Selecting Learning Algorithms for the Design of Virtual Players in Natural Resources Management Platforms

Guillaume MULLER and Jaime S. SICHMAN
Laboratório de Técnicas Inteligentes (LTI/PCS)
Escola Politécnica de Universidade de São Paulo
Av. Luciano Gualberto, 158, Trav. 3
05508-010 – São Paulo – SP – Brasil

Abstract

This paper focuses on the problem of designing Virtual Players for a particular type of serious game, in the domain of Natural Resources Management. Indeed, in some cases, it is impossible to play a particular game because there is not enough participants. Designing and implementing Virtual Players, that would stand in for some of the lacking human players, would allow to circumvent this limitation. In order to get realistic substitute players, it has been proposed to design the Virtual Players by learning actual human behaviours based on traces of games in which they are involved. However, there exists plenty of learning algorithms, with very different requirements and properties and no a priori best algorithm. This paper addresses more particularly the problem of selecting learning algorithms for the design of Virtual Players in Natural Resources Management "ABPS" platforms. Its main contributions are: a list of criteria for practically comparing the learning algorithms and the selection of some algorithms to be used in the design of Virtual Players for Natural Resources Management "ABPS" platforms.

Keywords:: Serious Games, Virtual Players, Learning Algorithms, Artificial Intelligence

Author's Contact:

guillaume.muller@freesurf.fr, jaime.sichman@poli.usp.br

Introduction

Serious Games are games whose ultimate goal is not only pure amusement. For some time now, researchers and teachers have shown a high interest in such games, since their entertaining aspect allows to raise or maintain the interest of the participants, even when addressing complex and serious problems. Such games have been used in various domains, like Natural Resources Management, as tools to support either training, multi-actor decisions, or social observation [Barreteau et al. 2001].

As a technical support for such games, a new kind of computer-mediated interaction platform has recently emerged. It is generally named "MAS/RPG" [Barreteau et al. 2001] or "ABPS" (Agent-Based Participatory Simulations) [Guyot and Honiden 2006], as it combines both Role Playing Games (RPG) and Multi-Agent-Based Simulation (MABS). It consists in making social actors play some roles in a simulation game, supported by (in the case of "MAS/RPG") or mediated by (in the case of "ABPS") a multi-agent platform.

Due to lack of availability, competence, interest, or to test repeatedly given configurations, it is sometimes impossible to group all the social actors at the same time to play all the mandatory roles of a particular simulation game. A solution to this problem is to design, implement and run Virtual Players, that would enact some of the roles needed by a particular simulation game. One of the challenges of the design of Virtual Players is to get realistic substitutes for the lacking human social actors, in order not to slant the results of the games. In order to get such realistic Virtual Players, it has been proposed to make the Virtual Players learn their behaviour directly from the ones of human social actors, based on traces of execution of the simulation games involving these Human Players.

However, there are plenty of learning algorithms in the literature, with very different requirements and properties, which makes them more or less suitable for the task. But, as WOLPERT's "No Free Lunch" theorem states, there is no a priori best learning algorithm. As a consequence, there is a need for a more practical methodology of selection of the algorithms, based on the particular application targeted. The general methodology of our work is to use the description of the application domain as a mean to extract a list of constraints on the Virtual Players, which is then translated into a list of properties that the requested learning algorithms must show. The application domain in our case is constituted by the "ABPS" platforms ViP-JogoMan [Adamatti 2007] and SimParc [Briot et al. 2008].

The results of this methodology and the main contributions of the paper are: (i) a list of criteria for a practical comparison of learning algorithms; and (ii) the selection of some algorithms to be used in the design of Virtual Players for "ABPS" platforms of Natural Resources Management.

Section 2 presents the criteria for comparing the algorithms, Section 3 details the comparison of the algorithms and Section 4 conclude and establishes directions for future works.

