all players somehow prioritize it) and the absence of the first 100 turns in the dataset. As we have shown in the last chapter, the first turns are very important to characterize *Growth*. We kept the dataset without the first 100 turns to be coherent with [den Teuling, 2010; Spronck and den Teuling, 2010]. A future study should further evaluate the impact of removing these first 100 turns, using the complete dataset in experiments similar to those we performed.

Let us now analyze *Military* and *Religion*, which we were able to correctly classify all instances (accuracy of 100% using *SVM, AdaBoost* and *JRip*, generating an improvement greater than 50%). These happened because two features available in the end of the game, *match result* and *peace*, are able to differ *Military* and *Religion* preferences by themselves. They are only available when performing off-line review. Since we do not have this information in the second dataset, we retrained our model without them to classify the unknown agents. These results will be presented in Section 6.2.2.

Finally, we compared the accuracies obtained by different classifiers. We performed paired t-tests with 95% of confidence to present them. The unique preference in which *SVM* presents a higher accuracy, when compared to the other methods, is *Culture* (78.9% while the second higher is 69.2%). For this preference, we may obtain the following ordering: *Naive Bayes* $<$ *AdaBoost* $<$ *JRip* $<$ *SVM*, with 95% of confidence.

For all other preferences, *JRip* and *AdaBoost* presented the higher accuracies. *SVM* presented accuracies similar to the higher ones for some preferences, while similar to *Naive Bayes* in others. We decided not to use *SVM* in the next experiment, since its results vary a lot from one preference to another, its running time is costly and, most importantly, its results are statistically inferior to those obtained by *JRip* and *AdaBoost* for most preferences.

Table 6.3: *Accuracy of our methods (Binary-SMO, Naive Bayes, AdaBoost and JRip) contrasted with the most frequent class (Majority Class) and with Spronck and den Teuling [2010]'s approach (Multi-Class SMO). The results are the average accuracy of 10-folds. The Root Mean Squared Error (RMSE) is shown in parenthesis.*

Preference	Majority Class	Multi-Class SMO	Binary-Class SVM	Naive Bayes	AdaBoost	JRip
Culture	67.0%	78.9% (0.46)	73.1% (3.08)	67.1% (2.99)	66.5% (5.20)	69.2% (4.66)
Gold	67.2%	74.6% (0.38)	63.9% (7.41)	62.6% (3.04)	64.0% (5.51)	67.6% (5.14)
Growth	81.7%	83.5% (0.41)	78.0% (4.27)	76.1% (3.55)	83.1% (3.52)	82.3% (2.79)
Military	66.1%	61.0% (0.43)	100.0% (0.00)	84.4% (2.58)	100.0% (0.00)	100.0% (0.00)
Religion	64.8%	79.0% (0.46)	100.0% (0.00)	84.9% (2.93)	100.0% (0.00)	100.0% (0.00)
Science	82.0%	88.4% (0.34)	81.0% (7.24)	79.1% (1.14)	83.4% (3.07)	84.8% (4.02)

Table 6.4: *Improvement of each approach over Majority Class. Since our approach has a different number of classes, it is not fair to evaluate our improvement over Spronck and den Teuling [2010]'s approach. The improvement is computed as the difference between the accuracy and the baseline divided by the baseline.*

Preference	Binary-Class SMO	Naive Bayes	AdaBoost	JRip
Culture	9.0% ▲	0.1% ▲	-0.7% ▼	3.3% ▲
Gold	-4.9% ▼	-6.8% ▼	-4.8% ▼	0.7% ▲
Growth	-4.5% ▼	-6.9% ▼	1.7% ▲	0.7% ▲
Military	51.3% ▲	27.7% ▲	51.3% ▲	51.3% ▲
Religion	54.3% ▲	31.0% ▲	54.3% ▲	54.3% ▲
Science	-1.2% ▼	-3.5% ▼	1.7% ▲	3.4% ▲

To conclude this first discussion, it is interesting to stress some observed behaviors. A first conclusion that can be drawn is that *SVM* seems to not be the best approach for preference modeling, despite the preference given to it by researchers in the area, such as [Spronck and den Teuling, 2010; Machado et al., 2012a]. Regarding *Naive Bayes*, we were able to see that in three preferences it is as good as *SVM*, despite running instantly while *SVM* may take days to classify agents. Additionally, *Naive Bayes* presented the lowest variation between classifications (no RMSE higher than 4.0), an indicative that it may be a more stable method to this problem. *JRip* may seem an interesting approach, because it does not only present the best performance overall, with all accuracies higher than 65%, but it also gives us rules that allow game designers to understand and "debug" a specific classification. These rules may be even intuitive, such as a learned rule that says that: "If you have preference for *Gold* then you will not be at war".

6.2.2 Online Tracking of Unknown Agents

To conclude our experiments regarding virtual agents, we evaluated how general the learned models are. In this case, we used all instances of the first dataset as a training set, and the *Alternative Dataset* as test set, composed of agents that were not used to generate the matches in the training set. Recall the class distribution of all preferences is presented in Table 6.1. As previously explained, due to the low *SVM* performance and its high computational cost, we did not use it in this second experiment. The results are presented in Table 6.5 and the improvement over the baseline in Table 6.6.

Table 6.5: *Accuracy of the evaluated approaches (Naive Bayes, AdaBoost and JRip) contrasted with the most frequent class (Majority Class) and with Spronck and den Teuling [2010]'s approach (Multi-Class SMO) for unknown agents, not seen in the training process. No error is shown since we just executed this classification once.*

Preference	Majority Class	Multi-Class SMO	Naive Bayes	AdaBoost	JRip
Culture	100.0%	88.2% (0.34)	74.8%	97.7%	96.5%
Gold	69.9%	38.6% (0.50)	50.4%	68.1%	62.5%
Growth	69.5%	30.8% (0.83)	35.1%	30.5%	33.1%
Military	66.6%	34.6% (0.56)	51.7%	66.6%	59.3%
Religion	83.2%	59.0% (0.64)	61.3%	83.2%	73.0%
Science	83.0%	71.0% (0.54)	67.7%	83.0%	81.5%

Table 6.6: *Improvement of each approach over the baseline [den Teuling, 2010; Spronck and den Teuling, 2010] when predicting unknown virtual agents. The improvement is computed as the difference between the accuracy and the baseline divided by the baseline.*

Preference	Naive Bayes	AdaBoost	JRip
Culture	-15.2% ▼	10.8% ▲	9.4% ▲
Gold	30.6% ▲	76.4% ▲	61.9% ▲
Growth	14.0% ▲	-1.0% ▼	7.5% ▲
Military	49.4% ▲	92.5% ▲	71.4% ▲
Religion	3.9% ▲	41.0% ▲	23.7% ▲
Science	-4.6% ▼	16.9% ▲	14.8% ▲

A first interesting result to observe is that the used algorithms, with the binary classification approach, surpass den Teuling [2010]'s (*SVM*) in accuracy, corroborating our assumption that *SVM*, despite being considered the state-of-the-art for several classification problems, may not be adequate for modeling player preferences.

AdaBoost and *JRip* performances were remarkable, mainly when compared to our baseline. For most of the preferences, these methods were able to obtain accuracies above 60%, such as 68.1% (*AdaBoost*) and 62.5% (*JRip*) against 38.6% of accuracy obtained by the baseline when predicting the *Gold* preference; and 66.6% (*AdaBoost*) and 59.3% (*JRip*) against 34.6% of the baseline predicting the *Military* preference.

AdaBoost seems to have learned to classify every instance as being of the most frequent class and this approach will be further evaluated in the next section, when we use the same models to classify human players preferences. *JRip* presented slightly worse results when compared to *AdaBoost*, but it still presented very good accuracies, such as 96.5% for *Culture*, 73.0% for *Religion* and *81.5%* for *Science*. Additionally, as already discussed, this method has the advantage of generating comprehensive rules, which can be verified by a game designer or AI programmer.

After removing the *Peace* and *Victory Type* features, *JRip* started to generate longer rules that, as we can observe, were able to generalize well. As an example, some rules that were able to correctly classify some preferences are (correctly/incorrectly classified instances between parenthesis):

- **Culture**: *CitiesDiff = -1* $\wedge$ *CitiesTrend = 5* $\wedge$ *CumulativeWar = 20* $\wedge$ *War = 0* $\rightarrow$ *Culture Preference* (722/0);

- **Military**: *CumulativeDeclaredWar = 0* $\wedge$ *StateReligionDiff = 0* $\wedge$ *CumulativeWar = 12* $\wedge$ *War = 0* $\rightarrow$ *No Military Preference* (309/0);

- **Religion**: *CumulativeDeclaredWar = 0* $\wedge$ *CumulativeWar = 35* $\wedge$ *War = 0* $\rightarrow$ *Religion Preference* (570/0);

- **Science**: *CitiesDiff = 3* $\wedge$ *CumulativeWar = 77* $\wedge$ *Cities = 8* $\rightarrow$ *Science Preference* (182/0);

These rules already give us a glimpse about virtual agents behaviors and preferences, such as the importance of wars and number of cities to define a player preference. In fact, we can even observe very sound rules, such as a lower number of war declarations for virtual agents with preference for *Culture* and *Religion*. The listed rule that detects no *Military* preference is also very intuitive, corroborating our discussion about *JRip* benefits.

