Real-time rule-based classification of player types in computer games

Ben Cowley, Darryl Charles, Michaela Black & Ray Hickey

User Modeling and User-Adapted Interaction
The Journal of Personalization Research

ISSN 0924-1868

User Model User-Adap Inter
DOI 10.1007/s11257-012-9126-z

Springer

User Model User-Adap Inter
DOI 10.1007/s11257-012-9126-z

ORIGINAL PAPER

Real-time rule-based classification of player types in computer games

Ben Cowley · Darryl Charles · Michaela Black · Ray Hickey

Received: 8 September 2011 / Accepted in revised form: 16 July 2012

Abstract The power of using machine learning to improve or investigate the experience of play is only beginning to be realised. For instance, the experience of play is a psychological phenomenon, yet common psychological concepts such as the typology of temperaments have not been widely utilised in game design or research. An effective player typology provides a model by which we can analyse player behaviour. We present a real-time classifier of player type, implemented in the test-bed game Pac-Man. Decision Tree algorithms CART and C5.0 were trained on labels from the DGD player typology (Bateman and Boon, 21st century game design, vol. 1, 2005). The classifier is then built by selecting rules from the Decision Trees using a rule-performance metric, and experimentally validated. We achieve ∼70 % accuracy in this validation testing. We further analyse the concept descriptions learned by the Decision Trees. The algorithm output is examined with respect to a set of hypotheses on player behaviour. A set of open questions is then posed against the test data obtained from validation testing, to illustrate the further insights possible from extended analysis.

Electronic supplementary material The online version of this article (doi:10.1007/s11257-012-9126-z) contains supplementary material, which is available to authorized users.

B. Cowley (✉)
Cognitive Science Unit, Department of Behavioural Science, University of Helsinki, PL 9, 00014 Helsinki, Finland
e-mail: ben.cowley@helsinki.fi

D. Charles · M. Black · R. Hickey
School of Computing and Information Engineering, University of Ulster, Coleraine, Northern Ireland

B. Cowley
Center for Knowledge and Innovation Research, Aalto Business School, Aalto University, Helsinki, Finland

Published online: 22 August 2012

Keywords Player typology · Player profiling · Computer games · Decision trees · Classification · Experimental validation

1 Introduction

In player-centred computer games it is valuable to leverage, as much as possible, the current understanding of human cognition and emotion. Obtaining a better understanding of the differences between players can help substantially in the design of games, and accurate player modelling may be used to create adaptive real-time game mechanics. In recent years, the importance of player modelling in commercial games has grown greatly, and many companies now data mine to support this technology (Grimes 2007; Herbrich et al. 2007). Applying *game metrics* to create a player model may have many purposes, basic and applied, including implementation of persistent player profiles (e.g. in MMOGs such as *World of Warcraft*, Blizzard 2005) or to aid investigation of the psychology of play (Mountain 2010). In either case, a player model will only be as good as the theoretical model that supports it and the learning model that instantiates it.

However, as may be surmised from a reading of the literature, player behaviour can be complexly varying and difficult to interpret (Caillois 1961; Huizinga 1949; Salen and Zimmerman 2004). Various models of player psychology have been put forward to facilitate the classification of different player groups, with varying degrees of success (Bateman et al. 2011), including the Demographic Game Design (DGD) typology which we have used (Bateman and Boon 2005). On the other hand, approaches that use machine learning (ML) methods have shown that learning a non-theoretical model of player behaviour can still produce reasonable results (Drachen et al. 2009). Therefore, combining the psychological and ML approaches holds the potential to surpass either approach alone, and yet to our knowledge this has not been decisively explored. Our contention is that, using a psychologically-derived theory of player type as the target of a supervised ML algorithm, one can build a real-time classifier and obtain useful accuracy results.

Thus we demonstrate a method of player modelling by learning class labels from a player typology, and deploy the output as a real-time classifier in the test-bed game Pac-Man. Pac-Man was chosen for the many advantages it offers when player modelling—in other words, when building features of play. As a well-known example of the 'predator-prey' genre of games, the player motivation will be quite clear at almost all times. The control scheme is simple. The game space is completely represented onscreen so player confusion is unlikely. These aspects help ensure that subjects spend their time playing, and not confused or stuck. However, simplicity of elements does not preclude complexity of play, which must be possible to allow differentiated player behaviour—Pac-Man contains both immediate and evolved tactics. The 2D grid-like game space allows efficient representation and reasoning, and although agents in the game move in real-time, their movement can be quantised in the metrics, allowing certain turn-based logic such as tree search to be applied.

Our classifier is built using ML methods from the robust Decision Tree (DT) class, which were chosen after a comparative analysis of DTs, Radial-Basis-Function Neural

Networks (RBFNs) and K-Means clustering. DTs were in fact a logical choice because they give good concept descriptions, which are communicable and help the investigator derive applications that retain fidelity to the psychological model of player type. The rules derived from the learned trees were filtered by their performance and implemented within the Pac-Man game engine. This is a relatively complex multi-stage process, and we summarise these steps briefly in order below.

1. Materials of the experiment were created:
    a. Bateman and Boon's DGD questionnaire was adapted.
    b. A nonstandard version of Pac-Man with game-play logging was developed.
    c. Log and DGD data was collected from 100 players.
2. Methods were implemented to learn from the data:
    a. Observation and deconstruction of Pac-Man game-play led to a list of behavioural traits and relevant to those, game metrics/features were devised.
    b. A set of hypotheses of player-type behaviour was proposed.
    c. Feature data was used to learn ML models of the play behaviour for one dichotomous or binary class, *Conqueror/Not Conqueror*.
    d. DT results were chosen as the best performers, and trees and rules were ranked by performance.
    e. A subset of best-performing rules was extracted.
    f. We specified scope and conflict resolution policies, combinations of which defined a number of potential classifiers.
3. The classifiers were evaluated:
    a. A new set of game log and DGD data was collected from a further 37 players.
    b. Based on accuracy results over this test set, the best performing classifier was chosen.
4. Analysis of results was performed to test our proposed hypotheses and generate new insights that could lead to new and improved models in future work.

The structure of this work has been strongly motivated by the target audience—while it is an academic work, the motivation is to also serve practitioners such as game designers. For instance it is notable that this approach has many qualitative elements which may reduce the generality; but qualitative methods may be appropriate in a practical context where the level of concrete knowledge (knowledge of the game and its players) is high. Similarly, the ML methods we use are not groundbreaking, but they are accessible to the ML layperson and they are transparent, in that the output can be easily 'read'. The latter point is of high importance: as we mentioned above, communicable concept descriptions can aid the interpretation and application process—in Sect. 5 we demonstrate the point. While one may always utilise rule extraction methods such as G-REX (Johansson et al. 2004) on more opaque ML methods but this might not be so appropriate in a practitioner's context where the level of abstract methodology knowledge might not be so high. This is not done to undersell the talents of practitioners, but merely to pitch the work at an 'entry level' audience.

In the next section we review the background and relevant state of the art. Then we describe the classification experiment methodology in Sect. 3, and present our results in Sect. 4. Section 5 gives our analysis of the results while Sect. 6 is a summarising discussion. Finally, Sect. 7 concludes the paper. Supplementary material is given in Appendices A–D.

2 Background

We build our thesis on the assumption that every player of games has to possess some form of play-related 'personality'. By this we mean that the act of playing requires *adopting an attitude* to the game being played that constitutes a personality, even in the case of Artificial Intelligence (AI) players. The act of play requires commitment to a course of action that ends with an invested outcome, winning or losing being the most common type of outcome. This *commitment* of a player, and their particular *style* in undertaking the play actions, forms most of their play personality. Style of play is used here to encapsulate modes of cognitive processing which can differ in the approach to play tasks. A play personality is not to be thought of as static but quite contextualised and relative.

Considering one's own style of playing games (be they computer or traditional games) can help illustrate these points for the case of human players. One may play 'just for fun' or 'for keeps' and may prefer to strategise or intuit, deliberate or 'dash about'. In the case of an AI player, their commitment would presumably be indefatigable and their style, perhaps rational if rule-based, or stochastic, but less probably heuristically close-to-optimal as for a skilled human player (Acuña and Parada 2010). This may be quite un-human-like, but it is still recognisably an attitude toward the game, and thus constitutes a personality in the context of play. Eliciting insights from observable indicators of the building blocks of player personality is thus the main challenge of player modelling.

We consider our player modelling approach as one primarily aimed at facilitating *player-centred design* with a method that supports ease of interpretation and adaptation of results. *User*-centred design (Katz-Haas 1998) arose in domains other than entertainment such as Technology Enhanced Learning (TEL). These systems in fact often use the game-play paradigm of interaction (Cowley et al. 2011; Malone 1980; McGinnis et al. 2008), and rely heavily on user modelling techniques (Beal et al. 2002; Zhou and Conati 2003). In recent years there has grown a rich literature on modelling players of entertainment games; for reviews see Bowling et al. (2006), Fürnkranz (2010), and Galway et al. (2008).

2.1 Classification

Methods to support the understanding of player behaviour generally involve some form of pattern recognition from either labelled or unlabelled training data. Kaukoranta et al. (2004) and Kaukoranta and Smed (2003) provide a high-level overview of the application of generic pattern recognition in games. Provision of the labels is the reason we use a player typology—although there are difficulties inherent in a supervised

approach (since the solution space is constrained by the training data), for our design-oriented method we favour these difficulties over those of an unsupervised approach, i.e. interpreting algorithm output which has no labels.

Acuña and Parada (2010) attempt to model the behaviour of expert players' through data mining (DM), in order to show that such modelling could find novel solutions to 'NP hard' problems. Others have used DM to look for the distinguishing characteristics of expert play. Fu et al. (2004) implemented this with DTs. The algorithmic basis of DTs can be traced back to Breiman (1984) and has been used widely in games (Levillain et al. 2010). Geisler (2002) used a trio of ML algorithms—Decision Trees, Artificial Neural Networks and Naïve Bayes Classifiers to model feature vectors from the play of a single expert over an hour of play. In a very similar vein, there is some work on modelling player behaviour that has the goal of lending a game-controlled avatar some of the behavioural *characteristics* of a human player. The approach seems popular in the racing game genre, both because the set of inputs to learn is relatively small and also good training data and fitness functions exist. Both Togelius et al. (2006) and Chaperot and Fyfe (2006) attempted to model human players of racing games using neural networks in order to build better automatically controlled drivers. Commercially, a similar method was implemented in the driving games *Colin McRae Rally* (Codemasters 1998) and *Forza Motorsport* (Microsoft Game Studios 2005), in order to train their AI to mimic the playing style of the human player (Tipping and Hatton 2006). In a similar vein in a multiplayer setting, Thurau and Bauckhage (2006) used Hidden Markov Models for behaviour recognition in team soccer games. Tveit and Tveit (2002) have suggested a process for DM Massively Multiplayer Online Game (MMOG) usage, inspired by existing processes for web usage mining. Kennerly (2003) outlines the key steps required when DM an MMOG, with the motivation of improving cyclical game design updates through player analysis.

Our particular use for modelling player actions (through DM) is to uncover the player's behavioural *preferences*, not just noting what they do, but attempting to infer what they prefer to do. In previous similar work, Johansson et al. (2006) used a definition of player types drawn directly from descriptions in the poker literature (Jones 2000; Sklansky and Malmuth 1999) of how poker is played. The DT algorithm J48 (based on C4.5 (Quinlan 1993)), and rule extractor G-REX (Johansson et al. 2004), were then used to build descriptions of how the differing types of poker player behave in online games. This approach differs to our own in that the authors have constructed their typology without reference to a surveyed sample of players. Thawonmas and Ho (2007) used a classic MMOG typology (Bartle 1996) to model players, although their data is derived artificially, i.e. their player logs are in fact based on artificial agents. Wong et al. (2009) derive a typology of play styles for 2D shooting games, also based on the Bartle typology. They use self-organising maps to learn the differences in behaviour shown by real players. The clusters that result are manually mapped to the Bartle types, and players are classified based on their proximity to these clusters.

Recently two approaches have tried to do player modelling by learning patterns of behaviour from the play logs. One approach, which is particularly relevant to our work, is that of Baumgarten (2010), who used Linear Discriminant Analysis (LDA) to group a set of Pac-Man players into similar playing styles, and then interpreted their activity by examining the linear combinations of features that define the axes in

the LDA space. In a superficially similar approach, Drachen et al. (2009) analysed a large data set of complete games of *Tomb Raider: Underworld* (Eidos 2008), using Emergent Self-Organising Maps (ESOMs) to cluster and visualise a set of six game metrics. The clusters enabled classification of players into four types, labelled by the manual process of examining the typical game-play behaviour associated with each cluster.

All the cited work had slightly different aims than our core focus of classifying real players by an empirically-derived typology in real-time. As far as we are aware, this is the only automated classifier of its kind in this domain.

2.2 Player typology

As stated above, players exhibit some form of play-related personality, which may be described using a typology. Player typologies aim to classify the variation in abilities and temperament that impact on the play experience—thus they are an abstracting model. The player's personality may relate differently to each given game's content [or other contextual factors such as presence of other players (Ravaja et al. 2006)]; thus they may express their play-related personality as entirely different types on different occasions. This reflects the fact that typologies have been shown to be more valid as descriptors of trait-variances rather than dichotomies (McCrae and Costa 1989), as players move within continuous dimensions of personality. This may raise the question of how any given set of game-play data (such as our data) can represent a player type since it might reflect one or another type at different times during play (over one or more games). Naturally this is a key question for type classifiers, the answer to which lies in the ability to handle concept drift, which we touch on later in Sect. 5.2.2 Thus, with a modelling method capable of handling concept drift (Black and Hickey 1999), typological classification can be a useful basis for player modelling.

