Práctica 3: Reconocimiento de caras utilizando el Análisis de las Componentes Principales (PCA)

El problema de reconocer a las personas en función de su cara es realmente un problema de clasificación. Para resolver estos problemas debemos realizar un proceso de aprendizaje que permita obtener una función, a partir de los ejemplos de entrenamiento disponibles (de los que conocemos su clase), que nos permita clasificar nuevos ejemplos que debamos clasificar en el futuro.

En nuestro caso, tenemos tantas clases como estudiantes y tantos ejemplos de entrenamiento como fotos (cogeremos 3 fotos de cada alumno y la restante la utilizaremos para comprobar si el clasificador funciona). Como es evidente conocemos la clase (estudiante) de cada ejemplo (foto). Además, cada foto tiene tantos atributos como píxeles. Es decir, tenemos muchos más atributos que clases.

Existen muchos algoritmos para resolver problemas de clasificación. En esta práctica vamos a utilizar uno de los más sencillos puesto que en la fase de aprendizaje solo tenemos que crear prototipos y en la fase de clasificación solo tenemos que calcular distancias:

1. Obtener el prototipo de cada clase (estudiante): la forma más fácil de obtener el prototipo de una clase es coger todos sus ejemplos (fotos) y calcular la media para cada variable. Es decir, obtendremos una nueva imagen que será la imagen media de todas las de esa clase.
2. Para clasificar un ejemplo tenemos que calcular la distancia entre el ejemplo y cada uno de los prototipos. Asignaremos la clase correspondiente al prototipo con el que la distancia sea menor, es decir, el más similar al ejemplo a clasificar.

La Figura 1 muestra este proceso. Este es un problema de 3 clases (c1, c2 y c3) y 2 variables (u1 y u2). Disponemos de 11 ejemplos de entrenamiento, 3 de la clase c2 y 4 de las clases c1 y c3 que están enmarcados dentro de los tres círculos grises. Los prototipos creados para cada clase están representados mediante las cruces rojas. El punto verde es un nuevo ejemplo a clasificar para lo que calcularíamos la distancia entre este punto y cada prototipo. Es comprensible, que dada la ubicación del nuevo ejemplo, será asignado a la clase 2 puesto que su prototipo es el más cercano (el más similar).

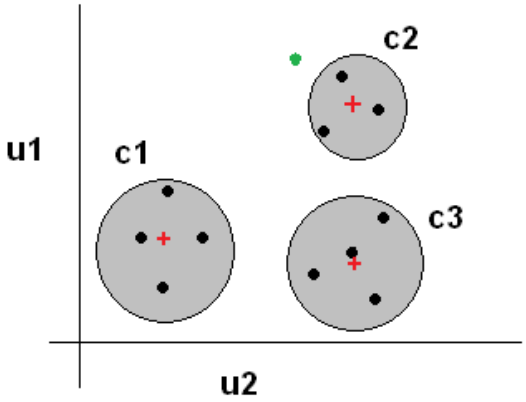

Figura 1. Proceso de clasificación tras aplicar el algoritmo PCA, que habrá aprendido el mecanismo para proyectar los ejemplos en las nuevas variables u1 y u2.

Este es un problema fácil puesto que tenemos muchas más clases (3) que atributos (2) y además no existe solapamiento entre los ejemplos de las diferentes clases (los ejemplos de las diferentes clases no se mezclan entre ellos). Por este motivo los prototipos creados están alejados entre ellos y no existen dudas.

Sin embargo, si tuviéramos muchos más atributos que clases y además los ejemplos tuvieran problemas como ruido, atributos redundantes (nos ofrecen la misma información), etc… los prototipos creados podrían no representar bien las características (distribución) del problema.

Para tratar de afrontar este problema se suele utilizar el análisis de las componentes principales (PCA en inglés) puesto que transforma los datos originales en unos nuevos datos que permiten reflejar mejor las características del problema de forma que facilita la posterior clasificación. Además, el PCA también ofrece la posibilidad de reducir la información guardada para afrontar el problema. Para realizar esta transformación lo que realiza el PCA es un cambio de base (pasa de la base canónica a otra) y obtiene las coordenadas de los datos originales en la nueva base. Estas nuevas coordenadas serán los nuevos atributos del problema.

