Álgebra Lineal Computacional

Primer Cuatrimestre 2023

Trabajo Práctico N° 2

Indroducción

El objetivo de este trabajo es utilizar el método de descomposición de matrices en valores singulares (SVD) para resolver el problema de identificación de dígitos manuscritos en imágenes, es decir los números del 1 al 9, en forma automática.

Figure 1: Ejemplos del conjunto de imágenes de prueba de Mnist [3][4].

Una de las ideas más fructíferas en la teoría de matrices es la de la descomposición matricial, que se ha convertido a lo largo de las últimas décadas del siglo veinte y principios de éste, en el pilar fundamental de algoritmos computacionales para poder resolver diversos problemas [1][2].
Los dígitos manuscritos son muy comunes en nuestras vidas y hay casos en los que es necesario reconocer dígitos o letras manuscritas en forma automática para acelerar procesos de ingreso de datos, llenado de formularios, etc.
Es por ello que se utilizará la SVD para resolver la clasificación de un determinado número manuscrito en una imagen.
Supongamos que tenemos un conjunto de k imágenes, a las cuales llamaremos "imágenes de entrenamiento", y que además se han clasificado. Esto quiere decir que se cuenta con imágenes de dígitos de las cuales se sabe a qué dígito corresponde cada una. Luego usaremos estas imágenes para formar nuestro "sistema de reconocimiento".
Las imágenes de entrenamiento son imágenes en escala de grises que tienen una dimensión de 28×28 píxeles. Cada imagen se puede representar por una matriz de 28×28, es decir que cada elemento de la matriz representa un píxel de la imagen. El valor de cada elemento de la matriz puede tomar un valor entre 0 y 255 que se corresponde a la escala de grises, siendo 0 el negro y 255 el blanco.

Ejercicio 1.
Se dispone de dos archivos en formato CSV, **mnist_train.csv** y **mnist_test.csv** [4], que se corresponden con las imágenes de entrenamiento y testeo respectivamente. Estos datos fueron extraídos de la base de datos MNIST [*], y contienen 60.000 imágenes de entrenamiento con sus correspondientes etiquetas de clasificación y 10.000 imágenes de prueba para identificar.
Estos archivos guardan las imágenes en forma "vectorizada", una fila se corresponde con una imagen. Cada fila tiene 785 columnas, donde la primer columna indica qué dígito guarda y el resto es la imagen de 28×28 píxeles (784) ordenada por filas.

(a) Realizar una función en Python que dado los datos de las imágenes de entrenamiento y una fila, grafique la imagen guardada en esa fila y en el título del gráfico se indique a qué número corresponde, es decir su clasificación. Usar la función `imshow()` de Pyplot.

(b) ¿Cuántas imágenes hay por cada dígito en el conjunto de entrenamiento? ¿Y en el conjunto de testeo?

(c) Para las primeras 2.000 imágenes del conjunto de entrenamiento realizar una función en Python que devuelva la imagen promedio de cada uno de los dígitos.

(d) Graficar cada una de las imágenes promedio obtenidas.

Ejercicio 2.
Una forma sencilla de tratar de identificar las imágenes de testeo, es tomar la imagen promedio para cada dígito hallada en el ejercicio 2(c) y ver cuán diferente es para el caso que queremos testear. Una medida posible de esta diferencia puede ser calcular la distancia Euclídea entre la imagen promedio de cada dígito y la imagen con el dígito que queremos clasificar. De esta manera, aquella que tenga menor distancia será la respuesta al problema.

(a) Realizar una función en Python que dadas las imágenes promedio del ejercicio 2(c), calcule la menor distancia Euclídea entre todos los dígitos y cada una de las primeras 200 imágenes de testeo. La función debe devolver un arreglo con las 200 predicciones.

(b) Realizar una función en Python que tome el arreglo de predicciones anteriores y evalúe si es correcta o no la predicción. Debe devolver la precisión en la predicción. Se define la precisión como:

$$Precisión = \frac{\Sigma(Casos\_acierto)}{\Sigma(Casos\_totales)}$$

(c) Graficar un par de casos de imágenes de testeo en los cuales no se haya acertado. ¿Considera buena la precisión?

Ejercicio 3.
Implemetar una función en Python que dada una matriz A halle la descomposición SVD de A, por el método de la potencia.
Llamamos descomposición SVD en valores singulares a:

$$A = U\Sigma V^T \tag{1}$$

Donde U es una matriz ortogonal, Σ una matriz diagonal con la raíz de los autovalores de A^tA en la diagonal y V contiene a los autovectores singulares en sus columnas.
El método de la potencial utiliza el siguiente resultado para asegurar la convergencia:

$$B = A^tA = (U\Sigma V^t)^t(U\Sigma V^t) = V(\Sigma^t\Sigma)V^t \tag{2}$$

Algoritmo:

- Inicialmente multiplicar un vector aleatorio x_0 de norma 1 a B y luego continuar iterando utilizando la siguiente regla:

$$x_1 = Bx_0 \tag{3}$$

$$x_{k+1} = Bx_k \tag{4}$$

para encontrar el primer autovector (con máximo autovalor).