There are only a small number of player typologies; for a review, see Bateman et al. (2011). Bartle's typology is perhaps the most well-known, developed from observation of the players of Multi-User Dungeon (MUD) online social games pioneered in the late 70s (Curtis 1992). There are four types of players specified in this work: Achievers, Explorers, Socialisers and Killers. These types describe the play attitudes expressible within a MUD environment. Given the contextualised nature of play preferences, Bartle could be considered unsuitable for more general applications; what was required for our modelling approach was a typology of players that was applicable to any kind of game and preferably based on a wide set of empirical observations. After careful review we chose the DGD typology (Bateman and Boon 2005), illustrated in Fig. 1. This figure shows the distribution of types across the sample which was surveyed in the original work to build the typology. The values on the ordinate are actual numbers of respondents.

The DGD typology has taken the Myers Briggs Type Indicator (MBTI) and Temperament Theory approaches as its basis in describing player types and their associated game-play-preferences over four categories, which are each two-valued. The MBTI (Ludford and Terveen 2003) describes one's instinctive preference for modes of thought and behaviour. MBTI could be said to derive from Temperament theory

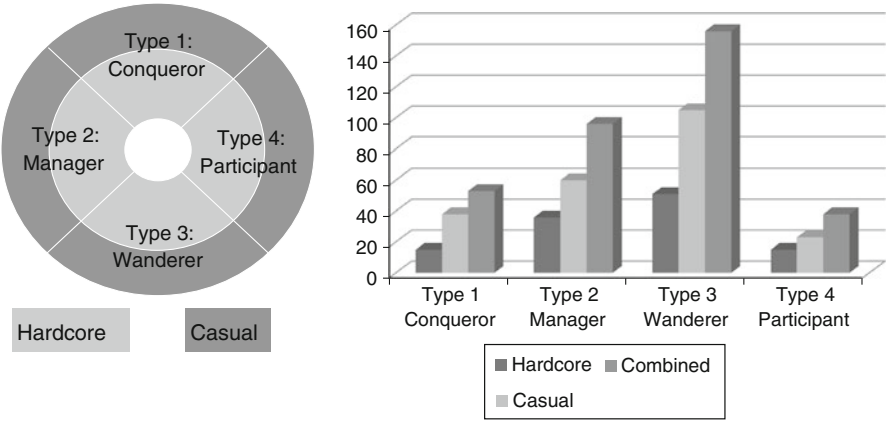

Fig. 1 DGD types and distribution across survey respondents (in actual numbers of respondents); adapted from Bateman and Boon (2005)

(Jung 1971), and is referenced by Keirsey and Bates (1984). The MBTI is a controversial instrument since despite strong empirical support it has theoretical issues: namely that, despite having construct validity, a broader assessment found insufficient evidence to support all of the claims made about MBTI (Pittenger 1993). The deeper implications of this on the DGD typology are covered in Bateman et al. (2011).

As is typical in typologies, players will belong to each of the DGD types to a greater or lesser degree, because type membership is *non-exclusive*. The types can be briefly described as follows:

1. *Conqueror*: Competitive, win-at-all-costs. Players of this type are goal-oriented and enjoy feeling dominant in the game or in social circles set around the game.
2. *Manager*: Logistical, plays to develop mastery. Such players are process-oriented and will replay completed games if they can use their newfound mastery to unearth novelty at deeper levels of detail.
3. *Wanderer*: Desires new and fun experiences. Less challenge-oriented than the above types, these players primarily seek constant, undemanding and novel enjoyment.
4. *Participant*: Enjoys social (living-room) play, or involvement in an alternate world.

In this typology there is a further cross-type split between dedicated or hardcore players, and recreational or casual players.

3 Materials and methods

Our method involves matching logs of game-play to labels of player type, using supervised ML algorithms whose output can be interpreted, to provide the researcher/developer a picture of player behaviour for that game.

Although the ultimate goal would be to learn the four DGD types, learning multi-class targets in a high-dimensional search space (i.e. the possibility space of player

behaviour) runs the risk that algorithms would not converge to an accurate discriminator. Our approach is thus to consider binary targets: classifying only a single type, with all records not of that type classified as an 'anti-type' class (Mitchell 1997). Ultimately we should thus derive four binary targets, and when classifiers for all four such targets were learned, they would then be run in parallel and a conflict resolution strategy used to pick one final answer. However, for the duration of this paper we will evaluate the binary target *Conqueror*/*Not Conqueror*.

We begin with *Conqueror* because as a play style it may have the best match with the Pac-Man test-bed alongside *Manager*—which we did not choose because it was not so well represented in the recorded sample. While the other types are not equally well-suited to Pac-Man play, this does not invalidate the approach because a type does not represent a particular player at all times—players may adopt the type that suits them *in the context of a particular game*. Thus it is really only relevant that the game can recognise the type *that a player is expressing at a given time*, and react to it. Indeed, what type the player expresses another time is of much less importance.[1]

We list in Table 1 a set of hypotheses about the probable behaviour of Pac-Man players, split between *Conqueror* and *Not Conqueror*. These were based on our DGD typology and Pac-Man descriptions. The *Not Conqueror* set was obtained by first creating hypotheses for each type (other than *Conqueror*) alone, joining them as one set and then removing any overlapping or mutually conflicting hypotheses. For each hypothesis, the corresponding null hypothesis is simply that the opposed type of player would *not* pursue such play behaviours (which is not to say that they would pursue opposite kinds of behaviours—omission is sufficient).

Thus the key essence of the hypotheses is that the types would differ mainly on their approach to risk: the *Conqueror* should differ from the *Not Conqueror* by chasing Ghosts more, and by being less cautious when doing so. They may have shorter level-clearance times, and have speedier games overall.

3.1 Materials: Pac-Man game and questionnaire

A Pac-Man game was constructed as a test bed for the experiments, and is an interpretation of the original Namco game rather than a clone (see Fig. 2). In Pac-Man the goal is to move around the game level and obtain all the collectables, thus progressing to the next level. Points are awarded for collectables and eating Ghosts. A full list of game rules is given in Appendix D.

Ghosts move according to a pseudo-random probabilistic control function based on the game state, the (increasing) difficulty level and Pac-Man's relative location. In the normal game state, the probability of moving closer to Pac-Man is higher than the probability of moving to any other adjacent point. The direction is reversed when the game is in the Hunt state.

Compared to the original Pac-Man ours is very similar, except for the control interface where we used keyboard rather than joystick. Also with the control function

[1] This is also a reason to avoid typologies designed for online role-playing games, which are designed to track players who remain largely similar over instances of play, *because they are playing a role*.

Table 1 Our hypotheses on the behaviour of DGD player types in Pac-Man

Conqueror
This is our key type, as it assumed to provide the most easily distinguishable characteristics of play in Pac-Man. It is expected that *Conqueror* type play revolves around the **conflict with the Ghosts**. Ghosts provide the heart of the challenge, and as such would be the focus of the *Conqueror's* **fiero-seeking** drive to overcome obstacles. Thus we propose five testable hypotheses:
H1. The *Conqueror* will try to **hunt as many Ghosts** as possible at one time.
H2. The *Conqueror* will aim to clear more levels, faster—manifesting as faster clear times.
H3. The *Conqueror* will show less caution, and be more likely to take risks. Thus **average distance from Ghosts will be lower** (or distance will equal 1 more often).
H4. The *Conqueror* will make more attempts to catch Fruit when it appears.
H5. The *Conqueror* returns to the game more often in the face of defeat.
Not Conqueror
Unlike the *Conqueror*, the strategic approach for high scoring is to treat Ghosts cautiously, not chase them too far, to **prolong the game and score from Dots**. The key here is that our Pac-Man **Ghosts are not predictable**; they operate on probabilities (based on a time-seeded pseudo-random generator). **Getting near them is risky**. Thus the best strategy is to **lure the Ghosts to the Pill**, waiting beside it until they are in close proximity and then chase them. A non-strategic player, meanwhile, might simply play cautiously to clear new levels. Play would thus be straightforward, with little attempt at strategy or risky manoeuvres.
Thus we have predicted:
H6. The *Not Conqueror* has longer level clearance times.
H7. The *Not Conqueror* performs more backtracking.
H8. The *Not Conqueror* keeps the Ghosts at a distance.
H9. The *Not Conqueror* does more waiting, especially in proximity to a Pill.
H10. The *Not Conqueror* does not perform risky runs to collect the Cherry.
H11. The *Not Conqueror* **does no extensive chases after Ghosts** when in Hunt mode.
H12. The *Not Conqueror* has a shorter overall game length.

Bolded expressions emphasise the most important points

for Ghost movement we use probabilistic controllers, whereas original Pac-Man had a set of simple rules (which ultimately made Ghost movement deterministically predictable).

Players were labelled according to the DGD typology after answering a Questionnaire instrument based on material from Bateman and Boon (2005). This instrument gives 16 five-point Likert scale statements or questions (see Appendix C), where each question corresponds to one of four factors: *Conqueror*, *Manager*, *Wanderer* and *Participant*. The agreement expressed with each question is added to the score for a player type to generate a play style profile for each player.

3.2 Learning from Pac-Man players

To obtain a machine-learned classifier we collected data from 100 players, logging their Pac-Man games and recording play preferences. The protocol followed three steps: first players were located in a computer lab and briefed on the requirements and procedure. Secondly they answered a questionnaire to give their play preferences

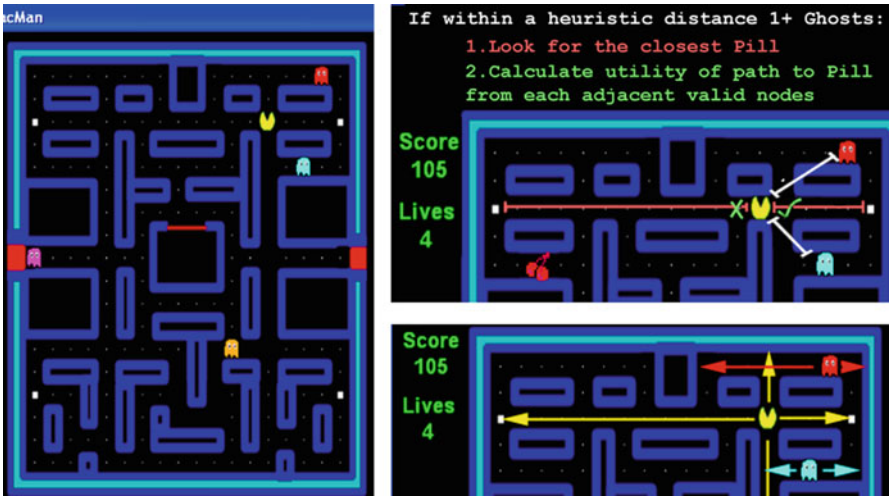

Fig. 2 Annotated screenshots from a Pac-Man game level. *Left panel* the entire map, in the course of being played. *Right panels* (*top* and *below*): the game screen is overlaid with metrics of game-play, such as distance from Pac-Man to Ghosts or to Pills

and background information (age, gender, skill level in Pac-Man, Casual/Hardcore/No idea). Thirdly each player was asked to play Pac-Man three times, two practice games and a test game, to diminish the difference between experienced and inexperienced Pac-Man players. Only games of at least two levels were accepted as either test or practice games; only data from the test game was used for modelling. All software was pre-loaded and data recording was automated, restricting participant effort to the game.

The distribution of types over our respondents was {*Conqueror*: 45, *Manager*: 24, *Wanderer*: 19, *Participant*: 12}—thus the *Conqueror* type dominates and using it in a binary target implies a default of 55 % (i.e. 55 % of the sample will be of one type). Moving on to the distribution over other values, there were 77 **males** and the age range was from 17 to 41 (mean: 21.4). Demographic information suggests a bias in the respondent set toward non-expert casual players {*Expert*: 8, *Intermediate*: 14, *Beginner*: 69, *Newbie*: 9} and {*Casual*: 59, *Hardcore*: 31, *No Idea*: 10}. Note that these demographic attributes are only mentioned for completeness as they had negligible impact on the results. This may have been because they contributed no information regarding game-play behaviour, perhaps due to our protocol for data collection which had practice games to make all players more equal in skill.

In order to describe player behaviour, we built a set of game metrics, including simple game engine variables and more complex features. The first step in this process was to create a list of high-level behavioural *traits* that can be expressed, either positively or negatively and to varying degrees, in most games (certainly in Pac-Man) and by most players. The list comprises Aggression (forward and hasty action with respect to opponents or obstacles), Speed, Caution (guarding against risk to failure), Planning (achieving higher level goals with a premeditated, linked series of actions), Decisiveness (lack of backtracking or vacillation), Thoroughness (attempting to do

Table 2 Example features of game-play

Speed	Natural language and pseudo-code descriptions
Sp1_Average states	Count how many moves it takes the player to clear a level
Sp2_Average cycles	Count how long in computer cycles it takes the player to clear a level (because. cycles are hardware specific, they relate more precisely to player actions than does time, since player actions are limited by the hardware's own speed of execution)

These two features primarily indicate a player's speed of play, and thus may be related to several of the traits—but primarily Aggression. Other features may relate more precisely to single traits

or see everything available), Control Skill (fine, precise and detailed command of control schemes) and Resource Hoarding [these traits are further defined in Cowley et al. (in preparation, a)]. Each member of this list was then matched to a number of features, which were derived by analysis of the game-play (both observational and deconstructing the mechanics). Analysis allowed us to build a natural language description for each feature and then quantise it with a function (generally of first-order logic) over the state-space of the game-play log. These natural and functional descriptions are illustrated by the simple example features in Table 2, while a full list is given in Appendix A.[2] Although it is not in scope to discuss these DM details here, all these steps are described in more detail in Cowley et al. (in preparation, a). These feature-based descriptions of player behaviour were applied to individual levels, and then all level-specific data were averaged over the duration of the game. For the final step in pre-processing, the data were normalised (by division by largest member), and appended to demographic data and the DGD type labels for the players. This resulted in one 'feature-vector' for each of the 100 players, forming the train-and-test data-set we used to learn classifiers for the DGD labels.

