DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP Nº 2

Cara a cara

Métodos Numéricos

Primer cuatrimestre 2017

Integrante	LU	Correo electrónico
Ramos Ricardo	841/11	riki_german@yahoo.com.ar
Ingani Bruno	50/13	chino117@hotmail.com
Lucero Emiliano	108/12	eslucero2010@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://www.fcen.uba.ar

Índice

1. Introducción

El presente trabajo práctico propone una introducción a las técnicas de Aprendizaje Automático (Machine Learning) aplicando conceptos básicos de la materia: autovalores y autovectores, métodos de la potencia, descomposición en valores singulares, métodos iterativos y cuadrados mínimos.

1.1. El problema a resolver

El reconocimento de caras es un método biométrico para la identificación de personas mediante características faciales. Con muchas aplicaciones prácticas se busca un sistema de reconocimiento que sea flexible en cuanto condiciones de capacidad de procesamiento, memoria y velocidad de una máquina.
Nuestro sistema de reconocimiento de rostros debe clasificar datos de alta dimensión por lo tanto se debe reducir la dimensión, es decir reducción de características principales de un rostro, para permitir una velocidad y consumo de recursos aceptable. Para lograr esto utilizaremos el método de reducción de dimensiones, PCA, junto con el de clasificación kNN. Adicionalmente queremos saber si una imagen cualquiera pertenece a un rostro o no.
Se cuenta con una base de datos con n cantidad de imágenes($cant\_imgs$) de rostros, en escala de grises, tomadas como vectores de tamaño $m = alto \times ancho$, etiquetadas por nombres según corresponda.
El objetivo es utilizar dicha información para, dada una nueva imagen de un rostro sin etiquetar, determinar si pertenece a la base de datos teniendo en cuenta factores de calidad y tiempo de ejecución requeridos.

1.2. Método kNN

Este algoritmo considera a cada objeto de la base de entrenamiento como un punto en el espacio, para el cual se conoce a qué rostro corresponde. Luego, al obtener una nueva imagen, se buscan los k vecinos más cercanos y se le asigna el rostro que posea el mayor número de repeticiones dentro de ese subconjunto, es decir, la moda.

Con este objetivo, representamos a cada imagen de nuestra base de datos como un vector $x_i \in R^{ancho \times alto}$, con $1 \leq i \leq n$, $n = \#imagenes$ y $ancho \times alto$ las dimensiones de la imagen, e interpretamos las imágenes a clasificar mediante el algoritmo kNN. En otras palabras, nuestra base de datos pasa a ser una matriz $R^{n \times m}$, $m = ancho \times alto$, donde cada fila es una imagen y la distancia entre imágenes pasa a ser distancia entre vectores.

Tiene la desventaja de que puede ser muy costoso dependiendo de la dimensión de los objetos y que la clasificación sea lenta dependiendo del contexto. Teniendo en cuenta esto, una alternativa es preprocesar las imágenes para reducir la cantidad de dimensiones de las muestras con el objetivo de trabajar con una cantidad de variables más acotada y, al mismo tiempo, buscar que las nuevas variables tengan información representativa para clasificar los objetos de la base de entrada. Utilizaremos uno de estos métodos, Análisis de Componentes Principales(PCA), el cual se detalla a continuación.

1.3. Método de Análisis de Componentes Principales(PCA)

PCA es un método de extracción de características. El objetivo principal que tiene es reducir la dimensionalidad de un conjunto de observaciones con una gran cantidad de variables, mediante el estudio de la varianzas−covarianzas entre las variables que integran los datos de entrada.
A partir de la proyección de los datos de entrada sobre las direcciones de máxima varianza se obtendrá un nuevo espacio de representación de los datos en el que se puede eliminar fácilmente aquellas componentes con menor varianza, garantizando la mínima pérdida de información. Sin embargo, este método presenta

algunas desventajas como la dificultad de análisis de los datos resultantes o una función de coste poco robusta frente al ruido.

2. Desarrollo

Dada la introducción teórica, comenzaremos a detallar el rol de cada método con el fin de describir como es la metodología utilizada para la clasificación de dichas imágenes.

