El Clustering de Jugadores de Tetris

Diana Sofía Lora Ariza

Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid, Spain
dlora@ucm.es

Abstract. El análisis del comportamiento en el contexto de los videojuegos se ha convertido en una práctica muy popular debido permite obtener información vital de los jugadores. En modelos de negocio como los videojuegos Free-to-Play (F2P) esta información es importante para aumentar el interés de los jugadores/clientes en el videojuego, la cantidad de jugadores y las ganancias obtenidas. De igual forma, también permite realizar mejoras en el diseño. En este trabajo se realiza análisis de comportamiento de trazas del videojuego TetrisAnalytics. En éstas se encuentran los movimientos tácticos del jugador durante la partida, a las que se aplican los métodos de clustering SimpleKMeans y EM(Expectation-Maximization) vía Weka.

1 Introducción

Los videojuegos se han convertido en sofisticados sistemas de información. A través de éstos se obtiene gran cantidad de datos por medio de la interacción que tiene el usuario con el videojuego [7]. La telemetría y las métricas de videojuegos han demostrado ser indispensables para la medición de diferentes aspectos del rendimiento de los sistemas, la finalización exitosa de los proyectos y el análisis de sus jugadores [11]. Existe una gran cantidad de información escondida en conjuntos de datos de comportamiento. Sin embargo, para obtener esa información es necesario aplicar diferentes técnicas que permitan extraer patrones de los datos [7]. Por medio de técnicas de Aprendizaje No Supervisado, como clustering, es posible extraer patrones de comportamiento para la toma de decisiones de diseño y refinar la estrategia de ganancias utilizada. La aplicación de técnicas de inteligencia artificial a conjuntos de datos de comportamiento de jugadores es reciente en comparación con otros campos de aplicación [5].

En este artículo, utilizando la herramienta de minería de datos Weka se aplican pruebas sobre conjuntos de datos de comportamiento con el fin de clasificar los diferentes jugadores de dicha muestra. Se aplican los algoritmos *K-means* y *EM*. Los métodos de clustering, y particularmente el algoritmo k-means, es uno de los más populares utilizados para la evaluación de bases de datos de comportamiento en el contexto de los videojuegos. Este análisis se realiza con la finalidad de verificar la eficiencia de la clasificación de los diferentes jugadores en juegos con bajo rango de movimientos disponibles. Asimismo, la fase inicial para el ajuste dinámico de la dificultad en videojuegos es el desarrollo de un

modelo de comportamiento que posteriormente será utilizado para relacionar las habilidades de los jugadores con la dificultad estimada para este.

En la sección 2 de este artículo, se describe la importancia del análisis del comportamiento de los jugadores en videojuegos, una de las técnicas de Aprendizaje No Supervisado más utilizadas y la herramienta de minería de datos empleada para realizar los experimentos. La sección 3 contiene el diseño del videojuego, las variables y los algoritmos utilizados. En la sección 4 se presentan los resultados obtenidos. Y por último, en la sección 4 se comentan las conclusiones y trabajo futuro.

2 Análisis del Comportamiento de Jugadores

El análisis del comportamiento en videojuegos ha pasado a ser una práctica ampliamente utilizada debido a que esta provee información vital sobre la población de jugadores. En modelos de negocio como videojuegos Free-to-Play (F2P) esta información permite conocer mejor los clientes/jugadores con el fin de aumentar los ingresos [5, 7]. El análisis del comportamiento de los usuarios permite conocer las características más atractivas del juego, los artículos más utilizados e incluso, la forma en que los usuarios interactúan entre sí. Al conocer esta información, es posible realizar mejoras en el diseño de los elementos más utilizados y así mantener el interés de los jugadores e incluso atraer más.