1 Constraints from the platforms

In this section, we identify some constraints that the functioning of the studied "ABPS" platforms [Adamatti 2007; Briot et al. 2008] imply on the choice of the learning algorithms used to design the Virtual Players.

1.1 Constraints for the category of algorithms

In this section we detail the characteristics of the platforms that will influence the choice of a particular category of algorithms.

- **Type of players** It is possible to differentiate three targets for the learning algorithms: (i) a Virtual Player to replace the Game Master (who plays special role in the process of decision-making); (ii) a Virtual Player for any other role; or (iii) a Virtual Assistant. The inputs and outputs of the players' decisions are different in each case: for instance, designing a normal player requires to integrates environmental information to make a decision, whereas designing a Game Master requires to handle partial decisions already made by other players.

- **Profiles for the roles** There can be different profiles for each role. For instance, a Game Master can have more environmental or more social tendencies. As a consequence, it is possible to distinguish two classes of learning problems: the first learning problem consists in grouping similar traces of behaviours for a same role into profiles; the second problem consists in learning to reproduce a particular profile.

- **Acquiring traces** Because the availability of the Human Players might be limited and the games take a long time to run, it is generally difficult to generate a large amount of traces. As a consequence, it is necessary: (i) to select games that will promote the production of relevant traces; and (ii) to select a category of learning algorithms able to rapidly extract important salient features.

1.2 Constraints for the internal model

In this section, we present the characteristics of the platforms that will influence the choice of the internal model of the players.

- **Granularity** The traces collected in the platforms will take the form of a sequence of actions (communicative or on the environment). The Virtual Players, however, can be trained and ran on pieces of information of different granularity levels: the whole sequence of actions, sub-sequences, or single decisions. It is particularly important to look at this criterion as the traces reveal the negotiation strategies of the players and a too small piece of information may make the learning algorithm unable to capture the link between actions, whereas a too large piece may make it link actions that should not.
- **Abstraction** The negotiations occurring in the platforms might involve several interactions referring to a same object and discussing some attributes of this object. For instance, during the negotiation consisting of a land owner buying a new area, the object of interest will be the land area and an attribute could be the price. It would therefore be interesting to learn to abstract the price variable, in order to learn the negotiation strategy, not the particular price proposed in this negotiation.
- **Human understandability** It can be important that the internal model is directly readable by humans, either for the designers to access the extracted features or to enable Virtual Players to explain their decisions (introspection). Other possibilities are that the internal model is easily translatable into an intelligible model, or that the model is questionable by the designers.

1.3 Constraints for the algorithm

The characteristics of platforms that will influence the final choice of the learning algorithms are presented in this section.

- **RPG rules and players' interactions** The rules of the RPG generally constrain the interactions between the players. There are two major elements that will influence the choice of the algorithm: (i) the way interactions occur, i.e. if there exists interaction protocols that the players must follow; and (ii) the communication language used, as it can be very structured or relatively free.
- **Stealth** It is important that the Virtual Players act like real Human Players. This means that they must be difficult to detect by Human Players. As a consequence, there is a need to choose a learning algorithm that makes Human-like decisions, not necessarily the most rational/logical ones.
- **Unpredictability** The behaviours of the Virtual Players should not be too predictable: even in similar configurations, they should not necessarily play exactly the same move. Of course, this should occur only when it is possible, i.e. the Virtual Players should not overstep the bounds of their profile nor of legal moves. This aspect is related to stealth as it is clear that both a very predictable and unpredictable Virtual Players will quickly appear to be artificial.
- **Innovation** It is not wanted that Virtual Players for Natural Resources Management platforms, learn to reproduce *exactly* what they observed in the traces. There is no notion of optimality or uniqueness of the solution to be reached by the learning algorithms. Also, Virtual Players will not be trained on every possible situation, they will face unknown situations. As a consequence, the learning algorithms should help explore new possibilities and innovate. This property is related to unpredictability and stealth, as Human Players generally have a prioris about machines not being innovative.
- **Adaptability** The players may evolve with time, as players gather more information in each time step, and with game repetitions, if the same players participate in several games. The Virtual Players could benefit adapting their behaviour dynamically. This step is related to their stealth, particularly if the same Human Players encounters the same Virtual Players in multiple games.

