MEMÓRIA DE PRÁCTICAS: FISHERFACES

Biometría – Master en IARFID

Enric Bonet Cortés
Universidad Politécnica de Valencia

Implementar FisherFaces:

- Probar con ORL:
  - Training: imágenes [1 – 5] de cada individuo
  - Test: imágenes [6 – 10] de cada individuo
- Utilizar el vecino más cercano
- Obtener curvas de error variando d'

Tal y como se describe en el enunciado, para este problema recomienda utilizar el mismo dataset y separación de corpus que el anterior. Así pues, se ha utilizado la misma implementación anterior sobre un Notebook de *Google Collaboratory*, utilizando de nuevo las funciones **get_images** para cargar las imágenes en los corpus y **readpgm** para leerlas.

Pasando a la resolución del problema, se ha empleado la misma conversión a la estructura de datos Matrix de la librería **Numpy** de las imágenes de los corpus para así, poder reutilizar la implementación de la reducción PCA del ejercicio anterior, puesto que FisherFaces cuenta básicamente con dos métodos de reducción: PCA y LDA.

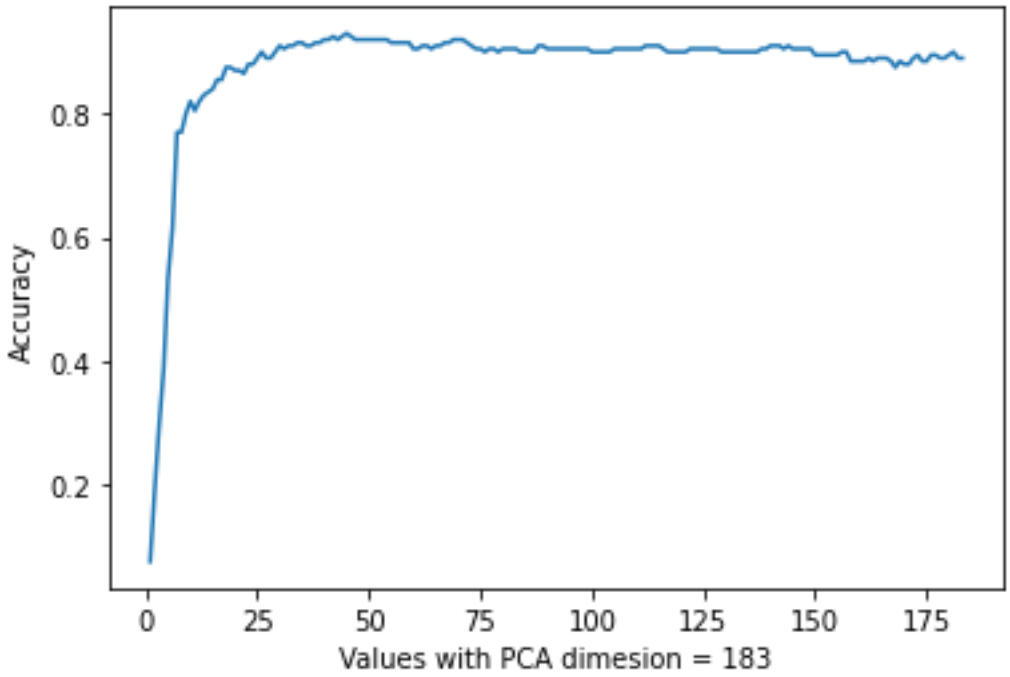
Teniendo en cuenta pues la anterior implementación de PCA, se ha implementado una reducción LDA a través de 2 funciones más: **LDA** y **reduce_LDA**.

La función **LDA** se encargará de devolver los EigenVectores calculados a partir de los datos ya reducidos por el método PCA. Al igual que en el ejercicio anterior, se aprovechan los diferentes atributos de la estructura Matrix de **Numpy** para realizar el cálculo de la matriz de covarianzas y se calculan los EigenVectores a partir de la función **eigh** de **Numpy.lianlg**. En este caso, estos si que son los EigenVectores que se deben de devolver por el método, por lo que no hay que tener ninguna consideración adicional, más allá de ordenarlos según EigenValues (de igual forma que la función PCA) antes de devolverlos.

Para la función **reduce_LDA**, se le tienen que pasar como parámetros: la matriz de datos ya reducida por **PCA**, los EigenVectores calculados y ordenados por **LDA**, y el número de componentes o, mejor dicho, la dimensión a la que se quieren volver a reducir los datos (d'). Con toda esta información, la función reduce la dimensionalidad de los datos de forma similar a la función **reduce** del ejercicio anterior (**reduce_PCA** en este ejercicio), con la única diferencia de que esta vez no se les resta a los datos su propia media antes de multiplicarse por los EigenVectores "reducidos".

Al igual que en el problema anterior, se ha construido bucle que pruebas, que esta vez consta de dos niveles, para probar los diferentes valores obtenidos por el clasificado 1-NN de **Sklearn**. En el primer nivel del bucle, se reducen (para cada valor posible de 2 a 200) los datos mediante PCA, mientras en el segundo nivel del bucle, se reducen por LDA los datos para cada valor posible de d' entre 1 y la d' prima utilizada por PCA, para finalmente utilizar el clasificador de "Vecino Más Cercano" y calcular su *score*.

Para este experimento, el valor máximo obtenido es del 93% de precisión (claramente superior al del anterior ejercicio), y se ha obtenido aplicando una dimensionalidad reducida (d') de 183 en PCA y de 45 en LDA.



En la gráfica anterior, se puede observar cuál es la evolución de la precisión utilizando LDA tras haber reducido una primera vez la dimensionalidad a d' = 183 con PCA.