3.2.1 Learning from features of Pac-Man game-play

We used the DM program SPSS Clementine (now known as SPSS Modeller) for its implementations of ML algorithms CART, C5.0, RBFN and K-Means. CART and C5.0 were chosen because of their facility for handling categorical as well as continuous features, and have been shown to perform well by comparison to other DT algorithms (Anyanwu and Shiva 2009). RBFN and K-Means were chosen as exemplar alternative ML approaches to test whether the DT approach was deficient in accuracy. The 'feature vectors' described above were fed to these algorithms and the results for all four were compared for accuracy and interpretability. We ran each algorithm with a battery of parameter settings designed to explore the solution space and obtain best accuracy (parameters are described in the documentation for Clementine, but briefly they were: for CART, feature importance and tree depth; for C5.0, Boosting, cross-validation, pruning and winnowing). In short, the valid range (or categories) of each

[2] Thoroughness and Control Skill traits generated several features, but none of these feature survived the feature selection process to be included in the methodology—thus they are not included in the paper.

parameter was explored step-wise, holding all other variables fixed, generating several hundred DTs.

Train and test sets were randomly extracted from the 100 vectors in a 2:1 ratio, with 7× replication of every parameter setting, such that seven runs were made for each setting with randomly selected train/test sets. Sets did not overlap except where the C5.0 algorithm cross-validation setting was enabled. All seven test results from each parameter were averaged to get the accuracy. With this approach, the best accuracies obtained were: CART—59 %, C5.0—61 %, RBFN—54 %, K-Means—43 %.

This superior accuracy result reinforced our choice of DTs. Based on the exploratory analysis described a large collection of DTs was obtained. Converting this collection to an ensemble of rules, which can be seen in full in Appendix B, a heuristically optimised classifier was built by selecting a 'best bet' rule set, as described next.

3.2.2 'Highly-ranked' rule set

In order to represent what our DTs learned about the labels {*Conqueror*, *Not Conqueror*}, and to ensure we only had rules of good quality, we built one rule-set for each label: rules were taken from the best performing DTs by a rule selection policy. There are two main principles behind not simply using all the rules from every DT. Firstly, there is real-time efficiency. We need to ensure computational parsimony of the classifier, because in the middle of game-play, the less features calculated in order to provide data for a rule set, the more computational resources are free for other purposes. This is a general principle of AI for commercial games, where the emphasis of computation allocation is on aesthetic aspects.

Secondly, cutting rules from a tree also enables weak-feature pruning, because excluding a rule will also exclude features which occur **only** in that rule. The DT algorithms act to test and relegate the least informative features. Thus a feature, which (for instance) only occurs in one rule which classifies two individual cases, represents either an instance of non-descriptive data or outlier behaviour (within the scope of our data set). It is likely that describing such outlier behaviour would imply over-fitting, thus it is generally good to find and exclude corresponding features.

In addition, the rule extraction is empirically justified by comparing the poor results obtained for the DTs during the initial train and test, 59 % for CART and 61 % for C5.0, against the results given in Sect. 4 below which depended on an extracted rule-set.

The primary measure for the quality of a DT rule is the size of the leaf node, i.e. how many records it holds; and how pure it is: in our case, we define this as the proportion of the leaf that the majority class[3] occupies (other definitions include, e.g. entropy). Leaf purity and size are respectively defined as *confidence* and *support*. In this work we prioritised the relative support of the leaf, that is, how proportionally representative it is. We define this as *support* divided by *total number of records* in the test set. Thus a rule with support 31 for a set of 61 players is about one-fifth better than a support 31 rule for 75 players. Support tends to diminish as trees get deeper and purity increases (although purity may increase spuriously, e.g. a leaf node containing one record is

[3] That is, the class which represents the majority of records at that leaf. If only a single class is represented in the leaf, the corresponding rule is a 'pure' representation of that class, and thus the leaf is very pure.

Table 3 The ranked value of a CART rule

Rule ID	Support	Split	Relative support	Confidence	Rank
Not_Conqueror 6	21	61	0.344262295	0.952	0.327737705

completely pure). Thus, if a good rule could be found within a shallower depth (and consequently a smaller number of features) it may be reasonably thought of as more generally applicable than a rule from a greater depth, which may be over-fitted. It will also require utilising less potentially unreliable features during deployment, an issue discussed in detail elsewhere (Mitchell 1997).

3.2.2.1 Rule ranking. The first step in extracting the best rules was to extract the best trees. Our exploratory analysis (see Sect. 3.2.1) of CART and C5.0 algorithms generated several hundred trees with over a thousand rules. From these, we took only those trees with positive lift[4] (Tuffery 2011) over the default of their individual test set. Each tree was ranked based on its relative accuracy using the following policy: the squared difference between the tree accuracy and the mean accuracy of all trees which used the same parameters (i.e. 1 replication) was divided by the squared difference between the overall default and the default for that replication. Formula 1 illustrates this valuation, and was used to pick out 52 trees with an average of five rules each.

$$\text{TreeRank} = \frac{(\text{TreeAccuracy} - \overline{\text{TreeAccuracy}})^2}{(\text{OverallDefault} - \text{ReplicationDefault})^2} \tag{1}$$

In order to have a minimum standard for rules, we set some exclusion thresholds for support and confidence. Rules with a very low support can be regarded as 'case unproven' and rules with a low confidence do not discriminate very well between two class values. These thresholds were initially set very high—minimum confidence to 0.9 and minimum relative support to 0.2—and then iteratively lowered to obtain a useful number of rules. One might visualise as increasing the union between two sets. Setting high thresholds for both measures excludes any rules with very good score in only one. Rules were ranked according to their relative support and confidence, as in formula 2 below.

$$\text{RuleRank} = \text{Support/Test Set Size} \times \text{Confidence} \tag{2}$$

For example, a rule from a tree using the 75:25 train and test split, that classified 45 records, would have a relative support of $45/75 = 0.6$ (which is relatively high). That rule classified 34 Conquerors and 11 *Not Conqueror*s (as Conquerors), which means its confidence was $34/45 = 0.756$. Thus we gave it a rank of $0.6 \times 0.756 = 0.4536$. In this way we specified a valuation for all rules produced using CART and C5.0. Table 3 illustrates an example ranked rule, where Rule ID is the target classified with the rule number.

[4] Lift is defined as the ratio of response in the target group compared to the average, i.e. the default value.

Table 4 The chosen rules which have a single condition

Rank	Rule #	Target	Support	Confidence	Rule condition (averaged data)	Rule
0.2353	15	Not Cq	19	0.842	if Lives_StdDev $>$ 0.597	then Not Cq
0.2133	20	Not Cq	14.22	0.915	if D2_Player Vacillating $>$ 0.678	then Not Cq
0.2132	21	Not Cq	14	0.929	if A4_Hunt Even After Pill Finishes $\leq$ 0.212	then Not Cq
0.1618	32	Conqr	12.99	0.847	if Points_Max $\leq$ 0.446	then Conqr
0.1475	36	Conqr	9	1.0	if A4_Hunt Even After Pill Finishes $>$ 0.563	then Conqr
0.1475	38	Not Cq	10	0.9	if Sp2_AverageCycles $>$ 0.573	then Not Cq
0.1333	39	Not Cq	10	1.0	if Cherry Onscreen Time $\leq$ 0.137	then Not Cq
0.1067	48	Not Cq	8	1.0	if C5_Moves With No Points Scored $>$ 0.584	then Not Cq
0.1065	49	Conqr	7.219	0.9	if P6_Put Off Dots Near Ghost House $>$ 0.656	then Conqr

The final set comprised 50 high-performing rules. We have illustrated a sample of (shortest) rules in Table 4 the full list of rules appears in Appendix B. These short rules help illustrate the ease of interpretation rules can give, as the fact that they use one Boolean split of a value to classify a binary target means they describe a quite unambiguous hypothesis with respect to their feature. They are listed in order of rank.

3.2.2.2 Overview of rules. We examined the frequency of features within the chosen rule set to evaluate their match to our hypotheses. Based on this analysis, the overall trend of these highly ranked rules is to distinguish between different approaches to the generic behaviours of Aggression, Caution and Speed (Planning is represented to a lesser degree). Features like *A4_Hunt Even After Pill Finishes, Lives_StdDev* and *Sp2_Average Cycles* dominate in terms of frequency of use. The top 16 (or ∼50 % of) features were used three times or more. Of these 16 most-used features, four had to do with how lives were gained and lost, four dealt with aspects of the Hunt mode, four were associated with Caution, and two with Speed (the last two dealt with other aspects of play). Therefore on first inspection we would say that this rule set, targeted on the *Conqueror*/*Not Conqueror* classes, supports our hypotheses regarding the *Conqueror* tendency to concentrate on Ghosts and to play quickly with incaution. Section 5 furthers this approach to building insight from rules.

It may be noted that the rules tend to be quite short; the majority have fewer than four conditions. Although we noted earlier that short hypotheses *might* be an advantage, we didn't choose rules based on this principle, but on their coverage and accuracy. The higher ranking of shorter rules reflects the higher performance of shallow, and therefore general, trees. In our complex data space and small testing data sets, longer and more 'fitted' rules either missed many players, or included too many by not discriminating. Having said that, good long rules may still exist—such as our highest ranked rule with five conditions—they are just rarer.

3.3 Classification experiment methods

Prior to describing the results of rule testing in Sect. 4 we will briefly discuss implementation: in Sect. 3.3.1 we address the particular issues involved in implementing the

rules within the Pac-Man game engine, and in Sect. 3.3.2 the validation experiment format is described.

3.3.1 Rule set implementation

In deployment the values of features and output of rules were derived level-by-level. This was done for practical reasons, because firstly the history of a level cannot be held in working memory indefinitely, it must be written out. Therefore the classifier must utilise the level state history before it is saved to disk and its memory de-allocated. All that then needs to be stored is the values of features, which can be retrieved when required so the value for each feature, *averaged* over some interesting number of levels, is readily available for the rule set classifier. Secondly, it is desirable to have a classifier that works on any length of game, even if the player only lasts one level. This left two main issues to resolve: the scope of previous levels to be considered when classifying, and the resolution of conflict between different rules in the set. Both are discussed in detail in the two next sections.

3.3.1.1 Scope of the rule set over prior levels. The operational time-frame for the classifier was set at one level. This works because in principle Pac-Man is a modular repetition of a single level design with some variation in speed of events and behaviour of Ghosts over time. Although most features are calculated over a handful of states (to capture an individual play event), features are evaluated only at the end of a level because there can be little certainty in conclusions derived from too short a time-frame; rather they should rely on aggregates of player behaviour such as the reaction of a player to all four Pills in a level (not just their reaction to one Pill). However we may want to make classifications based on feature-averages from more than just a single-level at a time, since average values over multiple levels might fire a different rule subset. This approach enables evaluation of the player type in the moment where it matters most, but also as an aggregate which is necessarily only an approximation of the truth but may still be useful (much like demographic data on the shopping public is useful to supermarkets). This creates the issue that our classifier must consider more than one scope of levels.

The scope is equivalent to the value of a feature representing some number of levels: from the value of the last level only, to the average of all levels played, or any window size in between. The classifier's scoping system is defined over windows of prior levels, with labels (in bold) as Window of 1 level: **single-level**; Window of three levels: **average-of-3**; Window of all levels: **average-all**.

Features which have been averaged over multiple levels will thus appear in most rules as real numbers, such as the number of Cherries eaten per level which could be either zero or one per level, thus varying in the range 0–1. However due to the discrete nature of some of the features, **single-level** rules cannot use the same values as for a larger window. This is because some behaviours occur at most once per level, so the corresponding features such as *P6_Put Off Collecting Dots Near Ghost House* can have value 0 or 1; and some behaviours occur in a discrete rather than continuous range. For instance Pills are eaten four times per level, so *A4_Hunt Even After Pill Finishes* can have only the values {0, 1, 2, 3, 4}. Thus to make the **single-level**

implementation of the rule-set operable, we rounded off those conditions which were real-valued but which used features that must be integer or binary when measured for only one level.

3.3.1.2 Conflict resolution for the rule set. Deploying a classifier built out of multiple learned discriminators, such as a rule set, requires a method to resolve conflict between rules. To resolve any conflict between the rules, we looked to the statistics with which they were ranked and selected.

Each rule was weighted by the value of its importance in the ranking (in the range 0.1–0.45). *Not Conqueror* rule weights were sign-inverted so they would return a negative value. There was statistical justification for weighting according to our rule ranking, since higher rules applied more uniquely (confidence) to more players (support).[5] The two methods we tried were: counting a vote from each rule, or counting only the first rule to fire; each method could itself be implemented in two ways, giving four resolution options.