- En cada paso se debe volver a normalizar el siguiente x_k.
- Continuar mientras $(x_{k+1}^t x_k) < (1-\epsilon)$, que es el criterio de parada para el algoritmo iterativo.
- Devolver x_{k+1} que es la aproximación al autovector buscado v_1.
- Obtenemos $\sigma_1 = \|Av_1\|$ y $u_1 = Av_1/\|Av_1\|$.
- Se procede recursivamente y se calcula la nueva matriz A' sobre la cual se va a iterar nuevamente:
$$A' = A - \sigma_1 u_1 v_1^t \tag{5}$$
para encontrar el siguiente autovector v_2 y luego σ_2 y u_2.
- Continuar hasta encontrar k autovectores que se corresponden con la cantidad de columnas de A.
- La función finalmente debe devolver las matrices U, Σ y V.

Ejercicio 4.
Se utilizará la descomposición SVD para resolver la clasificación de imágenes correspondiente a números manuscritos.

(a) Tomar las primeras 2.000 imágenes del conjunto de imágenes de testeo y ordenarlas según el dígito al que corresponde de 0 a 9. Obtener 10 matrices correspondientes a cada dígito. Estas matrices deben tener una dimensión de 785 $\times$ cantidad imágenes, puede no haber la misma cantidad de imágenes para cada dígito en las primeras 2.000 imágenes. Recordar que la primer columna es la clasificación. Finalmente obtener $M_{i=0,\dots,9}$ matrices de 784 $\times$ cantidad imágenes quitando la primer columna. Se pueden guardar las matrices en un arreglo de tipo lista donde cada ítem de la lista se corresponde con una matriz M_i y la posición hace referencia al dígito que representan.

(b) Realizar la descomposición SVD de cada una de las matrices M_i utilizando la función creada en el ejercicio (3). Para ello realizar una función en Python que tome la lista de matrices M_i y devuelva en 3 listas la solución de la descomposición, es decir U_i, Σ_i y V_i.

(c) Las columnas de U_i son combinación lineal del espacio columna de M_i. Teniendo esto presente tomar la primer columna de cada U_i y graficarla como imagen, es decir convertir a una matriz de 28×28 y graficar. Explique qué representa.

(d) Repetir el ítem anterior pero para las columnas 2 y 3 de cada una de las U_i. Comparar con lo obtenido en (c) y explicar las diferencias.

(e) El teorema de la descomposición en valores singulares nos asegura que dada una matriz $A \in \mathbb{R}^{m\times n}$ con rango r, y $\sigma_1, \dots, \sigma_r$ los valores singulares de A, con $u_1, \dots, u_r$ y $v_1, \dots v_r$ las primeras r columnas de las matrices U y V respectivamente, y sea $k \leq r$, se tiene que:
$$A = U\Sigma V^t = \sum_{j=1}^{r} \sigma_j u_j v_j^t \tag{6}$$
Y la matriz de rango k que mejor aproxima a la matriz A en norma 2 es:
$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^t \tag{7}$$

Utilizaremos este resultado para encontrar distintas aproximaciones de U_i, que nos permitirá componer el espacio imagen de U_i con aproximación en rango k, y de esta forma poder encontrar la menor distancia entre una imagen de testeo x y el espacio generado por las primeras k columnas de U_i.

Para ello:

- Sea $\hat{U}_{i,k} \in \mathbb{R}^{784\times k}$ la matriz que resulta de tomar las primeras k columnas de U_i (hacerlo para k de 1 a 5).
- Obtener la matriz $\hat{U}_{i,k}\hat{U}_{i,k}^t$, es decir, la matriz que proyecta ortogonalmente sobre la imagen de $\hat{U}_{i,k}$.
- Obtener el residuo como: $r_{i,k}(x) = x - \hat{U}_{i,k}\hat{U}_{i,k}^t x$, donde x es la imagen (vectorizada) de la cual se quiere conocer la mínima distancia al subespacio generado por la imagen de $\hat{U}_{i,k}$.
- Guardar el $\hat{i}$ cuyo residuo es el menor, es decir, $\hat{i} = \min\{i : \|r_{i,k}(x)\|\}$, para $i = 0, \ldots, 9$. Ésta es la predicción para la imagen x, en la aproximación de rango k.
- Comparar con el dígito esperado.

La función de Python debe tomar como entrada las matrices U_i obtenidas a partir de las primeras 2.000 imágenes del conjunto de entrenamineto (ya calculadas en el ejercicio 3), junto con las 200 primeras imágenes de testeo, y devolver la precisión (como se calculó en el ejercicio 2.b), para cada valor de aproximación en rango k. Es decir se obtendrán 5 valores distintos de acuerdo a los valores de k utilizados.

Ejercicio 5.
Finalmente comparar los resultados obtenidos para la precisión entre los ejercicios 2 y 4. ¿Qué puede observar? Graficar y comentar aquellos casos en los cuales la predicción falló en el ejercicio 4 para distintos k. Por ejemplo, algún caso en el que la predicción haya fallado con algún k y luego con otro k haya sido exitosa.

[1] G. W. Stewart, On the early history of the singular value decomposition, SIAM review, 35 (1993), pp. 551–566.
[2] G. Strang, Introduction to linear algebra, vol. 3, Wellesley-Cambridge Press Wellesley, MA, 1993.
[3] Y. LeCun, C. Cortes, C.Burges, The Mnist Database, http://yann.lecun.com/exdb/mnist/
[4] El conjunto de datos en formato CSV fue extraido de: https://pjreddie.com/projects/mnist-in-csv/ en la fecha 22/05/2023.