Sistemas de Recomendación

Reducción de la dimensionalidad

Evaluación de sistemas de recomendación

Agenda

- Algoritmos en Surprise
- Reducción de dimensionalidad
- Evaluación de sistemas de recomendación

Algoritmo básico

- BaselineOnly
  - Calcula: $\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$
    - mu es el promedio general
    - b_u es el sesgo del usuario (0 si usuario desconocido)
    - b_i es el sesgo del ítem (0 si ítem desconocido)
  - Estimar b_u y b_i se resuelve por mínimos cuadrados:

$$\min_{b_*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right).$$

Vecinos más cercanos

- KNNBasic
  - Calcula (según sea ítems o usuarios)

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)}$$

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i,j) \cdot r_{uj}}{\sum\limits_{j \in N_u^k(j)} \text{sim}(i,j)}$$

- KNNWithMeans: tiene en cuenta el promedio de cada usuario / item
- KNNBaseline: tiene en cuenta el *baseline*

Co-Clustering

- CoClustering
  - Los usuarios y los ítems son asignados a algunos *clusters* C_u, C*i*, y algunos a los *co-clusters* C_{ui}
  - Calcula:

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i})$$

Agenda

- Algoritmos en Surprise
- Reducción de dimensionalidad
- Evaluación de sistemas de recomendación

Reducción de la dimensionalidad

- La matriz de utilidad es una representación sobreajustada de los gustos del usuario y de las descripciones de los ítems
  - Sinonimia: Me gustan “El Silmarillion” y “El Hobbit” y a vos “Los trabajos completos de JRR Tolkien”
  - Complejidad computacional, potencialmente peores resultados
- Ideal: tener una representación más compacta de los gustos del usuario y las descripciones de los ítems

En recuperación de la información…

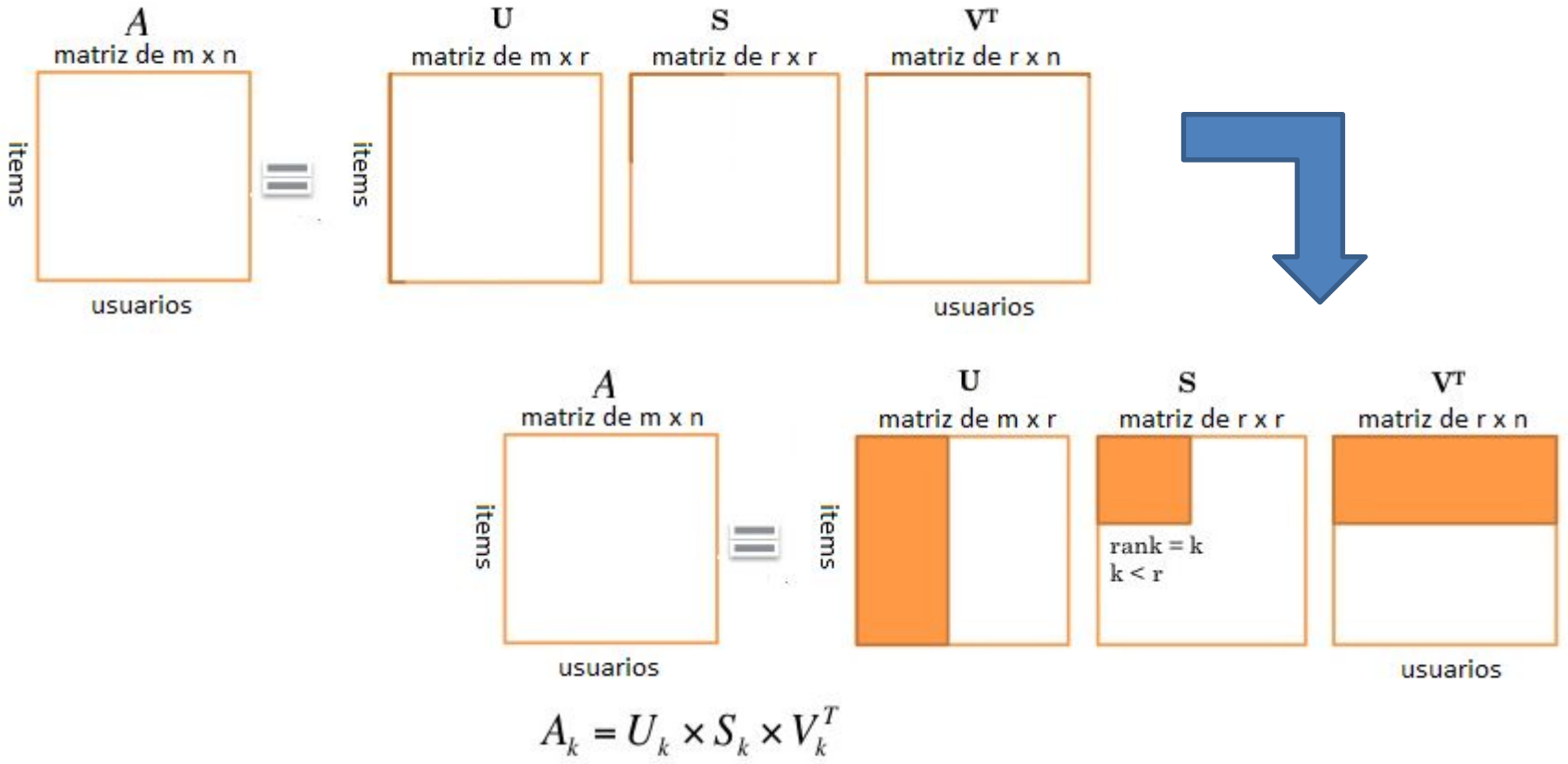
- Las consultas y los documentos están representados en forma pobre por los vectores de palabras
- Se quería reconocer **conceptos**, no palabras
- Si se buscaba “compilador” nunca se iba a encontrar el documento “traductor de pascal en código de máquina”

