Métodos Numéricos Avanzados

Trabajo Práctico Especial 1

Segundo Cuatrimestre de 2020

Grupo L

Integrantes

- Donath, Holger 58110
- Karpovich, Lucia 58131
- Rolandeli, Alejandro 56644
- Reyes, Santiago 58148
- Tarradellas, Manuel 58091
- Vedoya, Pedro 57605

Introducción

Este trabajo práctico consiste en desarrollar un sistema de reconocimiento facial de fotos a partir de una base de datos. Para el preprocesamiento de los datos se utilizaron dos algoritmos: PCA y KPCA. También se implementó un algoritmo para encontrar autovalores y autovectores. Para finalizar, se utilizó SVM para la clasificación de los resultados y un algoritmo propio para sacar un porcentaje similitud a la persona que se reconoció.

Funcionamiento

El sistema obtiene todos sus parámetros de un archivo llamado *configuration.ini*.

Las fotos que el programa utiliza como base de datos para el "entrenamiento" se transforman en una matriz donde cada fila representa una foto. Luego se transforma la foto a analizar en un vector. Dicha foto se puede obtener de dos modos: pasando un path para la foto a analizar (en el archivo provisto arriba), o correr el sistema en modo video el cual accede a la webcam de la computadora que permite sacar la foto en tiempo real.

A continuación, la foto a analizar (a la cual se llamó anónimo) y la matriz de fotos de entrenamiento pasan al procesamiento. Se puede optar entre los dos métodos posibles (PCA y KPCA) en el archivo de configuración. Ambos harán uso del mismo algoritmo para la obtención de autovalores y autovectores.

Dado que el análisis de los datos procesados se hará con una proyección, es decir, una parte de los datos, el sistema permite seleccionar la cantidad de vectores a analizar.

Para la clasificación de resultados se utiliza SVM (Support Vector Machines) de la librería *sklearn*. Este algoritmo, dado un conjunto de ejemplos de entrenamiento (en nuestro caso la base de datos de imágenes) permite etiquetarlos en clases (en este caso, en las personas de la base de datos) y entrenarse para predecir la clase de una nueva muestra (la foto anónima) y este es el resultado que se entrega junto con un procedimiento propio adicional que se explica en la siguiente sección.

Metodología

PCA

El algoritmo recibe una matriz con todas las imágenes como filas, y calcula la imagen media y dispersión de ellas. Luego procede a estandarizar las imágenes en la matriz inicial ((imagen - media) / dispersión). A partir de esta matriz estandarizada (llamémosla A), se calcula una matriz C reducida haciendo $C = (\frac{1}{n-1}) * A * A^t$, y calculamos los autovectores y autovalores con nuestro método, el cual explicamos más abajo. Luego filtramos los autovectores que corresponden a autovalores grandes (y así nos deshacemos de autovectores de autovalores que son 0), y estos autovectores los multiplicamos por la matriz de imágenes estandarizadas transpuesta. Luego normalizamos estas, y procedemos a generar una matriz de pesos utilizando estos autovectores como matriz de cambio de bases (explicado más abajo), y devuelve esta matriz. El algoritmo también calcula la matriz de pesos correspondiente a la imagen que queremos averiguar quién es.

Cambio de bases

El algoritmo de cambio de bases busca obtener una matriz de pesos a partir de la matriz de imágenes, su media y la base de autovectores. Partiendo de la matriz de imágenes M, queremos llegar a $M_i = W_1V_1 + W_2V_2 + ... + media$, para cada fila i de M (que representa una imagen). Para esto planteamos $W = V^{\dagger} * (M' - media)$ y obtenemos la matriz de cambio de base.

