Reducción de Dimensiones

75.06 Organización de Datos

April 14, 2014

1 Introducción

Este apunte aborda la reducción de dimensiones en grandes matrices. El manejo de matrices de enormes dimensiones se ha vuelto un problema recurrente en el procesamiento de datos masivos, existen muchos problemas de almacenamiento y procesamiento de datos en los cuales los mismos se presentan en forma matricial. A modo de ejemplo veamos algunos casos:

- La matriz de términos x documentos en un sistema de indexación de textos
- La matriz de utilidad de un sistema de recomendaciones (usuarios x items)
- La matriz de datos de entrenamiento en un sistema de clasificación automática

La reducción de dimensiones genera dos cosas: por un lado se logra una matriz reducida que es mas fácil de manejar y procesar logrando algoritmos mas eficientes. Por otro lado la matriz reducida es una representación de los datos originales en un espacio diferente lo cual permite extraer información que se desconoce de los datos, a esto lo llamaremos "modelo latente".

En este apunte vamos a describir dos técnicas diferentes pero con muchos puntos en común: la descomposición singular por valores (SVD) y la descomposición UV.

2 SVD

La descomposició singular por valores (SVD) es un miétodo algebraico para descomponer una matriz en el producto de tres matrices: U,S y T:

$$M_{nm} = U_{ns} S_{ss} T^t_{ms}$$

Donde S es la matriz diagonal formada por los autovalores de la matriz original ordenados en forma descendente de mayor a menor. Esta descomposición puede usarse facilmente para reducir las dimensiones de una matriz nulificando una cierta cantidad de autovalores en S. Al hacer eso automaticamente perdemos columnas de U y columnas de T, es decir filas de T transpuesta.

Los autovalores a eliminar serán siempre los mas chicos es decir que si queremos reducir la matriz a "k" dimensiones lo que haremos es solamente quedarnos con los "k" autovalores mas grandes de la matriz S.

$$M_{nm} \cong U_{nk} S_{kk} T^t_{mk}$$

Se demuestra (por suerte no aquí que la matriz en la que eliminamos los autovalores mas chicos es la matriz que mejor aproxima a la original para esa cantidad de autovalores no-nulos. Esto se hace calculando la norma de Frobenius.

Notar que hasta el momento aun quedandónos solo con algunos autovalores no hemos logrado reducir las dimensiones pues U*S*T sigue siendo de mxn. Pero al eliminar ciertos autovalores estamos nulificando columnas de U y columnas de T por lo que podemos tomar T transpuesta y usarla como estimación de nuestra matriz.

Notar que T es de mxk al transponerla nos queda de kxm siendo k la nueva cantidad de dimensiones. Es decir que pasamos de M de mxn a T transpuesta de kxn.La cantidad de "documentos" m no se modifica.

Esta matriz T transpuesta la podemos usar por ejemplo para resolver consultas.

2.1 Determinando el número de Dimensiones

Para saber a cuantas dimensiones se puede reducir una matriz suele tomarse como medida estandar el la suma de cuadrados de los autovalores y buscar eliminar autovalores de forma tal que la suma de cuadrados de los que nos queden sea de aproximadamente un 80% de la suma original. A esta cifra se la suele llamar "energía" de la matriz.

2.2 Modelo Latente

La matriz reducida en dimensiones originada de la SVD tiene igual cantidad de columnas que la matriz original pero su cantidad de filas es muy inferior. Es decir que cada fila de la matriz reducida expresa ahora una combinación lineal de varias filas de la matriz original. A esto lo denominamos "modelo latente".

El modelo latente permite descubrir automáticamente las relaciones que existían entre las filas de la matriz original. Por ejemplo en una matriz de términos x documentos el modelo latente agrupa términos en "conceptos". Esto permite, por ejemplo, que al buscar un cierto término en la colección algunos documentos que no contienen el término igual sean relevantes para la consulta. Por ejemplo aquellos que contienen un sinónimo del término buscado. Si estamos buscando "auto" aquellos documentos que tienen "automóvil" deberían ser relevantes. Esto es díficil de lograr en sistemas de indexación pero ocurre automáticamente al reducir las dimensiones de la matriz. A esto se lo llama "Indexación semántica latente" (LSI).

