UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

FACTORIZACIÓN DE MATRICES PARA FILTRADO COLABORATIVO

P R O Y E C T O I

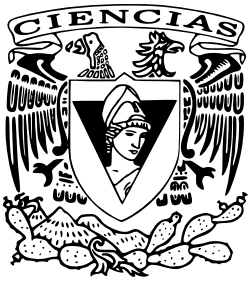
PRESENTA:

JORGE ESTEBAN MENDOZA ORTIZ

TUTORES:

M. EN C. MARÍA FERNANDA SÁNCHEZ PUIG
JOSÉ FERNANDO MÉNDEZ TORRES

Ciudad de México, 2021



Índice

1. Introducción

Históricamente, los sistemas recomendadores han sido algunas de las primeras aplicaciones de *aprendizaje de máquinas* de las que se tiene registro. Sin entrar en muchos detalles, los orígenes de los sistemas recomendadores se encuentran en el desarrollo de disciplinas de las ciencias de la computación como *information retrieval*, lo que tradicionalmente se conocía como *data mining —minado de datos—* y áreas afines como teoría de la información.

Estas disciplinas encontraron en la Internet temprana un rico panorama para abordar problemas básicos de los sistemas de información distribuidos, cuyas soluciones hoy damos más o menos por sentado, como: (1) cómo hallar contenido basados en metadatos de ciertos documentos, (2) cómo crear motores de búsqueda, (3) cómo obtener resultados personalizados para usuarios al buscar en un *corpus* de documentos, o (4) cómo buscar contenido de medios distintos de texto.

Los sistemas recomendadores son uno de los pilares fundamentales de la Internet contemporánea; y así, en sus aplicaciones modernas dichos sistemas nos sugieren qué productos comprar, con qué personas conectar, qué contenidos ver, qué música escuchar, qué lugares visitar, qué lecturas hacer, qué sitios visitar, y un largo etcétera.

Existen multitud de tipos de sistemas recomendadores: desde los que se basan en las calificaciones de usuarios sin tener conocimiento de las propiedades de ningún documento para hacer recomendaciones generales, pasando por los que utilizan metadatos o atributos de los documentos para hacer recomendaciones y los que utilizan el contenido de dichos documentos, hasta llegar a los que utilizan el conocimiento de las preferencias de usuarios y de las similitudes entre estos para hacer recomendaciones. Este último tipo de sistemas, conocidos como de filtrado colaborativo, es el que nos interesa.

El presente proyecto tiene por objetivo mostrar cómo se puede utilizar la técnica de descomposición en valores singulares para implementar un sistema recomendador de filtrado colaborativo.

La idea general del proyecto es partir de una base de datos de usuarios que han calificado películas, y utilizar la estructura de preferencias de los demás usuarios para predecir cómo calificarían otras películas que no han calificado.

La base de datos que utilizamos en este proyecto se llama *ml–latest–small*, y contiene —entre otras cosas— calificaciones de entre 1 y 5 estrellas de una muestra del sitio *MovieLens*, un servicio recomendador de películas administrado por el grupo *GroupLens*, un laboratorio de investigación de la Universidad de Minnesota.

La base de datos contiene 100,836 calificaciones de 9,742 películas. Los datos fueron creados por 610 usuarios entre el 29 de marzo de 1996 y el 24 de septiembre de 2018 [1]. Los usuarios fueron seleccionados aleatoriamente, y todos los usuarios calificaron por lo menos 20 películas.

En este documento delineamos primero el marco teórico, describiendo brevemente varias formas de sistemas recomendadores y centrándonos en los principios del filtrado colaborativo. Procedemos a tratar el problema de alta dimensionalidad al que uno se enfrenta cuando hace filtrados colaborativos; describimos la lógica bajo la que se implementa la descomposición en valores singulares y describimos brevemente cómo se evalúa la eficacia de este método para hacerlo comparable con otros.

Posteriormente describimos brevemente la implementación del proyecto. Un *notebook* con código en *Python* puede encontrarse en este repositorio: `https://github.com/esteban-mendoza/svd-recommender` con la implementación documentada completa. Al final mostramos los resultados obtenidos y hacemos comentarios finales en las conclusiones.

2. Marco Teórico

En esta sección introducimos la teoría necesaria para entender el sistema recomendador que hemos implementado.

2.1. Sistemas recomendadores

Para construir un sistema recomendador, tenemos dos requisitos estructurales:

- **Usuarios**: entidades que interactúan con recursos de algún sistema de información, a partir de cuyas acciones podemos recolectar datos explícitos e implícitos y generar perfiles de preferencias.
- **Documentos**: recursos del sistema de información sobre los que podemos recolectar metadatos y sobre los que deseamos predecir las preferencias de los usuarios.