2.1. k-Vecinos más cercanos(kNN)

Consideramos cada una de las características de una imagen en nuestra base de datos como una dimensión diferente en algún espacio, y tomamos el valor que una observación tiene para esta característica como su coordenada en esa dimensión, obteniendo un conjunto de puntos en el espacio. Entonces consideramos la similitud de dos imágenes como la distancia entre sus coordenadas en este espacio bajo una métrica apropiada.
La manera en que el algoritmo kNN decide cuales de las imágenes de nuestra base de datos son suficientemente similares para ser consideradas cuando buscamos la clase de una nueva observación es tomar las k imágenes más cercanas a la nueva observación, y luego tomar la clase más común entre ellas. Esta es la forma canónica de kNN.

El algoritmo puede ser resumido como:

1. Un entero k es especificado, junto con una nueva imagen
2. Seleccionamos las k imágenes en nuestra base de datos que están más cerca de la nueva imagen
3. Encontramos la clase más común de estas imágenes
4. Esta es la clasificación que le damos a la nueva imagen

Una variante que utilizamos de este método consiste en considerar, además de la cantidad de repeticiones, las distancias entre la imagen nueva y los k vecinos más cercanos a la hora de clasificarla. Básicamente, por cada repetición de rostro sumamos a un contador la inversa de la distancia, luego elegimos como rostro para la imagen nueva aquel cuyo contador sea el máximo. Esto ayuda a reducir el efecto negativo que puede tener en la clasificación el hecho de que haya muchas más imágenes correspondientes a una persona en particular en comparación con el resto. Llamamos a esto kNN promediado por distancia.

En ambos casos utilizamos la norma 2, queda como propuesta de experimento ver si al usar norma 1 o norma infinito cambia la performance del sistema.

2.2. PCA

El objetivo principal de PCA es encontrar P tal que maximice $var(XP)$.

Dada una matriz $X \in R^{n \times m}$ centrada a la media, busca una matriz $P \in R^{m \times m}$ ortogonal tal que permita realizar el siguiente cambio de variables

$$T = XP$$

de forma tal que las nuevas variables $t_1, \ldots, t_p$ no estén correlacionadas entre sí y estén ordenadas por su varianza.

¿ Por qué importa la varianza? Nosotros queremos que el sistema sepa diferenciar entre imágenes de rostros. Éstas pueden tener características en común y características que las distinguen. Las que son en común no nos dan mucha información a la hora de diferenciar imágenes, mientras que las otras sí. Entonces, nos importa más que el sistema las diferencie en base a las características que no son comunes. Mientras más común es una característica, menor será la varianza de las variables que influyen en ella. Por lo tanto,

si tomamos las variables con mayor varianza y comparamos imágenes en base a ellas, diferenciaremos considerando las características que más distinguen a las imágenes.

Esto lo logra tomando P de forma tal que esté compuesto por los autovectores de la matriz $cov(X, X)$. Veamos por qué. Como dijimos antes, buscamos la matriz $P \in R^{m \times m}$ ortogonal tal que el cambio de variables $T = XP$ haga que las nuevas variables $t_1, \ldots, t_m$ no estén correlacionadas entre sí y que estén ordenadas por su varianza. En otras palabras, queremos que la matriz de covarianza $cov(T, T)$ sea diagonal y que sus elementos estén en orden decreciente. Esto es porque como los elementos fuera de la diagonal representan la covarianza entre distintas variables entonces si la matriz de covarianza queda diagonal estos valores son cero, es decir cero covarianza entre dos variables distintas. A su vez la i-ésima diagonal representa la varianza de la i-ésima variable, para $1 \leq i \leq \#variables$, y por construcción quedan en orden decreciente dando como resultado que las primeras diagonales son asociadas a variables con mayor varianza.
Como X es centrada a la media, T también lo es. Entonces vale que

$$cov(T, T) = \frac{1}{n-1} T^t T = \frac{1}{n-1} (P^t X^t)(XP) = \frac{1}{n-1} P^t X^t XP = P^t SP$$

donde $S = \frac{1}{n-1} X^t X$ es la matriz de covarianza de X. Como S es simétrica, se la puede diagonalizar ortogonalmente, valiendo entonces que $S = UDU^t$ donde D es una matriz diagonal que contiene los autovalores de X ordenados en forma descendiente y U está formada por los autovectores correspondientes a dichos autovalores. Reemplazando en la primera ecuación queda que

$$cov(T, T) = P^t SP = P^t UDU^t P$$

Tomando $P = U$, donde U es ortogonal y por lo tanto $U^t U = I$, resulta que

$$cov(T, T) = U^t UDU^t U = D$$

que era lo que queríamos que cumpla P.

