Universidad de Chile
Facultad de Cs. Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC5113 - 1

Aprendizaje Automático Bayesiano

Definición de Proyecto

Autores: Vicente Oyanedel
Francisco Clavero
Profesor: Pablo Guerrero
29 de julio de 2016
Santiago, Chile.

Índice

1. Descripción Detallada del Proyecto

League of Legends es un videojuego del estilo MOBA (Multiplayer Online Battle Arena), donde dos equipos de 5 jugadores se enfrentan en una lucha por destruir el nexo (base) enemigo.

Figura 1: Mapa con los objetivos del juego.

Durante una partida, cada jugador puede controlar uno de aproximadamente 150 campeones distintos. Cada uno tiene distintas habilidades y atributos, que determinan su forma de uso, dificultad y proposito dentro del juego. Esto provoca que los campeones sean muy distintos entre si, y que cada jugador tenga una afinidad distinta por cada campeón, sea por gusto o por su rendimiento con ellos. La simplicidad de los objetivos y a variedad de los campeones hace a League of Legends un juego muy dinámico, a lo cual debe su fama.

Cada jugador tiene una cuenta de usuario con la cual se autentifica para jugar, y almacena estadistidísticas de cada partida jugada. Esto incluye datos como la cantidad de partidas jugadas, ganadas, o perdidas, la cantidad de contrincantes vencidos, los campeones que ha usado, entre otros, con las cuales se puede caracterizar a los usuarios.

El proyecto consiste en crear un sistema que recomiende campeones a un usuario de League of Legends con los cuales pueda tener un buen rendimiento en el juego, basado en las características de dicho usuario.

La recomendación se realiza en base a los campeones que utilizan usuarios con características similares a las del usuario que ingresa al sistema. Para lograr identificar los distintos grupos de jugadores similares, se utilizarán técnicas de clustering.

Una vez terminado el *Clustering* se generan 5 rankings a modo de recomendaciones de campeones, generadas en base a los jugadores del cluster del usuario. Estos 5 rankings corresponden a distintas recomendaciones que son explicadas a continuación:

1.1. Collaborative Filtering

Cada usuario de League of Legends puede jugar con 127 campeones distintos. Al jugar una partida un jugador gana *Champion Points* con **el campeón con el que jugó**. Los *Champion Points* de un jugador se pueden utilizar para identificar con cuales campeones el usuario juega más, en terminos de cantidad de partidas y desempeño (Al tener mejor desempeño se ganan más *Champion Points* por partida).

En base a esto se hace un *Collaborative Filtering* técnica de recomendación para un usuario dado. Éste encuentra campeones que le gusten al usuario mediante inspeccionar, para cada campeón, los *Champion Points* que tengan usuarios similares; para ello se define una medida de similitud entre dos usuarios para correlacionarlos. Luego se hace una suma ponderada sobre los (otros) usuarios del cluster de los *Champion Points* por su puntaje de similitud con el sujeto, así los usuarios más parecidos tienen más peso en la suma. Finalmente se entrega un ranking de los campeones con mejor puntaje obtenido de ésta suma ponderada.
La recomendación para un usuario se explica en 2 pasos:

1.1.1. 1 Paso

Se construye un ranking de los usuarios más similares, correlacionando al usuario con todos los otros usuarios del cluster, comparando sus *Champion Points* de cada campeón mediante una función de correlacion de Pearsons. Esta función es una medida de correlacion

entre dos jugadores, la cual se ocupa usualmente en este tipo de sistemas dado que es independiente de la escala de medida de los puntajes, en este caso, champion points. La medida de similitud que entrega va entre -1 a 1 (1 es mas similar), y dos usuarios son similares si comparten *Champion Points* en los mismos campeones (Se puede analogar con puntuar peliculas o escuchar canciones). Al final de este paso, para el sujeto se obtiene una lista descendiente de usuarios según su puntaje de similitud que vá entre -1 y 1, siendo 1 similitud máxima con el usuario en cuestión.

1.1.2. 2 Paso

Por cada campeón que el usuario no juega (No juega si tiene *Champion Points* < 6000), se hace una suma ponderada, sobre todos los usuarios del cluster, de sus *Champion Points* ponderado por su puntaje de similitud, obtenido en la parte 1. Esto permite darle más peso a los usuarios más similares al sujeto de la recomendación. Finalmente se construye un ranking de campeones ordenado por el resultado de la suma ponderada, siendo esta la recomendación del sistema de filtrado colaborativo.

