OBLIGATORIO 2 – Métodos Numéricos

Noviembre 2014.

Descomposición SVD

en Tratamiento de Imágenes.

Nombre	Cédula
Fernando Fagúndez	4.296.514-1
Germán Faller	4.520.304-7
Andrés Grignola	4.435.560-9
Juan Lima	4.730.889-7

Contenido

Descomposición en valores singulares aplicada al Tratamiento de Imágenes.

En este obligatorio estudiaremos la descomposición de matrices en valores singulares (de aquí en más SVD). La misma consiste en una factorización de una matriz real o compleja, con muchas aplicaciones en la ingeniería. En particular, veremos su aplicación a la compresión de imágenes.

Parte 1: Descomposición en valores singulares.

Primeramente, procederemos a la demostración del teorema de Descomposición en valores singulares.

Una SVD de A es una factorización del tipo $A = U \sum V^T$, con $U \in R^{m\times m}$, $V \in R^{n\times n}$ ortogonales y $\sum \in R^{m\times n}$ una matriz formada con los Valores Singulares de A en su diagonal principal.

Teorema

Sea $A \in M(R)_{n\times n}$ de rango r. Existen tres matrices; $U \in R^{m\times m}$, $V \in R^{n\times n}$ y $\sum \in R^{m\times n}$ de la forma:

$$\sum = \begin{bmatrix} \sum_r & 0 \\ 0 & 0 \end{bmatrix}, \text{ con } \sum_r = \begin{bmatrix} G_1 & 0 \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 \cdots & G_r \end{bmatrix}$$

$\sum_r \in M(R)_{r\times r}$, donde a los G_i se los denomina valores singulares de A y satisfacen:

$G_1 \geq G_2 \geq \cdots \geq G_r > 0$. Entonces se cumple que $A = U \sum V^T$.

Demostración

Sea $G_1 = \|A\|_2$ donde $\|A\|_2 = \max_{\substack{z\in R^n \\ z\neq\vec{0}}} \frac{\|Az\|_2}{\|z\|_2}$ =$\max_{\substack{z\in R^n \\ \|z\|_2=1}} \|Az\|_2$

Existe $X \in R^n$, con $\|X\|_2 = 1$ tal que $\|AX\|_2 = \|A\|_2$ y además $\exists\, Y \in R^n$, con $\|Y\|_2 = 1$ tal que $AX = G_1 Y$ con $Y = \frac{AX}{G_1}$.

Sea $V = [X, V_1] \in R^{n\times n}$ matriz ortogonal construida a través de Gram-Schmidt

Sea $U = [Y, U_1] \in R^{m\times m}$ matriz ortogonal construida a través de Gram-Schmidt

Se puede observar que: ${U_1}^T AX = {U_1}^T G_1 Y = G_1\left({U_1}^T Y\right) = 0$, pues ${U_1}^T \perp Y$.

Definimos: $A_1 = {U_1}^T AV$ con $A_1 = \begin{bmatrix} Y^T \\ {U_1}^T \end{bmatrix} A[X, V_1] = \begin{pmatrix} Y^T AX & Y^T AV_1 \\ {U_1}^T AX & {U_1}^T AV_1 \end{pmatrix}$

Para ${U_1}^T AX = 0,\ Y^T AX = Y^T G_1 Y = G_1 {\|X\|_2}^2 = G_1$ Entonces,

$$A_1 = \begin{pmatrix} G_1 & W^T \\ 0 & B \end{pmatrix} \text{ con } \begin{bmatrix} W^T = Y^T AV_1 \\ B = {U_1}^T AV_1 \end{bmatrix}$$

Consideremos lo siguiente: $\left\|A_1 \begin{pmatrix} G_1 \\ W \end{pmatrix}\right\|_2$ Entonces: $\left\|A_1 \begin{pmatrix} G_1 \\ W \end{pmatrix}\right\|_2 = \left\|\begin{pmatrix} G_1 & W^T \\ 0 & B \end{pmatrix}\begin{pmatrix} G_1 \\ W \end{pmatrix}\right\|_2 = \left\|\begin{pmatrix} {G_1}^2 + W^T W \\ BW \end{pmatrix}\right\|_2 \geq {G_1}^2 + W^T W$

