Abstract—Extracting behavioural prototypes plays an important role in player profiling. Understanding the type of players present in the game goes alongside improving game-play experience as well as player engagement over time. In this paper, we introduce the application of Kernel Minimum Enclosing Balls (KMEBs) as a tool to extract meaningful extreme prototypes in games and present an example use-case analyzing a behavioural dataset from a Massively Multiplayer Online Role Playing Game. Unlike the majority of the methods covered in this context, our approach allows for modelling nominal and numerical behavioural features, extending the scope and capability of the profiling methods as well as improving the interpretability of the results.

I. INTRODUCTION

Prototype selection is a diverse field allowing for many applications: it can be used to condense the training data when further computations are expensive, to provide hidden patterns to human analysts, to explain the versatile results of classification or clustering routines [1]. The main goals of Game Analytics research include building models to understand the game dynamics and improve the game balance. Moreover, common playability problems occur if playability heuristics are violated [2]. For instance, if the difficulty ramps up too quickly, game items wear out too fast or if the game does not support different playing styles, it poses an obstacle in the game process. In this case, profiling the behaviour of players would show the overall picture, namely, whether it is possible to reach the highest level of the game using a different strategy, or if the game is bound to the usage of only one factor. Nevertheless, the interpretability of the results and samples being “representative” of the data remains the key point of the underlying research [3]. Then providing a tool for extracting a small number of prototypes (so that domain specialists can understand the data) stays an acute research direction nowadays.

The method of Kernel Minimum Enclosing Balls (KMEBs), introduced in the context of prototyping in [4], is simple and easy to implement. It has a better run-time than commonly used in behavioural profiling in digital games Archetypal Analysis method, as can be seen on the example of multi-player shooter Destiny in [5] and UltimaIV in [6]. Archetypal Analysis extracts extreme behaviour basis vectors that are easy to interpret [1]. Opposed to the basis vectors, KMEBs offer the concept of support vectors, that are the data points corresponding to the non-zero values of an optimized vector.

Having a number of hyper-parameters that affect the resulting prototypes, KMEBs allow us to introduce variation into algorithm output, paving the way to the ensemble approach. One of such parameters is the kernel, which is a powerful tool given prior knowledge about the data. Depending on the data type, be it categorical, mixed or continuous data, there exist numerous task-specific kernels for each of the described cases. All the above-mentioned advantages make KMEBs an interesting research objective in terms of analyzing the real-world player data.

II. CONTRIBUTION

In this paper, we provide an ensemble of well-known Distance Substitution kernels within the context of prototype extraction using KMEBs. We augment the Divide and Conquer algorithm from [7] to return a fixed small number of prototypes (support vectors), prepared for analysis by a human analyst, and provide an ensemble approach, calculating the frequency of extracted prototypes over a variety of distance-kernel pairs. Moreover, we conclude that the kernel matrix in KMEBs is not required to be positive definite to reach substantial results on the example of this study’s data.

III. KERNELS IN KERNEL MINIMUM ENCLOSING BALLS PROBLEM

Recall the kernel MEB problem in terms of minimization, derived in [4] and solved via the Frank-Wolfe or the Projected Gradient Descent algorithm:

$$\mu_* = \arg\min_{\mu} \mu^T K \mu - \mu^T k$$

s.t. $\mu \in \Delta^{n-1}$,

where $K \in \mathbb{R}^{n \times n}$ is a kernel matrix computed on input dataset, $k$ contains its diagonal, $\Delta^{n-1}$ is the standard $(n-1)$-simplex, $\mu \in \mathbb{R}^n$ is the Lagrange multipliers, $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is an appropriate Mercer kernel.
The data points of data matrix $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{m \times n}$ corresponding to the non-zero values of Lagrange multiplier $\mu$ are called support vectors. Support vectors computed by KMEBs showed themselves finding central as well as varying prototypes of the data within the example of an online multiplayer shooter game [4], thus we use support vectors as prototypes for player data.

The choice of the kernel $\mathbf{K}$ is a way to inject prior knowledge about the data into the model. For instance, choosing the special distance measure (a city block or a hamming distance) for categorical data reflects its structure. One of the most popular kernel choices is the Distance Substitution (DS) kernel using the hamming distance.