Bases e imágenes

Si hablamos de un espacio definido por 2 atributos (x e y) somos capaces de visualizar fácilmente la situación de cualquier ejemplo que tengamos. Por ejemplo, si nos llega el ejemplo (3, 2) lo visualizamos como se muestra en la Figura 2.

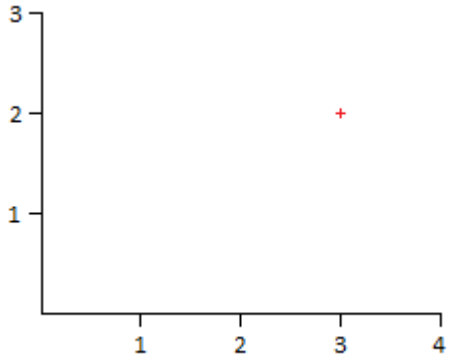

Figura 2 Representación del punto (3,2) utilizando la base canónica

Esto es debido a que estamos utilizando la base canónica compuesta por los vectores:

- (1,0) para la componente x.
- (0, 1) para la componente y.

De esta forma el punto (3,2) es el resultado de hacer la siguiente operación: 3*(1, 0) + 2*(0, 1).

En nuestro problema, cada ejemplo (foto) tiene tantos atributos como píxeles. Es decir, como las imágenes son de 120x160, van a tener 120*160 = 19200 atributos. Si pensamos en una imagen como un vector de 19200 elementos (transformamos la foto de matriz a vector) nos damos cuenta que cada píxel sería una coordenada que multiplicaría a su vector correspondiente. Es decir:

- Píxel 1 * (1, 0, …, 0): tendremos 19199 ceros después del 1.
- Píxel 2 * (0, 1, 0, …, 0): tendremos 19198 ceros después del 1.

- Píxel 3 * (0, 0, 1, 0, …, 0): tendremos 19197 ceros después del 1.
- …
- Píxel 19200 * (0, …, 0, 1): tendremos 19199 ceros antes del 1.

En resumen, la base canónica en este caso está compuesta por 19200 vectores de 19200 elementos.

Tanto en el caso del ejemplo de 2 atributos como en el caso de las fotos podemos representar cualquier ejemplo (sus coordenadas) en otra base diferente de la canónica.

El algoritmo PCA nos provee un procedimiento para aprender una nueva base en función de los ejemplos de entrada de tal forma que se magnifiquen las diferencias entre los mismos. Como consecuencia, las coordenadas de los ejemplos en la nueva base reflejaran mejor las diferencias entre ellos y facilitarán el proceso de clasificación.

Implementación

En esta práctica vamos a implementar el algoritmo PCA para afrontar el problema del reconocimiento de caras. Vamos a dividir este proceso en varias etapas.

Etapa 1: Aprendizaje de la nueva base

Esta etapa está compuesta por 6 pasos:

1. Crear el conjunto de datos inicial. Para ello se deben leer las 3 primeras fotos (1, 2 y 3) de cada alumno, transformarlas de una matriz a un vector columna (imagen(:)) y almacenar cada foto como una columna de una matriz llamada ***R***. Las dimensiones de R son NumeroPixeles x NumeroFotos.
2. Calcular la media por filas de la matriz R y almacenarla en una variable llamada ***media***. Es decir, un vector columna en el que cada elemento i será la media de los elementos de R en la fila i.
3. Calcular los vectores con media nula. Es decir, a cada foto (columna de R) se le debe restar el vector media calculado en el paso anterior. Almacenar cada vector obtenido en la resta en una columna de una matriz llamada ***A***. Las dimensiones de A son iguales a las de R.
4. Calcular la matriz de correlación ***C***. Para ello hay que multiplicar la matriz A transpuesta por la matriz A. La matriz C es una matriz cuadrada compuesta por NumeroFotos x NumeroFotos elementos.

   NOTA: este punto es debido a que vamos a aplicar la transforamción de Karhunen-Loève.

5. Calcular los NumeroFotos valores y vectores propios de C. Utlizar la función ***[v,μ] = eig(C)***. Las variables ***v*** y ***μ*** son matrices cuadradas del mismo tamaño de C. En la primera columna de v tendremos el primer vector propio de la matriz, que estará asociado al valor propio almacenado en la posición (1,1) de la matriz μ. En la segunda columna de v tendremos el segundo vector propio de la matriz, que estará asociado al valor propio almacenado en la posición (2,2) de la matriz μ y así sucesivamente.

