Recomendador de películas basado en factorización de matrices

Francisco Rubin Capalbo
Universidad Politécnica de Valencia

16 de mayo de 2018

1. Introduction

Los sistemas recomendadores son ampliamente usados en aplicaciones que presentan contenido a sus usuarios, como Netflix, Spotify, Booking, etc. Para este tipo de aplicaciones, es esencial que a los usuarios les guste el contenido que están viendo, ya que esto hace que vuelvan a por más y pasen más tiempo en la página. En este trabajo presento una aplicación web de un sistema recomendador de películas, entrenado con la base de datos de MovieLens. Utilizo React para el frontend, expressjs para el backend, y un algoritmo de filtro colaborativo para el sistema recomendador, basado en factorización de matrices.

Recommender systems are widely used in applications that present content to their users. Netflix for movies, Spotify for music, Booking for hotels, etc. For this kind of applications, it is essential that the users like the content they are seeing, so they come back for more and stay more time in the page. In my work, I present a complete web application of a recommender system for Movies, based on the MovieLens dataset. I use React for the frontend, expressjs for the backend, and a collaborative filtering recommender based on Matrix factorization.

2. Descripción del problema

2.1. Dataset

La parte más importante en cualquier proceso de entrenamiento de un modelo, es la elección de los datos a utilizar. Para esta aplicación la elección de Dataset es obvia. He utilizado MovieLens[1], que contiene información de más de 200 mil películas y la valoración de éstas por más de 100 mil usuarios. Este Dataset también contiene información de género de cada película, y un conjunto de Tags asociados a éstas. No obstante, para este trabajo me he limitado a utilizar la lista de valoraciones.

Casi todas las pruebas se han realizado en el conjunto de datos reducido (unas 100 mil valoraciones). Únicamente al final, para entrenar el modelo definitivo, he empleado el dataset completo (más de 26 millones).

2.2. Filtrado Colaborativo

Una de las técnicas que dominan los sistemas recomendadores es el filtrado colaborativo. En términos generales, el filtrado colaborativo busca predecir las preferencias de un usuario basándose en las preferencias de otros usuarios similares a éste. Este tipo de sistema es muy popular, ya que funciona muy bien cuando hay una cantidad de datos extensa, como es MovieLens.

Otro beneficio del filtrado colaborativo es que no necesita ni está limitado por información sobre los items o los usuarios. En el ejemplo de las películas, no es necesario conocer el género de éstas

y asignar ciertos géneros preferidos a ciertos usuarios. La única información utilizada son las valoraciones. Esto permite generalizar más, y encontrar características de los datos que no se conocían anteriormente.

No todo es positivo. El filtrado colaborativo tiene un problema conocido como inicio frío, o 'Cold start'. Esto significa que cuando un usuario nuevo entra en la aplicación, y no ha hecho ninguna valoración anteriormente, el filtrado colaborativo no funciona. Es necesario un número mínimo de valoraciones para que los resultados empiecen a tener sentido. Esto se debe a que lo único que nos da información acerca del usuario son las valoraciones que hace, y sin valoraciones no tenemos ninguna información. Ya veremos más adelante que he utilizado para resolver este problema.

3. Implementación

3.1. SVD

La técnica que he escogido para implementar un Filtrado Colaborativo es SVD (Singular Value Decomposition) [2], o factorización de matrices. Empezamos con la matrix M, en la cual el índice i representa los usuarios, el índice j las películas, y el valor en Vij representa la valoración asignada por el usuario i a la película j. Al aplicar SVD, obtenemos las matrices U, S y V, que modelan M de esta manera:

$$M = U * S * V^T$$

La matriz S es diagonal cuadrada, y cada una de sus dimensiones representa cada una de las características extraídas. Las matrices U y V representan los pesos asignados para las características por cada uno de los usuarios y películas, respectivamente.

Hay varias implementaciones de SVD en el ecosistema de Python, en librerías como Scipy[3] o Numpy, pero ninguna de estas nos sirve. Esto se debe a que la técnica de SVD común tiene en cuenta todos los valores de la matriz M para hacer la aproximación. En nuestro caso sólo podemos utilizar los valores que representan una valoración realizada por el usuario, e ignorar el resto (los ceros). Sino, nuestro modelo intentará que estos valores se acerquen al cero lo máximo posible, y las predicciones serán erróneas. Tenemos que utilizar una versión modificada del SVD, llamada SVD Iterativo, o Funk SVD. Llamada así en honor al descubridor, Simon Funk [4].

3.2. Funk SVD

Básicamente, esta técnica consiste en aproximar las matrices U y V (ignorando S), sólo para los valores diferentes a 0. Se hace iterativamente, utilizando descenso por gradiente.

$$M \approx U * V^T$$

La función de pérdida utilizada para la aproximación es RMSE.

Finalmente, teniendo las matrices U y V, las podemos multiplicar para obtener una matrix Mén la cual los valores faltantes han sido reemplazados por predicciones basadas en las características y los pesos obtenidos.

He realizado una implementación propia de este algoritmo, la cual se puede encontrar en los documentos del trabajo o en mi github[5].

3.3. Cold Start

Para tratar con el problema del Cold Start, he optado por una solución sencilla pero efectiva. Cuando el usuario entra por primera vez a la página muestro una lista con películas aleatorias que tienen un mínimo de 1000 valoraciones. El objetivo es que al usuario se le muestren películas que conozca, para que así pueda valorarlas y se pase la fase fría rápidamente.

4. Aplicación Web

4.1. Backend

* Mencionar que tuve que implementar parte del algoritmo en JS, que habria sido mejor hacerlo en django pero por falta de tiempo y tener mas experiencia elegi node * Enumerar api expuesta y explicar

4.2. Frontend

* Explicar brevemente los componentes del frontend * Redux store, dejar claro que cosas se guardan en memoria del browser

5. Conclusiones

* ???????

Referencias

[1] MovieLens Dataset

[2] Singular Value Decomposition

[3] Scipy SVD

[4] Funk Journal, Netflix Prize

[5] Implementación Funk SVD