Clustering Game Behavior Data

Christian Bauckhage, *Member, IEEE,* Anders Drachen, *Member, IEEE,* and Rafet Sifa

Abstract

Recent years have seen a deluge of behavioral data from players hitting the game industry. Reasons for this data surge are many and include the introduction of new business models, technical innovations, the popularity of online games, and the increasing persistence of games. Irrespective of the causes, the proliferation of behavioral data poses the problem of how to derive insights therefrom. Behavioral data sets can be large, time-dependent and high-dimensional. Clustering offers a way to explore such data and to discover patterns that can reduce the overall complexity of the data. Clustering and other techniques for player profiling and play style analysis have therefore become popular in the nascent field of game analytics. However, the proper use of clustering techniques requires expertise and an understanding of games is essential to evaluate results. With this paper, we address game data scientists and present a review and tutorial focusing on the application of clustering techniques to mine behavioral game data. Several algorithms are reviewed and examples of their application shown. Key topics such as feature normalization are discussed and open problems in the context of game analytics are pointed out.

Index Terms

Game Analytics; Behavior Mining; Clustering

I. Introduction

The game industry is facing a surge of data which results from increasingly available highly detailed information about the behavior of software and users [1]–[6]. Exploding amounts of behavioral data result from growing customer bases for digital games, new business models, better persistence of games, improving possibilities of collecting contextual data around user behavior, and modern techniques for recording in-game data [3]. Regardless of what causes the data deluge, game industry and game researchers alike face a fundamental problem: how to derive actionable insights from large amounts of high-dimensional and time-dependent data that people generate while playing games?

This is one of the key question in the emerging area of *game analytics* which analyzes how games are played. It aims at a better understanding of player behavior in order to help improving a game's design, ensure optimal user experience, identify valuable players or those at risk of leaving a game, personalize or adapt gameplay, or assist matchmaking.

C. Bauckhage and R. Sifa are with Fraunhofer IAIS, 53754 St. Augustin, Germany; e-mails: {christian.bauckhage, rafet.sifa}@iais.fraunhofer.de

A. Drachen is with Aalborg University, 2450 Copenhagen, Denmark; email: drachen@hum.aau.dk

Manuscript received November 15, 2013; revised August 10, 2014.

However, analyzing behavioral data from games can be challenging. Consider, for example, Massively Multi-Player Online Games such as *World of Warcraft, Tera*, or *Eve Online*. Each of these games features up to hundreds of thousands of simultaneously active users spread across hundreds of instances of the same virtual environment [7]. Each player controls one or more characters with up to hundreds of abilities that evolve over time. While playing, users can perform dozens of actions per minute which lead to hundreds of state updates by the game. Add log data recorded by game servers, purchasing histories and other monetization data, information about gaming hardware, and the result easily exceeds thousands of features per player. When players are active for months or years, this results in longitudinal datasets [8] which may be further enhanced through contextual information such as demographic data for marketing or customer profiling [4]–[6].

Game behavioral data are therefore often subject to the *curse of dimensionality* [9], i.e. the tendency of high dimensional data to behave counter-intuitive [10]–[12]. Finding patterns under such conditions can be difficult but is rewarding because patterns detected in behavioral data can inform the game development process [13]–[16]. In practice, this kind of analysis typically still operates on highly granular levels. For example, 3D heatmaps that visualize event frequencies are used to investigate if a specific area within a game world sees too high or too low a concentration of the feature being investigated (e.g. death events as in Fig. 1). Yet, although they are intuitive and easy to compute, heatmaps alone hardly reveal why a specific concentration occurs and approaches are called for that can uncover correlations within the given data.

One way of dealing with massive, high dimensional game data is *clustering*. As an unsupervised method, it permits the exploration of data and can identify groups of players of similar behaviors or detect the features that constitute such behaviors. Cluster analysis is widely applied across disciplines and has been readily adopted in game analytics to find patterns in player or customer behavior [16]–[20], but also to compare and benchmark games and to improve game AI [21]–[26].

However, the correct use of clustering algorithms requires expertise and an understanding of the application context. We stress this, because toolboxes for scientific computing have made it easy to run cluster algorithms even if one is unaware of the underlying principles. Yet, without a proper understanding of the assumptions algorithms operate on, conclusions derived from such an analysis can be misleading. Additionally, without knowledge of the game under examination and its mechanics, choices made during analysis —ranging from feature selection and data pre-processing to visualization and interpretation— run the risk of leading to flawed or useless results.

In this paper, we provide a review and tutorial on the use of clustering techniques in mining behavioral data from games. We introduce the basic theory of clustering, underlying assumptions, and common pitfalls and discusses three clustering techniques in detail: the fundamental k-means algorithm, matrix factorization methods, and spectral approaches. Game related application examples of these algorithms are provided with references to related work and open problems in this domain are pointed out.

II. Clustering Player Data

While an exhaustive treatment of cluster analysis is beyond our scope, we still review several fundamental concepts, since our goal is to provide a tutorial on cluster analysis for game behavioral data. For even deeper expositions, we refer to the vast literature; a recent overview is provided in [27].

A. Foundations of Cluster Analysis

Cluster analysis is to group a set of objects such that similar ones are assigned to the same *cluster*. In our context, objects of interest are players or artificial agents which are represented via finite sets of measurements or *features*.

There are many different clustering algorithms with various strengths and weaknesses. Often, they are tailored to different problems in different fields so that the definition of what constitutes a cluster may vary and an understanding of cluster models is vital for their correct application [28].

Objects are often represented in terms of *feature vectors* $\mathbf{x} \in \mathbb{R}^m$ and thus embedded in an m dimensional Euclidean space where each dimension represents a measurable object attribute or property. Data as to n objects are often gathered in an $m \times n$ matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ whose n columns correspond to the feature vectors of the objects. Alternatively, data may be given as an $n \times n$ proximity (or distance) matrix $\mathbf{S}$ containing pairwise similarities (or dissimilarities) between objects. More complex data structures can be clustered, too, but we will not consider them here.

One of the first operations typically necessary in a game context is feature standardization in order to normalize the data (see section IV-B).

Any assignment of objects to clusters can be either *hard* or *soft*. While hard clustering unequivocally assigns each object to a single cluster, soft assignments produce degrees of memberships of objects to clusters (see sections IV and V).

Note that clustering is not an automatic process, but rather an iterative procedure of knowledge discovery that requires choosing and comparing algorithms and parameters. Accordingly, clustering requires care because the same data can lead to different outcomes depending on algorithms and parameters considered. There is thus no "correct" algorithm but rather a question of testing algorithms and parameters.

Classification methods, too, are used for player or agent behavior analysis. Classification differs from clustering in that the goal is to assign new objects to pre-defined categories. For game behavior data where suitable categories are typically unknown a-priori, it is therefore common to apply classification only after cluster analysis has been performed.

B. Types of Clustering Algorithms

Clustering algorithms can be categorized according to their underlying models. The following taxonomy provides a basic overview but is not the only possible typification (see [27]).

Hierarchical clustering algorithms are greedy approaches based on proximities. Clustering is agglomerative (beginning with individual objects and merging them) or divisive (beginning with the whole set of objects and

partitioning it). Care must be taken to ensure that outliers do not cause cluster merging (chaining). Generally, these models do not scale well to big data and critically depend on appropriate distance functions. Finally, hierarchical clustering does not produce subsets but hierarchies of objects.

Centroid clustering is frequently used in game analytics, mainly because of the simplicity and popularity of the k-means algorithm (Lloyd's algorithm). Centroid models represent clusters in terms of central vectors which do not need to be actual objects; yet, variants such as k-medoids determine centroids from among the given data. Using centroid methods, analysts must define the number of clusters; algorithms then determine suitable centers and assign objects to the nearest one. Again, different distance measures are possible and lead to different variants of centroid clustering (see section IV).

Distribution-based clustering uses statistical distribution models such as Gaussian mixtures. Clusters reflect how likely it is that objects belong to the same distribution. Distribution models provide information beyond mere cluster assignments. They can reveal correlations of attributes, but may suffer from over-fitting if the model complexity is not constrained. Importantly, these models produce questionable results, if the distribution that is tested for does not match the data well. In particular, the common practice of assuming data to adhere to Gaussian distributions is often erroneous.

Density clustering determines areas of high density and applies local density estimators so that clusters (i.e. regions in a data space) can have arbitrary shapes. Density clustering requires analysts to define density parameters and termination criteria. Common methods are self organizing maps, mean-shift, or DBSCAN [14], [29], [30]. The key limitation of these methods is the need for a density to drop sharply at cluster boundaries. So far, density-based models have rarely been used on game data but might find use soon as they can be tuned for efficiency [31] and are able to handle noise which is a common occurrence in behavioral game telemetry [3], [14].