Naive Bayes presented the worst performance among the evaluated algorithms, despite being better than the baseline in four different preferences. This poor performance may be explained by *Naive Bayes* assumption of independence between features. This independence assumption does not hold in our problem, since the turn number is a feature and the score features are directly related to the game turn.

The unique preference to which our methods performed poorly was *Growth*, with no accuracy greater than 35%. We believe this is mainly due to the removal of the first 100 turns of each player. As we already showed in Chapter 5, these turns are extremely relevant to differ artificial players preference for *Growth*, while the later turns may be misleading.

6.3 Player Modeling

After presenting an approach to classify virtual agents preferences that was able to surpass previous approaches in the literature, it is important to test our method with "real" data. In this last section we discuss how we gathered data from human players and the performance of the algorithms, already mentioned, when applied to these data.

6.3.1 Players' Data

Aiming at evaluating our approach in data generated by human players, we recruited volunteers to play the game CIVILIZATION IV. They were required to add an script in the game directory in order to sniffer their scores along a match. This script is called *AIAutoPlay* and it was modified and used by den Teuling [2010] to generate the datasets with virtual agents. We also used it in this thesis, as previously discussed. Its installation only consists in replacing some *dlls* of the original game. It is important to stress that, at the end, we had the same 128 features used in the *online tracking* experiment.

We did not make any restrictions to players regarding their experience or any other characteristic. We required them to play an 1x1 match and to log the game. Before playing, we requested them to sign a consent form and to fill a pre-test questionnaire about their experience in TBS games and, more specifically, games of the CIVILIZATION series. After filling this questionnaire, they played the game and then filled a post-test questionnaire informing their self-labeled preference in the played game and evaluating their confidence in this self-labeling. These questionnaires were written in Portuguese and are available in Appendix B.

Seven players participated in the tests and sent us their data. We judge this number satisfactory due to the difficult to obtain players. A *Civilization IV* game may take longer than four hours and few people accepted to participate in such test.

For usability tests, for example, it is said that three to five users are enough to perform an experiment [Nielsen and Landauer, 1993].

Table 6.7 contains the classes distribution in our dataset, while Figure 6.1 shows the players' self-reported information about the frequency they play(ed) TBS games and, more specifically, games of the series CIVILIZATION.

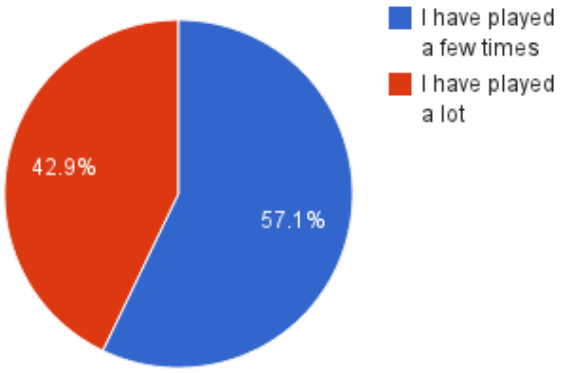
Table 6.7: *Number of samples in each class for the dataset generated from matches played by human beings. As all other experiments, we removed the first 100 turns of each match.*

Preferences	-1	1
Culture	1,725	100
Gold	1,466	359
Growth	647	1,178
Military	1,436	389
Religion	848	977
Science	29	1,796

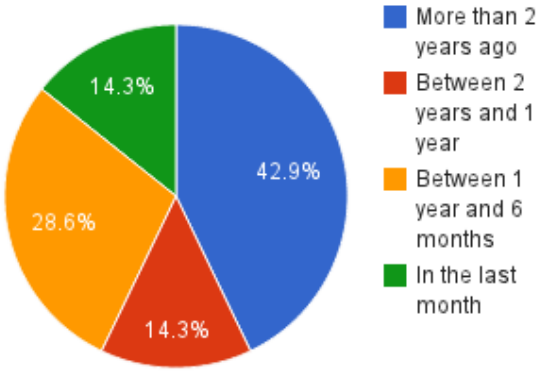
We can see that there are players who consider themselves "good" and those who have never played the game. We believe to have an interesting range to perform our experiments, mainly because den Teuling [2010] suggested that experienced players are easier to be classified than those without experience.

6.3.2 Classification of Players' Preference

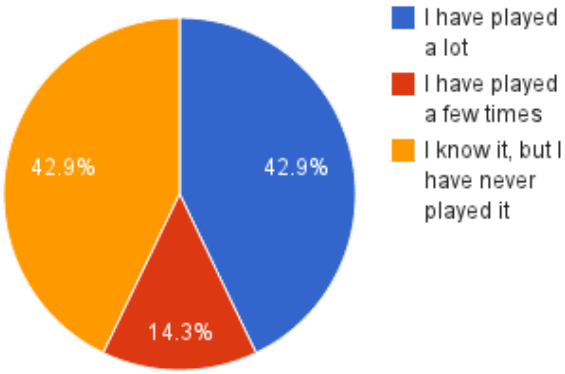
To classify human players using supervised learning, a large dataset containing labeled samples is required. This is unfeasible in practice because no player can provide so much data



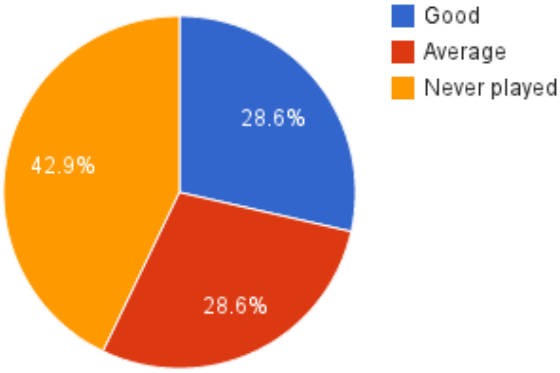
(a) *Have you ever played Turn Based Strategy games?*



(b) *When did you play a turn-based strategy game for the last time?*



(c) *More specifically, have you ever played any game of the Civilization IV series?*



(d) *How do you classify your ability as a Civilization player?*

Figure 6.1: *Players' experience in TBS and* CIVILIZATION *games.*

in a short period of time, thus another approach must be used. We tackle this problem using data generated from AI $\times$ AI matches to train each classifier. This one of the reasons to perform the *online tracking* experiment in last section, to evaluate the feasibility of this approach.

We use the same training set used in the *online tracking* experiment, containing all matches of the *Traditional Dataset*, to train the models that classify players' matches. In fact, we used those already generated. The results of the classification are presented in Table 6.8. We decided not to report den Teuling [2010] results regarding classification of human players because they performed experiments with only two players, using a different modeling and algorithm.

Observing the results of all seven players, we notice that, among all classifiers, at least one of them obtained an accuracy above 50% for each preference. *Naive Bayes* obtained accuracies over 50% for *Growth* (62.7%), *Military* (61.7%) and *Science* (89.7%) while *AdaBoost* and *JRip* obtained accuracies over 50% for *Culture* (91.5%, 94.5%), *Gold* (74.5%, 74.5%) and *Religion* (60.7%, 54.5%).

JRip seems to have generated rules that classify most turns in the most frequent class for the preference *Culture*, and to the most infrequent class for the preferences *Growth* and *Science*. As we already discussed, the *Growth* preference has a particularity that we removed its most discriminative turns: the first 100. This may have impaired the learning algorithm when dealing to the *Growth* preference, what was constistent with all other obtained results.

Despite presenting a bad performance for some preferences, different from other classifiers, *JRip* allows improvement with assistance of human designers, since its generated model is comprehensible. Maybe the reported results would have been improved if an expert, such as the game designer, analyzed the generated rules and decided which ones to use, or if a specific rule is classifying a behavior in a wrong class.

Table 6.8: *Accuracy of our approaches (Naive Bayes, AdaBoost and JRip) for classifying human players' preferences. No error is shown since we just executed this classification once.*

Preference	Majority Class	Naive Bayes	AdaBoost	JRip
Culture	94.5%	12.8%	91.5%	94.5%
Gold	80.3%	35.9%	74.5%	74.5%
Growth	64.6%	62.7%	35.5%	36.3%
Military	78.7%	61.7%	21.3%	25.7%
Religion	53.5%	25.3%	60.7%	54.5%
Science	98.4%	89.7%	1.6%	1.6%

Naive Bayes presented the worse results in average. However, it is interesting to point out that it succeeded where more complex algorithms did not. This may be an indicative that some preferences may be biased, causing overfitting in more complex algorithms. Maybe *Naive Bayes* was able to avoid this overfitting due to its simpler paradigm. It was the unique algorithm that presented good results for the preferences *Growth, Military* and *Science*.

