Sistemas de recomendación I

Sistemas no personalizados

Le recomiendo a todos los usuarios lo mismo basado en un promedio de puntuaciones de los usuarios.

Dados los upvotes / downvotes, queremos saber con una confianza del x% cuál es el límite inferior para la probabilidad de que el comentario sea positivo.

Definiendo $n = U + D$ y $p = \frac{U}{U+D}$ entonces nuestro intervalo de confianza es:

$$\frac{p + \frac{z^2}{2n} \pm z\sqrt{\frac{p(1-p)+z^2}{4n^2}}}{\frac{1+z^2}{n}}$$ (para comentarios)

Donde z es el cuantil correspondiente a la distribución normal. Por ejemplo si queremos una confianza del 95% entonces $z = 1,96$.

La fórmula anterior no puede ser utilizada para noticias, ya que penalizaría las noticias nuevas. A su vez, con el correr del tiempo, nos interesaría darle menor peso a las noticias viejas ya que pierden relevancia.

Supongamos la función de utilidad "u", siendo "p" la prob. de que a un usuario le guste la noticia, y "q" la prob. de que no sea nueva:

$$u(p,q) = a \cdot pq + (1-p)q \cdot b + p(1-q) \cdot c + (1-p)(1-q) \cdot d$$

Noticia nueva y le gusta	Noticia nueva y no le gusta	Vieja y le gusta	Vieja y no le gusta

siendo $p = \frac{U+1}{U+D+2}$ y $q = e^{-\lambda t}$.

La fórmula real usada por Reddit es:

$$u(U,D,A,B) = \log_2(U-D) + \frac{A-B}{45000}$$

Donde A es el Timestamp de la noticia y B es el Timestamp en el que fue creado Reddit

Sistemas basados en contenido

Busca recomendar a un usuario ítems similares a los que le han gustado.

Por cada ítem, mantenemos un profile.

Por cada usuario mantenemos un profile.

Para determinar si a un usuario le puede interesar cierto ítem calculamos la dist. coseno entre ambos profiles:

$$\cos(\theta) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|}$$

sistemas de recomendación 2.

Pros:

+ Solo necesita la información del usuario
+ Puede detectar gustos particulares
+ Puede recomendar ítems nuevos o no populares.
+ Permite explicar el motivo de la recomendación.

Cons:

- Es difícil encontrar los features a utilizar.
- No puede realizar recomendaciones a usuarios nuevos
- No puede recomendar gustos más allá de los gustos del usuario.
- No tiene en cuenta la opinión de otros usuarios.

Collaborative Filtering.

Suponiendo que tenemos n usuarios y m ítems. Podemos construir una matriz de utilidad con las calificaciones:

	A	B	C	D	E	→ ítems
①	2		2	4	5	
②	5		4			
③			5		2	
④		1		5		
⑤			4			

↓ usuarios

→ calificaciones.

Buscamos predecir los valores faltantes.

User - User

Dado el usuario X necesitamos encontrar a los N usuarios con calificaciones más similares. Estimaremos luego la calificación de X en base a las de los demás N usuarios.

Supongamos que tenemos la siguiente opinión (binaria) de un conjunto de usuarios.

	A	B	C	D
①	1	1	1	1
②	0	1	0	1
③	1	0	1	0
④	0	0	1	1
⑤	1	?	?	?

¿Qué ítem le recomendaríamos al usuario ⑤?

Viendo que a ① y a ③ les gustó A y a ambos también les gustó C entonces le recomendaríamos C.

Para trabajar en calificaciones que no son binarias nos basamos en que la semejanza entre usuarios, calculamos la calificación como un promedio de las calificaciones de los demás usuarios, ponderado por la semejanza que tengan con el usuario en cuestión.

Para calcular la semejanza entre dos usuarios $\bar{x}$ e $\bar{y}$ (vectores con las calificaciones en común de ambos usuarios). Primero debemos restar el promedio de cada usuario a sus calificaciones:

Sistemas de Recomendacion 3.

