Técnicas de recomendación

PID_00264731

Xavi Font

Tiempo mínimo de dedicación recomendado: 3 horas

Universitat Oberta de Catalunya

Xavi Font

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Jordi Ayza Graells (2019)

Primera edición: marzo 2019

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Oberta UOC Publishing, SL

Índice

Introducción

¿Qué importancia tienen los sistemas de recomendación para despertar tanto interés? La realidad es que detrás del auge de los sistemas de recomendación hay algunas grandes empresas como Netflix. Su popular premio de 1 millón de dólares al equipo que mejor sistema de predicción/recomendación realizara queda ya unos años atrás (competición abierta en Kaggle, 2009). Los diferentes equipos debían predecir los *ratings* de los usuarios sobre nuevos títulos a partir de los *ratings* de películas ya vistas.

La realidad es que parte del éxito de muchas empresa reside en su habilidad para realizar buenas recomendaciones. Pensemos en el caso de Amazon. El sistema de recomendación consigue incrementar el valor percibido de los usuarios por ofrecer propuestas que se ajustan a nuestras necesidades. Otros casos notables de uso de sistemas de recomendación se pueden encontrar en el sector musical. El caso de Spotify es muy interesante. En principio, el sistema de recomendación tiene como objetivo satisfacer al cliente, pero también conseguir mejorar los resultados de Spotify. Por esta razón el sistema de recomendación de Spotify tiende a recomendar grupos musicales emergentes. Con esta recomendación, Spotify consigue reducir sustancialmente los costes asociados a la licencia (las condiciones son más ventajosas para Spotify que con otros grupos más conocidos y con un coste de licencia muy superior).

¿Dónde podemos encontrar más ejemplos de sistemas de recomendación? En el sector inmobiliario, donde se deben ajustar las peticiones de los potenciales compradores con las diferentes ofertas. Hay multitud de ejemplos en este sector: Idealista, etc.

Otros sectores, como algunos relacionados con la relaciones sociales, se basan en algoritmos de recomendación eficientes para encontrar a tu media naranja. La misma idea que subyace tras cliente/usuario frente a preferencia/producto se aplica de forma similar. La empresa eHarmony publicita como garantía de buenas recomendaciones para sus clientes: *Cada pareja recomendada compatible se preselecciona usando 29 categorías de análisis*. El éxito radica en que la recomendación ofrecida sea buena y dé resultados satisfactorios.

Si bien existen dos aproximaciones básicas para los sistemas de recomendación, como son los sistemas de filtrado colaborativo y los sistema basados en contenido, este módulo se centrará principalmente en la primera aproximación, que es una de las más utilizadas (en sus dos variantes).

Este sería el esquema de opciones que podemos encontrar:

- *Collaborative filtering*
  - Basado en usuario
  - Basado en ítem (producto)
- *Content based*
- Otras variantes: *knowledge-base* (*case-based*)

Otro de los aspectos que cabe considerar es ¿bajo que métrica se realiza la recomendación? En otras palabras, qué característica o variable es objeto de optimización:

- incrementar ventas,
- reducir costes,
- retención de usuarios,
- páginas visitadas,
- enlaces visitados,
- satisfacción del usuario...

En general, si se consideran aspectos de escalabilidad, no todas las aproximaciones tienen la misma factura de coste operacional. Es importante evaluar el coste y el retorno de la inversión para que el sistema finalmente se ajuste a las necesidades de la organización.

Objectivos

Los principales objetivos que se persiguen en este módulo son los siguientes:

1. Entender el planteamiento del problema general de cualquier sistema de recomendación.
2. Entender el enfoque de filtrado colaborativo y sus dos variantes: la aproximación basada en usuario y la basada en ítem.
3. Visualizar posibles problemas que esta solución puede tener y cómo se pueden mitigar.
4. Realizar un proceso de recomendación usando un caso real (a través de la librería recommenderlab).
5. Validar los resultados del sistema de recomendación.

1. Importancia y planteamiento del problema

1.1. Introducción

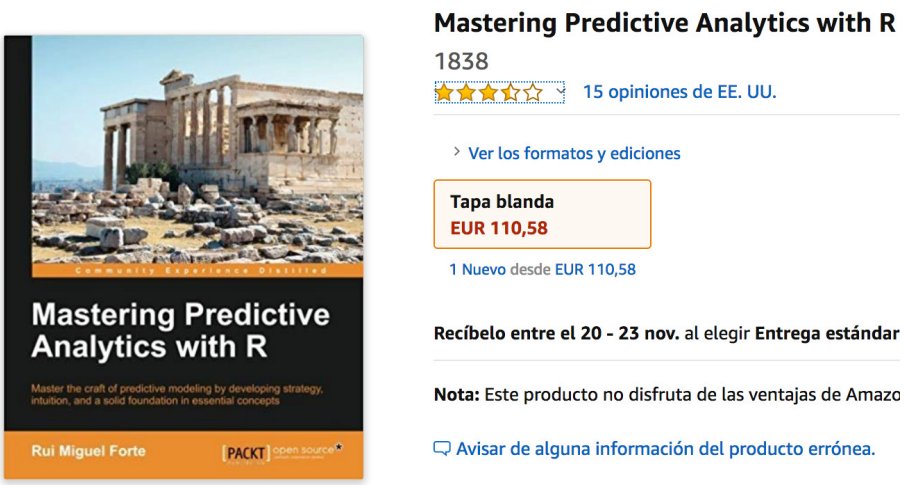
Hay una gran colección de situaciones donde es necesaria la articulación de valoraciones (*ratings*) sobre una serie de conceptos. ¿Qué conceptos son sujetos de esta valoración?

- Películas de un proveedor de *streaming* tipo Netflix: si pueden valorar películas de cartelera, series o documentales, entre otros.
- Libros de una librería tipo Amazon: este es un ejemplo típico de valoración que los usuarios realizan sobre los libros dando un número de estrellas (*ratings* 1-5). Valores bajos indican libro no recomendado, valores altos indican libro recomendado.
- Canciones de un proveedor de *streaming* tipo Spotify: se pueden valorar las canciones y los grupos para facilitar conocimiento de las preferencias de los clientes.
- Restaurantes de una cadena o de una ciudad.
- Hoteles.
- Lugares turísticos.
- Universidades/grados.

Esta no es una lista completa de todos los conceptos que pueden ser utilizados para valorar y, por tanto, sujetos a la resolución de una recomendación. En general, los problemas de recomendación realizarán una previsión a partir del conjunto ya disponible de recomendaciones utilizando cualquier estrategia estadística de extracción de conocimiento.

Si observamos el caso concreto de Amazon, los usuarios pueden visualizar el promedio de los *ratings* realizados por clientes de amazon que han adquirido previamente este producto. En la figura 1 se muestra un libro con unas recomendaciones promedio que pueden ayudar a decidir sobre la compra.

Figura 1. Sistema de valoración de Amazon con un *rating* de estrellas



En ocasiones, empresas como Amazon no solo disponen de las valoraciones realizadas por su cartera de clientes, sino también de una información adicional y que en ocasiones es fundamental tanto para Amazon como para los propios clientes. Esta información adicional corresponde a los comentarios que pueden realizar los clientes sobre el producto que han adquirido. Esta información textual (ver figura 2) puede ser utilizada por los sistemas de recomendación para mejorar el rendimiento y la eficacia en sus pronósticos.

Figura 2. Valoración textual vía comentarios que Amazon pone a disposición de cualquiera de sus clientes/usuarios que han adquirido el producto

M. Bryant

Good book if your at the right level. Read further.

6 de noviembre de 2016 - **Publicado en Amazon.com**

Compra verificada

Into chapter 2 right now, so I will update review later. But so far so good. This book is the for the following audience:

*have some experience with R, at least understand basic code, but not much more needed
*have some experience with statistics. this isn't entry level, but its not advanced either. I've take a few analytics courses, and a good amount of this is review, but it serves well as a refresher and bringing a little more to the table.
*want to get introduced to decision tree analysis and other models outside of regression (MNL, multivariable regression etc).
*want to learn a little more about R and get introduced to good R packages for predictive analytics.

Overall good buy so far, I am not disappointed

Imaginemos una lista de empresas y cuestionemos si utilizan o pueden utilizar sistemas de recomendación:

- Uber
- Nike
- Seat
- Vueling

Para visualizar si tenemos un problema de recomendación, vamos a detallar y describir su formalización así como las diferentes aproximaciones de solución.

1.2. Planteamiento del problema

En primer lugar, debemos entender que tenemos dos conceptos básicos en cualquier sistema de recomendación. Por un lado, los usuarios; podemos imaginar que tenemos *m* usuarios o clientes y, por otro lado, tenemos el segundo concepto que hace referencia a los objetos que adquiere, utiliza, visita, ve, etc., que habitualmente se denominan ítems. Tenemos una lista de ítems, *n* ítems listos para ser valorados por los usuarios.

Esta estructura usuario-ítem es la que define el problema que deberemos resolver. El sistema de recomendación obtiene valoraciones de los usuarios sobre el conjunto de los ítems disponibles.

1.2.1. Tipos de valoraciones

¿De qué modo se realizan las valoraciones? Depende de la situación, pero podemos encontrar diferentes formas de evaluar:

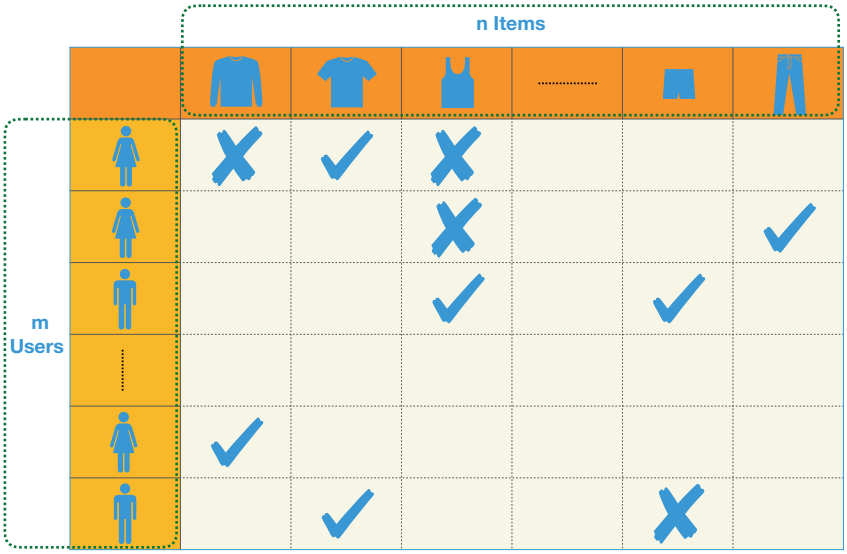
- Con una escala de valoración o *rating*: valoramos de 1 a 5, asociando el valor más bajo a una situación de poca valoración y que, por tanto, no sería recomendado por este usuario y el valor alto en el otro extremo, dándole mucha importancia e indicando recomendación.

