Chapter 12
Game Data Mining

Anders Drachen, Christian Thurau, Julian Togelius, Georgios N. Yannakakis, and Christian Bauckhage

Take Away Points:

1. The data revolution in games – and everywhere else – calls for analysis methods that scale to with dataset size. The solution: game data mining.
2. Game data mining deals with the challenges of acquiring actionable insights from game telemetry.
3. Read the chapter for an introduction to game data mining, an overview of methods commonly and not so commonly used, examples, case studies and a substantial amount of practical advice on how to employ game data mining effectively.

A. Drachen, Ph.D. (✉)
PLAIT Lab, Northeastern University, Boston, MA, USA

Department of Communication and Psychology, Aalborg University, Aalborg, Denmark

Game Analytics, Copenhagen, Denmark
e-mail: andersdrachen@gmail.com

C. Thurau, Ph.D.
Game Analytics, Refshalevej 147, Copenhagen, K 1422, Denmark

J. Togelius, Ph.D.
Center for Computer Games Research, IT University of Copenhagen,
Rued Langgaards Vej 7, Copenhagen, S 2300, Denmark

G.N. Yannakakis, Ph.D.
Department of Digital Games, University of Malta, Msida MSD 2080, Malta

Center for Computer Games Research, IT University of Copenhagen,
Rued Langgaards Vej 7, Copenhagen, S 2300, Denmark

C. Bauckhage, Ph.D.
Fraunhofer Institute Intelligent Analysis and Information Systems IAIS,
Schloss Birlinghoven, 53754 Sankt Augustin

Bonn-Aachen International Center for Information Technology B-IT Dahlmannstraße 2
D-53113 Bonn, Germany

M. Seif El-Nasr et al. (eds.), *Game Analytics: Maximizing the Value of Player Data*,
DOI 10.1007/978-1-4471-4769-5_12, © Springer-Verlag London 2013

12.1 Introduction

During the years of the Information Age, technological advances in the computers, satellites, data transfer, optics, and digital storage has led to the collection of an immense mass of data on everything from business to astronomy, counting on the power of digital computing to sort through the amalgam of information and generate meaning from the data. Initially, in the 1970s and 1980s of the previous century, data were stored on disparate structures and very rapidly became overwhelming. The initial chaos led to the creation of structured databases and database management systems to assist with the management of large corpuses of data, and notably, the effective and efficient retrieval of information from databases. The rise of the database management system increased the already rapid pace of information gathering.

During later years, a virtually exponential increase in the availability of data is emerging in fields across industry and research, such as bio-informatics, social network analysis, computer vision and not the least digital games. Today, far more data is available than can be handled directly: business transaction data, scientific data from, e.g., telescopes, satellite pictures, text reports, military intelligence and digital games (Berry and Linoff 1999; Han et al. 2005; Larose 2004; Kim et al. 2008; Isbister and Schaffer 2008; Mellon 2009; Drachen and Canossa 2009; Bohannon 2010).

Modern digital games range from simple applications to incredibly sophisticated information systems, but common for all of them is that need to keep track of the actions of players and calculate a response to them, as is discussed in most chapters in this book. In recent years, the tracking and logging of this information termed telemetry data in computer science – as well as data on technical performance of the game engines and applications themselves – has become widespread in the digital entertainment industry, leading to a wealth of incredibly detailed information about the behavior of – in the case of major commercial titles – millions of players and installed clients (Mellon 2009; King and Chen 2009; Drachen and Canossa 2011).

Applied right, **game telemetry** can be a very powerful tool for game development (Kim et al. 2008; Isbister and Schaffer 2008; King and Chen 2009). Not only for analyzing and tuning games, QA, testing and monitoring infrastructure (Mellon 2009), figuring out and correcting problems and generally learning about effective game design, but also to guide marketing, strategic decision making, technical development, customer support, etc. However, it is generally far from obvious how to employ the analysis (Kim et al. 2008; Mellon 2009): what data should we record, how can we analyze it, and how should it be presented to facilitate effect transformation of raw data to knowledge that if fully integrated into the organization?

Narrowing the focus to **behavioral** game telemetry, i.e. telemetry data about how people play games (Chap. 2), there are a wide variety of ways that this kind of game telemetry data can be employed to assist a variety of stakeholders (as

discussed in Chap. 3), during and following the development process, e.g., informing game designers about the effectiveness of their design via user modeling, behavior analysis, matchmaking and playtesting, something that is evident from the range of publications and presentations across academia and industry, including: Kennerly (2003), Hoobler et al. (2004), Houlette (2004), Charles and Black (2004), Thurau et al. (2004, 2007, 2011), Ducheneaut and Moore (2004), DeRosa (2007), Thompson (2007), Kim et al. (2008), Isbister and Schaffer (2008), Thawonmas and Iizuka (2008), Thawonmas et al. (2008), Coulton et al. (2008), Drachen et al. (2009, 2012), Missura and Gärtner (2009), Williams et al. (2009, 2008), Thurau and Bauckhage (2010), Pedersen et al. (2010), Yannakakis and Hallam (2009), Weber and Mateas (2009), Mellon (2009), Seif El-Nasr and Zammitto (2010), Seif El-Nasr et al. (2010), Thurau and Drachen (2011), Yannakakis and Togelius (2011), Moura et al. (2011), Erfani and Seif El-Nasr (2011), Drachen and Canossa (2011), Yannakakis (2012), Bauckhage et al. (2012) and Gagne et al. (2012).

There is a wealth of information hidden in good game telemetry data, but not all of it is readily available, and some very hard to discover without the proper expert knowledge (or even with it). This has led to much game telemetry data being tracked, logged and stored, but not analyzed and employed. The challenge faced by the game industry to take advantage of game telemetry data mirrors the general challenge of working with large-scale data. Simply retrieving information from databases – irrespective of the field of application – is not enough to guide decision-making. Instead, new methods have emerged to assist analysts and decision makers to obtain the information they need to make better decisions. These include: automatic summarization of data, the extraction of the essence of the stored information, and the discovery of patterns in raw data. When datasets become very large (we consider any dataset that does not fit into the memory of a high-end PC as large-scale, i.e. several GB and beyond) and complex, many traditional methodologies and algorithms used on smaller datasets break down. In fact, any algorithm that scales quadratically with the number of samples is not feasible to use on large data. Instead, methods designed for large datasets must be employed. These methods are collectively referred to as **data mining**. Data revolutions call for novel analysis methods that scale to massive data sizes and allow for effective, rapid analysis, as well as results that are intuitively accessible to non-experts (Han et al. 2005; Larose 2004).

Using data mining methods in the context of game telemetry data – what we will here call game data mining – we can for example:

- Find weak spots in a games' design (Chap. 7; Thompson 2007; Kim et al. 2008; Drachen and Canossa 2009; Gagne et al. 2012)
- Figure out how to convert nonpaying to paying users (Chap. 4; King and Chen 2009)
- Discover geographical patterns in our player community,
- Figure out how players spend their time when playing (Chaps. 18 and 19, DeRosa 2007; Moura et al. 2011; Drachen et al. 2012)
- Discover gold farmers in an MMORPGs (Thawonmas et al. 2008),

- Explore how people play a game (Chap. 14; Drachen and Canossa 2009),
- How much time they spend playing (Williams et al. 2009)
- Predict when they will stop playing (Mahlman et al. 2010; Bauckhage et al. 2012)
- Predict what they will do while playing (Weber and Mateas 2009)
- Which assets that are not getting used (Chap. 14)
- Develop better AI-controlled opponents or make games that adapt to the player (Charles and Black 2004; Thurau et al. 2007; Missura and Gärtner 2009; Pedersen et al. 2010; Yannakakis and Hallam 2009)
- Explore and use of social grouping (Thurau and Bauckhage 2010) – and much, much more.

This chapter will outline how large-scale data mining can be effectively carried out on game telemetry data (i.e. telemetry from game clients/game servers, which can include data on players), and cover a range of scenarios, from behavior analysis of individual players and how they give rise to patterns, to interpretation of larger scale structures like guilds in massively multiplayer online games. To begin with, we will provide an introduction to data mining in general and game data mining specifically, good data mining practices and methods, as well as notes on tools and challenges to game data mining. This should by no means be viewed as a thorough introduction to data mining – that requires an entire book. Fortunately such books exist, for example Han et al. (2005) is a good starting place for the novice data miner.

Following, the major categories of data mining will be outlined and a number of case studies used to exemplify some of the commonly used game data mining approaches, covering supervised and unsupervised methods, with examples from, e.g., *World of Warcraft* (Blizzard, 2003)*, Tomb Raider: Underworld* (2008, Eidos Interactive) and *Heroes of Newerth* (2010, S2 Games), as well as case examples obtained from concrete production contexts. As the interpretability of data analysis results is important, we focus on methods that go well beyond descriptive techniques to provide more meaningful, useful and actionable data representations, enabling the analysis of player behavior across millions of individuals.

On a practical note, the first half of this chapter (Sects. 12.1 and 12.2) is written for the general audience and does not require previous knowledge of data mining. The second half (Sects. 12.3 and 12.4), which focus on case examples of different data mining methods, use however data mining terminology, includes some use of formulas, and some sub-sections may require knowledge of statistics and dimensionality reduction methods. Section 12.5 takes a specific look at online games, including Free-to-Play (F2P), as these have received particular attention with respect to data mining.

12.2 Data Mining in Games Background and Overview

In this section, data mining is introduced and an overview of the main types of methods presented, leading up to the subsequent sections, which will cover the specific methods employed for analyzing game telemetry data.

12.2.1 What Is Data Mining?

Data mining is a somewhat nebulous concept, and there is no single definition of what it is (Chen et al. 1996; Fayyad et al. 1996; Han et al. 2005). The name itself – data "mining" – is derived due to the similarity between searching for valuable information in large databases and mining rocks for valuable ore. Both imply either sifting through large amounts of material randomly or use intelligent methods to pinpoint where the valuable material is. The term is something of a misnomer though, as the goal of mining data is not data, but knowledge (i.e. "knowledge mining"). However, data mining sounded sexier and became the accepted term, overshadowing other terms such as knowledge discovery, knowledge extraction and pattern discovery, which better describe the complete process. According to the Gartner Group (www.thegartnergroup), data mining is: *"the process of discovering meaningful new correlations, patterns and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques."* Similar to this definition, others usually emphasize the exploration and analysis of large quantities of data, in order to discover meaningful patterns. Data mining forms an amalgam of methods and ideas, drawing inspiration from different fields of science and business, including machine learning, artificial intelligence, pattern recognition, medical science, statistics and database systems – in many ways, data mining has emerged in the space between these fields of research (Berry and Linoff 1999).

Depending on the definition, data mining is either a step in or the whole of, the process of knowledge discovery in databases (KDD), a concept originating from 1989, referring to the non-trivial extraction of implicit, unknown and potentially useful information from data in databases (Berry and Linoff 1999). It used to be that KDD was viewed as the overall process of discovering useful knowledge from data, while data mining was the application of particular algorithms to extract patterns from data without the additional steps of the KDD process, but the difference is largely academic.

The same is the case for the difference between data mining and statistics. Many data mining methods stem from statistics, and data mining itself largely arose due to the need in statistics to adopt and invent new ways of working with huge masses of data. Over the 1990s, working with large masses of data became synonymous with data mining, although the methods applied could also be called statistics. At the mathematical level, there are various arguments that can be leveraged, but the main difference again relates to whether data mining and statistics refers to specific methods, or the entire process of working and extracting meaning from data. If statistics is viewed as a set of methods, and data mining the entire process, they are different – but again, in practice, from the trenches of game analytics, there is not a lot of difference.

Irrespective, there is a very strong human element in data mining. Early definitions of data mining emphasized automatic or semi-automatic methods, but it is important to note that the human element is vital when exploring and analyzing large datasets – data mining has become synonymous with automatic techniques, which has led people to believe that it is a product that can be bought. There is a variety of black box software available on the market, which embeds powerful algorithms, making their misuse proportionally more dangerous. As with any other technology, data

mining is easy to do badly. Analysts may apply inappropriate algorithms to data sets that call for a completely different approach, for example, or models may be derived that are built upon wholly specious assumptions. Therefore, an understanding of the statistical and mathematical model structures underlying the software is required.

Data mining is a discipline. In our view, humans need to be involved at every phase of the data mining process. In a game development context, at least in some steps, those humans need to be the people who design, code, test and ultimately play, games. There is no quick fix for a game developer wanting to employ data mining on e.g. user telemetry data, however, as noted by Larose (2004), purely explorative methods can be quite powerful in their own right, and require much less training and specialized knowledge than semi-automatic or automatic processes, although a general understanding of the data is always required.

12.2.2 The Knowledge Discovery Process in Data Mining

It can be tempting to approach data mining haphazardly, without an overall strategy. Therefore, a cross-industry standard was developed in 1996, which took an industry-, application- and tool-neutral approach to data mining, defining the *Cross-Industry Standard Process for Data Mining* (CRISP-DM) (www.crisp-dm.org) (Chapman et al. 2000). CRISP represents a fundamental good approach to data mining processes, and various specialized variants exist aimed towards particular industries or problems. It can seem a bit cumbersome to apply the full CRISP process to each and every question that we want answered via game telemetry, and in practice, some of the phases will be very quick to go through, especially if the analysis has already been performed on a previous version of a game or earlier user-behavior dataset. However, CRISP provides a non-proprietary and freely available standard process for fitting data mining into the general problem-solving strategy of a business or research organization. It is an iterative process, fitting well into the typical agile and rapid-iterative approaches applied in game development (Mellon 2009). The phase sequence of CRISP is adaptive – i.e., the next phase in the sequence of six defined phases depends on the outcomes association with the preceding phase. For example, it may be necessary to return to the data preparation phase for refinement before moving on with the modeling phase – it is a typical situation that the solution to a question leads to further questions, not the least when working with player behavior in games. Importantly, lessons learned from past projects should be brought to bear as input into new projects.

In the context of game development, the data in question can originate in game telemetry (data from installed game clients) or any of the traditional sources of business intelligence, e.g. production and marketing (Romero 2008; Mellon 2009; Drachen and Canossa 2011). The CRISP phases are as follows (modified from: www.crisp-dm.org, and Larose (2004)):

1. **Business/research understanding**
   - Define the project objectives and requirements.

- Translate the goals and restrictions into the formulation of a data mining problem definition.
- Prepare a preliminary strategy for achieving these objectives.

2. **Data understanding**
   - Collect the data (extract the data from a database).
   - Use exploratory data analysis (EDA) to familiarize yourself with the data and discover initial insights.
   - Evaluate the quality of the data.
   - If desired, select interesting subsets that may contain actionable patterns.
3. **Data preparation (typically the most time-consuming phase)**
   - Prepare from the initial raw data the final data set that is to be used for all subsequent phases.
   - Select the cases and variables you want to analyze and that are appropriate for your analysis (this is sometimes performed after transformation and cleaning).
   - Perform transformations (or consolidation) on certain variables, if needed. Under transformation, the selected data are transformed into the form needed to perform the mining procedure in question, e.g. normalizing values.
   - Cleaning: Clean the raw data (remove noise and irrelevant data) so that it is ready for analysis.
4. **Modeling**
   - Select and apply appropriate data mining technique (modeling, exploration, synthesis etc.). Different techniques may be used for the same problem.
   - If the technique results in a model (most do outside of explorative analysis), calibrate model settings to optimize results.
   - The process can be repeated with new selections and transformations of the data, gradually refining the result or integrating new relevant data sources, in order to get different, more appropriate/valuable results.
5. **Evaluation**
   - Evaluate the results and/or models delivered for quality and effectiveness.
   - Check that the model in fact achieves the objectives set for it in the first phase.
   - Check whether some important facet of the problem has not been accounted for sufficiently.
   - Come to a decision regarding use of the data mining results.
6. **Deployment**
   - The discovered knowledge is presented to the relevant stakeholder (designers, producers, marketing, management), using a choice of knowledge representation, e.g. a visualization or report. E.g., a game telemetry analyst develops a heatmap of a multi-player shooter level to present to the designer of that level (see also Chap. 17).
   - Make sure the presentation is done in such a way that the target stakeholder can understand. Explain the result in a way that helps the target stakeholder to understand, interpret and act upon the data mining results.
   - The target stakeholder carries out deployment within the organization.

