Obligatorio 2

Sistemas de Recomendación

Un *sistema de recomendación* es un algoritmo que busca predecir el puntaje o preferencia que un usuario le daría a un ítem (que puede ser una película, una canción, un producto, etc). Son usados principalmente en aplicaciones comerciales, como en Netflix, Youtube, Spotify, Amazon/Mercado Libre, etc.

En su mayoría, se construyen usando los siguientes modelos:

- *Filtrado basado en contenidos*: se utilizan características discretas predefinidas de un ítem, a fin de recomendar ítems de características similares a un determinado usuario. Por ejemplo, para una película podríamos usar su género, productor, qué actores aparecen, año, etc. Se necesitan conocer las características de cada ítem (lo cual en algunas situaciones puede no cumplirse, o ser costoso de obtener). Se recomiendan ítems para cada usuario particular, sin utilizar los gustos de otros usuarios que puedan ser similares.
- *Filtrado colaborativo*: se construye el modelo basándose en comportamiento pasado de un conjunto de usuarios (películas anteriormente puntuadas, productos comprados, etc). Se asume que los usuarios que tienen comportamientos similares en el pasado lo tendrán en el futuro, y se utiliza la información pasada (puntajes asignados) para crear perfiles de usuarios similares. Estos sistemas generan recomendaciones usando solamente información sobre puntajes asignados por los usuarios en diferentes ítems (no utiliza características predefinidas).

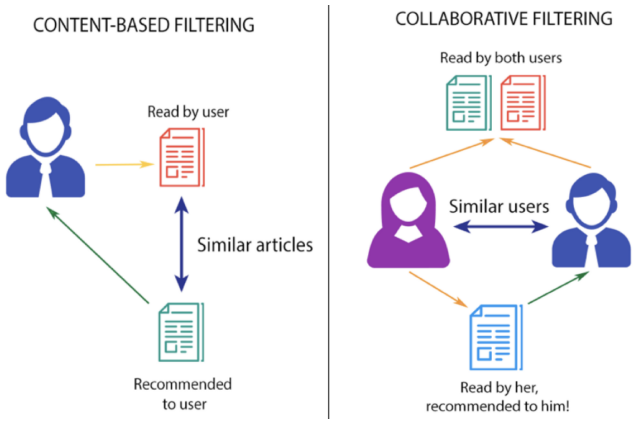

Figura 1: Modelos para construir sistemas de recomendación.

El objetivo de este trabajo es implementar la versión básica del sistema de recomendación de películas que ganó el millón de dólares del Netflix Prize [1]. Por más información, ver [2].

Dataset

Usaremos el conjunto de datos pequeño (*small*) de MovieLens [3]. El mismo consiste en 100836 puntajes realizados por 610 usuarios en un total de 9742 películas. Cada película se identifica por un número natural i entre 1 y $N = 9742$, y la lista de películas (junto a su índice) se incluye en el archivo `peliculas.csv`. Dicha lista incluye además la clasificación de cada película en al menos uno los siguientes géneros:

Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western.

Los usuarios incluídos en dicho conjunto puntuaron al menos 20 películas. Cada usuario se identifica por un número natural u entre 1 y $M = 610$. Los puntajes se dividieron en dos partes: El archivo `puntajes_ajuste.csv` contiene una lista de largo 95836 donde cada fila corresponde a una terna `(usuario_Id, pelicula_Id, puntaje)` con el `puntaje` que el usuario de índice `usuario_Id` le asignó a la película de índice `pelicula_Id`. El `puntaje` es un valor entre 0.5 y 5 (con incrementos de 0.5 puntos). El archivo `puntajes_test.csv` contiene una lista similar con 5000 puntajes reservados para evaluar el desempeño de los algoritmos implementados.

Parte 1 - Filtrado basado en contenidos

Intentaremos predecir el puntaje r_{ui} que un usuario u le asigna a la película i, basándonos en la clasificación en géneros de la misma. Así, a cada película i le asociamos un vector q_i de 0's y 1's donde en la posición k ponemos 1 si dicha película es del género k, y 0 en caso contrario.

