



## Reconocimiento de caras

---

### 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 años 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  $\mathcal{D} = \{x_i : i = 1, \dots, n\}$ .
- 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  $x \notin \mathcal{D}$ , para clasificarla simplemente se busca el subconjunto de los  $k$  vectores  $\{x_i\} \subseteq \mathcal{D}$  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  $\mathcal{D}$  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  $\mu = (x_1 + \dots + x_n)/n$  el promedio de las imágenes  $\mathcal{D} = \{x_i : i = 1, \dots, n\}$  tal que  $x_i \in \mathbb{R}^m$ . Definimos  $X \in \mathbb{R}^{n \times m}$  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  $i = 1, \dots, n$  la *transformación característica* de  $x_i$  como el vector  $\mathbf{tc}(x_i) = (v_1 x_i, v_2 x_i, \dots, v_\alpha x_i)^t \in \mathbb{R}^\alpha$ , donde  $\alpha \in \{1, \dots, m\}$  es un parámetro de la implementación. Este proceso corresponde a extraer las  $\alpha$  primeras *componentes principales* de cada imagen. La idea es que  $\mathbf{tc}(x_i)$  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  $100 \times 100$  píxeles implica trabajar con matrices de tamaño  $10000 \times 10000$ . 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  $\hat{M} = X * X^t$ .

## 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  $\mathbf{tc}(x)$ , y se compara con  $\mathbf{tc}(x_i)$ ,  $\forall x_i \in \mathcal{D}$ , 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 *cross validation*[2], en particular el *K-fold cross validation*<sup>1</sup>, para realizar una estimación de los parámetros de los métodos que resulte estadísticamente más robusta.

## 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 las 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 de antemano. También, puede resultar de utilidad comparar con los datos provistos por la cátedra, utilizando Matlab/Octave o alguna librería de cálculo numérico.

---

<sup>1</sup>No confundir el  $K$  de  $K$ -fold con el  $k$  de  $kNN$ , ambos son parámetros de los métodos respectivos que no están relacionados necesariamente

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:

- Dada una matriz de covarianza  $M = X^t * X$ , encontrar una relación entre sus autovalores y sus autovectores con los de la matriz  $\hat{M} = X * X^t$ .

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  $\alpha$ . Llamamos  $k$  a la cantidad de vecinos a considerar en el algoritmo *kNN* y  $\alpha$  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*<sup>2</sup>, donde  $K$  a la cantidad de particiones consideradas para el cross-validation.

---

<sup>2</sup>Para esta tarea en particular, se recomienda leer la rutina `cvpartition` provista por Octave/MATLAB.

- 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 que obtenemos 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 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 píxeles, 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>`

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](mailto: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 juntar 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.