SVD

- Descomposición en valores singulares (SVD)
- Reduce el espacio de utilidades a uno más robusto y compacto
- Factoriza la matriz de utilidad en tres matrices
- Muy relacionado con componentes principales (PCA)

SVD: Cómo funciona

$$A = U \times S \times V^T$$



$$A_k = U_k \times S_k \times V_k^T$$

SVD: Matrices resultantes

- La matriz R reconstruida es la matriz de rango k más aproximada a la R original
  - La matriz S tiene las dimensiones de gustos
  - La matriz U tiene la opinión (negativa o positiva) de cada usuario para cada gusto
  - La matriz V tiene cuánto exhibe cada ítem los gustos
- La predicción es directa: Los gustos de un usuario (una columna de U) se multiplica por los gustos de ítems (una columna de V)

SVD: Implementación en Surprise

- Se quiere predecir

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

- Minimizando el error cuadratico regularizado

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 \right)$$

- La minimización se hace...

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned}$$

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

SVD: Beneficios

- El resultado es un modelo rápido y compacto
- Tiene una red de vecinos más rica que KNN
- Se necesita experimentar para encontrar el mejor valor de k para un dominio

SVD: Problemas

- Valores perdidos
  - Imputarlos: usualmente con promedios o 0
- Complejidad computacional
  - $O(m^2n + n^3)$
  - Agregar usuarios, items: *folding*
- No se pueden explicar los vectores
  - Las dimensiones óptimas no son conceptos que el usuario pueda entender

NMF

- Factorización de matrices no negativas (NMF)
- Muy similar a SVD
- Los factores se mantienen positivos
- Se quiere predecir

$$\hat{r}_{ui} = q_i^T p_u$$

- En cada paso se actualizan los factores *f* para el usuario *u* e ítem *i*

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r_{ui}} + \lambda_u |I_u| p_{uf}}$$

$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r_{ui}} + \lambda_i |U_i| q_{if}}$$

Agenda

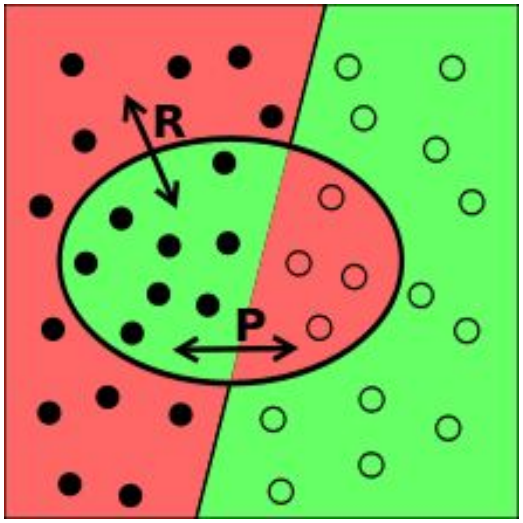
- Algoritmos en Surprise
- Reducción de dimensionalidad
- Evaluación de sistemas de recomendación