As previously mentioned, *AdaBoost* learned to classify all instances in a specific class. It happened in the previous experiment (*online tracking*) and also here. Its results were good for half of the preferences because it correctly selected the most frequent class for them, while it did not to the other half.

Interestingly, *Naive Bayes* succeeded in three different preferences (*Growth, Military* and *Science*) while *AdaBoost* and *JRip* presented an accuracy over 50% in the other three (*Culture, Gold* and *Religion*). Further studies must be performed in order to assure that one specific classifier is better and always obtains better results for a specific preference. If it is not true, a hybrid approach considering different classifiers for different preferences may lead to a higher overall accuracy . Nevertheless, the first thing to evaluate is whether these results may be due to players wrongly self-reporting their preferences. We will present this discussion in sequence.

6.3.2.1 Evaluating Players' Preference Considering their Expertise

The post-test questionnaire asked users to list their preferences in the played match, and their confidence on the provided list. Based on the pretest questionnaire and in the question about player's confidence, we clustered players in two different groups:

- Experienced: Three players are part of this group, those who had already played a game of the CIVILIZATION series "a lot", and that stated to have a high confidence in their self-labeling.
- Beginners: This group contains 4 players, those who played, in the best case, few times a game of the CIVILIZATION series.

This experiment was designed to evaluate whether the results were impaired by inexperienced players who did not label themselves "properly", i.e. as experienced players, with a great knowledge of the game, would do. In other words, we evaluate the assumption whether the generated model is correctly identifying preferences, but human players do not have a common understanding about these preferences definition. It is based on the assumption that we have sufficient data to represent both types of players in our training set.

We classified both groups separately, and its accuracies are presented in Tables 6.9 and 6.10. These results confirm our assumptions about *AdaBoost*. The algorithm was not able to learn a model different from classifying all instances in a class, and for most of the results, in Tables 6.9 and 6.10, we can see that it either obtained an accuracy close to the majority class or it obtained an accuracy close to 100% minus the majority class (when it chose the wrong class). Due to that, the *AdaBoost* performance in the two groups (experienced and beginners) should not be considered, since it is much more a matter of luck than any other topic.

Just as the first experiment, we can observe that *JRip* presents a good accuracy for *Culture, Gold* and *Religion*, while *NaiveBayes* obtained good accuracies classifying *Growth, Military* and *Science*.

Table 6.9: *Accuracy of the evaluated approaches (Naive Bayes, AdaBoost and JRip) for classifying the preferences of experienced human players. No error is shown since we just executed this classification once.*

Preference	Majority Class	Naive Bayes	AdaBoost	JRip
Culture	86.1%	15.5%	86.1%	85.9%
Gold	100.0 %	8.5%	99.6%	80.9%
Growth	63.9%	65.6%	36.1%	40.5%
Military	100.0 %	96.2%	0.0%	11.4%
Religion	63.9%	3.8%	100.0%	83.7%
Science	100.0 %	97.5%	0.0%	0.0%

Table 6.10: *Accuracy of the evaluated approaches (Naive Bayes, AdaBoost and JRip) for classifying the preference of beginner human players. No error is shown since we just executed this classification once.*

Preference	Majority Class	Naive Bayes	AdaBoost	JRip
Culture	100.0 %	11.0%	94.9%	100.0%
Gold	67.6%	53.7%	58.2%	70.3%
Growth	65.0%	60.8%	35.0%	33.6%
Military	64.9%	39.3%	35.1%	35.0%
Religion	64.9%	39.3%	35.1%	35.5%
Science	97.4%	84.6%	2.6%	2.6%

Regarding the division between player's trustfulness, we observe that there is not a consistent pattern when classifying the two groups. Differently from den Teuling [2010], we did not obtained higher accuracies for experienced players. *JRip*, for example, had half of its results improved in the experienced group (*Gold, Growth* and *Religion*) and the other half reduced (*Culture, Military* and *Science*).

Observing the results of *NaiveBayes*, we notice that it leads to higher accuracy variation. The algorithm obtained an accuracy, when classifying beginners' *Gold* preference, equals to 53.7%, while it obtained only 8.5% of accuracy for experienced players. A similar variation occurred for *Religion* (39.3% against 3.8%) and *Military* (39.3% against 96.2%).

Due to these results, we can conclude that, in this experiment, the separation between player's experience, or confidence in their self-labeled preferences, does not consistently improves classifiers' accuracy. These results contradict the assumption presented by den Teuling [2010] that experienced player are "easier" to be classified. Table D.1, at Appendix D,

presents the accuracies obtained classifying each individual player, and its experience level (classified by us using their answers in the questionnaires). We can see that there are experienced players who are not correctly classified while there are beginners who are. Thus, further investigation is necessary.

6.4 Overview about Modeling Players using ML

After we executed experiments classifying virtual agents and human players, we are able to perform a higher level discussion about the machine learning approach applied to the *player modeling* problem. As an overview, we can say that we were able to generate models, from virtual agents, that are generic enough to classify other virtual agents with a satisfactory accuracy. However, the same generated models were not always effective when classifying human players, what may be justified by virtual agents behaviors not generalizing to human players.

Despite our efforts to validate our assumption that matches played between virtual agents are capable to generate useful examples for the classifier, we must consider the possibility that the generated data may not be sufficient to determine the classifier, *overfitting* it. This may be the reason for our consistent poor performance when classifying the *Growth* preference, as well as our results when classifying human players' *Military* and *Science* preferences using *JRip*.

To better understand this topic, we quote Domingos [2012]:

> "One way to understand overfitting is by decomposing generalization error into *bias* and *variance* [Domingos, 2000]. Bias is a learner's tendency to consistently learn the same wrong thing. Variance is the tendency to learn random things irrespective of the real signal. "

When decomposing overfitting in these two topics, we observe the need to further evaluate both. Regarding bias, it may be possible that the classifiers we used did not assume the correct frontier shape of the CIVILIZATION IV *player modeling* problem. This may cause a poor performance of the classifier when predicting preferences that are not structured as it assumes. This may also justify the better performance of the *NaiveBayes* classifier when modeling *Gold, Growth* and *Science*, since it is known that simpler algorithms may yield better solutions in complex spaces, if more sophisticated algorithms are not properly tunned [Domingos, 2012]. It is also important to stress that *NaiveBayes* assumes independence between features, different from *JRip*, for example. This assumption may also have an impact when classifying different preferences.

This problem is a complex topic that must be further studied. Not surprisingly, we were able to show that, as it seems, the algorithm that was considered the state of the art for several classification tasks, and more specifically to this problem [Spronck and den Teuling, 2010], was surpassed by those we used in our methodology.

Related to variance, "After overfitting, the biggest problem in machine learning is the *curse of dimensionality* (...) Generalizing correctly becomes exponentially harder as the dimensionality (number of features) of the examples grows (...)" [Domingos, 2012]. This may be the second problem with ML approaches. In this thesis, we have shown that a subset of our features was useful to be used by a classifier. However, we added many other features by intuition. This is a common flaw in *player modeling* papers that may lead to worse results. Domingos [2012] also discusses it:

> "Naively, one might think that gathering more features never hurts, since at worst they provide no new information about the class. But in fact their benefits may be outweighed by the curse of dimensionality."

It is clear that the features used are also extremely important when performing an ML task, specifically to our problem, when *modeling players*. As far as we know, this is an unexplored topic in the field, mainly because games present several features and its combination may provide useful information. This justifies why automate feature engineering does not solve the problem, since selecting features by information gain, for example, may be useless once it will ignore relations between features. This is why we performed the extensive analysis in Chapter 5, relating turns with game score indicators, and analyzing its meaning in the game.

Chapter 7

Conclusion

> "All right," said Deep Thought. "The Answer to the Great Question..."
> "Yes..!"
> "Of Life, the Universe and Everything..." said Deep Thought.
> "Yes...!"
> "Is..." said Deep Thought, and paused.
> (...)
> "Forty-two," said Deep Thought, with infinite majesty and calm.
>
> Douglas Adams, The Hitchhiker's Guide to the Galaxy

This chapter presents the conclusions of this thesis. In the next section, we summarize our main contributions, present some considerations regarding our results and discuss some limitations of our work. Section 7.2 lists some possible extensions of this research.

7.1 Contributions and Discussion

In this work we have presented an extensive discussion about player modeling. Among our main contributions, and its correspondent publications, we can list:

- The organization of a taxonomy to organize the field [Machado et al., 2011a];
- The proposal of a generic approach to tackle player modeling as a machine learning problem [Machado et al., 2012a];
- The evaluation of a generic representation that may be used to model players in different games [Machado et al., 2012b];
- The proposition of an approach based on linear regressions to evaluate features' ability to distinguish different classes [Machado et al., 2011b]; and

- The evaluation of different classifiers, regarding its accuracy and generalization capabilities, when applied to virtual agents and human players.