1.2. Ranking Champion Points, KDA, WinRate y Heuristico

Aparte de la recomendación personal que obtiene el sujeto mediante el Filtrado Colaborativo, se generan 4 ranking de campeones obtenidos de sumar o promediar ciertas caracteristicas que se obtienen del clustering. A diferencia del Filtrado Colaborativo esta recomendación es igual para cualquier integrante del cluster, pero aporta información útil sobre el tipo de campeones que se juegan dentro del cluster dado que las caracteristicas que se toman en cuenta son reflejo directo del desempeño de los jugadores en el juego con cada campeón.
Cada ranking se explicará con detalle:

1. Ranking Champion Points (CP): Como se mencionó, cada usuario tiene una cantidad de Champion Points por cada Campeón de los 127 existentes, partiendo desde 0 CP (Nunca usado), y aumentando por cada partida jugada con ese campeón, dependiendo del desempeño. Por cada campeón, se recorre todos los usuarios del cluster sumando sus Champion Points con ese campeón. Finalmente se entrega el ranking de campeones con mayor Champion Points Totales dentro del Cluster. Este ranking muestra los campeones más usados dentro del cluster.

2. Ranking KDA: Otra medida de información ampliamente utilizada en la comunidad es conocida como KDA, la cual consiste en la suma total de los asesinatos + asistencias en asesinatos, dividido por la cantidad de muertes con cada campeón. Esto entrega un factor que al ser mayor significa que el campeón logra asesinar (o participar de uno) más por cada vez que él es asesinado. Dificilmente se obtiene con un campeón un KDA muy superior a 1; por lo mismo, un ranking de KDA obtenido de la suma de los KDA sobre todo el cluster entrega información notable sobre campeones que participan positivamente en el juego.

3. Ranking WinRate (WR): El winrate de cada campeón corresponde a la cantidad de victorias totales obtenidas con él, divididas en la cantidad de derrotas totales. Al tener un WR > 1 significa que se han ganado más partidas que las perdidas con ése campeon. Al igual que el KDA dificilmente crece mucho más que 1. Al hacer un ranking de WR sobre el cluster, los campeones más arriba son los que ganan más partidas de las que pierden.

4. Ranking Heuristico: Para congregar los 3 últimos ranking en uno solo, se construyó una función a mano la cual hace una suma ponderada para cada campeon de: log(ChampionPointsTotales), WinRateTotal y KDAChampionPointsTotales. Donde éstos corresponden a la suma de esas cantidades entre todos los usuarios del cluster. Se calcula el log de los CP dado que crece mucho mas rapido que el KDA y el WR. Y las ponderaciones para cada factor se impusieron a mano, haciendo diversas pruebas.

2. Alcances del Proyecto

En primer lugar, los datos se obtienen mediante la API oficial del juego, la cual provee un servicio de requests en formato json. Esta permite acceder directamente a datos de todos usuarios del mundo. Para el funcionamiento preliminar se descargaron todos los datos disponibles de 163000 usuarios de la región 'Latino America Sur' (El sistema se limita sólo con usuarios de esta región).

Al momento de hacer una nueva petición por parte de un nuevo usuario se incorporan sus datos a la base de datos para las posteriores recomendaciones. La velocidad de respuesta de la recomendación puede tardar bastante dado que el algoritmo de clustering tiene coste n^3, y para obtener recomendaciones hacen falta una buena cantidad de usuarios por

cluster. Se considera que mejorar esta velocidad está fuera de los alcances del proyecto.

Los pilares fundamentales del proyecto son el **Clustering** y la **recomendación sobre los usuarios del cluster**. Los cuales resultan de aplicar los conocimientos del curso de Aprendizaje Automatico Bayesiano, además de fomentar la investigación personal para temas que no se ven en el curso, pero si son relevantes en el area. Además, dados los buenos resultados expandimos los alcances proveyendo una app web desde donde consultar y mostrar los resultados; además de generar otras recomendaciones que no necesariamente aplican machine-learning pero de todas maneras entregan información relevante.