Analizamos $\|A_1\|_2$:

$$\|A_1\|_2 \geq \frac{\left\|A_1\begin{pmatrix} G_1 \\ W \end{pmatrix}\right\|_2}{\sqrt{{G_1}^2 + W^T W}} \geq \frac{{G_1}^2 + W^T W}{\sqrt{{G_1}^2 + W^T W}} \geq \sqrt{{G_1}^2 + W^T W} \Rightarrow \|A_1\|_2 \geq \sqrt{{G_1}^2 + W^T W} \quad (*)$$

Como U y V son ortogonales, se cumple lo siguiente: $\|A_1\|_2 = \left\|{U_1}^T AV\right\|_2 = \|A\|_2$, la última igualdad se verifica debido a que el producto de una matriz por otra ortogonal, no cambia la norma.

Llegamos entonces a que $\|A_1\|_2 = \|A\|_2 = G_1$.

Luego, por (*) tenemos que

$$G_1 = \|A_1\|_2 \geq \sqrt{{G_1}^2 + W^T W} \Rightarrow {G_1}^2 \geq {G_1}^2 + W^T W \Rightarrow W^T W = 0 \Leftrightarrow W = 0.$$

Por lo que se concluye que $A_1 = \begin{bmatrix} G_1 & 0 \\ 0 & B \end{bmatrix}$ con $B = U_1 AV_1 \in M(R)_{(m-1)\times(n-1)}$.

Aplicando el mismo razonamiento sobre B, nos queda:

$\sum = U^T AV \Rightarrow \sum V^T = U^T AVV^T \Rightarrow \sum V^T = U^T A$ (recordando que $VV^T = I_d$

Finalmente $A = U \sum V^T$

Implementación en Octave:

Se desarrolló la función mysvd en Octave, la cual realiza la descomposición SVD mediante el algoritmo de Golub-Kahan.

Dada una matriz $B \in R^{n\times n}$, bidiagonal estricta (no tiene ceros en la diagonal ni en la superdiagonal), este proceso iterativo utiliza matrices de Givens, buscando anular el elemento superdiagonal, el de la posición $(n-1, n)$.

Dado $\varepsilon > 0$pequeño, se considera que Bbidiagonal, es estricta si $|b_{ii}| \geq \varepsilon\|B\|$, $i = 1 \dots n$,

$|b_{i,i+1}| \geq \varepsilon(|b_{ii}| + |b_{i+1,i+1}|)$, $i = 1 \dots n$-1,

Datos: $B_0 \in R^{n\times n}$, ε
$B \leftarrow B_0$
Mientras Bes bidiagonal estricta
    $T \leftarrow B^T B$
    Obtener los valores propios de $T(n-1{:}n, n-1{:}n)$
    $u \leftarrow$valor propio más cercano a t_{nn}
    $y \leftarrow t_{11} - u$
    $z \leftarrow t_{12}$
    Para $k = 1{:}n-1$
        $[c, s] \leftarrow$csGivens (x, y)
        $B \leftarrow BG(k, k+1, c, s)$
        $y \leftarrow b_{kk}$
        $z \leftarrow b_{k+1,k}$
        $[c, s] \leftarrow$csGivens (x, y)
        $B \leftarrow BG(k, k+1, c, s)^T B$
        Si $k \leq n-2$
            $y \leftarrow b_{k,k+1}$
            $z \leftarrow b_{k,k+2}$
        Fin-Si
    Fin-Para
Fin-Mientras

Comparación entre mysvd y svd nativo de Octave

Se tomaron matrices aleatorias de distintos tamaños como punto de partida para ambas funciones, con resultados dispares.

Como primer referencia, cuando comparamos el tiempo de procesamiento de una misma matriz en una y otra función, queda claro que el svd nativo de octave es más eficiente que el desarrollado por nosotros. Esto se hace mucho más evidente en la medida que el tamaño de la matriz aumenta.

Como ejemplo de esto último, se ejecutó la descomposición svd de la imagen dada con la función mysvd y llevó más de 9 horas.