Domingos [2012] states that "the fundamental goal of machine learning is to generalize beyond the examples in the training set", and we were able to do this for most of the preferences, what indicates the effectiveness of the approach proposed here. When comparing all the evaluated classifiers, we can conclude that there are two main choices that may be done for future applications regarding player modeling: one may choose *NaiveBayes* as a simpler and faster approach to the problem, but which may lead to worse performances; or *JRip*, which is a method that obtains higher accuracies and generates rules that may be evaluated by game designers, despite being a more computational expensive method.

In spite of being able to surpass, sometimes by far, the state of the art of player's preferences classification [den Teuling, 2010; Spronck and den Teuling, 2010], it became evident that player modeling is a very hard task. We were not able to obtain, with a single classifier, good accuracies for all preferences modeled, and this leads us to the challenges of the field. A first obvious challenge is the difficult to label human preferences. In spite of not further discussing this topic in the last chapter, sometimes it is extremely hard to know, even analyzing by hand, whether a player has or has not a preference for a given game feature. It is even harder to ask him to label himself, since different comparison basis may exist between those who design and those who play the game. It is also important to stress that it may be difficult to a player distinguish between what are the preferences he would like to have and what are the preferences he expresses in the game.

Another challenge in this field is the difficulty to obtain gameplay data from human players. In this thesis, we used matches played by AI agents in order to generate a training set. This approach may be deceptive since the assumption that the training data is similar to the data that will be classified is weaker than in other applications of ML.

As we discussed in last chapter, the selection of the appropriate classifier to a specific task is a hard topic. We have shown that a method that is considered the state of the art for several classification problems (*SVM*) did not perform well in this problem, and we cannot guarantee that another, different from those tested, will not perform better. We selected methods based on different assumptions to evaluate the impact of their differences on this type of problem.

At this point it is important to discuss the main limitations of our work. We address some of them as future work, in the next section. A specific characteristic that we did not explore extensively in our approach was the fact that each feature we used was dependent of its previous values, and that they represented, in fact, the story of a match. We assumed each turn as independent, and classified them separately, sometimes labeling different turns of a

same match with different preferences. We are aware that a player may change its preference along a match, but we believe the reported results are more likely to indicate that the set of turns should be taken as an evidence. We tried to do this adding temporal features such as derivatives, but it is not clear whether it is enough.

Finally, despite performing evaluations about the representativeness of some features, we assumed that all the others available would also be representative and we used all of them in our classifiers. This contradicts what Domingos [2012] says, that "one might think that gathering more features never hurts, since at worst they provide no new information about the class. But in fact their benefits may be outweighed by the curse of dimensionality". So a different approach should be evaluated in the future.

7.2 Future Work

There are several paths for future work. Regarding the proposed taxonomy, a more quantitative work can be developed to analyze the current status of the field, classifying more papers with the proposed taxonomy and looking for trends. Also, a complete new analysis can be done in terms of learning algorithms, answering questions as "what can we learn in games?". We could, for instance, correlate the main learning techniques to the main problems of the field.

Related to the generic representation discussed in Chapter 4, a possible future study is the impact of its application in the game development process, evaluating its benefits for level designers and programmers. We could also analyze the application of hierarchical representations. This topic was proposed by Houlette [2003], which suggests the generation of weights that represent very low level actions, such as *throw grenade* or *use rifle* and a hierarchical organization that extracts higher level information as the combination of its leaf nodes. Higher level representations could be *aggression* or *intelligence*, for example. A promising evaluation research on this topic is the evaluation of the impact of each preference on game indicators, as well as the impact of the interaction between preferences. Additionally, we could also correlate the victory types and players' preferences, aiming an *off-line review*; this could also assist us in our discussing regarding the impact of match results in the classification.

Finally, related to the main topic of this work, the automatic classification of player's preferences, several tasks can be performed. Regarding the evaluation of features, we want to perform different data analyses, which could assist us in the task of selecting appropriate features to the classifiers. Some interesting approaches are: to evaluate the impact of using the first 100 turns in each preference classification and to apply some feature selection methods

to the dataset, maybe trying to consider indicators' semantics. As we already discussed, we believe that the absence of the first 100 turns was determinant in the poor accuracy obtained when classifying *Growth* preference. The feature selection problem is a hard topic since, as discussed, a feature by itself may be meaningless (such as turn), but when associated with others, it may be very useful. In fact, there are several possibilities to further evaluate this topic and, as far as we know, few works have presented concerns in automatically selecting the smaller and most representative set of features for modeling players.

Additionally, our results may be improved by the application of over/undersampling techniques in our dataset, what could deal with unbalanced classes. Another topic that we did not discuss, but may also worth investigation due to its applicability in the whole field, is the evaluation of the correctness of players' self labeled preferences.

A last possible future work, and most ambitious, is the use (or proposal) of a technique that considers each turn as an intermediate state of the player. Instead of classifying each player turn, the algorithm would understand each turn as a stream and would classify a player after a given number of turns. These turns would be responsible for increasing the algorithm confidence that a player has a specific preference. A work that used this approach is Bard and Bowling [2007], who used particle filtering to model POKER players. However, their approach was much simpler than the one that would be required here.

In summary, we have made several contributions to the field, such as the taxonomy, the evaluation of a generic representation and the study of appropriate features to model players, as well as the evaluation of several different classifiers, discussing its possible use in industry. In fact, we believe that this thesis highlighted the specific need of better evaluating ML decisions that are generally made without much care, such the selection of features and classifiers. We consider that player modeling is still an open topic that deserves lots of investigation.

Bibliography

2K Games (2005). Civilization IV Manual.

Aha, D. W., Molineaux, M., and Ponsen, M. J. V. (2005). Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. In *Proceedings of the 6th International Conference on Case-Based Reasoning*, ICCBR, Lecture Notes in Computer Science, pages 5–20. Springer.

Aiolli, F. and Palazzi, C. (2008). *Enhancing Artificial Intelligence in Games by Learning the Opponent's Playing Style*, volume 279 of *IFIP International Federation for Information Processing*, pages 1–10. Springer Boston.

Aiolli, F. and Palazzi, C. E. (2009). Enhancing Artificial Intelligence on a Real Mobile Game. *International Journal of Computer Games Technology*, 2009(1):1–9.

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2nd edition.

Bakkes, S. C., Spronck, P. H., and van Lankveld, G. (2012). Player Behavioural Modelling for Video Games. *Entertainment Computing*, 3(3):71–79. Games and AI.

Bakkes, S. C. J., Spronck, P., and van den Herik, J. (2009). Rapid and Reliable Adaptation of Video Game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):93–104.

Bard, N. and Bowling, M. (2007). Particle Filtering for Dynamic Agent Modelling in Simplified Poker. In *Proceedings of the 22nd Conference on Artificial Intelligence*, AAAI, pages 515–521. AAAI Press.

Billings, D. (2000). Thoughts on RoShamBo. *International Computer Games Association Journal*, 23(1):3–8.

Billings, D. (2006). *Algorithms and Assessment in Computer Poker*. Ph.D. Thesis, Department of Computing Science, University of Alberta, Edmonton, AB, Canada.

Billings, D., Schaeffer, J., and Szafron, D. (1998). Opponent Modeling in Poker. In *Proceedings of the 15th National Conference on Artificial Intelligence*, AAAI, pages 493–499. AAAI Press.

Bohil, C. and Biocca, F. (2007). Cognitive Modeling of Video Game Players. Technical report, East Lansing, MI, USA.

Brazdil, P., Giraud-Carrier, C., Soares, C., and Vilalta, R. (2009). *Metalearning: Applications to Data Mining*. Cognitive Technologies. Springer, 1st edition.

C. J. Darken, B. G. A. (2008). *Particle Filters and Simulacra for More Realistic Opponent Tracking*, pages 419–428. Charles River Media.

Cardona-Rivera, R. E. and Young, R. M. (2012). Characterizing Gameplay in a Player Model of Game Story Comprehension. In *Proceedings of the 7th International Conference on the Foundations of Digital Games*, FDG, pages 204–211. ACM.

Carmel, D. and Markovitch, S. (1993). Learning Models of Opponent's Strategy in Game Playing. In *Proceedings of the AAAI Fall Symposium on Games: Planing and Learning*, pages 140–147. AAAI Press.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1--27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Charles, D. and Black, M. (2004). Dynamic Player Modelling: A Framework for Player-Centered Digital Games. In *Proceedings of the 5th International Conference on Computer Games: Artificial Intelligence, Design and Education*, CGAIDE, pages 29–35. University of Wolverhampton School of Computing.

Charles, D., Kerr, A., McNeill, M., McAlister, M., Black, M., Kcklich, J., Moore, A., and Stringer, K. (2005). Player-Centred Game Design: Player Modelling and Adaptive Digital Games. In *Proceedings of the 2nd Digital Games Research Conference*, DiGRA.

Cohen, W. W. (1995). Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning*, ICML, pages 115–123. Morgan Kaufmann.