Por ejemplo, la película *8684, Star Wars: Episode VII - The Force Awakens (2015)* está clasificada como *Action, Adventure, Fantasy, Sci-Fi*. Luego, tenemos

$$q_{8684} = (1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0).$$

Cada fila i del archivo `clasificacion_peliculas.csv` contiene el índice i de una película (primer columna) y su vector q_i correspondiente.

A cada usuario u le asignaremos un vector de *gustos* $p_u = (a_1, a_2, \ldots, a_{18})$ donde a_k es la ponderación (puntaje) que le da al género k. Así, estimamos que u le dará a la película i el puntaje $\hat{r}_{ui} = \langle p_u, q_i \rangle$. Como los q_i son fijos (dados por los géneros de cada película) queremos encontrar p_u que minimice el error cuadrático entre los puntajes estimados por nuestro modelo y los puntajes reales (los del conjunto de datos) para dichas películas:

$$p_u^* = \arg \min_{p_u \in \mathbb{R}^{18}} \left[\sum_{i \in I_u} (\langle p_u, q_i \rangle - r_{ui})^2 \right] \tag{1}$$

donde I_u es el subconjunto de películas (índices) para los cuales disponemos del puntaje r_{ui} en el conjunto de datos. Luego de encontrado dicho vector óptimo p_u^*, estimamos el puntaje que le dará a una película $i \notin I_u$ que aún no vio por $\langle p_u, q_i \rangle$, y ordenando en forma decreciente estas películas según este puntaje estimado, generamos una lista de recomendación para este usuario.

En esta parte, se pide:

1. Observar que definiendo la matriz Q_u cuyas *filas* son los vectores q_i para $i \in I_u$ (las películas para las cuales tenemos el puntaje r_{ui} que el usuario u le dio) y r_u el vector con los puntajes que dio el usuario u a dichas películas, el problema anterior se puede formular así:

   $$p_u^* = \arg \min_{X \in \mathbb{R}^{18}} \|Q_u X - r_u\|^2 \tag{2}$$

   Hallar el vector óptimo p_u^* a partir de (2) para el usuario $u = 4$, usando sólo los puntajes que se encuentran en el archivo `puntajes_ajuste.csv`.

2. En el archivo `puntajes_test.csv` hay algunos puntajes que dio el usuario $u = 4$ y que no se usaron en el ajuste, los cuales pueden usarse ahora para evaluar el desempeño de nuestro algoritmo de recomendación. Calcular el puntaje $\hat{r}_{ui}$ que estima el algoritmo y comparar con los puntajes verdaderos, tanto en los datos de ajuste como en los datos test.

Parte 2 - Filtrado colaborativo

Como se habrá apreciado, el modelo implementado en la Parte 1 no es muy performante. Modificando el algoritmo anterior, ahora también los vectores q_i son desconocidos. Es decir, los vectores q_i que representan a cada película i no se eligen de antemano (como en la parte anterior que se basó en los géneros de las películas) sino que también se estimarán a partir de los datos. Y esta vez, usaremos todos los datos del conjunto para estimar todos los vectores p_u y q_i a la vez (*filtrado colaborativo*).

Formalmente, elegimos $f \in \mathbb{N}$ la dimensión de los vectores p_u y q_i (con cuántas *características* se representa a cada película) y denotamos P y Q las matrices cuyas filas son los vectores p_u y q_i respectivamente, por lo que buscamos

$$(P^*, Q^*) = \arg\min_{P,Q} \left[\sum_{u=1}^{M} \sum_{i \in I_u} (\langle p_u, q_i \rangle - r_{ui})^2 \right] \tag{3}$$

M es la cantidad de usuarios, N es la cantidad de películas.