En un sistema de recomendaciones en donde tenemos en cada fila a un item y en cada columna a un usuario la reducción de dimensiones combina linealmente los items transformándolos en "conceptos". Por ejemplo si estamos recomendando películas tendremos que en la matriz reducida en lugar de películas individuales tenemos conceptos como por ejemplo "películas de ciencia ficción", "musicales", "películas con Kevin Bacon", etc. No tenemos dudas de que Kevin Bacon merece una fila completa en la matriz de películas una vez reducida.

Veamos con matrices muy chicas y ejemplos construidos a próposito si podemos ver como la reducción de dimensiones agrupa las filas.

2.2.1 Ejemplo de LSI

Vamos a comenzar con un ejemplo de indexación semántica latente en donde tendremos una matriz de 5 términos (filas) por 8 columnas (documentos) vamos a suponer que de nuestros 12 documentos hay 4 que hablan de química y otros 4 que hablan de planetas.

$$M_{5,8} = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 & d7 & d8 \\ jupiter & 5 & 4 & 3 & 5 & 0 & 0 & 0 & 0 \\ venus & 3 & 3 & 8 & 4 & 0 & 0 & 0 & 0 \\ marte & 4 & 5 & 4 & 3 & 0 & 0 & 0 & 0 \\ hierro & 0 & 1 & 0 & 0 & 4 & 3 & 2 & 4 \\ magnesio & 0 & 0 & 0 & 0 & 3 & 3 & 2 & 3 \end{pmatrix}$$

Realizamos la SVD de la matriz:

$$M_{5x8} = U_{5x5} S_{5x5} T^t_{8x5}$$

$$U_{5x5} = \begin{pmatrix} -0.55 & 0.01 & 0.58 & 0.57 & -0.12 \\ -0.63 & 0.03 & -0.75 & 0.14 & -0.07 \\ -0.53 & 0.00 & 0.28 & -0.75 & 0.26 \\ -0.04 & -0.77 & 0.02 & -0.18 & -0.60 \\ 0.00 & -0.63 & -0.05 & 0.23 & 0.73 \end{pmatrix}$$

$$S_{5x5} = \begin{pmatrix} 14.84 & 0 & 0 & 0 & 0 \\ 0 & 8.72 & 0 & 0 & 0 \\ 0 & 0 & 4.01 & 0 & 0 \\ 0 & 0 & 0 & 1.73 & 0 \\ 0 & 0 & 0 & 0 & 0.61 \end{pmatrix}$$

$$D_{8,5} = \begin{pmatrix} -0.45 & 0.02 & 0.44 & 0.150.28 & \\ -0.46 & -0.06 & 0.37 & -0.71 & -0.07 \\ -0.59 & 0.04 & -0.78 & -0.09 & 0.05 \\ -0.46 & 0.02 & 0.19 & 0.66 & -0.27 \\ -0.01 & -0.57 & -0.02 & 0 & -0.37 \\ 0 & -0.48 & -0.02 & 0.09 & 0.62 \\ 0 & -0.32 & -0.01 & 0.06 & 0.41 \\ -0.01 & -0.57 & -0.02 & 0 & -0.37 \end{pmatrix}$$

Si nos quedamos solo con los dos autovalores mas grandes entonces nos queda U5x2 S2x2 y T8x2. Tomando T transpuesta nos queda la siguiente matriz:

$$M_{2,8} = \begin{pmatrix} -0.45 & -0.46 & -0.59 & -0.46 & -0.01 & 0 & 0 & 0 \\ 0.02 & -0.06 & 0.04 & 0.02 & -0.57 & -0.48 & -0.32 & -0.57 \end{pmatrix}$$

Notemos que automáticamente al reducir las dimensiones de la matriz a dos nos han quedado divididos los documentos 1-4 de los documentos 5-8 ya que los primeros trataban sobre astronomía y los segundos sobre química.

Es mas claro ahora el significado de "modelo latente" se dice que es "latente" porque la información esta contenida en la matriz original pero oculta hasta que se reducen las dimensiones de la misma. Al hacer la reducción se realiza el agrupamiento de filas similares y surge una nueva representación para los datos.