- Valoración binaria: con un me gusta frente a no me gusta. En estos casos puede tener incidencia cómo codificamos el no me gusta respecto al no lo he valorado.

- Con una acción realizada implícitamente por el usuario, por ejemplo al realizar una compra ya puede interpretarse con que recomendaría el producto. O si visita una página podríamos interpretar que está valorando la página. Obviamente, si nunca más realiza esta acción, quizá no deberíamos computarla como una buena recomendación.

En la figura 3 se visualiza una situación estandarizada del problema de recomendación. La matriz muestra una valoración de usuarios tipo me gusta frente a no me gusta. En otras situaciones la matriz usuario-ítem representará un valor de rango o puntuación.

Figura 3. Visualización de la matriz usuario-ítem base de los sistemas de recomendación



1.2.2. Información adicional para el sistema de recomendación

Para que el sistema de recomendación realice buenas predicciones, quizá no es suficiente con la matriz ítem-usuario. Incrementar la información que hay que suministrar al sistema puede mejorar sustancialmente la eficiencia de las recomendaciones del sistema. Entre estas informaciones adicionales que el sistema podría incorporar cabe destacar las siguientes:

- **Características de segmentación**: entre las variables típicas que ayudan a segmentar están el género, la edad, el estado civil, el número de hijos y el nivel de educación, entre otras. La forma como obtenemos esta información no siempre es la misma, pero habitualmente la facilita el propio usuario cuando se da de alta de un servicio.

- **Características de contenido**: basadas en el análisis textual de las valoraciones que realiza el usuario sobre un ítem. En este caso, esta información adicional es procesada (vía *text-mining*) e introducida en el sistema de recomendación.

1.3. Objetivo y salida de un sistema de recomendación

El objetivo de cualquier sistema de recomendación es:

- predecir las valoraciones sobre todos los ítems y
- generar una lista con las mejores (*top*) recomendaciones.

Formalmente, disponemos de un conjunto de usuarios:

$$\mathcal{U} = \{u_1, u_2, \ldots, u_{m-1}, u_m\}$$

y de un conjunto de ítems:

$$\mathcal{I} = \{i_1, i_2, \ldots, i_{n-1}, i_n\}$$

que se relacionan a través de una matriz que denominamos matriz usuario-ítem, que definimos como:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ r_{21} & r_{22} & \ldots & r_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ r_{m1} & r_{m2} & \ldots & r_{mn} \end{bmatrix}$$

El conjunto de las valoraciones las tenemos almacenadas en la matriz $\mathbf{R}$, donde el valor r_{jk} representa la valoración del usuario j sobre el ítem k. Un ejemplo de las valoraciones que podemos encontrar se muestra en la siguiente matriz:

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i1	i11	i12
u1	2			5		4		3	1			
u2					1	3					2	
u3							2	1				3
u4	2	1					4			3		
u5		2			4				3		1	
u6		4	3					2	1		5	

La realidad de este tipo de problemas queda mínimamente reflejada en la anterior tabla. Por un lado, el número de ítems es muy superior al número de usuarios (podemos tener un millón de títulos de libros). Estos últimos son un número considerablemente menor que el número de ítems (podemos tener 10.000 usuarios). Otra de las problemáticas representada es la cantidad de valores desconocidos (*missing values*).

Una de las medidas que podemos considerar es el coeficiente de *sparsity* o escasez, es decir, la proporción de valores que son no nulos respecto al total.

$$Coef_{sparsity} = \frac{\#r_{jk} | r_{jk} \neq 0}{m \times n}$$

Es evidente que sin una matriz de valoraciones, también llamada matriz de utilidades, ningún sistema de recomendaciones puede resolver el problema.

¿Cuál es la salida de un sistema de recomendación?

Podemos imaginar una primera opción donde el sistema de recomendación genera una predicción para el usuario sobre un ítem determinado. Si imaginamos este usuario como el usuario a: u_a y el ítem que predecir como i_j, entonces obtenemos una predicción sobre r_{aj}. Podemos entender que en realidad el problema que estamos resolviendo es un problema de regresión dado que a partir de un conjunto de variables se explica un determinado ítem. Esta forma de recomendación también se denomina *individual scoring*.

Una segunda opción a la salida del sistema de recomendación es una lista con las mejores recomendaciones para el usuario. A menudo se conoce como *Top-N recommendation* o también *Ranked Scoring*.

En cualquier caso, debemos garantizar que la solución obtenida por el sistema de recomendación se ajuste a las necesidades de los objetivos que el negocio se plantea. Es importante considerar alguno de los problemas que se pueden generar:

- **Validez de las recomendaciones**: se deben evitar los falsos positivos. Esto es, ítems que el sistema recomienda e identifica como positivos pero que el usuario no quiere. Es decir, la realidad para el usuario es de ítem negativo. Si esto sucede, este usuario no estará muy contento con la recomendación y a la larga puede dejar de ser cliente.
- ***Sparsity***: una matriz con un nivel exageradamente grande de valores no disponibles puede suponer una degradación del sistema de recomendación.
- **Escalabilidad**: es el efecto de la maldición de la dimensión (*curse of dimensionality*). Nuestro sistema puede tener problemas al inicio, no tenemos usuarios (problema conocido como el *cold start problem*) y también cuando pasamos de unos miles a varios millones de usuarios. Este notable salto en el crecimiento de los elementos involucrados en la construcción de la matriz puede provocar que el sistema deje de ofrecer respuestas en tiempo real o razonable.

1.4. Aproximación intuitiva

Si calculamos la proximidad entre usuarios, podemos identificar a los usuarios más próximos y, por tanto, podemos entender cómo realizar una recomendación.

Si calculamos la distancia entre usuarios a través de la distancia euclídea, obtenemos una matriz de distancias que permite interpretar que los usuarios u_3 y u_2 están muy relacionados. En negrita se han indicado las relaciones más distantes y en cursiva, las más próximas.

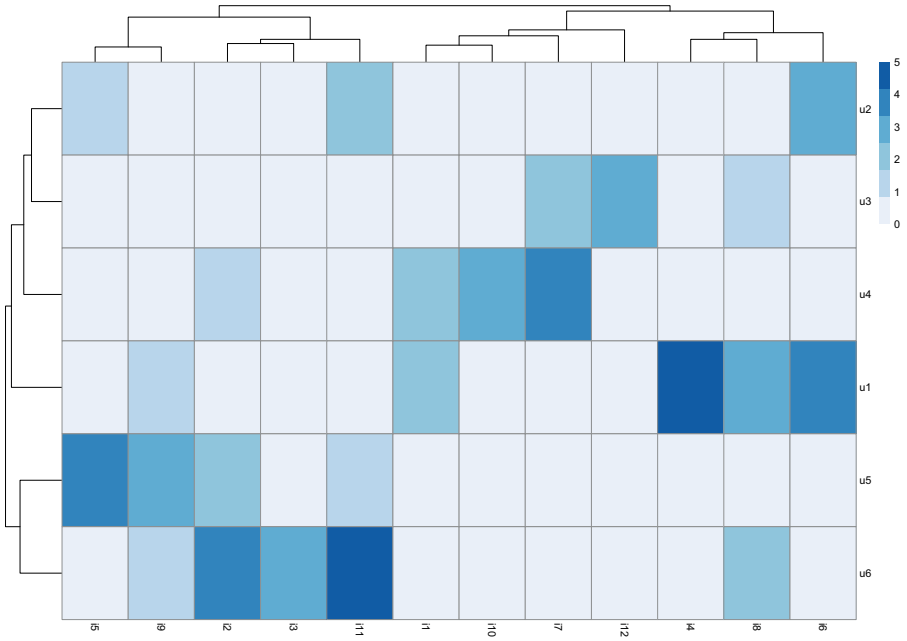
	u1	u2	u3	u4	u5	u6
u1	0.000000	6.708204	7.937254	8.774964	8.888194	9.797959
u2	6.708204	0.000000	5.291503	6.633250	5.656854	7.000000
u3	7.937254	*5.291503*	0.000000	5.291503	6.633250	8.062258
u4	8.774964	6.633250	*5.291503*	0.000000	7.483315	8.774964
u5	**8.888194**	*5.656854*	6.633250	7.483315	0.000000	7.280110
u6	**9.797959**	7.000000	8.062258	**8.774964**	7.280110	0.000000

Este cálculo se puede repetir para los ítems. Vemos que los ítems más próximos son el *i*5 y el *i*9.

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12
i1	0.00	5.00	4.12	3.61	5.00	4.12	3.46	3.16	3.87	2.24	6.16	4.12
i2	5.00	0.00	2.45	6.78	4.69	6.78	5.74	4.36	3.46	4.90	2.65	5.48
i3	4.12	*2.45*	0.00	5.83	5.10	5.83	5.39	3.32	3.74	4.24	3.00	4.24
i4	3.61	6.78	5.83	0.00	6.48	3.16	6.71	3.00	5.10	5.83	7.42	5.83
i5	5.00	4.69	5.10	6.48	0.00	6.00	6.08	5.57	2.00	5.10	5.92	5.10
i6	4.12	6.78	5.83	3.16	6.00	0.00	6.71	3.87	5.29	5.83	6.56	5.83
i7	3.46	5.74	5.39	6.71	6.08	6.71	0.00	5.48	5.57	2.24	7.07	4.12
i8	3.16	4.36	3.32	3.00	5.57	3.87	5.48	0.00	3.87	4.80	4.90	4.12
i9	3.87	3.46	3.74	5.10	*2.00*	5.29	5.57	3.87	0.00	4.47	5.00	4.47
i10	*2.24*	4.90	4.24	5.83	5.10	5.83	2.24	4.80	4.47	0.00	6.24	4.24
i11	6.16	*2.65*	*3.00*	7.42	5.92	6.56	7.07	4.90	5.00	6.24	0.00	6.24
i12	4.12	5.48	4.24	5.83	5.10	5.83	4.12	4.12	4.47	4.24	6.24	0.00

Esta información se puede visualizar de forma gráfica para facilitar la proximidad entre usuarios e ítems. En la figura 4 se pueden identificar los usuarios más próximos, así como los ítems que también tienen más elementos en común.

