DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2

Reconocimiento de caras

26 de mayo de 2018 — Métodos Numéricos

Grupo

Integrante	LU	Correo electrónico
Buceta, Diego	001/17	diegobuceta35@gmail.com
Jiménez, Gabriel	407/17	gabrielnezzg@gmail.com
Salvador, Alejo	467/15	alejo.antonio.salvador@gmail.com
Springhart, Gonzalo	308/17	glspringhart@gmail.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

En este trabajo práctico vamos a implementar un programa que dada la foto de una persona identifique a quien pertenece usando como referencia una base de datos de entrenamiento que guardará junto a la imagen a quién pertenece. Para poder realizarlo, se transformará la imagen a clasificar y las de la base de dato de entrenamiento usando PCA, lo cual logrará reducir la cantidad de componentes de las imágenes de modo tal que los que queden sean los más relevantes. Luego con estas imágenes preprocesadas se encontrarán los k vecinos mas cercanos en la base de datos de entrenamiento, para determinar cual es clasificación mayoritaria entre dichos vecinos. Finalmente para calcular la eficiencia del algoritmo se hará uso de "K-fold cross validation".

Palabras clave: *PCA*, *K vecinos mas cercanos*, *K Fold cross validation*, *Método de la potencia*

1. Introducción Teórica

El objetivo del presente informe es resolver un problema práctico mediante el modelado matemático del mismo. Este problema consiste en poder clasificar imágenes según su parecido a las imágenes de una base de datos de entrenamiento.

Para lograr este objetivo se aplicará el algoritmo de k vecinos mas cercanos (KNN por sus siglas en inglés). Este algoritmo es uno de los algoritmos mas simples de aprendizaje automático pero presenta el defecto de ser muy sensible a las dimensiones de la imagen (maldición de la dimensionalidad) y la intensidad de la misma entre otras características. Por esa razón será conveniente mitigar estos problemas preprocesando las imágenes mediante análisis de componentes principales (PCA según sus siglas en ingles). Este método busca representar la mayor cantidad de información posible de la imagen con el menor número de dimensiones.

Como lo primero a efectuar es el preprocesamiento de la imagen empezaremos describiendo los fundamentos teóricos de PCA.

1.1. Análisis de Componentes Principales

Al trabajar con las imágenes tenemos un problema con sus dimensiones, si se tienen N personas, cada una con M imágenes, entonces se tienen $N \times M$ imágenes. Si cada uno de esas imágenes tiene un tamaño de 100×100 entonces el vector utilizado para representarla va ser un vector en $\mathbb{R}^{10000}$. No solamente va a resultar muy costoso trabajar en una dimensión tan alta sino que también la densidad de los datos se va a disminuir a medida que aumenta el tamaño de la dimensión, esto es lo que se conoce como maldición de la dimensionalidad.
El Análisis de Componentes Principales tiene como objetivo reducir las dimensiones del conjunto de las imágenes para poder tener un conjunto más pequeño que describa bien a los datos. Para poder realizar la reducción de dimensión lo que hacemos es aplicarle una transformación lineal que proyecte a los datos a una dimensión menor de forma tal que la dirección de la proyección sea la que tenga mayor varianza en los datos, ya que donde hay mayor varianza se puede obtener mayor información. Las dimensiones con mayor varianza se pueden encontrar con autovalores y autovectores, un autovalor grande implica que hay más varianza en la dirección del autovector, así podemos descartar las direcciones con autovalor más chico y quedarnos solamente con las que tienen información relevante.
Para lograr eso definimos una matriz X tal que las filas de esta matriz sean:

$$(x_i - \mu)^t / \sqrt{n-1}$$

Donde:

- $x - i$ es la i-ésima imagen en forma de vector
- μ es el vector promedio de las imágenes definido como: $(x_1 + ... + x_n)/n$
- n es la cantidad de imágenes

Se puede observar que en cada fila de X queda un estimador de la varianza de los datos de una imagen.

Luego definimos a la matriz $M = X^t X$, esta es la matriz de covarianza de los datos, y es a la que le vamos a obtener los autovalores y autovectores. Usamos el método de la potencia hasta obtener α (siendo esta la cantidad de dimensiones a las que queremos llevar a las imágenes) autovectores y autovalores. Si $v_1 ... v_\alpha$ son los autovectores obtenidos, entonces la transformación característica de una imagen x_i se define como $tc(x_i) = (v_1 x_i, v_2 x_i, ..., v_\alpha x_i)$ y aplicando esta transformación obtenemos las α primeras componentes principales de cada imagen.

1.2. Método de la Potencia

Como fue mencionado anteriormente para calcular los autovectores y autovalores de la matriz $M = X^tX$ utilizamos el Método de la Potencia, este método nos permite obtener buenas aproximaciones de los valores de los autovectores. El método se define de la siguiente forma:
Sea $A \in \mathbb{R}^{n\times n}$, sean $\lambda_1, ..., \lambda_n$ los autovalores de A tales que se cumple que $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_n|$ y sean $v_1, ..., v_n$ los autovectores de A tales que forman una base. Entonces el Método de la Potencia calcula el autovalor más grande y el autovector asociado al mismo y se realiza con el siguiente algoritmo.