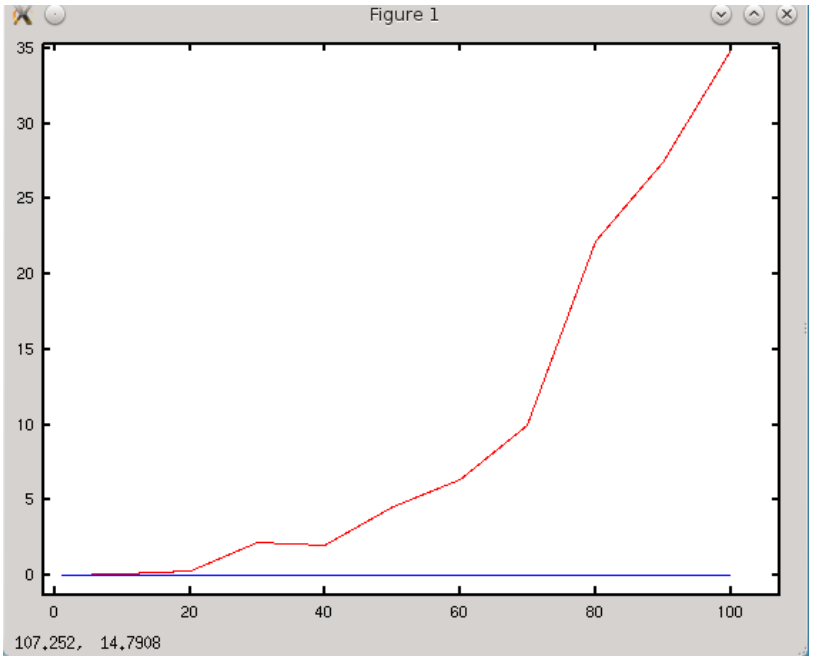

Ilustración 1 - Gráfica de Tiempos de Procesamiento, en rojo mysvd, en azul svd nativo. Eje x tamaño de matriz cuadrada, Eje y, tiempo en segundos.

Por otro lado, si comparamos las gráficas de la relación de compresión vs error en cada caso, los resultados son similares; a mayor compresión, mayor error.

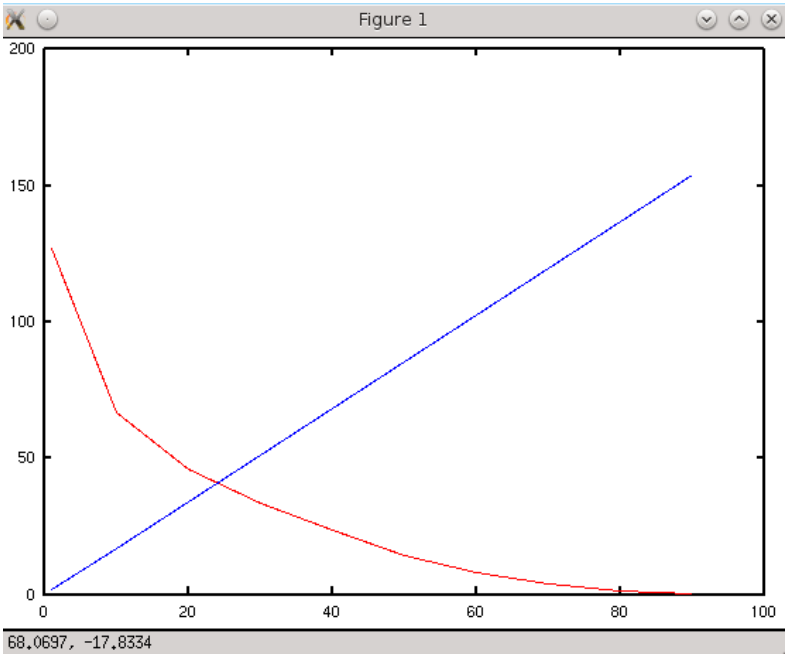

Ilustración 2 – Tamaño de archivo (azul) vs Error (rojo) Eje x cantidad de valores singulares significativos, Eje y %del tamaño de archivo o error (mse), según el caso – mysvd