Figura 4. Visualización de la matriz de distancias usuario-ítem junto a las agrupaciones realizadas vía una agrupación aglomerativa a través de las distancias euclídeas



Aproximación con otras distancias

En ocasiones se utiliza la distancia del coseno. Esta medida refleja el ángulo entre los dos vectores involucrados.

$$d_{cos} = \frac{u_i \cdot u_j}{\|u_i\|\|u_j\|}$$

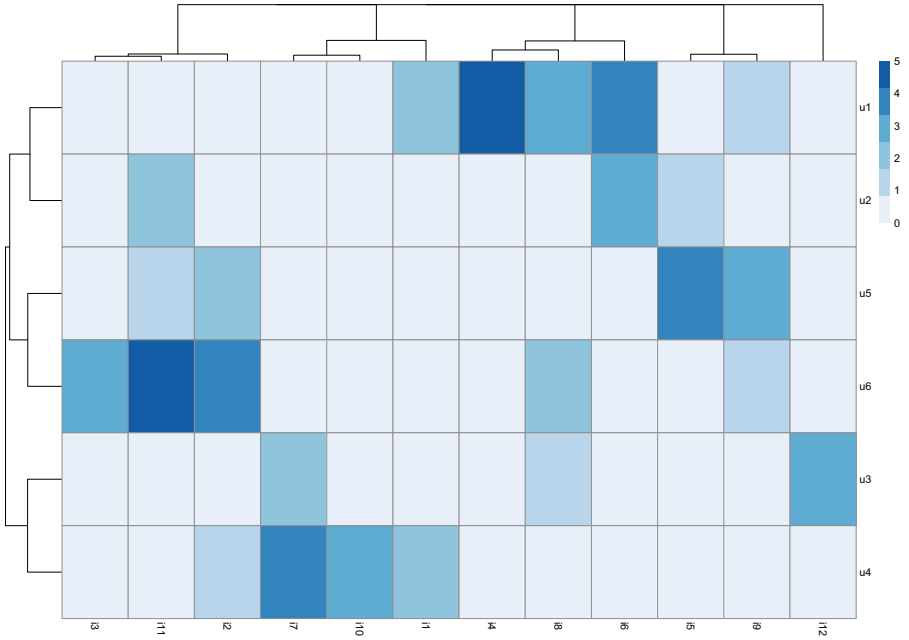
Si recalculamos las distancias, obtendremos otra matriz.

Para los usuarios obtendremos:

	u1	u2	u3	u4	u5	u6
u1	0.0000000	0.567550	0.8918875	0.9015268	0.9261451	0.8727273
u2	*0.5675500*	0.000000	1.0000000	1.0000000	0.7072300	0.6396250
u3	0.8918875	1.000000	0.0000000	0.6096400	1.0000000	0.9279250
u4	0.9015268	1.000000	0.6096400	0.0000000	0.9333333	0.9015268
u5	0.9261451	0.707230	1.0000000	0.9333333	0.0000000	0.6061072
u6	0.8727273	0.639625	0.9279250	0.9015268	*0.6061072*	0.0000000

Donde un valor de 0 indica que los vectores son paralelos y un valor de 1 indica que los vectores son perpendiculares. Podemos así interpretar que los usuarios más próximos corresponden al par u_2 y u_1. A continuación tendríamos el par u_6 y u_5.

Figura 5. Visualización de la matriz de distancias usuario-ítem junto con las agrupaciones realizadas vía una agrupación aglomerativa a través de la distancia del coseno



Cabe realizar una última consideración relativa a las distancias cuando la matriz es de tipo binario. La distancia de Jaccard se basa en el índice de Jaccard, que expresa la similitud entre dos conjuntos y se define como la cardinalidad de la intersección dividido por la cardinalidad de la unión.

$$J_{AB} = \frac{|A \cap B|}{|A \cup B|}$$

Si queremos obtener una medida de distancia, se usa el complementario de este índice. Y por tanto la distancia queda como:

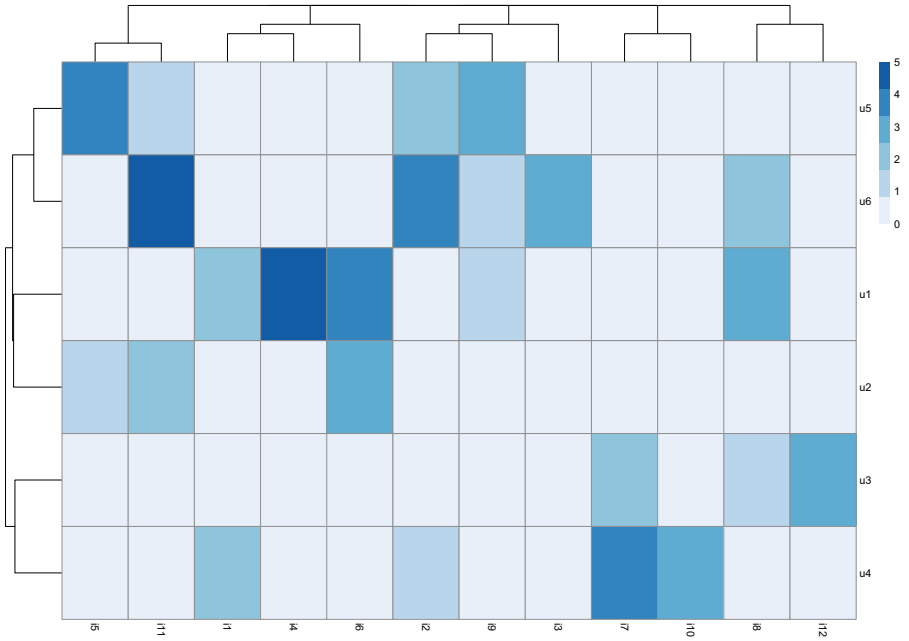
$$d_{jaccard} = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Utilizando el mismo ejemplo de inicio de sección podemos visualizar la matriz de distancias de Jaccard:

	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11	i12
i1	0.00	5.00	4.12	3.61	5.00	4.12	3.46	3.16	3.87	2.24	6.16	4.12
i2	5.00	0.00	2.45	6.78	4.69	6.78	5.74	4.36	3.46	4.90	2.65	5.48
i3	4.12	2.45	0.00	5.83	5.10	5.83	5.39	3.32	3.74	4.24	3.00	4.24
i4	3.61	6.78	5.83	0.00	6.48	3.16	6.71	3.00	5.10	5.83	7.42	5.83
i5	5.00	4.69	5.10	6.48	0.00	6.00	6.08	5.57	2.00	5.10	5.92	5.10
i6	4.12	6.78	5.83	3.16	6.00	0.00	6.71	3.87	5.29	5.83	6.56	5.83
i7	3.46	5.74	5.39	6.71	6.08	6.71	0.00	5.48	5.57	2.24	7.07	4.12
i8	3.16	4.36	3.32	3.00	5.57	3.87	5.48	0.00	3.87	4.80	4.90	4.12
i9	3.87	3.46	3.74	5.10	2.00	5.29	5.57	3.87	0.00	4.47	5.00	4.47
i10	2.24	4.90	4.24	5.83	5.10	5.83	2.24	4.80	4.47	0.00	6.24	4.24
i11	6.16	2.65	3.00	7.42	5.92	6.56	7.07	4.90	5.00	6.24	0.00	6.24
i12	4.12	5.48	4.24	5.83	5.10	5.83	4.12	4.12	4.47	4.24	6.24	0.00

Igual que en el ejemplo anterior, se puede visualizar la relación entre usuarios e ítems de una forma más visual con la figura 6.

Figura 6. Visualización de la matriz de distancias usuario-ítem junto con las agrupaciones realizadas vía una agrupación aglomerativa a través de la distancia de Jaccard



2. Modelos basados en filtrado colaborativo

2.1. Introducción

Los sistemas de filtrado colaborativo (*collaborative filtering*, CF) usan la matriz de utilidades (valoraciones) para generar las recomendaciones. Evidentemente, la utilización de toda la matriz puede comportar problemas de dimensionalidad.

Existen dos aproximaciones:

- ***Memory-Based CF***: en esta aproximación se utiliza toda la matriz de recomendaciones. La opción más extendida es conocida como *user-based CF*.

- ***Model-Based CF***: en esta aproximación primero se crea un modelo de preferencias para los usuarios y posteriormente se usa este modelo para generar recomendaciones. En este enfoque encontramos los modelos *item-based CF*.

2.2. Filtrado colaborativo basado en usuario

La idea detrás de este tipo de algoritmos es aplicar el concepto de pasar la voz (boca a boca) analizando la totalidad de las valoraciones que los usuarios han realizado sobre el conjunto de ítems. La suposición de este modelo es que usuarios con preferencias similares valorarán de manera similar. Esta suposición es importante para entender el procedimiento que seguirán este tipo de soluciones.

Si faltan valoraciones, utilizaremos el promedio que ofrece un conjunto de usuarios próximos. Es decir, a partir de una distancia podemos agrupar, y los usuarios próximos se utilizan para promediar la valoración que falta.

2.2.1. Formalización

Sea un usuario definido u_a, calculamos para este usuario la vecindad:

$$\mathcal{N}(u_a) \subset \mathcal{U}$$

El conjunto de usuarios permite agregar las valoraciones para obtener una predicción para el usuario u_a. Es decir:

$$\hat{r_{aj}} = \frac{1}{|\mathcal{N}(u_a)|} \sum_{i \in \mathcal{N}(u_a)} r_{ij}$$

Para crear una lista de recomendaciones top-N, los ítems se ordenan según la predicción de la valoración. Esta lista puede adaptarse a un entorno más real definiendo un umbral (*threshold*) para descartar los ítems con valores más bajos.