	A	B	C	D	E	Prom.
①	2		2	4	5	→ 13/4
②	5		4			→ 9/2
③			5		2	→ 7/2
④	.	1			5	→ 3
⑤	4	5		1		→ 10/3

⇒

	A	B	C	D	E
①	$-\frac{5}{4}$		$-\frac{5}{4}$	$\frac{3}{4}$	$\frac{7}{4}$
②	$\frac{1}{2}$		$-\frac{1}{2}$		
③			$\frac{3}{2}$		$-\frac{3}{2}$
④		-2			2
⑤	$\frac{2}{3}$	$\frac{5}{3}$		$-\frac{7}{3}$	

Con esta nueva matriz calculamos la semejanza entre dos usuarios como el coseno.

$$sim(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|}$$

Para estimar la calificación de un usuario a un ítem, utilizaremos entonces la información de los N usuarios más similares:

$$\Gamma_{xi} = \frac{\sum_{y \in N} s_{xy} \Gamma_{yi}}{\sum_{y \in N} s_{xy}}$$

Donde:

Γ_{xi}: calificación del usuario x a ítem i (predicción).

s_{xy}: $sim(\vec{x}, \vec{y})$

Γ_{yi}: calificación del usuario y a ítem i.

→ Hablaría usar LSH.

N: conjunto de los N usuarios más similares.

Item-Item

Para calcular la semejanza basado en ítems la semejanza item-item el procedimiento es el mismo pero en vez de usar los filas como vectores (usuarios) usamos las columnas (items).

Item-Item en general da mejores resultados ya que los ítems son más simples, tienden a pertenecer a cierta categoría y eso no varía. Los usuarios tienen comportamiento más complejos, cambia de gustos, puede gustarle distintas cosas, etc.

Matrix Factorization. (SVD++)

El principio es muy simple: descomponer la matriz de utilidad en el producto de dos matrices Q y P. En donde Q tendrá tantas filas como ítems y tantas columnas como factores latentes usemos y P tendrá tantas filas como usuarios y tantas columnas como factores:

$$U_{m \times n} = Q_{m \times r} \; P^{T}_{n \times r}$$

El objetivo es encontrar Q y P tales que $Q \cdot P$ minimice el RMSE con la matriz de utilidad original.

De esta forma podemos estimar la calificación del usuario i para la película j haciendo $q_i \cdot p_j^T$

El problema es que como la matriz no está completa, no es posible calcular la SVD.

Tomar e buscar entonces los valores de p y q que minimicen el RMSE entre el valor estimado y el real:

$$\min_{P,Q} \sum_{(i,x)} (r_{xi} - q_i \cdot p_x^T)^2$$

Para esto utilizaremos gradient descent o alternating least squares.

Factorization Machines.

Supongamos que tenemos calificaciones en tripletas del estilo:

$$\{\text{user id}, \text{movie id}, \text{rating}\}.$$

Podríamos pensar en una simple regresión lineal para estimar la calificación de un usuario a una película de la forma:

$$\hat{y} = w_0 + w_1 \text{user id} + w_2 \text{movie id}$$

Convirtiendo el user id y movie id en "n" y "m" variables binarias mediante "one-hot encoding"

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{j=1+n}^{n+m} w_j \cdot x_j$$

Al "factorizar" agregamos todos los coeficientes de interacción entre cada una de las variables del modelo en total agregamos n^2 nuevas variables de tipo w_{ij} indicando la relación entre la variable i y la variable j del modelo

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} \cdot x_i \cdot x_j$$

Este modelo es muy completo pero tiene como problema que la cantidad de coeficientes sufre de un problema de explosión combinatoria.

La idea es representar a cada variable (columna) de nuestro modelo mediante un vector de "k" elementos siendo "k" la cantidad de factores latentes que queremos usar. Si hacemos esto entonces podemos reescribir nuestro modelo de regresión de la forma:

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle \cdot x_i \cdot x_j$$

La cantidad total de parámetros a estimar ahora es: w_0, "n" coeficientes de w_i y "n" vectores v_i de "k" dimensiones cada uno. Este número de parámetros es perfectamente manejable incluso en sistemas con millones de usuarios o ítems.