Reconocimiento Facial mediante los métodos PCA y KPCA

Autores:
Damm, Eugenio - Legajo: 57094
Calatayud, Agustín - Legajo: 57325

1. Resumen

El siguiente trabajo describe las metodologías utilizadas para el cálculo de autovectores y autovalores así como su integración en los métodos de análisis de componentes principales PCA y KPCA. Estas técnicas fueron utilizadas para reconocimiento facial a fin de reducir la dimensión del problema en particular. Cabe destacar que dado el gran volumen de datos requerido para una base de datos viable para reconocimiento facial, los métodos de análisis de componentes principales son fundamentales para una implementación eficiente.

2. Palabras Clave

- PCA: Análisis de componentes principales
- KPCA: Análisis de componentes principales usando Kernel

3. Introducción

En los últimos años, la necesidad de mejorar la seguridad en diferentes contextos de la sociedad actual, dieron lugar al desarrollo de nuevo hardware y software informático empleando técnicas de reconocimiento automático basadas en rasgos biométricos. Los sistemas biométricos son sistemas automatizados que permiten la identificación y verificación de individuos de forma rápida y segura, puesto que solo se analizan determinados patrones biométricos que no pueden ser alterados, manipulados, falsificados o robados para recrear información personal o acceder a la información de otros.

Frente a este avance tecnológico, nos pareció sumamente interesante trabajar con ésta temática. Sumando los conocimientos adquiridos a lo largo de la cursada con otras investigaciones presentes en internet, pudimos desarrollar nuestro propio programa que lleve a cabo ciertos métodos para el reconocimiento facial. Los métodos implementados para resolver la problemática fueron KPCA y PCA. Ambos requieren en algún momento u otro el cálculo de autovectores y autovalores, para esto fueron implementados métodos de descomposición QR para su iteración y búsqueda. A su vez, se implementó un método para obtener los right singular vectors.

En este informe mostraremos cómo funciona nuestra implementación en la sección Metodología, y a su vez, en la sección de Resultados daremos evidencia de la eficiencia de la misma mediante gráficos y datos numéricos.

4. Metodología

4.1 Descomposición QR

Para la descomposición QR de matrices se implementaron dos métodos, el primero el método de Gram Schmidt modificado y luego la descomposición por medio del método de Householder. Por investigación realizada, con el primer método la matriz Q tiene un error de ortogonalización superior al del segundo método.

Mientras que Householder acarrea un error similar al epsilon de máquina, el procedimiento de Gram Schmidt modificado trae un error considerablemente mayor por lo que este método no resultará en una matriz Q ortogonal. Además de esto, se pudo notar una diferencia notable en el tiempo de ejecución. Es por esto que se utilizó el método de Householder presentado a continuación:

```python
def householderQR(A):
    m,n = A.shape
    R = np.matrix(A)
    Q = np.identity(m)

    for k in range(n):
        norm = np.linalg.norm(R[k:,k])
        s = -np.sign(R[k,k])
        u = R[k,k] - s * norm
        w = R[k:,k]/u
        w[0] = 1
        tau = -s * u/norm
        R[k:, :] = R[k:, :] - (tau*w) * w.T.dot(R[k:, :])
        Q[:, k:] = Q[:, k:] - (Q[:,k:] * w).dot((tau * w).T)
    return Q, R
```

Código 1: Método de Householder

4.2 Obtención de autovalores y autovectores

Para la obtención de autovalores y autovectores se utilizó el método de iteración de QR mediante el cual se converge a una matriz triangular superior, o diagonal en el caso de que la matriz original fuese simétrica. Mediante este método se consiguen los autovalores y los autovectores, siendo los primeros, los valores de la diagonal de la matriz triangular superior y los segundos siendo las columnas de la multiplicación sucesiva de las Q obtenidas. Se puede ver el codigo a continuacion:

```python
def iterateQR(A):

    maxIterations = 100
    eigenvectors = np.identity(A.shape[0])
    T = A

    for i in range(maxIterations):
        Q, R = householderQR(T)
        T = R.dot(Q)
        eigenvectors = eigenvectors.dot(Q)
        if np.allclose(T, np.triu(T), atol = 1e-4):
            break

    eigenvalues = np.diag(T)

    sort = np.argsort(np.absolute(eigenvalues))[::-1]
    return eigenvalues[sort], eigenvectors[sort]
```