Davidson, A., Billings, D., Schaeffer, J., and Szafron, D. (2000). Improved Opponent Modeling in Poker. In *Proceedings of the 2nd International Conference on Artificial Intelligence*, ICAI, pages 493–499. AAAI Press.

de Freitas Cunha, R. L., Machado, M. C., and Chaimowicz, L. (2013). RTSMate: Towards and Advice System for RTS Games. *Computers in Entertainment*. In Press.

den Teuling, F. (2010). *Player Modelling in Civilization IV*. M.S. Thesis, Faculty of Humanities of Tilburg University, Tilburg, Netherlands.

Doirado, E. and Carlos Martinho, C. (2010). I Mean It!: Detecting User Intentions to Create Believable Behaviour for Virtual Agents in Games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 83–90. International Foundation for Autonomous Agents and Multiagent Systems.

Domingos, P. (2000). A Unifeid Bias-Variance Decomposition and its Applications. In *Proceedings of the 17th International Conference on Machine Learning*, ICML, pages 231–238. Morgan Kaufmann.

Domingos, P. (2012). A Few Useful Things to Know about Machine Learning. *Communications of the ACM*, 55(10):78–87.

Dormans, J. and Bakkes, S. (2011). Generating Missions and Spaces for Adaptable Play Experiences. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):216–228.

Drachen, A., Canossa, A., and Yannakakis, G. N. (2009). Player Modeling using Self-Organization in Tomb Raider: Underworld. In *Proceedings of the 5th International Conference on Computational Intelligence and Games*, CIG, pages 1–8. IEEE Press.

Drachen, A., Sifa, R., Bauckhage, C., and Thurau, C. (2012). Guns, Swords and Data: Clustering of Player Behavior in Computer Games in the Wild. In *Proceedings of the 8th International Conference on Computational Intelligence and Games*, CIG, pages 163–170. IEEE Press.

Egnor, D. (2000). Iocaine Powder. *International Computer Games Association Journal*, 23(1):33–35.

Fairclough, C., Fagan, M., Mac Namee, B., and Cunningham, P. (2001). Research Directions for AI in Computer Games. In *Proceedings of the 12th Irish Conference on Artificial Intelligence and Cognitive Science*, AICS, pages 333–344.

Freund, Y. and Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, ICML, pages 148–156. Morgan Kaufmann.

Gow, J., Baumgarten, R., Cairns, P. A., Colton, S., and Miller, P. (2012). Unsupervised Modeling of Player Style With LDA. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3):152–166.

Hladky, S. and Bulitko, V. (2008). An Evaluation of Models for Predicting Opponent Positions in First-Person Shooter Video Games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, CIG, pages 39–46. IEEE Press.

Houlette, R. (2003). *Player Modeling for Adaptive Games*, pages 557–566. Charles River Media.

Iida, H., Uiterwijk, J., Herik, H. v. d., and Herschberg, I. (1993). Potential Applications of Opponent-Model Search (Part 1: The Domain of Applicability). *International Computer-Chess Association Journal*, 16(4):201–208.

Isla, D. (2005). Handling Complexity in the Halo 2 AI. In *Proceedings of the Game Developers Conference*, GDC.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley, 1st edition.

Kendall, G. (2005). Iterated Prisoner's Dilemma Competition. [Online], Available: `http://www.prisoners-dilemma.com`.

Laird, J. E. (2001). It knows what you're Going to do: adding Anticipation to a Quakebot. In *Proceedings of the 5th International Conference on Autonomous Agents*, AGENTS, pages 385–392. ACM.

Laird, J. E. and Lent, M. V. (2001). Human-Level AI's Killer Application. *AI Magazine*, 22(2):15–26.

Laird, J. E. and van Lent, M. (2000). Human-Level AI's Killer Application: Interactive Computer Games. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, AAAI/IAAI, pages 1171–1178. AAAI Press / The MIT Press.

Laviers, K., Sukthankar, G., Molineaux, M., and Aha, D. (2009). Improving Offensive Performance through Opponent Modeling. In *Proceedings of the 5th AAAI Conference on Artificial Intelligence for Interactive Digital Entertainment*, AIIDE, pages 58–63. AAAI Press.

Lockett, A. J., Chen, C. L., and Miikkulainen, R. (2007). Evolving Explicit Opponent Models in Game Playing. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO, pages 2106–2113. ACM.

Lucas, S. M., Mateas, M., Preuss, M., Spronck, P., and Togelius, J. (2012). Artificial and Computational Intelligence in Games (Dagstuhl Seminar 12191). *Dagstuhl Reports*, 2(5):43–70.

Machado, M. C., Fantini, E. P. C., and Chaimowicz, L. (2011a). Player Modeling: Towards a Common Taxonomy. In *Proceedings of the 16th International Conference on Computer Games*, CGames, pages 50–57. IEEE Press.

Machado, M. C., Pappa, G. L., and Chaimowicz, L. (2012a). A Binary Classification Approach for Automatic Preference Modeling of Virtual Agents in Civilization IV. In *Proceedings of the 8th International Conference on Computational Intelligence and Games*, CIG, pages 155–162. IEEE Press.

Machado, M. C., Pappa, G. L., and Chaimowicz, L. (2012b). Characterizing and Modeling Agents in Digital Games. In *Proceedings of the XI Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, SBGames, pages 26–33. IEEE Press.

Machado, M. C., Rocha, B. S. L., and Chaimowicz, L. (2011b). Agents Behavior and Preferences Characterization in Civilization IV. In *Proceedings of the X Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, SBGames, pages 43–52. IEEE Press.

Mahlmann, T., Drachen, A., Togelius, J., Canossa, A., and Yannakakis, G. N. (2010). Predicting player behavior in Tomb Raider: Underworld. In *Proceedings of the 6th International Conference on Computational Intelligence and Games*, CIG, pages 178–185. IEEE Press.

Manovich, L. (2001). *The Language of New Media*. Massachusetts Institute of Technology, 1st edition.

Martinez, H. P., Hullett, K., and Yannakakis, G. N. (2010). Extending Neuro-evolutionary Preference Learning through Player Modeling. In *Proceedings of the 6th International Conference on Computational Intelligence and Games*, CIG, pages 313–320. IEEE Press.

Missura, O. and Gärtner, T. (2009). Player Modeling for Intelligent Difficulty Adjustment. In *Proceedings of the 12th International Conference on Discovery Science*, DS, Lecture Notes in Computer Science, pages 197–211. Springer.

Nareyek, A. (2004). AI in Computer Games. *Queue*, 1(10):58–65.

Nielsen, J. and Landauer, T. K. (1993). A Mathematical Model of the Finding of Usability Problems. In *Proceedings of the Conference on Human Factors in Computing Systems*, INTERCHI, pages 206–213. ACM.

Pedersen, C., Togelius, J., and Yannakakis, G. N. (2010). Modeling Player Experience for Content Creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67.

Ponsen, M., Gerritsen, G., and Chaslot, G. (2010). Integrating Opponent Models with Monte-Carlo Tree Search in Poker. In *Proceedings of the Interactive Decision Theory and Game Theory Workshop at the 24th International Conference on Artificial Intelligence*, AAAI. AAAI Press.

Ramon, J., Jacobs, N., and Blockeel, H. (2002). Opponent Modeling by Analysing Play. In *Proceedings of the 1st Workshop on Agents in Computer Games at the 3rd International Conference on Computers and Games*, CG, Lecture Notes in Computer Science, pages 1–8. Springer.

Roberts, D. L., Strong, C. R., and Isbell, C. L. (2007). Using Feature Value Distributions to Estimate Player Satisfaction Through an Author's Eyes. In *Proceedings of the AAAI Fall Symposium on Intelligent Narrative Technologies*, AAAI. AAAI Press.

Rohs, M. (2007). *Preference-based Player Modelling for Civilization IV*. PhD thesis, Faculty of Humanities and Sciences, Maastricht University, Maastricht, Netherlands.

Schadd, F., Bakkes, S., and Spronck, P. (2007). Opponent Modeling in Real-Time Strategy Games. In *Proceedings of the European GAMEON Conference on Simulation and AI in Computer Games*, GAME-ON, pages 61–68. EUROSIS.

Scott, B. (2002). *The Illusion of Intelligence*, pages 16–20. Charles River Media.

Sharma, M., Mehta, M., Ontan, S., and Ram, A. (2007). Player Modeling Evaluation for Interactive Fiction. In *Workshop on Optimizing Player Satisfaction in the 3rd AAAI Conference on Artificial Intelligence for Interactive Digital Entertainment*, AIIDE. AAAI Press.

Slagle, J. R. and Dixon, J. K. (1970). Experiments with the M & N tree-searching program. *Communications of the ACM*, 13(3):147–154.

Smith, A., Lewis, C., Hullett, K., and Smith, G. (2011a). An Inclusive Taxonomy of Player Modeling. Technical report, Santa Cruz, CA, USA.