Evaluación

- En predicciones, predecir “4 versus 2.5” es peor que “2.5 versus 1”
- En recomendaciones, el comienzo de la lista es lo que más importa
- “Error”
  - Medida de lo malo que es una predicción
  - Cuando una peli buena para el usuario tiene una predicción/posición en la lista mala (y al revés)

Precisión y exhaustividad (1)

- Métricas de recuperación de la información
  - Precisión: Porcentaje de ítems seleccionados que son “relevantes”
  - Exhaustividad: Porcentaje de los ítems “relevantes” que son seleccionados



Precisión y exhaustividad (2)

- Precisión trata de encontrar las cosas más relevantes
  - Asume que hay más cosas relevantes de las que el usuario quiere
- Exhaustividad trata de no perder cosas relevantes
  - Asume que se tiene el tiempo para filtrar los resultados para encontrar lo que necesito
- Balance de los dos: Medida F1 = 2PR / (P+R)

Precisión y exhaustividad (3)

- Problema 1
  - Para todos los ítems tengo que saber si los ítems son buenos o no, para todos los usuarios
  - Pero si lo sé, para qué quiero un recomendador
  - Solución: esconder algunos ratings conocidos y tratar de predecirlos

Precisión y exhaustividad (4)

- Problema 2
  - Cubre el conjunto de datos completo
  - No funciona para los recomendadores top-N
  - Solución: P@n y R@n
    - Precision@n es el porcentaje de los top-n ítems que son “relevantes”
    - Recall@n: de hecho es lo mismo

Curva ROC

- Refleja la performance de un clasificador a diferentes niveles
- En sistemas de recomendación, la curva refleja donde “corto” las predicciones para recomendar
  - En las películas, si corto en 4 estrellas, solamente recomiendo las mejores, voy a estar cerca del (0,0)
  - Si corto en 2 estrellas, recomiendo casi todas, voy a estar cerca del (1,1)

Métricas para *rankings*

- Las medidas anteriores no toman la calidad del *ranking* predicho
- Si tengo un solo ítem “relevante” predicho, es mejor acertar a la primera posición del ranking que a la tercera
- Idea: medida en que el ranking predicho respeta el ranking del usuario

Mean Reciprocal Rank (MRR)

- MRR: promedio del inverso de las posiciones recíprocas correctas

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}.$$

- Ejemplo:

Ítems	Recomendación	Posición	Posición recíproca
A	B, C, **A**	3	1/3
B	**B**, C, A	1	1
C	B, **C**, A	2	1/2
			Promedio: 0.61

Correlación de Spearman para *rankings*

- Es simplemente la correlación de Pearson para *rakings* de ítems

$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}.$$

- donde d_i es la diferencia entre los dos ratings del ítem *i*
- Predicho: {A, B, C, D} y la verdad es {A, C, B, D} entonces 1 – [6*(0+1+1+0)/(4*15)] = 0.8

Discounted Cumulative Gain (1)

- Penaliza a los ítems relevantes que están más abajo en el *ranking*
- Para un ranking de *p* posiciones se define como…

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$$

- Donde rel_i es la relevancia del ítem *i*

Discounted Cumulative Gain (2)

- El resultado se normaliza porque no todos los *rankings* tienen la misma longitud

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

- Teniendo

$$\mathrm{IDCG_p} = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

- Donde REL es el conjunto de ítems relevantes

Discounted Cumulative Gain (3)

- Ejemplo:

i	rel_i	$\log_2(i+1)$	$\frac{rel_i}{\log_2(i+1)}$
1	3	1	3
2	2	1.585	1.262
3	3	2	1.5
4	0	2.322	0
5	1	2.585	0.387
6	2	2.807	0.712

- $DCG_6 = 6.861$

- IDCG sería lo mismo pero ordenado:
  - 3, 3, 2, 2, 1, 0
- $IDCG_6$: 7.141
- Entonces $nDCG_6$ = 0.961

¿Preguntas?