We can show it as $k_0$ kernel using the hamming distance.

$$k_{0(i,j)} = f_{DS}(\frac{1}{n} \sum_{k=1}^{n} f_{a}([x_{ik} = x_{jk}])) = f_{DS}(s_{hamming}) = f_{DS}(1 - d_{hamming}).$$

### IV. Dataset

The telemetry data is provided from a massively multiplayer online role-playing game TERA, used for behaviour clustering in [6]. TERA has typical MMORPG features such as questing, crafting, and player versus player action. It also enables in-game purchases. Characters may be of different race and different class, whereas each race has a set of unique skills and class induces class-specific skills and abilities. Moreover, certain set of skills, i.e. Mining, Planting, Auctions are not essential to be completed in the game course, being optional resource gathering skills the player can choose to develop.

The data comprises of 10 features, that are Level (level of character in the game), Friends (number of friends in the game), Quests Completed (number of quests completed), Achievements (number of achievements earned), Mining and Plants (level in skill respectively), Kills Monsters (number of AI-controlled enemies killed by character), Loot Total Items (total number of items that character picked up during the game), Death Monsters (the number of times the character had been killed by AI-controlled enemies) and Auctions (number of times that character created or purchased something from an auction).

We extract the prototypes based on level 32 (the highest level in the dataset) to understand the behavioural patterns of players that have achieved considerable results. The final data results in 3817 characters. For the analysis we use nine above-mentioned gameplay features (related to how characters are representing the players) of times that character created or purchased something from the game), Death Monsters (the number of times the character had been killed by AI-controlled enemies), Loot Total Items (total number of items that character picked up during the game), and Auctions (number of times that character created or purchased something from an auction).

The data was pre-processed using Min Max scaler, to avoid the problems of mixing of data types, as advised in [6]:

$$F' = \frac{F - \min F}{\max F - \min F}.$$
In this table we observe the number of support vectors extracted by Divide and Conquer algorithm for each respective kernel and distance. As the number of support vectors is not monotone, it poses a challenge to find a corresponding hyperparameter and extract the exact number of prototypes. Hence, we provide the approximation of the algorithm, that returns the ‘closest’ biggest and smallest number of support vectors. We note, that for the case of Hamming distance the minimum number of support vectors is 19 for all hyperparameter interval, hence, it is not possible to run hyperparameter optimization at that setting. The values marked by star are not included in the ensemble as, by our empirical observation, the greater the difference in support vectors number, the more different would extracted prototypes prototypes be.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>sq_euclidean</th>
<th>euclidean</th>
<th>minkowski (p=4)</th>
<th>city block</th>
<th>cos</th>
<th>hamming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Distance</td>
<td>5/8</td>
<td>4/7/8*</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Laplace</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6/7</td>
</tr>
<tr>
<td>Inverse Multiquadric</td>
<td>6</td>
<td>6</td>
<td>2/7/7</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Rational Quadratic</td>
<td>6</td>
<td>5/7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6/6</td>
</tr>
</tbody>
</table>

TABLE III

Concise overview of the selected distances used in ensemble.
For Hamming distance \(|x_i = y_i| = 1\) if \(x_i = y_i\), 0 if \(x_i \neq y_i\)

for each feature \(F\) of the dataset. We note that before using KMEBs we do not use bins for ranges of categorical values, rather use the data only processed by Min Max scaler to map intervals to \([0, 1]\). It shows the ability of the algorithm to process the categorical data even given the different number of unique values in each feature.

Categorical variables are frequently handled using one-of-k encoding or one-hot-encoding and then use of discrete kernels. However, as the number of values that is possible to take is over 4 thousand, it becomes computationally inefficient when it comes to distance calculation, as the dimensionality of data rises.

V. Ensemble approach

For Ensemble approach we set up a goal for every distance-kernel pair to extract \(p^* = 6\) prototypes. It makes a total of 30 distance-kernel pairs with 5 kernels (Table I) and 6 distances (Table II). We define Kernel Minimum Enclosing Balls training with \(t_{max} = 100\) iterations of Frank-Wolfe on the column data matrix \(X = [x_1, \ldots, x_n] \in \mathbb{R}^{m \times n}, n = 3817\) as for the number of training points and \(m = 9\) the number of features described in Table III.