De igual forma, la clasificación de los jugadores a partir de la interacción que estos tienen con el juego permite conocer el nivel de habilidad que cada uno. La finalidad de un juego es que sus jugadores se diviertan. Esta se deriva del dominio de las habilidades y la satisfacción de cumplir los retos propuestos. Después de dominar las habilidades, el juego da una recompensa al jugador por su buen trabajo y crea nuevos objetivos por alcanzar. Así se crea un ciclo donde el jugador se encuentra constantemente aprendiendo, dominando habilidades y obteniendo recompensas, lo que lo mantiene interesado [1]. El ciclo de aprendizaje-dominio-recompensa debe estar en el nivel adecuado; así, el juego no es lo suficientemente fácil para ser aburrido o demasiado difícil para ser frustrante. Ambos casos llevan al mismo resultado, donde el usuario abandona el juego. Por medio del ajuste dinámico de la dificultad, es posible modular cómo el juego responde a las habilidades particulares de sus jugadores durante la sesión de juego. La dificultad es considerada un factor subjetivo que se deriva de la interacción entre el jugador y el reto propuesto. Además, esta no es estática, ya que entre mayor dominio tiene el jugador sobre una habilidad concreta, más difícil se convierten las tareas a realizar [9, 12].

La clasificación de jugadores basándose en su comportamiento es parte de una línea de investigación que se centra en el desarrollo de juegos personalizables [7, 17]. Esta permite realizar un análisis completo de las bases de datos de comportamiento y obtener resultados a través de los diferentes perfiles. Con los resultados es posible realizar mejoras en el diseño del juego y en las estrategias de monetización utilizadas [3, 10]. Con técnicas de Aprendizaje No Supervisado, como clustering, es posible disminuir la dimensionalidad de un conjunto de datos

y obtener aquellas características más relevantes entre los jugadores. En el caso en que se desee clasificar el comportamiento de los jugadores, se usa clustering si se desconoce cómo varía su comportamiento o si no existen clases definidas [7].

El *Clustering* es uno de los métodos de Aprendizaje No Supervisado más utilizados en diferentes campos de aplicación. Esta técnica ha sido utilizada en el análisis de los videojuegos como una forma de encontrar patrones en el comportamiento de los jugadores, la realización de comparativas entre videojuegos(bechmark), la evaluación del desempeño del jugador, y el diseño y entrenamiento de personajes no controlados. Los algoritmos existentes para la asignación de los objetos a un cluster se clasifican en parcial(*"soft"*) o total(*"hard"*). Un algoritmo es de asignación total, cuando el objeto es identificado completamente por el cluster. Análogamente, un algoritmo es considerado de asignación parcial, cuando un objeto puede pertenecer a diferentes cluster por tener cierto grado de semejanza [2].

En las categorías principales de los métodos de clustering se encuentran los *métodos de particionamiento*, también conocido como clustering de centroides [6, 8]. Estos métodos construyen grupos que cumplen dos requerimientos: cada grupo debe contener por lo menos un objeto y cada objeto solo puede pertenecer a un grupo. Los métodos de particionamiento empiezan realizando una división inicial entre los datos. Luego utiliza una técnica iterativa de re-ubicación de objetos, donde intercambian objetos de un grupo a otro con el fin de validar la semejanza de los elementos pertenecientes al mismo cluster. El criterio utilizado en el algoritmo k-means es uno de los más populares. En este cada cluster es representado por el valor medio de los objetos en dicho cluster [8].

Weka es una herramienta sencilla de usar con herramientas de Minería de Datos y Aprendizaje Automático [15,16]. Esta herramienta ofrece diferentes algoritmos para la transformación de bases de datos y su pre-procesamiento, la ejecución de técnicas de minería de datos y el análisis del resultado. Entre los algoritmos implementados para clustering se encuentra k-Means y EM. El algoritmo EM es una adaptación de k-means, excepto que este calcula la probabilidad que tiene cada instancia de pertenecer a un cluster. En clustering, EM genera una descripción de probabilidad de los clusters en términos de la media y la desviación estándar para los atributos numéricos y realiza un conteo para los valores nominales [14].