12.2.3 Database Navigation

When dealing with data stored in databases – irrespective of the format – there are three fundamental **navigational tasks** that the chosen database format must allow: drilling down (vertical), drilling across (horizontal), and controlling time. The latter is the most obvious: we have to be able to select data from only a specific build, a particular user test, or segment of time. We mention these here because basic navigation of game telemetry data is often the first step needed in data mining, and sometimes the only step needed to answer questions.
Drilling (or navigating) is a method for interactively navigating or exploring data horizontally and vertically in the dimensional structure of a database.

Drill-across (or drill-through) **navigation** occurs across multiple dimensions (used commonly with OLAP, Online Analytical Processing, a class of functions that enable users to dynamically and in real-time analyze and navigate data, e.g. in a data cube), and is used for e.g. comparing different variables for a specific dimension, for example playtime and item purchases for all players from Europe. Similarly, identifying the top ten most profitable players in a free-to-play game from each game server, is an example of a drill-across analysis. Drill-across navigation can operate across dimensions, measures or attributes in OLAP and data warehouses.

Drill-down navigation is a means for exploring data in greater detail (more low-level) than is currently displayed (Kim et al. 2008). The term drill-down is commonly encountered in game data mining contexts. This is because drill-down analysis is a method for in-depth analysis of data, which makes it very useful to especially player behavior analysis, where the root causes of behavioral patterns are often nestled deep within the data – e.g. a specific checkpoint missing, a single mob being too powerful, a pathway between to areas going unnoticed (Kim et al. 2008; Drachen and Canossa 2011).

For example, viewing aggregated completion times across ten levels in a game, and noting that level 5 completion times are very high, drilling down would then be to look at the data for pertaining to level 5 only for each player. This kind of process is commonly used in game analytics to locate the root causes of an observed effect. See e.g. Romero (2008), Kim et al. (2008), Drachen et al. (2009) and Chap. 14 for examples.

To take an example (Fig. 12.1): A game developer considers a simple breakdown of data consisting of a few variables, here the average completion times for the levels of a game (five levels). At this top level, he notices that a level appears to take longer to complete than others (level 4). This is not intended by the design, and could therefore indicate a problem. In order to explore why, the underlying data need to be exposed, in this case, a breakdown of the completion times for individual components of the level. In this more detailed view of the data, it may be noticed that a specific sector of the level is where the players spend a lot of time (Main Hall). Drilling down further into the metrics data captured from the level, it may be found that the root cause is that players have trouble beating a specific type of enemy and keep dying (Evil Conductors – they stamp your ticket really, *really* hard), whose difficulty can be adjusted to accommodate. If the cause of the observed pattern is not obvious from looking at the data, it can be

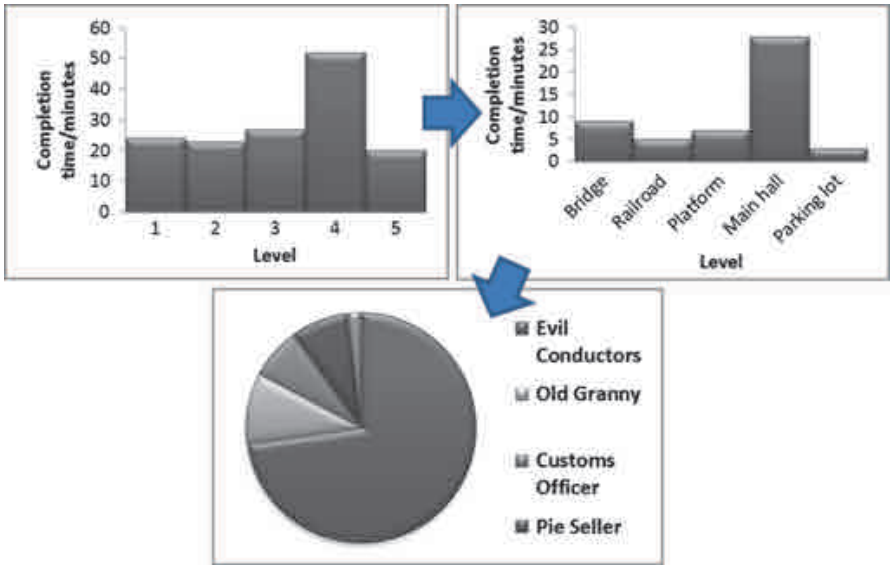

Fig. 12.1 An example of drill-down navigation from a fictive game. See text for explanation

Six Myths About Game Data Mining

1. *With game data mining, we can fire our designers as the testers/users will tell us what they want! We will save heaps of money!* Wrong: mining gameplay telemetry data is incredibly useful for evaluating and testing design, but telemetry data cannot tell you what design is, nor how your players feel or if they have a good experience – game data mining is not a replacement for good game design.
2. *With game data mining, we do not need user testing! We can fire our testing department and save heaps money!* Wrong: Game data mining goes hand in hand with user-oriented testing and -research, but does not replace it. With mining of gameplay metrics data, a powerful supplement to playtesting, usability testing and physiological testing is gained.
3. *Game data mining is autonomous, requiring little human oversight! The tools are automated; we just turn them loose on our data and find the answer to our design/business/marketing problem! We can fire our business analysts and save heaps of money!* Wrong: There are no automatic tools that will solve your problems – data mining is a process, as highlighted by CRISP. Additionally, data mining requires significant human interactivity at each phase, and also for the subsequent quality monitoring.

(continued)

(continued)

4. *Game data mining pays for itself quickly! Let us invest in tools, infrastructure and people right away and save heaps of money!* Wrong: well, partly wrong. The return rates vary, depending on the specific situation, the game, the size of the developer or publisher, etc. The return will be there in terms of improved knowledge, but careful planning needs to go into deciding on an initial setup and the strategic and practical goals.
5. *Game data mining will identify the causes of all our problems! We will make heaps of money integrating game data mining in our business!* Wrong: the knowledge discovery process will help identify and uncover patterns of behavior in the data whether user-derived or business-derived, and these can be highly valuable, but it requires human interpretation to identify the causes of the patterns (with the help of analysis).
6. *We need to obtain data on everything! Data equals value, we will make a heap of money!* Wrong: you need the right data, to solve the problems you have. Just measuring everything will waste resources. Getting the right data requires as much thought as their analysis.

useful to consider the actual game environment – e.g. finding that players do not spot the big weapon they needed in the room preceding the boss. Drill-down analysis work in this way to identify the root cause of patterns that emerge much "higher" in the data. At the lowest level of a drill-down analysis are the raw data.

There are a number of other fundamental actions that users should be able to take in a database system, e.g. filtering, sorting, morphing etc., but a full discussion is out of scope here – the reader is referred to Han et al. (2005) and Larose (2004) for additional information.

12.2.4 Separating Gold from Rock in Data Mining Results

When running a game data mining analysis, for example a classification analysis of the behavior of players during development of a game, there will typically be more than one result. So how do we know which one to pick?

Data mining allows the discovery of patterns in the data, and there may be quite a lot of them if the dataset is big enough and heterogeneous enough. Finding the best, most useful and interesting pattern is not always straight-forward. In order to choose the best pattern (or result), we need to be able to evaluate the patterns based on how "interesting" or useful, they are to the specific situation. There are three approaches that can be employed:

Objective measures: these are based on the war data themselves, e.g., validity of the patterns when tested on new data with some degree of certainty. They are by far the easiest to employ and provide hard numbers to evaluate results.

Subjective measures: these are based on the data and the user of the data. As noted by (Geng and Hamilton 2006) *"to define a subjective measure, access to the users domain or background knowledge about the data is required. This access can be obtained by interacting with the user during the data mining process or by explicitly representing the users' knowledge or expectations."* Novelty or understandability of a result is an example or a subjective measure of interestingness.

Semantic measures: considers the semantics and explanations of the patterns. A good example is "utility" or "actionability" as an evaluation mechanism.

Identifying and measuring the interestingness of patterns is essential for the evaluation of the mined knowledge and the data mining process in general. Concretely, interestingness measures are useful because they can be used to: (1) prune uninteresting patterns during the mining process so as to narrow the search space and thus improve mining efficiency; (2) rank patterns according to the order of their interestingness scores; (3) be used during post-processing to select interesting patterns (Larose 2004). Fortunately, interestingness measures have been the focus of considerable research interest. All of the method groups outlined above have associated suggestions of interestingness measures, although most are objective.

12.2.5 Data Formats

An important aspect of working with game telemetry data is how they are stored and accessed. It is one thing having collected behavioral data from ten million players, another to store these in a way that makes it as easy as possible to apply data mining techniques to them. There are currently a plethora of database formats available, with SQL/MySQL being one of the most commonly used, and used to be the default for new web applications. However, SQL has problems with scaling up to very large datasets, despite recent innovations such as SSD enhancements and 32+ core scalability, and can be overly complex for many operations (and making changes to large databases can be hard). Therefore, in recent years more "elastic" means of data storage, running on cloud computing frameworks with up to 100's of servers, offering scaling on demand. These new database formats are commonly referred to as "NoSQL" (and NewSQL) and have become popular in big data contexts due to the need for fast, efficient data access. A full review of different database formats is dramatically out of scope, but interested readers can find useful information in Chaps. 6 and 7 of this book. It is also recommended to look the NoSQL database formats MongoDB, Cassandra, Couch and HBase (Hadoop), for information on newer database formats. In general, the Net is a good source for information on database formats.

Data mining methods are applicable to any kind of data or media and independent on the specifics of the repository of the data (relational database, unstructured database, multimedia database, time-series database, flat file, object-oriented database, spatial database, etc.). However, algorithms may vary when applied to different types of data, e.g. images vs. behavior measures.

12.2.6 Tools for Game Data Mining

There are a wide variety of software tools available for data mining. Some are specialized for particular sectors of industry or research; others are more open and accommodating to game data mining. However, in our experience, some software vendors have a tendency to market analytical software as being "plug and play" applications that will provide solutions to all kinds of problems without the need for human interaction. This is blatantly not the case there is a strong need for a human element in data mining, possibly especially in games, where the fundamental goal of providing the user with a good experience is at the forefront; results, therefore, need to be interpreted with user experience in mind (Drachen et al. 2009).

In recent years, several companies have started to offer middleware technologies specifically for game data mining or game analytics (e.g. www.gameanalytics.com, www.playtomic.com, www.honeytracks.com, www.kontagent.com), supplementing the tools and services offered by traditional analytics companies. However, game-specific data mining tools remain in their infancy, and traditional game data mining companies, used to working with, e.g., business analysis or web analytics, do not always have the intimate understanding of game design necessary to fully understand game telemetry data, and deliver relevant and interesting results. This has led to several major publishers, notably Microsoft Studios Research, to develop their own solutions to game analytics (Kim et al. 2008). The barrier of entry for non-experts in game design and data mining remains, therefore, relatively high. However, the current rapid development in game telemetry analysis favors a wider availability of solutions and methods evolving over the next few years.

In the open-source market, there are many freely available tools developed by researchers and practitioners that are useful to novices and experts alike, for example tools like Weka (www.cs.waikato.ac.nz/ml/weka/), which is used to supervised learning, RapidMiner, a general data mining tool or Shogun, a library for large scale machine learning (www.shogun-toolbox.org), Pymf, a toolbox for matrix factorization in Python (pymf.googlecode.com), or QGIS for spatial problems (www.qgis.com). There are many of these tools (see e.g. Chaps. 7, 10, and 14). At the practical level, the easiest way to locate an open-source toolbox useful to a particular data mining task is to figure out what type of problem we are dealing with, and then browse the Net for relevant tools.

12.2.7 Practical Issues in Game Data Mining

There are a range of important issues to consider when planning to or performing collection of game telemetry and mining of this type of data. Confidentiality of user data, security of hosting servers, transparency of analysis results, and effective preprocessing approaches are among the most important. In this section, these issues and their implications are briefly introduced, but the interested reader is strongly advised to consult the literature at the end of the chapter for more detail.

Transparency: the patterns discovered by data mining tools are useful only if they are interesting and understandable to the user they are aimed for. Any data mining result (model) should be as transparent as possible, i.e. the result should describe a pattern that is intuitively interpretable and which is followed by an explanation, targeted at the specific stakeholder or user of the result e.g. decision trees are intuitive and almost self-explanatory in terms of their results, but neural networks are comparatively opaque to the non-expert (as are non-linear models in general). For example, a game designer may not be a statistics expert and therefore providing the results of a variance analysis in the standard statistical reporting form (a series of values), will not be conducive to the designer understanding the result and being able to act upon it. Transparency is vital to ensure that the various users of game data mining results are able to understand and act upon them. Another issue in visualizations is screen real-estate, information rendering and user-pattern interaction. Interacting with raw data or mining results is important, because it provides the means for users to focus and refine the mining tasks. Additionally, it allows users to model the discovered knowledge from different angles or conceptual levels.

Data cleaning: data analysis can only be as good as the data that is being analyzed, and most algorithms assume the data to be noise-free. This is an important assumption. Depending on the technical back end, game telemetry data may be more or less complete or saddled with different types of problems. Data cleaning (or cleansing) is the process of detecting and removing inconsistencies from data, towards improving and ensuring the quality of the data (Han et al. 2005; Larose 2004).

Quality problems in raw data come in many forms, e.g. misspellings during data entry, missing information or the presence of invalid data. When multiple sources of data are integrated, for example in a data warehouse, or analysis run across multiple data sources (e.g. telemetry from different games), the requirement for careful data cleaning increases due to the potential for error introduced when datasets are combined.

Performing data mining on low-quality data ("dirty data"), with, for example, missing or duplicate information, can compromise the validity and accuracy of the results, or even worse, can lead to outright wrong results, following the "garbage in, garbage out"-principle in data mining. As a consequence, data cleaning and data transformation (commonly referred to as pre-processing) is vital, but is often erroneously viewed as lost time. As frustrating as data cleaning may be, it is one of the most important phases of the knowledge discovery process. Data cleaning is a complex topic. Unfortunately, it is not possible to provide simple guidelines to address this topic. There is also a general lack of research in the area despite the importance.

Performance and sampling: many methods for data analysis and interpretation were not originally designed for the very large datasets that exist today. In game development, telemetry datasets easily reach the terabyte size for online social games or for large commercial games with hundreds of thousands or millions of players. In addition to the size of the data, the dimensionality of the data, i.e. the number of variables in the dataset (e.g. the number of variables such as completion time, class, level, etc., known for each player in a game), is decisive to the choice of data mining techniques. In general, the search space grows exponentially with the number of dimensions in a dataset, and its effect is so dramatic that it is currently one of the most important research problems in data mining.

Many techniques have issues with scalability and efficiency at large scales and dimensionalities, especially those that scale quadratically with dataset size, or algorithms with exponential or polynomial complexity (Mahlman et al. 2010). Sampling is a possible solution, i.e. mining part of the dataset rather than the whole, and extrapolating results from the sample to the whole dataset. Sampling has its own complexities and challenges, for example in relation to ensuring a representative sample that captures the features of the entire dataset. Sampling is covered in more detail in Chap. 9. Another approach is parallel programming, where the dataset is subdivided and results for each subset merged later.

Security: is an important issue with any game telemetry data collection, whether intended for low-level work or strategic decision making. Game telemetry data are generally considered confidential in the industry, and should be kept safe, which includes considerations on how to handle data access, transfer of data and transfer of results.

Social and privacy issues: One of the key issues in data mining is the question of individual privacy. The immense collections of data on people, and the many opportunities for collecting additional information, combined with data mining, makes it possible to analyze, e.g., routine business transactions, and obtain a substantial amount of information about the habits and preferences of individuals or businesses. Additionally, when data is collected for player profiling, behavior, correlations of personal data with other information, and so forth, sensitive and private information about individuals or businesses is collected and stored. This is controversial given the confidential nature of such data, and the potential illegal access to it. Another issue is how the data is being used. Because this type of data is valuable, databases of all kinds are traded. It is, thus, important to be aware of what data and analysis results that are being distributed, e.g. email addresses of players.