KPCA

Kernel PCA es una extensión del algoritmo de PCA para los casos donde la muestra no es linealmente separable en la dimensión d, para lo cual se amplía la dimensión (a N) de los datos a través de una función que se suele llamar Φ.

$$\Phi : \mathbb{R}^d \to \mathbb{R}^N$$

La idea es generar una matriz llamada K donde cada una de sus celdas está dada por una función. Pero el truco de kernel nos permite generarla sin calcular la función Φ explícitamente, lo cual permite la posibilidad de usar data en altas dimensiones, pero no tener que evaluarla en ese espacio.

$$K = k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}), \Phi(\mathbf{y})) = \Phi(\mathbf{x})^T \Phi(\mathbf{y})$$

Hay varias funciones de k conocidas, nosotros tomamos la polinomial con grado regulable por configuración, pero en principio de 2.

A esta matriz K se le aplica la siguiente función para centralizar los datos:

$$K' = K - \mathbf{1_N}K - K\mathbf{1_N} + \mathbf{1_N}K\mathbf{1_N}$$

donde $\mathbf{1_N}$ denota una matriz de NxN donde cada casilla tiene el valor 1/N.

Luego se obtienen los autovectores y autovalores de este nuevo K' y se los normaliza (dividiendo a cada autovector por su correspondiente autovalor. A partir de estos valores se genera una proyección de las imágenes de base de datos y una de la imagen anónima.

Autovalores y autovectores

Se calcularon a partir de la descomposición QR Householder de manera iterativa de la matriz original.

- Householder busca encontrar una descomposición a partir de una matriz A, de manera iterativa, donde en cada iteración se saca un Qi de tal modo que la expresión final se escribe como $Q_n * Q_{n-1} * ... * Q_2 * Q_1 * A = R$, donde $Q_n * Q_{n-1} * ... * Q_2 * Q_1 = Q^T$. Para hacer esto, se parte de un R = A, y en cada iteración i se busca que un Qi, multiplicado por la sucesión ya existente de Qi-1*A, convierta todos los valores debajo de R[i,i] en 0.
- Una vez que se tienen los valores QR, se itera partiendo de unos Q y R iniciales de A, donde se guardan como autovectores = Q, X = R*Q y autovalores = diag(X). Y entonces procede a repetirse ese procedimiento, donde se calcula iterativamente la descomposición QR a partir de X, que luego X se reemplaza con los nuevos Q y R, los autovalores son reemplazados por la diagonal de este X, y los autovectores se acumulan de manera autovectores = autovalores * Q. Una vez que termina de iterar devuelve los autovalores ya autovectores.

Porcentaje de similitud y confianza

Luego de calcular la matriz y vector de las fotos en las bases de PCA o KPCA, utilizamos SVM para obtener a qué persona de la matriz pertenece el vector de la imagen. Generamos el SVM de manera que se puede seleccionar la cantidad de iteraciones y el valor de regularización C a utilizar. Luego de varias pruebas concluimos que los más precisos eran los default de 1000 iteraciones, C = 1,0. Los warnings de falta de convergencia fueron solucionados al estandarizar las matrices antes de entregarlas al SVM. Una vez generados los grupos de clasificación (personas), obtenemos a que grupo/persona pertenece la imagen anónima y calculamos el nivel de confianza de este resultado con el procedimiento explicado a continuación.

Se calcula el promedio de los vectores de cada grupo/persona, para obtener un vector que sea el representativo de ese grupo. Llamémoslo: ***Vn*** con ***n*** siendo el grupo.

Luego se toma el vector de pesos de la imagen que se está evaluando, llamado previamente como el "anónimo", que se comparará con cada uno de los ***Vn*** para obtener un valor de similitud entre la persona "anónima" que se está evaluando y cada uno de los grupos (es decir cada una de las personas en el dataset original). Llamémoslo a este: ***a***

Para calcular la similitud hay que tener en cuenta estos dos conceptos principales:

- La desigualdad de Cauchy-Schwartz.
- El producto interno de dos vectores da 0 cuando son ortogonales.

La desigualdad puede escribirse como $0 \leq |< V_1, V_2 >| \leq norm_2(V_1) * norm_2(V_2)$.

Si el producto de las normas es distinto a 0, se puede pasar dividiendo y queda la siguiente expresión: $S = \frac{|<V_1,V_2>|}{norm_2(V_1)*norm_2(V_2)}$, donde S pertenece al rango [0, 1].