Los métodos de clustering han sido utilizados para la categorización del comportamiento de los jugadores en diferentes casos de estudio. Missura y Gärtner [12] utilizan k-means y máquinas de vectores de soporte para predecir el ajuste dinámico de la dificultad en un juego sencillo de disparar a naves extraterrestres. Este caso de estudio se enfoca en la construcción de un modelo de dificultad encargado de agrupar los diferentes tipos de jugadores en principiante, promedio y experto; encontrar el ajuste de dificultad asociado a cada grupo y utilizar estos modelos para realizar predicciones a partir de pocas trazas del juego. Por otro lado, Drachen et al. [4] aplican los algoritmos k-means y SIVM (Simplex Volume Maximization) a un conjunto de datos de 260,000 jugadores de MMORPG *Tera Online* y el multijugador FPS *Battlefield 2: Bad Company 2.*

Utilizando la técnica de Drachen et al. [3], desarrollan perfiles de los jugadores a partir de los algoritmos mencionados. Con k-means obtienen los perfiles más comunes entre los jugadores, y con Archetypal Analysis vía SIVM, obtienen los comportamientos extremos. Otro caso de estudio similar es el realizado por Drachen et al. [5] con el popular MMORPG *World of Warcraft*.

La mayoría de trabajos académicos realizados actualmente se centran en un banco de pruebas pequeños con datos de comportamiento obtenidos a través de telemetría. En la investigación académica, es complejo realizar este tipo de pruebas debido a la falta de disponibilidad de bases de datos grandes. A pesar que existen algunos estudios de juegos comerciales, esto continúa siendo escaso. Sin embargo, la colaboración entre estos dos sectores(empresas e instituciones de investigación) ha mejorado [5].

3 Configuración Experimental

Para realizar la clasificación de comportamiento de los jugadores se utiliza Tetris-Analytics. La implementación de este juego de Tetris es sencilla. Se tiene 7 diferentes piezas y al pasar el tiempo, la ficha cae más rápido por el tablero. El jugador puede realizar los movimientos tradicionales(izquierda, derecha, rotación y bajar la ficha). El juego sólo finaliza cuando el jugador pierde; es decir, cuando las fichas alcanzan la parte superior del tablero.

Las variables obtenidas de cada partida son las siguientes:

- **PosX:** posición X donde el jugador ubica la pieza. Esta variable es una enumeración con los siguientes valores: x0, x1, x2, x3, x4, x5, x6, x7, x8, x9.
- **PosY:** posición Y donde el jugador ubica la pieza. Esta variable es una enumeración con los siguientes valores: y0, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, y16, y17, y18, y19.
- **Rot:** rotación de la ficha. Esta variable es una enumeración con valores: r0, r1, r2, r3.
- **NextPiece:** pieza siguiente que saldrá a continuación(valor entero entre 0 y 6).
- **Celda_X_Y (X e Y enteros desde 0 hasta el ancho/alto - 1):** estado de cada celda concreta del tablero. Valores 0 o 1 dependiendo de si está vacía u ocupada respectivamente.
- **TipoFicha:** tipo de ficha. Enumerado p0, p1, p2, p3, p4, p5, p6.

Al finalizar la partida, se guarda un archivo binario con la información completa de la partida. A partir de este binario se pueden generar archivos ARFF que posteriormente serán ingresados en Weka para su análisis. Las políticas de generación de archivos ARFF utilizadas son:

- **PosYTipoFichaARFF:** guarda todos los atributos de la política anterior mas el tipo de ficha actual.
- **LastXLinesARFF:** guarda la información de las fichas (posición, rotación, tipo de ficha y ficha siguiente). Al igual que la parte superior del tablero; es

decir, las últimas filas no vacías. La "X" del nombre puede variar entre 1-4, lo cual indica la cantidad de filas que se guardan. La posición de la ficha se almacena relativa a la Y de la fila superior.
- **EstadoCompletoARFF:** guarda todo el estado del juego, esto incluye: la posición de la ficha, rotación y tipo, así como la ficha siguiente y el estado completo del tablero (casilla libre=0, casilla ocupada=1).