Esto sólo nos asegura que $T = XP$ cumple que maximiza $var(XP)$, no cambiamos la dimensionalidad. Para lograrlo, armamos P sólo con los primeros k autovectores de $cov(X, X)$ asociados a los k primeros autovalores que por construcción son los de mayor valor y representan la mayor varianza dentro del conjunto de imágenes, por lo que queda que $P \in R^{n \times k}$ y $T \in R^{n \times k}$. Luego la dimensión se reduce a k que es la cantidad de vectores de P y son los que mejor describen la distribución de datos.

La forma en la que usamos PCA en el sistema es similar a lo anterior. Dada nuestra base de datos de imágenes $X \in R^{n \times m}$, tomamos su media, creamos la matriz de covarianza $cov(X, X)$ y buscamos sus primeros k autovectores para armar la matriz P, para realizar el cambio de variables (y dimensionalidad). La matriz de covarianza se calcula usando su definición matemática y los autovectores se obtienen usando Método de la Potencia + Deflación. En el contexto de reconocimiento de rostros, a los autovectores de $cov(X, X)$ también se los llama autocaras o eigenfaces y forman un espacio vectorial.

2.3. Relación entre autovectores de XX^t y $X^t X$

Como vimos antes, nos interesa hallar los autovectores de la matriz $cov(X, X) = \frac{X^t X}{n-1}$ cuando $X \in R^{n \times m}$. Un problema que puede surgir antes de lograrlo es que el tamaño de las imágenes, m, sea muy grande causando que la matriz $X^t X \in R^{m \times m}$ sea demasiado grande. Si tenemos que m es mucho mayor que la cantidad de imágenes, n, se nos ocurre entonces hallar sus autovectores de la siguiente manera:

1. Sea v un autovector de $XX^t \in R^{n \times n}$ con autovalor $\lambda \neq 0$. Entonces vale que $XX^t v = \lambda v$, y $X^t X(X^t v) = \lambda(X^t v)$.

   Como $\lambda \neq 0$, se deduce que $X^t v \neq 0$, por lo tanto $X^t v$ es un autovector de $X^t X$ con autovalor λ.

2. Los autovectores del autovalor 0 son los elementos del espacio nulo de X: $X^tXv = 0$ implica $v^tX^tXv = \|X^tv\|^2 = 0$, por lo que $X^tv = 0$.

Por lo tanto, se obtienen los autovectores de X^tX a partir de:

- X^tv con v autovector de XX^t relacionado a un autovalor no nulo de XX^t.
- Vectores en el espacio nulo de X

2.4. Método de la potencia + Deflación

PCA arma la matriz P utilizada en el cambio de variables como una matriz compuesta por los autovectores de $cov(X, X)$. El problema que tenemos ahora es cómo obtener dichos autovectores.

La matriz $cov(X, X)$ es por definición simétrica, es X^tX por un escalar y por lo tanto $cov(X, X)^t = (kX^tX)^t = kX^tX^{tt} = kX^tX = cov(X, X)$. Luego, sus autovectores forman una base ortogonal y sus autovalores son números reales, por lo tanto tienen orden y vale que $\|\lambda_1\| \geq \|\lambda_2\| \geq \cdots \geq \|\lambda_n\|$ donde λ_i es un autovalor de la matriz.

El método que usamos para extraer sus autovalores y autovectores es el Método de la Potencia. Éste es un método iterativo (osea, repite una serie de pasos una determinada cantidad de veces) que nos da el autovalor de mayor módulo y su autovector asociado.