$metodoPotencia(A,\ x_0,\ iteraciones)$
$....autovector \leftarrow x_0$
$....for\ (i = 0...iteraciones)\ do$
$.....autovector = \frac{A*autovector}{||A*autovector||}$
$....endfor$
$....autovalor \leftarrow \frac{autovector^t*A*autovector}{autovector^t*autovector}$
$....devolver \quad autovalor,\ autovector$

Esta última cuenta que se hace para calcular el autovalor se conoce como cociente de Rayleigh.

Este método si bien obtiene una aproximación muy buena del autovalor y el autovector dominante de la matriz A (en más, si *iteraciones* tiende a infinito, dan los valores exactos), no nos sirve en su forma actual para calcular **todos** los autovalores y autovectores que esta posee. Para eso vamos a utilizar este método con la deflación.

1.2.1. Deflación

Sea $A \in \mathbb{R}^{n\times n}$, sean $\lambda_1, ..., \lambda_n$ los autovalores de A tales que se cumple que $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_n|$ y sean $v_1, ..., v_n$ los autovectores de A tales que forman una base.
Defino $B_1 = A - \lambda_1 v_1 v_1^t$, entonces la matriz B tiene como autovalores a $0, \lambda_2, ..., \lambda_n$ y tiene a $v_1, ..., v_n$ como autovectores.
Demostraciones:
Demo 1. El autovalor asociado a v_1 es 0:
$Bv_1 = (A - \lambda_1 v_1 v_1^t)v_1 = Av_1 - \lambda_1 v_1 v_1^t v_1 = \lambda_1 v_1 - \lambda_1 v_1 (v_1^t v_1) = {}^1\lambda_1 v_1 - \lambda_1 v_1 = 0$
Demo 2. El autovalor asociado a algún v_i es λ_i:
$Bv_i = (A - \lambda_1 v_1 v_1^t)v_i = Av_i - \lambda_1 v_1 v_1^t v_i == \lambda_i v_i - \lambda_1 v_1 (v_1^t v_i) = {}^2\lambda_i v_i$
Entonces aplicando la deflación sucesivamente a la matriz M con los autovectores y autovalores obtenidos en el metodo de la potencia en cada iteración puedo obtener todos sus autovalores y autovectores. Ahora que PCA ya se encuentra explicado podemos pasar a proceder a describir el

[1] Como los autovectores de la base son ortogonales se cumple que $v_1^t v_1 = 1$
[2] Como los autovectores de la base son ortogonales, $v_1^t v_i = 0$ entonces $\lambda_1 v_1 (v_1^t v_i) = 0$

algoritmo de K-vecinos mas cercanos. El mismo lo que hace es que dado una series de vectores (en este caso los mismos describirán las imágenes de la base de datos de entrenamiento, transformadas o no) los representa en el espacio de N dimensiones siendo N la cantidad de elementos de los vectores y para cada uno de ellos guarda la identificación. Una vez hecho esto procederemos para cada imagen a identificar los K elementos mas cercanos, en cuanto a distancia euclideana, del espacio de entrenamiento. La identificación correspondiente a la imagen a clasificar será la que sea mayoritaria entre esos vecinos previamente hallados. Este algoritmo al ser una heurística claramente no hallará la identificación con total certeza pero será una buena aproximación de la misma. Por este motivo se procederá a testear la efectividad del mismo. Una buena forma de hacerlo sería con k-fold cross validation. Además también es posible aprovechar la validación para encontrar que parametros de las funciones arrojan los mejores resultados.

k-fold cross validation es un método muy simple que lo único que hace es partir el conjunto de entrenamiento en k partes del mismo tamaño de forma aleatorio de modo tal que tome k-1 partes para la base de datos de entrenamiento de KNN y la parte restante se usa de validación para ver con que grado de precisión identifica cada una de las imágenes. Lo interesante del mismo es que al partir el conjunto de forma aleatoria y usar solo una parte como validación cada vez permite verificar el rendimiento previniendo caer en "sobre-entrenamiento".

2. Desarrollo

La implementación de KNN tomará como parametros: un vector *conjunto* donde cada uno de sus elementos es el vector que describa cada imagen (antes o después de PCA); un vector *clase* donde el elemento i contenga la ID de la foto representada en el elemento i del vector *conjunto*; vector que representa la imagen a clasificar, *imagen*,y el parámetro K que indica la cantidad de vecinos a clasificar.

Lo primero que hará es encontrar la distancia euclideana entre el vector *imagen* y cada uno el los vectores de *conjunto*. Eso se hace simplemente haciendo un ciclo for que recorra *conjunto* y otro anidado que recorra cada uno de los elementos de ambos vectores para de ese modo poder ir sumado el cuadrado de las diferencias entre los mismos y de ese modo hallar la distancia al aplicar una raíz cuadrada al final del proceso.Entonce para cada uno de los elementos de conjunto se guarda en un vector, *distancias*, que guarda el par (*distancia*,*numerodeelemento*).

