Trabajo Práctico III

Matías Pérez (2/05)
Rodrigo Campos Catelin (561/06)

December 5, 2010

Abstract

En el siguiente trabajo se utiliza la técnica de PCA (principal component analysis) para generar a partir de una imágen digital un archivo de texto plano que se le asemeje.

Para esto se contará con un conjunto de imágenes de entrenamiento con los 92 caracteres ASCII en alguna tipografía con los cuales se comparará con subimágenes de la imágen original y decidirá a cuál de los 92 caracteres se asemeja más.

Se analizarán los distintos criterios de parada para los algoritmos iterativos utilizados y la cantidad de componentes principales a considerar.

Palabras clave: Principal component analysis, algoritmo QR, autovalores, autovectores, ASCII Art.

1 Introducción teórica

1.1 Descomposición QR

Dada $A \in \mathbb{R}^{n \times n}$, una descomposicion QR de la matriz consta en dos matrices $Q \in \mathbb{R}^{n \times n}$, $R \in \mathbb{R}^{n \times n}$ tal que:

- Q es ortogonal
- R es triangular superior
- $A = QR$

Un método para lograr la factorización QR es el método de Givens. Este consiste de rotaciones de los vectores de la matriz, de forma tal que las componentes de la matriz por debajo de la diagonal se anulen. Para ello se construyen *matrices de rotacion* $G^{a,b}$ para anular el elemento $A_{a,b}$, estas tienen la forma:

$$G^{a,b}_{i.j} = \begin{cases} 1 & i = j \wedge i, j \neq a, b \\ 0 & i \neq j \wedge i, j \neq a, b \\ c & (i = a \wedge j = a) \vee (i = b \wedge j = b) \\ -s & (i = a \wedge j = b) \\ s & (i = b \wedge j = a) \end{cases}$$

Con $c = \frac{A_{b,b}}{\sqrt{(A_{b,b})^2 + (A_{a,b})^2}}$ y $s = \frac{A_{a,b}}{\sqrt{(A_{b,b})^2 + (A_{a,b})^2}}$
Entonces:

$$G^{n,n-1}...G^{n,2}...G^{3,2}G^{n,1}...G^{2,1}A = R$$

$$G^{n,n-1}...G^{n,2}...G^{3,2}G^{n,1}...G^{2,1} = Q^{-1}$$

Como Q es ortogonal:

$$Q = \left(G^{n,n-1}...G^{n,2}...G^{3,2}G^{n,1}...G^{2,1}\right)^{\mathsf{T}}$$

Vale aclarar que las matrices G de un paso se generan sobre el producto de la matriz original y las matrices G anteriores.

1.2 Método QR para autovalores y autovectores

Para el cálculo de los autovalores de una matriz se utilizó el *algoritmo QR* el mismo consta en:

- A partir de $A^{(1)} \in \mathbb{R}^{n \times n}$ (con autovalores reales) se genera su descomposición QR, obteniendo $Q^{(1)} \in \mathbb{R}^{n \times n}$, $R^{(1)} \in \mathbb{R}^{n \times n}$.
- Se itera generando $A^{(i+1)} = R^{(i)}Q^{(i)}$
- Hasta que $A^{(n)}$ es triangular superior.

Mediante este método se obtiene en la diagonal de $A^{(n)}$ los autovalores de la matriz original ($A^{(1)}$). Si la matriz $A^{(1)}$ es simétrica, la matriz final $A^{(n)}$ resulta ser diagonal.

Como:

$$A^{(n)} = R^{(n-1)}Q^{(n-1)}$$
$$A^{(n)} = Q^{(n-1)\intercal}Q^{(n-1)}R^{(n-1)}Q^{(n-1)}$$
$$A^{(n)} = Q^{(n-1)\intercal}A^{(n-1)}Q^{(n-1)}$$

$$...$$

$$A^{(n)} = Q^{(n-1)\intercal}...Q^{(1)\intercal}A^{(1)}Q^{(1)}...Q^{(n-1)}$$

Sea $E = Q^{(1)}...Q^{(n-1)}$:

$$A^{(n)} = E^{\intercal}A^{(1)}E$$

$$EA^{(n)} = A^{(1)}E$$

$$A^{(1)}E = EA^{(n)}$$

Como $A^{(n)}$ es diagonal se puede ver que:

$$col_i\left(A^{(1)}E\right) = A^{(1)}col_i(E) = col_i(E)A^{(n)}_{i,i} = col_i\left(EA^{(n)}\right)$$

Sea $x = col_i(E)$, $\lambda = A^{(n)}_{i,i}$:

$$A^{(1)}x = \lambda x$$

Entonces se puede ver que los autovalores resultan los elementos de la diagonal (como ya se dijo previamente) y los autovalores las columnas de la matriz E.