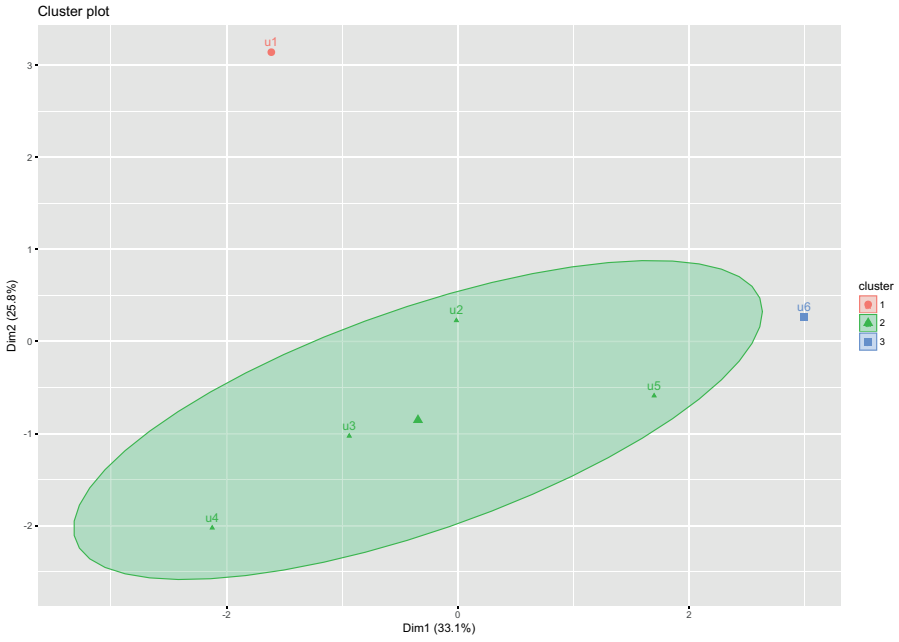
El hecho de que algunos usuarios estén más próximos a nuestro usuario u_a permite modificar la anterior ecuación para incorporar el peso relativo a la proximidad que este tenga. Usuarios próximos tendrán mayor peso en el valor promedio que se calcula.

Es decir, si definimos s_{ai} como el grado de similitud entre el usuario u_a y el u_i, el promedio ponderado por el peso quedará como:

$$\hat{r_{aj}} = \frac{1}{\sum_{i \in \mathcal{N}(u_a)}} \sum_{i \in \mathcal{N}(u_a)} s_{ai} \times r_{ij}$$

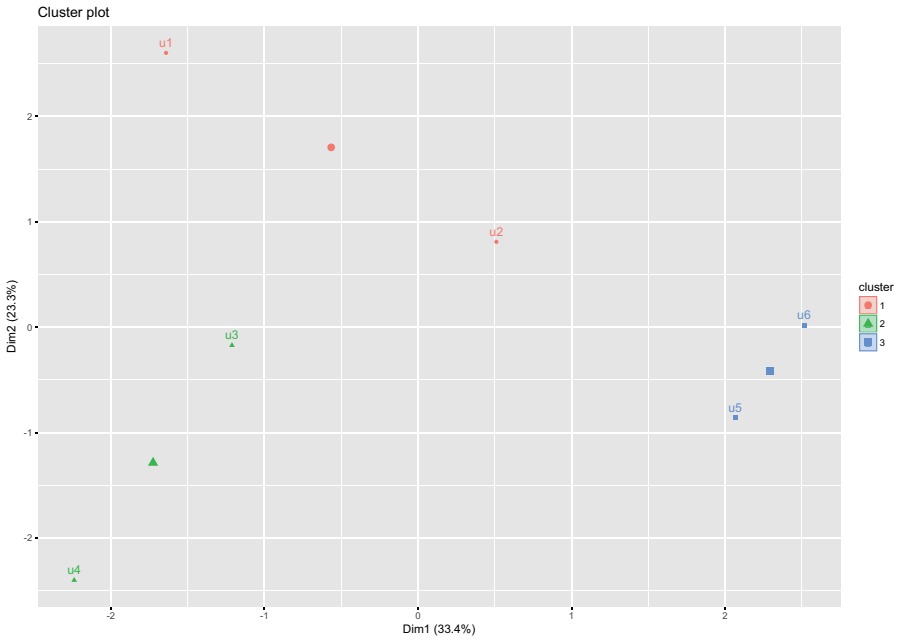
En la figura siguiente se puede apreciar la aproximación descrita. Para predecir los valores de *ratings* utilizaremos los usuarios que están más cerca del usuario u_3. Es posible ponderar proporcionalmente a la distancia a este usuario.

Figura 7. Visualización de la distribución de usuarios vía k-means mostrando una agrupación próxima al usuario u_3



En el caso de realizar un escalado, la representación puede variar. A continuación mostramos que en este caso el usuario más próximo a u_3 es el u_4.

Figura 8. Visualización de la distribución de usuarios vía k-means con un escalado previo de los datos para evitar posible *bias*, y mostrando una agrupación próxima al usuario u_3



2.2.2. Alternativas y mejoras

En algunas situaciones el rendimiento del sistema de recomendación puede mejorar aplicando alguna de estas transformaciones:

- **Reducción del *bias* en las valoraciones de usuario:** $r_{ij} - \overline{r_i}$. Esta operación es un centrado sobre los usuarios.

- **Reducción a través de la transformación z-score**. En este caso centramos y reducimos (dejamos las valoraciones por usuario con media 0 y desviación1).

2.2.3. Ventajas e inconvenientes

La principal desventaja de este modelo es el hecho de mantener toda la matriz de valoraciones. Del mismo modo, el coste computacional de calcular las similitudes entre los usuarios no es despreciable.

Como ventaja encontramos la sencillez del procedimiento.

2.3. Filtrado colaborativo basado en ítems

Los modelos bajo esta aproximación producen recomendaciones basadas en las relaciones entre ítems. Este procedimiento se basa en la suposición de que los usuarios preferirán ítems similares a los que ellos prefieren.

Qué procedimiento se sigue en este caso:

- Cálculo de la matriz de similitud entre todos los ítems.
- La matriz calculada de dimensión $n \times n$ se intenta reducir.
- Definimos un valor k siendo este k mucho más pequeño que n :$k <<< n$.
- Se guardan los k ítems más próximos/similares al ítem i, que podemos representar como $\mathcal{S}(i)$.

El hecho de mantener únicamente k similitudes por ítem permite reducir los problemas de memoria y computación proporcionalmente a la reducción que aporta k sobre el total n.

Nota

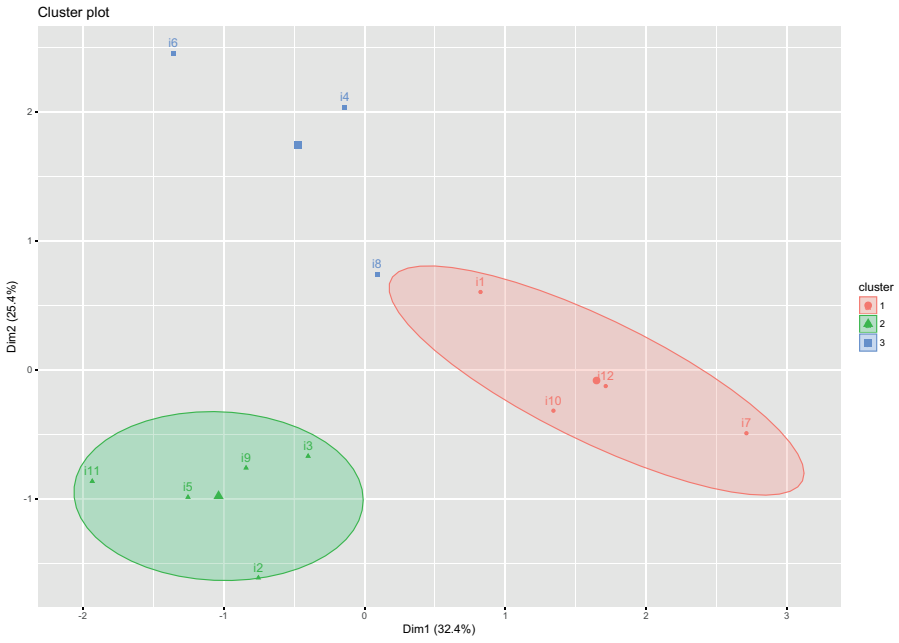
Lo de proporcional no es del todo cierto en caso del coste computacional asociado.

2.3.1. Formalización

Sea un usuario definido u_a:

$$\hat{r_{ai}} = \frac{1}{\sum_{j \in \mathcal{S}(i) \cap l;\ r_{al} \neq ?} S_{ij}} \sum_{j \in \mathcal{S}(i) \cap l;\ r_{al} \neq ?} s_{ij} \times r_{aj}$$

Figura 9. Visualización de la distribución de ítems vía k-means con un escalado previo de los datos para evitar posible *bias*, y mostrando agrupación



3. Modelos basados en ítem frente a usuario

3.1. Introducción

Para entender las dos aproximaciones vamos a desarrollar un ejemplo que corresponde a unos datos ya disponibles dentro de la librería recommenderlab.

Los datos se recopilaron a través del sitio web de MovieLens* durante un periodo de siete meses (desde el 19 de septiembre de 1997 hasta el 22 de abril de 1998). El conjunto de datos contiene aproximadamente 100.000 calificaciones en un rango comprendido entre 1 y 5 (1-5) de 943 usuarios sobre 1.664 películas.