C. Model Validation

Clustering results need to be validated before used further. A good clustering consists of clusters with high intra-cluster similarity and low inter-cluster similarity. However, these notions critically depend on whether vector, ratio, ordinal, categorical, Boolean, or interval-scaled features are considered.

Many quality measures have been proposed to assess how well different algorithms perform. Two common approaches are internal and external evaluation. While internal evaluation operates on the data itself, external validation evaluates results based on data not used during cluster analysis. Both approaches have their strengths and weaknesses. For example, methods that assign high scores to algorithms that produce high intra-cluster similarity and low inter-cluster similarity are biased towards centroid clustering algorithms and thus may overrate their results. External validation may suffer from the fact that clustering is often used in situations where there are no known classes or other prior knowledge to compare with.

III. Challenges in Game Development

Clustering behavioral data from games does not pose unique challenges per se since problems due to dimensionality and scale exist in other fields as well. Similarly, challenges related to the choice of algorithms and the application of results are known in other domains, too. However, the diversity in the design of digital, analog, and augmented reality or real world games means that cluster models may not transfer well across games even within the same genre thus adding further complexity [3], [32]. Hence, key concerns encountered when using cluster analysis to evaluate player behavior include:

a) Validation: This is a common problem in analytics that requires expertise. Moreover, while clustering algorithms are increasingly available in statistical software, testing procedures are not always provided.

b) Interpretation and visualization: Validated results should be interpreted w.r.t. the application context. A model can match the data but nevertheless be uninformative. Careful feature selection helps avoiding results of low practical value.

c) Time-dependency: Players may change their behavior, game communities evolve, and game content may be updated. Therefore, clusters, too, may evolve [20]. Traditional cluster analysis provides a static snapshot of behavior and thus has a limited shelf-life. Accordingly, analysis should be re-run after patch releases and should follow a regular schedule for persistent games.

d) Progress dependency: Games often feature progression mechanics, for example, in form of characters gaining experience points or improved equipment. Because players might be at different stages, certain inquiries should not be done across an entire population, but consider stages. The work in [16] highlighted this issue by looking at players in a MMORPG where features such as, say, a character's wealth would not make sense without taking into account the character's level of progression.

e) High dimensionality and big data: Cluster analysis of game data must cope with feature vectors of up to thousands of dimensions. In addition, very large data sets are becoming increasingly common. Yet, many common clustering algorithms do not scale well to large data sets or are inept when dealing with the peculiar geometry of high dimensional spaces where, among others, distances between points become relatively uniform [10]–[12] which invalidates the notion of a nearest neighbor. Big data and high dimensionality are active areas of research and clustering methods for these settings are becoming increasingly available [33]–[35].

f) Data type mixing: A typical problem of behavior analysis in games is the mixing of data types as behavioral features are often measured in different ways. For example, ratios (e.g. kill/death ratio or hit/miss ratio) may be mixed with categorical data (e.g. character type). Such data require a careful consideration of data normalization strategies [36].

g) Feature selection: Because of the wide range of behavioral features, feature selection is of increasing relevance for the game industry. While monetization metrics such as DAU (Daily Active Users) consider features that apply across games, these can be defined in different ways. For example, DAU provides a concrete temporal frame, but what constitutes an "active" player is nebulous. Problems with feature selection usually occur when evaluating player behavior w.r.t. game mechanics. Observing, say, that players will likely churn at level 8 is in and

of itself not valuable; discovering the underlying causes is. Identifying relevant features in situations like this can be difficult but cluster analysis can assist in unveiling reasons such as overly parsimonious reward mechanisms [5].

IV. A Baseline Algorithm: k-Means Clustering

We begin the tutorial part of this paper by studying k-means clustering. Although the k-means algorithm is a popular and widely used baseline technique, experience shows that practitioners are often not familiar with its principles. That is to say that the algorithm operates on implicit assumptions which, if ignored, may lead to seemingly unreasonable results. We analyze these principles, discuss when and how to apply k-means, and point out pitfalls in its use.

We begin by considering data in Euclidean spaces. Hence, assume a data set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$ whose n elements $\mathbf{x}_j$ are m dimensional real-valued vectors that are to be clustered. Whenever we set out to cluster such data, we are interested in regularities within the sample and must decide how to characterize and how to search for latent structures.

The k-means algorithm provides the arguably most popular approach to these problems. It represents structures in terms of *subsets* of X and attempts to subdivide the data into k different clusters $C_i \subset X$ which meet the following criteria: First of all, the clusters should be pairwise disjoint so that $C_i \cap C_j = \emptyset$ for $i \neq j$. Second of all, the k different clusters should cover the data so that $C_1 \cup C_2 \cup \ldots \cup C_k = X$. Third of all and most importantly, data points assigned to a cluster C_i are supposed to be *similar*.

While the first two criteria are well defined, the third one hinges on the rather intuitive notion of similarity. Indeed, there are many ways of defining similarity and differences between clustering algorithms usually trace back to different notions of what it means to be similar. The idea applied in k-means clustering is to represent clusters C_i by *centroids* $\boldsymbol{\mu}_i \in \mathbb{R}^m$ and to measure similarities in terms of Euclidean distances between centroids and data points. Hence, two data points are considered similar if their distances to a common centroid are small and point $\mathbf{x}_j$ will be assigned to cluster C_i if its distance to $\boldsymbol{\mu}_i$ is less than that to any other centroid.

This basic idea reduces k-means clustering to the problem of finding appropriate centroids. Mathematically, this can be cast as the problem of minimizing the following objective function

$$E(k) = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2 \tag{1}$$

with respect to k centroids $\boldsymbol{\mu}_i$. In other words, minimizing (1) requires to partition the data into k clusters C_i such that sums of distances between data points and their closest cluster centroid become as small as possible.

Note that the objective in (1) can also be expressed in terms of indicator variables z_{ij} which register for any data point $\mathbf{x}_j$ whether it belongs to cluster C_i. That is, by defining

$$z_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \in C_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

we find the objective function in (1) to be equivalent to

$$E(k) = \sum_{i=1}^{k} \sum_{j=1}^{n} z_{ij} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2. \tag{3}$$

Although both objectives formalize an intuitive problem, an optimal solution, i.e. global minimizers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$, may be difficult to obtain. Even for 2D data and $k = 2$, there is no guarantee for the best possible solution to be found in reasonable time [37]. This is important to know, because it indicates that **the k-means algorithm is but a heuristic for dealing with a surprisingly hard problem**.

Indeed, the k-means algorithm merely realizes greedy optimization. When started, say at $t = 0$, it randomly initializes the parameters $\boldsymbol{\mu}_1^{(t)}, \boldsymbol{\mu}_2^{(t)}, \ldots, \boldsymbol{\mu}_k^{(t)}$ to optimize. Given these initial guesses, the data are clustered accordingly:

$$C_i^{(t)} = \left\{ \mathbf{x}_j \in X \;\middle|\; \left\| \mathbf{x}_j - \boldsymbol{\mu}_i^{(t)} \right\| \leq \left\| \mathbf{x}_j - \boldsymbol{\mu}_l^{(t)} \right\| \; \forall \, l \neq i \right\}. \tag{4}$$

Once clusters have been determined, the algorithm updates its estimates of their centroids using

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{n_i} \sum_{\mathbf{x}_j \in C_i^{(t)}} \mathbf{x}_j \tag{5}$$

where $n_i = |C_i^{(t)}|$ denotes the number of elements currently assigned to cluster C_i. As these updates may lead to a new clustering, the iteration counter is set to $t = t + 1$ and steps (4) and (5) are repeated until the assignment of data points to clusters does not change anymore or t exceeds a predefined number of iterations (see Fig. 2 for an illustration).

It is easy to prove that updating each centroid to the sample *mean* of its cluster will never increase the value of objectives (1) or (3). Each iteration therefore improves on the previous result and k-means usually converges quickly. However, as the algorithm starts from a random initialization, it is not guaranteed to find the global minimum of its objective. Since it typically converges to a local minimum, **it is pivotal to run k-means several times so as to empirically determine good clusters**. Yet, even then, k-means may produce questionable results when seeded inappropriately. Indeed, intelligent initializations are actively researched [38], [39] and **arbitrary initialization are considered harmful; initial centroids should at least be selected among the available data points**.

A. Probabilistic Interpretation

In appendix A, we apply maximum likelihood arguments to show that the k-means algorithm fits a simplified mixture of Gaussians. In other words, **k-means clustering implicitly assumes the given data to consist of k Gaussian densities and produces corresponding results**. Here, we resort to linear algebra to show that, in addition, **the k-means algorithm implicitly clusters data into Gaussians of low variance**.

