Facultad de Matemática y Computación (UH)

Matemática Numérica

Sistemas de Recomendación basados en SVD

SVD: Siempre Venciendo Desafios

Autores:
Daniel Abad Fundora C212
Anabel Benítez González C211
Raudel Alejandro Gómez Molina C211
Alex Sierra Alcalá C211

Noviembre 2022

1 Sistemas de Recomendación

Desde el nacimiento de internet como tecnología se han abierto nuevas posibilidades que facilitan el acceso a la información, también esta se ha aumentado considerablemente. Tal impacto se ha visto reflejado en todo tipo de áreas, una de ellas es el comercio, donde sitios especializados en internet ofertan millones de productos o servicios, también hay sitios donde la mayoría de datos pueden ser efímeros o de poca utilidad, produciendo un caos de información que, al contrario de satisfacer requerimientos de los usuarios, pueden complicar más su respuesta; para que sea efectiva, depende de la habilidad y experiencia que tenga el usuario para expresar sus necesidades mediante una consulta y está demostrado que en la mayoría de casos es imprecisa y vaga. Por lo anterior surgen los sistemas de recuperación de información, que si bien apuntan a mejorar la forma en que se almacena, representa y organiza la información, su función no es devolver la información deseada por el usuario sino únicamente indicar qué documentos son potencialmente relevantes para dicha necesidad de información. A partir de esto, surge una nueva necesidad de encontrar herramientas que se encarguen de buscar por los usuarios, que los conozcan y recomienden un conjunto limitado de opciones acordes a sus intereses.

Un sistema de recomendación se puede definir, de manera formal, como aquel sistema que tiene como principal tarea seleccionar ciertos objetos de acuerdo con los requerimientos del usuario. Estos sistemas son muy atractivos en situaciones donde la cantidad de información que se ofrece al usuario supera ampliamente cualquier capacidad individual de exploración.

La creación de un sistema de recomendación cuenta con tres fases principales:

- Captura de las preferencias y los gustos e intereses del usuario.
- Extracción del conocimiento, aquí el sistema se encarga de interpretar la información que recopiló anteriormente para poder predecir los gustos y las preferencias del usuario.
- A partir del contenido procesado anteriormente el sistema se encarga de se seleccionar los ítems que podrían interesarle a un determinado usuario.

Ahora debemos encontrar un mecanismo para procesar toda la información que recopilemos previamente de los usuarios, para ello debemos reducir la dimensionalidad de nuestra matriz de votaciones, ya que esta puede ser muy grande y dispersa, a este proceso lo llamaremos *Reducción de la Dimensionalidad*.

Por lo que es necesario determinar una matriz que sea equivalente a la original y que sea mas concisa a la hora de brindar la información para realizar la recomendación. Esto permite hacer más eficiente el proceso pues solo tendremos que considerar las características principales en vez de analizar completamente toda nuestra extensa matriz original, además esto permite minimizar los problemas relacionados con la presencia de datos erróneos en nuestra recopilación.

2 Descomposición en Valores Singulares(SVD)

La factorización SVD consiste en expresar una matriz A de tamaño(n,d) como el producto de tres matrices:

$$\underset{(n,d)}{A} \approx \underset{(n,n)}{U} \cdot \underset{(n,d)}{\Sigma} \cdot \underset{(d,d)}{V}^{T}$$

Donde:

- A es una matriz ortogonal de tamaño (n,n) que contiene a los vectores singulares izquierdos de A
- $\sum$ es una matriz diagonal de tamaño (n,d)cuyos valores son los valores singulares de A ordenados en valor decreciente
- V es una matriz de tamaño (n,d) que posee los valores singulares derechos de A

Existe una propiedad aplicada a SVD enfocados en los sistemas de recomendación, esta consiste en que reduciendo el número de valores singulares de la matriz Σ a los primeros k valores, se obtendrá una aproximación de la matriz original A, que permite ser reconstruida a partir de las versiones reducidas de las otras matrices cometiendo un cierto error, pero disminuyendo el tamaño. Es decir:

$$A_{n,m} \approx U_{n,k} \cdot \Sigma_{k,k} \cdot V^{T}_{k,m}$$

Esta propiedad es derivada del teorema de Eckart-Young que aborda la mejor aproximación a la matriz original A, obteniéndola poniendo a 0 los n valores singulares más pequeños, así se reducirán las matrices al número de valores singulares no nulos que tenga la matriz Σ. Esto resulta entonces en la transformación de gran cantidad de datos en su representación reducida, siendo por lo tanto una propiedad muy importante que permite reducir considerablemente el tiempo de cómputo de cálculo y de uso de memoria para las tres matrices.