By counting all the rules that fire as equal (positive for *Conqueror* and negative for *Not Conqueror*), the majority decides the type classification: an **equal-voting** system. In **equal-voting** the difference between the count of *Conqueror* and *Not Conqueror* firing rules provides the classification. Yet this ignores the rank of rules, so an alternative is to sum the rule-ranks as weights to determine the type: a **weighted-voting** system. The **weighted-voting** method would increment (for *Conqueror*) or decrement (for *Not Conqueror*) an aggregate variable by the rank-value of each firing rule. The final value of the aggregate would be positive or negative, and thus provide the classification.

Voting methods run a small risk of arriving at no conclusion, i.e. a result of 0 (nevertheless the record of which and how many rules fired for and against could still be useful). An alternative approach is to take the classification from the first rule to fire in a sequence ordered by ascending or descending value of rule weights. These methods are **first-rule-to-fire**: the rules are evaluated in an order based on their rank, and the first to fire decides the classification. The obvious ordering follows the ranking of the rules, highest down to lowest rank, but we also tried the inverse ordering. This is because the ranking of rules was largely influenced by rule support, meaning that the highest ranked rules tended to be the most general, and thus potentially firing for a greater range of players. Such an ordering has a chance of obscuring more discriminatory, informative rules lower in rank. If we propose the hypothesis that a rule's generality will be roughly inversely proportional to its discrimination in this context, and discrimination of players is a classification goal, then it is reasonable to also test the **inverse-first-rule** to fire. The **inverse-first-rule** will be the least general.

Tests and results for each method of conflict-resolution and scoping in the classifier validation are described in Sect. 4 below. Prior to this, the format of the classifier validation experiment is described in the next section.

[5] This does not necessarily mean that a higher rule more strongly represents a tendency of the player to whom the rule applies, which would imply a weak induction from statistics to psychology.

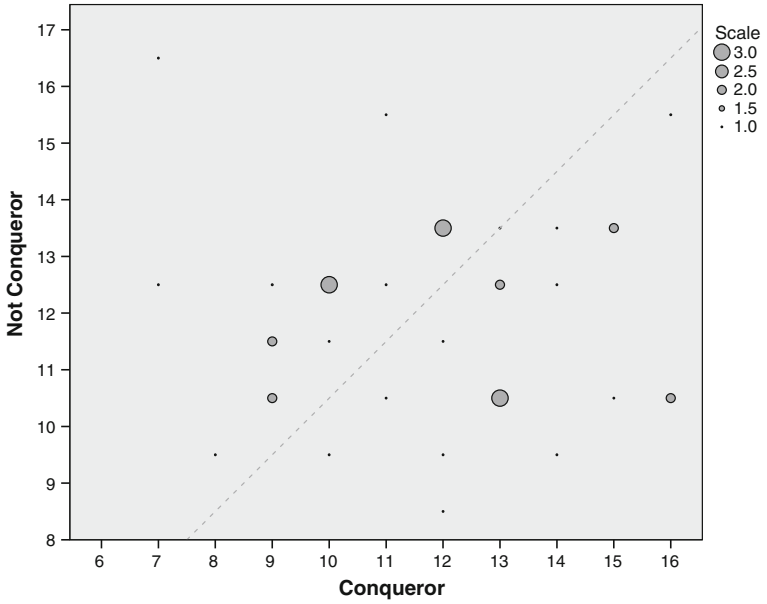

Fig. 3 Plot of every player's *Conqueror* score against their highest *Not Conqueror* score. The *dividing line* is the function $y = x$, thus points below represent *Conqueror* players

3.3.2 Experiment format

In order to test the classifier in the Pac-Man game we recruited 37 subjects to play the game, none of whom were among the respondents from the first phase of data gathering. Subjects each played at least three test games so that the average classification from all games could be used as a final result.

The data acquisition method here was similar to that described above in Sect. 3. Again, the test subjects provided data on their DGD type by taking the player type survey, resulting in a four-valued profile which was abstracted to the binary target *Conqueror*/*Not Conqueror*. The distribution was 19, or 51 % with a main type of *Conqueror*, 17 were *Not Conqueror* and one was tied *Conqueror*/*Manager*. Figure 3 plots the players over the space between their *Conqueror* and highest *Not Conqueror* scores, representing overlapping data points with a circle of area proportional to the number of players. The diagonal line separating *Conqueror* and *Not Conqueror* classes illustrates relative scores (by proximity to the data points).

Maximum score in the DGD survey data is used to define players as either *Conqueror* or *Not Conqueror*. However, in this experiment we also wished to know the relative '*Conqueror*-ness' of a player, because we anticipated that the magnitude of the difference between a player's *Conqueror* score and their other DGD scores might relate significantly to the rule set classification results. We measured the '*Conqueror*-ness' of a player by taking the difference between *Conqueror* score and highest other type score, as shown on the abscissa of Fig. 4. The entropy[6] of DGD scores on the

[6] Calculated using, $H(x) = -\sum_{i=1}^{4} p(x_i), \log_2 p(x)_i$, where $p(x_i) = x_i / \sum_{i=1}^{4} x_i$.

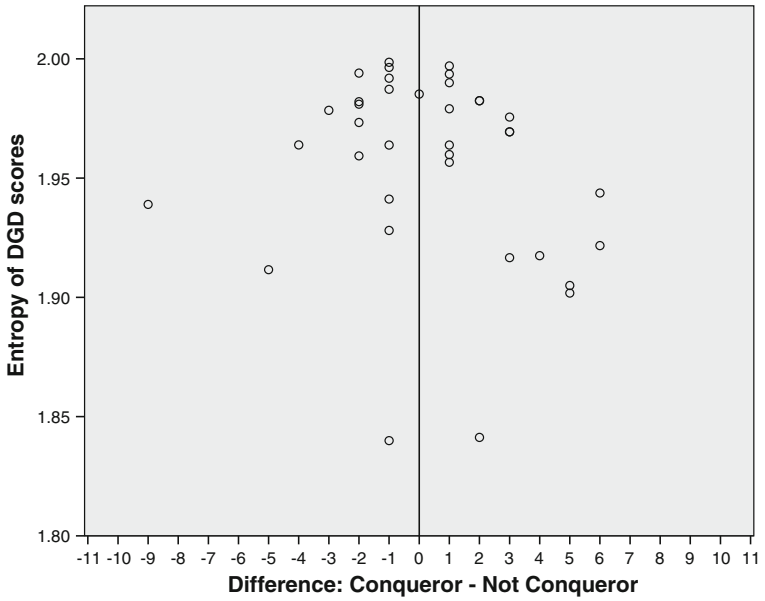

Fig. 4 On the abscissa, the '*Conqueror*-ness' of the 37 players based on their DGD scores. On the ordinate, the entropy of the player's DGD profile scores

ordinate of Fig. 4 shows the strength of variation of players' profiles, and thus the strength of bias expressed in their play preferences in the survey. Only 12 out of 37 players, just under one third, have entropy less than 1.95, suggesting that if behaviour in Pac-Man directly followed DGD profiles, we should find our test subjects difficult to discriminate amongst. Relatedly, the players also clearly tend to have quite small difference scores, whether positive or negative. The *Conqueror* and *Not Conqueror* groups do not statistically differ on these two measures, according to Student's t test.

Aside from the DGD scores, the respondents were asked the same questions as in the previous experiment: to self-report their sex, age, previous Pac-Man experience, and kind of gamer (hardcore, casual or no idea).

The next section shows results of the classifier validation over the 37 players' game logs, testing all 12 settings for the rule conflict resolution and scope methods.

4 Results

The primary output of the testing process was 12 result values from 12 classifiers, reporting once per level and then averaged over all levels and all games to produce a final classification. Each of the 12 results could be said to be a statement about the null hypothesis: that there is no linear relationship between the game-play and the typology instrument. Our accuracy results and associated p values (with confidence interval of 16.1 for a confidence level of 95 %) test this hypothesis, as shown in Table 5. All tests represent the classifiers' output at the *end* of three test games, averaged over the results from each level. These accuracy results were found by comparing the classification

Table 5 The final accuracy results from our classifier, sorted by settings

Setting		%Accuracy			Bias	Correlation	p
Scope	Conflict resolution	Total	Conqueror	Not Conqueror			
Single-level	Equal-voting	61.75	***100***	23.5	0.24	0.26	***<0.0005***
	Weighted-voting	***64.7***	***100***	29.4	0.29	0.39	<0.5
	First-rule-to-fire	63.6	***68.4***	58.8	***0.90***	0.18	<0.3
	Inverse-first-rule	58.35	57.9	58.8	***0.99***	0.25	<0.2
Average-of-3	Equal-voting	45.2	31.6	58.8	0.73	0.18	***<0.03***
	Weighted-voting	50.75	36.8	64.7	0.72	0.18	<0.1
	First-rule-to-fire	49.85	52.6	47.1	***0.95***	0.04	<0.3
	Inverse-first-rule	45.5	26.3	64.7	0.62	0.17	<0.2
Average-all	Equal-voting	***65.15***	42.1	***88.2***	0.54	***0.4***	***<0.006***
	Weighted-voting	***73.35***	52.6	***94.1***	0.59	***0.42***	<0.1
	First-rule-to-fire	62.85	31.6	***94.1***	0.38	***0.41***	<0.1
	Inverse-first-rule	47.5	42.1	52.9	0.89	0.12	<0.3

for each setting with the players' maximum DGD type score. The accuracy obtained compares to a *population* baseline of 50 % (i.e. a random choice from a binary target), or a *sample* baseline of 51 % represented by the *Conqueror* class in our sample. Table 5 describes the accuracy in percentiles, plus individual accuracy for each class, along with a bias figure[7] (the ratio between accuracy for each label), the Pearson correlation and t test p value (one-tailed, two sample unequal variance). The best three values in each column are in bolded and italicised text.

Each setting can be ranked according to its place in each column, and a crude evaluation can pick the best setting as the one with the most columns among the top three. From this we can at least see that **average-of-3** and **Inverse-first-rule** are relatively poor performers. To make a further distinction between settings requires a more principled evaluation, given that relying only on the values above could lead to over-fitting. The naive approach would combine all 12 of these results together and this is presented first, below. However this lacks justification as a solution since no setting is in principle better than another, so we also compare combinations of the 12 rule-set results in Sect. 4.2.

4.1 Exploration of results

The average across all games of the 12 conflict-resolution and scope-setting values gave a single classification score for each player. That combined result is detailed in

[7] Note that the final score could be achieved with a high accuracy for one type and a low accuracy for the other. For the current context such a biased classifier would be a poor solution overall. It is preferable to have a classifier that performs similarly with both player types, so that accuracy for each type is roughly equal (across multiple tests). Thus the performance differential between accuracy in each type can be considered a metric for classifier quality, and that is what the Bias column shows.

Fig. 5 below: the abscissa indicates the familiar *difference* score between *Conqueror* and highest *Not Conqueror*, and the ordinate indicates classification scores from the rule-set. Both axes are negative for *Not Conqueror*, positive for *Conqueror*. Note that the player with a difference score of 0 is included in the graph for completeness, but not in accuracy calculations.

The averaged approach rendered a final classification accuracy percentage of 69.4 %. The Spearman correlation coefficient between these final results and the DGD type scores was 0.35, statistically significant with a two-tailed $p < 0.05$. Accuracy was 73.7 % for *Conqueror*s alone, while for *Not Conqueror*s accuracy was 64.7 %. For brevity's sake, these partition-accuracies will be written below as {*Conqueror* score, *Not Conqueror* score}.

The slope and intercept of the linear trend-line in Fig. 5 suggests that this correlation represents data with a linear relationship, and is not an incidental value for a non-linear or other type of relationship (such as demonstrated by Anscombe 1973). Though not a strong correlation it is still potentially sufficient for successful classification.