To expose this, we consider an arbitrary cluster C_i computed during one of the iterations. Looking at (1), we note that this cluster contributes a term of $\sum_{\mathbf{x}_j \in C_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2$ to $E(k)$ and that the smaller this contribution the more likely this cluster will persist throughout the optimization procedure.

If we set $\mathbf{y}_j = \mathbf{x}_j - \boldsymbol{\mu}_i$ and collect the new vectors $\mathbf{y}_j$ in a matrix $\mathbf{Y}_i \in \mathbb{R}^{m \times n_i}$, it is easy to see that

$$\sum_{\mathbf{x}_j \in C_i} \left\|\mathbf{x}_j - \boldsymbol{\mu}_i\right\|^2 = \left\|\mathbf{Y}_i\right\|_F^2 \tag{6}$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm. However, for the Frobenius norm we have $\left\|\mathbf{Y}_i\right\|_F^2 = \mathrm{tr}[\mathbf{Y}_i \mathbf{Y}_i^T]$ where the trace operator $\mathrm{tr}[\cdot]$ sums the diagonal entries of a matrix. We also note that $\boldsymbol{\Sigma}_i = \mathbf{Y}_i \mathbf{Y}_i^T$ is a covariance matrix. It can be diagonalized using $\boldsymbol{\Lambda}_i = \mathbf{U}_i^T \boldsymbol{\Sigma}_i \mathbf{U}_i$ where the orthogonal matrix $\mathbf{U}_i$ contains the eigenvectors of $\boldsymbol{\Sigma}_i$ and $\boldsymbol{\Lambda}_i$ is a diagonal matrix whose non-zero entries denote eigenvalues or variances along the principal axes of cluster C_i. Finally, we recall that traces are invariant under similarity transformations, i.e. $\mathrm{tr}[\boldsymbol{\Sigma}_i] = \mathrm{tr}[\boldsymbol{\Lambda}_i]$. This, however, proves our claim: By minimizing the objective in (1), the k-means algorithm minimizes traces of covariance matrices and therefore produces compact clusters of low internal variance.

B. Common Pitfalls

We just saw that the k-means algorithm is tailored towards locally Gaussian data. Put differently, if k-means clustering is applied to data that do not contain compact convex subsets, it is unrealistic to expect it to identify reasonable clusters. Figures 3 and 4 illustrates what this may mean in practice.

Figure 3(a) displays a set of 2D data in which human observers immediately recognize two distinct linear structures. Yet, if we set $k = 2$ and run k-means, the result will look like in Fig. 3(b). That is, for data such as these, the algorithm will determine clusters that hardly agree with human intuition. In particular, we note that the cluster centroids (□) found are rather far from actual data points but form the centers of two imaginary circles which each encompass about half of the data. Given our above analysis, this result was to be expected and perfectly illustrates the tendency of k-means clustering to produce compact, blob-like clusters.

As a remedy, the literature often suggests to normalize the data to zero mean and unit variance using $y_{dj} = (x_{dj} - \mu_d)/\sigma_d^2$ where x_{dj} is the dth component of $\mathbf{x}_j$ and μ_d and σ_d^2 denote mean and variance along dimension d. However, Fig. 3(c) shows that this hardly affects the result; the data are simply not Gaussian enough for k-means to identify the structures a human would consider important.

Hence, if we insist on applying k-means clustering to non-Gaussian data, we need more appropriate pre-processing. A reasonable idea is to *whiten* the data. Using the sample mean $\boldsymbol{\mu} = \frac{1}{n}\sum_j \mathbf{x}_j$ and setting $\mathbf{y}_j = \mathbf{x}_j - \boldsymbol{\mu}$ provides the sample covariance matrix $\boldsymbol{\Sigma} = \mathbf{Y}\mathbf{Y}^\mathbf{T}$ which can be decorrelated $\boldsymbol{\Lambda} = \mathbf{U}^T \boldsymbol{\Sigma} \mathbf{U}$ through principal component analysis. Still, variances along the different dimensions will usually differ; but since $\boldsymbol{\Lambda}^{-1/2} \boldsymbol{\Lambda} \boldsymbol{\Lambda}^{-1/2} = \mathbf{I}$, the transformation

$$\mathbf{w}_j = \boldsymbol{\Lambda}^{-1/2} \mathbf{U}^T (\mathbf{x}_j - \boldsymbol{\mu}) \tag{7}$$

maps the data to vectors for which $\frac{1}{n}\sum_j \mathbf{w}_j \mathbf{w}_j^T = \mathbf{I}$. That is, after whitening, variances in the directions of the principal axes of the data are identically 1. Geometrically, we can picture this transformation as mapping the data (close) to the surface of a sphere where possible clusters will be more separated than in the original representation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games

Figure 3(d) shows the whitened data and the corresponding result of k-means clustering. Based on this example, it appears that **proper preprocessing enhances the chances for the k-means algorithm to identify structures that are not necessarily Gaussian**. Nevertheless, there is still no guarantee for it to do so.

Figure 4 presents another example of where k-means may be inappropriate. It shows a waypoint graph that indicates how non-player characters may move through a dungeon. Faced with the problem of path planning, modern game AI resorts to hierarchical variants of the A^* algorithm. Here, clustering can help to automatically identify coherent areas of a map and thus facilitate hierarchical path planning. However, even though the waypoints in this example form rather blob-like clusters, the k-means algorithm constantly produces unintuitive results as it groups together waypoints from different rooms. This example therefore illustrates that k-means clustering considers geometry rather than topology. If two waypoints are spatially close they may end up in the same cluster, even though their geodesic distance is actually large. Yet, the example also points out that algorithms such as spectral clustering which consider topological information (i.e. edges between vertices in a graph) may produce much more reasonable results. Therefore, **k-means clustering should not be considered the only tool available to analysts**. Rather, clustering algorithms should be chosen according to the type of information available. Further details on spectral clustering will be discussed in section VI.

To conclude this section, we point out yet another pathology, namely the case of very high dimensional data where we are given n data points $\mathbf{x}_j \in \mathbb{R}^m$ but $n \ll m$. Here, the notion of similarity invoked by k-means may be useless, because as dimensions grow so do distances and it becomes very likely that all the given data are equally far apart [10]–[12]. Hence, centroids determined by k-means clustering will be far from any data point and the decision of whether to assign a data point to a cluster will basically depend on minuscule random variations rather than on structural properties. Therefore, **for high dimensional data, k-means clustering should be applied only after dimensionality reduction**.

C. Soft k-Means

Appendix A shows that k-means clustering is closely related to Gaussian mixture modeling. This insight suggests several extensions of the original approach. On the one hand, the algorithm can be extended such that it also determines variance parameters. On the other hand, the requirement of pairwise disjoint clusters may be relaxed by allowing the indicator variables z_{ij} in (2) to assume values $0 \leq z_{ij} \leq 1$ such that $\sum_i z_{ij} = 1$. This way, data points can be assigned degrees of membership to different clusters. Both extensions are variants of what is called *soft* or *fuzzy* k-means [36] and usually require the expectation maximization (EM) algorithm for parameter estimation [40].

D. Kernel k-Means

Looking at the objective in (3), we note that we may write

$$E(k) = \sum_{i=1}^{k} \sum_{j=1}^{n} z_{ij} \left(\mathbf{x}_j^T \mathbf{x}_j - 2\boldsymbol{\mu}_i^T \mathbf{x}_j + \boldsymbol{\mu}_i^T \boldsymbol{\mu}_i \right). \tag{8}$$

Furthermore, we note that

$$\boldsymbol{\mu}_i^T \mathbf{x}_j = \frac{1}{n_i} \sum_{l=1}^{n} z_{il} \mathbf{x}_l^T \mathbf{x}_j \tag{9}$$

and that

$$\boldsymbol{\mu}_i^T \boldsymbol{\mu}_i = \frac{1}{n_i^2} \sum_{l=1}^{n} \sum_{j=1}^{n} z_{il} z_{ij} \mathbf{x}_l^T \mathbf{x}_j, \tag{10}$$

The k-means objective function can thus be written entirely in terms of inner products between data vectors. This allows for invoking the *kernel trick* where we replace inner products $\mathbf{x}_j^T \mathbf{x}_l$ by non-linear kernel functions $k(\mathbf{x}_j, \mathbf{x}_l)$. This trick has become a staple of data analysis as it allows for using linear techniques to deal with nonlinear problems [41]. Regarding k-means clustering, one may thus obtain reasonable clusters even for non-Gaussian data. However, invoking the kernel trick usually increases computation times, requires experience in choosing an appropriate kernel function, and necessitates particularly careful initializations of the algorithm.