Smith, A. M., Lewis, C., Hullett, K., Smith, G., and Sullivan, A. (2011b). An Inclusive View of Player Modeling. In *Proceedings of the 6th International Conference on the Foundations of Digital Games*, FDG, pages 301–303. ACM.

Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., Billings, D., and Rayner, C. (2005). Bayes' Bluff: Opponent Modelling in Poker. In *In Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence*, UAI, pages 550–558. AUAI Press.

Spronck, P. (2005). A Model for Reliable Adaptive Game Intelligence. In *Workshop on Reasoning, Representation, and Learning in Computer Games in the 19th International Joint Conference on Artificial Intelligence*, IJCAI, pages 95–100. Professional Book Center.

Spronck, P., Balemans, I., and van Lankveld, G. (2012). Player Profiling with Fallout 3. In *Proceedings of the 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE, pages 179–184. AAAI Press.

Spronck, P. and den Teuling, F. (2010). Player Modeling in Civilization IV. In *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE, pages 180–185. AAAI Press.

Tastan, B., Chang, Y., and Sukthankar, G. (2012). Learning to Intercept Opponents in First Person Shooter Games. In *Proceedings of the 8th International Conference on Computational Intelligence and Games*, CIG, pages 100–107. IEEE Press.

Taylor, L. N. (2002). *Video games: Perspective, Point-of-View, and Immersion*. M.S. Thesis, Art School, University of Florida, Gainesville, FL, USA.

Thue, D., Bulitko, V., Spetch, M., and Wasylishen, E. (2007). Interactive Storytelling: A Player Modelling Approach. In *Proceedings of the 3rd AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE, pages 43–48. AAAI Press.

Togelius, J., Yannakakis, G. N., Stanley, K. O., and Browne, C. (2011). Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186.

Valkenberg, A. J. J. (2007). *Opponent Modelling in World of Warcraft*. B.S. Thesis, Faculty of Humanities and Sciences, Maastricht University, Maastricht, Netherlands.

van den Herik, H. J., Donkers, H. H. L. M., and Spronck, P. H. M. (2005). Opponent Modelling and Commercial Games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, CIG, pages 15–25. IEEE Press.

van der Heijden, M., Bakkes, S., and Spronck, P. (2008). Dynamic Formations in Real-Time Strategy Games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, CIG, pages 47–54. IEEE Press.

van Lankveld, G., Schreurs, S., Spronck, P., and van den Herik, H. J. (2010). Extraversion in Games. In *Proceedings of the 7th International Conference on Computers and Games*, CG, Lecture Notes in Computer Science, pages 263–275. Springer.

van Lankveld, G., Spronck, P., van den Herik, H. J., and Arntz, A. (2011). Games as Personality Profiling Tools. In *Proceedings of the 7th International Conference on Computational Intelligence and Games*, CIG, pages 197–202. IEEE Press.

Weber, B. G. and Mateas, M. (2009). A Data Mining Approach to Strategy Prediction. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, CIG, pages 140–147. IEEE Press.

Weber, B. G., Mateas, M., and Jhala, A. (2011). A Particle Model for State Estimation in Real-Time Strategy Games. In *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE, pages 103–108. AAAI Press.

Yannakakis, G. N. (2012). Game AI revisited. In *Proceedings of the 9th ACM International Conference on Computing Frontiers*, CF, pages 285–292. ACM.

Yannakakis, G. N. and Hallam, J. (2009). Real-Time Game Adaptation for Optimizing Player Satisfaction. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):121–133.

Yannakakis, G. N., Maragoudakis, M., and Hallam, J. (2009). Preference Learning for Cognitive Modeling: a Case Study on Entertainment Preferences. *IEEE Transactions on System, Man, and Cybernetics, Part A*, 39(6):1165–1175.

Yannakakis, G. N. and Togelius, J. (2011). Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing*, 2(3):147–161.

Zook, A. and Riedl, M. (2012). A Temporal Data-Driven Player Model for Dynamic Difficulty Adjustment.

Appendix A

CIVILIZATION IV Dataset Features

Table A.1: *List of all features used in the training process. den Teuling [2010] also presents this list, including features' range. Recall that features 129 and 130 were used only in the first classification experiment called off-line review, in Chapter 6. The operators Derivate, Trend, TrendDerivate, Diff, DiffDerivate, DiffTrend and DiffTrendDerivate are in Table 5.2.*

1: Turn
2: War
3: Cities
4: CitiesDerivate
5: CitiesTrend
6: CitiesTrendDerivate
7: CitiesDiff
8: CitiesDiffDerivate
9: CitiesDiffTrend
10: CitiesDiffTrendDerivate
11: Units
12: UnitsDerivate
13: UnitsTrend
14: UnitsTrendDerivate
15: UnitsDiff
16: UnitsDiffDerivate
17: UnitsDiffTrend
18: UnitsDiffTrendDerivate
19: Population
20: PopulationDerivate
21: PopulationTrend
22: PopulationTrendDerivate
45: PlotsTrend
46: PlotsTrendDerivate
47: PlotsDiff
48: PlotsDiffDerivate
49: PlotsDiffTrend
50: PlotsDiffTrendDerivate
51: Techs
52: TechsDerivate
53: TechsTrend
54: TechsTrendDerivate
55: TechsDiff
56: TechsDiffDerivate
57: TechsDiffTrend
58: TechsDiffTrendDerivate
59: Score
60: ScoreDerivate
61: ScoreTrend
62: ScoreTrendDerivate
63: ScoreDiff
64: ScoreDiffDerivate
65: ScoreDiffTrend
66: ScoreDiffTrendDerivate
89: AgricultureDiffTrend
90: AgricultureDiffTrendDerivate
91: Power
92: PowerDerivate
93: PowerTrend
94: PowerTrendDerivate
95: PowerDiff
96: PowerDiffDerivate
97: PowerDiffTrend
98: PowerDiffTrendDerivate
99: Culture
100: CultureDerivate
101: CultureTrend
102: CultureTrendDerivate
103: CultureDiff
104: CultureDiffDerivate
105: CultureDiffTrend
106: CultureDiffTrendDerivate
107: Maintenance
108: MaintenanceDerivate
109: MaintenanceTrend
110: MaintenanceTrendDerivate

23: PopulationDiff
24: PopulationDiffDerivate
25: PopulationDiffTrend
26: PopulationDiffTrendDerivate
27: Gold
28: GoldDerivate
29: GoldTrend
30: GoldTrendDerivate
31: GoldDiff
32: GoldDiffDerivate
33: GoldDiffTrend
34: GoldDiffTrendDerivate
35: Land
36: LandDerivate
37: LandTrend
38: LandTrendDerivate
39: LandDiff
40: LandDiffDerivate
41: LandDiffTrend
42: LandDiffTrendDerivate
43: Plots
44: PlotsDerivate
67: Economy
68: EconomyDerivate
69: EconomyTrend
70: EconomyTrendDerivate
71: EconomyDiff
72: EconomyDiffDerivate
73: EconomyDiffTrend
74: EconomyDiffTrendDerivate
75: Industry
76: IndustryDerivate
77: IndustryTrend
78: IndustryTrendDerivate
79: IndustryDiff
80: IndustryDiffDerivate
81: IndustryDiffTrend
82: IndustryDiffTrendDerivate
83: Agriculture
84: AgricultureDerivate
85: AgricultureTrend
86: AgricultureTrendDerivate
87: AgricultureDiff
88: AgricultureDiffDerivate
111: GoldRate
112: GoldRateDerivate
113: GoldRateTrend
114: GoldRateTrendDerivate
115: ResearchRate
116: ResearchRateDerivate
117: ResearchRateTrend
118: ResearchRateTrendDerivate
119: CultureRate
120: CultureRateDerivate
121: CultureRateTrend
122: CultureRateTrendDerivate
123: StateReligionDiff
124: DeclaredWar
125: CumulativeDeclaredWar
126: AverageDeclaredWar
127: CumulativeWar
128: AverageWar
129: VictoryType
130: Peace

Appendix B

Summary of Indicators' Linear Regressions

Table B.1: *Summary table of the linear regressions discussed in Chapter 5, where the adopted model was $y = b_0 + b_1 x$. The column meanings are, respectively: the data collected in the game, the agent who generated the data, the interval (in turns) the data represents, the game result evaluated (general (victory + defeat), only victories or only defeats), the coefficient of determination (how much of the data is explained by the regression), both coefficients and its confidence intervals and, finally, the confidence used to generate these intervals.*