Código 2: Método iterativo de QR

4.3 Obtención de right singular vectors

Dado el gran tamaño de las matrices con las que se trabaja se decidió conseguir los right singular vectors a partir de la siguiente ecuación:

$$A^T x\, u = \sigma\, x\, v \quad (1)$$

donde A es la matriz de las imágenes, u es un left singular vector, σ su respectivo valor singular y v es el right singular vector. De esta forma se computa:

$$A\, x\, A^T \quad (2)$$

de manera tal que se reduzca el tamaño de la matriz de la cual se obtienen los autovalores y autovectores. Con estos autovalores, se computan los valores singulares, y con los autovectores se tienen los left singular vectors para la obtención de los right singular vectors, los cuales fueron necesarios para la resolución del problema de PCA.

4.4 PCA y KPCA

Para la implementación de PCA primero se restó la imagen media para centrar los datos y luego se obtuvieron los right singular vectors de la matriz de imagenes centrada. Luego se proyectaron las imágenes sobre una porción de los right singular vectors (autocaras), analizando así el error proveniente de utilizar una distinta cantidad de autocaras para el entrenamiento del sistema de categorías.

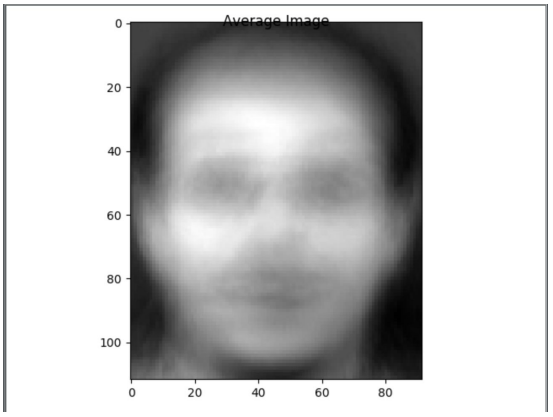

Figura 1: Ejemplo de Imagen Media

La idea de esta metodología es utilizar la mínima cantidad de datos para entrenar el sistema y que produzca resultados favorables.

Para la implementación de KPCA se utilizó un kernel de grado 2 en para poder realizar una separación no lineal de las componentes principales. Se centraron las imágenes mediante la siguiente transformación:

$$K = K - U/m\, x\, K - K\, x\, U/m + U/m\, x\, K\, x\, U/m \quad (3)$$

donde K es el kernel, U es una matriz de mxm de unos y m es la cantidad de imágenes.

Luego con los autovectores del kernel normalizados por la raíz de los autovalores correspondientes se realiza la proyección. Una vez proyectado se realizó el mismo procedimiento que con PCA para obtener el error del sistema.

5. Resultados

Al utilizar nuestras implementaciones de tanto de PCA como de KPCA, pudimos observar resultados como los siguientes:

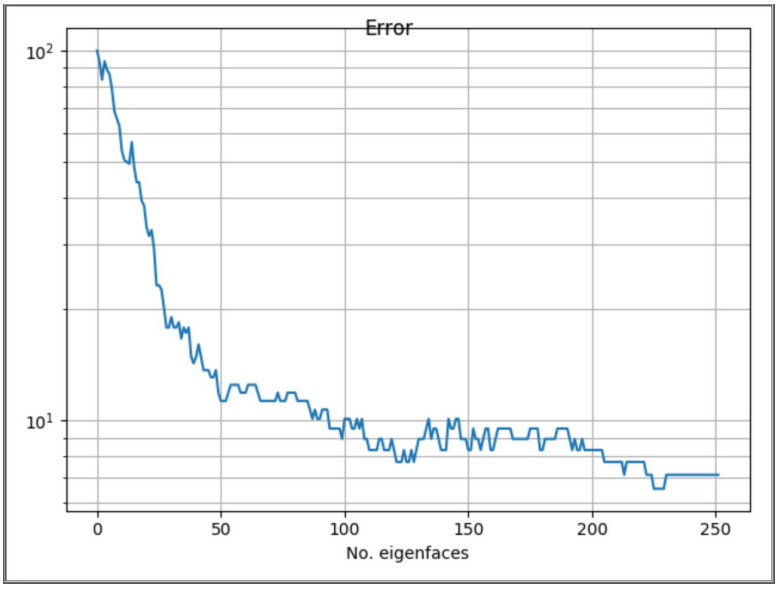

Figura 2: Error obtenido mediante KPCA

En la Figura 2, observamos un gráfico del error cometido en la clasificación en función de la cantidad de autocaras utilizadas por medio del método del kernel. A simple vista podemos observar cómo frente al incremento en la cantidad de autocaras, progresivamente el error se hace cada vez menor. Esta mejora en el error se estabiliza a medida que el total de autocaras aumenta.