2.1 Filtrado Colaborativo

El filtrado colaborativo es un método para hacer predicciones automáticas (filtrado) sobre los intereses de un usuario mediante la recopilación de las preferencias o gustos de información de muchos usuarios (colaborador). El Filtrado colaborativo se basa, en que si una persona A tiene la misma opinión que una persona B sobre un tema, A es más probable que tenga la misma opinión que B en otro tema diferente que la opinión que tendría una persona elegida azar.

Dentro del filtrado colaborativo, es necesario el manejo de la matriz de votaciones de usuarios e ítems, por lo tanto, es posible considerar esta matriz como la base para aplicar SVD y obtener la factorización de matrices U, S y V. Luego, usando el teorema de Eckar-Youn, se puede reducir a k dimensiones. La matriz de factorización sería:

$$A_{usuarios,tems} \approx U_{usuarios,k} \cdot \Sigma_{k,k} \cdot V^T_{k,tems}$$

Por otro lado, es posible simplificar el proceso de SVD obteniendo solo los factores de usuarios e ítems, esto es posible descomponiendo la matriz Σ en dos matrices iguales, factores de usuarios $Ufac$ y factores de ítems $Ifac$ de esta manera:

$$\Sigma_{k,k} = \begin{pmatrix} \lambda_1 & & ... \\ & \lambda_2 & \\ ... & & \\ & & \lambda_k \end{pmatrix}$$

$$\Sigma_{k,k} = \begin{pmatrix} \sqrt{\lambda_1} & & ... \\ & \sqrt{\lambda_2} & \\ ... & & \\ & & \sqrt{\lambda_k} \end{pmatrix} \cdot \begin{pmatrix} \sqrt{\lambda_1} & & ... \\ & \sqrt{\lambda_2} & \\ ... & & \\ & & \sqrt{\lambda_k} \end{pmatrix}$$

Nota : Los λ_i son valores singulares de A.

Por tanto la matriz de votaciones se puede expresar como $Votos = Ufact \cdot Ifact$, donde :

$$Ufact = U_{n,k} \cdot \begin{pmatrix} \sqrt{\lambda_1} & & ... \\ & \sqrt{\lambda_2} & \\ ... & & \\ & & \sqrt{\lambda_k} \end{pmatrix}$$

$$Ifact = \begin{pmatrix} \sqrt{\lambda_1} & & ... \\ & \sqrt{\lambda_2} & \\ ... & & \\ & & \sqrt{\lambda_k} \end{pmatrix} \cdot V^T_{k,m}$$

Para hallar esta aproximación, la idea es buscar los factores de usuarios e ítems a partir de la matriz de votación abordándolo como un problema de regresión, donde se quieren encontrar los valores de las matrices de factores. Así, si se considera a q_i al vector que represente los factores de un ítem y p_j al vector que representa los factores del usuario se tendría que cumplir :

$$A_{i,j} = q_i^T \cdot p_j \approx \sum_{n=1}^{k} q_{i,n} \cdot p_{n,j}$$

Ahora solo se ajustan los factores de aquellos usuarios que hayan definido un voto reduciendo considerablemente el sistema de ecuaciones y solucionando el problema de la dispersión, cometiendo un pequeño error $e_{i,j} = |A_{i,j}$ - $q_i^T \cdot p_j|$. Debido a que el error absoluto es una función complicada de tratar, se usará el error cuadrado medio:

$$(e_{i,j})^2 = (A_{i,j} - q_i^T \cdot p_j)^2 \approx (A_{i,j} - \sum q_{i,n} \cdot p_{n,j})^2$$

O sea, intentar encontrar el error mínimo para hacer la mejor aproximación de la siguiente forma:

$$min_{p,q} = \sum (A_{i,j} - q_{i,n} \cdot p_{n,j})^2$$

Pero es necesario además hacer una regularización para evitar el *Outfitting*, así, es posible llegar a:

$$(e_{i,j})^2 = (A_{i,j} - \sum q_{i,n} \cdot p_{n,j})^2 + \frac{\lambda}{2} \cdot (||q_i||^2 + ||p_j||^2)(1)$$

Donde λ es una constante de regularización.

Para encontrar el mínimo de la función ya detallada existen técnicas como la del descenso del gradiente que buscan ajustar las matrices de factores poco a poco de manera automática. Esto se realizará en la función (1), con el fin de encontrar el mínimo de la función, o acercarse lo suficiente a la solución. Esta técnica consiste en ir moviéndose poco a poco por la función hasta encontrar el mínimo evaluando, la función en un punto dado para luego moverse una distancia hacia otro punto; ahora se calcula la derivada en ese punto y se desplaza al lado opuesto de la derivada de la función que se quiere minimizar.