Reemplazando ***V1*** y ***V2*** por los ***a*** y ***Vn*** mencionados previamente, y teniendo en cuenta que dos vectores son ortogonales cuando su producto interno da 0, se puede concluir que si el valor de ***S*** da 0 entonces los vectores son lo "más diferente posible" por lo tanto la persona anónima es muy diferente a la persona correspondiente al vector ***Vn***. Análogamente, si el valor de ***S*** da 1 quiere decir que ***a*** y ***Vn*** son iguales, por lo tanto, la persona anónima ***a*** es la misma que la persona correspondiente al vector ***Vn***.

Entonces, para calcular el porcentaje de similitud tomamos como ***n*** al grupo encontrado utilizando el SVM y calculamos su similitud ***S*** como fue mencionado previamente.

Luego la confianza se devuelve como uno de 5 valores: '**Very high**', '**high**', '**moderate**', '**low**', '**very low**'. Esas categorías se definen en base al valor de similitud donde una imagen muy parecida a alguna de las personas que se encuentran en el set devolverá '**Very high'** y una muy diferente devolverá '**very low'**

Los rangos de similitud que definen cada categoría varían depende si se utilizó PCA o KPCA y fueron definidas por nosotros empíricamente luego de hacer pruebas con varias imágenes dentro y fuera del set.

Resultados

Dividimos los casos entres teniendo en cuenta quien era la persona en la foto a analizar, donde cada resultado es un porcentaje.

Personas fuera de la base de datos

KPCA degree 3: 17.5; 18.4; 24,6; 17,6; 29; 24; 20; 29; 16
KPCA degree 2: 39; 39; 36; 21; 20; 23; 32; 37; 22
PCA: 53.07; 67.6; 56.09; 67.37; 68; 53; 21; 68

Personas en la base de datos (pero fotos que no lo están)

KPCA degree 3: 20.05; 34.69; 29.29; 27.24
KPCA degree 2: 39.10; 29.12; 44.89
PCA: 84.64; 90.11; 74.58; 79.04; 86

Personas en la base de datos (fotos sacadas de la base de datos)

KPCA degree 3: 49.59, 27.29, 46.66, 40.66, 40.93
KPCA degree 2: 44.44,49.07, 34.60,57.01
PCA: 88.72, 97.53, 88.72, 92.36

Mejoras futuras

- Notamos que el método que calcula los autovectores y autovalores tarda mucho más que el método utilizado en la librería de numpy, por lo que notamos que podría mejorarse el algoritmo en sí, o buscar una mejor manera de encontrar autovectores y autovalores (como otro algoritmo más paralelizable). Si bien implementamos el método de las potencias y el método de Rayleigh para conseguir los autovectores, nunca hicimos testeos profundos con ellos (sin embargo, la implementación se encuentra en el código fuente).
- El programa podría tener un mejor manejo de los errores y podría tener más feedback hacia el usuario sobre lo que pudo haber fallado, o sobre las acciones que se están ejecutando en el programa, para poder seguir el procesamiento de este.
- Mejorar KPCA dado que el porcentaje de confianza es significativamente más bajo que en PCA.
- Hacer un mayor análisis sobre el funcionamiento de SVM para mejorar sus resultados, como por ejemplo intentar el Kernel Trick o alguna otra función.
- Agregar más emojis.

Conclusiones

Se puede observar a partir de los resultados, que el nivel de confianza promedio de KPCA es en general más bajo que en PCA acorde al cálculo de similitud aplicado. Esto podría deberse a que el cálculo que desarrollamos en base al promedio de fotos de la persona no es tan estable para las transformaciones que aplica KPCA. Además, si sólo observamos los resultados de SVM, la diferencia de falsos positivos entre KPCA y PCA no es tan distinta como los niveles de confianza de cada uno.

El cálculo de los autovectores y autovalores es más lento de lo que podría ser, comparado con la librería de numpy. Se puede ver también que la selección del SVM es, en la gran mayoría de los casos, precisa, pero tiene amplio margen para mejorar.