E. k-Medoids Clustering

Sometimes, we are dealing with data which are not additive so that the notion of a mean is ill defined. An example related to game mining is the problem of clustering player names. As such, names, i.e. strings of characters, do not allow for computing averages. Nevertheless, we can compute, say, the edit distance between strings and the fact that the k-means algorithm can be kernelized indicates that it is applicable to situations like these. An even simpler solution is to consider the use of the k-medoids algorithm. In appendix B, we prove that the data point $\mathbf{x}_j \in X$ which minimizes

$$\mathbf{x}_j = \underset{\mathbf{x}_l}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left\| \mathbf{x}_l - \mathbf{x}_i \right\|^2 \tag{11}$$

is the point in X that is closest to the sample mean $\boldsymbol{\mu}$. It is called the *medoid* of X. Since the minimizer in (11) can be computed for any norm rather than just for the Euclidean norm, the notion of a medoid applies to any kind of data for which we can define distances. This suggests a variant of k-means clustering where we proceed as usual but replace the computation of means in (5) by the computation of medoids.

V. Clustering as Matrix Factorization

Interestingly, the task of k-means clustering can be viewed as a *matrix factorization problem*. In data mining and pattern recognition, factorization methods are frequently used for dimensionality reduction or latent component detection. In the basic setting, we are given a data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ whose n columns are m dimensional vectors $\mathbf{x}_j$ and we look for two factor matrices $\mathbf{W} \in \mathbb{R}^{m \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times n}$ such that

$$\mathbf{X} \approx \mathbf{W}\mathbf{H} \tag{12}$$

where $k \ll \min\{m, n\}$. Usually, the columns $\mathbf{w}_i \in \mathbb{R}^m$ of $\mathbf{W}$ are called *basis vectors* and the columns $\mathbf{h}_j \in \mathbb{R}^k$ of $\mathbf{H}$ are coefficient vectors. This becomes clear if we consider that for every column $\mathbf{x}_j$ in $\mathbf{X}$, there is a column

$\mathbf{h}_j$ in $\mathbf{H}$ and that $\mathbf{x}_j \approx \mathbf{W}\mathbf{h}_j$. That is, upon successful factorization every data point $\mathbf{x}_j$ can be approximated as a linear combination of the columns of $\mathbf{W}$.

To see how this relates to k-means clustering, we consider a data matrix $\mathbf{X}$, a matrix of centroid vectors $\mathbf{M}$, and a matrix of indicator variables $\mathbf{Z}$. Given the above notation, we then find that, upon convergence of the k-means algorithm, we have

$$\begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \dots & x_{mn} \end{bmatrix} \approx \begin{bmatrix} \mu_{11} & \dots & \mu_{1k} \\ \vdots & & \vdots \\ \mu_{m1} & \dots & \mu_{mk} \end{bmatrix} \begin{bmatrix} z_{11} & \dots & z_{1n} \\ \vdots & & \vdots \\ z_{k1} & \dots & z_{kn} \end{bmatrix}$$

or simply $\mathbf{X} \approx \mathbf{M}\mathbf{Z}$. In other words, we can think of the k-means algorithm as an approach towards approximating

$$\mathbf{x}_j \approx \mathbf{M}\mathbf{z}_j. \tag{13}$$

where actually $\mathbf{x}_j \approx \mathbf{M}\mathbf{z}_j = \boldsymbol{\mu}_i$, because the coefficient vector $\mathbf{z}_j$ is a binary vector with exactly one component equal to 1.

Accordingly, the objective in (1) can also be stated as a quadratic optimization problem involving matrices, namely

$$\min_{\mathbf{M},\mathbf{Z}} \left\| \mathbf{X} - \mathbf{M}\mathbf{Z} \right\|^2. \tag{14}$$

Even though this form may look even more innocent than the ones in (1) and (3), it still constitutes a difficult problem. Indeed, this formulation allows us to better appreciate the difficulty of k-means clustering. Note that (14) is convex in $\mathbf{M}$. That is, by fixing $\mathbf{Z}$, (14) turns into a quadratic problem in $\mathbf{M}$ for which there is a closed form solution. Likewise, it is convex in $\mathbf{Z}$ so that, by fixing $\mathbf{M}$, one can easily compute the optimal $\mathbf{Z}$. Yet, the problem is anything but convex in the product $\mathbf{M}\mathbf{Z}$. If both matrices are to be determined simultaneously, the objective function typically shows numerous local minima and no algorithm is known to date that is guaranteed to find a globally optimal solution in reasonable time.

A. k-Means as Constrained Quadratic Optimization

Looking at (14) reveals that the k-means heuristic is an alternating least squares scheme. In our new terminology, the k-means algorithm first fixes $\mathbf{M}$ and solves for $\mathbf{Z}$, then fixes $\mathbf{Z}$ and solves for $\mathbf{M}$, and repeats these steps until convergence. Alas, things are more complicated because k-means actually deals with a *constrained quadratic optimization* problem. Note that the centroid vectors $\boldsymbol{\mu}_i$, i.e. the columns of $\mathbf{M}$, must be convex combinations of data points $\mathbf{x}_j$. That is, $\boldsymbol{\mu}_i = \mathbf{X}\mathbf{y}_i$ where $\mathbf{y}_i$ is an n dimensional vector with n_i entries equal to $1/n_i$ and $n - n_i$ entries equal to 0. To express all centroids at once, we may write $\mathbf{M} = \mathbf{X}\mathbf{Y}$ and keep in mind that $\mathbf{Y}$ must be non-negative, i.e. $\mathbf{Y} \succeq \mathbf{0}$, that its columns must sum to one, i.e. $\sum_j y_{ji} = 1$, and that they should have high entropy $H(\mathbf{y}_i) = -\sum_j y_{ji} \log y_{ji} \gg 0$.

Second of all, matrix $\mathbf{Z}$ is supposed to be a binary matrix. We can enforce this by requiring $\mathbf{Z} \succeq \mathbf{0}$, $\sum_i z_{ij} = 1$, and $H(\mathbf{z}_i) = -\sum_i z_{ij} \log z_{ij} = 0$. Incorporating these constraints into (14), we find that k-means clustering can

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games

be formalized as the following constrained quadratic problem

$$\begin{aligned} &\min_{\mathbf{Y},\mathbf{Z}} \left\|\mathbf{X} - \mathbf{XYZ}\right\|^2 \\ &\text{s.t. } \mathbf{Y} \succeq \mathbf{0},\ \mathbf{1}^T\mathbf{y}_i = 1,\ H(\mathbf{y}_i) \gg 0 \\ &\quad \mathbf{Z} \succeq \mathbf{0},\ \mathbf{1}^T\mathbf{z}_j = 1,\ H(\mathbf{z}_j) = 0. \end{aligned} \tag{15}$$

B. Archetypal Analysis

If we drop the entropy constraints in (15), we obtain

$$\begin{aligned} &\min_{\mathbf{Y},\mathbf{Z}} \left\|\mathbf{X} - \mathbf{XYZ}\right\|^2 \\ &\text{s.t. } \mathbf{Y} \succeq \mathbf{0},\ \mathbf{1}^T\mathbf{y}_i = 1,\ \mathbf{Z} \succeq \mathbf{0},\ \mathbf{1}^T\mathbf{z}_j = 1 \end{aligned} \tag{16}$$

and recover a problem known as *archetypal analysis* (AA) [42]. Dropping entropy constraints has an interesting effect. Instead of computing basis vectors $\mathbf{M} = \mathbf{XY}$ that correspond to local means, AA determines basis vectors that are extreme points of the data. In fact, the *archetypes* in matrix $\mathbf{M}$ reside on the data convex hull. Thus, **AA does not operate on any implicit density assumptions**; it can be computed quickly and often yields results that are more pronounced than those of k-means clustering (see section VII).

C. Non-negative Matrix Factorization

If we also drop the stochastic constraints in (16) and relax the requirement that $\mathbf{M} = \mathbf{XY}$, we obtain

$$\begin{aligned} &\min_{\mathbf{M},\mathbf{Z}} \left\|\mathbf{X} - \mathbf{MZ}\right\|^2 \\ &\text{s.t. } \mathbf{M} \succeq \mathbf{0},\ \mathbf{Z} \succeq \mathbf{0} \end{aligned} \tag{17}$$

a problem known as *non-negative matrix factorization* (NMF) [43]. **NMF is particularly useful whenever given data are non-negative and it typically yields sparse basis vectors**. That is, one can show that NMF basis vectors form a cone that includes the data [44] and therefore contain only few entries significantly larger than zero. This way, NMF allows for part based representations, i.e. for linear combinations over latent factors that each represent independent aspects of the data.