  Utilizamos la palabra *documento* de acuerdo con la terminología clásica de *information retrieval*; no obstante, dependiendo del contexto el *documento* en cuestión puede ser algún recurso del sistema como productos, lugares, otros usuarios o películas —que será nuestro particular interés más adelante—. No obstante, utilizaremos el término *documento* mientras describamos el marco teórico del proyecto.

Realizamos ahora un breve recuento de las características de diversos tipos de sistemas recomendadores:

- **No personalizados**: son los que infieren qué documentos son globalmente más importantes de acuerdo con las interacciones de los usuarios: más comprados, mejor puntuados, etc., sin importar las preferencias del usuario que recibe la recomendación.

- **Basados en contenido**: este tipo de sistemas recomendadores utilizan propiedades (metadatos) de los documentos para hallar similitudes con otros documentos a partir de métricas como similitud de coseno o TF-IDF (*Term Frequency–Inverse Document Frequency*). Una vez que se ha establecido una métrica adecuada para determinar el grado de similitud entre documentos podemos crear un perfil de preferencias de cada usuario y utilizar esta información para recomendar otros documentos nuevos. Este tipo de sistemas funcionan bien para recomendar documentos nuevos —en el sentido de que los usuarios no han interactuado con ellos— siempre que se cuente con suficientes metadatos de cada documento.

- **Filtrado colaborativo**: el filtrado colaborativo surge cuando existen suficientes registros de interacciones de los usuarios con los documentos de tal forma que no determinamos la similitud de los documentos a partir de análisis de metadatos o de contenido, sino a partir de las preferencias de usuarios similares. Esto es: si el usuario A es similar en sus gustos a los usuarios B y C, y estos usuarios han calificado positivamente un documento nuevo para el usuario A, nosotros podemos recomendar este documento al usuario.

  Esta aproximación a los sistemas recomendadores funciona cuando tenemos calificaciones de usuarios, y es particularmente apropiado cuando las preferencias de los usuarios son altamente dependientes del contenido, como con películas, música, videos, etc.

Existen otros tipos de sistemas como sistemas híbridos que permiten modificar las recomendaciones dadas a los usuarios a partir de otras fuentes de información distintas del perfil del usuario. Ejemplos de esto son los contenidos patrocinados que se muestran a ciertos grupos demográficos. Ejemplos de técnicas utilizadas para sistemas recomendadores híbridos son *mixes*, *switches* y *stacks*. Otras técnicas que no exploramos en este documento son conocidas como *learning-to-rank*, que evitan por completo hacer predicción, a diferencia de la técnica de factorización de matrices que presentamos ahora.

Hemos elegido implementar un sistema de filtrado colaborativo porque presenta problemas interesantes como el de alta dimensionalidad y dispersión de los datos, y porque las técnicas de descomposición de matrices que utilizamos son sumamente populares hoy día [2] y porque estos métodos son altamente extensibles y flexibles.

2.2. Problemas con alta dimensionalidad y factorización de matrices

Como hemos descrito antes, la base de datos necesaria para implementar un sistema de filtrado colaborativo requiere: (1) usuarios, (2) documentos, que en este caso son películas, y (3) calificaciones realizadas por los usuarios —o alguna otra métrica ordinal comparable entre todos — [3].

Con la anterior base de datos construimos la matriz R (de *rankings*), que tiene a los usuarios en las filas y a los documentos (*items*) en las columnas. Dicha matriz R tiene la siguiente

estructura:

$$R = \begin{pmatrix} — & u_1 & — \\ — & u_2 & — \\ & \vdots & \\ — & u_m & — \end{pmatrix} = \begin{pmatrix} | & | & & | \\ i_1 & i_2 & \cdots & i_m \\ | & | & & | \end{pmatrix},$$

con m usuarios y n documentos. Observemos que, desde luego, no todos los usuarios han calificado todos los documentos, de modo que la estructura de datos resultante tiene entradas vacías —que desde el punto de vista computacional se representan como *NA*—.

En la práctica, la razón de entradas vacías a entradas no vacías es sumamente alta. En otras palabras, la matriz es altamente *dispersa*. En el caso de la base de datos que utilizamos, esto se debe a que el número de películas es mucho mayor al número de usuarios. Los usuarios sólo califican una pequeña porción de las películas totales cada vez, de modo que las preferencias de los usuarios son difíciles de calificar por sí solas utilizando medidas de similitud: es altamente probable que dos usuarios distintos no tengan películas en común.

