Diseño y Análisis de Algoritmos ICI-522

Sergio Hernández.
PhD computer science

Departamento de Computación e Informática
Universidad Católica del Maule.
shernandez@ucm.cl

Sistemas Recomendadores [1]

- Los sistemas recomendadores son herramientas que sugieren items a un usuario.
- Los sitios como YouTube utilizan estos software para encontrar items (videos, aplicaciones, libros, etc..) basado en los intereses del usuario.

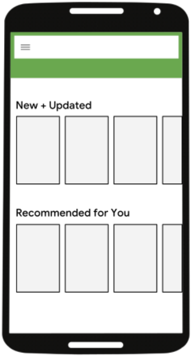

Figure: Google Playstore

[1] https://developers.google.com/machine-learning/recommendation/overview

Filtrado Colaborativo

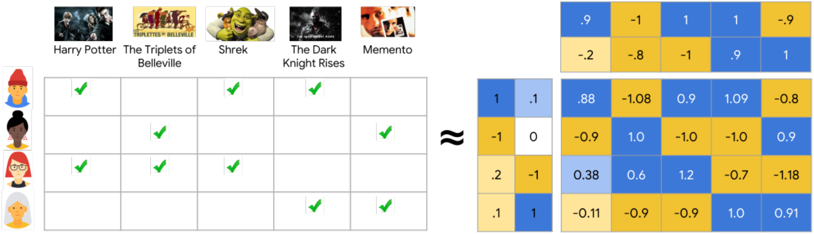

Figure: Filtrado colaborativo para una matriz de intereses de usuario $A \in \mathbb{R}^{m \times n}$ usando factorización de matrices $U_k \in \mathbb{R}^{m \times k}$ y $V_k \in \mathbb{R}^{n \times k}$, tal que $A \approx U_k \times V_k^T$

Filtrado Colaborativo

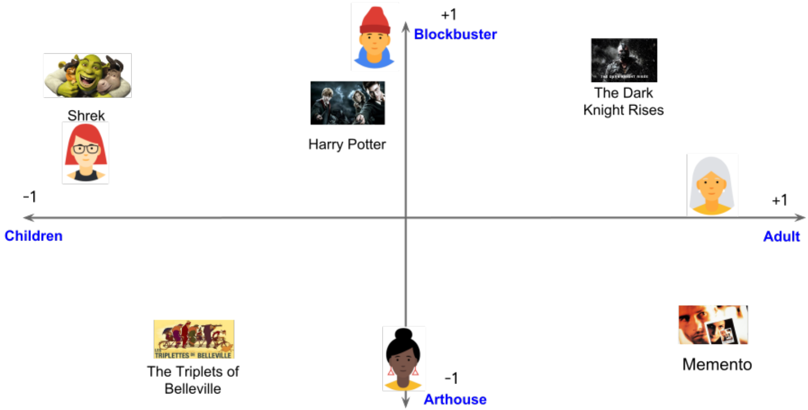

Figure: La idea es encontrar un conjunto de k factores latentes que caractericen a la relación entre los usuarios y los items (embeddings)

Descomposición por Valores Singulares

Sea $A \in \mathbb{R}^{m \times n}$ una matriz de intereses de usuario, $U \in \mathbb{R}^{m \times m}$,$S \in \mathbb{R}^{m \times n}$ y $V \in \mathbb{R}^{n \times n}$

SVD

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

Descomposición por Valores Singulares

Sea $A \in \mathbb{R}^{m\times n}$ una matriz de intereses de usuario, $U \in \mathbb{R}^{m\times m}$,$S \in \mathbb{R}^{m\times n}$ y $V \in \mathbb{R}^{n\times n}$

SVD

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

Propiedades

Sean $\sigma_1, \ldots, \sigma_r$ una secuencia de valores no-negativos en orden decreciente y $r = \mathrm{MIN}(m, n)$

$$U^T \times U = \mathbb{I}_m, \quad S = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \cdots & & & \\ 0 & \cdots & \sigma_r & 0 \end{pmatrix}$$

$$V^T \times V = \mathbb{I}_n$$

SVD Truncada

Ahora consideramos una aproximación de bajo rango utilizando $k < r$ factores lantentes:

SVD truncada

$$A \approx U_k \times S_k \times V_k^T$$

SVD Truncada

Ahora consideramos una aproximación de bajo rango utilizando $k < r$ factores lantentes:

SVD truncada

$$A \approx U_k \times S_k \times V_k^T$$

Propiedades

$$\tilde{A} = \left[\begin{array}{c|c|c} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_k \end{array}\right] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^T \\ \hline \vdots \\ \hline \boldsymbol{v}_k^T \end{bmatrix}$$

Figure: Aproximación de rango bajo

Análisis de Complejidad

- Para matrices A de tamaño $m \times n$ el costo computacional de obtener la descomposición es $\mathcal{O}(mnr)$
- Para el problema de valores propios, podemos calcular

$$T = \left[\begin{array}{c|c} T_1 & \beta \\ \hline \beta & T_2 \end{array}\right] \tag{1}$$

- La relación de recurrencia está dadá por $T(n) = 2T\left(\frac{r}{2}\right) + r^2$
- Por lo tanto, el costo del algoritmo divide y vencerás es $\mathcal{O}(r^2)$

Cuppen, J.J.M. A divide and conquer method for the symmetric tridiagonal eigenproblem. Numer. Math. 36, 177?195 (1980). https://doi.org/10.1007/BF01396757

Valores Propios de Matrices

La relación entre los valores singulares y los valores propios está dada por la matriz de Gram.

```
>>> import numpy as np
>>> from scipy import linalg
>>> A=np.array([[1,4],[5,2]])
>>> U,S,V = linalg.svd(A,full_matrices=False,
    lapack_driver='gesdd')
>>> D=np.diag(S)
>>> np.matmul(D,D.T)
array([[37.31782106,  0.        ],
[ 0.        ,  8.68217894]])
>>> np.linalg.eig(np.matmul(A,A.T))[0]
array([ 8.68217894, 37.31782106])
```

Kevin P. Murphy. 2012. Machine Learning: A Probabilistic Perspective. The MIT Press (pp. 392).

SVD para big data

```
>>> import time
>>> import numpy as np
>>> from scipy import linalg

>>> A = np.random.random((200000, 100))

>>> t1=time.time()
>>> U,S,V = linalg.svd(A,full_matrices=False,lapack_driver='
    gesdd')
>>> print(time.time()-t1)
3.533274412155 1514
```

SVD para big data

```
>>> import dask.array as da
>>> import dask

>>> X = da.random.random((200000, 100), chunks=(50000, 100))
    .persist()

>>> t2=time.time()
>>> U, S, V = da.linalg.svd(X)
>>> print(time.time()-t2)
0.0014061927795410156
```

SVD para big data

```
>>> dask.visualize(S)
```

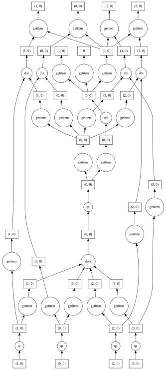

Figure: Grafo de cómputo usando Dask