NOTA: estos vectores propios tendrán NumeroFotos componentes. Sin embargo, necesitamos que tengan NumeroPixeles componentes. Para ello hay que aplicar el siguiente paso.

6. Para cada vector propio, v_i (y su correspondiente valor propio, μ_i) vamos a obtener su correspondiente vector propio, u_i, de NumeroPixeles elementos. Para ello, hay que aplicar:

$$u_i = \frac{1}{\sqrt{\mu_i}} * A * v_i$$

Guardar cada vector como una columna de una variable llamada ***nuevaBase***, cuyas dimensiones son NumeroPixeles x NumeroFotos.

NOTA 1: tras este paso obtenemos la nueva base, que estará formada por NumeroFotos vectores de NumeroPixeles componentes.

NOTA 2: Por tanto, las coordenadas de los ejemplos utilizando la nueva base tendrán NumeroFotos componentes en lugar de NumeroPixeles componentes.

Implementa una función llamada ***aprendeBase.m*** sin parámetros de entrada que devuelva las variables ***media***, ***A*** y ***nuevaBase***.

Etapa 2: Creación de los prototipos

Una vez que hemos calculado la nueva base podemos generar los prototipos para cada clase (estudiante). Para ello debemos aplicar dos pasos: proyectar los ejemplos originales en la nueva base y crear los prototipos de cada alumno.

- Proyección de los ejemplos en la nueva base. En este paso vamos a calcular las NumeroFotos coordenadas de cada ejemplo i (foto) para representarlos en la nueva base. Para calcular cada coordenada j, aplicamos la siguiente ecuación:

$$w_{ji} = nuevaBase_j{}' * (imagen_i - media)$$

NOTA: daos cuenta que las imágenes menos la media las tenemos guardadas en la matriz A. Por tanto, podemos aplicar

$$w_{ji} = nuevaBase_j{}' * A_i$$

- Con las 3 fotos de cada estudiante proyectadas en la nueva base hay que hacer su prototipo. Para ello simplemente calculamos la media de los 3 vectores w obtenidos en el paso anterior (obviamente correspondientes al mismo estudiante). Guardar cada prototipo como una columna de una matriz llamada ***prototipos***, cuyas dimensiones son NumeroFotos x NumeroAlumnos.

Implementa una función llamada ***creaPrototipos.m*** que reciba como argumentos de entrada las variables ***nuevaBase***, ***media*** y ***A*** y devuelva como parámetro de salida la variable ***prototipos***.

Etapa 3: Clasificación de nueva fotos

En esta última etapa debemos clasificar las fotos de cada alumno que no hemos utilizado para aprender la nueva base y para generar los prototipos. De esta forma comprobaremos si el aprendizaje realizado es bueno o no. Para clasificar la foto de cada alumno debemos aplicar los siguientes 4 pasos:

1. Leer la última imagen (4) de cada alumno y transformarla de una matriz a un vector.
2. Proyectar la foto en la nueva base. Para calcular cada coordenada j, aplicamos la siguiente ecuación:

$$w_j = nuevaBase_j{}' * (imagen - media)$$

3. Calcular la distancia entre el vector w obtenido en el paso anterior y cada uno de los prototipos. La distancia se calcula como

$$distancia_i = \sum_{j=1}^{NumeroFotos} \left(w_j - prototipos_{ji}\right)^2$$

4. Calcular el mínimo de las distancias y obtener su correspondiente prototipo (índice dentro del vector de distancias). Si el prototipo corresponde al estudiante de la foto (valor de la variable que realiza el bucle for) hemos acertado.

Para completar este apartado podemos calcular el porcentaje de aciertos. Para ello, incluiremos un contador de aciertos que incrementaremos cuando el algoritmo acierte. Al acabar con todos los estudiantes dividiremos el contador entre el número de estudiantes y lo multiplicaremos por 100 (llamar a esta variable ***porcentajeAciertos***). Además, en caso de error podemos mostrar por pantalla el estudiante que realmente es y el que ha asignado el programa (sus identificadores: índices).

Crea una función llamada ***clasificar.m*** que reciba como parámetros de entrada las variables ***nuevaBase***, ***media*** y ***prototipos*** y devuelva como salida la variable ***porcentajeAciertos***.

Crea una función main.m que realice las llamadas a las 3 funciones anteriores.