1.4 Constraints on the inputs

This section presents the constraints that the characteristics of platforms will imply on the inputs of the learning algorithms.

- **Similarity** Whatever the use of learning algorithms: to identify profiles or reproduce a profile, it is necessary to define a similarity measure between two traces. In the identification task, the similarity measure is used to compare two traces to decide to group or separate them, whereas in the reproduction task, it is needed to evaluate the performance of the learning process, by comparing the output traces, generated by the algorithm, to the input traces, initially generated by the Human Players.

2 Criteria for a practical comparison

In this section, we briefly present the criteria, extracted from the application domain, that will enable us, in the next section, to compare the (classes of) learning algorithms.

The following items presents these criteria, the links with the constraints of the previous section and the possible values for each criterion:

- **Category**: Due to the type of decision to be made, there is a direct relation between the *Type of Player* (Section 1.1) and the *Category* of the learning algorithm to be selected. Algorithms are compared based on the category they belong to. This criterion admits six possible values: *Supervised*, *Unsupervised*, *Semi-supervised*, *Reinforcement*, *Transduction*, and *Multi-task*.
- **Aim**: Section 1.1 (*profiles*) shows that there are two learning problems: classify the traces into profiles (without necessarily having the labels), and learning to reproduce a labelled profile. Algorithms are compared based on the problems they can address. This criterion admits two possible values: *classification* (supervised or not) into profiles; and *reproduction*.
- **Inputs/Outputs**: The learning algorithms are compared according to the *granularity* of the inputs and outputs that they can handle (Section 1.2). This criterion admits four values, combinations of: *single*, if the algorithm is able to handle single players' decisions; and *sequence*, if the algorithm is able to handle (sub-)sequences.
- **Number of examples**: In our application domain, there are generally *few traces* (Section 1.1). The algorithms are compared according to the quantity of examples needed. This criterion will admit three possible values: *many* examples, if the algorithm needs many examples to learn correctly; *few* examples, if it is able to produce correct results even with few examples, and *normal* if it does not fall in the previous categories.
- **Generalisation**: the learning algorithms are compared based on their ability to abstract or not the examples learnt. Generalisation helps cope with *abstraction* (Section 1.2), *innovation*/unknown situations and *stealth* (Section 1.3). This criterion admits a binary response: *yes*, if it is able to generalise or *no*, if it is not.
- **Internal model**: The learning algorithms are compared based on the *internal models* they accept (Section 1.2) This criterion admits two possible values: *numerical*, for learning algorithms that can accept an internal model like statistics; and *symbolic*, for learning algorithms that accepts an internal model like logics.
- **Intelligibility**: the learning algorithms are compared based on their ability produce *intelligible* models (Section 1.2). This criterion admits three values: *direct*, if it is directly intelligible; *translatable*, if it is translatable into an intelligible model;

and *questionable*, if it can be kept as a questionable "black-box".

- **Non-determinism**: The algorithms are compared along their ability to produce output that is not completely deterministic. This helps for *unpredictability* (Section 1.3), *innovation* and *stealth* (Section 1.3). This criterion admits a binary response: *yes*, if it is able to produce output that is not necessarily completely deterministic or *no*, if it is not.
- **Incremental**: the learning algorithms are compared based on their ability to cope incrementally with new examples. This helps for *adaptation* (Section 1.3) and to run on-line Virtual Assistants (Section 1.1). This criterion admits two possible values: *yes*, if the algorithm is working in an incremental way; and *no*, if it is not.
- **A priori knowledge**: The learning algorithms are compared judging their ability to accept *a priori knowledge* (e.g., heuristics) or not (Section 1.3). This criterion admits three possible values: *cannot*, if it cannot accept a priori knowledge; *need*, if it needs a priori knowledge to be able to function; and *can*, if it can accept a priori knowledge, but do not necessarily need it to work.