Collection strategies: There are two fundamental ways to obtain data from an installed game client or hardware unit (e.g. Xbox 360, PS3, PSP, smartphone), irrespective of the protocol employed (e.g. restAPI). Choosing the right strategy for capturing data from game clients is vital to avoid excessive data cleaning issues and data loss. There are pros and cons to both approaches, as follows (adopted from Mahlman et al. (2010)):

- **Fire and forget:** game telemetry data are stored locally in queues. Depending on the memory allocated to telemetry tracking, the size of the queue can vary. The game client will attempt to transmit data to the collection server, but may or may not receive confirmation of receipt from the server. If a queue is full, the oldest stored data are deleted first to make space for new data. This solution ensures a specific memory use and is, thus, useful for mobile platforms or consoles where memory resources are limited; or for high-frequency data (e.g. navigation), where some random losses are unimportant.
- **Reliable metrics:** the game client keeps storing telemetry data until they have been successfully transferred to the collection server, and confirmation of receipt has been received. The solution is resistant to loss and useful in situations, where the data must be collected as completely as possible, e.g., during playtests. In both

cases, a key rule is that game execution must not be affected by the collection or transfer of telemetry data to the collection server. The approaches can be combined, e.g., using limited queuing for navigational data and unlimited queuing for important variables.

12.2.8 Data Mining Approaches

It is difficult to generalize about data mining methods given the many fields of research and business that employ data mining techniques. However, the various methods are usually divided into either the categories **descriptive/prescriptive** or **unsupervised/supervised learning**. Depending on the person or book being consulted, either of these two divisions will be used – they are not; however, completely interchangeable – descriptive data mining is not the same thing as unsupervised learning, for example. To be more precise, predictive/descriptive data mining are concepts, and supervised/unsupervised learning are concrete categories of methods – and not the only ones used for data mining, although the main ones. This means that for example correlation methods are referred to as descriptive data mining, but not assigned to the unsupervised learning group of data mining methods. Similarly, interpolation is a technique used for prescriptive data mining, but can in at least some cases be argued to be not a form of supervised learning. As with so many other things, the difference is good to be aware of, but not vital in practice. In this chapter, we adopt the division of methods into unsupervised and supervised categories.

Descriptive data mining is used to describe the general properties of existing data in a concise way. In addition, it presents any interesting characteristics of the data without having a predefined target. For example, exploring the number of daily users and pointing to a sharp increase in active users on a specific day, say Saturdays. Some authors equate descriptive data mining with statistics.

Predictive data mining is used to forecast explicit valued, based on patterns determined from known data. In other words, it is used to attempt to predict something based on inference on the data at hand. For example, predict how many paying users a game will have based on data on previous subscriptions.

Supervised learning originates in machine learning – a branch of artificial intelligence science that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on data. A learning algorithm takes advantage of a test dataset, "training data" (observed examples), to capture characteristics of interest of the underlying, unknown probability distribution of data and make intelligent decisions based on their properties. In supervised learning, training data is combined with knowledge of desired outputs. The output of the algorithm can be a continuous value (regression) or a prediction of a class label of the input object (classification). The task of the supervised learning algorithm (the "learner") is to predict the value of the function for any valid input, after seeing a number of

training examples (i.e. pair of input and target output). In order to achieve this ability, the algorithm has to generalize from the training data to unknown situations in a way that is reasonable. In the context of digital games, predictive data mining can be used to forecast when a player will stop playing, if a player will convert from a non-paying to a paying user, what types of items players will purchase, classify player behavior, etc.

Unsupervised learning also originates in machine learning, and also focuses on fitting a model to observations. However, unlike supervised learning, there is no a priori output. The input objects are generally treated as random variables, and a density model built for the dataset. For example, if we want to classify player behavior, we can use unsupervised learning if we not know how the behaviors varied, or if no previous classes had been defined. We can use supervised learning if, for example, we already run a classification on earlier data, and are interested in fitting some new players into these pre-defined classes.

12.2.9 Data Mining Methods

The classification of data mining methods beyond descriptive/predictive and supervised/unsupervised has always been a somewhat sensitive issue in data mining, leading to some confusion when attempting to learn about the different methods – a popular class or concept may have dozens of different names (Han et al. 2005). In the context of game data mining, some of the most common methods used are:

Description: is when analysts are simply trying to describe patterns of trends in game data, and is usually accomplished using Explorative Data Analysis (EDA), which is a graphical method for exploring data in search of trends or patterns. For example, plotting class level vs. playtime per level in a bar chart across six classes in a MMORPG, and finding that the "warrior" class progresses more slowly than the other classes. Descriptions of patterns often suggest possible explanations for them. EDA is particularly useful for basic analysis and for obtaining an understanding of the game data prior to the application of advanced algorithms.

Characterization: is simply the summation of general features of objects in a target class (or sample), producing a characteristic rule. For example, we may want to characterize all players who complete the first 100 quests in a RPG in less than 5 h (an example of characterization is shown in Fig. 12.2, where telemetry data from the ranges at which weapons were used in a FPS are averaged across the weapons).

Discrimination: is when features of objects across two classes (or samples) is compared. For example, comparing the most popular item purchases for players between 10–15 and 16–20 years, or comparing the navigation path of two types of players through a game level (Chaps. 7, 14, and 19; Drachen and Canossa 2011). Discrimination is identical to characterization except that discrimination results include comparative measures.

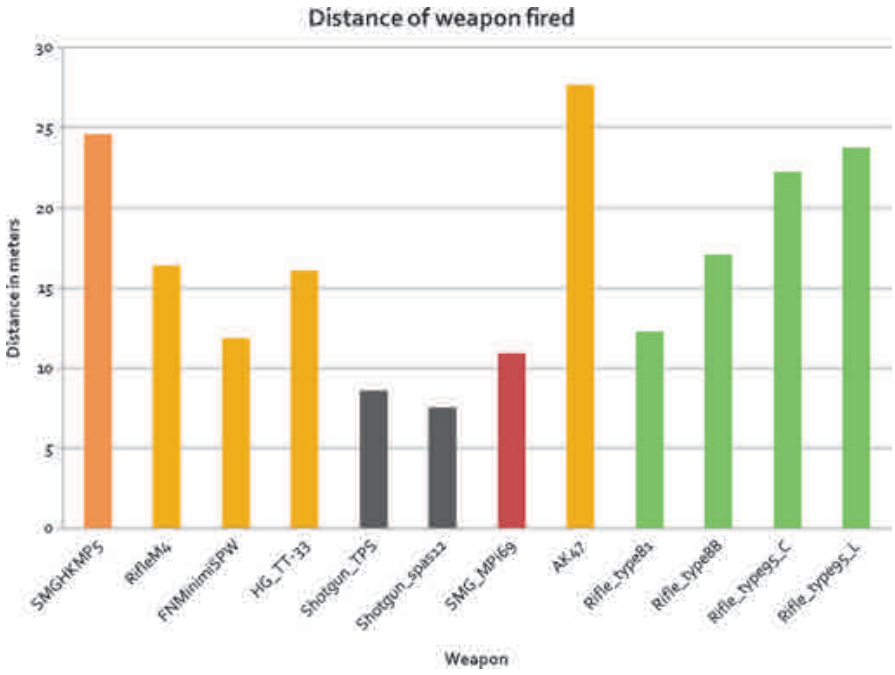

Fig. 12.2 An example of a descriptive analysis: A simple bar chart providing an overview of the average distance at which playtesters of the multiplayer shooter *Fragile Alliance* (Eidos Interactive, 2007) used different weapons, during early production of the game, for a particular map (note that the published version of the game has other rates and weaponry) (used with permission from IO Interactive)

Classification: is used to organize data into classes, which is hugely useful to game development. For example, classifying players based on their potential to become paying users vs. non-paying, or classifying player behavior in a shooter game to test if the players play the game as intended by the games' design. Classification uses class labels to order objects in a data collection, normally using a training data set where all objects are already associated with known class labels (e.g. playtime per level associated with character class). The classification algorithm used leans from the test data and builds a model that can be applied to other or future data.

Estimation: is similar to classification, but the target variable is numerical, not categorical. In statistics, methods such as regression and correlation are estimation methods. For example, we are interested in knowing a value, not obtain information about how our data groups into distinct classes. For example, estimating how much money a player will spend on in-game items, or how long a player will continue playing a specific game. In estimation, models are built using training data of complete records, which provide the value of the target variable as well as the predictors (causal variables). For new observations, estimates of the value of the target variable are made, based on the values of the predictors. For example, using simple regression to find the relationship between two variables, such as playtime and money spent on in-game items.

Prediction: is reminiscent of classification and estimation, but with prediction, we want to know about the future. The core idea is to use a large number of known values to predict possible future values. For example, how many players an MMORPG will have 3 months into the future, or when there will only be 1,000 active players left in a social casual game or how many players are needed to reach the critical threshold when player communities become self-sustaining. There are many approaches to prediction, from traditional statistical methods to more specialized knowledge discovery methods, such as neural networks, decision tree analysis, and k-nearest neighbor (Mahlman et al. 2010). Prediction is one of the most widely applied data mining methods in the analysis of data from multi-player and massively multi-player persistent games, where predicting the effect of design changes or the behavior of the player community, is important for revenue. Prediction can be used to forecast in many contexts around game development and -publishing.

Clustering: is a lot like classification, in that the aim is to order data into classes. However, the class labels are unknown and it is up to the clustering algorithm to discover what the classes are and evaluate their acceptability. The core goal of clustering algorithms is to group or segment objects (e.g. players, asset, items, games or any observation, case or record) in such a way that the similarity between objects in one group (cluster) is high (intra-cluster similarity), while between groups is dissimilar (intercluster similarity).

Association (affinity): when performing an association analysis, the goal is to find features (attributes) that "go together", thus defining association rules in the data. An association rule specifies that if X, then Y, e.g., *"if players buy Stribed Trousers of Strength +3, they will also buy Girdle of Charisma +2."* The association rule is accompanied by a measure of support, and of confidence. The support threshold identifies the frequency of the features occurring, and the confidence threshold defines the probability one appears when the other does. For example, it may be found that out of 1,000 players, 500 bought the *Stribed Trousers of Strength +3*, and of those 500, 250 bought a *Girdle of Charisma +2*. The association rule then specifies: "if players buy *Stribed Trousers of Strength +3*, they will also buy *Girdle of Charisma +2,* with a support of 50."

There are many other methods that can be used for game data mining, such as outlier analysis (looking at the exceptions to normal behavior, which can be pretty useful in the analysis of player behavior, e.g. for locating gold farmers), evolution analysis, and deviation analysis (the investigation of time related data that changes as a factor of time). However, these are out of scope for this chapter. The reader is referred to the reference list for further information.

12.3 Unsupervised Methods

As discussed above, unlike supervised models, in unsupervised learning there is no *a priori* defined output, i.e. we are not trying to predict target values, but rather focus on the intrinsic structure of and relations in the data. In particular, the data

mining algorithms searches for patterns among all variables of a given dataset. A traditional example is clustering, e.g. for classifying player behavior, causes for game crashes, etc. In such examples, we are not typically sure how these behaviors vary or whether particular causes are more typical than others. In the following sections some basic mathematical properties are described in the interest of accuracy.

In this section, we concentrate on a few examples of the application of unsupervised models for analyzing game telemetry data. We will give an overview over a few common methods applicable for unsupervised data analysis in games. We will demonstrate the usefulness of recent data mining techniques in terms of acquiring interpretable data representations.

12.3.1 Clustering

In the context of customer behavior analysis in computer game development, cluster (and classification) analysis provides a means for reducing the dimensionality of a dataset in order to find the most important features, and locate patterns which are expressed in terms of user behavior as a function of these features, which can be acted upon to test and refine a game design (or specific parameters of a design) (see Drachen et al. 2012, for a more in-depth discussion). For example, figuring out how people play the game or identifying groups of players who display unwanted behavior. Clustering is thus a highly useful data mining method, containing a plethora of algorithms, the most commonly used being k-means (Golub and van Loan 1996).

There are however notable challenges (Drachen et al. 2012):

1. The potentially high dimensionality of behavioral data from games
2. There is sometimes a need to mix datatypes, e.g. binominal and categorical features which makes normalization challenging
3. Telemetry datasets can be noisy
4. Clustering generally require informed decisions as to the number of clusters to extract
5. The results have to be actionable. What this means is that it should be possible to relate the results to the design of the game in question, which entails converting results to a language understandable by the target stakeholder group (designers, marketing, management etc.).

Successful clustering of player behaviors in computer games requires that these challenges are addressed. Furthermore, it is important to note that the integration of knowledge of the design of the game being investigated is necessary to guide the process of selecting which behavioral variables (or features) to work with. Also, depending on the goals of the analysis, different clustering algorithms may be more or less applicable (Thurau and Drachen 2011), because the algorithms have different properties (this is discussed in further detail below).

To exemplify the process of clustering, we applied k-means clustering to a simple bivariate (playtime and character level) dataset derived from *World of Warcraft* (approx. 70,000 characters) Fig. 12.3 shows the result, indicating that there are

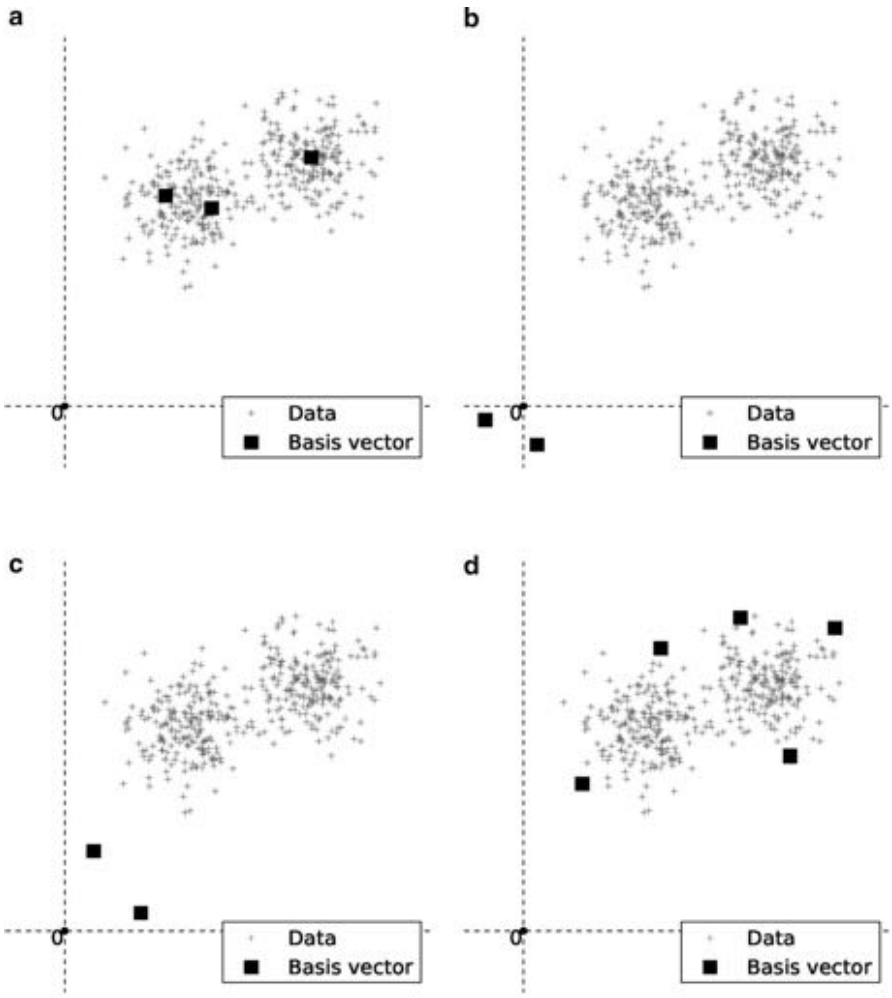

Fig. 12.3 Different cluster/matrix factorization methods can yield completely different views on the same game telemetry data. (**a**) k-means clustering, (**b**) PCA, (**c**) NMF, and (**d**) Archetypal analysis

roughly two or three separate "clouds" of behaviors in the dataset. It can be seen that the resulting cluster centroids (the central point in each cluster cloud) or basis vectors reside within the data and represent certain cluster regions (top left diagram), i.e. one particular cluster center can now be used to represent a vast number of data samples. Each data sample is assigned to exactly one cluster centroid. This is, arguably, the most common way of cluster analysis as it tries to approximate larger dense data distributions by one particular cluster centroid. However, various other unsupervised methods exist (e.g. Principal Component Analysis, Archetypal Analysis and Non-negative Matrix Factorization – results of which are shown in the

other diagrams of Fig. 12.3) in that yield different cluster centroid locations and are less restrictive with respect to cluster membership of individual data samples. Often, these methods have advantages over the standard mean-based approach and can lead to more interpretable data representation (Thurau and Drachen 2011; Drachen et al. 2012).