The above method for combining all 12 results poses a problem: the value of the classifier output was about one order of magnitude greater for **equal-voting** than for the **first-rule-to-fire** and **inverse-first-rule** approaches. This suggests that simply combining these values to obtain a final result may in fact obscure the results from the two first-rule methods. An alternative approach is to normalise the results for each separate setting before taking their average as the final result. This normalising approach rendered a classification accuracy percentage of 69.4 % again, with one more misclassification for *Conqueror* and one less for *Not Conqueror*. The score for each

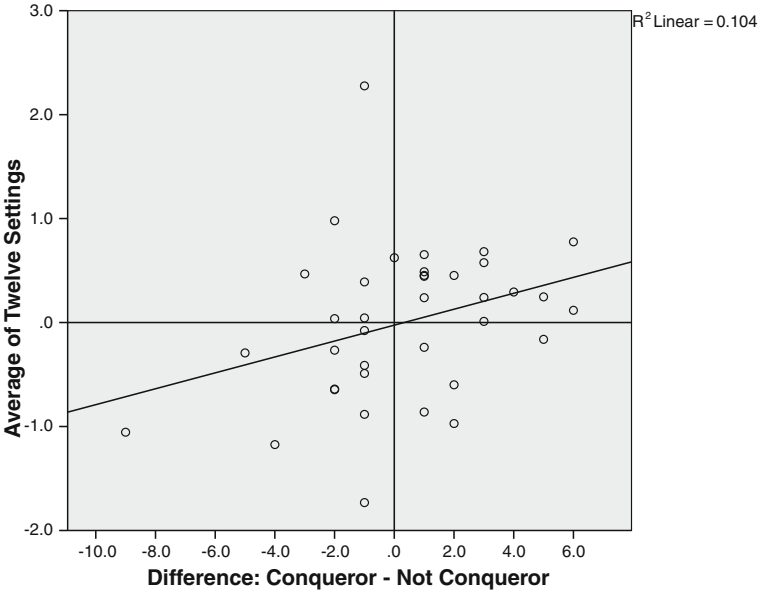

Fig. 5 Final results for the average of all games for each player using all 12 settings combined. They are plotted against the difference scores between *Conqueror* and *Not Conqueror*

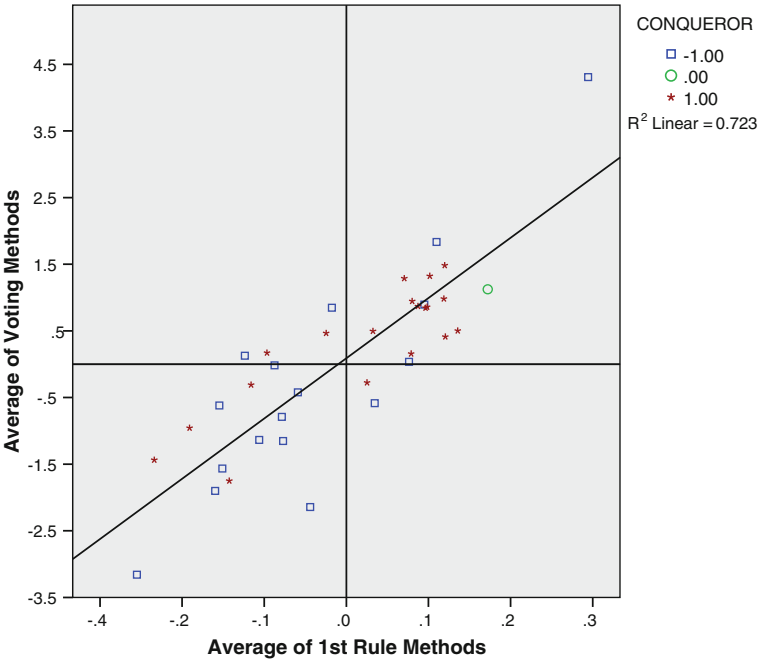

Fig. 6 The average results of **voting** methods versus the average results of 1st rule methods, for all players, *coloured* by the DGD type of each player. (Color figure online)

type alone was {68.4, 70.6}. After normalisation, the Spearman coefficient was still 0.35, statistically significant with two-tailed $p < 0.05$: so little change in correlation with the DGD differences.

It should be noted however that normalisation as we have performed it cannot be the *default* approach, because the maximum value used to normalise is not available when the classifier is used in situ—this is the reason we did not use it first.

4.2 Combining results from parameter settings

Perhaps the most important distinction between the various test settings is the difference between the two **first-rule-to-fire** methods, and the two **voting** methods, for conflict resolution. The distinction is important because only admitting one rule per level has the potential to exclude important information on the player from lesser ranked rules, and it is desirable to know if that fact renders the two **first-rule** methods less accurate or biased. Figure 6 shows the results for the two **voting** methods on the ordinate, against the results for the **first-rule-to-fire** and **inverse-first-rule** methods on the abscissa. The linear correlation between the two methods illustrates that they are largely in agreement. Each data point is labelled according to that player's DGD type score.

The accuracy of the **first-rule** methods is 66.7 % or {57.9, 76.5}, while the **voting** methods maintain an accuracy of 69.4 % or {73.7, 64.7}. The **first-rule** methods also have a slightly lower correlation coefficient to the DGD types of the players, of 0.31,

with a two-tailed p-value of marginal significance at 0.06. The **voting** methods have a correlation of 0.35 and $p < 0.05$. Thus only voting methods for conflict resolution give statistically significant results, suggesting that **voting** methods make a slightly better classifier than **first-rule** methods.

Next we tested the three scope settings. For these settings, each final result combines values from four conflict resolution settings, so the data was normalised beforehand (by division by maximum value). The **average-all** setting gives an accuracy of 66.7 % or {47.4, 88.2}. Interestingly correlation of the **average-all** scope result with the DGD difference score is the highest seen at 0.36, and statistically significant with $p < 0.03$. On the other hand the correlation coefficient for the **average-of-3** setting is 0.16. With a non-significant p-value of 0.35 these are the least predictive of all the results. Additionally, this setting has worse accuracy at 58.3 % or {47.4, 70.6}. Clearly, the method of averaging over a fixed-size window of levels is sub-optimal without tuning. The **single-level** setting shows accuracy of 69.4 % or {84.2, 52.9}. Correlation to the DGD scores is comparable to the **average-all** setting with a coefficient of 0.33, and statistically significant with $p < 0.05$.

The accuracy of classifications using each scope setting was surprising, since it did not relate linearly to the number of levels in scope. This may have been because different pre-processing steps were used for single-level as compared to multi-level approaches. Figure 7 illustrates the two higher-scoring settings, **average-all** on the ordinate and **single-level** on the abscissa. The linear correlation again shows that both

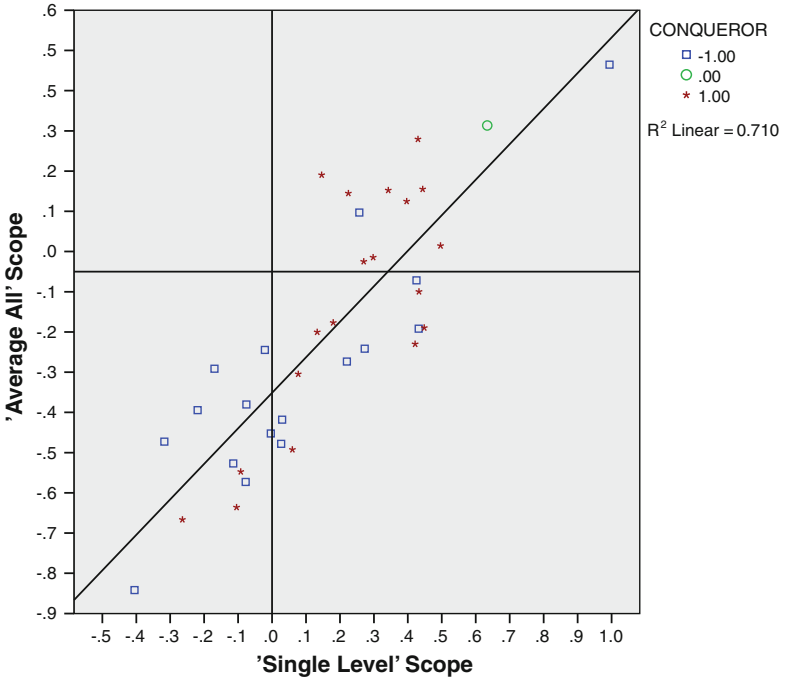

Fig. 7 Results for the scope setting **average-all** plotted against results for the **single-level** scope setting (in each case combining values from the four conflict resolution settings)

settings are quite similar. The colouring by Conqueror membership shows the bias of each setting: i.e. the majority of values are on one side of the zero-line for each setting, the negative side for **average-all** and the positive side for **single-level**.

From an accuracy perspective, taking any given scope setting in isolation seems to guarantee a definite bias either for or against *Conqueror*. One setting, such as **single-level**, might still have the same accuracy as the combined settings, but that would be because its *Conqueror* bias will correctly classify more *Conqueror*s just as it misclassifies more *Not Conqueror*s. This evidence of bias emphasises the need to combine settings to generate a usable final accuracy result. In the final analysis, obtaining accurate reliable classifications and overcoming (apparent) bias leads us to the combined voting methods over the combined **average-all** and **single-level** scopes as the best solution. This combination achieves 72 % accuracy {78.9, 64.7} with correlation of 0.36.

Concluding this section on classifier accuracy, it could be considered over-fitting to obtain high accuracy in classifying players, from a classifier built with the data of only 100 subjects and tested on less than 40. So reliable accuracy at ∼70 % (up from 51 %, based on the default of the set of test players) is a satisfactory result, given the inherent instability of the testing domain and the fuzziness of the DGD survey instrument, and it motivates further development of this approach.

5 Analysis

This analysis illustrates the support for the proposed hypotheses (next section) and shows a deeper examination of the results with the aim of uncovering further features operating beyond the level-by-level scope of those already used (Sect. 5.2). In both subsections we look mainly at individual outputs from the classifier or DTs. While this does not give generalised results, it is rather our goal to illustrate the *start* of a DM iteration (in this case a later iteration), in the domain of player modelling. This is partly motivated by the fact that in reading existing player modelling literature (e.g. as in Sect. 2.1), one might be led to think that the first iteration (which is usually the only one described) is the only one that *needs to be done*.

5.1 Model output analysis

In this subsection we examine how concepts learned by the DTs corresponded to our hypotheses of player behaviour in Pac-Man, referring back to Table 1 for comparison. The value of this analysis is both to get an impression of the rules' descriptive accuracy, and to show how one can deepen the understanding of types by examining the rules and features.

We might conduct a simple hypothesis test on the group mean differences of the features relevant to each hypothesis. Results from using one-tailed t tests are shown in Table 6 below—only the best supporting feature for each hypothesis is shown (or the second best where the hypothesis is complementary to one already tested: H2 and H6; H3 and H8; H1 and H11; Table 1). However, as we can see, most of our hypotheses cannot be captured in such a simple statistic, suggesting that the investigator of player behaviour must try to interpret the output of ML algorithms as we do below. In fact

Table 6 Results of t tests on group mean differences for hypothesis-related features

Hypothesis	Feature	Expected	t-value	df	p
H1	*A4_Hunt Even After Pill Finishes****	+	2.4	98	0.008
H2	*Sp1_Average States**	−	−1.6	98	0.06
H3	*Pac-Man-Ghost1 Distance_Max****	−	−2.6	98	0.005
H4	*R2_Average Cherry Eaten Per Level*	+	0.7	98	0.2
H6	*C5_Moves With No Points Scored**	+	1.6	98	0.06
H7	*D2_Player Vacillating*	+	1.0	98	0.16
H8	*S4_Teleport Use*	+	0.8	98	0.2
H9	*P1.b_Lure: All Ghosts Lured***	+	1.7	76	0.04
H10	*Lives Std Dev**	+	1.4	94	0.09
H11	*P4.b_Average Speed Hunting 2nd Ghost*	−	−0.6	98	0.3

two hypotheses, #5 and #12, could not even be tested as they are out of scope of the current feature set. This is discussed further below.

In Table 6, column 1 shows the hypothesis tested and column 2 lists the features used for testing with their significance (* marginal <0.1, ** <0.05, *** <0.01). Column 3 holds the expected direction of the relationship as per the hypothesis, and then in columns 4–6 the statistical values are listed for the t test. In all cases Levene's test was used to ensure equality of variance.

These simple tests support less than half of the hypotheses significantly (although tellingly the sign of the t-value always agrees with the expected direction of the test). Nevertheless, the real support is found in the data derived from the DTs.

Raw logged data, such as the variables representing score or Pac-Man position, encode player behaviour at a very high resolution and low level of interpretability. By representing moments of the log data, features encode behaviour at a higher level of abstraction and can be easier to interpret. DTs or rules represent yet a further 'level' of encoding, logically combining features to enable interpretation. The upshot is that these encodings allow interpretation of play just as would personal observation, but in a more generalised way. Features, DTs and rules are thus 'interpreted' below as an example of what can be done. Note that these discussions relate to the original data set of 100 players, not the 37 players from the validation test reported in Sect. 4. Thus one will see numbers of players in the DTs which are the *training* sets of these 100.

5.1.1 Tree analysis—correspondence to hypotheses

We use aggregate data features to represent game-play logs compactly. DTs represent facts about that data space by iteratively using linear separations of individual features to maximally discriminate between classes, which in this case are DGD types. Rules are thus a combination of one or more facts or separations, which are termed conditions, and represent a direct prediction for behavioural preferences. However, we must be wary of claiming the fact associated with each rule condition as a trait for all *Conquerors*, as rules are only inferences from a sample of the population. Thus instead

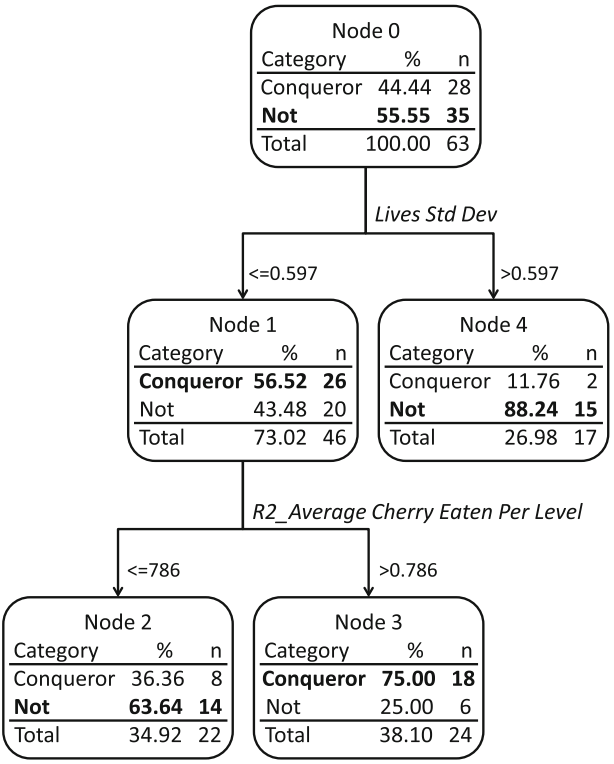

Fig. 8 DT for C5.0 algorithm with 63:37 train:test ratio and accuracy of 70 %. Players captured in each node are from the *test* set

of associating each condition like a trait to all *Conqueror*s, the value of *Conqueror* is applied to each condition/trait. Then at the final analysis, we may look at all traits and derive an over-arching tendency for behaviour for the *Conqueror* class. We begin by examining the tree in Fig. 8, learned by C5.0 with one of the better performing parameter configurations of this algorithm (i.e. this tree represents one replication for a parameter set where the average over seven replications performed relatively well).