3 Practical comparison of the learning algorithms

As there exists many algorithms in the literature, we have presented in the table only large classes of learning algorithms. The selection of the classes is based on the functional approach exploited in [Mitchell 1997], which discerns nine classes of algorithms based on their functioning: Decision tree, Neural Networks, Bayesian, Instance-based, Genetic, Set of Rules, Analytical, Reinforcement, and SVM. The filling of the table has been made considering a few representatives for each class: for **Decision tree**, we have used ID3 and C4.5; for **Neural Networks**, we have used the perceptron and multi-layered neural networks; for **Bayesian**, we have used Naive Bayes and Expectation-Maximisation (EM); for **Instance-based**, we have used Case-Based Reasoning (CBR) and k-Nearest Neighbours (kNN); for **Genetic**, we have used the general concepts, plus the Strongly-Typed Genetic Programming (STGP) [Montana 1995]; for **Set of Rules**, we have used RIPPER, AQ and CN2; for **Analytical**, we have used Prolog-EBG; for **Reinforcement**, we have used Q-Learning and Temporal differences plus some Inverse Reinforcement Learning [Abbeel and Ng 2004; Ng and Russell 2000]; for **SVM**, we have used general concepts.

Table 1 presents, in a concise way, the results of our comparison of these various classes of learning algorithms. A legend of the abbreviations and symbols used is present in the last row.

The values for the first column have been filled based on information from several classical works on learning algorithms [Kotsiantis 2007; Alpaydın 2004; Mitchell 1997]. The line about Analytical Learning is mainly based on [Knuutila 1999] , the line about Genetic Programming on [Koza 1992] and the line about Reinforcement Learning on [Mitchell 1997].

In the "Aim" columns, the ability to classify has been mainly established thinking about a typical task of supervised classification. The ability to reproduce is based on a synthesis of the four next columns: a check-mark is put where the four next columns have a check-mark and a tilde is put as soon as there is more than one tilde in the next columns.

In order to fill the "In/Out" columns, we have considered the task of reproduction has a supervised classification task: the algorithm is trained to associate a decision or sub-sequence of decisions with the reacting decision or sub-sequence of decisions; and the algorithm is used by being provided the current decision or sub-sequence of decisions occurring in the game and asked to provide the next move (single decision) or moves (sub-sequence of decisions). The tildes 1–4, 5–7, 10–12, represent the fact that it can be hard to represent an arbitrary length sequence as the label for Neural Networks, Instance-based, and SVM algorithms.

The column for "number of examples" has been filled mainly based on [Kotsiantis 2007], where *normal* is put when the paper does not explicit the need for few or many examples. The *not relevant* marks have been put for Genetic Programming and Reinforcement Learning because they do not naturally work based on a set of training examples like the other algorithms, but respectively on size and iteration over a population and reinforcement signal.

In order to fill the "Generalisation" column, we have considered the task of responding to unknown situations. The tildes 13, 14, 16, 17 are there to underline potential problems with over-fitting. However, there are simple solution to prevent this, for instance pruning for Decision Trees and Sets of Rules, so these kinds of algorithms have not been totally rejected.

The "Internal model" values have been filled based on information from the same classical works on learning algorithms than column one. The coma separator signifies that both models can be used, whereas the pipe (|) signifies that there are models that do use rather symbolic representation (e.g., Case-Based Reasoning) whereas others use rather numerical internal models (e.g. k-Nearest Neighbours).

Generally, symbolic are more intelligible then numerical models. *questionable* has been put when [Kotsiantis 2007] criticises the algorithm's intelligibility and *direct* when it says it is good. *translatable* is used for models like Bayesian Networks and Reinforcement Learning, which can be understood with a little practise or converted into a more human-readable model

The non-determinism has been judged based on the ability of each learning algorithm to produce non-deterministic models. For instance, to produce probabilistic models that say: "in situation S, do action X with 80% probability or action Y, with 20% probability".