12.3.1.1 Clustering – Formal Basis

Mathematically, when running a cluster analysis we are dealing with n samples of d-dimensional vectorial data gathered in a data matrix $\mathbf{V}^{\mathbf{d}\times\mathbf{n}}$. The problem of determining useful clusters corresponds to finding a set of $k << n$ centroid vectors $\mathbf{W}^{\mathbf{d}\times\mathbf{k}}$ (note: not all clustering methods use centroid vectors, see e.g. AA below). If we express the membership of data points in $\mathbf{V}$ to the centroids in $\mathbf{W}$ using a coefficient matrix $\mathbf{H}^{\mathbf{k}\times\mathbf{n}}$, we note that clustering can be cast as a matrix factorization problem which aims at minimizing the expected Frobenius norm $\left\|\mathbf{V}-\mathbf{WH}\right\|$. For example, for k-means clustering, where each data sample exclusively belongs to a particular cluster center, the columns of $\mathbf{H}$ are all zeros, except the row to the i-th cluster centroid which is 1, assuming the i-th cluster centroid is the closest.

Generalizing clustering as a matrix factorization task immediately extends the range of applicable approaches. Common methods to achieve the desired factorization include principal component analysis (PCA) (Jolliffe 1986; Golub and van Loan 1996), non-negative Matrix Factorization (NMF) (Paatero and Tapper 1994; Lee and Seung 1999), or Archetypal Analysis (AA) (Cutler and Breiman 1994), among others. However, resulting basis vectors (or cluster centroids) $\mathbf{W}$ considerably differ among the mentioned algorithms. While all mentioned methods roughly try to minimize the same criterion (the expected norm $\left\|\mathbf{V}-\mathbf{WH}\right\|$), they impose different constraints that yield different matrix factors. For example, PCA (Fig. 12.3b) constrains $\mathbf{W}$ to be composed of orthonormal vectors and produces a dense $\mathbf{H}$, k-means clustering constrains $\mathbf{H}$ to unary vectors, and NMF (Fig. 12.3c) assumes $\mathbf{V}$, $\mathbf{W}$, and $\mathbf{H}$ to be non-negative matrices and often leads to sparse representations of the data. While the mentioned factorizations have their specific application in data analysis, it is often not obvious which method to choose for a particular task. Therefore, we will first take a closer look at the specific requirements of data analysis in games.

A common goal of unsupervised data analysis in games is player categorization, or grouping (the supervised learning equivalent is classification), ideally resulting in representations of the telemetry data which is interpretable by non-experts. Ideally, one could assign a simple expressive label to each found basis vector or centroid. While there is no objective criterion on what a descriptive representation is, it is widely assumed that approaches yield interpretable results when they embed the data in lower dimensional spaces whose basis vectors $\mathbf{W}$ correspond to actual data points. This is e.g. the case for Archetype Analysis (AA) as the basis vectors or archetypes the method produces are restricted to being sparse mixtures of individual data points. This makes the method interesting as a means for game data mining as it does not require expert knowledge to interpret the results. This contrasts with

other dimensionality reduction methods, such as PCA (Jolliffe 1986), where the resulting elements can lack physical meaning (Fig. 12.3), and NMF, which yields characteristic parts (Fig. 12.3) (Finesso and Spreij 2004). K-means clustering is similar to AA as the basis vectors reside within cluster regions of the data samples. However, the centroids do not necessarily have to reside on existing data samples.

Taking a closer look at Archetypal Analysis, we note that it uses a constraint that expresses data as convex combinations of certain points in **V**, exemplary resulting clusters (Fig. 12.3). It can be seen that the resulting basis vectors come to reside on the convex hull of the data distribution, and thus, unlike most other methods, data is expressed by the most extreme and not the most average samples. Searching for certain extremal elements in a set of data as it is done for AA accommodates human cognition, since memorable insights and experiences typically occur in form of extremes rather than as averages (on a side note, Philosophers and Psychologists have noted this for long, since explanations of the world in terms of archetypes date back to Plato). In contrast, k-means clustering focuses on the average, and is therefore in the context of other centroids usually more difficult to interpret. While the centroid vectors all cover different regions of the data space, their overall similarity is often too high as it would help a human observer in assigning it a concrete label, i.e. description of the cluster.

The AA problem can be formulated as $\mathbf{V} \approx \mathbf{VGH}$ where $\mathbf{G} \in \mathbf{R}^{n \times k}$, $\mathbf{H} \in \mathbf{R}^{k \times n}$ are coefficient matrices such that **H** is restricted to convexity and **G** is restricted to unary column vectors $\mathbf{1}^T \boldsymbol{h}_j = 1, \boldsymbol{h}_j \mathbf{1} \geq 0$, and $\mathbf{g}_i = [0,\ldots,0,1,0,\ldots,0]^T$ In other words, the factorization approximates **V** using convex combinations where the basis vectors $\mathbf{W} = \mathbf{VG}$ are data points selected from **V**. The goal now is to determine a basis that minimizes the Frobenius norm $E = \left\|\mathbf{V} - \mathbf{VGH}\right\|^2 = \left\|\mathbf{V} - \mathbf{WH}\right\|^2$.

When minimizing the Frobenius norm, we have to simultaneously optimize **W** and **H**, which is generally considered a difficult problem and known to suffer from many local minima. AA, as introduced in (Cutler and Breiman 1994), applies an alternating least squares procedure, where each iteration solves several constrained quadratic optimization problems. It solves the case where **G** is restricted to convexity instead of to unarity. It is important to note that Archetypal Analysis originally was restricted to smaller datasets due to the demanding computation; very recent work has discovered ways of extending Archetypal Analysis to large-scale datasets (Thurau et al. 2009, 2010), making the method effective for implementation in the context of game metrics. Namely, the authors introduced convex-hull non-negative matrix factorization (CHNMF) and simplex-volume maximization as an approximation to AA (Thurau et al. 2009, 2010) (A Python implementation of the two methods is available from pymf.googlecode.com).

12.3.1.2 Example 1: Clustering Players in Battlefield 2: Bad Company 2

The following case study is drawn from Drachen et al. (2012), and is focused on

Battlefield 2: Bad Company 2 (*BF2BC2*) (2010, Electronic Arts), a first person shooter with tactical wargame elements, usually played in online multiplayer supporting up to 32 players, but including off-line (single-player campaign) capability.

In the multi-player mode of BF2BC2, each player controls one character in a team, playing against another team. There are various types of modes of play, and players can select between a range of classes, referred to in the game as "kits". These are: Assault, Demolition, Specialist, Recon and Support. Each class provides different starting equipment. In addition, players can earn awards, ranks and special equipment.

Drachen et al. (2012) used behavior telemetry data from randomly selected 10,000 BF2BC2 players, all playing on PC. A total of 11 variables (features) were included in their analysis, with some of these being compound features. Given the hundreds of possible behavioral variables that can be tracked from players in BF2BC2, selecting these 11 required consideration. Drachen et al. (2009), working with data from *Tomb Raider: Underworld*, suggested that any initial and explorative cluster or classification analysis of player behavior should focus on behaviors related to the central mechanics of a game, and this principle was adopted, leading to a selection of features relating to character performance (score, skill level, accuracy etc.) and game asset use (kit stats, vehicle use), and playtime – as follows (quoted from Drachen et al. 2012):

- **Score:** Total number of points scored
- **Skill level:** An aggregate measure of player skill
- **Total playtime:** The sum total of time the player´s account has been active
- **Kill/Death ratio:** K/D ratio, the number of kills the player has scores divided with the number of deaths suffered
- **Accuracy:** The percentage of hits scores with weapons
- **Score per minute:** The average number of points scored per minute of play while on active combat missions
- **Deaths per minute/Kills per minute:** Dpm/Kpm – Average deaths or kills per minute
- **Rounds played:** The number of game rounds the player has played
- **Kit stats:** The number of points scored with each kit (class) and the number of kills and deaths for each class
- **Vehicle use:** Total time spent in air, water, land-based or stationary vehicles

Following pre-processing and normalization of the telemetry data, two algorithms were applied to the data: k-means, which produce cluster centroids (Fig. 12.3), and Simplex Volume Maximization (SIVM), a variant of Archetype Analysis extended to large-scale datasets. SIVM does not look for commonalities between players, but rather archetypical (extreme) profiles that do not reside in dense cluster regions, but at the edges of the space spanned by the data points (Fig. 12.3). Both algorithms resulted in seven clusters, but the behavioral profiles that could be extracted from these varied somewhat – this is to be expected given the different natures of the algorithms. This number was decided upon using Scree plots and means squared error, two techniques for deciding on the number of clusters to work with. We will here focus on the results from the SIVM analysis, which resulted in the following behavioral profiles, three of which are largely independent of the classes in the game, and four which are closely related to them:

- **Assassins:** characterized by having extremely high Kill/Death ratios and highest Kpm ratio, but surprisingly low-middle playtime. Assassins are the most lethal players in the game, but also highly specialized.
- **Veterans:** are the all-round elite. Where the Assassins are specialized, the Veterans display the highest or second highest values across all the behavioral variables measured, but have also invested a lot of playtime into the game, indicating that these players are committed and stable. They represent a small fraction of the players, however, on the scale of 2–4%.
- **Target dummies:** These are the opposite of the Veterans, with lowest or very low values for all the behavioral variables, comprising about a quarter of the players in the sample. They have not played BF2BC2 for long, have low K/D ratios, often get killed, and their Score per minute is the lowest of all the profiles. Their only redeeming factor is a middling Accuray.
- **Assault-Recon:** These players display high performance with the Assault and Recon kits, correlating with high kill rates and death rates (they are on the frontline), and the second highest K/D rate overall. They also exhibit low accuracy, which may relate to the rapid-fire weapons associated with the assault class. Only about 1.5% of the players are included in this cluster.
- **Medic-Engineer:** These players have very high skill levels and accuracy, score many points (second only to Veterans) and drive in vehicles a lot. Only about 1% of the players are included in this cluster, representing a highly specialized type of behavior.
- **Assault "specialist":** While this cluster of players mainly plays the assault class, they do it relatively badly. They die a lot, but have invested a lot of playtime into the game, with low skill, K/D ratio and accuracy. They are not quite at the level of the Target Dummies, but perhaps represent the typical novice player. About 5% of the players fall into this cluster.
- **Driver Engineers:** These players favor the Engineer class and have extremely high vehicle times (4 times higher than any other cluster), i.e. they spend a lot of time driving, sailing or flying the various kinds of vehicles in BF2BC2. They have high playtimes, scores and accuracy, very high K/D ratio but kill very few players, and also die rarely. Only about 1% of the players are included in this cluster.

The latter four behavioral profiles represent well two of the fundamental ways of playing *BF2BC2*, either combat-oriented or support-oriented.

12.3.1.3 Example 2: Comparing Clustering Algorithms in *World of Warcraft*

Our intention here is to demonstrate how common clustering techniques perform on game metric data with respect to (a) descriptive representations, and (b) cluster separation. Four different clustering algorithms are applied to find clusters in this dataset, the results compared and evaluated, and recommendations made. The example presented here is drawn from Thurau and Drachen (2011).

The data for this case study contains a selection of approximately 70,000 player records, covering a period of about 5 years. The telemetry data are player/guild logs gathered from *WarCraft Realms* (http://www.warcraftrealms.com). The logs show for a certain number of dates the records of currently online players from European and United States *World of Warcraft* realms. In addition, character names, level, class, and guild membership are recorded.

The *World of Warcraft* dataset contains a set of players recordings, their online time, and their level for a specific date. We aggregate the recordings into a 2:555 dimensional feature vector, where each entry corresponds to the level the player reached for each day in the last 6 years. Note that the maximal level of a character was increased twice via expansion packs (from levels 60 to 70 and 70 to 80) during the period of recording (and in December 7th 2010 a third time, following the end of the data logging period, from levels 80 to 85), usually when a new expansion got released. We applied AA, NMF, k-means, and PCA to the dataset. Note that unsupervised methods usually suffer from the problem of having no objective way of defining threshold values, which makes the definition of the number of classes (or cluster centroids) to use a subjective decision. These aspects of classification analysis add to the difficulty in adopting these methods by non-experts in a game design/development context. For the presented experiments we set the number of basis vectors/classes to $k=8$ (note that we only visualized the first five) based on a consideration of variance explained vs. retaining a useful number of basis vectors with respect to the end goal being to produce player classes that are significantly different behaviorally.

The resulting basis vectors or cluster centroids for AA are visualized in Fig. 12.4, and for PCA, K-means and NMF in Figs. 12.5, 12.6, and 12.7, respectively. For AA, for example, the left most plot shows the level/time history plot of a specific player who only very slowly increased his experience level from level 10 to level 20, and Fig. 12.4 (second plot from the left) shows a player who quickly increased his level to 70, and then after some time to level 80. These two player types can be immediately labeled as: "casual player" and "hardcore player". Comparing the resulting basis vectors of the different methods shows that only for k-means clustering and AA we obtained an interpretable factorization. However, the k-means centroids (Fig. 12.5) are overall very similar and do not allow a straight-forward labeling. Basically, they all show is the same curve where only the slopes vary slightly. In contrast, the AA basis vectors in Fig. 12.4 are intuitively easier to interpret. From these, we can also make assumptions about the leveling behavior of the players. The steepest increase in the level seems to correlate with the release of expansion packs and the simultaneous increase of the maximal level. The basis vectors of PCA and NMF are, as expected given the nature of the algorithms, not or only partly interpretable. However, this does not necessarily mean they are useless. We could think about various tasks were a representation of individual by meaningful parts (NMF) is desired. For example, it is reasonable to assume that social groups (guilds in *World of Warcraft*) consist of linear non-negative combinations of meaningful parts, e.g. leaders and followers. This could be captured more accurately using NMF, as it does not restrict the basis vectors to actually existing data samples.