* movielens.umn.edu

Para poder acceder a los datos, podemos realizar las siguientes operaciones:

```
data("MovieLense")
str(MovieLense)
```

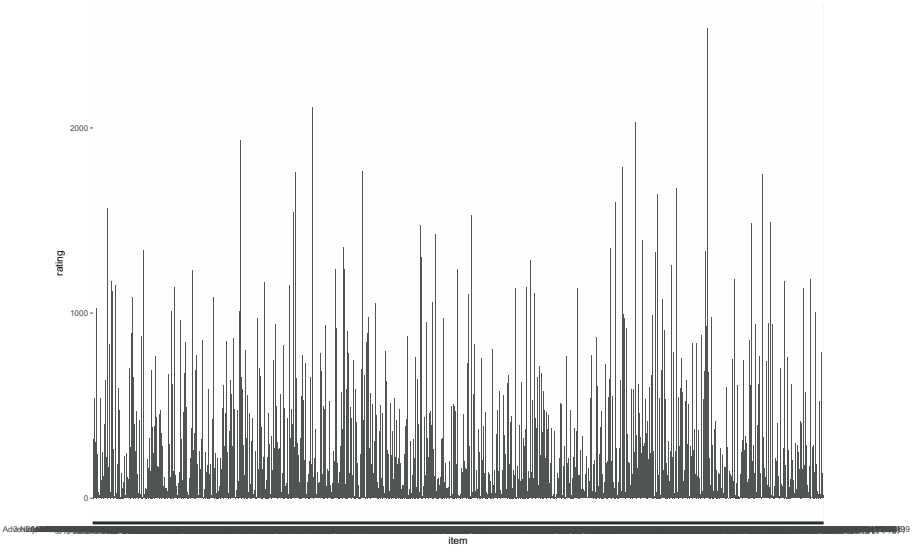
```
Formal class 'realRatingMatrix' [package recommenderlab"] with 2 slots
..@ data :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
.. .. ..@ i : int [1:99392] 0 1 4 5 9 12 14 15 16 17 ...
.. .. ..@ p : int [1:1665] 0 452 583 673 882 968 994 1386 1605 1904 ...
.. .. ..@ Dim : int [1:2] 943 1664
.. .. ..@ Dimnames:List of 2
.. .. .. ..: chr[1 : 943]"1""2""3""4"...
..... : chr [1:1664] "Toy Story (1995)GoldenEye (1995)Four Rooms (1995)Get Shorty
(1995)"...
.. .. ..@ x : num [1:99392] 5 4 4 4 4 3 1 5 4 5 ...
.. .. ..@ factors : list()
..@ normalize: NULL
```

Esta información nos dice que tenemos los datos correctamente preparados para ser utilizados por el sistema de recomendación de la librería recommenderlab.

3.2. *Exploratory data analysist*

Podemos apreciar las valoraciones realizadas sobre el total de las visualizaciones. Se aprecia lo que ya sabemos: hay algunas películas muy vistas y apreciadas por los usuarios, mientras que hay otras que no son prácticamente vistas.

Figura 10. Visualización de la distribución de valoraciones sobre todas las películas



Obviamente, las películas más vistas correspondes a los picos más altos. De una manera simple podemos listar estas películas.

	item	mean	n
1	Star Wars (1977)	4.358491	583
2	Contact (1997)	3.803536	509
3	Fargo (1996)	4.155512	508
4	Return of the Jedi (1983)	4.007890	507
5	Liar Liar (1997)	3.156701	485
6	English Patient, The (1996)	3.656965	481
7	Scream (1996)	3.441423	478
8	Toy Story (1995)	3.878319	452
9	Air Force One (1997)	3.631090	431
10	Independence Day (ID4) (1996)	3.438228	429

Este conjunto de operaciones son relativamente sencillas si sabemos cómo manejar el objeto *MovieLense*. Una recomendación que sugerimos es pasar este objeto a un data.frame para que posteriormente podamos aplicar las operaciones vistas en el módulo de *data wrangling*.

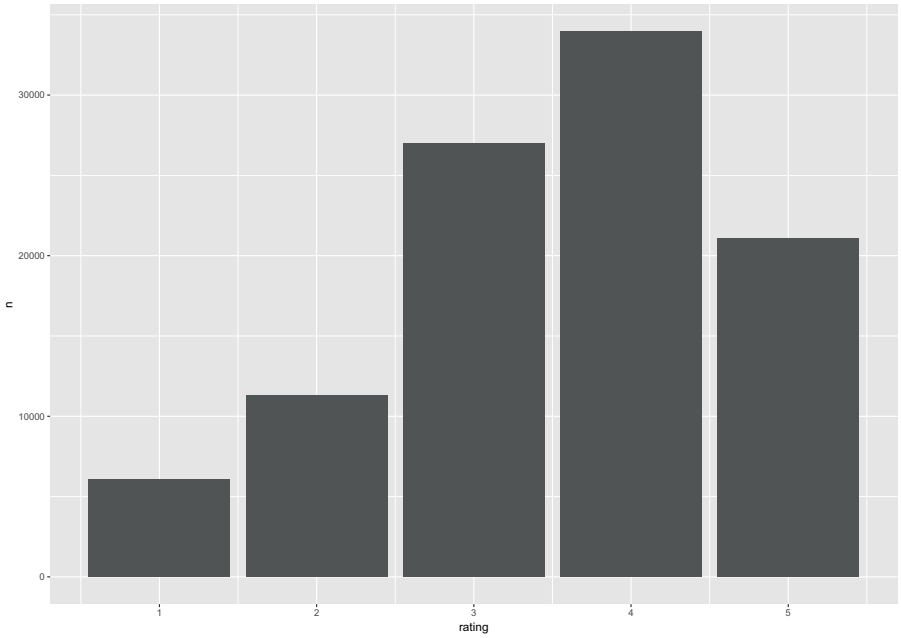
```
dff <- as(MovieLense,"data.frame") # Obtenemos un Data Frame
head(dff)
    user                                                item rating
1      1                                     Toy Story (1995)      5
453    1                                     GoldenEye (1995)      3
584    1                                    Four Rooms (1995)      4
674    1                                    Get Shorty (1995)      3
883    1                                       Copycat (1995)      3
969    1 Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)      5
```

Con la estructura ya conocida, las operaciones que generan el gráfico anterior y la lista de películas más vistas corresponden a dos líneas de código:

```
# Generar gráfico
ggplot(dff,aes(x=item,y=rating))+geom_bar(stat="identity")
# Generar tabla
dff %>% group_by(item) %>% % agrupamos por ítem (película)
# Calc. el promedio y el número de valoraciones
  summarise(mean = mean(rating), n = n()) %>%
   arrange(desc(n,mean)) %>% # ordenamos de forma descendiente
    top_n(10) # nos quedamos con los 10 primeros de la lista
```

Otra posible representación gráfica corresponde a las valoraciones más realizadas. En este caso debemos eliminar los *ratings* iguales a 0 porque indican *valor missing*. En la figura 11 se aprecia cómo el valor de *rating* más utilizado es el 4. A continuación el 3 y 5, y los últimos *ratings* corresponden al valor 2 y 1.

Figura 11. Visualización de la distribución de valoraciones (eliminando ceros)

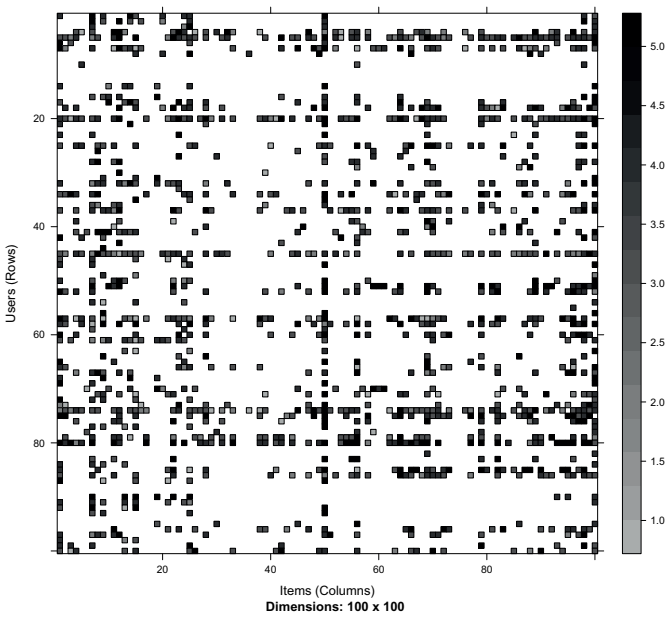


Los valores representados gráficamente provienen de la siguiente tabla:

	rating	n
1	1	6059
2	2	11307
3	3	27002
4	4	33947
5	5	21077

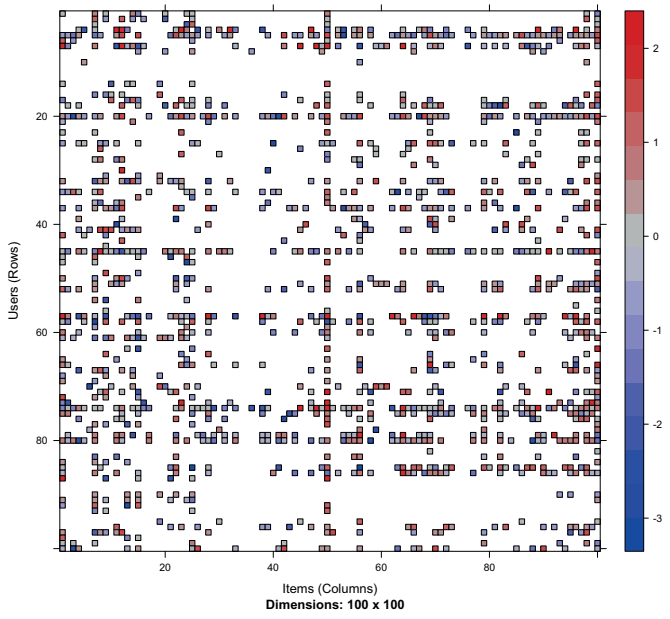
Si visualizamos el contenido de la matriz de forma aleatoria para un conjunto de 100 usuarios y sobre las primeras 100 películas, obtendremos una vista como la de la figura 12:

Figura 12. Matriz de *rating* de 100 usuarios tomados aleatoriamente y sus valoraciones sobre las 100 primeras películas



La misma figura se puede obtener si normalizamos la matriz de valoraciones utilizando la función para este propósito llamada *normalize*. La figura 12 muestra cómo algunos usuarios visualizan y valoran muchas películas, mientras que la mayoría solamente unas pocas. Y también observamos una serie de películas que son vistas por muchos usuarios. Da la sensación de una línea vertical que sobresale del conjunto restante de puntos.