2 Desarrollo

El problema a resolver es dada una imágen digital tratar de representarla con caracteres ASCII.

Para esto se divide la imágen original en tantas sub-imágenes como caracteres de ancho y alto se desee y luego se elije el caracter que más se asemeje a la sub-imágen. Esto implica varios puntos relevantes, tanto la descomposición en sub-imágenes como la elección del caracter más representativo entre un conjunto de entrenamiento de imágenes de caracteres ASCII.

Para la descomposición en sub-imágenes la primer restricción a tener en cuenta es que ésta debe ser del mismo tamaño que las imágenes de entrenamiento. Además la cantidad de caracteres en una línea (*ancho*) y la cantidad de líneas (*alto*) de la imágen representada en ASCII debe ser configurable. Esto se traduce en una cantidad fija de sub-imágenes. Teniendo en cuenta que las imágenes de entrenamiento son de tamaño fijo $h \times w$ se decidio escalar la imágen de entrada a $h.ancho \times w.alto$. De esta manera, tomado cuadrantes de tamaño $h \times w$ de la imagen escalada se consiguen $ancho \times alto$ sub-imágenes.

Para elegir el caracter más representativo se debe tener en cuenta que los caracteres ASCII no tienen colores. Por lo que es preferible analizar la imágen original convertida a escala de grises. Tanto las imágenes de entrenamiento como las sub-imágenes se representarán con un vector de $m = h * w$ coordenadas. Dado que m puede ser un valor considerablemente grande, como sugiere el enunciado del trabajo práctico, se utiliza la técnica PCA para reducir la cantidad de coordenadas a considerar.

El primer paso es, a partir de los vectores de entrenamiento calcular la matriz de covarianza asociada M y sus autovalores y autovectores. Para calcular los autovalores y autovectores, y teniendo en cuenta que M es simétrica, se utilizó el algoritmo QR. Dicho procesamiento es computacionalmente costoso y es independiente de la imágen que se desea transformar, por lo que se decidió guardar los resultados del cómputo de forma persistente en archivos de texto. También se decidió persistir la transformación característica asociada a los autovalores de la matriz aplicada a cada vector de entrenamiento, ya que ésto también es independiente de la imágen a procesar.

Luego, se aplica la transformación característica asociada a un subconjunto de k elementos de los autovectores calculados a los vectores de cada cuadrante de la imágen. Y se lo compara con las primeras k componentes de los vectores de entrenamiento ya transformados para elegir el más similar y asignarle el correspondiente caracter ASCII.

2.1 Escalado de la imágen

Para escalar la imágen se utilizó la librería *ImageMagick*, en particular el comando *convert*. Al mismo tiempo que se escala la imágen se la tranforma a formato RAW en escala de grises.

2.2 Descomposición QR

Para lograr la desomposición QR de la matriz M se decidió utilizar el método de las rotaciones de Givens.

La implementación que resulta de una traducción inmediata del método resultó ser muy poco eficiente. Luego de analizar las funciones que más costo computacional tenían, se encontró una mejora en la multiplicacion de las matrices de Givens generaría un aumento de performance considerable. Dada la particular forma de las matrices de Givens vale que:

$$fila_i(G^{a,b}A) = \begin{cases} fila_i(A) & i \neq a \wedge i \neq b \\ fila_i(G^{a,b})A & sino \end{cases}$$

Aprovechando esto, para obtener el resultado sólo es necesario calcular la multiplicación de las filas a y b de la matriz, ahorrando así una gran cantidad operaciones en cada multiplicación.