3. Rol Específico de cada Integrante

Se utilizará el software de control de versiones y coordinación GitHub, lo cual permite que los integrantes puedan trabajar paralelamente en el proyecto, sin la necesidad de especificar roles claros. Gran parte de la asignación de tareas se hizo de manera practica en muchas de las juntas de programación hechas con el fin de trabajar en el proyecto.

4. Metodología

Toda la programación será hecha en el lenguaje Python por diversas razones. en primer lugar, las librerías *json* y *requests* facilitan la obtención de los datos de la API, y su posterior manejo como base de dato basada en documentos (en formato json).

En segundo lugar, para realizar el clustering se utilizarán los métodos de la librería de python *sklearn*, y la librería *numpy* para manejar los datos.Clavero modifica esto

Para aprender y programar el sistema de recomendación 'Colaborative Filtering' se utilizó el siguiente articulo:`http://dataaspirant.com/2015/05/25/collaborative-filtering-recommendati`

Como se mencionaba anteriormente, para organización y control de versión, se utilizará GitHub.

5. Link github

`https://github.com/fcoclavero/machinelol`

6. Avances

A partir de los 163000 usuarios extraidos de la API, se programó un parser que exporta una cantidad variable de estos usuarios a usa base de datos SQLite, almacenando parte de sus caracteristicas primarias como usuarios (TotalChampionKills, TotalDeaths, etc). En particular, el sistema creado provee una forma de cargar cualquier usuario en la base de

datos para testear.

Para poder encontrar usuarios similiares a aquel que solicita la recomendación se realiza un paso de clustering sobre los datos. Debido a la gran cantidad de usuarios a los que se tienen acceso gracias a la API, primero se obtiene una muestra aleatoria reducida de estos, con el fin de agilizar el procesamiento de los datos. Esto es de especial importancia pensando en un front-end web.

Los datos que se tienen de los usuarios, como número promedio de kills, assists, deaths, o minion kills por partida tienen la particularidad de que una menor cantidad de usuarios tiene métricas más altas. Esto provoca una diferencia importante de densidad de puntos en el espacio de características. La densidad variable hace que algorítmos normales de clustering (como K-Means o DBSCAN) produzcan clusterings inadecuados, tomando, por ejemplo, a los puntos mas lejanos al origen (jugadores mejores) como outliers en el caso de DBSCAN, o clusterings con poco sentido, como el en el caso de K-Means.

Por esta razón se decide utilizar el algoritmo de Jarvis-Patrick ["Finding Clusters of Different Sizes, Shapes and Densities in Noisy, High Dimensional Data"; Ertöz, Steinbach, Kumar; February 20, 2003], basado en el grafo de Similar Nearest Neighbors (SNN), el cual se implementa desde cero. En este algoritmo se cambia la medida de similitud entre puntos, de la distancia eucilidiana (como en los ejemplos de otros algoritmos mencionados anteriormente), a una basada en la cantidad de vecinos similares que se comparten. Esto permite encontrar clusters de densidad variable, lo cual se ajusta más a los tipos de grupos que se pueden encontrar en los datos.

El primer paso del algoritmo consiste en generar un clique con todos los puntos. Luego se mantenien únicamente en cada punto las aristas a los k vecinos más cercanos. Después, la matriz de adyacencia del grafo se "sparcifica", visitando cada vertice y manteniendo las aristas con los puntos que compartan al menos epsilon vecinos cercanos con el punto visitado. Luego se generan clusters a cada componente conexa del gráfo, dejando a los puntos aislados como .ºutliers".

Se programo el sistema de recomendación que, dado un usuario sujeto, en primer lugar toma N datos aleatorios de la base de datos (incluido el sujeto), y ejecuta el algoritmo

de Clustering, posteriormente, se obtiene la lista de usuarios que pertenecen al cluster del sujeto. Con ésta lista, mediante el parser se extrae información relativa a los campeones que usa cada usuario del cluster, para luego ejecutar el 'Colaborative Filtering' y crear los 4 rankings, y entregarlos en pantalla.

Además se diseñó un front-end web para probar el sistema ingresando un nombre de usuario de League of Legends, y después de ejecutar el clustering y los scripts de recomendacion entrega las 5 recomendaciones de campeones.

7. Instrucciones

En el archivo readme.