Cada instancia contiene el movimiento táctico del jugador; es decir, cada instancia contiene la posición x e y donde el jugador ubica la ficha junto con los demás atributos que incluya la política de generación de archivos.

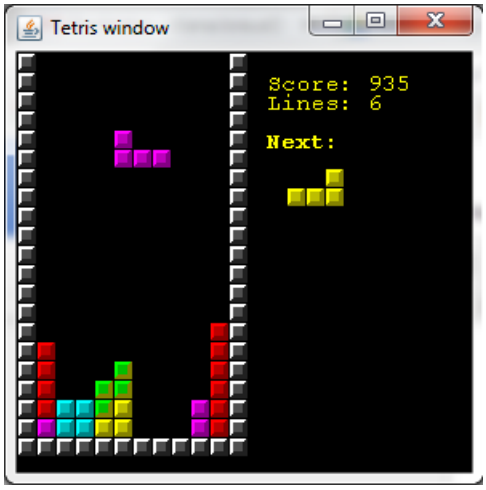

Fig. 1. Pantallazo del juego.

Se efectúa la evaluación clases a clusters con los algoritmos SimpleKMeans y EM para clustering de Weka, donde k=número de jugadores. En este modo, Weka inicialmente ignora la clase y genera los clusters. Posteriormente, en la fase de pruebas, asigna las clases a los clusters basándose en la clase con mayor cantidad de instancias que se encuentren en dicho cluster. Luego calcula la tasa de error de la clasificación, el cual se basa en la asignación de clases a clusters y muestra la matriz de confusión [14]. El algoritmo EM, en modo evaluación de clases a clusters, realiza el cálculo de *likelihood function*, asigna clases a los clusters, muestra la matriz de confusión y la cantidad de instancias incorrectamente clasificadas. De igual forma, el algoritmo SimpleKMeans en modo evaluación de clases a clusters, realiza la asignación de clases a clusters, muestra la matriz de confusión, la tasa de error y la sumatoria del error cuadrado de los clusters.

Las trazas evaluadas corresponden a 6 jugadores diferentes. Tres de estos, son IA's. Dos de las IA's utilizadas son sencillas, su táctica es ubicar la ficha en la parte izquierda o derecha del tablero. La tercera tiene una habilidad alta

para jugar y su comportamiento es similar a los movimientos de los jugadores reales. Esta IA, llamada HardCodedTacticalAI, evalúa el tablero de acuerdo a su altura(fila más alta con alguna posición ocupada) y el número de espacios en este(número de posiciones sin ocupar en una columna por debajo de la primera posición ocupada). Esta evaluación la realiza para cada una de las 4 rotaciones posibles de la pieza y selecciona la mejor ubicación.

4 Resultados

Para el análisis de las trazas de TetrisAnalytics se realizan dos pruebas diferentes con el fin de evaluar la clasificación de los jugadores con respecto a sus movimientos tácticos individuales y además, en una segunda prueba se realiza la unión de las trazas de la jugada actual con la siguiente con el fin de analizar el impacto que tienen la ubicación de las piezas si el jugador conoce cual es la ficha siguiente.

El análisis efectuado para el comportamiento de los jugadores respecto a los movimientos individuales se desarrollaron con 4312 instancias de 6 jugadores. Tres de los jugadores son IA's, una posee buenas habilidades para jugar Tetris(jugador 3). Las otras dos son "tontas" y sus movimientos consisten en poner las fichas siempre en el lado izquierdo o derecho(jugador 5 y jugador 4 respectivamente) del tablero; mientras que los demás jugadores distribuyen las fichas de manera más homogénea a través del tablero. Esto se puede examinar en la figura 2, donde cada color representa un jugador. Por medio de TetrisAnalytics es posible producir archivos ARFF con diferentes conjuntos de variables. Con la política de generación PosYTipoFichaARFF se exporta el tipo de ficha actual, el tipo de ficha siguiente, la posición x y y donde se ubica la ficha, y la rotación que tiene la pieza en su ubicación final. Con Last4LinesARFF y EstadoCompletoARFF, se agrega la información de las 4 filas no vacías superiores del tablero y el estado completo del tablero respectivamente.