Luego lo ordenamos de menor a mayor, ya que de este modo aseguro que al principio del vector *distancias* estén los k pares con menos valor de distancia. Extraigo los mismos y los pongo en nuevo vector que llamo clase de los cercanos que en lugar de guardar la distancia de los mismos y el lugar, i,que ocupaban en el vector *conjunto* contendría la ID de dichas imágenes. Esto simplemente se hace leyendo lo que este guardado en la posición i del vector *clase*. Dado esto lo ordenaré ya que ordenarlo se hace en $O(k.log_2(k))$ y es una forma efectiva de contar la cantidad de apariciones de cada elemento en un vector. Lo único que se restaría hacer es recorrer el vector contando cuantos elementos consecutivos hay iguales hasta que cambie el ID, ya que todos los elementos iguales deberán estar consecutivos al ser un arreglo ordenado. Finalmente KNN arrojará el valor de la ID con mayor cantidad de apariciones.

K-fold va a estar implementado como una clase que tiene 2 funciones publicas. La primera de ellas se encarga de construirla mientras que la segunda se encarga de arrojar los resultados requeridos. Empecemos describiendo el constructor. Al principio el mismo toma un vector de elementos de cualquier tipo y lo mezcla al azar para poder hacer métodos mas simples en la partición pero que mantengan la aleatoriedad. En caso de poder partirlo en K partes iguales, va colocando de a uno en cada fold en orden yendo los elementos a la fold $i \ mod \ K$, siendo i la posición en la que se encuentran en el vector de entrada y $a \ mod \ b$ es el resto de a divido por b. En caso de no poder realizar dicha partición tirara error. Una vez hecho eso hay una función que se encarga para la i-esima fold de devolver en un vector, *testing*, todos los elementos que pertenecen a dicha fold y en un vector, *training*, todos los elementos que no lo hacen.

Los procesos relacionados con PCA se implementaron como una clase que se realiza las siguientes operaciones:

- Obtiene la media de los vectores de imagen en base a un conjunto de imágenes
- Calcula la matriz M mencionada en la introducción en base a un conjunto de imágenes y su media
- Calcula α autovectores y autovalores de una matriz M en base a cierta tolerancia
- Calcula la transformación característica de un vector en base a los autovalores y autovectores pasados

Con respecto al cálculo de autovalores y autovectores, el Método de la Potencia explicado anteriormente fue implementado multiplicando la matriz M por un vector hasta que las distancias entre las iteraciones entren dentro de cierta tolerancia, a modo de poder simular tender la cantidad de iteraciones a infinito.

3. Resultados y Discusión

Durante todos los experimentos a realizar realizamos 10 particiones del conjunto de entrenamiento con distintas particiones aleatorias con k-fold (k=5) y realizamos una media obtenida de hacer el promedio de los resultados obtenidos de usar cada una de estas "foldçomo conjunto de verificación. Esto se hizo para evitar en la medida de lo posible los errores que podrían generarse por la forma en que se elija el conjunto de verificación.

Los próximos experimentos, salvo que se indique lo contrario, se realizaron con la totalidad de las imágenes de personas incluidas en el archivo comprimido que se adjunta junto a este informe.

3.1. valores de α y k para KNN con PCA

Primero definamos α como la cantidad de autovectores que se consideran para PCA y K como la cantidad de vecinos mas cercanos a tener en cuenta. Con eso realizaremos una comparación de K y α en función de la accurracy.(figura 1). Durante dicha comparación variamos los valores de K y alpha en cantidaes razonablemente grandes para obtener una mejor idea del valor ideal de k y α. En el particular se observa que los mejores resultados se obtienen cuanto mas chico es K y mas grande es α. En particular para todo α vale que su mejor resultado se obtiene para $k = 1$. Además como knn es un algoritmo relativamente rápido en relación a PCA, se que K no influirá en la complejidad del algoritmo. Así que es de suma importancia encontrar un valor de *alpha* tal que la relación entre tiempo de ejecución y calidad de la solución sea lo más óptima posible.