The tree is quite small, trading off size for generality, and neatly encapsulates the difference between approaches to acquiring and losing Lives in Pac-Man. In broad terms, this tree seems to suggest that *Conqueror*s both gain more Lives (Node 3 vs. Node 2), and yet stay close to the mean value for Lives (Node 1 vs. Node 4). Node 1 details how 91 % of the *Conqueror*s in this training set do not show wide variation in their number of Lives across a game. In other words (since they start off with five Lives) they are maintaining close to five Lives throughout the game.

Further, in Node 3, 69 % of those *Conqueror*s (from 'Node 1') ate the Cherry in four out of every five levels or more. As the game mechanics of our Pac-Man dictate, eating the Cherry confers a life. Thus we have a large group of *Conqueror*s, 64 % of the training set's *Conqueror*s, who gain a lot of Lives but do not show it in the standard deviation over Lives. Given how the features are calculated, this must mean

that these same *Conqueror*s also regularly lose Lives, and that loss and gain happen synchronously, one following another. If we take both facts together we obtain the *path* or rule ending at Node 3, and we have a complete statement about *Conqueror*s with an associated support and confidence. This statement is indicative of *traits* for the class *Conqueror*, implying that *Conqueror*s take more risks—pushing the envelope of caution—but are tenacious enough in play to prioritise survival by going after life-giving bonuses (i.e. the Cherry fruit) as well. This all bears out the first causal part of H3 (playing with less caution), and H4 (more attempts to collect fruit).

The path/rule terminating at Node 2 describes *Not Conqueror*s. The implication here is orthogonal to the *Conqueror* rule—low standard deviation in Lives, with few Lives gained, implies that few Lives were lost—and achieving that requires a more cautious attitude to play, supporting H8 and H10 to some degree.

Other signs of the *Conqueror* type's characteristic 'enthusiastic incaution' can be seen in other trees with better support and confidence. Figure 9 shows a C5.0 tree which managed both a modest amount of lift and high confidence: the average purity over leaf nodes is 92 %. From this high confidence quite a clear picture of *Conqueror* type behaviour can be interpreted.

Starting at Node 1, the feature *A4_Hunt Even After Pill Finishes* splits off 13 % of *Not Conqueror*s. A very low score associates them with a particular caution in the hunt, or even indisposition to hunt at all, providing support for H8 and H11 (this could be for strategic reasons—some Pac-Man players prefer to minimise exposure to risk to prolong survival—which is suggestive of *Manager* play). It also fits quite well with our H1, stating that the *Conqueror* type may be more likely to chase Ghosts with such aggression that they are still chasing *even* after the effects of the Pill end.

Then in Node 3, a low score in *Sp1_Average States* is associated with a relatively high number of *Conqueror*s, supporting H2. This feature does not necessarily equate to the speed of play, which is measured by *Sp2_Average Cycles*, but is based on the number of moves a player takes to finish a level. Being less cautious than the *Manager*, more eager to progress and get high scores than *Wanderer* or *Participant*, this trait fits the typological description of the *Conqueror*.

S4_Teleport Use splits off a small number of *Conqueror*s in Node 5, implying that less use of teleporters can be a *Conqueror* trait. Since teleporters are a device more useful at the strategic level of play for keeping Ghosts at bay, this is some evidence for H7 and H8, although because the opposite Node 6 is quite impure (still containing 30 % *Conquerors*) it is not compelling.

Finally, 'Node 7' and 'Node 8' are split by *C3_Close Calls*, where lower values indicate the *Conqueror* type—this runs contrary to the second evidential part of H3. This is not intuitive since we expect *Conqueror*s to show less fear of Ghosts and thus get closer. The fact that *Conqueror*s have such low scores (under 0.1) in the left branch of this split suggests that they *never* come too close to the Ghosts when not in Hunt mode. When we remember that these **particular** *Conqueror*s have higher than average scores (for *Conquerors*) in *Sp1_Average States* and *S4_Teleport Use*, we get a clearer picture. These players may simply be playing a more Ghost-aware game, using teleporters and extra moves to keep their distance, perhaps waiting until the time comes to Hunt. This insight goes beyond what was suggested in H1-H11, indicating further ways to look at player behaviour.

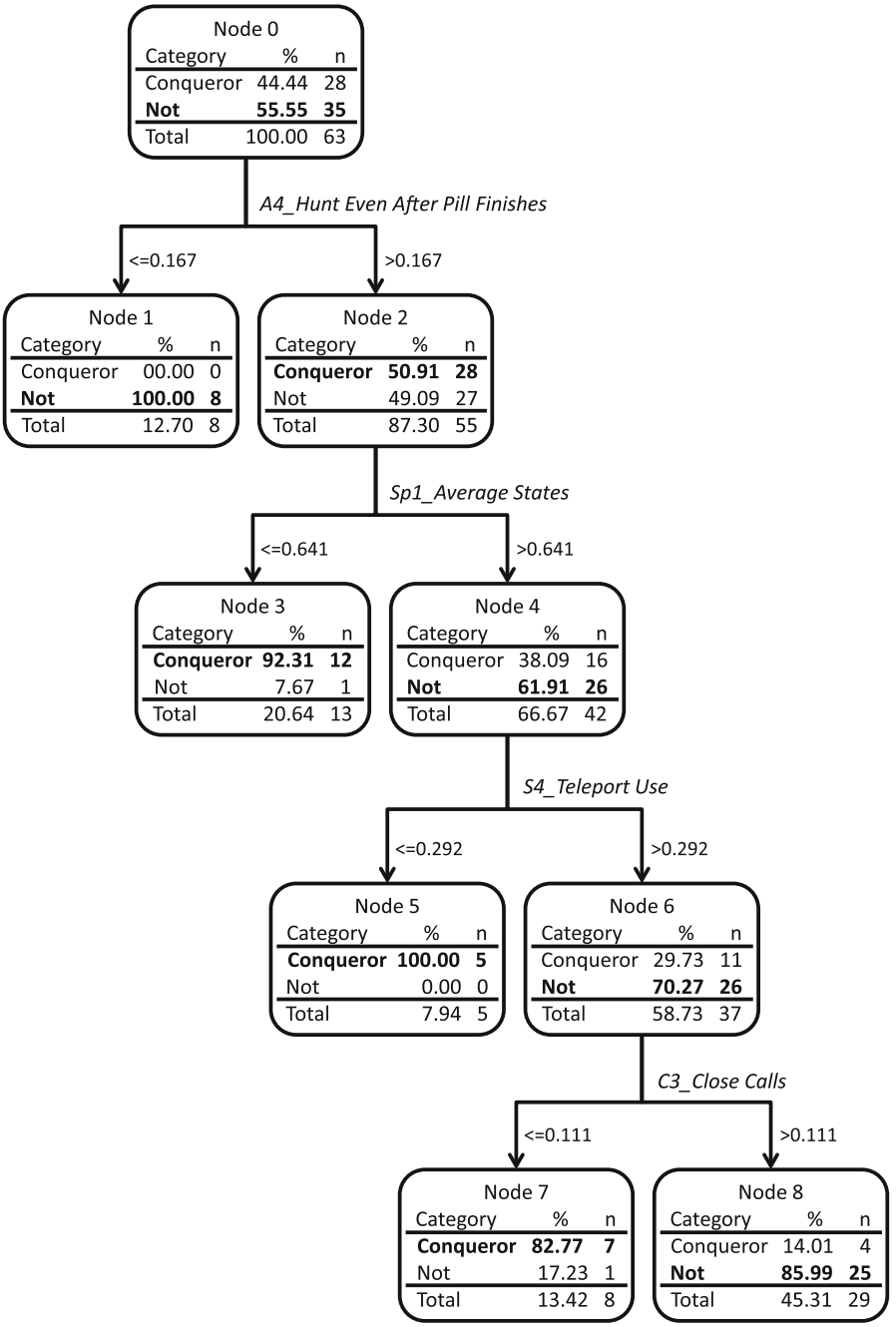

Fig. 9 C5.0 tree trained on 63 records, tested at 60 % accuracy on 37 records

Studying whole DTs offers a picture of player behaviour with good clarity because both sides of every split are shown. Yet classification rules can be also be taken alone to operate without reference to these splits. Thus we look at them individually to interpret how they represent player types. Additionally, the traits mentioned in Sect. 3.2 are neatly encapsulated in the trees shown, but we must look to the rules to see how significantly each trait is represented within the rule set.

5.1.2 Rule analysis—correspondence to hypotheses

Several of our rules stand out in distinguishing *Conqueror* from *Not Conqueror*. The support and confidence of rules is a factor, as is simplicity. To illustrate simpler aspects of the *Conqueror* type, we start off with two interesting rules, with single conditions unrelated to the trees examined above.

Rule 20; Not (support 14.22; confidence 0.915)	**if** D2_Player Vacillating $>$ 0.678	**then** Not
Rule 36; Conqr (support 9; confidence 1)	**if** A4_Hunt Even After Pill Finishes $>$ 0.563	**then** Conqr

The first rule (number 20 in the rule-set) showing that higher values of the feature *D2_Player Vacillating* indicates *Not Conqueror*s, agrees with the *Conqueror* traits of fast and incautious play and gives further support to H6 and H7.

Then rule 36, where *Conqueror*s score higher for *A4_Hunt Even After Pill Finishes*, shows the strong evidence for H1 since there are no *Not Conqueror* players scoring in the top half of the distribution for this feature (recall that feature values are normalised in these rules, thus $\sim$0.5 represents the mean).

We next examine a complex multi-dimensional rule as further illustration of the interplay between features, selecting rule 2 for its high support and confidence.

Rule 2; Not (support 27; confidence 0.889)
if C5_Moves With No Points Scored $>$ 0.413 **and** P4.b_Average Speed Hunting 2nd Ghost $\leq$ 0.504 **and** P6_Put Off Dots Near Ghost House $\leq$ 0.469
then Not

This rule classifies 24 players correctly as *Not Conqueror*s. There are three conditions in this rule to describe what *Conqueror* behaviour *isn't*. A higher value for *C5_Moves With No Points Scored* indicates more unnecessary movement (evidence for H7) which combines with less instances of *P6_Put Off Collecting Dots Near Ghost House*, and a lower average speed to catch the second Ghost during a Hunt (third condition). Together these conditions suggest a player who enacts strategies (to clear the map or to Hunt Ghosts), and thus does more manoeuvring over empty areas to 'set up' their strategies. Overall this seems like a *Manager* type, but there is another possible interpretation. Lower scores in *P4.b_Average Speed Hunting 2nd Ghost* could also be scores of *zero*, meaning that the player did not use their Pills to Hunt both Ghosts at all. Such a reading supports both the more timid types of *Wanderer* and *Participant*, and the high strategy of a *Manager* who uses the repulsive effect of the Pill to buy more time for Dot collecting. Although there is more than one possible interpretation

of the rule, none describe *Conqueror* play, so this ambiguity helps to illustrate how, even given some uncertainty, there exists a stable basis for *Not Conqueror* behaviour.

The rule set listed in Appendix B is accompanied by further analyses of the top five rules. These throw up additional support for our hypotheses of the *Conqueror* and *Not Conqueror* types. From all analyses, two hypotheses were not addressed out of the 12 proposed: #5 and #12. These pertain to the number of times a player repeats the game, and length of individual games. Such total-game features were left out of the rule set in this case, because we were focused on features that would apply over a single level, to facilitate real-time deployment. The inapplicable hypotheses are retained in this paper for completeness—they represent types of statements beyond the scope already discussed.

5.2 Test results analysis

In this section we use the test results to illustrate a further utility of the player modelling method: that the classifier output can be used to expand the investigator's knowledge by examining player actions, and generate fresh insights. This process represents iteration back to the initial stage of DM; the outcomes here could serve as the basis for a new round of ML, generating new testable models and insights. We however go only as far as generating the insights. In other words, we will present the domain exploration part of DM rather than the outcome of applying the resulting algorithm—thus please note Sect. 5.2 is not intended to present generalisable conclusions. Rather, investigators of game-effects on players, such as game developers, would benefit from the insights afforded by this step in DM their game.

The procedure presented so far has mainly addressed the question: how well do the DGD survey and rule set classifier descriptions of players match? Further questions of immediate interest are those which would generate readily-implemented updates to the existing models; and the first of these involve the meta-game, i.e. variations over an entire game. This section is structured in two parts around the following questions:

Q1. Does game length correlate either with player type or changes over levels?
Q2. Can we extend the current approach in a simple manner to detect concept drift: i.e. how players change type over their full set of games?

5.2.1 Game length

Examining the characteristics of game length with respect to DGD type certainly uncovers some distinct differences. The data in Fig. 10 shows the average levels played for all the games of each player plotted against their rule-set classification, with separately marked data series for DGD labels *Conqueror* and *Not Conqueror*.