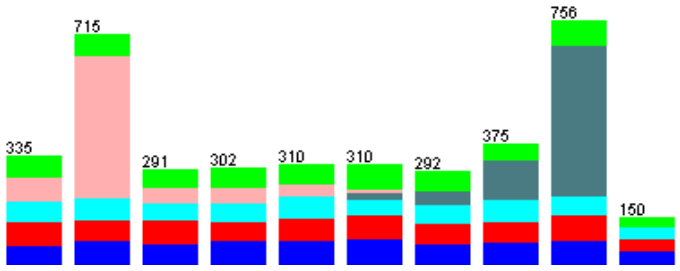

Fig. 2. Característica PosX de ubicación de la ficha. Varia de $x0$ a $x9$. Donde $x0$ es la posición más a la izquierda.

La tasa de error para las pruebas con los movimientos tácticos individuales son las siguientes:

- **PosYTipoFichaARFF:** SimpleKMeans 70.61% (3045 instancias), EM 71.12% (3067 instancias).
- **Last4LinesARFF:** SimpleKMeans 44.15% (1904 instancias), EM 48.46% (2090 instancias).
- **EstadoCompletoARFF:** SimpleKMeans 45.06% (1943 instancias), EM 53.43% (2304 instancias).

Para todos los casos SimpleKMeans clasifica mejor que EM. Asimismo, aunque se esperaba que al tener la información del tablero completo diera mejores resultados que la información parcial de este, la mejor clasificación fue realizada teniendo la información de las 4 filas superiores no vacías del tablero. Debido a que la evaluación es realizada con los movimientos tácticos individuales de los jugadores, es posible que el estado completo del tablero, hasta cierto punto, no sea tan relevante para su clasificación debido a que solo se tiene en cuenta la ubicación de cada ficha, no de un conjunto de fichas consecutivas.

En la matriz de confusión(tabla 1), las columnas representan el número de predicciones del algoritmo y las filas las instancias en la clase real. La mayoría de las instancias de los jugadores 4 y 5 pertenecen a los clusters 0 y 3 respectivamente. Estos dos jugadores, debido a tener comportamientos extremos, su clasificación es bastante acertada. Sin embargo, el resto de jugadores, debido a que realizan diferentes movimientos, pueden llegar a tener comportamientos similares y así ser clasificados erróneamente. Esto ocurre porque el espacio de movimientos posibles permitidos a los jugadores es bastante reducido, así que al evaluar las partidas con movimientos individuales por jugador puede llevar a una mala clasificación.

Table 1. Asignación de clases a clusters con SimpleKMeans, configuración Last4LinesARFF y movimientos individuales.

0(JUG6)	1(JUG2)	2(JUG4)	3(JUG3)	4(JUG1)	5(JUG5)	Clusters Asignados
131	104	28	147	203	56	JUG1
190	204	113	104	99	82	JUG2
29	76	1	371	194	30	JUG3
0	0	725	0	0	34	JUG4
0	0	0	0	4	760	JUG5
145	118	14	168	149	33	JUG6

Debido a que la evaluación de los movimientos de los jugadores de manera individual presentan una tasa de error considerablemente alta, también se realiza la evaluación de las trazas uniendo la instancia de la ficha actual con la siguiente por jugador y partida. Esta unión se realiza porque durante la partida el jugador puede observar la ficha siguiente a ubicar en el tablero. Por lo tanto, se puede

considerar que el jugador ubica la ficha actual de manera diferente si conoce cual será la siguiente. Así que se realiza un análisis de los dos movimientos consecutivos, con el fin de verificar si existe algún patrón entre la ubicación de la ficha actual con base en la ficha siguiente que este tendrá disponible. Por consiguiente, se realizan las pruebas con las tres políticas de generación de archivos ARFF de TetrisAnalytics.