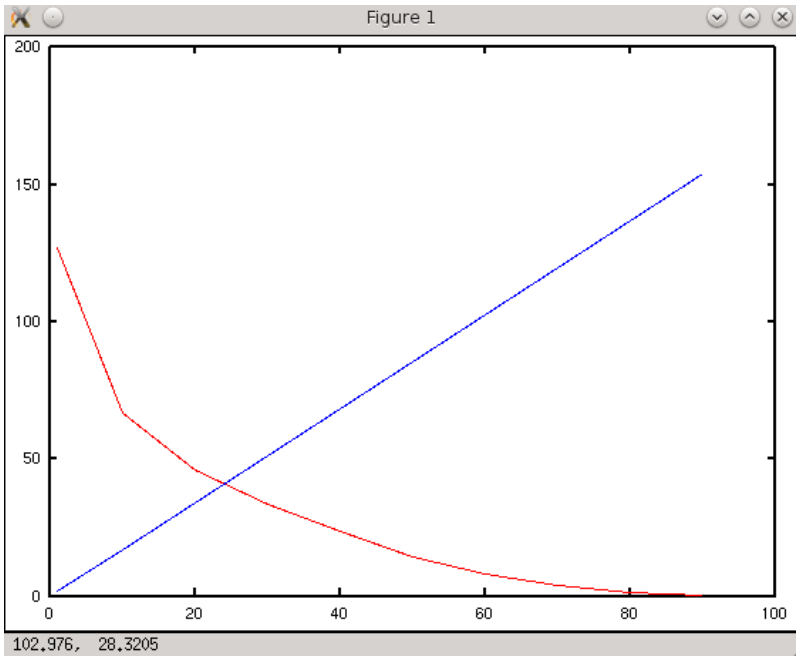

Ilustración 3 - Tamaño de archivo (azul) vs Error (rojo) Eje x cantidad de valores singulares significativos, Eje y %del tamaño de archivo o error (mse), según el caso - svd nativo de Octave

Teniendo en cuenta lo anterior, podemos afirmar que la función svd nativa es más escalable que la mysvd, ya que la tendencia de los tiempos que insume la segunda la torna impracticable.

Parte 2: Uso en Tratamiento de Imágenes.

Ilustración 4 - Imagen a procesar 'Fing.bmp'

En esta parte se trabaja haciendo una descomposición en valores singulares sobre la imagen dada (Ilustración 4).

Luego de descomponerla, podemos evaluar la efectividad del proceso para mejorar el almacenamiento de los datos.

Evaluamos 4 casos representativos: k=1, 20, 50 y 90, donde k es la cantidad de valores singulares más significativos de la matriz de datos de la imagen.

Se observa que al tomar k=90 el proceso es contraproducente, necesitando más espacio de almacenamiento para la misma calidad de imagen; aproximadamente aumenta un 50% el espacio necesario. En k=50 en cambio, la compresión baja a un 80% del valor original, manteniendo el error MSE por debajo de 20. Cuando evaluamos con k=20, la compresión llega a un 25% del valor original, mientras que el error ronda el 45. En el caso crítico de k=1, la compresión es cercana a cero y el error asciende a 130.

La descomposición SVD no implica necesariamente pérdida de calidad en la imagen, siempre y cuando se tome un valor de k suficientemente grande, se pueden despreciar los errores asociados. Tampoco implica necesariamente una reducción en el tamaño de la imagen, depende de la complejidad de la imagen y del error que se esté dispuesto a contemplar.

Es posible afirmar que el sistema permite buenos ratios de compresión con un margen de error relativamente chico, con lo cual es una herramienta útil para el tratamiento de imágenes.

Reconstrucción de Imagen en Valores Singulares variando la cantidad de valores considerados

A continuación veremos los resultados de la reconstrucción de la imagen 'Fing.bmp' utilizando los valores de k definidos anteriormente; k=1, 20, 50 y 90.