2.3 Algoritmo QR

Para realizar el cálculo de autovalores y autovectores, se realizó el algoritmo de QR previamente explicado. Dicho algoritmo es un algoritmo iterativo, como tal se le debe definir un criterio de parada. Los considerados en este trabajo fueron:

- EsDiagonal: Se fija si la matriz $A^{(i)}$ es una matriz diagonal.
- EsTriangularSuperior: Se fija si la matriz $A^{(i)}$ es una matriz triangular superior.
- TieneDiferenteDiagonal: Se fija si la matriz $A^{(i)}$ y $A^{(i-1)}$tienen diferentes elementos en la diagonal.

Todos estos criterios tienen una cota de configurable para la tolerancia. Es decir, si la diferencia entre dos números es menor que la tolerancia, se consideran iguales. Finalmente se decidió que una combinación de *EsDiagonal* y de *TieneDiferenteDiagonal* es un buen criterio de parada, ya que si la matriz no es diagonal los autovectores no serán adecuados, y si los autovalores cambian entre una iteración y la siguiente (tienen diferente diagonal) estos valores no son suficientemente precisos.

2.4 Semejanza entre vectores

Para definir cuál es el vector que más se asemeja a otro se realizó una busqueda lineal calculando la distancia entre dos vectores y quedandose con aquel de menor distancia. Para definir la distancia (d) entre dos vectores se definieron varios criterios teniendo en cuenta el dominio del problema.

Los criterios elegidos son todos una suma pesada de las distancias coordenada a coordenada, esto se consideró teniendo en cuenta que una componente del

vector tiene mayor relevancia que las siguientes. Quedando la distancia entre dos vectores $x, y \in \mathbb{R}^n$:

$$d = \sum_{i=1}^{n} \beta_i |x - y_i|$$

- Criterio 1: $\beta_i = 1$ Esto resulta en la suma del modulo de las distancias.
- Criterio 2: $\beta_i = i$ Se priorizan las primeras componentes.
- Criterio 3: $\beta_i = 100^{n-i}$ Se priorizan las primeras componentes.
- Criterio 4: $\beta_i = 2^{n-i}$ Se priorizan las primeras componentes.
- Criterio 5: $\beta_i = \frac{|a_i|}{\sum_{j=1}^{n} |a_j|}$ Siendo a el vector de autovalores ordenado de mayor a menor en módulo.

Finalmente el criterio utilizado es el criterio 5.

2.5 Componentes principales

La cantidad de componentes principales a utilizar para procesar una imagen se decidió que sea un parámetro opcional, de no especificar ninguno, se utilizarán 10.

3 Resultados

Probando los diferentes criterios de distancia, la cantidad de componentes principales y la presicion para el calculo de autovalores y autovectores se obtuvieron diferentes resultados.