The incremental property is related with the ability to run the algorithm on-line. To judge this aspect, we have considered the very functioning of the algorithm, i.e. whether the algorithm only revises a small part of its model when a new example is presented or if it needs to rebuild everything from scratch. According to [Kotsiantis 2007], Decision Trees, Sets of Rules and SVM are very bad at managing new examples. [Knuutila 1999] says that Prolog-EBG has a sequential covering. Neural Networks and Reinforcement Learning are very adapted to such conditions.

Concerning the last column, according to [Knuutila 1999] (resp. [Mitchell 1997]), Analytical Learning (resp. Bayesian Networks) requires a domain theory (resp. bootstrap information). With very simple algorithms like Q-Learning, Reinforcement Learning does not need a priori knowledge. However, utility-based algorithms require a significant amount of it. That is why we used the pipe notation. Finally, *can* has been used when heuristics can be implemented to help the algorithm learn more precisely or more quickly.

4 Possibilities

In this section, we discuss the possibilities regarding our needs and the characteristics of the learning algorithms.

By combining the two first columns of Table 1, our choice for the unsupervised classification problem is reduced to: Neural Networks, Bayesian Networks, and Instance-based learning. By adding that number of examples and ability to generalise are the more important criteria for our problem and that a priori knowledge is an interesting property, we can remove the first option and keep as possibilities Bayesian Networks (which is better because of few examples) and Instance-based learning (which is better as it accepts heuristics).

Concerning the reproduction problem, the criteria that are very important are: number of examples; ability to handle sequences (In/Out and internal model); ability to generate non-deterministic behaviours; and ability to generalise. Based on these criteria, only remain the Bayesian Networks, Genetic Programming and Reinforcement Learning possibilities. Bayesian Networks is rejected as it suffers from the assumption of independence of among child

	Category	Aim		In/Out				♯ Examples
		class	repro	dec/dec	dec/seq	seq/dec	seq/seq	
Decision tree	super	✓	✓	✓	✓	✓	✓	normal
Neural Networks	super,unsup^{al},multi^{al}	✓	$\sim_1$	✓	$\sim_2$	$\sim_3$	$\sim_4$	many
Bayesian	super,unsup^{al},semisup^{al}	✓	✓	✓	✓	✓	✓	few
Instance-based	super,unsup	✓	$\sim_5$	✓	$\sim_6$	✓	$\sim_7$	normal
Genetic	reinf^{al},multi^{al}	✓	✓	✓	✓	✓	✓	$-_8$
Set of Rules	super	✓	✓	✓	✓	✓	✓	normal
Analytical	super	✓	✓	✓	✓	✓	✓	few
Reinforcement	reinf	✓	✓	✓	✓	✓	✓	$-_9$
SVM	super	✓	$\sim_{10}$	✓	$\sim_{11}$	✓	$\sim_{12}$	many
Legend	**S**upport **V**ector **M**achine; **super**vised; **unsup**ervised; **semi-sup**ervised; **reinf**orcement; **multi**task. **class**ification; **repro**duction. ✓ = yes; ✗ = not possible without modifying the algorithm; $\sim$ = not immediate, but possible. singl **dec**ision; **seq**uence of decisions. – = not relevant.							

	Generalization	Int. Model	Intelligibility	Non-determinism	Incrementality	A priori
Decision tree	$\sim_{13}$	sym	direct	✗	✗	can
Neural Networks	$\sim_{14}$	num	quest	✓	✓	cannot
Bayesian	✓	num	transl	✓	✓	need
Instance-based	✓	sym^{al} |num	quest	✓	✓	can
Genetic	✓	sym,num	$-_{15}$	✓	✗	can
Set of Rules	$\sim_{16}$	sym	direct	✗	✗	can
Analytical	✓	sym	direct	✗	✓	need
Reinforcement	✓	num	transl	✓	✓	can|need
SVM	$\sim_{17}$	num	quest	✗	✗	can
Legend	✓ = yes; ✗ = not possible without modifying the algorithm; $\sim$ = not immediate, but possible. **num**eric; **sym**bolic. **transl**atable; **quest**ionable. – = not relevant. $_x$: referred in explanations below.					