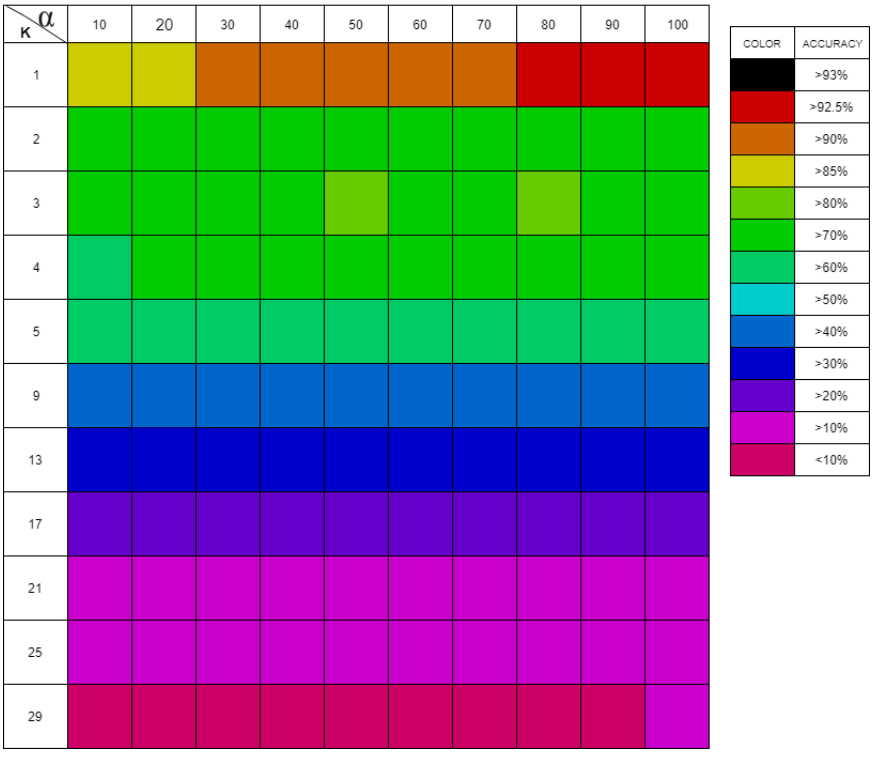

Figura 1: *
Accurracy de KNN con PCA en relacion de α y K .

3.2. valor de α para KNN con PCA

Para hallar dicho *alpha*, aprovecharemos el conocimiento previamente obtenido acerca de que siempre es mejor tener $k = 1$ para esta base de entrenamiento. Por lo tanto solo será necesario realizar un gráfico de α en función de tiempo de ejecución (figura 2) y otro de α en función de .ªccuracy"(figura 3). La decisión de cual tomar depende de lo que se prioriza, pero ya que en nuestro caso será una buena relación entre los resultados y el tiempo de ejecución, vamos a tomar el punto en que las mejores obtenidas dejan de ser notorias. Esto es buena idea ya el tiempo de ejecución aumenta de forma aproximadamente lineal mientras que los resultados no lo hacen. El punto será 20 ya que a partir de allí la mayoría deja de ser notable. Sin embargo si lo que se buscara es siempre calidad de la solución se puede ver que tiende a tener levemente mas accurracy en valores aún mas grandes, pero la ganancia es cada vez menor. Esto se debe a que por como funciona PCA los valores cargan progresivamente cada vez menos información