Indicator	Agent	Interval	Result	R^2	b_0	b_1	Confidence
GoldRate	Louis XIV	[1:460]	General	98.72%	$-19.7615(\pm 8.1688)$	$0.3853(\pm 0.0307)$	99%
GoldRate	Mansa Musa	[1:460]	General	96.14%	$-11.2732(\pm 20.0013)$	$0.3419(\pm 0.0752)$	99%
Gold	Louis XIV	[1:300]	General	62.39%	$44.0798(\pm 76.3993)$	$0.2969(\pm 0.4400)$	90%
Gold	Mansa Musa	[1:340]	General	31.08%	$47.5944(\pm 295.0593)$	$0.2771(\pm 1.4998)$	90%
Gold	Louis XIV	[301:460]	General	75.33%	$-948.8215(\pm 11127.5812)$	$3.5891(\pm 28.9012)$	90%
Gold	Mansa Musa	[341:460]	General	94.67%	$-2059.4734(\pm 4691.6026)$	$6.2651(\pm 11.6708)$	90%
$\sqrt[4]{CultureRate}$	Alexander	[1:460]	General	98.93%	$1.0939(\pm 0.0035)$	$0.0096(\pm 1 \times 10^{-5})$	99%
$\sqrt[4]{CultureRate}$	Hatshepsut	[1:460]	General	99.11%	$1.3567(\pm 0.0047)$	$0.0101(\pm 2 \times 10^{-5})$	99%
$\sqrt[5]{Culture}$	Alexander	[1:460]	General	99.86%	$1.7772(\pm 0.0019)$	$0.0183(\pm 7 \times 10^{-6})$	99%
$\sqrt[5]{Culture}$	Hatshepsut	[1:460]	General	99.85%	$2.1366(\pm 0.0023)$	$0.0194(\pm 9 \times 10^{-6})$	99%
Cities	Alexander	[1:220]	General	97.17%	$0.49439(\pm 0.0408)$	$0.03143(\pm 0.0003)$	99%
Cities	Hatshepsut	[1:220]	General	96.80%	$0.5561(\pm 0.0411)$	$0.0296(\pm 0.0003)$	99%
Cities	Alexander	[221:460]	General	71.39%	$6.2654(\pm 0.0072)$	$0.0021(\pm 2 \times 10^{-5})$	99%
Cities	Hatshepsut	[221:460]	General	56.02%	$5.9320(\pm 0.0099)$	$0.0018(\pm 2 \times 10^{-5})$	99%
Land	Alexander	[1:200]	General	97.90%	$7.7862(\pm 4.0299)$	$0.4899(\pm 0.0347)$	90%
Land	Hatshepsut	[1:200]	General	93.76%	$16.6760(\pm 13.2991)$	$0.5043(\pm 0.1147)$	90%
Land	Alexander	[201:460]	General	23.90%	$94.9505(\pm 0.8058)$	$0.0078(\pm 0.0015)$	90%
Land	Hatshepsut	[201:460]	General	50.04%	$99.2728(\pm 0.7268)$	$0.0166(\pm 0.0021)$	90%
Plots	Alexander	[1:200]	General	99.15%	$3.6923(\pm 7.0220)$	$0.8176(\pm 0.0606)$	99%
Plots	Hatshepsut	[1:200]	General	98.05%	$12.9458(\pm 19.3722)$	$0.8900(\pm 0.1671)$	99%
Plots	Alexander	[201:460]	General	78.73%	$149.6914(\pm 13.6018)$	$0.1109(\pm 0.0401)$	99%
Plots	Hatshepsut	[201:460]	General	88.22%	$163.7941(\pm 6.0591)$	$0.1053(\pm 0.0178)$	99%
GoldRate	Louis XIV	[1:460]	Victory	98.48%	$-25.3125(\pm 9.2060)$	$0.4694(\pm 0.0346)$	90%

GoldRate	Mansa Musa	[1:460]	Victory	97.03%	$-24.2833(\pm 19.5158)$	$0.4842(\pm 0.0733)$	90%
GoldRate	Louis XIV	[1:460]	Defeat	97.98%	$-11.8778(\pm 4.6118)$	$0.2867(\pm 0.0173)$	90%
GoldRate	Mansa Musa	[1:460]	Defeat	84.60%	$-0.4737(\pm 26.7967)$	$0.2326(\pm 0.1007)$	90%
$\sqrt[3]{Gold}$	Louis XIV	[1:460]	Victory	79.94%	$2.6224(\pm 0.1775)$	$0.0128(\pm 0.0007)$	99%
$\sqrt[3]{Gold}$	Mansa Musa	[1:460]	Victory	72.60%	$3.0379(\pm 0.1408)$	$0.0093(\pm 0.0005)$	99%
$\sqrt[3]{Gold}$	Louis XIV	[1:460]	Defeat	82.35%	$3.1341(\pm 0.0692)$	$0.0087(\pm 0.0003)$	99%
$\sqrt[3]{Gold}$	Mansa Musa	[1:460]	Defeat	60.15%	$2.7212(\pm 0.3289)$	$0.0108(\pm 0.0012)$	99%
$\sqrt[4]{CultureRate}$	Alexander	[1:460]	Victory	99.33%	$1.0614(\pm 0.0029)$	$0.0102(\pm 1 \times 10^{-5})$	99%
$\sqrt[4]{CultureRate}$	Hatshepsut	[1:460]	Victory	98.78%	$1.3165(\pm 0.0065)$	$0.0111(\pm 2 \times 10^{-5})$	99%
$\sqrt[4]{CultureRate}$	Alexander	[1:460]	Defeat	98.95%	$1.0867(\pm 0.0034)$	$0.0086(\pm 1 \times 10^{-5})$	99%
$\sqrt[4]{CultureRate}$	Hatshepsut	[1:460]	Defeat	98.20%	$1.4589(\pm 0.0057)$	$0.0085(\pm 1 \times 10^{-5})$	99%
$\sqrt[5]{Culture}$	Alexander	[1:460]	Victory	99.86%	$1.7152(\pm 0.0022)$	$0.0195(\pm 8 \times 10^{-6})$	99%
$\sqrt[5]{Culture}$	Hatshepsut	[1:460]	Victory	99.87%	$2.0610(\pm 0.0024)$	$0.0210(\pm 8 \times 10^{-6})$	99%
$\sqrt[5]{Culture}$	Alexander	[1:460]	Defeat	99.84%	$1.8082(\pm 0.0019)$	$0.0171(\pm 7 \times 10^{-6})$	99%
$\sqrt[5]{Culture}$	Hatshepsut	[1:460]	Defeat	99.65%	$2.2963(\pm 0.0046)$	$0.0176(\pm 2 \times 10^{-5})$	99%
Cities	Alexander	[1:200]	Victory	98.01%	$0.3077(\pm 0.0188)$	$0.0343(\pm 0.0002)$	90%
Cities	Hatshepsut	[1:200]	Victory	97.91%	$0.3045(\pm 0.0205)$	$0.0350(\pm 0.0002)$	90%
Cities	Alexander	[201:460]	Victory	95.11%	$4.9651(\pm 0.0143)$	$0.0082(\pm 4 \times 10^{-5})$	99%
Cities	Hatshepsut	[201:460]	Victory	81.79%	$6.1602(\pm 0.0202)$	$0.0047(\pm 5 \times 10^{-5})$	99%
Cities	Alexander	[1:200]	Defeat	96.98%	$0.4894(\pm 0.0278)$	$0.0309(\pm 0.0002)$	95%
Cities	Hatshepsut	[1:200]	Defeat	96.58%	$0.5466(\pm 0.0262)$	$0.0281(\pm 0.0002)$	95%
Cities	Alexander	[201:460]	Defeat	79.89%	$6.9878(\pm 0.0072)$	$-0.003(\pm 2 \times 10^{-5})$	95%
Cities	Hatshepsut	[201:460]	Defeat	39.59%	$6.0004(\pm 0.0199)$	$-0.0020(\pm 7 \times 10^{-5})$	95%
Land	Alexander	[1:180]	Victory	98.45%	$5.7677(\pm 3.8054)$	$0.5498(\pm 0.0364)$	95%
Land	Hatshepsut	[1:180]	Victory	96.59%	$11.5814(\pm 10.8481)$	$0.6202(\pm 0.1039)$	95%
Land	Alexander	[181:460]	Victory	93.65%	$89.7060(\pm 0.7846)$	$0.0606(\pm 0.0023)$	95%
Land	Hatshepsut	[181:460]	Victory	53.95%	$111.5217(\pm 1.3284)$	$0.0222(\pm 0.0040)$	95%
Land	Alexander	[1:180]	Defeat	97.96%	$6.7485(\pm 3.3395)$	$0.4475(\pm 0.0320)$	95%
Land	Hatshepsut	[1:180]	Defeat	93.27%	$15.6509(\pm 13.7272)$	$0.4877(\pm 0.1315)$	95%
Land	Alexander	[181:460]	Defeat	74.29%	$85.2093(\pm 0.2312)$	$-0.0145(\pm 0.0009)$	95%
Land	Hatshepsut	[181:460]	Defeat	2.74%	$91.6748(\pm 2.5132)$	$-0.0047(\pm 0.0076)$	95%
Plots	Alexander	[1:250]	Victory	98.43%	$7.9857(\pm 17.0135)$	$0.7904(\pm 0.1175)$	99%
Plots	Hatshepsut	[1:250]	Victory	97.26%	$17.4191(\pm 38.4130)$	$0.8929(\pm 0.2653)$	99%
Plots	Alexander	[251:460]	Victory	96.77%	$0.1694(\pm 0.0104)$	$151.8041(\pm 3.7521)$	99%
Plots	Hatshepsut	[251:460]	Victory	73.78%	$203.3449(\pm 10.4762)$	$0.0866(\pm 0.0290)$	99%
Plots	Alexander	[1:210]	Defeat	99.06%	$7.0925(\pm 6.3405)$	$0.7110(\pm 0.0521)$	99%
Plots	Hatshepsut	[1:180]	Defeat	97.21%	$14.0890(\pm 20.5100)$	$0.8248(\pm 0.1965)$	99%
Plots	Alexander	[211:460]	Defeat	50.76%	$145.9592(\pm 7.0910)$	$0.0424(\pm 0.0207)$	99%
Plots	Hatshepsut	[181:460]	Defeat	42.52%	$152.8303(\pm 11.8983)$	$0.0461(\pm 0.0359)$	99%