Table 1: *Practical comparison of general learning algorithms classes.*

nodes. The advantage of Genetic Programming is that the internal model can be chosen. With reinforcement learning the Virtual Player can adapt dynamically.

Conclusion and future work

In this paper, we address the problem of designing Virtual Players for a particular type of serious games, the "ABPS" platforms, in the domain of Natural Resources Management. As the Virtual Players must show a credible behaviour, it was proposed to make them learn their behaviour based on traces of execution of games involving the Human Players they should replace. In order to select the learning algorithms to perform such a task, we extracted a list of criteria from the functioning of the platforms, in order to make a practical comparison of the learning algorithms and proposed some learning algorithms that appear more adapted to our goal.

There are two main contributions of this paper: the methodology to choose criteria for practical selection of the learning algorithms and the selection of some algorithms to be used.

Practical future work consists in implementing and evaluating the Virtual Players. This will be possible only when a working platform will be publicly available, i.e. (i) *runnable*, in order to actually collect traces and (ii) *modifiable*, in order to insert the Virtual Players. A more theoretical future work will consist in generalising the methodology developed here, in order to propose algorithm selection criteria for other type of Serious Games.

Acknowledgements

Dr. MULLER is supported by CNPq grant 382737/2008-3.

References

ABBEEL, P., AND NG, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In International Conference on Machine Learning (ICML'04), ACM Press.

ADAMATTI, D. F. 2007. Inserção de Jogadores Virtuais em Jogos de Papéis para Uso em Sistemas de Apoio a Decisão em Grupo: um Experimento no Domínio da Gestão de Recursos Naturais. PhD thesis, Universidade de São Paulo, São Paulo, Brasil.

ALPAYDIN, E. 2004. Introduction to Machine Learning (Adaptive Computation and Machine Learning). MIT Press.

BARRETEAU, O., BOUSQUET, F., AND ATTONATY, J. 2001. Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to senegal river valley irrigated systems. Journal of Artificial Societies and Social Simulation (JASSS) 4, 2.

BRIOT, J.-P., VASCONCELOS, E., ADAMATTI, D., SEBBA PATTO, V., IRVING, M., BARBOSA, S., FURTADO, V., AND LUCENA, C. 2008. A computer-based support for participatory management of protected areas: The simparc project. In Seminário Integrado de Software e Hardware (SEMISH'08), Sociedade Brasileira de Computação (SBC), Brazil, Belém, PA, Brazil, 1–15.

GUYOT, P., AND HONIDEN, S. 2006. Agent-based participatory simulations: Merging multi-agent systems and role-playing games. Journal of Artificial Societies and Social Simulation (JASSS) 9, 4, 8.

KNUUTILA, T., 1999. Machine learning lectures, chapter 11: Analytical learning. Built in: 1999. Last update: May 24, 2004. Accessed July 2009. `http://guardian.cs.utu.fi/knuutila/Courses/ML/Default.htm`.

KOTSIANTIS, S. 2007. Supervised machine learning: a review of classification techniques. Informatica (October).

KOZA, J. R. 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.

MITCHELL, T. 1997. Machine Learning. McGraw Hill.

MONTANA, D. J. 1995. Strongly typed genetic programming. Evolutionary Computation 3, 2, 199–230.

NG, A. Y., AND RUSSELL, S. 2000. Algorithms for inverse reinforcement learning. In International Conference on Machine Learning (ICML'00), Morgan Kaufmann, 663–670.