VI. Spectral Clustering

In this section, we discuss *spectral clustering*, another approach that applies matrix factorization. While k-means clustering is guided by local properties of data (i.e. distances to centroids), spectral clustering assumes a global point of view. While k-means clustering is clustering with $m \times n$ data matrices $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]$, spectral clustering considers $n \times n$ similarity matrices $\mathbf{S}$ whose entries $S_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$ indicate possibly abstract affinities between data objects. It is therefore related to the problem of graph partitioning, because affinity matrices $\mathbf{S}$ can be seen as weighted graph adjacency matrices.

Since spectral clustering is less well known among game analysts, we discuss it using an instructive example, namely, the problem of automatically partitioning game maps into coherent units. This is motivated by the fact that in-game behavior often depends on in-game location. Consider, for instance, the part of the *Quake II* map *q2dm1* in Fig. 5. The figure highlights distinct architectural features for which the game mechanics demand different behaviors. Moving along the ledges, for instance, requires more prudence than moving in the yard, as players try to avoid dropping to their virtual deaths. If it was possible to automatically identify coherent parts of a map, say from analyzing movement behaviors of players, we might either use this information to distinguish different play styles based on localized activities [16], [17] or to train gamebots that behave more human-like [45], [46].

Here, we assume the data to be given in form of a waypoint graph derived from observations of human movement behavior. Indeed, using the QASE programming interface [47] for *Quake II* and k-means clustering, it is straightforward to compute waypoint graphs as shown in Fig. 6. Looking at the figure, we recognize a graph whose n vertices correspond to prototypical 3D player positions during a game and whose edges indicate observed transitions between waypoints; different measures of waypoint similarity are discussed below.

In a seminal paper [48], Fiedler showed that spectral decompositions allow for graph partitioning and proved that clusters can be determined from looking at the eigenvectors belonging to the k smallest eigenvalues of the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{S}$ where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_j s_{ij}$.

Spectral clustering therefore relies on algebraic graph theory and its name derives from the fact that it clusters according to spectral properties, i.e. eigenvectors, of the Laplacian matrix. Unfortunately, the related machine learning literature is often vague as to why the Laplacian emerges in this context and thus obscures implicit assumptions of this method. Yet, there is an intuitive physical analogy: Imagine the graph in Fig. 6 as a spring-mass system where every vertex corresponds to a point mass and every edge to a spring. If one or more of the masses were excited (moved from their location and let loose again), the system would vibrate. Now, recall that such vibrations are characterized by second order differential equations whose solutions are given in form of eigenfunctions of the Laplace operator. As the Laplacian matrix is a discrete analogue of the Laplace operator, we can thus understand spectral clustering as the process of looking for *standing waves* on a graph and grouping together vertices that show the same dynamics.

Though there are many variants of spectral clustering [49]–[52], they usually vary the following baseline approach. Given the normalized Laplacian

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{S}\mathbf{D}^{-\frac{1}{2}} \tag{18}$$

of $\mathbf{S}$, we compute its decomposition, determine the k smallest eigenvalues, collect the corresponding eigenvectors in a matrix $\mathbf{U} \in \mathbb{R}^{n \times k}$, apply k-means to the n rows of $\mathbf{U}$, and thus group the n objects under consideration.

Given the problem of finding four clusters in the graph in Fig. 6, the results in Fig. 7 illustrate differences between spectral clustering and other techniques. Ward-linkage (Fig. 7(a)) is a hierarchical clustering algorithm that tends to group densely co-located point and k-means clustering (Fig. 7(b)) produces rather equally sized blob-like clusters. In our example, neither approach yields appropriate partitions. The upper ledges, for instance, are horizontally split

into two clusters and some of the points in the yard and on the stairs are fused into amorphous clusters. Note that both methods only consider coordinates of data points but ignore additional relational information. This is different for spectral clustering. Because of this, it is able to detect intrinsic structures in the waypoint graph that correspond to architectural features of the map.

Figure 8(a) shows 4 areas that result from spectral clustering using waypoint similarities which were computed based on their Euclidean distances

$$s_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} \quad \text{where} \quad \sigma = 10. \tag{19}$$

Here, the upper ledges and the yard form prominent clusters. Still, we observe artifacts such as the amorphous cluster in the center and the solitaire point on the far right. However, if we consider similarities according to topological distances, a even clearer picture appears.

The similarity matrix used to produce the result in Fig. 8(b) was given by $s_{ij} = 1 - d_{ij}$ where d_{ij} indicates the length of the shortest path between waypoint i and j. Here, the upper ledges were correctly clustered into two vertically separated parts; the yard and the staircase leading to it are recognizable, too. Topology also explains why the waypoints in midair are clustered with the second highest ledge. First of all, the recording player mostly jumped from there. Second of all, as it is impossible to reach midair points from the ground level, their path length similarity to the points on the ledge is much higher than to the points in the yard even if some of the latter might be geometrically closer. This example therefore underlines that **choosing an appropriate metric is vital for obtaining reasonable clustering results** and that **cluster validation is a necessary step of any cluster analysis**.

VII. Application Examples and Guidelines

K-means clustering is a popular baseline in industry and academia and has been applied to game behavioral data before. For example, the authors of [21] used k-means for difficulty adjustments in a shooter-type game.

As a more detailed example, consider the the application of k-means and AA to player behavior data from the MMORPG *Tera* described in [16]. There, the authors considered 250,000 players' data and various features associated with character abilities and gameplay, such as class, race, numbers of monsters killed, number of in-game friends, skill levels, etc. Given the variation in these features, different normalization options were examined, before Min-Max normalization was used. The difference between AA and k-means is exemplified in Tables I and II. For both approaches, 6 to 7 clusters consistently emerged as the best fit. For both algorithms, these clusters shared a group of players with exceptionally high values across most features (termed *Elite*) and another exhibiting the lowest values (termed *Stragglers*). The remaining clusters were split into four groups: Two with middling scores but different high scores, one better than the other, but low Plants and Mining skills; and two with comparable scores, but high Plants and Mining skills. Within these, there are distinct variations across k-means and AA. For example, AA identified clusters of players with high or low distinctive scores, such as the *Auction Devils* (successfully use the auction house) and the *Friendly Pros* (many friends). In comparison, k-means provides more broadly defined clusters, e.g. providing several clusters of scores around the averages, and minor variation in specific features. Following [14], clusters were given descriptive names.

Recall that the central goal of cluster analysis in game analytics is to arrive at interpretable clusters that accurately encapsulate the behaviors of players. However, given how much games vary in their design, the different types of questions cluster analysis can be fielded against, and the existence of dozens or even hundreds of models, it is rather difficult to provide guidelines for which approach to use in which contexts. The degree of variance in purpose, data, design and behaviors makes this inherently difficult. As outlined in [17], the choice of cluster model can significantly effect outcome. For example, basic k-means clustering directly assigns players to groups (via cluster centroids that represent specific behaviors), whereas AA, NMF, and PCA provide basis vectors which span the space players reside in. This means that players can be described in terms of their coefficients w.r.t. each basis vector and clustered accordingly.

While this seems to make basic k-means clustering more attractive than other methods as it saves a step in the analysis, there is an important drawback: k-means clusters represent averages and are thus not always interpretable in terms of distinct differences in the behavior of players [16].

Given these considerations, it is unlikely that a specific model will generally outperform others. Context, data, and goals of the analysis must decide which approach to use; this requires expertise not only in cluster analysis, but also regarding the game under consideration. **Yet, there are questions which can inform the choice of a cluster model**:

1) Are the data high-dimensional and/or sparse? If so, consider models tailored to sparse data (AA or NMF).
2) What is the overall goal? To build general models of player behavior or to detect extreme behaviors (e.g. cheating, gold-farming)? For the former, consider centroid-seeking models (k-means, k-medoids); for the latter, consider models such as AA.
3) Are the data numerical or relational? For the latter, use spectral clustering or kernel methods.
4) Are the players tightly grouped in variance space, so that k-means might have difficulties distinguishing them? If so, consider density-based approaches that do not operate on Euclidean distances.
5) Are the data noisy? If so density-based methods might be appropriate as they are better tunable.

VIII. Conclusion

Owing to modern tools for data acquisition and storage, collecting behavioral telemetry data from games has become a commodity. Yet, the analysis of behavioral data still suffers from a dearth of knowledge. In part, this can be explained by the business value of data which provide competitive advantages to companies. More importantly, however, it may be because adopting analytics into game development and games research is a relatively recent idea [3].