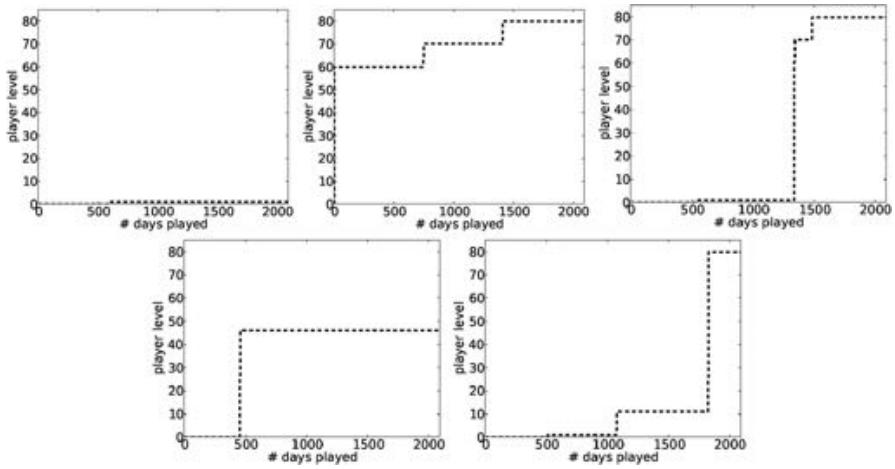

Fig. 12.4 Basis vectors for Archetypal Analysis. These reside on data samples (players in this case). All basis vectors correspond to legal player behavior (e.g. players do not loose levels). Note the straight line segments which map directly to level increases

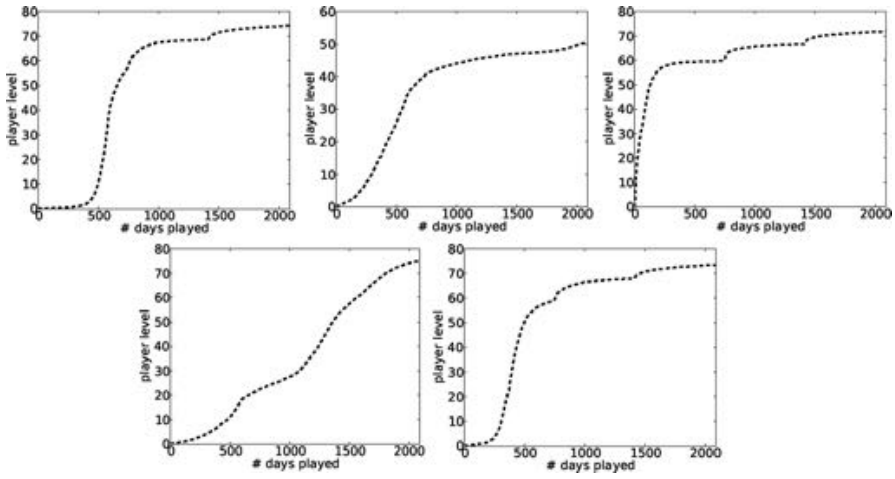

Fig. 12.5 Cluster centroids for k-means clustering reside on center locations of cluster regions. While they accurately represent a broad number of players, they are overall very similar to each other and do not allow straight forward interpretation

Besides a descriptive representation a quantitative discrimination of player types is desirable, i.e. how many players that belong to each behavioral class. This, however, is only fully supported using k-means clustering as it is the only method that builds hard cluster assignments, with each sample belonging to only one particular cluster. The other methods are usually soft (or more precisely linear, convex, or non-negative) combinations of their basis vectors. This means that players are expressed in terms of

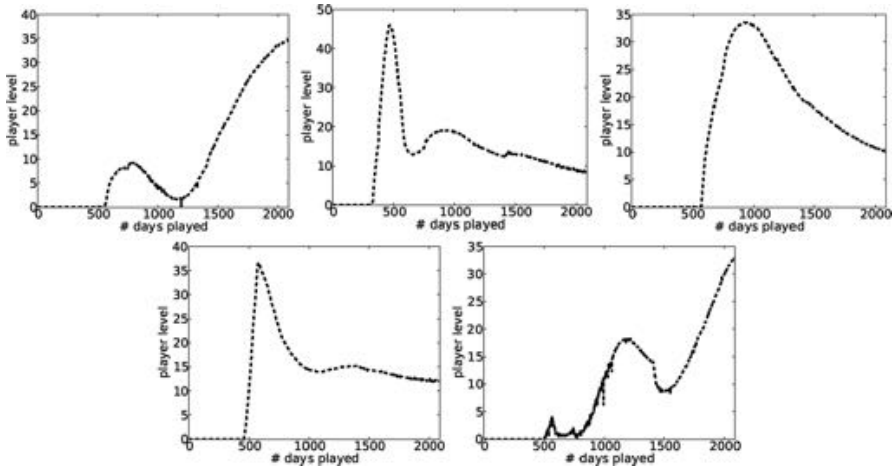

Fig. 12.6 Basis vectors for non-negative matrix factorization represent parts of original data samples. As they are strictly positive, they allow for interpretation but do not in this case correspond to actually existing players or behaviors that are possible in the game, e.g. characters are seen to loose levels

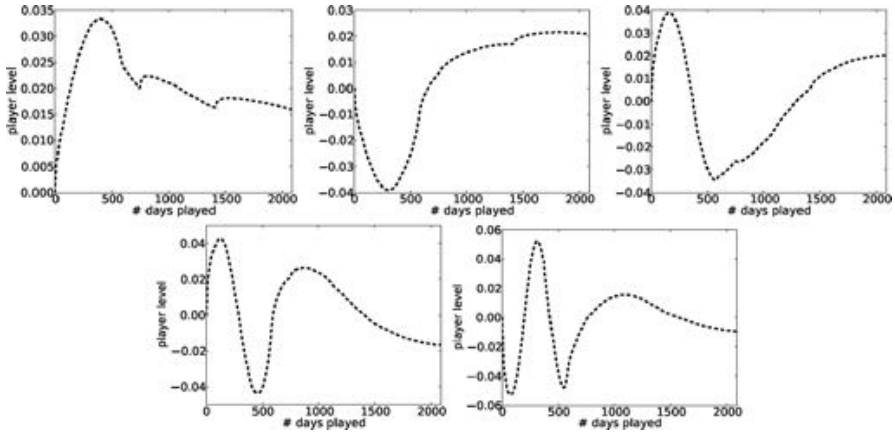

Fig. 12.7 Basis vectors for principal component analysis. These do not correspond to actual players, and correspond to behaviors that are not possible in the game, e.g. loss of character levels

their relationship to each of the eight behavioral profiles (basis vectors) located, and summarily grouped (clustered) according to their distribution in the space spanned by the basis vectors. For the numbers of players belonging to each basis vector provided here, players have been assigned to the nearest basis vector (behavioral profile). This provides clear profile divisions; however, a more precise way of grouping players is to define clusters in the space extended by the eight basis vectors. The results indicate that the distribution of players to eight basis vectors across the four methods included are not similar, with AA and PCA indicating three large groups, and k-means and NMF a division into four large and four smaller groups each (Fig. 12.8).

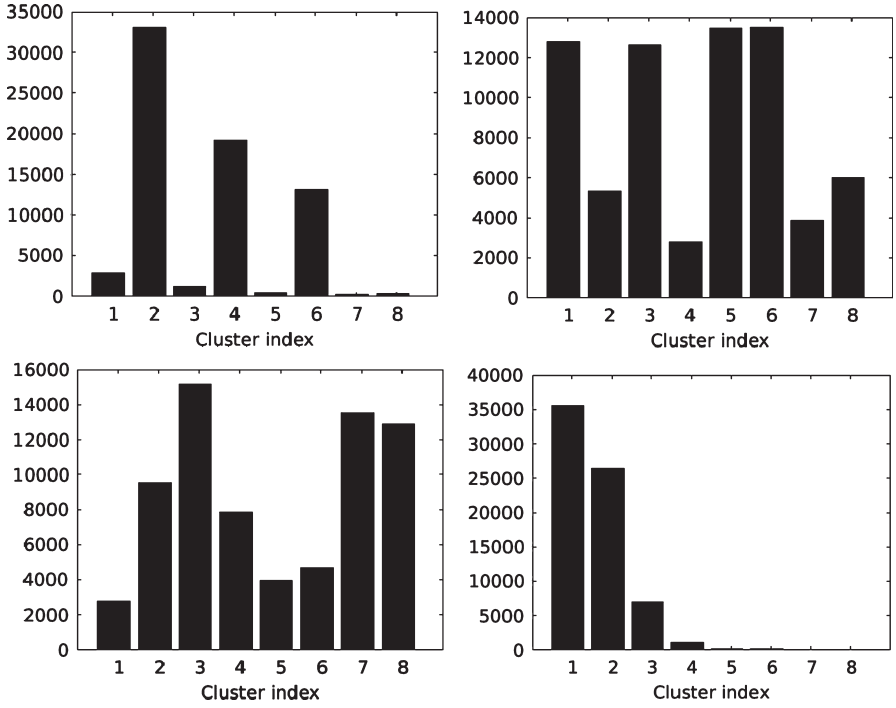

Fig. 12.8 Hard assignment of data samples to cluster centroids for (from top left and clockwise) AA, k-means, PCA and NMF. The bar charts highlight that the solutions generated by the four algorithms varies

12.3.1.4 The Evolution of Social Groups in *World of Warcraft*

Extracting meaningful information from very large amounts of data is a non-trivial task. Especially, if it is not entirely clear what to look out for. In these situations data mining resembles the proverbial search for a needle in a haystack.

Thurau and Bauckhage (2010) proposed the use of Convex-Hull Non-Negative Matrix Factorization (CH-NMF) as an efficient approach towards AA-like data embeddings by means of constrained matrix factorization. The goal was to try to get an accessible and interpretable description of very large amounts of game telemetry data, this towards developing an analysis of the development of guilds over time.

The dataset used by Thurau and Bauckhage (2010) in this example is similar to the one used in the above example (Section 12.3.1.3), but consists of 192 million recordings of 18 million characters belonging to 1.4 million guilds, and cover a period of 4 years, starting in 2005 (when *World of Warcraft* was released) and ending in early 2009.

The data recorded (roughly) summarizes some of the social in-game activities of players. That is to say, we know when players joined or left a guild, how many players were with a guild at what time, and how character experience levels were distributed among the members of a guild, as well as information about class and race. As mentioned before, the player's experience level provides a measure of skill for a

particular player. While a guild with a large number of high level players is more likely to be successful, a guild of only low level players is basically excluded from a large amount of the game content.

The distribution of experience levels among guilds, i.e. the number of players of a certain level that are with a particular guild, therefore provides a feature that characterizes a guild in terms of game success. The distribution can be approximated by means of building a histogram over experience levels of guild members (e.g. eight bins of ten levels each). If we build these histograms over all observations of a specific period of time, they also summarize the temporal evolution of a guild. A brief example should clarify this: If a guild is newly formed by level 80 players, it does not contain any observations of level 10 players and the corresponding histogram bin will be empty. A guild which is formed by level 10 players should, over a longer period, also have observations of level 40, 60, 80 (and intermediate levels) players, as the guild members usually increase their level over time.

In order to obtain an interpretable categorization of, Thurau and Bauckhage (2010) applied Archetype Analysis (more precisely its large-scale variant CH-NMF (Thurau et al. 2009)) to 1.4 million such guild histograms, containing data covering a period of 4 years. The result suggested eight clusters (basis vectors). A number of different basis vector numbers were tested, but it was found that eight basis vectors provide a convenient tradeoff between granularity and convenient visualization.

Following the definitions of CH-NMF, each basis vector resides on the convex hull (the outer surface of the point cloud in multi-dimensional space) of all the individual guild histograms, and thereby each basis vector represents an "archetypal" guild.

As noted before, this makes the basis vectors easy to interpret as there is usually only one salient characteristic – e.g. a guild comprised only of level 80 characters, or where the players level very rapidly from 0 to 60. Figure 12.9 shows a (constructed – due to copyright rules the original illustrations could not be presented here) example of a cluster centroid, i.e. an archetypal histogram. In the current case, the archetypal guilds are distinguishable from each other:

The eight basis vectors describe the following types of overall guild behavior:

1. Formed early, then disbanded
2. Active till the 2nd game expansion (*Wrath of the Lich King*), then disbanded
3. Seldom active
4. Formed before 2nd game update, then very active
5. Increasing activity, then disbanded
6. Increasing activity till the first expansion (*The Burning Crusade*), then disbanded
7. Active for character levels 10–80
8. Active between the 1st and 2nd expansion only

A wide variety of guilds are formed by a convex combination of these archetypal guilds, e.g. a guild can exhibit traits of both guild 2 and 5 for example.

Running the same data through k-means, Thurau and Bauckhage (2010) found that many of the resulting basis vectors were not as readily interpretable as the CH-NMF results, and the basis vectors tending towards being similar (an effect of the distribution of the data points in the variability space as well as the centroid-seeking behavior of the k-means algorithm).

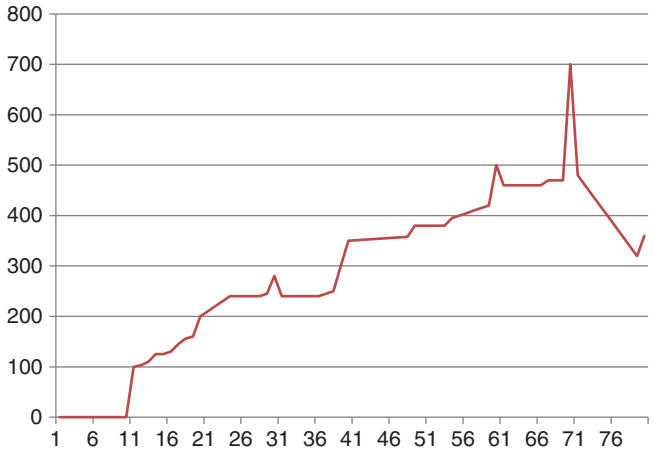

Fig. 12.9 Example of a basis vector resulting from the application of CH-NMF to the *World of Warcraft* guild dataset (constructed example). The x-axis denotes the level histogram bin, the y-axis denotes the number of observations for this bin. The guild described here has a gradually increasing number of players in the lower levels, with a noticeable spike and plateau structure at level 60–70, and a major spike at level 70

In order to obtain a better understanding of the development of the guilds over time, Thurau and Bauckhage (2010) projected time slices (90 days, 180 days, 1 year, 2 years, 3 years, 4 years) of the guilds into the space spanned by the CH-NMF basis vectors. They noticed that the total number of guilds (including disbanded guilds) increased considerably over time, following a roughly exponential growth rate, but also with a high abandonment rate. Also, a huge part of the guild space is densely covered – most guilds fall into the category of seldom active guilds (this could also indicate very small guilds) or are close to it. There is only a small number of guilds (still, many thousands) that completely fall into other categories (the eight basis vectors mentioned above).

On a final note, Thurau and Bauckhage (2010) did not find any significant differences between the development of guilds on US and EU servers.

12.3.2 Player Classification in Tomb Raider: Underworld

A Self-Organizing Map (or SOM) is a form of artificial neural network that is used in unsupervised learning to look for low-dimensional representations of the input data, similar to multidimensional scaling (Summit Kohonen 2001). For example, to find the main ways in which a group of people play a game. The input data are in this case gameplay metrics, the output are the classes into which the players are collected, along with the properties of the classes. For example, one class might be characterized by completing the game really fast, another completing the game really slow, and the third by not completing it at all.

SOMs (also called Kohonen maps after the inventor) work like most neural networks, in that the first step is building a model based on a training dataset, subsequently applying the model to map the main dataset. For example, in a 10,000 player sample, 1,000 could be used as training data, and then the SOM model achieved applied to the remaining 9,000. Without going into details, an SOM consists of neurons organized in a low-dimensional grid (usually two or three dimensions only). According to Drachen and Canossa (2009): "Each neuron on the grid (map) is connected to the input vector via a d-dimensional connection weight vector, **m = m1, m2, md**, where *d* is the size of the input vector, *x*. The connection weight vector is also named prototype or codebook vector. In addition to the input vector, the neurons are connected to neighbor neurons of the map through neighborhood interconnections which generate the structure of the map: rectangular and hexagonal lattices organized in two-dimensional sheet or three-dimensional toroid shapes are some of the most popular topologies used." Please see (Summit Kohonen 2001) for a more detailed description.

Drachen and Canossa (2009) provide an example of how to field SOMs in practice. They used gameplay metrics data from 1,365 players of *Tomb Raider: Underworld*, including data on completion time, number of deaths, causes of death, etc. An SOM was used to find the emergent structures in the data, i.e. to classify the players into distinct groups based on their behavior. The analysis revealed four distinct classes of behavior, encompassing 93.54% of the player sample:

Cluster 1 (Veterans): (8.68%) characterized by having very few death events, and these mainly caused by the environment. Fast completion times. Generally perform very well in the game.

Cluster 2 (Solvers): (22.12%) die rarely, and very rarely use the help system in TRU, apparently preferring to solve the many puzzles in the game themselves. Take a long time to complete the game, indicating a slow-moving, careful style of play.

Cluster 3: (Pacifists): (46.18%) form the largest group of players, characterized by dying primarily from enemies. Completion time relatively fast and help requests minimal indicating some skill at playing the game in terms of navigation, but not a lot of experience with the shooter-elements of *Tomb Raider: Underworld* (the game used shooting substantially more than previous iterations of the Tomb Raider series).

Cluster 4: (Runners): (16.56%) die often and by enemies as well as the environment, use the help system fairly often but complete the game very fast.

The results showcase how SOMs are useful to evaluate game designs. In this case, the analysis indicates that players of the game utilize the affordances provided by the game, rather than simply adopting a specific strategy to complete the game. When evaluating if people play a game as intended by the design, the type of results generated by SOMs are immediately useful. However, the results of an SOM analysis are not intuitively understandable, and need to be translated into language that the intended user of the analysis can act upon. Finally, SOMs provide a good first-strike method for classifying player behavior, providing an overall view useful in guiding drill-down analysis.