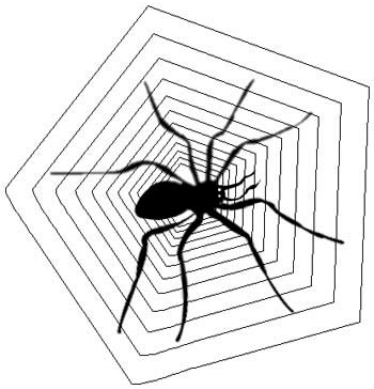

Figura 1: Imagen a procesar

Figura 2: K: 10, precisión:10^{-7}, criterio: 3

Figura 3: K: 1, precisión:10^{-7}, criterio: 5

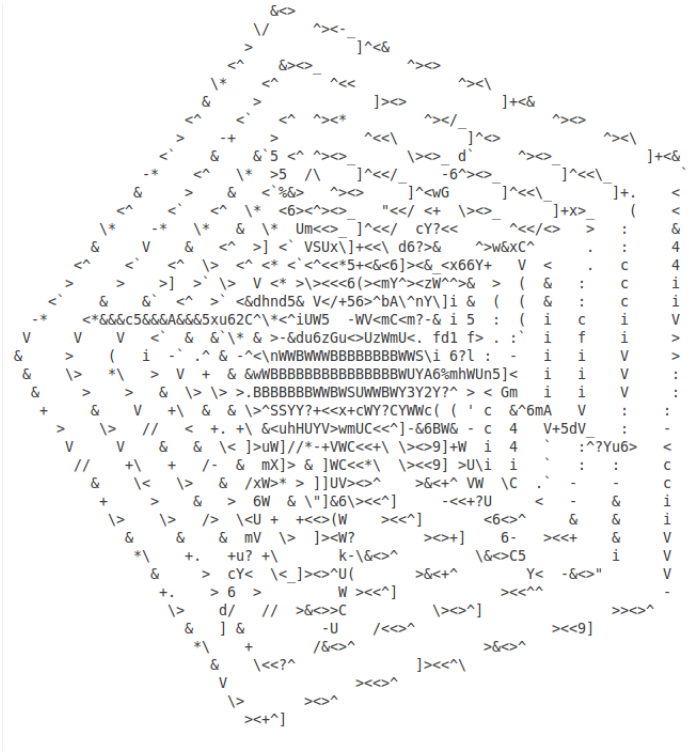

Figura 4: K: 3, precisión:10^{-7}, criterio: 5

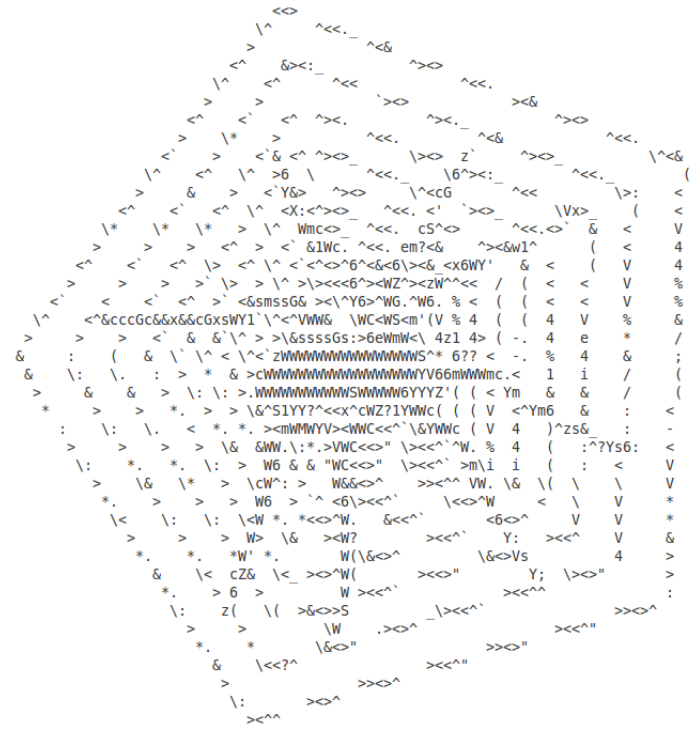

Figura 5: K: 10, precisión:10^{-7}, criterio: 5

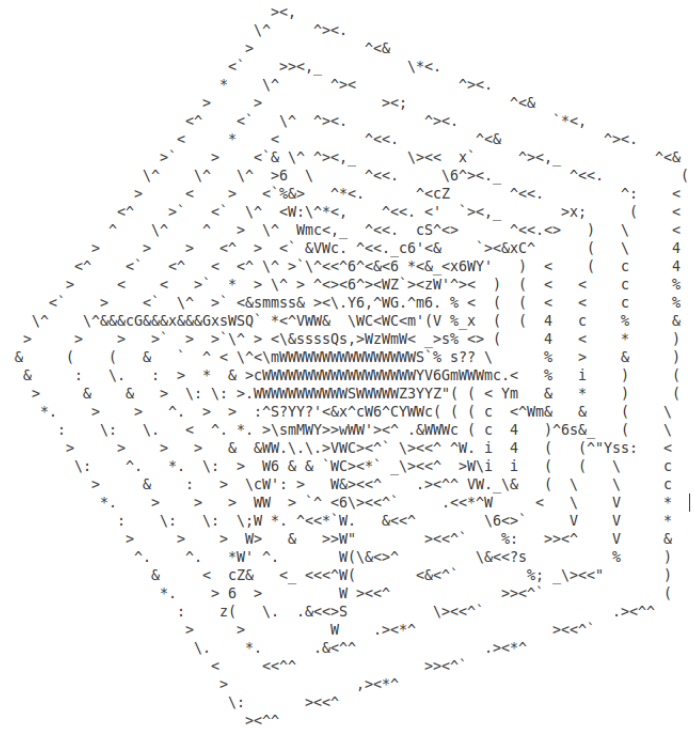