Cluster analysis allows for finding latent patterns in game data. Academics and practitioners apply it for game AI [21]–[24], behavioral profiling [14]–[17], identification of players likely to convert to paying customers [4]–[6], or progression analysis [18], [20]. The underlying principles are intuitive and there are many open source projects enabling everyone to run corresponding algorithms on behavioral data (see, for instance, Weka [53], RapidMiner [54], KNIME [55], or SciPy [56]).

Yet, while the idea of cluster analysis is straightforward, different algorithms have different strengths and weaknesses and rely on different assumptions. Further difficulties arise from typical properties of game data such as high dimensionality, time-dependency, and in-congruent measurement scales.

The goal of this paper was therefore to point out peculiarities of cluster analysis for game behavioral data. We discussed and reviewed pitfalls, common and uncommon problems, and theoretical foundations of popular algorithms. In particular, we discussed k-means clustering, matrix factorization, and spectral clustering and exposed the principles they work on. This was motivated by several years of experience obtained in the game industry and in games research. We showcased clustering applications by means of practical examples and references to the literature. These examples underlined the potential of cluster analysis for game design and development but also revealed that the application of clustering techniques to game behavioral data is still in its infancy.

Not all the issues identified in this paper pertain to cluster algorithms per se but rather to the contexts in which these techniques are applied. On the one hand, there are issues pertaining to the nature of behavioral data from games. On the other hand, there are issues related to making sure that analysis results can be delivered to and acted upon by stakeholders.

This contribution is therefore also "a call to arms" for future work on game analytics. The benefits of using computational intelligence techniques in game development and marketing have been clearly recognized by now. What is still missing are a better transfer of research into practice as well as increased efforts towards the development of methods, algorithms, and tools that clearly address the challenges of game behavior data that we have reviewed in this paper.

APPENDIX A
A PROBABILISTIC INTERPRETATION OF k-MEANS

In this appendix, we relate k-means clustering to statistical mixture models. Given a data set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we look at the probability $p(\mathbf{x}_j)$ of observing data point $\mathbf{x}_j$ and note that, if the data form k distinct clusters, $\mathbf{x}_j$ may have been produced as follows: First, sample a cluster C_i according to probability $p(C_i)$ where $\sum_i p(C_i) = 1$ and then sample a data point according to the conditional probability $p(\mathbf{x}|C_i)$. Under this model, the probability of observing $\mathbf{x}_j$ amounts to

$$p(\mathbf{x}_j) = \sum_{i=1}^{k} p(\mathbf{x}_j|C_i)\; p(C_i). \tag{20}$$

Next, we write $p(C_i) = \pi_i$ and assume the elements of each cluster to be distributed according to a multivariate Gaussian $p(\mathbf{x}|C_i) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. If we further assume them to be *isotropic*, their covariance matrices are $\boldsymbol{\Sigma}_i = \sigma_i^2\mathbf{I}$ and we have

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \sigma_i) \propto e^{-\frac{\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma_i^2}}. \tag{21}$$

Now, if we believe the data to originate from a mixture of k Gaussians, we must next determine suitable model parameters $\boldsymbol{\theta} = \{\pi_1, \boldsymbol{\mu}_1, \sigma_1, \ldots, \pi_k, \boldsymbol{\mu}_k, \sigma_k\}$. In statistical modeling, best estimates are often determined using

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games

maximum likelihood techniques. In our case, the likelihood function is given by

$$L(\boldsymbol{\theta}|X) = \prod_{j=1}^{n} p(\mathbf{x}_j) = \prod_{j=1}^{n} \sum_{i=1}^{k} \pi_i \, \mathcal{N}(\mathbf{x}_j|\boldsymbol{\mu}_i, \sigma_i) \tag{22}$$

and the corresponding log-likelihood $\mathcal{L} = \log L$ amounts to

$$\mathcal{L}(\boldsymbol{\theta}|X) = \sum_{j=1}^{n} \log \left[\sum_{i=1}^{k} \pi_i \, \mathcal{N}(\mathbf{x}_j|\boldsymbol{\mu}_i, \sigma_i) \right]. \tag{23}$$

Suitable parameters then result from deriving (23), equating to zero, and solving the resulting equations. Alas, for Gaussian mixtures, this leads to coupled systems of equations without closed form solutions. Moreover, the problem would be much easier, if we knew which component a data point came from. In their seminal paper on the expectation maximization (EM) algorithm, Dempster et al. [40] therefore proposed to pretend this information was available and to introduce additional latent variables whose values must be estimated, too. Let us therefore assume a set $Z = \{z_{11}, z_{12}, \ldots, z_{nk}\}$ of indicator variables just as introduced in (2). We can then express the *complete* likelihood of our problem as follows

$$L(\boldsymbol{\theta}|X, Z) = \prod_{j=1}^{n} \sum_{i=1}^{k} z_{ij} \; \pi_i \, \mathcal{N}(\mathbf{x}_j|\boldsymbol{\mu}_i, \sigma_i). \tag{24}$$

The key observation at this point is that $z_{ij} \in \{0, 1\}$ and that $\sum_i z_{ij} = 1$. Because of these peculiar properties, we have

$$\sum_{i=1}^{k} z_{ij} \; \pi_i \, \mathcal{N}(\mathbf{x}_j|\boldsymbol{\mu}_i, \sigma_i) = \prod_{i=1}^{k} \Big[\pi_i \, \mathcal{N}(\mathbf{x}_j|\boldsymbol{\mu}_i, \sigma_i) \Big]^{z_{ij}} \tag{25}$$

so that the corresponding complete log-likelihood becomes

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}|X, Z) &= \sum_{j=1}^{n} \sum_{i=1}^{k} z_{ij} \Big[\log \pi_i + \log \mathcal{N}(\mathbf{x}_j|\boldsymbol{\mu}_i, \sigma_i) \Big] \\ &= \sum_{j=1}^{n} \sum_{i=1}^{k} z_{ij} \log \pi_i - \sum_{j=1}^{n} \sum_{i=1}^{k} \frac{z_{ij}}{2\sigma_i^2} \big\| \mathbf{x}_j - \boldsymbol{\mu}_i \big\|^2 \end{aligned} \tag{26}$$

Note that we just found the k-means objective in (3) as a term on the right hand side of (26) and that minimizing (3) maximizes (26). Also, observe that the k-means procedure implicitly sets $\sigma_i^2 = 1/2$ and therefore determines a particularly simplified Gaussian mixture model.

APPENDIX B
THE BASIS FOR k-MEDOIDS CLUSTERING

Here, we prove a lemma that justifies the idea of k-medoids clustering. For brevity, we only consider Euclidean distances; extensions to other distance measures are straightforward.

Lemma 1. *Given $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$, let $\boldsymbol{\mu} = \frac{1}{n} \sum_i \mathbf{x}_i$ be the sample mean and $\|\cdot\|$ be the Euclidean norm. Then*

$$\frac{1}{n} \sum_i \big\| \mathbf{x}_j - \mathbf{x}_i \big\|^2 \leq \frac{1}{n} \sum_i \big\| \mathbf{x}_k - \mathbf{x}_i \big\|^2$$

implies that

$$\left\|\mathbf{x}_j - \boldsymbol{\mu}\right\|^2 \leq \left\|\mathbf{x}_k - \boldsymbol{\mu}\right\|^2.$$

That is, the point $\mathbf{x}_j \in X$ with the smallest average distance to all other points in X is closest to the sample mean $\boldsymbol{\mu}$.

Proof: Note that

$$\frac{1}{n}\sum_i \left\|\mathbf{x}_j - \mathbf{x}_i\right\|^2 = \frac{1}{n}\sum_i \left\|(\mathbf{x}_j - \boldsymbol{\mu}) - (\mathbf{x}_i - \boldsymbol{\mu})\right\|^2.$$

Expanding the square, the right hand side becomes

$$\begin{aligned}
&\frac{1}{n}\sum_i \left(\left\|\mathbf{x}_j - \boldsymbol{\mu}\right\|^2 + \left\|\mathbf{x}_i - \boldsymbol{\mu}\right\|^2 - 2(\mathbf{x}_j - \boldsymbol{\mu})^T(\mathbf{x}_i - \boldsymbol{\mu})\right) \\
=&\left\|\mathbf{x}_j - \boldsymbol{\mu}\right\|^2 + \frac{1}{n}\sum_i \left\|\mathbf{x}_i - \boldsymbol{\mu}\right\|^2 - 2(\mathbf{x}_j - \boldsymbol{\mu})^T(\boldsymbol{\mu} - \boldsymbol{\mu}) \\
=&\left\|\mathbf{x}_j - \boldsymbol{\mu}\right\|^2 + \frac{1}{n}\sum_i \left\|\mathbf{x}_i - \boldsymbol{\mu}\right\|^2
\end{aligned}$$

Plugging this into the above inequality establishes the claim. ■

REFERENCES

[1] J. Bohannon, "Game-Miners Grapple With Massive Data," *Science*, vol. 330, no. 6000, 2010.