Figura 13. Matriz de *rating* de 100 usuarios tomados aleatoriamente y sus valoraciones sobre las 100 primeras películas



3.3. Aplicación de los modelos

La utilización de la librería recommenderlab facilita la aplicación de los sistemas de recomendación descritos de filtrado colaborativo en sus dos variantes: de usuario y de ítem. Adicionalmente ofrece un conjunto más extenso de métodos que se pueden usar para contrastar con los métodos descritos en este módulo.

Una manera de conocer qué conjunto de opciones me ofrece la librería es a través del método *recommenderRegistry*. A continuación mostramos algunos de los métodos disponibles. En nuestro caso, vamos a utilizar las opciones identificadas como: *IBCF_realRatingMatrix* (*item-based collaborative filtering*) y *UBCF_realRatingMatrix* (*user-based collaborative filtering*).

```
recommenderRegistry$get_entries(dataType="realRatingMatrix")

$IBCF_realRatingMatrix
Recommender method: IBCF for realRatingMatrix
Description: Recommender based on item-based collaborative filtering.
Reference: NA
Parameters:
   k   method normalize normalize_sim_matrix alpha na_as_zero
1 30 "Cosine"  "center"                FALSE   0.5      FALSE

$POPULAR_realRatingMatrix
Recommender method: POPULAR for realRatingMatrix
Description: Recommender based on item popularity.
Reference: NA
Parameters:
  normalize  aggregationRatings
1  "center" new("standardGeneric", .Data = function (x, na.rm = FALSE, dims = 1,
                                                 aggregationPopularity
1 new("standardGeneric", .Data = function (x, na.rm = FALSE, dims = 1,


$SVDF_realRatingMatrix
Recommender method: SVDF for realRatingMatrix
Description: Recommender based on Funk SVD with gradient descend.
Reference: NA
Parameters:
   k gamma lambda min_epochs max_epochs min_improvement normalize verbose
1 10 0.015  0.001         50        200           1e-06  "center"   FALSE

$UBCF_realRatingMatrix
Recommender method: UBCF for realRatingMatrix
Description: Recommender based on user-based collaborative filtering.
```

```
Reference: NA
Parameters:
    method nn sample normalize
1 "cosine" 25  FALSE  "center"
```

3.3.1. Método = UBCF: user-based collaborative filtering

Para la aplicación de los métodos hemos seleccionado una muestra sobre el conjunto total de los usuarios. Esto es únicamente para acelerar el cálculo del modelo.

```
set.seed(12345) rSmpl <- sample(df,100)# Tomamos una muestra de 100 usuarios
rUBCFsamp <- Recommender(rSmpl,method=ÜBCF")
names(getModel(rUBCFsamp))
```

Con el resultado del modelo estimado, podemos predecir sobre un conjunto de nuevos usuarios (o no) y visualizar la lista de recomendaciones que el sistema genera para ellos. De nuevo, la mayor parte de la dificultad la resuelve la librería y nosotros únicamente hemos de llamar al método que permite realizar la predicción, *predict*:

```
nTopRecom <- 10 # Definimos el número de películas que queremos recomendar
rPredict <- predict(object = rUBCFsamp,
+ newdata = df[1:5,],
+ n = nTopRecom)
rPredict # nos indica que ha generado predict
Recommendations as 'topNList' with n = 10 for 5 users.
>str(rPredict)
Formal class 'topNList' [package recommenderlab"] with 4 slots
..@ items :List of 5
.. ..1 : int[1 : 10]301311275472287647323316354309
.... 2: int [1:10] 480 511 197 316 187 650 185 194 322 134
.. ..3 : int[1 : 10]502109879204480134216100511
.... 4: int [1:10] 127 268 472 480 100 137 316 134 511 285
.. ..5 : int[1 : 10]268301472311271107576843042 84
..@ratings : Listof 5
.... 1: num [1:10] 4.19 3.89 3.83 3.79 3.74 ...
.. ..2 : num[1 : 10]3,973,963,953,933,91...
.... 3: num [1:10] 3.1 3.04 3.02 3.01 3 ...
.. ..4 : num[1 : 10]4,74,664,624,614,55...
.... 5: num [1:10] 3.27 3.21 3.17 3.15 3.14 ...
..@ itemLabels: chr [1:1664] "Toy Story (1995)GoldenEye (1995)Four Rooms (1995)Get
Shorty (1995)"...
..@ n : int 10
```

Podemos visualizar qué recomendaciones genera para el usuario 1 y el usuario 2:

	User 1	User 2
1	L.A. Confidential (1997)	Casablanca (1942)
2	Titanic (1997)	Boot, Das (1981)
3	Leaving Las Vegas (1995)	Graduate, The (1967)
4	Trainspotting (1996)	Schindler's List (1993)
5	Scream (1996)	Godfather: Part II, The (1974)
6	Glory (1989)	Chinatown (1974)
7	Crash (1996)	Psycho (1960)
8	Schindler's List (1993)	Sting, The (1973)
9	One Flew Over the Cuckoo's Nest (1975)	Lost Highway (1997)
10	Wings of the Dove, The (1997)	Citizen Kane (1941)

En ocasiones quizá queramos la predicción de las valoraciones. En este caso podemos hacer uso del mismo método *predic* pero indicando que el tipo de predicción es de valoraciones.

```
# Predicciones via ratings
rPredictRat <- predict(object = rUBCFsamp,
                    newdata = df[1:5,],
                    type="ratings")
# Obtenemos las valoraciones como una matriz (sobre las 5 primeras películas)
as(rPredictRat,"matrix")[,1:5]
```

En la siguiente tabla queda más detallada la generación de *ratings* que ofrece el modelo de recomendación ajustado. Observad que las indicaciones de *NA* corresponden a valoraciones que este usuario tiene. Es decir, no es necesario predecir el valor porque ya está disponible en la matriz de valoraciones inicial. Los usuarios indicados no se corresponden al usuario 1, 2 y así sucesivamente del conjunto de datos inicial, sino del conjunto obtenido vía muestra (*sample*).

	Toy Story (1995)	GoldenEye (1995)	Four Rooms (1995)	Get Shorty (1995)	Copycat (1995)
U1	NA	NA	NA	NA	NA
U2	NA	3.6653	3.7049	3.7312	3.7456
U3	2.9667	2.8176	2.6390	2.8192	2.7875
U4	4.3581	4.2067	4.3276	4.2206	4.3043
U5	NA	NA	2.8743	2.8743	2.9081

Obviamente, si se quiere disponer de la matriz completa, es decir, de los valores estimados y de los ya disponibles, únicamente se utiliza la opción *ratingMatrix* en vez de *ratings*.

	Toy Story (1995)	GoldenEye (1995)	Four Rooms (1995)	Get Shorty (1995)	Copycat (1995)
U1	1.3948	-0.6052	0.3948	-0.6052	-0.6052
U2	0.2951	3.6653	3.7049	3.7312	3.7456
U3	2.9667	2.8176	2.6390	2.8192	2.7875
U4	4.3581	4.2067	4.3276	4.2206	4.3043
U5	1.1257	0.1257	2.8743	2.8743	2.9081

3.3.2. Método = IBCF: item-based collaborative filtering

La utilización de una muestra sobre el conjunto total de los usuarios no facilita para nada este método. Recordemos que en este caso el efecto de la reducción del número de usuarios no afecta al número de ítems (que evidentemente mantenemos).

```
set.seed(12345) rSmpl <- sample(df,100)# Tomamos una muestra de 100 usuarios
rIBCFsamp <- Recommender(rSmpl,method=ÏBCF")
names(getModel(rIBCFsamp))
```

La lista de recomendaciones obtenidas para el usuario 1 y 2 son las siguientes:

	User 1	User 2
1	Sense and Sensibility (1995)	Belle de jour (1967)
2	Time to Kill, A (1996)	Citizen Kane (1941)
3	Marvin's Room (1996)	Wrong Trousers, The (1993)
4	Ulee's Gold (1997)	Sting, The (1973)
5	Mrs. Brown (Her Majesty, Mrs. Brown) (1997)	Patton (1970)
6	Titanic (1997)	Ridicule (1996)
7	Schindler's List (1993)	One Flew Over the Cuckoo's Nest (1975)
8	Murder at 1600 (1997)	Close Shave, A (1995)
9	Lost Highway (1997)	Old Yeller (1957)
10	House of Yes, The (1997)	Fantasia (1940)

En los dos ejemplos realizados, correspondiente a cada variante del algoritmo de filtrado colaborativo, al realizar la previsión, los usuarios elegidos al azar tenían esta submatriz de datos para los primeros 5 ítems (películas):

	Toy Story (1995)	GoldenEye (1995)	Four Rooms (1995)	Get Shorty (1995)	Copycat (1995)
1	5	3	4	3	3
2	4	NA	NA	NA	NA
3	NA	NA	NA	NA	NA
4	NA	NA	NA	NA	NA
5	4	3	NA	NA	NA

4. Personalización: comparación y ranquin

4.1. Introducción

En la situación habitual deberemos decidir sobre un conjunto de modelos que reflejan diferente comportamiento al predecir recomendaciones para los usuarios. Deseamos disponer del sistema que sea más beneficioso para nuestros intereses. En este apartado pretendemos resaltar aquellos elementos que facilitan la valoración de los modelos ajustados.

En general, podemos entender que la lista de tareas que hemos de realizar para verificar, comparar y ordenar nuestros modelos es la siguiente:

- Preparar los datos de forma correcta. Una parte para el entrenamiento y otra para el test.
- Definir la lista de modelos que queremos aplicar y que desde nuestra perspectiva son correctos.
- Obtener medidas de rendimiento: la matriz de confusión o la visualización de la curva ROC.
- Comparar los modelos y seleccionar el mejor.

La ventaja de utilizar la librería recommenderlab es que nos facilita las tareas de verificación, comparación y selección.