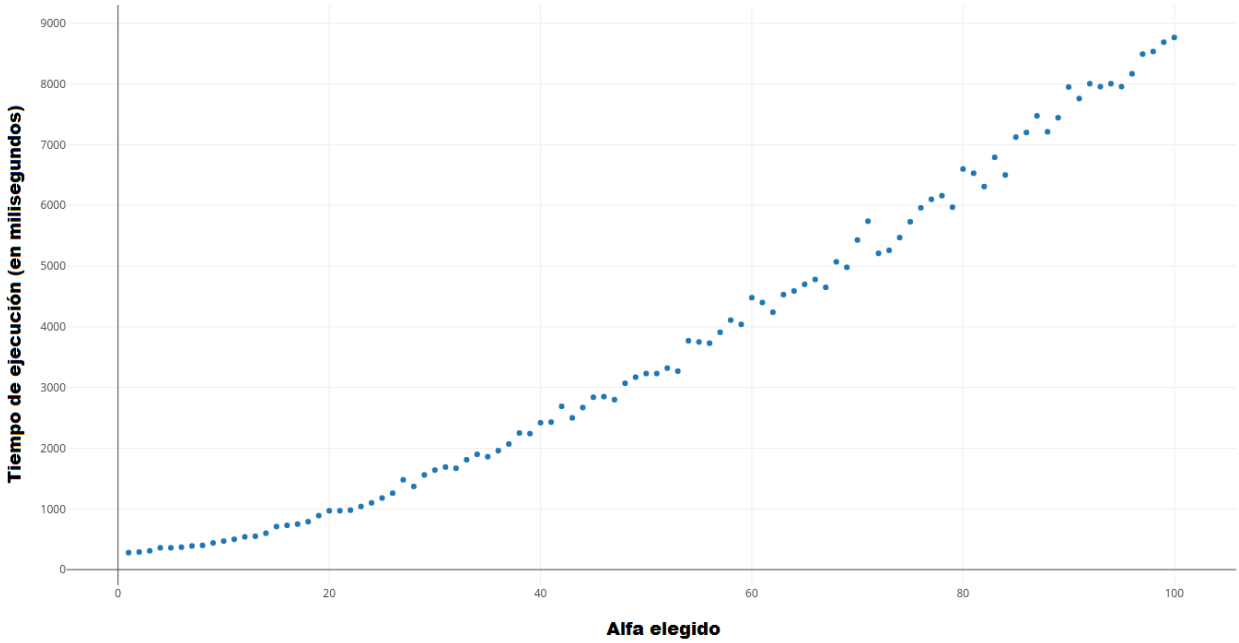

Figura 2: *
tiempo de ejecución de KNN con PCA en relación de α

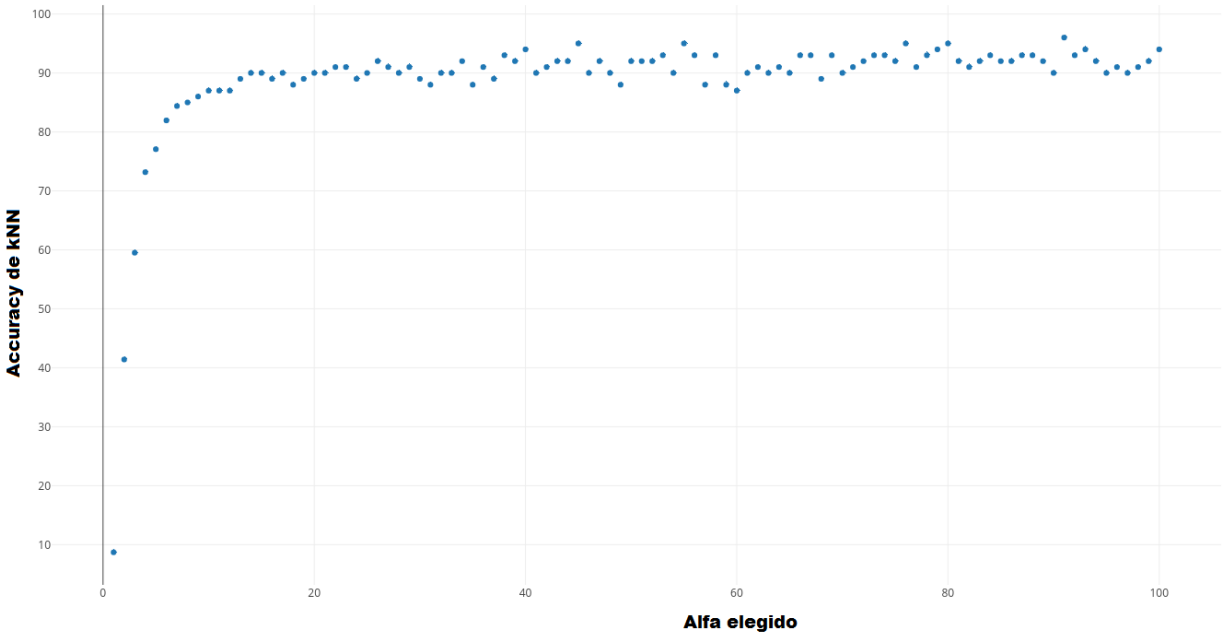

Figura 3: *
Accuracy de KNN con PCA en relación de α

3.3. Comparacion de KNN con y sin PCA para distintos valores de K

Para sacar conclusiones acerca de esto hicimos un gráfico de la .ªccuracy"de KNN con PCA en función del valor de K (usando $\alpha = 20$) y en el mismo gráfico incluimos un gráfico de la .ªccuracy"de KNN sin PCA, también en función del valor de K (figura 4). Esto nos permite fácilmente comparar para que valores de K es mas eficiente cada uno de estos métodos y por cuanto. Al hacerlo vimos la .ªccuracy"de KNN con y sin PCA fue similar. Por ese motivo haremos el mismo gráfico pero en función de su F-1 Score (figura 5) para concluir definitivamente que tan distinto es el rendimiento de ambos métodos.

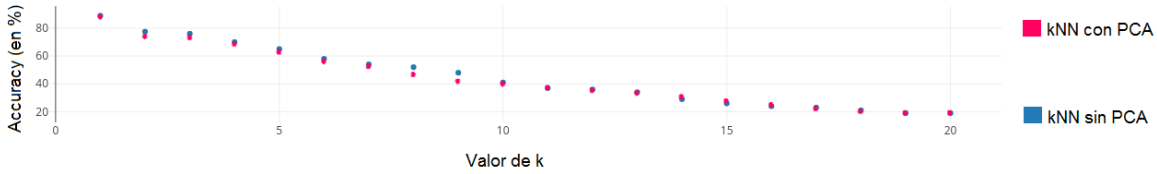

Figura 4: *
Accuracy de KNN con PCA en relación de k

3.4. Valor de K ideal para cada tamaño de la base de datos de entrenamiento

Finalmente analizamos el rendimiento de KNN con PCA, usando el valor de α previamente encontrado, en función del tamaño de la base de datos y de K. (figura 6). Lo que hallamos es que sin importar el tamaño de la base de datos el mejor rendimiento es con $k = 1$

4. Conclusiones

Al finalizar todos estos experimentos hallamos que el valor ideal de K para KNN es 1, mientras que el valor ideal de α en PCA es 20. . Para esto valores tanto KNN, con PCA, como sin PCA, obtuvieron una precisión extremadamente alta lo cual los hace muy útiles para trabajar con esta base de datos que presenta la particularidad de que las fotos son fotos carnet por lo que la diferencia en luminosidad, posición o color del fondo no son un factor. Sin embargo es importante mencionar que si las posiciones de las personas o las condiciones ambientales cambiaran lo suficiente estos algoritmos fallarían mientras que una red neuronal correctamente entrenada podría reconocerla a pesar de los cambios de posición/luminosidad/fondo de la imagen.