12.3.3 Frequent Pattern Mining

Frequent pattern mining is the name used for a set of problems and techniques related to finding patterns and structures in data. While related to other unsupervised learning problems and techniques, such as clustering, frequent pattern mining differs both in the methods and in the format of input and output data — in particular, the latter is discrete and in the form of sets or strings. Several important problems in game data mining, e.g. player type identification and identification of common player behavior patterns, can be cast as frequent pattern mining problems of some form. The whole field of frequent pattern mining is less than two decades old (introduced in Agrawal et al. (1993)), yet several efficient algorithms exist for solving these problems.

Two particular types of frequent pattern mining problems that we will discuss here are *frequent itemset mining* and *frequent sequence mining*. Frequent itemset mining aims to find structure among data points that have no internal order, similar to most other data mining algorithms, whereas frequent sequence mining aims to find structure among data that has an inherent sequential (e.g. temporal) order. In the two sections below, we describe the problems, some main algorithms and applications for game data mining.

12.3.3.1 Frequent Itemset and Association Rule Mining

In frequent itemset mining, the base data takes the form of sets of instances (also called transactions) that each has a number of features (also called items). For example, a dataset of the items players bought in a social online game might contain five transactions as follows:

1. *{Sword of Grungni, Shirt of Awesomeness, Pretty Pet}*
2. *{Shirt of Awesomeness, Pretty Pet, Healing Potion}*
3. *{Sword of Grungni, Healing Potion}*
4. *{Shirt of Awesomeness, Sword of Grungni, Fancy Hat, Pretty Pet}*

The task for the frequent itemset mining algorithm is then to find all common sets of items, defined as those itemsets that have at least a minimum support (exists at least a minimum amount of times). If the support is set to 3, the following 1-itemsets (sets of only one item) can be found in the dataset described above: *{Sword of Grungni}, {Shirt of Awesomeness}* and *{Pretty Pet}*.

It is also possible to find one 2-itemset: *{Shirt of Awesomeness, Pretty Pet}*, as three of the transactions contain both *Shirt of Awesomeness* and *Pretty Pet.* Other itemsets of the same lengths are considered non-frequent as they recur less than three times. The original algorithm for mining frequent itemsets, which was published in 1993 and is still frequently used, is *Apriori* Agrawal et al. (1993). This algorithm functions by first scanning the database to find all frequent 1-itemsets, then proceeding to find all frequent 2-itemsets, then 3-itemsets etc. At each iteration, candidate itemsets of length n are generated by joining frequent itemsets of length $n - 1$; the frequency of each candidate itemset is evaluated before being added to the set of frequent itemsets. However, there exist several alternatives to this

algorithm. A prominent such alternative is the *FP-growth* algorithm, which finds frequent itemsets through building prefix trees Han et al. (2000).

Once a set of frequent itemsets has been found, association rules can be generated. Association rules are of the form A → B, and could be read as "A implies B". Each association rule has *support* (how common the precondition is in the dataset), *confidence* (how often the precondition leads to the consequence in the dataset) and *lift* (how much more common the consequence is in instances covered by the rule compared to the whole dataset). From the dataset and frequent itemsets above, the association rule *Shirt of Awesomeness* → *Pretty Pet* can be derived with support 3 and confidence 1, whereas the rule *Shirt of Awesomeness* → *Pretty Pet and Sword of Grungni* only has a confidence of only 1/3 and so would most likely not be selected as a useful association rule.

Frequent itemset mining can be used in several different ways for understanding game data. One way is to find patterns among players. If a database is organized so that each instance describes a separate player and the (binary or ordinal) attributes of each instance describe the player's playing style (e.g. {violent, speedrunner, cleared_level_3, dies_from_falling}), frequent itemset mining can be used to find playing style characteristics that frequently co-occur.

12.3.3.2 Frequent Sequence Mining

Unlike frequent itemset mining, frequent sequence mining cannot be applied to separate, unordered instances (such as where each instance represents a player). Instead, frequent sequence mining requires the instances to be ordered in one or several sequences. The probably most common type of sequence data is temporal sequence data, where each instance represents the state of the system at some time t; the interval between each instance might or might not be constant (in some terminology, an instance with all its features is called a symbol; identical instances map to the same symbol). The sequence mining problem is to, given a sequence or a set of sequences, find frequently occurring subsequences. For example, if the support threshold is set to 3, the sequence "abbabbcbdabb" has the frequent 3-sequence "abb" and the frequent 2-sequences "ab" and "bb".

One of the most commonly used frequent sequence mining algorithms is SPADE Zaki (2001). SPADE works in a similar way to Apriori: first find frequent sequences of length 1 (i.e. single symbols), then combine these frequent sequences into candidate sequences of length 2, evaluate their frequency, combine into sequences of length 3 etc.

By virtue of being discrete-time systems, computer games constantly generate large amounts of sequential data. At one extreme, you could consider the complete state of a game at every frame (where a modern game usually runs at 30 or 60 frames per second) as a data stream to be mined. Of course, this data stream would generate far too much data for any existing algorithm to handle, and all practical applications require that only a few interesting features are logged rather than the complete game state. Additionally, often the temporal resolution is decreased. Identifying which features are interesting to log depends on the purpose of the data mining, but they may

include any aspect of the game state which is directly or indirectly affected by the player's actions, such as button presses, player character position and actions, non-player character position and actions, changing level geometry etcetera.

Kastbjerg (2011) combined frequent sequence mining with clustering in order to visualize the spatial form of common sequences of player actions in the multiplayer game *Heroes of Newerth* (2010, S2 Games). This work was an attempt to improve on the "heatmaps" that are commonly used to analyze player's movements in game levels, but which do not convey information on what players did at any particular point in time (see Chap. 17 for more on heatmaps).

A large, publicly available database of *Heroes of Newerth* game replays was mined; a few hundred thousand player traces were used in initial experiments (39,390 games, roughly 59 million events across 20,000 h of play data, average playtime per game around 30 min). For each game the actions taken by each player was recorded, along with the (in-game) time and position of the action. The most frequent 3-sequences of actions were then found using SPADE (the process is shown in Fig. 12.10). Once a particular 3-sequence had been decided on, the starting points of that sequence are clustered (for very frequent sequences, more than a hundred thousand repetitions of that sequences could be found in the database for a particular map). The user can then select a particular starting point cluster, and from there investigate how players typically move as they perform the chosen action sequence starting from the chosen point. This analysis revealed for example that the initial phase of a particular attack spell was frequently used before a teleport spell, and then unleashed on another part of the map; the spatial analysis pointed out which particular areas of the map this sequence typically started from and ended in (Fig. 12.10).

Sequences are not necessarily temporal data. Shaker et al. (2011) used frequent sequence mining as a way to find features with which to classify levels of the platform game *Super Mario Bros* (2004, Nintendo). The task was to classify which levels would be preferred over others, based on the survey results from over 700 players who had played at least two levels each. Here, the sequences are not based on the players' actions over time, but simply on scanning the levels from left to right. In the version of *Super Mario Bros* that was used for the experiments, levels are linear (the level starts at the left end and is won by reaching the right end) and constructed of blocks. A level is about 15 blocks high and a couple of 100 blocks long. Each level was turned into a single sequence with the same length as the level (one symbol per block). A few different ways were investigated for transforming each vertical slice of the level into a symbol, for example by simply using the height of the level at that point, or by basing the symbol on the topmost block in that slice. In the next step, SPADE was applied to the sequences generated from levels, in order to find commonly occurring subsequences, i.e. commonly used level segments — these included flat parts without any gaps, lines of coins, short gaps surrounded by platforms etc. Each level could then be categorized according to the incidence of these segments, and the segment counts were successfully used to form features when using a supervised learning algorithm to predict whether a particular level was preferred over another.

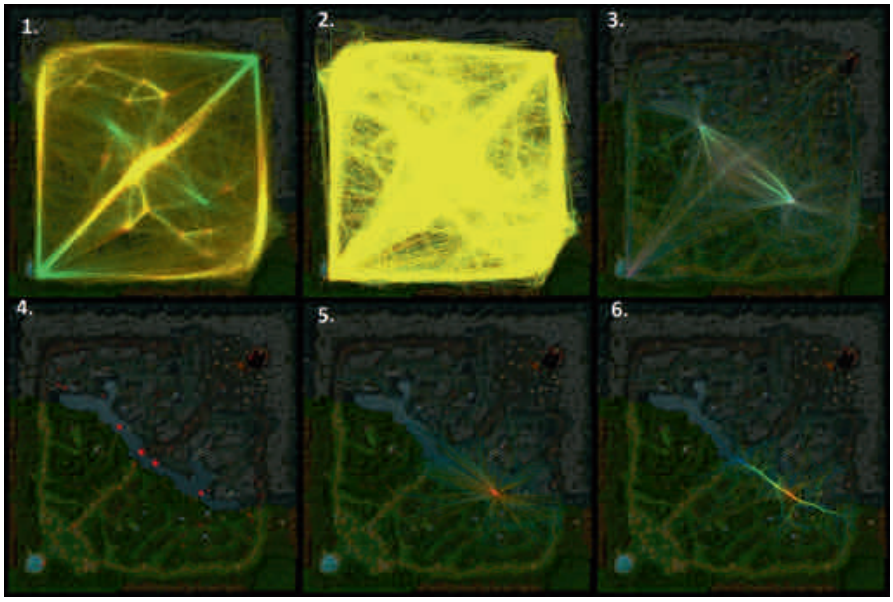

Fig. 12.10 Visualization of the sequence mining process applied to player traces in the *Heroes of Newerth* dataset. Image 1 show an intermediate step of the data loading and frequent sequence mining process SPADE. Each point on the map is colored in a heat map style, where the color of crossing edges is blended and increased in saturation. Image 2 shows the final result of the process. The result looks chaotic and is included to display how the data quickly becomes too excessive, thus the need for information extraction in the following steps: Image 3 shows a particular 4-sequence selected from the result pool of the previous sequence mining process. Image 4 shows the result of applying a modified version of FDEB on the start points of each instance in the sequence. In short the modified version omits the edge subdivision and intra attraction part (see Kastbjerg 2011, section 4.5 for an in-depth explanation). Image 5 shows the same as image 3, expect only sequences that start from a specific area, within a user defined radius. The particular area is selected by the user, but guide by the information found in step 4. Step 6 shows the final result, after the original FDEB algorithm has been applied to the edges selected in step 5 (Reproduced from Kastbjerg 2011 with permission)

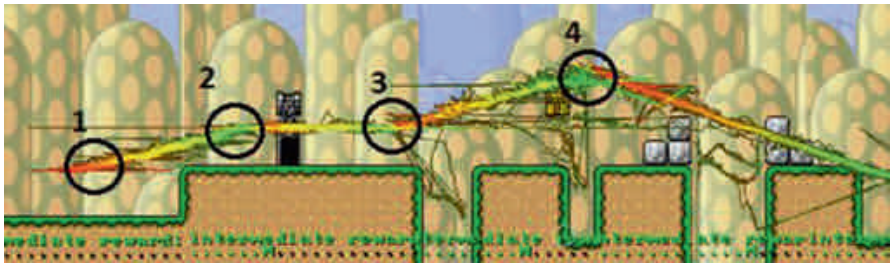

Fig. 12.11 A visualization of a frequent action sequence "jump-jump" from *Super Mario Bros*. Each *circle* marks a point where the player jumps (Reproduced from Kastbjerg 2011 with permission)

12.4 Supervised Learning

Supervised learning methods for data mining are drawn from Machine Learning (ML), a branch of artificial intelligence science that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on data, notably with the purpose of prediction. Machine learning – and thus supervised learning works from supervised training data. The training data, or signal, contains examples that the algorithms learn from, in order to be able to find the patterns or signals in data where the connection between input and output objects is unknown, e.g., which class to put a player in a given specific behavior. Supervised learning thus relies on a training dataset, or signal. The source of the signal defines the different clusters of all ML algorithms available. While in unsupervised learning the pattern (signal) is hidden in the internal structure of the data (interconnections among data attributes) and in reinforcement learning Sutton and Barto (1998) the training signal is derived as a reward from the learning environment, in supervised learning the signal is given in form of target data observations. Supervised learning is the process of training a function that approximates the mapping between attributes of the observations and the target observation. As a popular example for supervised learning, consider a machine being asked to distinguish between apples and pears (classes), given the color and size of the fruit (data attributes). Initially, the machine learner is trained on a number of attribute-class pair observations (i.e. training data providing the color and size of a number of apples and pears), from which it learns how to classify apples and pears. It can then subsequently be used to classify pears and apples based on the color and size input data only.

Popular supervised learning techniques include artificial neural networks, decision tree learning, support vector machines and bayesian learning (Bishop 2006). The primary use of supervised learning within games has been so far for the imitation of player behavior, the analysis of player behavior in online games, prediction of player behavior on massively multi-player online games, and notably for analysis of player behavior towards driving revenue in social online games (King and Chen 2009). For instance, the Drivatar system in *Forza Motorsport* (2005–2012, Microsoft Game Studios) is an artificial neural network that imitates the way a player drives a car and generates a race path that simulates the player's driving style. Similarly, the AI behind the player's deity avatar in *Black and White* (Electronic Arts 2003) uses supervised learning to imitate and respond to the player's actions and motivations.

A particular area of supervised learning in games is to imitate human playing behavior. Given a sufficiently large set of behavioral metrics data derived from players, supervised learning can be used for both imitating human playing behavior but also for predicting various aspects of the behavior. In AI research and industry, the main purpose of imitation is the creation of believable, human-like, non-player characters or similar computer-controlled entities, but there are also other purposes. Prediction can give answers to questions such as: "when will this player stop playing the game?" and: "how many times will this player use one weapon over another?" Essentially, supervised learning can be used for the prediction of any player attribute

given in the dataset. These kinds of questions are important to get answered during the development and testing of a game, and even after release. Supervised learning methods can thus be used to e.g. test and adjust designs, or even form the basis for systems controlling real-time adjustment of game features during play (e.g. dynamic difficulty adjustment).

12.4.1 Prediction Analysis and Decision Trees in Tomb Raider: Underworld

Prediction in data mining is performed with the goal of identifying a model of set of models that can be used to predict responses. For example, predicting which players will convert from non-paying to paying, or when particular players will stop playing a game. Notably in the context of social online games, prediction of player behavior, and design responses to changes in behavior, is important to ensure revenue. Prediction is similar to classification in that a model is constructed based on known data, and the model is used to predict unknown or missing values, e.g. future player behavior. The major method in prediction is regression, which generally attempts to construct either linear or non-linear models. Combining predictions from multiple models, which is particular useful when the types of models included in an investigation are very different, is referred to as "stacking" or "stacked generalization". Stacking is interesting because experiences have shown that predictions from multiple methods can yield more accurate predictions than any single method. When stacking, the predictions from different models are used as input to a meta-learner, which basically tries to combine the prediction models to create a "super-model" with the best predictions possible. The meta-learner can for example be neural network, which attempts to learn from the data how to combine the models for maximal accuracy in the predictions. Alternative approaches to combining prediction models are boosting and bagging (for further information see: Witten and Frank (2000), Han et al. (2005)). As an example of prediction based on game metrics (specifically gameplay metrics), we will use *Tomb Raider: Underworld* (Eidos Interactive, 2008) e. For a more in-depth description of the example, please see Mahlman et al. (2010).

Tomb Raider: Underworld consists of seven levels plus a prologue level. The goal of this analysis was to investigate if it was possible to develop a model that could predict when a player would stop playing the game, based on their early play behavior. This kind of prediction is useful to locate players who stop playing early in the game, and explore why this happens and how to modify the design to prevent players from leaving the game. For this experiment, the Weka (www.cs.waikato.ac.nz/ml/weka/) toolbox was used. Weka is a relatively easy-to-use toolset for data mining, and includes a wealth of prediction algorithms 76 just for classification of nominal attributes – and it is open-source.