La tasa de error que presentan las pruebas al unir la traza de la pieza actual con la siguiente son las siguientes:

- **PosYTipoFichaARFF:** SimpleKMeans 60.03% (2588 instancias), EM 58.33% (2515 instancias).
- **Last4LinesARFF:** SimpleKMeans 40.78% (1756 instancias), EM 47.08% (2030 instancias).
- **EstadoCompletoARFF:** SimpleKMeans 52.32% (2253 instancias), EM 53.032% (2296 instancias).

En el caso de PosYTipoFichaARFF, se puede observar una disminución importante en el porcentaje de instancias incorrectamente clasificadas para ambos algoritmos. Con la configuración EstadoCompletoARFF, ambos algoritmos alcanzan una tasa de error similar y menor con respecto a la prueba anterior. Last4LinesARFF sigue siendo la mejor clasificada. Para el caso de los jugadores JUG4 y JUG5, los cuales presentan comportamientos extremos, se puede observar una clasificación correcta de todas sus instancias en la tabla de confusión(Tabla 2) resultante de la evaluación con SimpleKMeans.

Table 2. Asignación de clases a clusters con SimpleKMeans, configuración EstadoCompletoARFF y unión de movimientos consecutivos.

0(JUG1)	1(JUG3)	2(JUG6)	3(JUG5)	4(JUG4)	5(JUG2)	Clusters Asignados
176	107	170	40	36	139	JUG1
124	52	139	71	122	283	JUG2
84	398	156	21	0	41	JUG3
0	0	0	0	758	0	JUG4
2	0	0	761	0	0	JUG5
116	134	174	28	12	162	JUG6

Debido a restricciones de espacio, no se muestra la tabla de confusión resultante de la asignación de clases a clusters utilizando el algoritmo EM. Sin embargo, la clasificación realizada por este algoritmo, a pesar de su mejora, no es tan buena como se espera. Los jugadores JUG4 y JUG5 siguen siendo clasificados en diferentes clusters a pesar que su comportamiento es diferente al resto de jugadores evaluados. A pesar de esto, se puede notar una mejora con respecto a las pruebas realizadas con movimientos individuales.

Prueba Estadística

Con el fin de conocer cual de los algoritmos es utilizados es el mejor, se realiza la t-test pareada corregida vía Weka Experimenter. Este método realiza la comparación de los resultados de N algoritmos de clasificación sobre el mismo conjunto de datos [13]. Por defecto, el nivel significativo es de 0.05 y en el campo de comparación se selecciona *percent_correct*. El conjunto de datos ingresado es el generado a partir de la política EstadoCompletoARFF con dos movimientos consecutivos por instancia. En el resultado muestra que con la configuración SimpleKMeans($t = 50.05$) supera a EM($t = 46.48$).

5 Conclusiones y Trabajo Futuro

Para un juego sencillo como TetrisAnalytics, se realiza el análisis del comportamiento de un conjunto de datos que contiene instancias tácticas de 6 jugadores aplicando los algoritmos SimpleKMeans y EM vía Weka. Se evalúan en dos instancias: primero utilizando los movimientos tácticos individualmente, y luego la unión de la traza actual con la siguiente de la misma partida. Debido a que el juego seleccionado tiene pocos movimientos posibles, de ahí que los algoritmos tienden a confundir los movimientos tácticos entre jugadores. En casos extremos, como los jugadores que siempre ubican las fichas a la izquierda y a la derecha, la tasa de error es baja e incluso, cuando se considera la información de la ficha siguiente, la tasa de error es nula para el caso de SimpleKMeans. No obstante, para jugadores que realizan diferentes movimientos la clasificación correcta es deficiente; aunque mejora cuando se unen las trazas de ficha actual y ficha siguiente.