Sea la matriz $B \in R^{n\times n}$ de la que se quiere extraer los autovalores y autovectores, x_0 un vector en R^n y $niter$ la cantidad de iteraciones a realizar. Si B tiene una base de autovectores y se cumple además que $\|\lambda_1\| \geq \|\lambda_2\| \geq \cdots \geq \|\lambda_n\|$ para los autovalores de B, entonces se puede conseguir el autovector buscado usando el siguiente código

$v = x_0$
for $i = 1, \ldots, niter$ **do**
  $v = \frac{Bv}{\|Bv\|_2}$
end
$\lambda = \frac{v^tBv}{v^tv}$ Devuelve λ y v

Como la matriz $cov(X, X)$ cumple las condiciones pedidas por el Método de la Potencia, podemos usarlo para calcular el autovector buscado.

Con este método sólo obtenemos un autovector. Para obtener una cantidad z de autovectores, se usa Deflación. ¿ En qué consiste? Sea una matriz $B \in R^{n\times n}$ con autovalores distintos $\|\lambda_1\| \geq \|\lambda_2\| \geq \cdots \geq \|\lambda_n\|$ y con una base ortonormal de autovectores. Entonces, la matriz $B - \lambda_1 v_1 v_1^t$ tiene autovalores $0, \lambda_2, \ldots, \lambda_n$ con autovectores asociados $v_1, v_2, \ldots, v_n$, donde v_i es el autovector asociado a λ_i de B para $1 \leq i \leq n$.

Entonces lo que vamos hacer para obtener los z autovectores es:

1. Extraer el autovector v_i asociado al autovalor λ_i de mayor módulo de B.
2. Nos guardamos v_i para armar la matriz P.
3. Actualizamos a B como $B - \lambda_i v_i v_i^t$.
4. Repetimos desde el primer paso hasta obtener z autovectores, $z \leq n$.

De esta forma, podemos obtener los autovectores requeridos en PCA.

2.5. ¿Cómo reconocer si una imagen desconocida es una cara?

Hasta ahora vimos los métodos para clasificar una imagen de un rostro al sujeto que le corresponde en nuestro conjunto de imágenes. Nos interesa ahora saber si una imagen desconocida es una cara o no.
Nuestra hipótesis es que la imagen de un rostro no cambia en gran magnitud cuando es proyectado en el espacio donde apreciamos mayor varianza y menor covarianza en los datos. Por otra parte la imagen de algo que no es rostro si cambiará radicalmente en este espacio.
Entonces la idea es ver la distancia entre la imagen original y su proyección en el espacio formado por las autocaras, donde autocaras se refiere a los autovectores que describen mejor la distribución de datos en nuestro conjunto de imágenes de rostros. Si excede un cierto valor, decimos que no es una cara.

Sea X la imagen en cuestión, la normalizamos, de forma tal que $\Phi = X - \Psi$, donde Ψ es la media de la imágenes de rostros de entrenamiento(osea, es el rostro promedio).

La proyección de Φ en el espacio de autocaras se define como:

$$\hat{\Phi} = \sum_{i=1}^{K} w_i u_i$$

donde K es la cantidad de autocaras que elegimos considerar, u_i es una autocara obtenida de nuestro conjunto de imágenes de rostros y $w_i = u_i^t \Phi$

Luego definimos la distancia entre la imagen y su proyección como:

$$e_d = \|\Phi - \hat{\Phi}\|_2$$

Entonces si $e_d < T_d$, siendo T_d un valor elegido por nosotros, la imagen es una cara, de lo contrario, no lo es.

Para definir T_d, buscamos la imagen de la base de entrenamiento cuya distancia a su proyección en el espacio de caras sea mayor a las demás, llamemos a esta distancia e_{max}. Entonces definimos $T_d = e_{max}$.

Anteriormente utilizamos la norma 2 para ver la distancia pero también podríamos utilizar norma 1 o norma infinito .

2.6. Métodos de evaluación del sistema y formas de medición

En este punto ya tenemos el sistema armado, ahora nos interesa analizar su comportamiento ante determinados parámetros de entrada y evaluar su desempeño, pero ¿cómo lo hacemos?.

Como sólo contamos con la base de datos ya etiquetada, podríamos considerar partirla en 2 partes: una actúa como si fueron imágenes aún no reconocidas (base de test) y otra como la base de datos etiquetada (base de entrenamiento).