[2] C. Thompson, "Halo 3: How Microsoft Labs Invented a New Science of Play," *Wired Magazin*, vol. 15, no. 9, 2007.

[3] M. Seif El-Nasr, A. Drachen, and A. Canossa, Eds., *Game Analytics – Maximizing the Value of Player Data*. Springer, 2013.

[4] T. Davenport and J. Harris, *Competing on Analytics: The New Science of Winning*. Harvard Business Review Press, 2007.

[5] T. Fields and B. Cotton, *Social Game Design: Monetization Methods and Mechanics*. Morgan Kaufmann, 2011.

[6] E. Seufert, *Freemium Economics: Leveraging Analytics and User Segmentation to Drive Revenue*. Morgan Kauffman, 2014.

[7] G. Zoeller, "Game Development Telemetry in Production," in *Game Analytics – Maximizing the Value of Player Data*, M. Seif El-Nasr, A. Drachen, and A. Canossa, Eds. Springer, 2013.

[8] Y.-T. Lee, K.-T. Chen, Y.-M. Cheng, and C.-L. Lei, "World of Warcraft Avatar History Dataset," in *Proc. Multimedia Systems*, 2001.

[9] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[10] D. Donoho and J. Tanner, "Neighborliness of Randomly Projected Simplices in High Dimensions," *PNAS*, vol. 102, no. 27, 2005.

[11] P. Hall, J. Marron, and A. Neeman, "Geometric Reprsentations of High Dimension, Low Sample Size Data," *J. Royal Statistical Society B*, vol. 67, no. 3, 2005.

[12] F. Murtagh, "The Remarkable Simplicity of Very High Dimensional Data: Applications of Model-based Clustering," *J. of Classification*, vol. 26, no. 3, 2009.

[13] R. Thawonmas and K. Iizuka, "Visualization of Online-game Players Based on Their Action Behaviors," *Int. J. of Computer Games Technology*, vol. 2008, 2008.

[14] A. Drachen, A. Canossa, and G. Yannakakis, "Player Modeling using Self-Organization in Tomb Raider: Underworld," in *Proc. CIG*, 2009.

[15] K. Shim and J. Srivastava, "Behavioral Profiles of Character Types in EverQuest II," in *Proc. CIG*, 2010.

[16] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau, "Guns, Swords, and Data: Clustering of Player Behavior in Computer Games in the Wild," in *Proc. CIG*, 2012.

[17] A. Drachen, C. Thurau, R. Sifa, and C. Bauckhage, "A Comparison of Methods for Player Clustering via Behavioral Telemetry," in *Proc. FDG*, 2013.

[18] C. Thurau and C. Bauckhage, "Analyzing the Evolution of Social Groups in World of Warcraft," in *Proc. CIG*, 2010.

[19] C. Bauckhage, K. Kersting, R. Sifa, C. Thurau, A. Drachen, and A. Canossa, "How Players Lose Interest in Playing a Game: An Empirical Study Based on Distributions of Total Playing Times," in *Proc. CIG*, 2012.

[20] R. Sifa, A. Drachen, and C. Bauckhage, "Behavior Evolution in Tomb Raider Underworld," in *Proc. CIG*, 2013.

[21] O. Missura and T. Gärtner, "Player Modeling for Intelligent Difficulty Adjustment," in *Proc. Discovery Science*, 2009.

[22] B. Weber and M. Mateas, "A Data Mining Approach to Strategy Prediction," in *Proc. CIG*, 2009.

[23] G. Yannakakis and J. Hallam, "Real-time game adaptation for optimizing player satisfaction," *IEEE Trans. Computational Intelligence and AI in Games*, vol. 1, no. 2, 2009.

[24] G. Yannakakis, "Game AI Revisited," in *Prof. Conf. on Computing Frontiers*, 2012.

[25] A. Drachen, C. Thurau, J. Togelius, G. Yannakakis, and C. Bauckhage, "Game Data Mining," in *Game Analytics – Maximizing the Value of Player Data*, M. Seif El-Nasr, A. Drachen, and A. Canossa, Eds. Springer, 2013.

[26] H. Munoz-Avila, C. Bauckhage, M. Bida, C. Bates Congdon, and G. Kendall, "Learning and Game AI," in *Artificial and Computational Intelligence in Games*, S. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2013.

[27] C. Aggarwal and C. Reddy, Eds., *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC, 2013.

[28] V. Estivill-Castro, "Why So Many Clustering Algorithms: A Position Paper," *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 1, 2002.

[29] M. Ester, H.-P.Kriegel, J. Sander, and X. Xu, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proc. KDD*, 1996.

[30] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, 1995.

[31] C. Bauckhage and K. Kersting, "Efficient Information Theoretic Clustering on Discrete Lattices," in *Proc. KDML-LWA*, 2012.

[32] K. Isbister and N. Schaffer, *Game Usability*. Morgan Kauffman, 2008.

[33] C. Thurau, K. Kersting, and C. Bauckhage, "Yes We Can – Simplex Volume Maximization for Descriptive Web-Scale Matrix Factorization," in *Proc. CIKM*, 2010.

[34] C. Thurau, K. Kersting, M. Wahabzada, and C. Bauckhage, "Convex Non-negative Matrix Factorization for Massive Datasets," *Knowledge and Information Systems*, vol. 29, no. 2, 2011.

[35] C. Thurau, K. Kersting, and C. Bauckhage, "Deterministic CUR for Improved Large-Scale Data Analysis: An Empirical Study," in *Proc. SDM*, 2012.

[36] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[37] D. Aloise, A. Deshapande, P. Hansen, and P. Popat, "NP-Hardness of Euclidean Sum-of-Squares Clustering," *Machine Learning*, vol. 75, no. 2, 2009.

[38] P. Bradley and U. Fayyad, "Refining Initial Points for K-Means Clustering," in *Proc. ICML*, 1998.

[39] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," in *Proc. SODA*, 2007.

[40] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm." *J. Royal Statistical Society B*, vol. 39, no. 1, 1977.

[41] J. Shaw-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[42] A. Cutler and L. Breiman, "Archetypal Analysis," *Technometrics*, vol. 36, no. 4, 1994.

[43] D. Lee and S. Seung, "Learning the Parts of Objects by Non-Negative Matrix Factorization," *Nature*, vol. 401, no. 6755, 1999.

[44] D. Donoho and V. Stodden, "When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts?" in *Proc. NIPS*, 2003.

[45] R. Sifa and C. Bauckhage, "Archetypical Motion: Supervised Game Behavior Learning with Archetypal Analysis," in *Proc. CIG*, 2013.

[46] C. Bauckhage and C. Thurau, "Towards a Fair 'n Square Aimbot – Using Mixtures of Experts to Learn Context Aware Weapon Handling," in *Proc. GAME-ON*, 2004.

[47] B. Gorman, M. Fredriksson, and M. Humphrys, "The QASE API - An Integrated Platform for AI Research and Education Through First-Person Computer Games," *Int. J. of Intelligent Games and Simulations*, vol. 4, no. 2, 2007.

[48] M. Fiedler, "A Property of Eigenvectors of Nonnegative Symmetric Matrices and its Application to Graph Theory," *Czechoslovak Mathematical J.*, vol. 25, no. 4, 1975.

[49] A. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," in *Proc. NIPS*, 2001.

[50] I. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, Spectral Clustering and Normalized Cuts," in *Proc. KDD*, 2004.

[51] U. von Luxburg, "A Tutorial on Spectral Clustering," *arXiv*, 2007.

[52] K. Rohe, S. Chatterjee, and B. Yu, "Spectral Clustering and the High-dimensional Stochastic Block Model," *The Annals of Statistics*, vol. 39, no. 4, 2011.

[53] Weka, www.cs.waikato.ac.nz/ml/weka/.

[54] RapidMiner, rapidminer.com/.

[55] Knime, www.knime.org.

[56] SciPy, www.scipy.org.

Christian Bauckhage is professor of media informatics and pattern recognition at the University of Bonn and lead scientists for media engineering at Fraunhofer IAIS in Bonn, Germany. Previously, he worked at the Centre for Vision Research in Toronto, Canada, and was a senior research scientist at Deutsche Telekom Laboratories in Berlin. His research focuses on descriptive data mining and machine learning for multimedia and computer games. He regularly publishes conference papers and journal articles and frequently serves on program committees and editorial boards; among others, Prof. Bauckhage is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES.