Desde el punto de vista matemático el que la matriz R sea dispersa implica que los vectores base que conforman la transformación lineal asociada a R son elementos de un espacio de muy alta dimensionalidad, pero que la información codificada en la matriz forma parte de un espacio de dimensionalidad mucho menor. Teniendo esto en mente, la técnica de factorización en valores singulares resuelve el problema al reducir la dimensionalidad de la matriz.

Para estudiar diversas aplicaciones en ciencia de datos e ingeniería de la descomposición en valores singulares (SVD, por sus siglas en inglés) como compresión de imágenes, procesamiento de señales, etc., refiérase al primer capítulo de [4], y refiérase a la sección 6.7 de [5] para tratar los detalles sobre la derivación matemática de esta descomposición.

En términos generales, SVD generaliza la descomposición en valores propios (descomposición espectral) de matrices cuadradas no singulares al caso de matrices reales o complejas no cuadradas. Esta descomposición existe para toda matriz, y si la matriz descompuesta es real, los factores resultantes serán matrices reales también.

Dada una matriz $R \in \mathbb{R}^{m \times n}$, la descomposición en valores singulares *compacta*[1] está dada por:

$$R = U \Sigma V^T,$$

donde $U \in \mathbb{R}^{m \times k}$ y $V \in \mathbb{R}^{k \times n}$ son matrices ortogonales y

$$\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \end{pmatrix}$$

es una matriz diagonal real con $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$. Las entradas en la diagonal de Σ se conocen como *valores singulares*.

[1] Más detalles sobre la diferencia entre SVD completa y SVD *compacta* o *truncada* pueden encontrarse en [4, p. 8]

Las filas de la matriz

$$U = \begin{pmatrix} — & u_1 & — \\ — & u_2 & — \\ & \vdots & \\ — & u_m & — \end{pmatrix}$$

representan la *afinidad* de cada usuario respecto de dimensiones conocidas en este contexto como características latentes (*latent features*).

Las filas de la matriz

$$V = \begin{pmatrix} — & i_1 & — \\ — & i_2 & — \\ & \vdots & \\ — & i_m & — \end{pmatrix}$$

representan, a su vez, la relevancia de estas características latentes en cada documento.

Por último, los valores singulares $\sigma_i = \Sigma_{i,i}$ determinan el peso de cada característica latente —motivo por el que están en orden descendente—. Es decir, que si deseamos reconstruir la matriz original truncando las matrices obtenidas y sólo preservamos los primeros l valores principales y las primeras l columnas de las matrices U y V, con $l < k$, estaremos *comprimiendo* la información codificada en la matriz.

Las dimensiones que hemos llamado *características latentes* son usualmente no interpretables y abstractas. Esto hecho es irrelevante desde el punto de vista de la tarea que nos atañe, puesto que utilizamos SVD para construir la matriz $\tilde{R}$, que es la mejor aproximación de rango l posible —utilizando alguna norma de matrices como la norma de Frobenius— a la matriz R.

La matriz resultante, al formar parte de un espacio de dimensión menor, en el que se codifica la información de la matriz, será *densa* (*no dispersa*). Más aún, las entradas resultantes representarán una predicción de la calificación que daría cada usuario a cada documento. En términos matemáticos:

$$\tilde{r}_{i,j} = \sum_{f=1}^{l} u_{i,f} \cdot \sigma_f \cdot v_{f,j}.$$

Una nota importante es que para poder implementar SVD requerimos imputar las casillas vacías (*NA*) con valores numéricos. La forma en que en la práctica resolvemos este problema es sustrayendo la media de calificaciones de cada usuario μ_i a las calificaciones de cada usuario —los vectores u_i—. Las calificaciones resultantes son comparables entre sí, y habiendo hecho esto podemos imputar las entradas vacías con 0, una calificación *neutral*.

Si sustrajimos la calificación media de cada usuario, la predicción de la calificación que daría cada usuario a cada documento estaría dada por:

$$\tilde{r}_{i,j} = \mu_i + \sum_{f=1}^{l} u_{i,f} \cdot \sigma_f \cdot v_{f,j}$$

Antes de concluir el apartado técnico referente a SVD relevante hacer mención de un par de aspectos prácticos de suma importancia.

En primer lugar, el lector debe notar que el algoritmo SVD es computacionalmente costoso —$O(\min\{mn^2, m^2n\})$ según [6]—, de modo que en la práctica el cálculo que presentamos se realiza con baja frecuencia. Esto implica que tal vez nos encontremos con usuarios para los que no podemos hacer predicciones porque no fueron incluidos en el cálculo.