Figura 6: K: 100, precisión:10^{-7}, criterio: 5

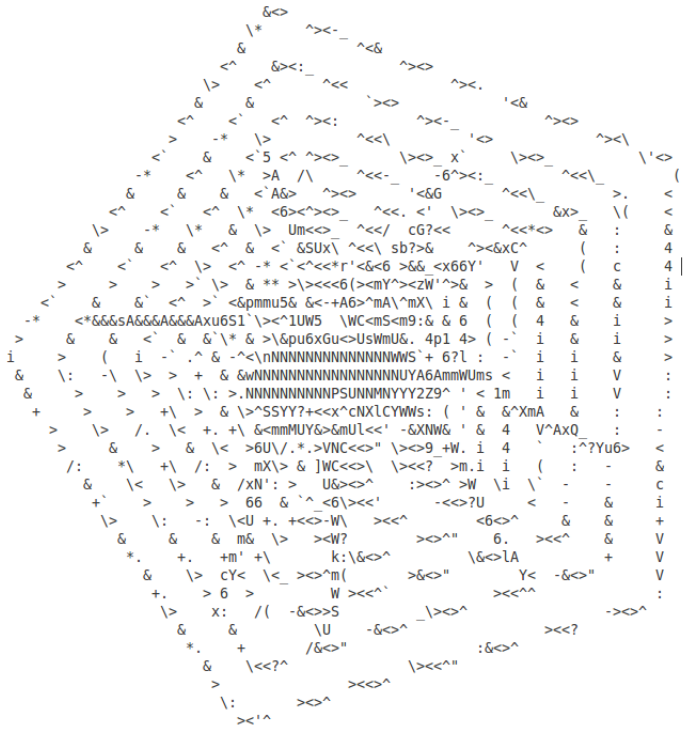

Figura 7: K: 10, precisión:10^{-7}, criterio: 4

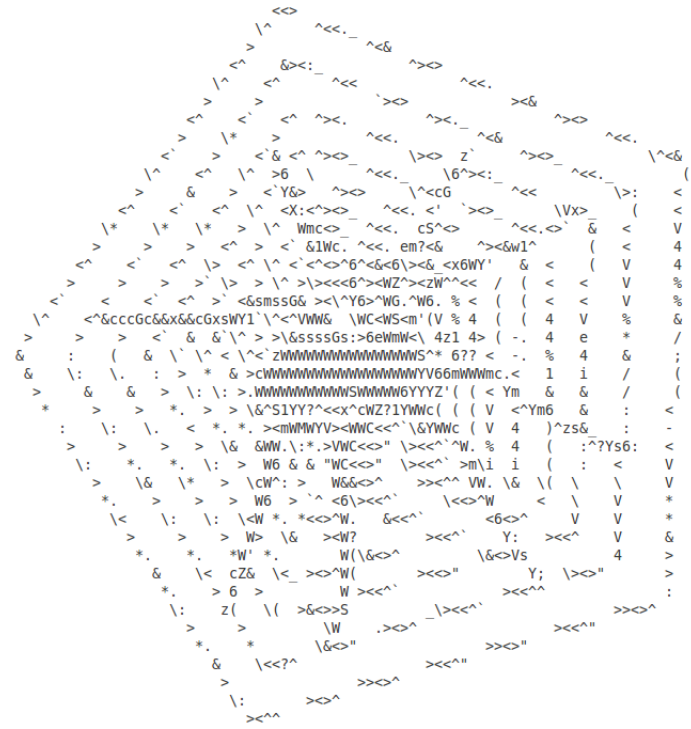

Figura 8: K: 10, precisión:10^{-1}, criterio: 5

Se notó que si se escala una imagen a varias veces su tamaño y se utiliza un tamaño de fuente pequeño para visualizarlo, el resultado que se logra es considerablemente fiel a la imagen original.