Appendix C

Questionnaires applied to Human Players

This appendix presents both questionnaires applied to the human players. After signing a consent form, they were asked to fill a pretest questionnaire. Its objective was to obtain the player's profile. After filling it, they played a match of the game CIVILIZATION IV and were asked to fill the post-test questionnaire. All questionnaires were presented to the players in Portuguese.

C.1 Pretest Questionnaire

The pretest questionnaire was the larger questionnaire players were asked to answer. Not all players have answered all the questions, since the questionnaire was dynamic, i.e. it does not ask the player how frequently he/she has played a game if, in the previous question, the player answered that he/she does not know turn-based strategy games.

All the available questions on the questionnaire are presented in Figures C.1, C.2, C.3, C.4 and C.5. Regarding its flow, it is presented below.

All players started answering the questionnaire C.1. At the last question, if the player selects one of the first two bullets, he/she is forwarded to questionnaire C.5, otherwise to questionnaire C.2. In questionnaire C.2, the next questionnaire to be answered is also defined by the last question. If the player choses one of the first two bullets, it is forwarded to questionnaire C.4, otherwise to questionnaire C.3. Finally, both questionnaires, when answered, forward the player to a final acknowledgment message (C.5), with further instructions to play the game.

Dados Pessoais

Qual o seu nome?

O nome será utilizado apenas para agrupamento das respostas deste questionário com as do questionário a ser respondido após os testes. Não será divulgado ou apresentado de forma alguma, de acordo com o termo de consentimento.

Qual a sua faixa etária?

- ○ Abaixo de 18 anos
- ○ 18 a 20 anos
- ○ 21 a 25 anos
- ○ 26 a 30 anos
- ○ Acima de 30 anos

Você já jogou jogos de estratégia baseados em turnos (TBS - Turn Based Strategy)?

Alguns exemplos de jogos TBS: "Civilization", "Heroes of Might and Magic", "Panzer General", "Galactic Civilizations", "Age of Wonders", "Colonization" e "Call to Power".

- ○ Não conheço
- ○ Conheço, mas nunca joguei
- ○ Já joguei um pouco
- ○ Já joguei bastante

Figure C.1: *Pretest questionnaire: Questions about player's personal information.*

Experiência com jogos de estratégia baseados em turnos

Qual foi a última vez que jogou uma partida de um jogo de estratégia baseado em turnos?

- ○ Há mais de 2 anos
- ○ Entre 2 anos e 1 ano
- ○ Entre 1 ano e 6 meses
- ○ Entre 6 meses e 1 mês
- ○ No último mês

Mais especificamente, você já jogou algum jogo da série CIVILIZATION?

- ○ Não conheço
- ○ Conheço, mas nunca joguei
- ○ Já joguei um pouco
- ○ Já joguei bastante

Figure C.2: *Pretest questionnaire: Questions about player's experience in turn-based strategy games.*

Experiência com jogos da série CIVILIZATION IV

Qual jogo da série CIVILIZATION você já jogou?

- ○ Civilization
- ○ Civilization II
- ○ Civilization III
- ○ Civilization IV
- ○ Civilization V

Qual foi a última vez que jogou uma partida de um jogo da série CIVILIZATION?

- ○ Há mais de 2 anos
- ○ Entre 2 anos e 1 ano
- ○ Entre 1 ano e 6 meses
- ○ Entre 6 meses e 1 mês
- ○ No último mês

Na época em que mais jogava algum jogo da série CIVILIZATION, com qual frequência jogava?

- ○ Menos de 1 vez por semana
- ○ 1 vez por semana
- ○ 2 vezes por semana
- ○ 3 vezes por semana
- ○ 4 vezes por semana
- ○ 5 vezes por semana
- ○ Mais de 5 vezes por semana

Como você classifica seu nível de habilidade como jogador dos jogos da série CIVILIZATION?

- ○ Fraco
- ○ Razoável
- ○ Bom
- ○ Excelente

Figure C.3: *Pretest questionnaire: Questions about player's experience in games of the* CIVILIZATION *series.*

Experiência com jogos de estratégia baseados em turnos

Como você classifica seu nível de habilidade como jogador de jogos de estratégia baseados em turnos?

- ○ Fraco
- ○ Razoável
- ○ Bom
- ○ Excelente

Figure C.4: *Pretest questionnaire: Questions about player's experience in turn-based strategy games.*

Conclusão

Obrigado por participar do questionário. Por favor, agora siga as instruções presentes no endereço abaixo `http://www.dcc.ufmg.br/~marlos/civ4.html` para continuar o teste.

Figure C.5: *Acknowledgment of the pretest questionnaire.*

C.2 Post-test Questionnaire

After playing the game, all players were required to fill the post-test questionnaire. Its objective was to obtain the players' preferences and their confidence on the self-labeled preferences. This questionnaire is presented in Figure C.6.

Questionário Pós-Teste

Qual o seu nome?

O nome será utilizado apenas para agrupamento das respostas deste questionário com as do questionário a ser respondido após os testes. Não será divulgado ou apresentado de forma alguma, de acordo com o termo de consentimento.

Qual o nome do seu agente, no jogo?

Exemplos de agentes são: *Alexander, Cyrus, Mansa Musa*. No arquivo de log que me enviará, existirá um nome em cada arquivo, um nome é o seu, o outro, do seu rival.

Como você classificaria suas preferências durante a partida jogada de CIVILIZATION IV?

Marque apenas os itens que você julga ter agido como se tivesse preferência por ele. Essa é uma avaliação binária, ou seja, ou você apresentou ou você não apresentou um comportamento condizente com cada uma das preferências abaixo.

- ○ Ciência
- ○ Crescimento
- ○ Cultura
- ○ Militar
- ○ Ouro
- ○ Religião

Quão confiante você se sente com relação à sua avaliação anterior?

Dê uma nota de 0 a 5, onde 0 significa sem confiança alguma e 5 total confiança.

- ○ 0
- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5

Figure C.6: *Post-test questionnaire: Questions about the match played and the player's preference.*

Appendix D

Accuracy Classifying each Human Player

Table D.1: *Accuracy of the three evaluated methods (Naive Bayes, AdaBoost and JRip) when classifying each individual human player (%). They are identified by numbers to hide their identity. Players' numbers with an asterisk represent experienced players, while those without asterisk are beginners.*

	Player #	Culture	Gold	Growth	Military	Religion	Science
NaiveBayes	1*	93.0	1.0	95.0	100.0	0.0	100.0
	2	2.8	84.1	91.1	85.5	85.5	91.6
	3	100.0	100.0	3.4	100.0	100.0	0.0
	4*	1.9	11.2	7.7	95.4	4.6	97.7
	5	6.1	2.2	95.0	5.8	5.8	99.7
	6	17.0	71.0	0.8	21.7	21.7	69.4
	7*	3.6	8.6	99.2	95.8	4.2	96.7
AdaBoost	1*	0.0	97.0	0.0	0.0	100.0	0.0
	2	100.0	0.0	0.0	0.0	0.0	0.0
	3	100.0	100.0	100.0	100.0	100.0	100.0
	4*	100.0	100.0	100.0	0.0	100.0	0.0
	5	100.0	71.1	0.0	100.0	100.0	0.0
	6	84.4	100.0	100.0	0.0	0.0	0.0
	7*	100.0	100.0	0.0	0.0	100.0	0.0
JRip	1*	0.0	74.0	0.0	29.0	59.0	0.0
	2	100.0	15.6	0.0	7.5	6.1	0.0
	3	100.0	100.0	41.4	100.0	100.0	100.0
	4*	100.0	71.8	100.0	20.1	77.6	0.0
	5	100.0	92.8	0.3	91.9	95.0	0.0
	6	100.0	100.0	100.0	0.0	0.0	0.0
	7*	99.7	89.4	8.9	0.3	95.0	0.0