Supongamos ahora que queremos hacer la consulta *jupiter, hierro* esto nos daria el vector Q=1,0,0,1,0 pero ahora no tenemos mas 5 dimensiones sino 2. Debemos entonces plegar el vector Q a 2 dimensiones. Esto lo hacemos de la forma:

$$Q_{1x2} = Q_{1x5} * (U_{5x2} S^{-1}_{2x2})$$

En nuestro caso Q quedaria = -0.04,-0.08

Ahora podriamos por ejemplo comparar este vector Q de 2 dimensiones contra cada vector de 2 dimensiones de la matriz D reducida aplicando por ejemplo el metodo del coseno y de esta forma calcular el documento mas relevante para nuestra consulta.

3 Descomposición UV

La descomposición UV es un método genérico e iterativo para descomponer una matriz en menos dimensiones. Este método sirve para aproximar matrices incompletas es decir aquellas en las cuales no tenemos todos los elemenos Aij de la matriz. Para esas matrices la SVD no funciona ya que debe partir de la matriz completa. Es por este motivo que esta descomposición se usa principalmente en sistemas de recomendaciones en donde se desconocen algunos valores de la matriz de utilidad y queremos estimarlos.

El algoritmo para generar la matriz que vamos a describir es iterativo, se basa en ir descomponer la matriz original de mxn en dos matrices: U y V siendo U de mxk y V de kxn. La matriz V de kxn representa el modelo latente y el producto de UxV sirve para estimar los datos que faltan en la matriz original.

$$M_{m,n} = U_{m,k} * V_{k,n}$$

En general se trata de usar un "k" grande para el modelo latente a fin de capturar todas las relaciones posibles entre las filas de la matriz original.

A continuación describimos un algoritmo básico para la descomposición UV

3.1 Algoritmo

```
repeat
 r,s =  pick random element of U
 U[r,s] = optimize(U,r,s)
 r,s =  pick random element of V
 V[r,s] = optimize(U,r,s)
until convergence or timeout
```

En donde la optimizació de los elementos de U está definida por:

$$U_{rs} = \frac{\sum_j V_{sj}(M_{rj} - \sum_{k \neq s} U_{rk} V_{kj})}{\sum_j V_{sj}^2}$$

Y la optimizació de los elementos de V está definida por:

$$V_{rs} = \frac{\sum_i U_{ir}(M_{is} - \sum_{k \neq r} U_{ik} V_{ks})}{\sum_i U_{ir}^2}$$

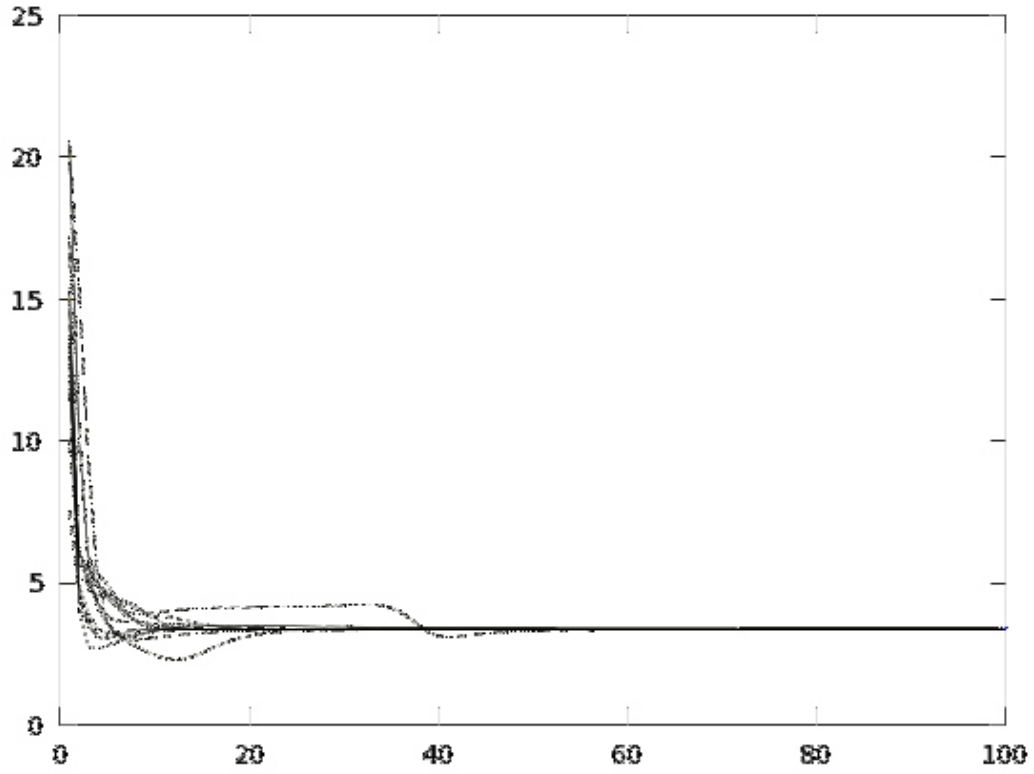
Con la salvedad de que al hacer la sumatoria del numerador hay que ignorar los elementos de la matriz original que no estan definidos. (no sumar nada).