Figura 9: Imagen a procesar

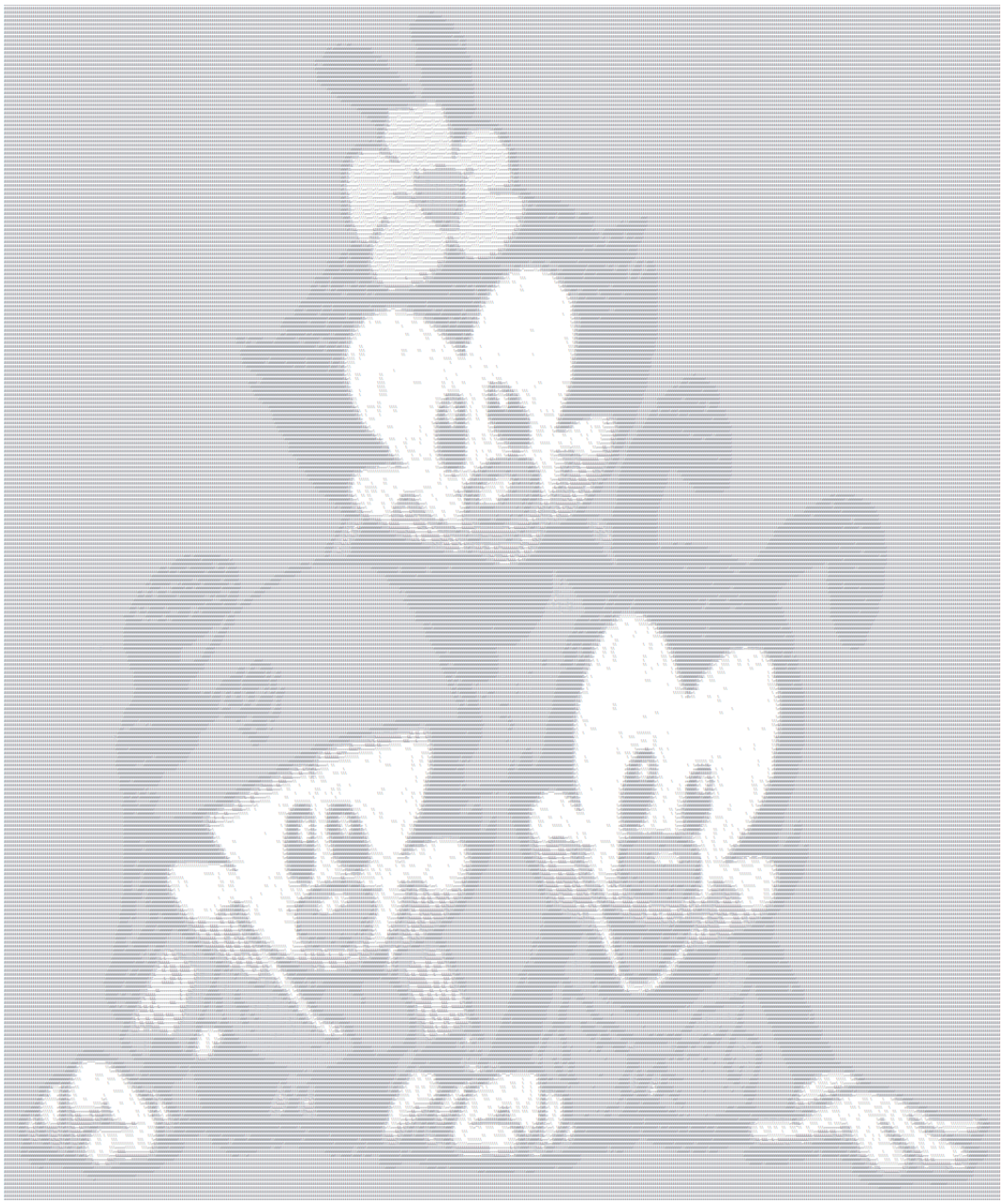

Figura 10: Imagen procesada, K: 209, precisión:10^{-7},criterio: 5

4 Discusión

Luego de obtener los resultados presentados en la sección anterior, se pueden obsevar distintos aspectos del comportamiento del progrma.