Formalmente, nuestra matriz de evaluaciones se divide por usuarios, es decir:

$$\mathbf{R} = \mathcal{U}_{train} \cup \mathcal{U}_{test}$$

La filas que configuran el conjunto $\mathcal{U}_{train}$ se utilizan para ajustar (aprender) el modelo. Posteriormente, cada uno de los usuarios del conjunto $\mathcal{U}_{test}$ se utiliza para predecir los *ratings*. El modo como se realiza la división o partición del conjunto de datos depende del método asociado al esquema de verificación definido por *evaluationScheme*. Entre estas opciones encontramos:

- **Split**: se divide el conjunto de datos de forma aleatoria según la proporción indicada.

- **Bootstrap**: se realizan muestras de $\mathcal{U}_{test}$ con sustitución para crear un conjunto de *training* y utilizar el resto como conjunto de test.

- **Cross-validation**: se divide $\mathcal{U}$ en k subconjuntos (aproximadamente de la misma dimensión). A continuación se evalúa k veces con el mismo procedimiento. Se selecciona un subconjunto de cada iteración, que se utiliza como test, mientras que el resto de los subconjuntos son utilizados para entrenar el modelo.

4.1.1. Medidas de predicción

La forma estándar de evaluar el rendimiento de los modelos es calculando la desviación producida entre la predicción y los valores reales.

Si entendemos que el modelo nos genera una previsión en formato de *rating* $\hat{r}_{ij}$ y tenemos el valor real asociado r_{aj}, podemos calcular el error promedio de la media (MAE, *mean absolute error*).

$$MAE = \frac{1}{|\mathcal{K}|} \sum_{(i,j)\in\mathcal{K}} |r_{ij} - \hat{r}_{ij}|$$

El valor de $|\mathcal{K}|$ representa el número de valoraciones en nuestro conjunto.

Otra posible medida que vamos a poder obtener de manera directa es la raíz cuadrada de la media del error (RMSE, *root mean square error*).

$$RMSE = \sqrt{\frac{\sum_{(i,j)\in\mathcal{K}}(r_{ij} - \hat{r}_{ij}|}{|\mathcal{K})^2}}$$

La evaluación de las recomendaciones Top-N pueden ser agregadas y visualizadas en forma de una matriz de confusión.

4.2. Evaluación de las predicciones sobre *ratings*

Para tener alguna medida de rendimiento, debemos preparar la evaluación. Lo primero que debemos especificar es el porcentaje que corresponderá al conjunto de datos de *training*. En nuestro caso, se ha optado por un 80 %. Debemos especificar el número de ítems sobre los que vamos a realizar predicciones. Se ha elegido el número 15 porque garantiza que no hay ningún usuario con menos valoraciones que este valor. También se debe especificar el umbral a partir del cual se considera correcto. En nuestro caso fijamos el valor de 3. Se fijan también el número de veces que se ejecuta la evaluación y el método que hay que utilizar como esquema de evaluación (*split*, *bootstrap*, *cross*).

```
percTraining <- 0.8
itemsToKeep <- 15
```

```
ratingTH <- 3
nEval <- 5
evalSets <- evaluationScheme(data = rSmpl,
                             method = "split",
                             train = percTraining,
                             given = itemsToKeep,
                             goodRating = ratingTH,
                             k = nEval)
```

Una vez definido el esquema de evaluación, podemos generar recomendaciones para obtener medidas de error. Estas tres etapas son las que ejecutamos para los dos modelos probados.

```
rUB <- Recommender(getData(evalSets,"train"),"UBCF")
rIB <- Recommender(getData(evalSets,"train"),"IBCF")

pUB <- predict(rUB, getData(evalSets,"known"),type="ratings")
pIB <- predict(rIB, getData(evalSets,"known"),type="ratings")

UBCF <- calcPredictionAccuracy(pUB,getData(evalSets,"unknown"))
IBCF <- calcPredictionAccuracy(pIB,getData(evalSets,"unknown"))
```

Se obtiene la siguiente tabla:

	RMSE	MSE	MAE
UBCF	1.0527	1.1082	0.8368
IBCF	1.4177	2.0099	1.0743

A partir de estos datos, ¿qué modelo parece ser mejor? Parece que el modelo basado en usuario es ligeramente mejor que el basado en ítem. ¿Qué sucede si repetimos este procedimiento con los otros esquemas disponibles: *bootstrap* y *cross-validation*?

Para el caso de *bootstrap*:

	RMSE	MSE	MAE
UBCF	1.0576	1.1185	0.8462
IBCF	1.5583	2.4284	1.1898

Y para el caso de *cross-validation*:

	RMSE	MSE	MAE
UBCF	1.0062	1.0124	0.7973
IBCF	1.0718	1.1487	0.7870

Es decir, que en función del esquema de valoración las diferencias entre las dos aproximaciones son muy parecidas o favorecen al sistema de recomendación de UBCF.

4.2.1. Evaluación de las recomendaciones para Top-N

Usamos las funciones siguientes para hacer el cálculo de la matriz de confianza para el modelo UBCF:

```
## Evaluacion Top-N
modelToEval <- ÜBCF"
results <- evaluate(x = evalSets,
                    method = modelToEval,
                    n = seq(10, 100, 10))
(getConfusionMatrix(results)[[1]])
```

Que nos permite evaluar la matriz de confusión:

	TP	FP	FN	TN	precision	recall	TPR	FPR
10	2.30	7.70	59.00	1580.00	0.2300	0.0477	0.0477	0.0048
20	4.20	15.80	57.10	1571.90	0.2100	0.1159	0.1159	0.0098
30	5.50	24.50	55.80	1563.20	0.1833	0.1433	0.1433	0.0153
40	6.65	33.35	54.65	1554.35	0.1662	0.1734	0.1734	0.0208
50	7.95	42.05	53.35	1545.65	0.1590	0.1821	0.1821	0.0262
60	9.15	50.85	52.15	1536.85	0.1525	0.1943	0.1943	0.0317
70	10.40	59.60	50.90	1528.10	0.1486	0.2156	0.2156	0.0372
80	11.35	68.65	49.95	1519.05	0.1419	0.2289	0.2289	0.0429
90	12.90	77.10	48.40	1510.60	0.1433	0.2413	0.2413	0.0481
100	14.00	86.00	47.30	1501.70	0.1400	0.2522	0.2522	0.0537

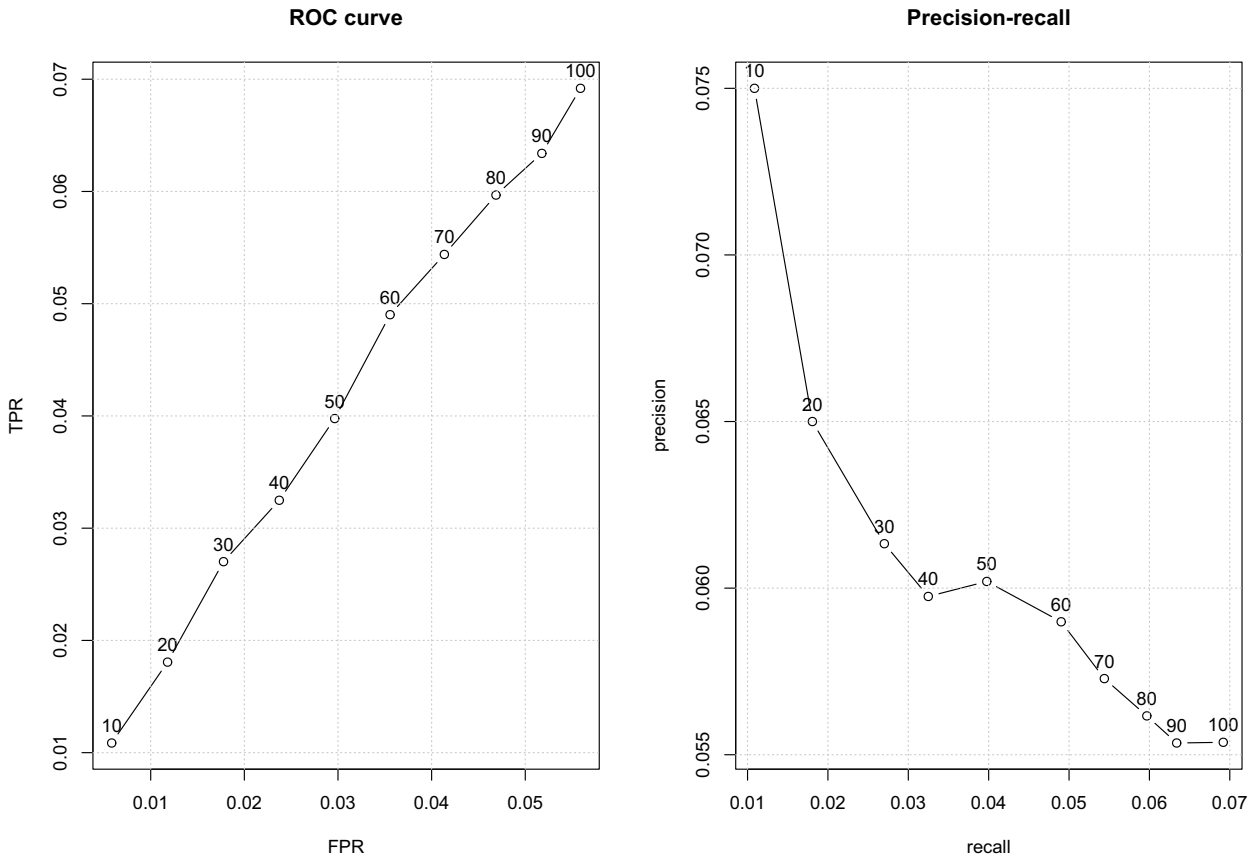
Para el modelo IBCF obtenemos la matriz de confusión después de proceder al cálculo:

```
IBCF run fold/sample [model time/prediction time]
1  [19sec/0sec]
2  [18.5sec/0sec]
3  [18.73sec/0.02sec]
4  [18.78sec/0.01sec]
5  [22.67sec/0.02sec]
```

	TP	FP	FN	TN	precision	recall	TPR	FPR
10	0.85	9.15	60.45	1578.55	0.0850	0.0150	0.0150	0.0058
20	1.60	18.40	59.70	1569.30	0.0800	0.0219	0.0219	0.0116
30	2.45	27.55	58.85	1560.15	0.0817	0.0344	0.0344	0.0173
40	2.90	37.10	58.40	1550.60	0.0725	0.0373	0.0373	0.0233
50	3.75	46.25	57.55	1541.45	0.0750	0.0482	0.0482	0.0291
60	4.15	55.60	57.15	1532.10	0.0694	0.0637	0.0637	0.0350
70	4.60	64.65	56.70	1523.05	0.0663	0.0773	0.0773	0.0407
80	4.95	73.30	56.35	1514.40	0.0631	0.0882	0.0882	0.0461
90	5.40	81.25	55.90	1506.45	0.0621	0.0902	0.0902	0.0511
100	5.85	87.60	55.45	1500.10	0.0623	0.0933	0.0933	0.0551

Es posible también visualizar la función ROC (*receiver operating characteristic*) y el gráfico de precisión y exhaustividad (una métrica empleada para evaluar el rendimiento de sistemas). Si el porcentaje de películas recomendadas es bajo, la precisión también decrece y ocurre que, si incrementamos el porcentaje de películas evaluadas, también se incrementa la variable *recall* (exhaustividad). Siempre tendremos presente este balance (*trade-off*) entre *precison* y *recall*.