Además encontramos que la diferencia de KNN con y sin PCA es extremadamente pequeña, favoreciendo a la versión sin PCA, para el conjunto de datos en los que se experimentó. Sin embargo si se desea buscar precisión a costa de rendimiento aplicar PCA con $\alpha > 80$sería de cierta utilidad.

5. Apéndices

Apéndice A - Enunciado

Introducción

Como plan estratégico para el desarrollo del país, las autoridades nacionales están a días de inaugurar el prometedor soja valley que será el pilar económico de los a nos venideros. Dado que el rimbombante emprendimiento albergará a las industrias más importantes para el país, se necesita un sistema de reconocimiento de trabajadores que ingresen a las instalaciones dado que se estará manejando información confidencial a diario.

Si bien es muy factible que existan mejores formas y más económicas para reconocer diariamente a un trabajador, las autoridades están muy interesadas en desarrollar un sistema biométrico basado en reconocimiento facial.

En este contexto, las autoridades han contactado al Departamento de Computación para el desarrollo del sistema de reconocimiento facial que se utilizará en el flamante soja valley. Como punto de partida para la realización de un prototipo, nos han provisto de una base de datos preliminar para poder realizar una prueba de concepto.

El foco de este trabajo está puesto en la experimentación científica de un método de reconocimiento caras simplificado usando clasificación por vecinos más cercanos y reducción de la dimensionalidad.

Metodología

Se quiere desarrollar un algoritmo de clasificación supervisada el cual se deberá entrenar con una base de caras conocidas que luego nos servirá para reconocer otras instancias de esas caras no presentes en la base de datos de entrenamiento. Como instancias de entrenamiento, se tiene un conjunto de N personas, cada una de ellas con M imágenes distintas de sus rostros en escala de grises y del mismo tamaño. Cada una de estas imágenes sabemos a qué persona corresponde.

Reconocimiento de caras

El objetivo del reconocimiento de caras consiste en utilizar la información de la base de datos para, dada una nueva imagen de una cara sin etiquetar, determinar a quién corresponde. En este trabajo práctico nos vamos a basar en el trabajo de Turk and Pentland [3] en el cual se plantea un esquema muy simple de reconocimiento de caras.

Una primera aproximación es utilizar el conocido algoritmo de k vecinos más cercanos o kNN [2], por su sigla en inglés. En su versión más simple, este algoritmo considera a cada objeto de la base de entrenamiento como un punto en el espacio euclídeo m-dimensional, para el cual se conoce a qué clase corresponde (en nuestro caso, qué persona es) para luego, dado un nuevo objeto, asociarle la clase del o los puntos más cercanos de la base de datos.

Procedimiento de k vecinos más cercanos:

- Se define una base de datos de entrenamiento como el conjunto $D = \{x_i : i = 1, ..., n\}$
- Luego, se define m como el número total de píxeles de la imagen i-ésima almacenada por filas y representada como un vector $x_i \in \mathbb{R}^m$.
- De esta forma, dada una nueva imagen $x \in \mathbb{R}^m$, talque $x \notin D$, para clasificarla simplemente se busca el subconjunto de los k vectores $\{x_i\} \subseteq D$ más cercanos a x, y se le asigna la clase que posea el mayor número de repeticiones dentro de ese subconjunto, es decir, la moda.

El algoritmo del vecino más cercano es muy sensible a la dimensión de los objetos y a la variación de la intensidad de las imágenes. Es por eso, que las imágenes dentro de la base de datos D se suelen preprocesar para lidiar con estos problemas.

Teniendo en cuenta esto, una alternativa interesante de preprocesamiento es buscar reducir la cantidad de dimensiones de las muestras para trabajar con una cantidad de variables más acotada y, simultáneamente, buscando que las nuevas variables tengan información representativa para clasificar los objetos de la base de entrada.

En esta dirección, consideraremos el método de reducción de dimensionalidad Análisis de componentes principales o PCA (por su sigla en inglés) dejando de la lado los procesamientos de imágenes que se puedan realizar previamente o alternativamente a aplicar PCA.

Análisis de componentes principales

El método de análisis de componentes principales o PCA consiste en lo siguiente. Sea $\mu = (x_1 + ... + x_n)/n$ el promedio de las imágenes $D = \{x_i : i = 1, ..., n\}$ tal que $x \in \mathbb{R}^m$. Definimos $X \in \mathbb{R}^n xm$ como la matriz que contiene en la i-ésima fila al vector $(x_i - \mu)^t/\sqrt{n-1}$. La matriz de covarianza de la muestra X se define como $M = X^t X$.
Siendo v_j el autovector de M asociado al j-ésimo autovalor, al ser ordenados por su valor absoluto, definimos para $i = 1, ..., n$ la transformación característica de xi como el vector $tc(x_i) = (v_1 x_i, v_2 x_i, ..., v_\alpha x_i)^t \in R^\alpha$, donde $\alpha \in \{1, ..., m\}$ es un parámetro de la implementación. Este proceso corresponde a extraer las α primeras componentes principales de cada imagen. La idea es que $tc(x_i)$ resuma la información más relevante de la imagen, descartando los dimensiones menos significativas.