The data for the analysis was drawn from the native metrics suite of Square Enix Europe, which contains data from approximately 1.5 million players of Tomb Raider: Underworld. From this population, a sub-sample of 10,000 players was

selected, randomly drawn from a larger sample of over 200,000 players, from which the metrics data was captured within the period of 1st December 2008–1st January 2009. The original sample of 10,000 players was cleaned thoroughly, removing instances where the metrics suite had missing data reported for a player. Because the aim of the analysis was to predict when players stop playing the game, only players who had completed level one were included. After cleaning the 10,000 player sample, 6,430 players remained. The data from level 1 were used as the training (learning) dataset. All features were normalized to a 0–1 scale via a uniform distribution to minimize the effect of outliers.

The input features (variables) were selected from the core mechanics of the game, a strategy which helps with ensuring that the features are relevant to player behavior analysis. Each feature was measured either per map unit or per level, giving a total feature set of over 400 variables (number of features * level/map unit):

- **Playing time:** the time that each player spent playing the game. This includes a number of features, notably the playing time spent for each sub-segment of each level in the game (there are over 70 such segments).
- **Total number of deaths:** how many times the player died.
- **Help-on-Demand:** how many times the player requested help from the native Help-on-Demand system in the game, which assists players with their progress in the game.
- **Causes of death:** the game features various ways in which a player can die. These were classified into four groups: Death by melee enemies, death by ranged enemies, death by environmental causes, and death by falling (by far the most common cause of death in Tomb Raider: Underworld – 62.92%).
- **Adrenalin:** the number of time the adrenalin feature was used. Using adrenalin allows the player to temporarily slow down time while performing special attacks.
- **Rewards:** the number of rewards collected (the average is 112.08). Treasure: The number of treasures found. Each level has one or a few of these major finds, which take particular exploration to locate.
- **Setting changes:** players can change various parameters of the game, and four of these impact directly on gameplay and were therefore included: Ammo adjustment, enemy hit points, player hit points, and saving grab adjustment (which adjusts the time a player has to secure a handhold after a jump).

From the dataset of 6,430 players (including all the variables mentioned above), who completed level 1 at the least, a smaller dataset was extracted which consisted of the 3,517 players who also completed at least level 2. A third set of data was created from the second, containing the 1,732 players that finished the entire game. The three datasets were used to try to predict the time taken to play through the game, with the underlying assumption that there is an (unknown) relationship or function between early playing behavior (levels 1 and 2) and the speed with which a game is completed, or conversely when a player will stop playing (i.e. last level played), which a classification algorithm will be able to predict.

Several classification algorithms were used on the two problems: completion time and last level played. The best results were found using logistic regression, a relatively simple algorithm which could predict when a player would stop playing *Tomb Raider: Underworld*, with a success rate of 77.3%. Several algorithms performed well on the dataset (notably SMO support vector machine, MLP/ Backpropagation), with a much better accuracy than the baseline of 39.8% (the baseline is the optimal predictor in case there is no data available, equal to the number of samples in the most common class (level completed) divided by the total number of classes). The accuracy of the prediction is in this case decent. Typical eprediction models were built on high-dimensionality gameplay metrics dataset (in this case hundreds of features), presumably due to either the high degree of variance in the datasets, i.e. in how people play games, and data losses during collection of telemetry data from game clients (Drachen et al. 2009).

The ability to predict when a player will stop playing a game, or for how long the game will be played, based on their early behavior is useful in user-oriented testing, where it is possible to use this information to locate the kinds of behaviors that lead players to quit playing. This is particularly useful in certain forms of social online games, where player retention (the ability of the game to keep people playing it) is central to the revenue stream (see Chap. 4).

12.4.2 Decision Trees

Results from prediction analysis need to be explained in a way that makes them understandable to the target user, e.g., a game designer. Apart from accuracy in the predictions, an advantage of some of the algorithms for predictive data mining is that they provide relatively transparent models, which means that changes to design elements can be easily understood.

Decision trees are a good example of this. They use a graphic approach to compare competing alternatives, and assign values to these alternatives, describing sequential decision problems. They provide a complementary approach to traditional statistical forms of analysis such as multiple linear regression and data mining approaches such as neural networks. They are relatively powerful analytically, easy to use, easy to interpret and robust within a range of data and levels of measurement. They are presented incrementally, in a collection of one-cause, one-effect relationships in the recursive form of a tree, which means they are easy to understand than more complex multiple variable techniques (Rokach and Maimon 2008).

Like other methods of multiple variable analyses, they allow the prediction, explanation, description or classification or an outcome. For example, a multiple variable analysis could be the probability that a player will convert from non-paying to paying as a result of the combined effect of multiple variables, e.g. a marketing campaign, being given a valuable in-game item, the size of their social network in

the game – or being given a free T-shirt if they sign up for a subscription. In essence, decision trees allow analysts to follow the effect of different decisions, and plan the optimal strategy for causing specific decisions (or situations) to occur in the games in question. For example, answering questions such as: which set of methods for encouraging player to become paying users work the best?

Decision trees are produced by algorithms, which try to split a dataset into branch-like segments – hence the name. The branches form inverted decision trees that originate with a root node at the top, and branch out from there. Decision trees attempt to find relationships between input values and target values in a dataset (group of observations). When an input value is identified as being strongly related to a target value, all of these variables are grouped in a bin that becomes a branch in the tree. A strong relationship is formed when the value of an input variable can be used to predict the value of the target. For example, if the amount of time a player spends in a particular map unit of *Tomb Raider: Underworld*, is a strong predictor of the completion time of the entire level. Or the number of times a player activates the adrenalin feature of the game (an advanced game mechanic) could be a predictor of whether the player is an experienced player or not.

To take a hypothetical example from *Tomb Raider: Underworld* (see Mahlman et al. 2010 for a more in-depth example), where decision tree analysis is employed to predict which level players will stop playing at, as a feature of playtime and rewards, the resulting tree could look like this:

```
Level-2 rewards
Rewards > 10
Level-3 playtime
     Ðplaytime > 43 minutes : 4
     Ðplaytime < 43 minutes : 7
Rewards < 10 : 2
```

The right arrow (–>) indicates a branch under the tree-node, which is directly above the symbol. The number to the right of the colon represents the predicted game level where the player will stop playing. What the tree means is that a strong relationship has been found between the level at which players stop playing, and the rewards earned at level 2, and the playtime at level 3. The first branch informs that if the player earns less than 10 rewards, they will stop playing at level 2. The second branch informs that if the players spend more than 43 min on level 3, they will stop playing on level 4, but if they complete below this time, they will play through the entire game.

Decision trees like this one can be employed on virtually any kind of variable tracked via telemetry, i.e. behavioral variables, in order to find out which features are the most important to determine when a player quits playing (or any other outcome being measured, e.g. how much money they spend on microtransactions) (Chap. 4), and the values of these features to prompt different end-points in the tree.

12.5 Game Data Mining in Free-to-Play Games

In this section we take a specific look at online games. These games are of particular interest in game data mining because they are highly dependent on understanding player behavior, and currently one of the major forces driving the use of and innovation in data mining in game development. It is not in the goal here to provide a comprehensive introduction to data mining in online games, but a brief overview. The reader is referred to the references for additional information.

To start with a brief (and generalized) historical perspective, the current push for data mining player behavior in the industry has to a certain extent been driven by the rise of the social online game – or Free-to-Play (F2P) – genre, as well as the widespread popularity of the Massively Multiplayer Online Game (MMOG) genre. MMOGs have an almost two decades long history, reaching back to games like *Meridian 59* and *EverQuest*. They started getting serious attention in the regular press with the realization that these online, persistent worlds contained intricate economies (Castranova 2001), and with *Second Life* (arguably a virtual world, not a game) and notably *World of Warcraft*, that they had become highly popular. With the evolution of Web 2.0 technologies, notably social networking platforms like Facebook, another type of game also increased in popularity: F2P, with early examples on Facebook including *Mafia Wars*.

MMOGs and F2P games were different from previous game forms in that they catered to very large groups of players who could interact in real time. MMOGs and many F2Ps are also persistent world games – they are always running – which facilitated the emergence of social communities in these games.

In the past few years metrics-driven development has almost become standard in online games development and –management. Acronyms and terms like ARPU, NOSQL and Big Data are becoming commonplace (see Chap. 4), and it is likely that most publishers and developers in the online games sphere are highly dependent on analytics and reports to keep their businesses running. While many Key Performance Indicators (KPIs) are common (Fig. 12.12), the level of sophistication in the analytics software and processes vary across the industry (Flood 2012). Competition, the cross-over of players between different sectors of the games industry, and the evolution in player communities over time, requires online games companies to field efficient data capture and storage, and the ability to generate KPIs and ad-hoc analysis and reporting.

12.5.1 Metrics-Driven Business Practices in Online Games

A lot more could be said about the historic background for MMOGs and F2Ps (for more information see Fields and Cotton 2011), but the essence of the matter is that these games need data mining because they have to *manage and monetize on a community of players*. Generalizing, the essential requirement in the MMOG business model is to keep people engaged so they continue to pay subscription fees.

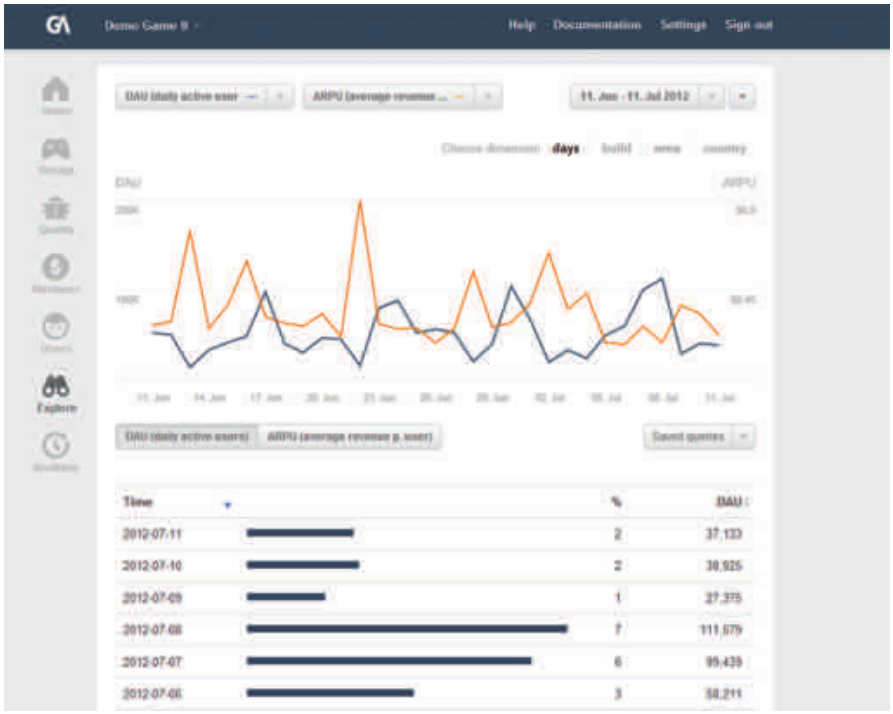

Fig. 12.12 Screenshot from the an early beta version of the analytics tool from Game Analytics, showing the development over time of two of the common Key Performance Indicators (KPIs) for online games: Daily Active Users (DAU) and Average Revenue Per User (ARPU) (© Game Analytics, used with permission, www.gameanalytics.com)

The requirement for F2P games is to convince players to spend money on buying in-game resources.

There are a number of ways to handle this kind of challenge, but fundamentally relate to Business Intelligence management. There are a number of similarities between managing and monetizing on player communities and the management of websites, online forums and web-based communities in general. These, similar to online games, have customers coming and going, interacting with the site and/or people via the site, for shorter or longer periods of time.

Web analytics is the field of research and practice dealing with quantitative analysis of user behavior on the Net (Jansen 2009), and back when MMOG and F2P models were gaining momentum, there was a lot of knowledge available that could be adapted for use in these – and other – types of games, for example with regards to online advertising and customer retention, and the use of techniques like funnel analysis and cohort analysis to understand the cost of acquisition, retention factors, revenue generation, social factors, etc.

The metrics-driven business practice in online games gained strong traction with the rapid growth of game companies like *Zynga, BigFish* and *Wooga*, who had

adopted a metrics-driven development practice and became highly successful in a short period of time, and the growth of the social application market in general (e.g. Facebook, InstaGram, LinkedIn, Google+, Twitter, Picasa …). Business intelligence has emerged as a key aspect of operating a successful online games company.

12.5.2 Data Mining Telemetry from Online Games

Whereas the publicly available knowledge about data mining in MMOGs is limited due to confidentiality issues, the available knowledge for F2P games is more comprehensive if somewhat fragmented, but generally originates in articles, blog posts or reports from the industry, and is therefore not falsifiable. With that in mind, the data mining techniques for analyzing player telemetry from online games can be broadly divided into three broad categories:

1. **Key Performance Indicators:** These are metrics like Daily Active Users (DAU) and Churn rate, which are generated using descriptive methods, e.g. aggregates or ratios, typically calculated as a function of time, game build or geographical area (Chap. 4; Fields and Cotton 2011).
2. **Adopted techniques:** These are techniques adopted – and sometimes subsequently adapted – from other areas where Business intelligence is applied, notably web analytics. Examples include acquisition analysis, funnel analysis, A/B testing and cohort analysis (Chap. 4; Fields and Cotton 2011). These methods are generally descriptive or examples of characterization and discrimination.
3. **Advanced techniques:** These are techniques that rely on data mining methods for clustering, classification, prediction, estimation and association. Notably user behavior prediction (Weber et al. 2011; Nozhnin 2012; Bauckhage et al. 2012; Lim 2012), classification of user behavior (Drachen et al. 2009, 2012) and retention modeling (Fields and Cotton 2011) has received interest in the online games sector, as these techniques are of key interest in driving revenue.

12.6 Discussion and Next Steps

In this chapter, we have presented an introduction to data mining and its particular application in game development, *game data mining*. A number of important issues in relation to working with game telemetry datasets have been discussed, covering topics such as methods, stakeholders and practice. Additionally, we have outlined a number of examples showing how to perform different types of supervised and unsupervised analysis on game telemetry data.

While the focus of the chapter is on game telemetry data, and the types of problems they can be applied to solve, the general principles and the methods presented are not unique, but rather common in data mining across several application areas, and therefore accessible in a wealth of literature to anyone interested in learning more about data mining (e.g. Han et al. 2005). For more on game data mining,

Chap. 4 outlines KPIs for online games; Chap. 7 describes developer-facing analytics, Chap. 17 discusses game data mining in the specific context of spatial data, i.e. data with a spatial component (e.g. data on player movement in a 3D environment). Chapter 18 and 19 go into more depth with visualization of game telemetry data.

Game telemetry presents some challenges that are uncommon or maybe even unique in large-scale user-oriented datasets:

1. The data can have a high dimensionality, often with thousands of features (or variables) being tracked for each user.
2. The data can be of substantial size, an average MMOG or social online game generating datasets on the terabyte scale.
3. It is often necessary to compile datasets for analysis from disparate sources, e.g., game telemetry and account systems, with associated challenges in merging data and avoiding redundancies.
4. Obtaining game telemetry from remote clients, across multiple hardware platforms (many games are released on multiple hardware platforms), requires well-designed back-end systems to ensure that the datasets are as complete as possible. This is in particular a challenge when collecting data from devices that are not online all the time while the user is playing, e.g. games for smartphones.

The list goes on, and it is out of scope here to provide a full discussion of all of the issues related to game data mining. However, we provide a starting point for non-experts, and hopefully some case studies that will also satisfy the game data mining expert as well.

It can seem like a daunting project to develop both the technical back-end for collecting and storing game telemetry, as well as learning how to pre-process and subsequently analyze the datasets, and finally finding the best ways to present the results to various stakeholders in development companies. The cost alone can seem prohibitive, but the simple fact that publishers like Microsoft, Square Enix, EA Games and Ubisoft are employing game telemetry, the success stories of companies like Zynga, Wooga, Bigfish and Bungie, are strong indications of the benefits that can be obtained via game data mining.