Although the DGD type clearly correlates with the number of levels played (Pearson $= 0.39$ $p < .02$), with *Not Conquerors* producing less levels on average, we found that the number of levels played does not really correspond with the *magnitude* of the DGD difference score. Players with low differences are playing as many levels as those with high differences, with non-significant correlations. The variation between

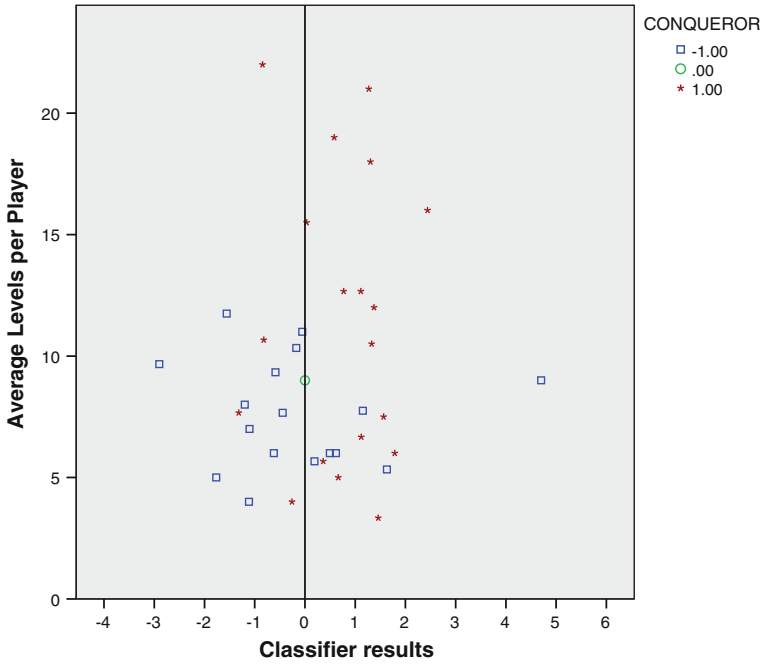

Fig. 10 Average levels plotted against rule classification of players, *coloured* by their DGD type *Conqueror* or *Not Conqueror*. (Color figure online)

players also seems to distinguish *Conqueror* from *Not Conqueror* players. The calculated variance across the *Conqueror* data series was 32.8 compared to 4.8 for *Not Conqueror*s. This is variance of average length of game *between* players; if we examine the within-player variance between the lengths of games, and then average the results over all players, *Conqueror* variance is 9.2 and *Not Conqueror* is 3.0.

If we use the rule classifier to divide players, these distinctions vanish: the gap between average numbers of levels and the respective variance of average levels within the *Conqueror* and *Not* groups become insignificant. The various differences related to game length are an empirical indication of a genuine difference between DGD-typed player behaviours, yet they are not predicted by the rule-set classifier. The implication is that game length and variation are two metrics operating beyond the scope of our classifier, but which help to distinguish playing styles. Thus they could be used as additional discriminants to improve end-of-game classification.

5.2.2 Concept drift

We anticipated that our player modelling solution would encounter problems involving concept drift in real-time data capture. Concept drift refers to the changes that occur over time in the data domain that our ML classifier operates on. According to the literature "such change may affect one or more classes and, within a class, may affect one or more of the rules which constitute the definition of that class" (Black and

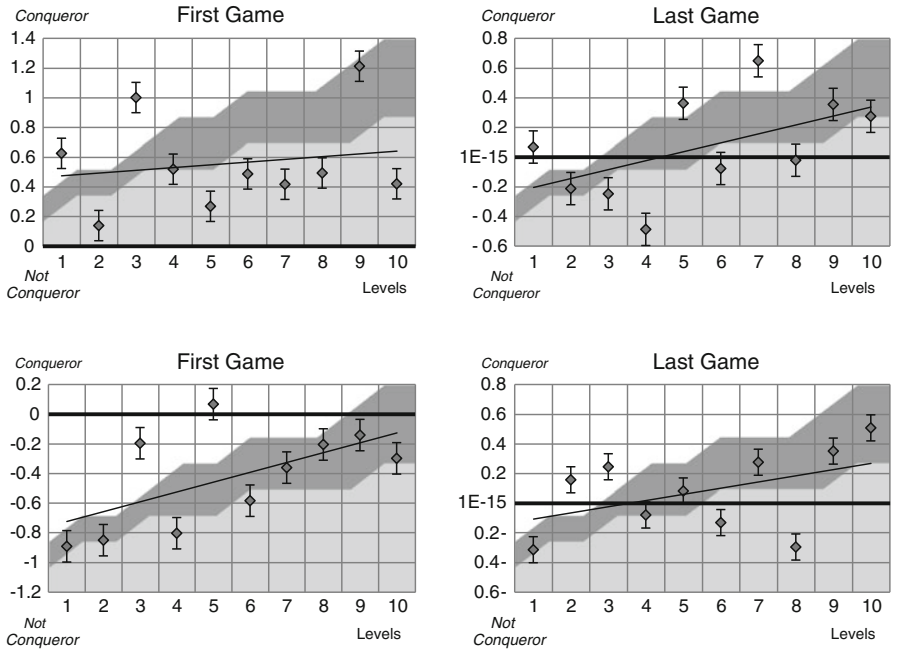

Fig. 11 *Top row* classification results from the first 10 levels of the first (*left*) and last (*right*) games played by eight *Conqueror* players (who achieved at least 10 levels in their first and last games). *Bottom row* the same data for *Not Conqueror* players

Hickey 1999). Although it is not within our scope to implement and test drift handling routines here, we can attempt to identify how and if drift is occurring.

We first note that most players exhibit behaviour classified as both *Conqueror* and *Not Conqueror* in levels across their games. Indeed it would be unusual if the classification was uniform across all levels, as players are not expected to exhibit behaviour conforming to their type absolutely 100 % of the time. For example, someone playing like a *Conqueror* and incautiously losing too many lives in a level might decide to play more cautiously in the next level, and exhibit *Not Conqueror* style behaviour. Thus evidence for concept drift must be sought across entire games and series of games, for instance as trends of classifications over levels within each game and over a series of games.

To illustrate the trends over games, classification scores for first and last games of both types of players are shown in Fig. 11. To show a representative sample of players and behaviour, we selected those players whose games had at least 10 levels (which was the average total game length)—that meant eight players of each type. In these figures, game levels are shown on the abscissa; positive values on the ordinate represent a *Conqueror* classification; the linear trend line and standard error bars are shown, and all figures share the same axes scaling for reference.

A major influence on player status is the difficulty curve. This is visualised in the shaded areas. The two bands of shading show two measures of difficulty—Ghost speed (top band, dark grey) and the probability of Ghosts moving toward Pac-Man. The values that the bands represent are the cumulative count of increments for both

difficulty measures (with no relationship to the ordinate), giving an indication of the overall difficulty trend to compare with player classifications.

Perhaps the most immediately interesting behaviour is the central tendency in play style from first to last games. In first games, we see only one contra-DGD type (average) classification for players of both types; in last games, classifications are split almost 50/50 between *Conqueror* and *Not Conqueror*. This change in classifications could reflect how players learned to improve: each type engaging in some non-typical behaviour, e.g. *Conquerors* showing more caution, in order to achieve higher scores. It is also notable that the correlations between classification and difficulty level are all positive—as the game becomes more difficult, classification results tend toward *Conqueror*.

Examining the trends of classification from first to last games suggests that a likely contender for causing concept drift is player learning. Additionally, reaction to context in terms of the changing game difficulty also impacts the *style* of play such that classification of player type is affected. This reinforces the notion that type is a local contextual construct of player personality. Overall, the analysis suggests that the trend of classifications over a game (i.e. the fitted curve), or else the correlation between the classifications and some metric of the global state of the game (such as the difficulty curve), could be considered an important indicator of player type and potential extra metric for classification.

6 Discussion

Our main result is achieving over 70 % classification accuracy on a player modelling task using typology class labels. Moreover, the main and much broader aim of the paper was to show a complete player modelling solution in fairly high detail, including the 'messy' domain analysis parts which are too often available only from expansive textbooks or even hidden in researchers tacit knowledge.

The DGD typology was chosen to be general, yet by its nature it is rather broad. Typologies are an approach that has an inherent error, because specificity must come from a consideration of not just the player psychology, but the context and game type. Even just considering player psychology, still the players' preferences, performance and learning must be disentangled. Such a general framework does not currently exist, although there are ongoing efforts.[8] Meanwhile, general typologies with more classes have been proposed since this work was initiated (Bateman et al. 2011). Before considering a general model including context and game type, the inherent error of the typology could be minimised by extending our approach to create an ensemble of binary classifiers, one for each type label.

One interesting convergence is evident looking at other approaches (Baumgarten 2010; Drachen et al. 2009), in that their feature creation also turned up meta-behavioural groups similar to our behavioural 'traits'. These methods are quite different yet complementary to ours, in that their features are derived more readily and yet the following interpretation is not as rich. The two approaches could be combined with a

[8] See for example Cowley et al. (in preparation, b), or the 'Fun of Gaming' project at http://fuga.aalto.fi/.

new test-bed game to further validate, and hopefully improve on, the kinds of results shown by all three papers.

In general, for work such as conflict resolution and rule selection, the methods we used are rather simple. In fact the aim of the paper is not to illustrate data-mining methods, but rather methods to model and *understand* player behaviour. The latter is the key reason why so much of the approach is manual. It is also potentially a more transparent read for the data-mining layperson. Nevertheless we would still advocate more sophisticated methods for, at least, feature and rule selection (Chen et al. 2006; Kludas et al. 2009). With a sufficiently large dataset, scope selection could be easily achieved with a basic hill-climbing algorithm or perhaps a genetic algorithm (GA), using classifier accuracy as fitness function. Chen et al. (2006) also addressed conflict resolution.

The classification result itself can be considered on its contextual merits, since it is (a) derived from a rather simple game, and (b) very plausible to improve via methodological upgrades. On the other hand, we can see from recent changes in the commercial game market that simple games are still very relevant [e.g. *Angry Birds* (Rovio 2010)]. Additionally, it is clear that we had a somewhat small data set size, compounded in this case by data complexity (the 'curse of dimensionality'). A small data set makes it difficult to both learn a good predictive model and test it, since as feature dimensions increase the degrees of freedom inherent in the data set decreases. In other words, the concepts learned from training set data may be over-fitted and not apply to test set data. We may say that the small size of the corpus militates against finding a split that provides representative, statistically significant training and test sets. A footnote to this issue is that when modelling with continuous predictor variables, linear DT algorithms are prone to the problem of sub-tree replication. This occurs especially with proportionally large continuous variables which may have to be split into many threshold-based Boolean attributes (Mitchell 1997). The outcome is an increase in spurious dimensionality and no correlated increase in descriptive power. A solution might be mutual information-gain based sub-tree pruning.

The hypothesis examination process threw up an interesting insight—the game itself plays an important role in polarising play into specific styles, which may correspond more or less closely with one of the existing types. Although *Not Conqueror* is probably not as homogenous a type as *Conqueror* in the context of Pac-Man, it is probably more homogenous than *Not Manager*. We believe this is the case because the *Conqueror* style seems more removed from the *Wanderer* or *Participant* style than does *Manager*, in the context of Pac-Man. However it is not proven, and remains speculation pending further work. The purported polarisation ties in with the argument of Bateman et al. (2011) that player type is more valid when viewed as a measure of trait rather than state.

In terms of the case study in Sect. 5.2.2, it is clear that this work is not meant to imply general results. Instead, the process throws up meta-level features which lie beyond the scope of the rule-set classifier and indicate how to extend the approach. Indeed, this examination of ML results, and subsequent (proposed) extension thereof, is characteristic of the recursive nature of practical DM. We apply statements of formal systems to a domain where we are in fact dealing with certain levels of

player psychology which cannot be captured by rules which exist within the formal system.

6.1 Future work

A first priority of future work would be to experiment with the selection of rules from the DTs, both to improve reliability and to ensure fidelity to the original trees built by the C5.0 and CART algorithms. Cherry picking rules based on a ranking selection method was necessary for the reasons explained in Sect. 3.2.2, but there was always a risk that this approach introduced methodological errors. An ensemble of trees might provide a more reliable rule deployment than a ranked set.

The rules themselves might be improved by binning the continuous predictor variables. This would have the effect of categorising the behaviours of player types, which could be valuable from an interpretation perspective. Worthwhile binning requires informed analysis of the data to ensure partitions are placed constructively, otherwise artificial limitations are placed on the learning algorithms and results can be made worse. This implies that binning might work well only in 2nd or later iterations of a DM process. In a larger project that used our method, where feature derivation and deployment are carried out by separate individuals, categorising data by binning might improve the comprehensibility of rules, a potential benefit for collaborative processes.

Instead of the conflict resolution approach used, it could be beneficial to implement a choice function, to allow automated selection of the most appropriate rules that fired in each set. This implies some higher level decision making, so that the modelling method could reason about the player based on lower level data gathering like the feature output so far.

Manually tuning the parameters of each algorithm we tested, to achieve best accuracy, was very time consuming (at the least). A novel method of tuning DT algorithm parameters for large DM projects is to apply a GA (Mugambi et al. 2004). The parameters of the DT algorithm are used to build the chromosomes of the GA's population, and the GA is run until the most effective set of parameters is found. This was not compatible with our Clementine-based approach but future efforts should take account of such advances in DT methods.

Finally, we would envision using meta- or compound features combining lower & higher levels of game-play, moving beyond the dichotomy between level-by-level features and whole-game features. For interesting work in this direction, see Kludas (2011).

7 Conclusions

We have presented a method for classifying players in an arcade-style game such as Pac-Man in real-time. After learning a set of DTs, the optimal rule-set for the task was 'cherry–picked' and deployed. Deployment of learned DT rules was a non-trivial process, due to the various conflict resolution and scope settings that had to be implemented. Nevertheless, results did prove the viability of the method we embarked upon to determine the type of computer game-players.