Controlling the number of support vectors is possible using the kernel parameters: \(\sigma\) for Laplace and Gaussian kernel, \(c\) for Rational Quadratic and Inverse Multiquadric, and finally \(\beta\) for Distance kernel. Considering the non-monotonous dependency of support vectors number on the respective parameter, we apply an algorithm called Divide and Conquer [7] that allows us to control the number of support vectors in the resulting solution of KMEBs. Namely given the fixed number \(p^*\) of support vectors, it attempts to find such parameter \(\chi\) that for fixed number of iterations inside of Frank-Wolfe algorithm \(t_{max}\) and fixed distance function \(d^*\) for DS-kernel, \(p(\chi, t_{max}, d^*)\) holds. The algorithm returns the value of parameter \(\chi\) and corresponding support vector matrix \(S_{m \times p^*}\), where support vector matrix is a matrix of datapoints that correspond to the non-zero values of the Lagrange multiplier. In case of the algorithm’s non-convergence, it provides the support vector matrix such that the number of support vectors in it is the closest to 6. The example of it can be seen at Table III showing the number of support vectors extracted for each distance-kernel pair.

The Divide and Conquer approach [7] finds the parameter values that correspond to the highest and lowest support vector numbers in the initial parameter interval and performs a bisection. We choose for next step the left half-interval if \(|p(t_{max}, \sigma_{higher}, d^*) - p^*| > |p(t_{max}, \sigma_{lower}, d^*) - p^*|\) and right interval otherwise, where \(\sigma_{lower}\) and \(\sigma_{higher}\) are two values defining the half-interval. Then we check, whether \(p^* \in [p(\sigma_{higher}), p(\sigma_{lower})]\). If it does not, then the algorithm returns one step back and chooses the other side of the interval. After that, the current half-interval is divided into \(g = 10\) equal parts and this division is passed as an interval to the next iteration.

Based on the previous step, we calculate the frequency of each appeared support vector for the kernel-distance pairs that returned exactly with weight \(w = 1\) and \(w = 0.5\) to both of the approximate solutions, in order to account for both returned support vector matrices. Then the frequency of the appearance of each support vector is calculated as \(\nu(s \in S_i) = \sum_{l=1}^{\#} w(l(S_i))\), where \(w(l(S_i)) = 0.5\), \(w(l(S_i)) = 6\), \(l\) is number of support vectors in the support vector matrix \(S_{m \times l}\), \(h\) is the number of resulting distance-kernel support matrices with \(h = 29\) according to Table III.

VI. Observations

The resulting classes obtained from the paper [6] using Simplex Volume Maximization (SVM) are Planters, Miners (that have respectively average scores across the performance features, but very high Planting or Mining skill respectively), Auction Devils (focused on using the Auction house feature of
the game, gaining Achievements, Loot, strong social networks and high Mining skills). Friendly Pros (similar to the Auction Devils, but exhibit low Auction and Loot scores, and otherwise strong scores in the performance features), Elite (high scores overall, except for Mining/Plants and deaths from monsters with no Auctions created) and finally Stragglers with low scores overall and a high number from Death from Monsters.

Indeed, reproducing the results of SiVM is illustrated in Table V. The first prototype is clearly distinguished as the Auction Devil, second and fifth prototypes as respectively Planter and Miner, majoring in the respective skill and having the other skills averaged. The third prototype is exposing the Straggler behaviour but rather according to the k-means definition of the class with the death of monsters peaking. The fourth prototype is identified as Elite with a high number of Quests completed. The sixth prototype exhibits the behaviour of Friendly Pro with the high number of Friends. Results in Table VI obtained from the ensemble show the similar narrative with two prototypes matching (highlighted in bold). Prototype number five exposes the behavior of Friendly Pro, yet due to high Mining skill, it can be posed as a Miner. With more extreme Straggler prototype four, the ensemble also returns Central prototype six acknowledging not only for the extreme data. The Planter at place seven exhibits high Mining revealing the complementary skills in the player’s behavior.

### VII. Conclusion

In this work, we have presented an ensemble approach for extracting prototypes from heterogeneous datasets with nominal and numerical behavioural data. Our approach showed itself successful with respect to determining all the player classes mentioned in [6] and moreover, provided us with more central prototypes. The ensemble showed another outlook on the data, exposing new dependencies and complementing the data analysis. At last, we have discussed the importance of Distance Substitution Kernels in terms of using the hamming distance and proved that for KMEBs the use of positive definite, or even conditional positive definite kernels is not necessary to acquire tangible results.

### REFERENCES