Los algoritmos SimpleKMeans y EM calculan las matrices de confusión con las cuales podemos identificar la clasificación de las instancias de los jugadores a cada cluster. Para 4 de los jugadores, sus instancias se encuentran distribuidas en todos los clusters debido a que estos realizan diferentes movimientos tácticos a través de la partida. Esto da como resultado un comportamiento similar entre jugadores debido a la baja cantidad de movimientos posibles en el juego. Por ello, se puede concluir que la clasificación del comportamiento en juegos con rango de movimientos disponibles restringido no es muy acertada. La evaluación de movimientos tácticos individuales por jugador no describe en su totalidad el comportamiento de este. Sin embargo, los jugadores tienden a ubicar la ficha actual dependiendo de la siguiente con lo que mejora la clasificación cuando se tienen en cuesta mas de un movimiento. Para trabajo futuro, se realizarán experimentos donde se unan más de dos movimientos tácticos de un jugador por partida. Al obtener una mejor clasificación de jugadores, la asignación de la dificultad dependiendo de las habilidades mostradas por cada uno es más acertada mejorando la experiencia de juego y el interés de este.

References

1. Cook, D.: The chemistry of game design. Gamasutra (2007)

2. Drachen, A.: Introducing clustering i: Behavioral profiling for game analytics. Game Analytics Blog (Mayo 2014), http://blog.gameanalytics.com/blog/introducing-clustering-behavioral-profiling-game-analytics.html
3. Drachen, A., Canossa, A., Yannakakis, G.: Player modeling using self-organization in tomb raider: Underworld. In Proceedings of IEEE Computational Intelligence in Games (CIG) 2009 (Milan, Italy) (2009)
4. Drachen, A., Sifa, R., Bauckhage, C., Thurau, C.: Guns swords and data: Clustering of player behavior in computer games in the wild (2012), in proceedings of IEEE Computational Intelligence in Games, 2012 (Granada, Spain)
5. Drachen, A., Thurau, C., Sifa, R., Bauckhage, C.: A comparison of methods for player clustering via behavioral telemetry. https://andersdrachen.files.wordpress.com (2013)
6. Drachen, A: Introducing clustering ii: Clustering algorithms. Game Analytics Blog (2014), http://blog.gameanalytics.com/blog/introducing-clustering-ii-clustering-algorithms.html
7. El-Nasr, M.S., Drachen, A., Canossa, A.: Game analytics: Maximizing the value of player data. Springer Science & Business Media (2013)
8. Han, J., Kamber, M.: Data Mining: Concepts and techniques. Morgan Kaufmann (2006)
9. Hunicke, R.: The case for dynamic difficulty adjustment in games. In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology. pp. 429–433. ACM (2005)
10. Mahlman, T., Drachen, A., Canossa, A., Togelius, J., Yannakakis, G.N.: Predicting player behavior in tomb raider: Underworld. In Proceedings of the 2010 IEEE Conference on Computational Intelligence in Games (2010)
11. Mellon, L.: Applying metrics driven development to mmo costs and risks. Versant Corporation, Tech. Rep. (2009)
12. Missura, O., Gärtner, T.: Player modeling for intelligent difficulty adjustment. In Proceedings of the 12th international conference on discovery science (2009), berlin
13. Rabbany, R.: Comparison of different classification methods. http://webdocs.cs.ualberta.ca/řabbanyk/research/603/short-paper-rabbany.pdf
14. Witten, I.: Using weka 3 for clustering. http://www.cs.ccsu.edu/m̃arkov/ccsu_courses/DataMining-Ex3.html
15. Witten, I.: Waikito courses. https://weka.waikato.ac.nz/dataminingwithweka/preview
16. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
17. Yannakakis, G.N., Hallam, J.: Real-time game adaptation for optimizing player satisfaction (2009), berlin