Imágenes de gran tamaño

La factibilidad de aplicar PCA es particularmente sensible al tamaño de las imágenes de la base de datos. Por ejemplo, al considerar imágenes en escala de grises de 100×100 pixeles implica trabajar con matrices de tamaño 10000×10000. Una alternativa es reducir el tamaño de las imágenes, por ejemplo, mediante un submuestreo (eliminación intercalada de filas y columnas de la imagen).

Sin embargo, es posible superar esta dificultad en los casos donde el número de muestras es menor que el número de variables, ya que es posible encontrar una relación entre los autovalores y autovectores de la matriz de covarianza $M = X^t X$ y la matriz $\hat{M} = XX^t$

Clasificación con kNN y PCA

El método PCA previamente presentado sirve para realizar una transformación de los datos de entrada a otra base y así trabajar en otro espacio con mejores propiedades que el original.
Por lo tanto, el proceso completo de clasificación se puede resumir como: Dada una nueva imagen x de una cara, se calcula $tc(x)$, y se compara con $tc(x_i), \forall x_i \in D$, para luego clasificar mediante kNN.

Validación cruzada

Finalmente, nos concentramos en la evaluación de los métodos y en la correcta elección de sus parámetros.
Dado que necesitamos conocer previamente a qué persona corresponde una imagen para poder estimar la correctitud de la clasificación, una alternativa es particionar la base de entrenamiento en dos, utilizando una parte de ella en forma completa para el entrenamiento y la restante como test, pudiendo así corroborar la clasificación realizada, al contar con el etiquetado del entrenamiento. Sin embargo, realizar toda la experimentación sobre una única partición de la base podría resultar en una incorrecta estimación de parámetros, dando lugar al conocido problema de overfitting.

Por lo tanto, se estudiará la técnica de *crossvalidation*[2], en particular el $K-foldcrossvalidation$[3], para realizar una estimación de los parámetros de los métodos que resulte estadísticamente más robusta.

[3]No confundir el K de K-fold con el k de kNN, ambos son parámetros de los métodos respectivos que no

Enunciado

Se pide implementar un programa en C o C++ que lea desde archivos las imágenes de entrenamiento correspondientes a distintas caras y que, utilizando los métodos descriptos en la sección anterior, dada una nueva imagen de una cara determine a qué persona pertenece.
Para ello, el programa deberá implementar el algoritmo de kNN así como también la reducción de dimensión utilizando PCA.

Con el objetivo de obtener la transformaciones características de cada método, se deberá implementar el método de la potencia con deflación para la estimación de autovalores/autovectores de la matriz de covarianza en el caso de PCA.

Se recomienda realizar tests para verificar la implementación del método en casos donde los autovalores y autovectores sean conocidos se antemano. también, puede resultar de utilidad comparar con los datos provistos por la cátedra, utilizando Matlab/Octave o alguna librería de cálculo numérico.

Se pide realizar un estudio experimental de los métodos propuestos sobre una base de entrenamiento utilizando la técnica K-fold cross validation mencionada anteriormente, con el objetivo de analizar el poder de clasificación y encontrar los mejores parámetros de los métodos.

Se pide desarrollar una herramienta alternativa que permita trabajar bajo ciertas condiciones con imágenes de tamaño mediano/grande y probar lo siguiente: Se deberá trabajar al menos con la siguiente base de datos caras: `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`

Experimentación.

Para guiar la experimentación, se detallan los siguientes lineamientos y preguntas:

- Analizar la calidad de los resultados obtenidos al combinar kNN con y sin PCA, para un rango amplio de combinaciones de valores de k y α. Llamamos k a la cantidad de vecinos a considerar en el algoritmo kNN α a la cantidad de componentes principales a tomar.

- Analizar la calidad de los resultados obtenidos al combinar kNN con PCA, para un rango amplio de cantidades de imágenes de entrenamiento. Utilizar desde muy pocas imágenes de entrenamiento hasta todas las disponibles para identificar en que situación se comporta mejor cada uno de los métodos.

- ¿Cómo se relaciona k con el tamaño del conjunto de entrenamiento? Pensar el valor máximo y mínimo que puede tomar k y qué sentido tendrían los valores.

También, **se debe** considerar en los análisis anteriores el tiempo de ejecución.

La calidad de los resultados obtenidos será analizada mediante diferentes métricas:

1. Accuracy
2. Precision/recall
3. F1-Score

En particular, la métrica más importante que **debe** reportarse en los experimentos es la tasa de efectividad lograda o accuracy, es decir, la cantidad caras correctamente clasificadas respecto a la cantidad total de casos analizados. también, se debe utilizar al menos otra de las métricas mencionadas, aunque no necesariamente para todos los experimentos realizados.