La estrategia para encontrar las matrices P y Q será la *minimización alternada*:

- Inicializar matrices $P^{(0)} \in \mathcal{M}_{M \times f}$ y $Q^{(0)} \in \mathcal{M}_{N \times f}$ (por ejemplo, aleatorias).
- Para $k \geq 0$:
  - $p_u^{(k+1)} = \arg\min_{X \in \mathbb{R}^f} \|Q_u^{(k)} X - r_u\|^2$
  - $q_i^{(k+1)} = \arg\min_{Y \in \mathbb{R}^f} \|P_i^{(k+1)} Y - r_i\|^2$

donde $Q_u^{(k)}$ tiene como *filas* a los vectores $q_i^{(k)}$ de las películas i que puntuó el usuario u (algunas filas de $Q^{(k)}$), r_u tiene los puntajes correspondientes a dichas películas, $P_i^{(k)}$ es la matriz cuyas *filas* son los vectores $p_u^{(k)}$ de los usuarios que puntuaron la película i (algunas filas de $P^{(k)}$), y r_i es el vector de dichos puntajes.

1. Implementar el algoritmo de minimización alternada descrito anteriormente, usando el script `filtrado_colaborativo.m` como base, en el cual se calcula el error promedio cometido en las estimaciones, tanto en los datos de ajuste como en test, en cada iteración. El *error cuadrático medio* (cuyo cálculo se implementó en `error_cuadratico_medio.m`) está dado por

   $$e(P, Q) = \frac{1}{T} \sum_{u=1}^{M} \sum_{i \in I_u} (\langle p_u, q_i \rangle - r_{ui})^2$$

   donde T es el total del puntajes en cada caso (datos de ajuste o test). Utilice el criterio de parada que estime conveniente.

2. Ejecutar el algoritmo anterior para varios valores de f. Comparar el error que comete el recomendador (luego de finalizado el ajuste de P y Q) para el usuario $u = 4$ sobre los datos de ajuste y los datos de test. ¿Estima bien los puntajes de ajuste? ¿Y los de test?

3. Una mejora del algoritmo anterior es agregar *regularización de Tychonoff* [4]: penalizar que los vectores p_u y q_i tengan norma grande para evitar que el recomendador se *sobreajuste* a los datos de ajuste [5]:

   $$(P^*, Q^*) = \arg\min_{P,Q} \left[\sum_{u=1}^{M} \sum_{i \in I_u} (\langle p_u, q_i \rangle - r_{ui})^2 + \lambda(\|p_u\|^2 + \|q_i\|^2) \right] \tag{4}$$

   donde $\lambda \geq 0$ es un parámetro que regula la penalización que se le da a las normas antes mencionadas.

Se puede modificar el algoritmo de la parte anterior para que contemple este nuevo término:

- $p_u^{(k+1)} = \arg\min_{X \in \mathbb{R}^f} \|\tilde{Q}_u^{(k)} X - \tilde{r}_u\|^2$
- $q_i^{(k+1)} = \arg\min_{Y \in \mathbb{R}^f} \|\tilde{P}_i^{(k+1)} Y - \tilde{r}_i\|^2$

donde $\tilde{Q}_u^{(k)} = \begin{pmatrix} Q_u^{(k)} \\ \sqrt{\lambda} I_f \end{pmatrix}$, $\tilde{P}_i^{(k)} = \begin{pmatrix} P_i^{(k)} \\ \sqrt{\lambda} I_f \end{pmatrix}$, $\tilde{r}_u = \begin{pmatrix} r_u \\ \vec{0}_f \end{pmatrix}$ y $\tilde{r}_i = \begin{pmatrix} r_i \\ \vec{0}_f \end{pmatrix}$.

(Aquí I_f es la matriz identidad $f \times f$ y $\vec{0}_f$ es el vector nulo de $\mathbb{R}^f$.)

Implementar esta mejora al algoritmo, y reestimar P y Q para varios valores de f y λ. Comparar el rendimiento del recomendador (luego de estimadas P y Q) para el usuario $u = 4$ en los datos de ajuste y de test, con el algoritmo sin regularización (parte anterior) y con el implementado en la Parte 1 (filtrado basado en contenidos).

Fecha límite de entrega: Viernes 29 de noviembre, 23:55 hs.

Referencias

[1] `https://en.wikipedia.org/wiki/Netflix_Prize`

[2] `https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf`

[3] `https://grouplens.org/datasets/movielens/latest/`

[4] `https://es.wikipedia.org/wiki/Regularizaci%C3%B3n_de_T%C3%ADjonov`

[5] `https://es.wikipedia.org/wiki/Sobreajuste`