Aplicando descenso del gradiente en (1) se tiene:

$$\frac{\delta(e_{i,j})^2}{\delta p_{i,n}} = -2q_{n,j} \cdot (A_{i,j} - \sum q_{i,n} \cdot p_{n,j}) + \lambda p_{i,n} (2)$$

$$\frac{\delta(e_{i,j})^2}{\delta q_{n,j}} = -2p_{i,n} \cdot (A_{i,j} - \sum q_{i,n} \cdot p_{n,j}) + \lambda q_{n,j} (3)$$

Teniendo ya el gradiente, solo queda aplicar las reglas de actualización tanto para $p_{i,n}$ como para $q_{n,j}$ en (2), (3) con las constantes de regularización y aprendizaje. Así, la regla de actualización parte de $\sigma_{n+1} = \sigma_n - \alpha \nabla f(x)$, donde σ_{n+1} es el nuevo valor en la matriz de votaciones, σ_n su valor actual, α constante de aprendizaje y $\nabla f(x)$ es el gradiente ya obtenido. Dando como resultado :

Para $p_{i,n}$:

$$p'_{i,n} = p_{i,n} + \alpha[2q_{n,j} \cdot (A_{i,j} - \sum q_{i,n} \cdot p_{n,j}) - \lambda p_{i,n}] (4)$$

Para $p_{n,j}$:

$$q'_{n,j} = q_{n,j} + \alpha[2p_{i,n} \cdot (A_{i,j} - \sum q_{i,n} \cdot p_{n,j}) - \lambda q_{n,j}] (5)$$

El enfoque se plasma en las expresiones (4) y (5) que serán las que actualizarán los valores de la matriz de votaciones a partir de cada *Ephochs*, tanto para los factores de usuarios, como para los factores de ítems.

3 Creación del sistema de Recomendación en Python

Conocemos lo complicado que resulta para un universitario elegir un tema para su tesis de grado, y aún más engorroso resulta el hecho de buscar entre la gran cantidad de temas que pueden existir y descartar aquellos que no interesan al estudiante, proponemos a partir del uso de un sistema de recomendación basado en SVD una solución para este problema. Dicha solución consiste en, teniendo en cuenta el recorrido del estudiante durante la carrera (en las asignaturas, proyectos e investigaciones extraclase) y además revisando las opiniones de usuarios con intereses afines a los suyos filtrar los temas y quedarnos solamente con los n (cantidad que se puede elegir) que resulten más interesantes para él. Así el indeciso universitario puede confiar totalmente en el programa y pedirle que le asigne un tema para la tesis, o puede simplemente pedir que le seleccione los 10 temas que más se apeguen a sus intereses y luego con calma elegir de esos cual es el más adecuado para él; pero siempre teniendo presente que no es lo mismo elegir uno entre 10 que elegir uno entre toda la lista de potenciales temas que puede haber en internet.

Para la creación del sistema de recomendación en Python usaremos las bibliotecas suprise y pandas. Necesitamos tener un dataset que contenga los temas de investigación (contaremos asiganturas o sus proyectos como temas para aquellas personas que no están muy vinculadas a los proyectos extraclase), este dataset debe estar ubicado en la carpeta data con el nombre de items.csv y contar con las propiedades: itemId (un entero positivo, diferente para cada tema), title (el nombre del tema de investigacion o asigantura), genres(departamentos a los cuales está vinculado dicho tema o asignatura). Además necesitaremos otro dataset que cuente con los temas que cada usuario ha podido evaluar, el mismo debe estar ubicado en la carpeta data con el nombre de ratings.csv y contar con las siguientes propiedades: userId (un entero positivo, diferente para cada usuario), itemId (el itemId correspondiente al tema que se evalúa), rating (un real en el rango [2,5], en el caso de las asignaturas lo interpretamos como la nota que se obtuvo, en el caso de los temas cada ususario les asigna un valor en función de cuán útil es para la sociedad, cuánto aporta la investigación del tema a la rama o cuán interesante resulta el tema para el usuario).

Primero necesitamos cargar los datasets, esto lo haremos mediante pandas, luego debemos darle formato a nuestros datos para que sean reconocidos por el sistema, seguidamente procedemos a realizar el entrenamiento y el testing de los datos. Después de culminar el proceso anterior pasamos anterior entrenamos todos nuestros datos y creamos nuestra función de recomendación la cual recibe el id del usuario al cual queremos recomendar, el dataset con los datos, el modelo entrenado y la cantidad de items que queremos recomendarle al usuario.

Actualmente probamos nuestro modelo con un dataset que contiene películas y las recomendacines hechas por varios usuarios, proximamente convocamos a la facultad a un trabajo conjunto para la elaboración de un dataset con los datos antes planteados para su ejecución y evaluación en el proceso descrito anteriormente.

El código fuente se encuentra disponible en `https://github.com/raudel25/SVD-SRI.git`