Anders Drachen is currently an Assistant Professor at Aalborg University (Denmark) and Lead Game Analyst for Game Analytics (Copenhagen/Berlin). His work is focused on game analytics, game user research, business intelligence for games, game data mining, behavioral modeling, virtual economics, game user experience, and business development. He writes about analytics for game development on blog.gameanalytics.com, and about game- and data science in general on andersdrachen.com. His contributions can also be found on the pages of trade publications such as Game Developer Magazine and Gamasutra.com and in the book “Game Analytics – Maximizing the Value of Player Data”.

Rafet Sifa received his M.Sc. degree in Computer Science from the University of Bonn in 2013. He is currently a Ph.D. student at the University of Bonn and a data scientist at Fraunhofer IAIS. His research focuses an statistical approaches to game data mining for decision making and game recommender systems.

LIST OF FIGURES

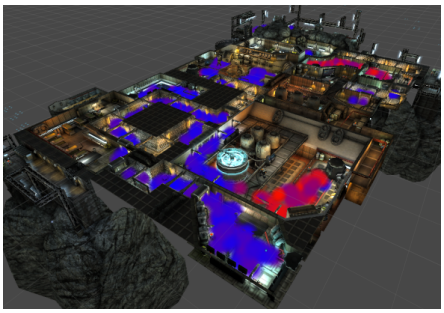
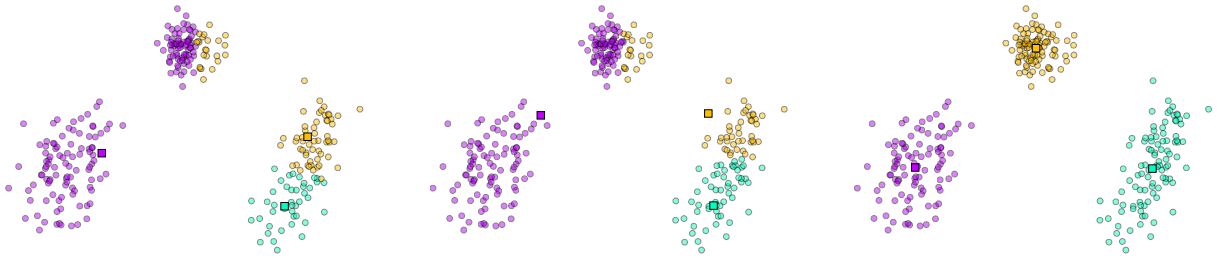


Fig. 1

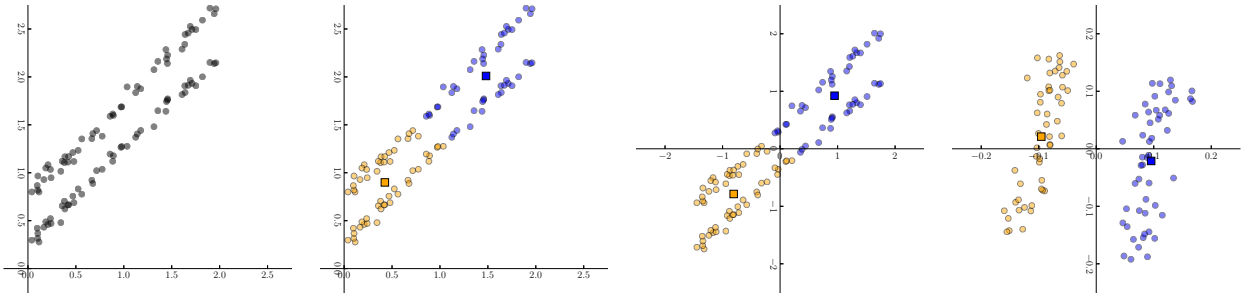
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games



(a) (b) (c)

Fig. 2

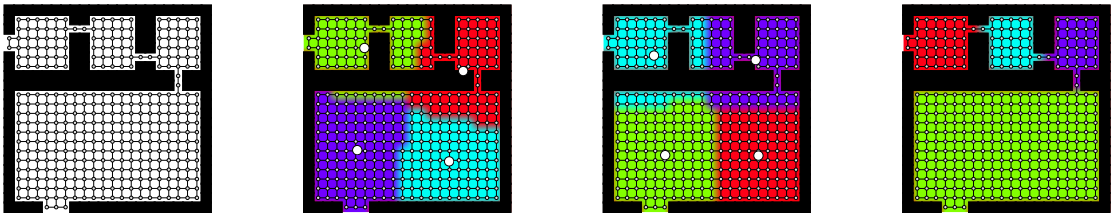
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games



(a) (b) (c) (d)

Fig. 3

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games



(a) (b) (c) (d)

Fig. 4

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games

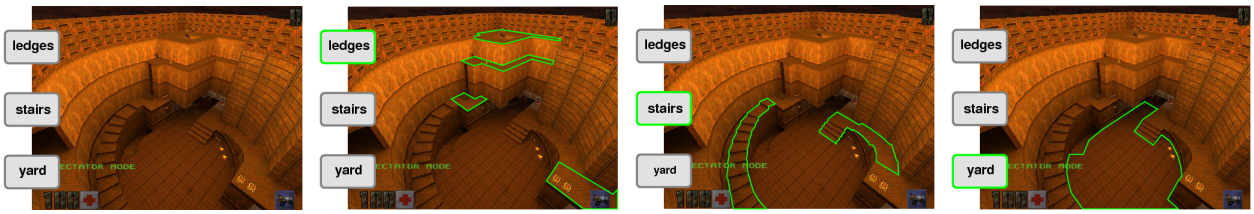


Fig. 5

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games

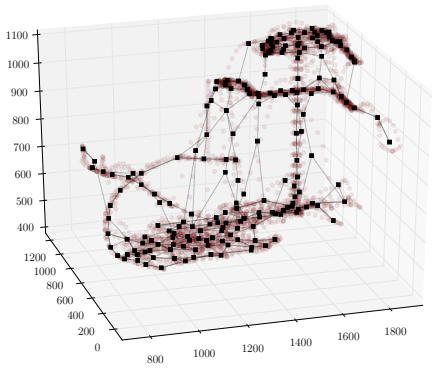
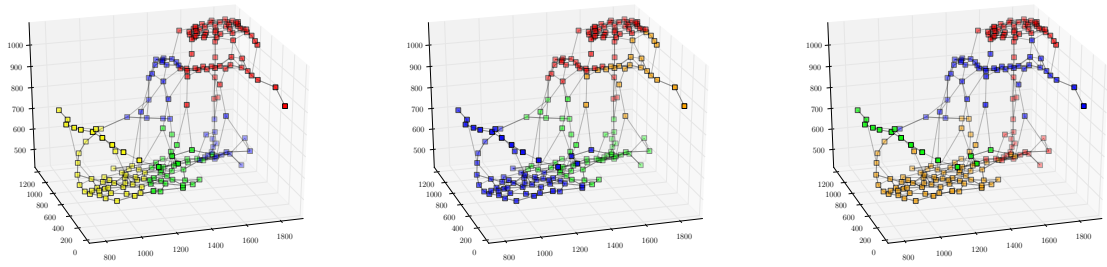


Fig. 6

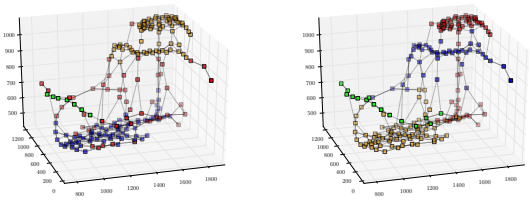
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games



(a) (b) (c)

Fig. 7

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games



(a) (b)

Fig. 8

List of Tables

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games

TABLE I

cluster designation	% players	characteristics
Elite	3.9	high scores except for mining, planting, or deaths; no auctions created
Stragglers	7.6	low scores overall; die a lot
Planters	21.6	medium scores; high planting skills
Miners	15.0	medium scores; high mining skills
Auction Devils	1.1	highest auction and achievement scores; 2nd ranked loot and kills scores; 2nd ranked friend scores; high mining score
Friendly Pros	50.8	highest friend scores; from low auction scores; 2nd lowest loot scores

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2014.2376982, IEEE Transactions on Computational Intelligence and AI in Games

TABLE II

cluster designation	% players	characteristics
Elite	5.8	high scores except for mining and planting
Stragglers	39.4	low scores overall; die a lot
Average Joes	12.7	better scores across all categories than Stragglers; 4th ranked overall
Dependables	18.6	average scores across all categories; high number of friends; 3rd ranked overall; 2nd rank in monster kills
Worker I	15.9	similar to the Average Joes, but high mining and plating scores; 3rd ranked loot scores
Worker II	7.6	similar to The Dependables, but highest mining and planting scores; 2nd ranked overall; 2nd ranked loot scores