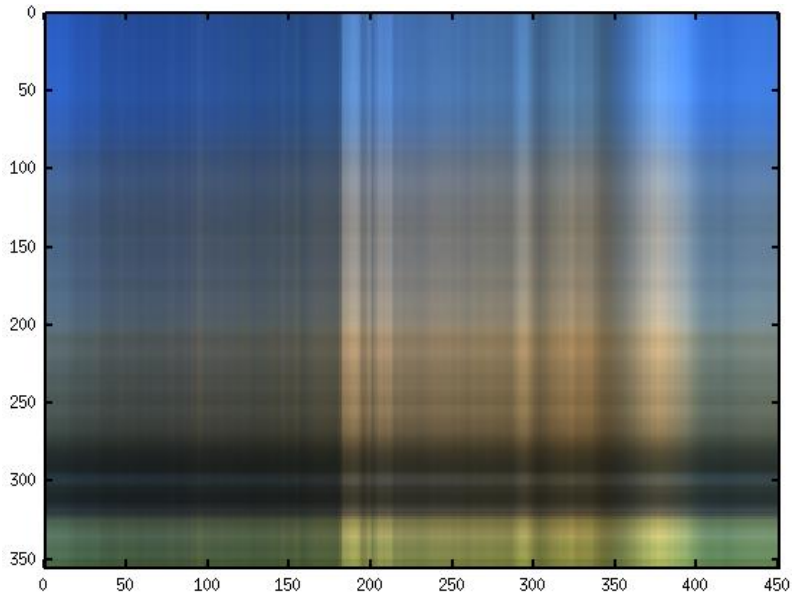

Ilustración 5 - Imagen 'Fing.bmp' luego del SVD, con k=1

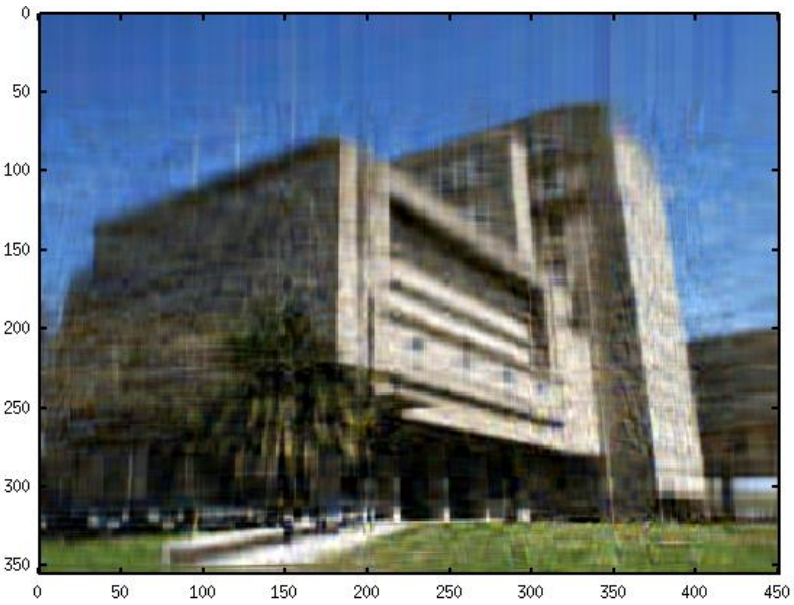

Ilustración 6 - Imagen 'Fing.bmp' luego del SVD, con k=20

Ilustración 7 - Imagen 'Fing.bmp' luego del SVD, con k=50

Ilustración 8 - Imagen 'Fing.bmp' luego del SVD, con k=90

Caso	Espacio Ocupado sobre el Espacio Original	Calidad de Imagen – Error en MSE
k=90	150%	Cercano a 0
k=50	80%	Cercano a 20
k=20	30%	Cercano a 45
k=1	1%	Cercano a 130

Representación de la imagen en Valores Singulares con Calidad Dada

En última instancia calculamos, con el error en MSE dado la cantidad mínima de Valores Singulares que hay que considerar para obtener la imagen con un error menor a 0,0028.

Utilizando el programa de Octave, nos queda que con un k=313, el error desciende a 0,0027835.

La siguiente imagen es el resultado:

Ilustración 9 - Imagen 'Fing.bmp' luego del SVD, con k=313 y un error de 0,0027835

Obviamente no es perceptible a simple vista la diferencia entre la imagen original y ésta última

En cuanto a la compresión final de la imagen, el resultado es coherente con lo expuesto antes; la imagen resultante ocupa más almacenamiento que la original.

Probando con imágenes menos complejas (mismo tamaño, un solo color), la compresión es mucho mayor, ya que los Valores Singulares más significativos tienen la mayoría de la información, es decir, para alcanzar el mismo nivel de error (MSE) se pueden considerar menor número de k (menor cantidad de valores singulares). Sucede exactamente lo contrario para imágenes más complejas.

Conclusiones

Luego de las pruebas, pudimos constatar que:

- El método SVD para compresión de imágenes en este caso es efectivo, solamente si se concibe un margen de error. Que aumenta directamente al aumentar la compresión, de forma no lineal.
- Se puede asumir que si la imagen a comprimir es relativamente sencilla, el método se torna mucho más efectivo.
- Estudiando otros algoritmos para aplicar SVD, sería posible alcanzar rutinas más eficientes en el tiempo. Un claro ejemplo de ello se observa en la comparación entre mysvd y svd nativo.