Afortunadamente, puesto que las matrices U y V son ortogonales, dado $R = U\Sigma V^T$, al multiplicar por V a cada lado de la anterior ecuación obtenemos que $RV = U\Sigma V^T V$. Dado que V es ortogonal, entonces $V^T V = I$, por cuanto $RV = U\Sigma$. Además, como Σ es una matriz diagonal es fácil de invertir, con lo que obtenemos que $RV\Sigma^{-1} = U$.

Es decir que si tomamos alguna «nueva fila» de R, correspondiente a un nuevo usuario, llamada r', podemos obtener su correspondiente proyección en el espacio de la matriz U de afinidad del usuario respecto de las dimensiones de características latentes, con lo que podemos utilizar Σ y V^T para predecir las calificaciones del nuevo usuario como lo hemos descrito antes.

2.3. Evaluación de métodos

En esta sección hacemos una breve nota sobre la forma en que hacemos evaluación cruzada de nuestro método.

Observemos que si deseamos guardar una fracción $0 < p < 1$ de datos de evaluación para nuestro modelo, dado que la matriz R tiene $m \cdot n$ entradas, entonces debemos elegir de manera aleatoria $\lfloor \sqrt{p} \rfloor m$ usuarios —filas— y $\lfloor \sqrt{p} \rfloor n$ documentos —columnas—, de modo que $\lfloor \sqrt{p} \rfloor m \cdot \lfloor \sqrt{p} \rfloor n \approx p \cdot mn$.

Una vez que hemos determinado aleatoriamente dichas filas y columnas, elegimos las entradas correspondientes al producto cruz de dichos índices y las guardamos en una submatriz de datos de evaluación y sustituimos dichas entradas por entradas vacías.

Una vez que hemos realizado las predicciones correspondientes, podemos utilizar alguna función de error (*loss function*) para determinar el error. Una métrica usual en este caso es *RMSE* (*Root Mean Squared Error*), que está dada por:

$$\text{RMSE} = \sqrt{\sum_{i,j} (r_{i,j} - \tilde{r}_{i,j})^2}$$

3. Resultados

La implementación del código documentada paso a paso junto con los datos utilizados se pueden encontrar en `https://github.com/esteban-mendoza/svd-recommender`.

El RMSE obtenido para nuestras predicciones es de 0.9260111308749108, que se interpreta como que en promedio las diferencias entre cada predicción y la calificación original es de 0.92 puntos, aproximadamente.

4. Conclusiones

La descomposición en valores singulares para crear filtrado colaborativo, como hemos visto, es un método robusto y fácilmente extensible que debería formar parte del bagaje elemental

de cualquier practicante de aprendizaje de máquinas.

Como hemos mencionado, los métodos basados en factorización de matrices dominan el campo del filtrado colaborativo moderno, de modo que la presente introducción representa una gran oportunidad para el lector interesado en explorar métodos más avanzados [2], [7].

Note que nuestra implementación es completamente reproducible, y nuestros resultados son, por tanto, comparables con los de otras técnicas. Así, además de ser una introducción a una técnica fundamental de la ciencia de datos moderna, uno de los logros de este trabajo es el de crear una línea base de comparación con otras técnicas.

Referencias y bibliografía

[1] F. M. Harper y J. A. Konstan, «The MovieLens Datasets,» *ACM Transactions on Interactive Intelligent Systems*, vol. 5, n.º 4, págs. 1-19, ene. de 2016. DOI: 10.1145/2827872.

[2] Y. Koren, R. Bell y C. Volinsky, «Matrix Factorization Techniques for Recommender Systems,» *Computer*, vol. 42, n.º 8, págs. 30-37, ago. de 2009. DOI: 10.1109/mc.2009.263.

[3] G. Orlandoni, «Escalas de medición en Estadística,» *Telos*, vol. 12, n.º 2, 2010. dirección: http://ojs.urbe.edu/index.php/telos/article/view/2415.

[4] S. L. Brunton y J. N. Kutz, *Data–Driven Science and Engineering*. Cambridge University Press, ene. de 2019. DOI: 10.1017/9781108380690.

[5] S. H. Friedberg, A. J. Insel y L. E. Spence, *Linear Algebra*, I. S. University, ed. Pearson Education, 2003.

[6] A. K. Cline e I. S. Dhillon, *Computation of the Singular Value Decomposition*. CRC Press, ene. de 2006. dirección: http://www.cs.utexas.edu/users/inderjit/public_papers/HLA_SVD.pdf.

[7] R. Mehta y K. Rana, «A review on matrix factorization techniques in recommender systems,» en *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, IEEE, abr. de 2017. DOI: 10.1109/cscita.2017.8066567.