4.1 Criterios de distancia

En cuanto a los criterios considerados del calculo de la distancia entre dos vectores se puede ver que el criterio 3 es claramente inferior ante los criterios 4 y 5. Se puede observar que entre estos dos ultimos criterios las diferencias no son demasiado considerables y no queda claro cuál se asemeja más a la imagen original. Pero se decidió por utilizar el criterio 5, ya que nos pareció intuitivamente mejor y nos gusta más la idea detrás del mismo.

4.2 Componentes principales

Se puede ver en los gráficos 3 al 6 cómo existe una mejora al aumentar la cantidad de componentes principales tomadas en cuenta. Se ve cómo la imagen gana definición y se logra ver con más claridad la imagen original.

4.3 Precisión en el cálculo de los autovectores

Tras realizar varias pruebas con distintas precisiones no se pudo ver claramente una diferencia entre la salida como se puede ver en los gráficos de la seccion anterior, se muestra un ejemplo de esto en las figuras 8 (con precisión 10^{-1}) y la figura 7 (con precisión 10^{-7}).

Si bien experimentalmente no se han encontrado mayores mejoras en los resultados al aumentar solo esta variable, decidimos dejar en la implementación definitiva 10^{-7}. El motivo principal de dicha decisión es que numéricamente existe una mejora considerable (que no se ve reflejada en nuestros experimentos), que el costo computacional, si bien es mayor, no es prohibitivo y dicho cómputo se realiza una sola vez sobre el conjunto de entrenamiento.

5 Conclusiones

Al finalizar el trabajo el resultado obtenido es considerado satisfactorio. Porque que se obtienen imágenes ASCII con un alto grado de semejanza con la imágen original.

Lo que más nos llamó la atención es que no pudimos encontrar ninguna diferencia en las imágenes ASCII generadas apartir de los autovectores calculados con distintas precisiones (10^{-1}y 10^{-7}), a pesar que los autovalores presenten una diferencia considerable (eran distintos a partir del segundo dígito en nuestros casos de prueba).

Otra curiosidad es el buen resultado obtenido por el criterio 4 sin justificación aparente. Los resultados son realmente muy parecidos a los obtenidos por el criterio 5, que es el que consideramos el mejor de los criterios y que posee una "justificación" más intuitiva.

También resulta interesante ver como la cantidad de componentes principales resulta una variable tan importante para obtener un buen resultado final. Se pudo ver como al aumentar las mismas, la precisión aumenta hasta llegar a cierto límite (con más de 100 componentes principales no hemos visto mejoras sustanciales) y considerando pocas componentes el resultado puede ser realmente pobre.

A la hora de realizar las pruebas y probar el funcionamiento del programa resultó útil la separación entre el procesamiento de las imágenes de prueba y de la imágen a transformar. El tiempo de cómputo necesario para generar dicha información es considerable, lo que hubiera complicado la realización de las pruebas y hubiera hecho prácticamente inusable el programa.

Una característica intrínseca del problema es que no existe una solución única. Por esto, no encontramos ninguna forma de medir la efectividad del programa. E hizo más dificil definir qué criterio es superior a otro. Sin embargo, la única prueba con resultado conocido que pudimos hacer es asegurarnos que usando como imágen de entrada uno de los caracteres de entrenamiento, la imágen ASCII generada sea ese mismo caracter.

6 Apéndice A

7 Apéndice B

Para compilar todos los métodos se debe ejecutar el comando "make". Se necesita tener el compilador gcc. Para generar la imágen ASCII se debe correr:

run-ascii-art.sh <src> <dst> <width> <height> [k]

Donde "src" es el path a la imágen original, "dst" es el archivo que contendrá la representación ASCII, "width" es la cantidad de caracteres por línea, "height" es la cantidad de líneas y "k" es un parámentro opcional que es la cantidad de componentes principales a utilizar.

Se asume que las imágenes de entrenamiento están en "../imgs", que existe el directorio "../data" y que está instalado el comando "convert" de ImageMagick

El ejecutable "gen-data" es el que procesa las imágenes de entrenamiento y "art-attack" es el que, usando los datos generados por "gen-data", transforma una imágen en una secuencia de caracteres ASCII. El script "run-ascii-art.sh" genera, si no existen, los datos con las imágenes de prueba, transforma la imágen en escala de grises y al tamaño adecuado, y llama a "art-attack" para generar la imágen ASCII.