As described, the rule set generated by learning from the training data of the first 100 players was then deployed to the Pac-Man game to do classification testing for a further 37 players, resulting in accuracy of ∼70 %. While the classification is interesting, we feel it is incomplete in the general case without further analysis of the behaviour of the test players, because this classified behaviour represents the ground truth of player interaction with the game. In other words, with enough deep analytical work, the classifier results *can* lead back to an understanding of how the player experienced the game.

In summary, solid results were already obtained in both classification and interpretation, two equally important aspects of player modelling. Yet there is significant room for improvement and cross-fertilisation with other ML methods or psychological theories. The rule-based learning demonstrated has the positive property of being highly legible, which leaves it very open to further innovation.

Acknowledgments The list of questions for the DGD typing instrument (Appendix C) and the hypotheses presented in Table 1 were developed through discussions with the author of the DGD typology, Chris Bateman, for whose collaboration we are very grateful. For technical assistance we are grateful to Leo Galway, and all play testers also. For examination and feedback on earlier drafts, we thank Michael McNeill and Colin Fyfe.

References

Acuña, D.E., Parada, V.: People efficiently explore the solution space of the computationally intractable traveling salesman problem to find near-optimal tours. PloS one **5**(7), e11685 (2010)

Anscombe, F.J.: Graphs in statistical analysis. Am. Stat. **27**(1), 17–21 (1973)

Anyanwu, M., Shiva, S.: Comparative analysis of serial decision tree classification algorithms. Int. J. Comput. Sci. Secur. **3**(3), 230–240 (2009)

Bartle, R.: Hearts, clubs, diamonds, spades: players who suit MUDs. J. Virtual Environ. **1**(1) (1996)

Bateman, C., Boon, R.: 21st Century Game Design, vol. 1. Charles River Media, Londan (2005)

Bateman, C., Lowenhaupt, R., Nacke, L.: Player typology in theory and practice. In: Proceedings of DiGRA: Think Design Play 2011. Utrecht, The Netherlands (2011)

Baumgarten, R.: Towards automatic player behaviour characterisation using multiclass linear discriminant analysis. In: Proceedings of the Artificial Intelligence and Simulation of Behaviour Symposium: AI and Games, March 2010. De Montfort University, Leicester, UK (2010)

Beal, C., Beck, J., Westbrook, D., Atkin, M., Cohen, P.: Intelligent modeling of the user in interactive entertainment. In: Forbus, K., El-Nasr, M.S. (eds.) Artificial Intelligence and Interactive Entertainment II. Papers from the AAAI Spring Symposium, Stanford, CA (2002)

Black, M., Hickey, R.J.: Maintaining the performance of a learned classifier under concept drift. Intell. Data Anal. **3**(6), 453–474 (1999)

Bowling, M., Fürnkranz, J., Graepel, T., Musick, R.: Machine learning and games. Mach. Learn. **63**(3), 211–215 (2006)

Breiman, L.: Classification and Regression Trees. Wadsworth International Group, Belmont (1984)

Caillois, R.: Man, Play, and Games. Free Press of Glencoe, New York (1961)

Chaperot, B., Fyfe, C.: Improving artificial intelligence in a motocross game. In: Proceedings of IEEE Symposium on Computational Intelligence and Games, pp. 181–186 (2006)

Chen, G., Liu, H., Yu, L., Wei, Q., Zhang, X.: A new approach to classification based on association rule mining. Decis. Support Syst. **42**(2), 674–689 (2006)

Cowley, B., Moutinho, J., Bateman, C., Oliveira, A.: Learning principles and interaction design for 'green my place': a massively multiplayer serious game. Entertain. Comput. **2**(2), 10 (2011)

Cowley, B., Charles, D., Black, M., Hickey, R.: Developing features of game metrics to describe videogame-player behaviour (in preparation, a)

Cowley, B., Kosunen, I., Kivikangas, J. M., Järvelä, S., Lankoski, P., Kemppainen, J.: Machine learning and play patterns: a pilot study for the PPAX framework. J. Simul. Gaming (in preparation, b)
Curtis, P.: Mudding: Social Phenomena in Text-Based Virtual Realities. Intertrek **3**(3), 26–34 (1992). http://w2.eff.org/Net_culture/MOO_MUD_IRC/curtis_mudding.article
Drachen, A., Canossa, A.,Yannakakis, G.N.: Player modeling using self-organization in Tomb Raider: Underworld. In: Proceedings of the 5th International Conference on Computational Intelligence and Games, pp. 1–8. IEEE Press, Milano (2009)
Fu, D., Houlette, R., Rabin, S.: Constructing a Decision Tree Based on Past Experience AI Game Programming Wisdom 2. pp. 567–577. Charles River Media, Hingham (2004)
Fürnkranz, J.: Machine learning and game-playing. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 633–637. Springer, New York (2010)
Galway, L., Charles, D., Black, M.: Machine learning in digital games: a survey. Artif. Intell. Rev. **29**(2), 123–161 (2008)
Geisler, B.: An empirical study of machine learning algorithms applied to modelling player behaviour in a first person shooter video game. M.S. thesis, Department of Computer Sciences, University of Wisconsin, Madison (2002)
Grimes, S.: Mining the game: when marketing and gaming meet they do a lot more than advertise. Escapist Magazine, 3, 27 February 2007
Herbrich, R., Minka, T., Graepel, T.: TrueSkill: a Bayesian skill rating system. Adv. Neural Inf. Process. Syst. **19**, 569–576 (2007)
Huizinga, J.: Homo Ludens: A Study of the Play-Element of Culture. Routledge, London (1949)
Johansson, U., König, R., Niklasson, L.: The truth is in there—rule extraction from opaque models using genetic programming. In: Proceedings of the 17 th International Florida Artificial Intelligence Research Society Conference, FL, p. 113 (2004)
Johansson, U., Sonstrod, C., Niklasson, L.: Explaining winning Poker—a data mining approach. In: Proceedings of the 5th International Conference on Machine Learning and Applications, pp. 129–134. IEEE Computer Society, Washington (2006)
Jones, L.: Winning Low-Limit Hold'em. ConJelCo, Pittsburgh (2000)
Jung, C.G.: Psychological Types (Collected Works of C.G. Jung, vol. 6). Princeton University Press, Princeton (1971)
Katz-Haas, R.: User-centered design and web development. Usability Interface **5**(1), 12–13 (1998)
Kaukoranta, J., Smed, H.: Role of pattern recognition in computer games. In: Sing, L.W., Man, W.H., Wai, W. (eds.) Proceedings of the 2nd International Conference on Application and Development of Computer Games, pp. 189–194. Hong Kong SAR, China (2003)
Kaukoranta, T., Smed, J., Hakonen, H., Rabin, S.: Understanding Pattern Recognition Methods. AI Game Programming Wisdom 2. pp. 579–589. Charles River Media, Hingham (2004)
Keirsey, D., Bates, M.M.: Please Understand Me: Character & Temperament Types. Distributed by Prometheus Nemesis Book Co, Del Mar (1984)
Kennerly, D.: Better game design through data mining. The art and business of making games, Gamasutra (2003). http://gamasutra.com/features/20030815/kennerly_01.shtml
Kludas, J.: Information fusion for multimedia: exploiting feature interactions for semantic feature selection and construction. PhD thesis, University of Geneva, Geneva (2011)
Kludas, J., Bruno, E., Marchand-Maillet, S.: Can feature information interaction help for information fusion in multimedia problems?. Multimedia Tools Appl. **42**(1), 57–71 (2009)
Levillain, F., Orero, J.O., Rifqi, M., Bouchon-Meunier, B.: Characterizing player's experience from physiological signals using fuzzy decision trees. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG), pp. 75–82. IT University of Copenhagen, Denmark (2010)
Ludford, P.J., Terveen, L.G.: Does an individual's Myers-Briggs type indicator preference influence task-oriented technology use? In: Human-Computer Interaction INTERACT '03: IFIP, pp. 623–630. IOS Press, Zurich(2003)
Malone, T.W.: What makes things fun to learn? Heuristics for designing instructional computer games. In: Proceedings of the 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems, Palo Alto, CA, pp. 162–169 (1980)
McCrae, R.R., Costa, P.T. Jr.: Reinterpreting the Myers-Briggs Type indicator from the perspective of the five-factor model of personality. J. Pers. **57**(1), 17–40 (1989)

McGinnis, T., Bustard, D. W., Black, M., Charles, D.: Enhancing e-learning engagement using design patterns from computer games. In: Proceedings of the First International Conference on Advances in Computer–Human Interaction, pp. 124–130. ACM, New York (2008)
Mitchell, T.M.: Machine Learning: McGraw-Hill Higher Education, New York (1997)
Mountain, G.: Psychology Profiling in SILENT HILL: SHATTERED MEMORIES. Video Presented at from the Paris Game/AI Conference, 2010. http://gameaiconf.com/?p=141
Mugambi, E.M., Hunter, A., Oatley, G., Kennedy, L.: Polynomial-fuzzy decision tree structures for classifying medical data. Knowledge-Based Syst. **17**(2-4), 81–87 (2004)
Pittenger, D.J.: The Utility of the Myers-Briggs type indicator. Rev. Educ. Res. **63**(4), 467–488 (1993)
Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
Ravaja, N., Saari, T., Turpeinen, M., Laarni, J., Salminen, M., Kivikangas, M.: Spatial presence and emotions during video game-playing: does it matter with whom you play?. Presence Teleoper. Virtual Environ. **15**(4), 381–392 (2006)
Salen, K., Zimmerman, E.: Rules of Play: Game Design Fundamentals, vol. 1. MIT, London (2004)
Sklansky, D., Malmuth, M.: Hold'em Poker for Advanced Players. Two Plus Two Publishing, Las Vegas (1999)
Thawonmas, R., Ho, J.-Y.: Classification of online game-players using action transition probability and Kullback Leibler entropy. JACIII **11**(3), 319–326 (2007)
Thurau, C., Bauckhage, C.: Smarter team mates: applying hidden Markov models in sports games. In: Proceedings of the SAB Workshop on Adaptive Approaches to Optimizing Player Satisfaction, Roma, Italy, pp. 11–20 (2006)
Tipping, M., Hatton, M.: Drivatar Theory. Online article (2006). http://research.microsoft.com/en-us/projects/drivatar/theory.aspx. Retrieved 7 Sep 2011
Togelius, J., De Nardi, R., Lucas, S.M.: Making racing fun through player modeling and track evolution. In: Proceedings of the SAB Workshop on Adaptive Approaches to Optimizing Player Satisfaction, Roma, Italy, pp. 61–70 (2006)
Tuffery S.: Data Mining and Statistics for Decision Making. Wiley, Chichester (2011)
Tveit, A., Tveit, G.B.: Game usage mining: information gathering for knowledge discovery in massively multiplayer games. In: Proceedings of the International Conference On Internet Computing (IC'2002), Session On Web Mining, vol. III, pp. 636–642 (2002)
Wong, C.-o., Kim, J., Jung, K., Han, E.: Modeling for style-based adaptive games. J. Zhejiang Univ. Sci. A **10**(4), 530–534 (2009)
Zhou, X., Conati, C.: Inferring user goals from personality and behavior in a causal model of user affect. In: Proceedings of the 8th International Conference on Intelligent User Interfaces, pp. 211–218 (2003)

Author Biographies

Ben Cowley Helsinki University, Cognitive Science Unit, Department of Behavioural Sciences, PL9 00014, Helsinki, Finland. He received his Bachelors degree on Information and Communications Technology from Trinity College Dublin, Ireland, in 2003, and his Ph.D. in Computer Science from the University of Ulster, Northern Ireland, in 2009. Initial post-doctoral projects focused on investigating the psycho-physiological correlates of learning in the domain of serious games, in the Centre for Knowledge Innovation and Research at Aalto University, Helsinki. Presently he studies neurofeedback therapy for attentional disorders at the University of Helsinki. Research interests are in computational neuroscience, cognitive science, and attention as a component of positive psychology.

Darryl Charles University of Ulster, Coleraine Campus, Cromore Road, Coleraine, Co. Londonderry, BT52 1SA. Dr. Darryl Charles is a senior lecturer in the School of Computing and Information Engineering at the University of Ulster. Dr. Charles has a BEng. Electrical and Electronic Engineering, MSc Microelectronics and Microcomputer Applications, and a Ph.D. in the area of Computational Intelligence. In the past decade he has published over 40 peer-reviewed games related research papers, and has over 30 peer-reviewed publications on artificial neural networks. He is currently working on serious games projects within health and education contexts.

Michaela Black University of Ulster, Coleraine Campus, Cromore Road, Coleraine, Co. Londonderry. BT52 1SA. She has a BSc in Computing Science and a DPhil in the area of ML. She is currently a Senior

Lecturer in the School of Computing and Information Engineering at the University of Ulster teaching Software Development and Intelligent Systems.

Ray Hickey University of Ulster, Coleraine Campus, Coleraine, Londonderry. BT52 1SA. He has recently retired as Lecturer in the University of Ulster where he taught first Statistics and then Computing Science, especially AI. He graduated in Mathematics from the New University of Ulster in 1973 and obtained his Masters degree from the University of Oxford in 1974. He spent a sabbatical at the Turing Institute in 1989 from which an interest in ML developed. He has published in mathematical journals and in the AI journal and the Journal of Machine Learning Research. With Dr. Michaela Black he undertook research into classification learning under concept drift for British Telecommunications which led to a US patent.