La pregunta es cómo separarlas, una primera aproximación sería tomando elementos al azar y asignarlo a una base o a la otra. Pero este método trae ciertos problemas. Por ejemplo, podría ocurrir que en la base de test queden los elementos que aportan mucha información a la hora de diferenciar cierto rostro, mientras que en la base de entrenamiento estén los que aportan poca información, lo que causaría que el sistema falle al reconocer imágenes de esa clase, por lo que las mediciones realizadas no reflejarían el comportamiento real del sistema.

Hay varios métodos alternativos al anterior, pero en este trabajo vamos a usar K-fold Cross Validation. Este método se resume en tres pasos:

1. Desordenamos la base de datos.

2. La separamos en K conjuntos (folds) del mismo tamaño.

3. Por cada $i \in [1, \ldots, k]$ tomamos el fold i-ésimo como base de test y el resto de los folds como base de entrenamiento. Entonces realizamos la validación y tomamos mediciones.

Este método minimiza el error del método al azar ya que si en una iteración en la base de test quedan los elementos que aportan mucha información y en la de entrenamiento no, en la siguiente iteración dichos elementos van a estar en la base de entrenamiento. Entonces, la clase que el sistema falla en reconocer en una iteración, será reconocida correctamente en la siguiente.

Ahora que tenemos cómo evaluar el comportamiento del sistema, nos falta ver qué evaluamos del mismo y cómo lo medimos.

Lo primero que se nos ocurre es medir cuantas veces el sistema reconoció correctamente las imágenes en relación a la cantidad total de imágenes analizadas. Esta medida se llama Hitrate. Si bien es importante, puede ser engañosa. Si acertamos el 95 % de las veces parece bueno, pero si en realidad tenemos 2 clases y la mayoría de las imágenes es de una misma clase, puede ocurrir que el sistema sólo sea bueno en reconocer miembros de esa clase. Nos falta entonces medidas más precisas.

Las medidas que se aplican a nuestro trabajo son las siguiente:

Precision: Mide cuántos aciertos relativos tiene un clasificador dentro de una clase particular. Está dado por la fórmula

$$\frac{tp_i}{tp_i + fp_i}$$

donde i es una clase particular, tp_i es la cantidad de veces que se reconoció a una imagen de categoría i correctamente y fp_i es la cantidad de veces que se clasificó a una imagen como de clase i cuando realmente no lo era.

Recall: Mide que tan bueno es el clasificador para identificar correctamente a los miembros de una clase en particular. Se define como

$$\frac{tp_i}{tp_i + fn_i}$$

donde i es una clase particular y n_i es la cantidad de muestras pertenecientes a la clase i pero que fueron identificadas con otra clase.

Tanto Precision como Recall miden por clase, por lo que se puede armar un Precision/Recall general como el promedio del Precision/Recall para cada una de las clases.

F1-Score: Mide el compromiso entre recall y precision de un mismo clasificador. Se define como

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

κ de Cohen:Mide cuanto concuerdan 2 clasificadores sobre un mismo conjunto de datos. Se define como

$$\kappa = \frac{P_o - P_a}{1 - P_a}$$

donde P_o es la probabilidad observada de que ambos concuerden y P_a es la probabilidad aleatoria de que lo hagan.

3. Experimentación

Habiendo ya definido el sistema, nos interesa analizar el tiempo de cómputo y la precisión general del sistema ante distintos parámetros de entrada. Estos parámetros son las imágenes de la base de datos, su composición, la cantidad de folds que tomamos en k-folds crossvalidation, la cantidad de vecinos más cercanos que consideramos en ambas variantes de kNN y el valor k usado en PCA, que representa al número de dimensiones a los cuales reducimos las imágenes.

Contamos con dos conjuntos de imágenes de entrada: uno con las imágenes originales de tamaño 92×112 y otro con un submuestreo de las mismas de tamaño 23×28. Contamos con 10 imágenes por cada uno de los 41 sujetos en cada conjunto, osea, un total de 410 imágenes por conjunto.

Nuestro objetivo final es encontrar una combinación de parámetros de entrada que tenga un buen balance entre tasa de éxito, rostros reconocidos correctamente, y tiempo de cómputo.