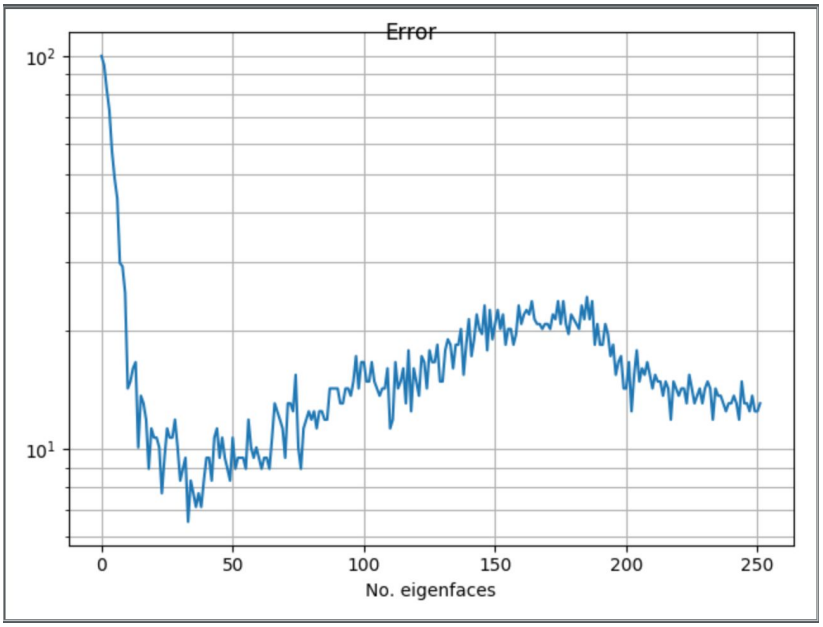

Figura 3: Error obtenido mediante PCA

En la Figura 3, a diferencia de la Figura 2, utilizamos el método de análisis PCA sin kernel. Si bien el gráfico muestra la misma relación entre el error cometido en la clasificación en función de la cantidad de autocaras utilizadas, la curva en sí es muy distinta. Previamente, comenzaba el análisis con un error de gran magnitud, decreciendo conforme aumentaba el número de autocaras de manera progresiva, mientras que en éste caso, si bien llegamos a un pico mínimo en una instancia temprana, lentamente el error aumenta, dada la mayor cantidad de autocaras.

6. Conclusiones

De los resultados se pueden extraer algunas conclusiones, el método de KPCA resulta más estable que el de PCA, al estar mapeando a un espacio mayor puede resolver las direcciones de mayor varianza que no son lineales mientras mantiene las operaciones realizadas en el espacio de origen.

Además se puede ver que a medida que se extraen más autocaras, el método de KPCA obtiene mejores resultados mientras que el PCA obtiene buenos resultados con menos autocaras pero luego su error aumenta. En este sentido si se quisiera obtener, al menos para este set de datos, una buena representación con pocas autocaras se podría utilizar PCA, pero como dicho previamente el error no se estabiliza con la cantidad de autocaras en aumento. Dado que KPCA mantiene un error estable es una buena elección de método para la obtención de las direcciones de mayor varianza si se quieren usar todas las autocaras.

En otro set de datos podría obtenerse una dirección de máxima varianza de forma tal que fuera imposible hacerlo con PCA. En ese sentido KPCA es superior por poder lograr unos resultados mejores sin la necesidad de aumentar la dimensión del espacio de origen de manera explícita.

7. Bibliografía

- Descomposicíon QR - Referencias de la cátedra.
- Householder QR - http://www.cs.cornell.edu/~bindel/class/cs6210-f09/lec18.pdf
- SVD - http://math.mit.edu/classes/18.095/2016IAP/lec2/SVD_Notes.pdf
- PCA y KPCA - Referencias de la cátedra.
- Código provisto por la cátedra.