To make game data mining easier, there is these years a proliferation of tech startups seeking to develop middleware tools that enable even small developers to work with telemetry data (e.g. Game Analytics, Honeytracks, Playtomic, Playnomics, Tableau, Kontagent), as well as a number of open-source tools for different engines (e.g. Unity) and analytics packages that can be used for analyzing game telemetry (e.g. statistics packages like SPSS, open-source data mining tools like WEKA and RapidMiner). While the main focus has hitherto been on analyses of player behavior (e.g. Zoeller 2010) (see also Chaps. 4, 7, 14, 17, 18, and 19) and customer data (e.g. King and Chen 2009) (see Chap. 4), the potential scope of application of game data mining as a source of business intelligence is substantial, crossing marketing, production (Mellon 2009) (see Chaps. 6 and 7), design, user testing, strategic decision making, etc., and if the current rapid development in the application of various types of measures to guide game development including game telemetry is any indication, the application of game metrics in the digital

entertainment industry will become a standard that is as normal as other types of processes in business, e.g. benchmarking and usability testing – if you need more evidence, read any other chapter in this book.

About the Authors

Anders Drachen, Ph.D. is a veteran Data Scientist, currently operating as Lead Game Analyst for Game Analytics (www.gameanalytics.com). He is also affiliated with the PLAIT Lab at Northeastern University (USA) and Aalborg University (Denmark) as an Associate Professor, and sometimes takes on independent consulting jobs. His work in the game industry as well as in data and game science is focused on game analytics, business intelligence for games, game data mining, game user experience, industry economics, business development and game user research. His research and professional work is carried out in collaboration with companies spanning the industry, from big publishers to indies. He writes about analytics for game development on *blog.gameanalytics.com*, and about game- and data science in general on *www.andersdrachen.wordpress.com*. His writings can also be found on the pages of Game Developer Magazine and Gamasutra.com.

Christian Thurau, Ph.D. is CTO of Game Analytics (www.gameanalytics.com) and a former researcher at the Fraunhofer IAIS in St. Augustin and at the Bonn-Aachen International Center for Information Technology B-IT. He works with developing game telemetry systems and the application of advanced data mining methods in games contexts, for example player modeling, behavior cloning and data analysis in the massive dataset size range. His fields of research include Pattern Recognition, Computer Vision, Data Mining, and Machine Learning.

Julian Togelius, Ph.D. is Associate Professor at the Center for Computer Games Research, IT University of Copenhagen, Denmark. He works on all aspects of computational intelligence and games, on geometric generalization of stochastic search algorithms and on evolutionary reinforcement learning. His current main research directions involve search-based procedural content generation in games, automatic game design, and fair and relevant benchmarking of game AI through competitions. He is also the chair of the IEEE CIS Technical Committee on Games, and an associate editor of IEEE Transactions on Computational Intelligence and Games. He holds a BA from Lund University, an MSc from the University of Sussex, and a PhD from the University of Essex.

Georgios N. Yannakakis, Ph.D. is an Associate Professor at the IT University of Copenhagen. He received the Ph.D. degree in Informatics from the University of Edinburgh in 2005. Prior to joining the Center for Computer Games Research, ITU, in 2007, he was a postdoctoral researcher at the Mærsk Mc-Kinney Møller Institute, University of Southern Denmark. His research interests include user modeling, neuro-evolution, computational intelligence in computer games, machine

learning, cognitive modeling and affective computing. He has published over 100 journal and international conference papers in the aforementioned fields. He is an Associate Editor of the IEEE Transactions on Affective Computing and the IEEE Transactions on Computational Intelligence and AI in Games, and the chair of the IEEE CIS Task Force on Player Satisfaction Modeling.

Christian Bauckhage, Ph.D. is professor of media informatics at the University of Bonn and lead scientists for multimedia pattern recognition Fraunhofer IAIS. He obtained a PhD in computer Science from Bielefeld University, Germany, and worked in academic and industrial research labs. He is an expert in large scale data mining and pattern recognition and researches computational approaches to artificial cognition and behavior.

References

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM-SIGMOD international conference on management of data (SIGMOD)* (pp. 207–216). Washington, DC.

Bauckhage, C., Kerstin, C., Sifa, R., Thurau, C., Drachen, A., & Canossa, A. (2012). How players lose interest in playing a game: An empirical study based on distributions of total playing times. In *Proceedings of IEEE computational intelligence in games,* Granada, Spain.

Berry, M., & Linoff, G. (1999). *Mastering data mining: The art and science of customer relationship management*. New York: Wiley.

Bishop, C. M. (2006). *Pattern recognition and machine learning* (Information science and statistics). New York: Springer.

Bohannon, J. (2010). Game-miners grapple with massive data. *Science, 330*(6000), 30–31.

Castranova, E. (2001). *Virtual worlds: A first-hand account of market and society on the Cyberian frontier* (CESifo Working Paper Series no 618). München.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinart, T., Shearer, C., & Wirth, R. (2000). Crispdm step-by-step data mining guide. http://www.crisp-dm.org/

Charles, D., & Black, M. (2004, November 8–10). Dynamic player modelling: A framework for playercentric digital games. In *Proceedings of CGAIDE 2004, 5th international conference on computer games: Artificial intelligence, design and education*. Microsoft Campus, Reading, UK. ISBN 09549016-0-6

Chen, M. S., Han, J., & Yu, P. S. (1996). Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering, 8*, 866–883.

Coulton, P., Bamford, W., Cheverst, K., & Rashid, O. (2008). 3D space-time visualization of player behavior in pervasive location-based games. *International Journal of Computer Games Technology Volume 2008 (2008), Article ID 192153, 5 pages*. doi:10.1155/2008/192153

Cutler, A., & Breiman, L. (1994). Archetypal analysis. *Technometrics, 36*(4), 338–347.

DeRosa, P. (2007, August 7). Tracking player feedback to improve game design. Gamasutra. Available from: http://www.gamasutra.com/view/feature/1546/tracking_player_feedback_to_.php

Drachen, A., & Canossa, A. (2009). Towards gameplay analysis via gameplay metrics. In *Proceedings of the 13th international MindTrek conference.* Tampere: ACM.

Drachen, A., & Canossa, A. (2011). Evaluating motion: Spatial user behavior in virtual environments. *International Journal of Arts and Technology, 4*, 294–314.

Drachen, A., Canossa, A., & Yannakakis, G. N. (2009). Player modeling using self- organization in Tomb Raider: Underworld. In *Proceedings of the international symposium on Computational Intelligence and Games, CIG'09*, Piscataway.

Drachen, A., Sifa, R., Bauckhage, C., & Thurau, C. (2012). Guns, swords and data: Clustering of player behavior in computer games in the wild. In *Proceedings of IEEE computational intelligence in games*, Granada, Spain.

Ducheneaut, N., & Moore, R. J. (2004). The social side of gaming: A study of interaction patterns in a massively multiplayer online game. In *Proceedings of the 2004 ACM conference on computer supported cooperative work*, Chicago.

Erfani Joorabchi, M., Seif El-Nasr, M. (2011, October, 5–8). Measuring the impact of knowledge gained from playing FPS and RPG games on gameplay performance. In *Proceedings of 10th international conference, ICEC 2011* (Lecture notes in computer science, Vol. 6972, pp. 300–306). Vancouver.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Menlo Park: AAAI Press.

Fields, T., & Cotton, B. (2011). *Social game design: Monetization methods and mechanics*. Waltham: Morgan Kauffman Publishers.

Finesso, L., & Spreij, P. (2004). Approximate nonnegative matrix factorization via alternating minimization. In *Proceedings 16th international symposium on mathematical theory of networks and systems*, Leuven.

Flood, K. (2012, March 27). Game analytics (series). Kevin's corner. URL: file:///G:/Work/METRICS/Metrics_references/Kevin%27s%20Corner%20%20Game%20Analytics.htm

Gagne, A., Seif El-Nasr, M., & Shaw, C. (2012). Analysis of telemetry data from a real time strategy game: A case study. Computers in Entertainment (CIE) - *Theoretical and Practical Computer Applications in Entertainment, 10*(3), Article No. 2. New York: ACM. doi:10.1145/2381876.2381878.

Geng, L., & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys, 38(3),* Article No. 9. New York: ACM. doi:10.1145/1132960.1132963.

Golub, G., & van Loan, J. (1996). *Matrix computations* (3rd ed.). Baltimore: Johns Hopkins University Press.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM-SIGMOD international conference on management of data (SIGMOD)* (pp. 1–12). New York.

Han, J., Kamber, M., & Pei, J. (2005). *Data mining: Concepts and techniques* (Morgan Kaufmann large-scale data mining in games 41 2nd ed.). San Francisco: Morgan Kaufmann Publishers.

Hoobler, N., Humphreys, G., & Agrawala, M. (2004). Visualizing competitive behaviors in multi-user virtual environments. In *Proceedings of the conference on visualization.* Los Alamitos: IEEE.

Houlette, R. (2004). Player modeling for adaptive games. In S. Rabin (Ed.), *AI game programming wisdom II* (pp. 557–566). Hingham: Charles River Media.

Isbister, K., & Schaffer, N. (2008). *Game usability: Advancing the player experience*. San Francisco: Morgan Kaufman.

Jansen, B. J. (2009). *Understanding user-web interactions via web analytics*. San Rafael: Morgan & Claypool Publishers.

Jolliffe, I. (1986). *Principal component analysis*. New York: Springer.

Kastbjerg, E. (2011). *Combining sequence mining and heatmaps to visualize game event flows (working title)*. Master's thesis, IT University of Copenhagen, Copenhagen.

Kennerly, D. (2003, August 15). Better game design through data mining. Gamasutra. Available from: http://www.gamasutra.com/view/feature/2816/better_game_design_through_data_.php

Kim, J. H, Gunn, D. V, Phillips, B. C, Pagulayan, R. J, & Wixon, D. (2008). Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems. In *Proceedings of the twenty-sixth annual SIGCHI conference on human factors in computing systems, CHI'08*, Florence.

King, D., & Chen, S. (2009). Metrics for social games. *Presentation at the social games summit 2009, game developers conference*. San Francisco, CA.

Larose, D. T. (2004). *Discovering knowledge in data: An introduction to data mining*. Hoboken: Wiley.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature, 401*(6755), 788–799.

Lim, N. (2012, June 26). Freemium games are not normal. Gamasutra. URL: http://www.gamasutra.com/blogs/NickLim/20120626/173051/Freemium_games_are_not_normal.php?goback=.gmr_4199042.gde_4199042_member_130240768.gmr_4199042.gde_4199042_member_128990050#comments

Mahlman, T., Drachen, A., Canossa, A., Togelius, J., & Yannakakis, G. (2010). Predicting player behavior in Tomb Raider: Underworld. In *Proceedings of the international conference on Computational Intelligence and Games, CIG'10*, Copenhagen.

Mellon, L. (2009). *Applying metrics driven development to MMO costs and risks*. White paper, Versant Corporation.

Missura, O., & Gärtner, T (2009). Player modeling for intelligent difficulty adjustment. In *Proceedings of the 12th international conference on discovery science, DC'09*, Berlin.

Moura, D., Seif El-Nasr, M., & Shaw, C. D. (2011). Visualizing and understanding players' behavior in video games: Discovering patterns and supporting aggregation and comparison. In *Proceedings of the 2011 ACM SIGGRAPH symposium on video games (Sandbox '11)* (pp. 11–15). New York. ISBN:978-1-4503-0775-8, doi:10.1145/2018556.2018559.

Nozhnin, D. (2012, May 17). Predicting churn: Data-mining your game. Gamasutra. URL: http://www.gamasutra.com/view/feature/170472/predicting_churn_datamining_your_.php

Paatero, P., & Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics, 5*(2), 111–126.

Pedersen, C., Togelius, J., & Yannakakis, G. N. (2010). Modeling player experience for content creation. *Transactions on Computational Intelligence and AI in Games, 2*, 54–67.

Rokach, L., & Maimon, O. (2008). *Data mining with decision trees: Theory and applications*. New Jersey: World Scientific Publishing.

Seif El-Nasr, M., & Zammitto, V. (2010). User experience research for sports games. *Presentation at the GDC summit on games user research*. San Francisco, CA.

Seif El-Nasr, M., Aghabeigi, B., Milam, D., Erfani, M., Lameman, B., Maygoli, H., & Mah, S. (2010). Understanding and evaluating cooperative games. *CHI 2010* (pp. 253–262). New York.

Shaker, N., Yannakakis, G., & Togelius, J. (2011). Feature analysis for modeling game content quality. In *Proceedings of the 2011 IEEE conference on computational intelligence and games* (pp. 126–133). Seoul, Korea

Summit Kohonen, T. (2001). *Self-organizing maps*. Heidelberg: Springer.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Adaptive computation and machine learning). Cambridge: The MIT Press.

Thawonmas, R., & Iizuka, K. (2008). Visualization of online-game players based on their action behaviors. *International Journal of Computer Games Technology, 2008*, 1–9.

Thawonmas, R., Kashifuji, Y., & Chen, K. T. (2008, December 3–5). Design of MMORPG Bots based on behavior analysis. In *Proceedings of the 2008 international conference on advances in computer entertainment technology, ACE'08*, Yokohama, Japan (ACM International Conference Proceeding Series 352, pp. 91–94). doi:10.1145/1501750.1501770, ISBN:978-1-60558-393-8.

Thompson, C. (2007). Halo 3: How Microsoft labs invented a new science of play. *Wired Magazine.*

Thurau, C., & Bauckhage, C. (2010). Analyzing the evolution of social groups in world of warcraft. In *Proceedings of the international conference on Computational Intelligence and Games, IEEE, CIG'10*, Copenhagen.

Thurau, C., & Drachen, A. (2011). Introducing archetypal analysis for player classification in games. In *Proceedings of the international workshop on evaluating player experience in games (EPEX'11) hosted at the 6th international conference on the foundations of digital games (FDG2011)*. Bordeaux.

Thurau, C., Bauckhage, C., & Sagerer, G. (2004, July 13–17). Learning human-like movement behavior for computer games. In *Proceedings of the 8th international conference on the Simulation of Adaptive Behavior, SAB'04*. Los Angeles, USA. ISBN: 9780262693417.

Thurau, C., Paczian, T., Sagerer, G., & Bauckhage, C. (2007). Bayesian imitation learning in game characters. *International Journal of Intelligent Systems Technologies and Applications, 2*(2–3), 284–295.

Thurau, C., Kersting, K., & Bauckhage, C. (2009). Convex non-negative matrix factorization in the wild. In *Proceedings of the IEEE international conference on data mining*, Miami.

Thurau, C., Kersting, K., & Bauckhage, C. (2010). Yes we can – Simplex volume maximization for descriptive web–scale matrix factorization. In *Proceedings of the international Conference on Information and Knowledge Management, ACM, CIKM'10*, Toronto.

Thurau, C., Kersting, K., Wahabzada, M., & Bauckhage, C. (2011). Descriptive matrix factorization for sustainability: Adopting the principle of opposites. *Journal of Data Mining and Knowledge Discovery, 24*, 325–354.

Weber, B., & Mateas, M. (2009). A data mining approach to strategy prediction. In *Proceedings of the international symposium on Computational Intelligence and Games, CIG'09*, Piscataway.

Weber, B. G. John, M. Mateas, M. & Jhala, A. (2011). Modeling player retention in Madden NFL 11. In *Proceedings of the association for the advancement of artificial intelligence conference*, San Francisco.

Williams, D., Yee, N., & Caplan, S. E. (2008). Who plays, how much, and why? Debunking the stereotypical gamer profile. *Journal of Computer-Mediated Communication, 13*, 993–1018.

Williams, D., Consalvo, M., Caplan, S., & Yee, N. (2009). Looking for gender (LFG): Gender roles and behaviors among online gamers. *Journal of Communication, 59*, 700–725.

Witten, I. H., & Frank, E. (2000). *Data mining*. New York: Morgan-Kaufmann.

Yannakakis, G. A. (2012). Game AI revisited. In *Proceedings of the conference on computing frontiers*, Caligari.

Yannakakis, G. N., & Hallam, J. (2009). Real-time game adaptation for optimizing player satisfaction. *Transactions on Computational Intelligence and AI in Games, 1*, 121–133.

Yannakakis, G. A., & Togelius, J. Experience-driven procedural content generation. IEEE Transactions on Affective Computing, 2 (3), 147–161

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning, 42*, 31–60.

Zoeller, G. (2010). Game development telemetry. *Presentation at the game developers conference 2010.*