Se debe aclarar el motivo por el cual se toman al azar los elementos a optimizar de cada matriz en cada paso. En primer lugar y principalmente es para poder tener una cota sobre el tiempo que va a tardar el algoritmo. En una matriz chica podemos optimizar todos sus elementos n veces sin problemas. En una matriz de miles de millones de elementos tenemos que acotar cuanto tiempo invertimos en buscar una solución y eso lo determinamos controlando la convergencia del algoritmo.

Dentro del algoritmo se puede calcular el error del mismo simplemente calculando U*V y luego haciendo cuadrados mínimos con la matriz original, es decir que a cada elemento de M le restamos el estimado por U*V y luego esa diferencia la elevamos al cuadrado. De esta forma se puede controlar la convergencia del algoritmo y/o el tiempo que hemos invertido.

Es de destacar que por la gran cantidad de variables que presenta esta descomposición es posible que se produzca un sobre-ajuste, es decir que se encuentre un par de matrices U,V que ajusten bien a los datos que tenemos en la matriz original pero que haga predicciones pobres. Para evitar esto se puede correr el algoritmo varias veces partiendo de diferentes matrices iniciales y luego hacer un promedio entre todas las matrices candidatas obtenidas.

El siguiente grafico muestra la forma en que converge el algoritmo, hay varias curvas porque se corrio varias veces el algoritmo.



3.1.1 Ejemplo de Recomendaciones

Veamos un ejemplo en donde usamos la descomposición UV para un sistema de recomendaciones.

Partimos de la siguiente matriz:

$$M = \begin{pmatrix} pelicula & U1 & U2 & U3 & U4 \\ TheHobbitt & 1 & 5 & 5 & ? \\ StarWars & 4 & ? & ? & 4 \\ Terminator & 4 & 1 & 1 & 5 \\ HarryPotter & ? & ? & 4 & 1 \end{pmatrix}$$

Observemos en la matriz que a los usuarios 1 y 4 parecen gustarle las películas de ciencia ficción mientras que a 2 y 3 les gustan las de fantasía. Vamos a realizar la descomposición de U y V usando k=2

$$U = \begin{pmatrix} 5.10 & 0.41 \\ 0.21 & 2.56 \\ 0.11 & 3.48 \\ 5.90 & 0.13 \end{pmatrix}$$

$$V = \begin{pmatrix} 0.02 & 0.72 & 0.80 & 0.09 \\ 1.61 & 0.01 & 0.28 & 0.74 \end{pmatrix}$$

Observemos que la matriz V es el modelo latente, aqui tenemos tantas columnas como usuarios pero solo dos filas. La primera fila representa el concepto latente de fantasía mientras que la segunda fila representa la ciencia ficción. Notar como para cada usuario se puede ver claramente que tipo de películas prefiere.

Para estimar las calificaciones que nos faltan en la matriz A realizamos U*V

$$U * V = \begin{pmatrix} pelicula & U1 & U2 & U3 & U4 \\ TheHobbit & 0.8 & 3.6 & 4.2 & 0.8 \\ StarWars & 4.1 & 0.2 & 0.9 & 1.9 \\ Terminator & 5.6 & 0.1 & 1.1 & 2.6 \\ HarryPotter & 0.3 & 4.2 & 4.8 & 0.6 \end{pmatrix}$$

Tomando de U*V los valores que nos faltaban en M podemos ver que para el usuario 1 estimamos 0.3 para Harry Potter, para el usuario 2 estimamos 0.2 para Star Wars y 4.2 para Harry Potter. Para el usurio 3 estimamos 0.9 para Star Wars y para el usuario 4 estimamos 0.8 para The Hobbitt. Notar como esto coincide con las preferencias de los usuarios que habíamos detectado.