- Realizar los experimentos de los ítems anteriores para valores distintos de K del método K-fold[4], donde K a la cantidad de particiones consideradas para el cross-validation.
  - Justificar el por qué de la elección de los mismos. >Cuál sería su valor máximo?
  - ¿En qué situaciones es más conveniente utilizar K-fold con respecto a no utilizarlo?
  - ¿Cómo afecta el tamaño del conjunto de entrenamiento?
- En base a los resultados obtenidos para ambos métodos, seleccionar aquella combinación de parámetros que se considere la mejor alternativa, con su correspondiente justificación, compararlas entre sí y sugerir un método para su utilización en la práctica.

En todos los casos es obligatorio fundamentar los experimentos planteados, proveer los archivos e información necesaria para replicarlos, presentar los resultados de forma conveniente y clara, y analizar los mismos con el nivel de detalle apropiado. En caso de ser necesario, es posible también generar instancias artificiales con el fin de ejemplificar y mostrar un comportamiento determinado.

Puntos opcionales (no obligatorios)

- Mostrar que si tenemos la descomposición $M = U\Sigma V^t$, V es la misma matriz queobtenemos al diagonalizar la matriz de covarianzas.
- Realizar experimentos utilizando otras imágenes de caras tomadas por el grupo. Tener en cuenta lo mencionado sobre el tamaño de las matrices a procesar con PCA. Reportar resultados y dificultades encontradas.
- Proponer y/o implementar alguna mejora al algoritmo de kNN. Por ejemplo, no considerar votación y utilizar la cercanía a la media de de cada clase como criterio de clasificación.
- Implementar y experimentar un métodos de detección de caras para encontrar si una imagen contiene o no una cara. Proponer un valor de confianza para la respuesta de la detección. Es decir, se quiere intentar responder la pregunta de si el sistema es capaz no solo de reconocer caras con los que fue entrenado, sino si es posible discernir entre una imagen de una cara que no se encontraba en la base de entrenamiento y una imagen con un objeto o contenido que no sea una cara.
- Aplicar técnicas de procesamiento de imágenes a las imágenes de caras previo a la clasificación[1]. Analizar como impacta en la clasificación la alteración de la intensidad de los pixeles, el ruido introducido en las imágenes y la variación de la ubicación y posición de las personas.

Formato de entrada/salida

El ejecutable producido por el código fuente entregado deberá contar con las funcionalidades pedidas en este apartado. El mismo deberá tomar al menos tres parámetros por línea de comando con la siguiente convención:

```
$ ./tp2  -m <method> -i <train_set> -q <test_set> -o <classif>
```

donde:

- `<method>` el método a ejecutar con posibilidad de extensión (0: kNN, 1: PCA + kNN,... etc)
- `<train_set>` será el nombre del archivo de entrada con los datos de entrenamiento.
- `<test_set>` será el nombre del archivo con los datos de test a clasificar
- `<classif>` el nombre del archivo de salida con la clasificación de los datos de test de `<test_set>`

[4]Para esta tarea en particular, se recomienda leer la rutina cvpartition provista por Octave/MATLAB.

Todos los archivos de entrada/salida deberán estar en .csv y siguiendo el formato siguiente:
`<archivo1>, id1,`
`...`
`<archivoN>, idN,`
donde `idX` es un entero positivo entre 1 y la cantidad de personas en el dataset.

Un ejemplo de invocación con los datos provistos por la cátedra sería el siguiente:
```
$ ./tp2 -m 1 -i train.csv -q test.csv -o result.csv
```

Además, el programa deberá imprimir por consola un archivo, cuyo formato queda a criterio del grupo, indicando la tasa de reconocimiento obtenida para cada conjunto de test y los parámetros utilizados para los métodos.
Nota: cada grupo tendrá la libertad de extender las funcionalidades provistas por su ejecutable. En particular, puede ser de utilidad alguna variante de toma de parámetros que permita entrenar con un porcentaje de la base de datos de entrenamiento y testear con el resto (ver archivos provistos por la cátedra). Además, puede ser conveniente separar la fase de entrenamiento de la de testeo/consulta para agilizar los cálculos.

Fecha de entrega

- Formato Electrónico: Viernes 25 de Mayo de 2018 hasta las 23:59 hs, enviando el trabajo (informe + código) a la dirección metnum.lab@gmail.com. El subject del email debe comenzar con el texto [TP2] seguido de la lista de apellidos de los integrantes del grupo separados por punto y coma ;.
  Se ruega no sobrepasar el máximo permitido de archivos adjuntos de 20MB. Tener en cuenta al realizar la entrega de no ajuntar bases de datos disponibles en la web, resultados duplicados o archivos de backup.
- Formato físico: Lunes 28 de Mayo de 2018 a las 18 hs. en la clase de laboratorio.
- Pautas de laboratorio: `http://www-2.dc.uba.ar/materias/metnum/homepage.html`

Importante: El horario es estricto. Los correos recibidos después de la hora indicada serán considerados re-entrega.

Referencias

[1] Tinku Acharya and Ajoy K Ray. Image processing: principles and applications. John Wiley & Sons, 2005.

[2] Richard O Duda, Peter E Hart, and David G Stork. Pattern classification. John Wiley & Sons, 2012.

[3] Matthew Turk and Alex Pentland. Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1):71-86, 1991.