Figura 14. Visualización de la curva ROC y las de *precision/recall*



4.3. Comparación de modelos

Este último subapartado lo incluimos para poder automatizar el proceso de selección del mejor modelo de recomendación que queremos verificar.

Definimos los modelos anteriores con las variantes de distancia (coseno y Pearson) y, adicionalmente, se utiliza un modelo base para poder comparar.

```
modelsToEval <- list(
  IBCFcos = list(name = "IBCF",
                 param = list(method = "cosine")),
  IBCFcor = list(name = "IBCF",
                 param = list(method = "pearson")),
  UBCFcos = list(name = "UBCF",
                 param = list(method = "cosine")),
  UBCFcor = list(name = "UBCF",
                 param = list(method = "pearson")),

  random = list(name = "RANDOM", param=NULL)
)

nRecomm<- c(1, 5, seq(10, 100, 10))
listResults <- evaluate(x = evalSets,
                        method = modelsToEval,
                        n = nRecomm)
```

Podemos comparar los tiempos empleados entre cada uno de los modelos propuestos:

```
IBCF run fold/sample [model time/prediction time]
 1  [19.29sec/0sec]
 2  [19.77sec/0sec]
 3  [18.95sec/0sec]
 4  [18.17sec/0.02sec]
 5  [18.23sec/0.01sec]

IBCF run fold/sample [model time/prediction time]
 1  [18.33sec/0sec]
 2  [18.7sec/0sec]
 3  [31.4sec/0.02sec]
 4  [32sec/0.03sec]
 5  [32.13sec/0.02sec]

UBCF run fold/sample [model time/prediction time]
 1  [0sec/0.11sec]
```

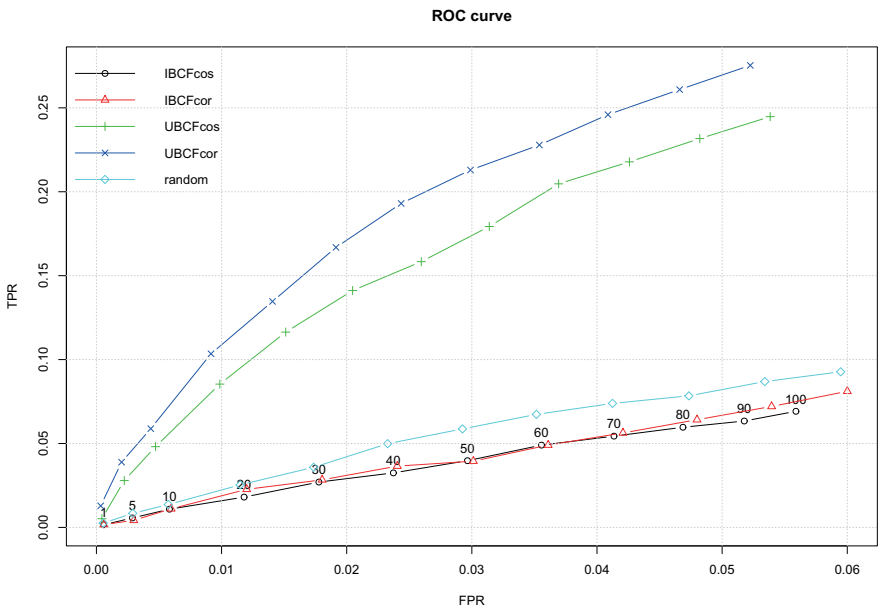
```
 2   [0sec/0.13sec]
 3   [0sec/0.09sec]
 4   [0.01sec/0.13sec]
 5   [0sec/0.16sec]

UBCF run fold/sample [model time/prediction time]
 1   [0.02sec/0.09sec]
 2   [0sec/0.09sec]
 3   [0sec/0.11sec]
 4   [0sec/0.1sec]
 5   [0sec/0.08sec]

RANDOM run fold/sample [model time/prediction time]
 1   [0sec/0.04sec]
 2   [0sec/0.02sec]
 3   [0.01sec/0.05sec]
 4   [0sec/0.04sec]
 5   [0sec/0.03sec]
```

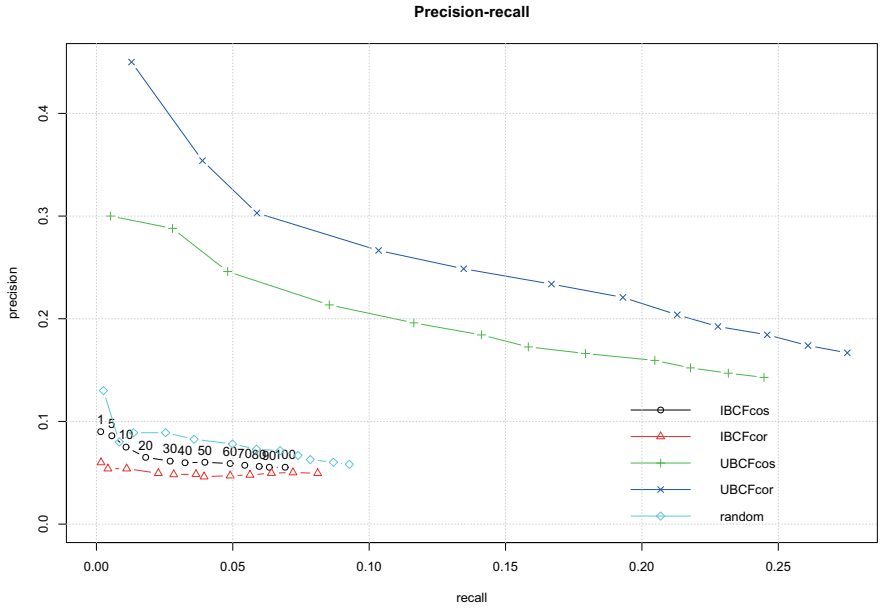
La visualización de los resultados muestra que el modelo basado en usuario ofrece mejor rendimiento. La opción relativa a la medida de distancia tiene un efecto menor, si bien la gráfica del modelo con la correlación de Pearson es mejor. Se aprecia que los modelos basados en ítem y el modelo base tienen rendimientos parecidos.

Figura 15. Visualización de la curva ROC para los 5 modelos



En este gráfico tenemos una representación de las curvas de precisión y exhaustividad, que llevan a la misma conclusión anterior. El mejor modelo es el que corresponde al UBCF con la distancia de Pearson.

Figura 16. Visualización de la curva de *precision/recall*



Resumen

Este módulo ha mostrado algunas de las estrategias más utilizadas para realizar recomendaciones a través de la aproximación del filtrado colaborativo. La dos variantes que se ofrecen, la de usuario y la de ítem, han sido descritas y puestas en aplicación práctica para mejorar la comprensión de este tipo de métodos.

La aproximación de usuario, que podríamos entender como el binomio usuario-usuario, se basa justamente en qué productos han adquirido en el pasado usuarios similares. Ya hemos visto la efectividad de esta solución pero a expensas de utilizar toda la matriz de valoraciones. En situaciones reales de *big data* esta solución puede degradarse sin un sistema que sea altamente paralelizable y de esta manera mitigar estos costes computacionales.

La aproximación de ítem, o del par ítem-ítem, se basa en localizar ítems parecidos. Esta visión por columna de la matriz de valoraciones ofrece escalabilidad cuando incorporamos nuevos usuarios, dado que el conjunto de los ítems en general se mantiene de una manera más estable a lo largo del tiempo.

Al ser un campo de interés dentro de la comunidad científica, hay otras aproximaciones a este tipo de problemas, como los sistemas de recomendación basados en contenidos, y que tienen su punto fuerte en conocer la estructura de preferencias del usuario. En el caso práctico de las películas, podríamos añadir información relativa a los actores, directores, tipo de película, etc., para obtener información relevante para este tipo de soluciones.

En cualquier caso, hay una base suficientemente sólida para poder apreciar la utilidad y relevancia de este tipo de modelos.

Como ya hemos visto en otros módulos, la aplicación de los sistemas de recomendación se ajusta a los procedimientos de evaluación y valoración presentados en los métodos de clasificación. Es decir:

- Preparación de los datos: matriz de valoraciones.
- Transformaciones que realizar: normalización, centrado o binarización son opciones que pueden, en ocasiones, mejorar el rendimiento del algoritmo de recomendación.
- Modelización a través de una serie de modelos candidatos.
- Evaluación: obtenemos medidas de rendimiento.
- Comparación y selección.

En la práctica, el modelo finalmente seleccionado para poner en ejecución deberá ser cuidadosamente analizado durante su fase de ejecución con el fin de garantizar que las recomendaciones generadas son correctamente valoradas por los usuarios del sistema. La medida de eficiencia del sistema de recomendación desde la perspectiva técnica es clara, pero desde la perspectiva de negocio también es relevante y puede determinar la solución final que implantar.

Estas son algunas de las medidas o KPI que se pueden tener en consideración:

- Valoración del incremento de ventas que provoca. Cualquier KPI de interés puede ser utilizado.

- Valoración del incremento de la actividad del cliente/usuario en nuestro negocio (virtual, visitando una URL, o físico, visitando una tienda). En esta situación hay otras medidas de interés que se pueden utilizar.

- Valoración del incremento de satisfacción del cliente/usuario.