En el contexto de este trabajo, como nuestro sistema va a ser utilizado para controlar el ingreso de trabajadores a las instalaciones, tenemos tres objetivos:

- Reconocer y dejar ingresar a los trabajadores.
- Evitar que personas que no sean trabajadores ingresen ya que pueden filtrar información confidencial
- Evitar fallar al reconocer un trabajador, de lo contrario no puede ingresar y se pierde tiempo y dinero.

Para medir qué tan bien cumplimos estos objetivos, vamos a utilizar las siguientes métricas:

- Hitrate: Si tenemos un bajo hitrate quiere decir que no estamos identificando bien a los trabajadores.
- Precision: Mientras mayor sea, mejor identificamos a los trabajadores y hay menos chances de dejar ingresar a una persona no autorizada
- Recall: Mientras mayor sea, menos chances hay de prohibirle la entrada a un trabajador.

Por lo tanto, queremos un sistema que sea eficiente según esas métricas. Para obtenerlas utilizamos K-fold crossvalidation con $K = 5$, consideramos que este valor es apropiado ya que si fuera menor disminuiría los beneficios de este método y si fuera mayor tendríamos problemas ya que el número de imágenes no es muy grande.

Como ya dijimos anteriormente, utilizamos el método de la potencia y como condición de parada elegimos que se detenga cuando el autovector obtenido en esta iteración no difiere del obtenido en la iteración anterior en más de un valor constante dado por C++, *DBL_EPSILON* , o si supera las 10000 iteraciones.

3.1. Experimento 1: Tiempo de cómputo y consumo de memoria

En este experimento queremos analizar el tiempo de cómputo y el consumo de memoria del sistema al procesar ambos conjuntos de imágenes.

Para ver esto, medimos el tiempo promedio que tarda en hacer k-fold crossvalidation.

A priori sabemos que las imágenes del submuestreo son mas chicas en dimensiones que las del conjunto original, y como muchos algoritmos utilizados dependen del tamaño de las imágenes, se nos ocurre que esto va a afectar a lo que queremos ver.

También sabemos que el k de PCA afecta la cantidad de veces que se ejecuta el método de la potencia.

Por lo tanto nuestra hipótesis para ambos métodos de clasificación utilizados es que a medida que aumentemos el k de PCA y a su vez, al incrementar el tamaño de las imágenes, aumentará el tiempo de cómputo.

En cuanto al consumo de memoria, vamos a analizar cuánta se necesita cuando en vez de hallar los autovectores de X^tX buscamos los de XX^t, y cuando no hacemos esto. Nuestra hipótesis sobre esto es que al ser menor la dimensíon en XX^t las medidas van a ser menores con respecto a X^tX.

3.2. Experimento 2: kNN por votación vs kNN promediado por distancia

En este experimento vamos a ver cómo se comparan las dos variantes usadas para clasificar, en base a las medidas de precisión elegidas anteriormente, para ambos conjuntos de imágenes. La cantidad de vecinos más cercanos a considerar y el valor k de PCA van a ser los parámetros de entrada a variar.

Nuestro objetivo es hallar la combinación de parámetros que genere las mejores medidas de precisión.

3.3. Experimento 3: Cantidad de imágenes.

En este experimento vamos a ver cómo se comporta el sistema utilizando los parámetros obtenidos en el experimento anterior para versiones alteradas de los conjuntos de imágenes. Nos interesa esto para ver si usando el subconjunto podemos obtener una precisión similar a la de usar el conjunto entero, reduciendo el consumo de memoria.

Para esto, vamos tomar un subconjunto de las imágenes en el que haya sólo 5 muestras por sujeto. Vamos a ver su tiempo de cómputo y medidas de precisión usando K-fold crossvalidation sobre ese subconjunto y lo comparamos con su equivalente que use la conjunto entero. Además tomamos las mediciones en el caso en el que la base de entrenamiento es parte del subconjunto pero las imágenes de test son las del conjunto. Probamos esto para ambos tamaños de imágenes, el de las originales y el de las reducidas(submuestreada).

3.4. Experimento 4: Detección de rostros

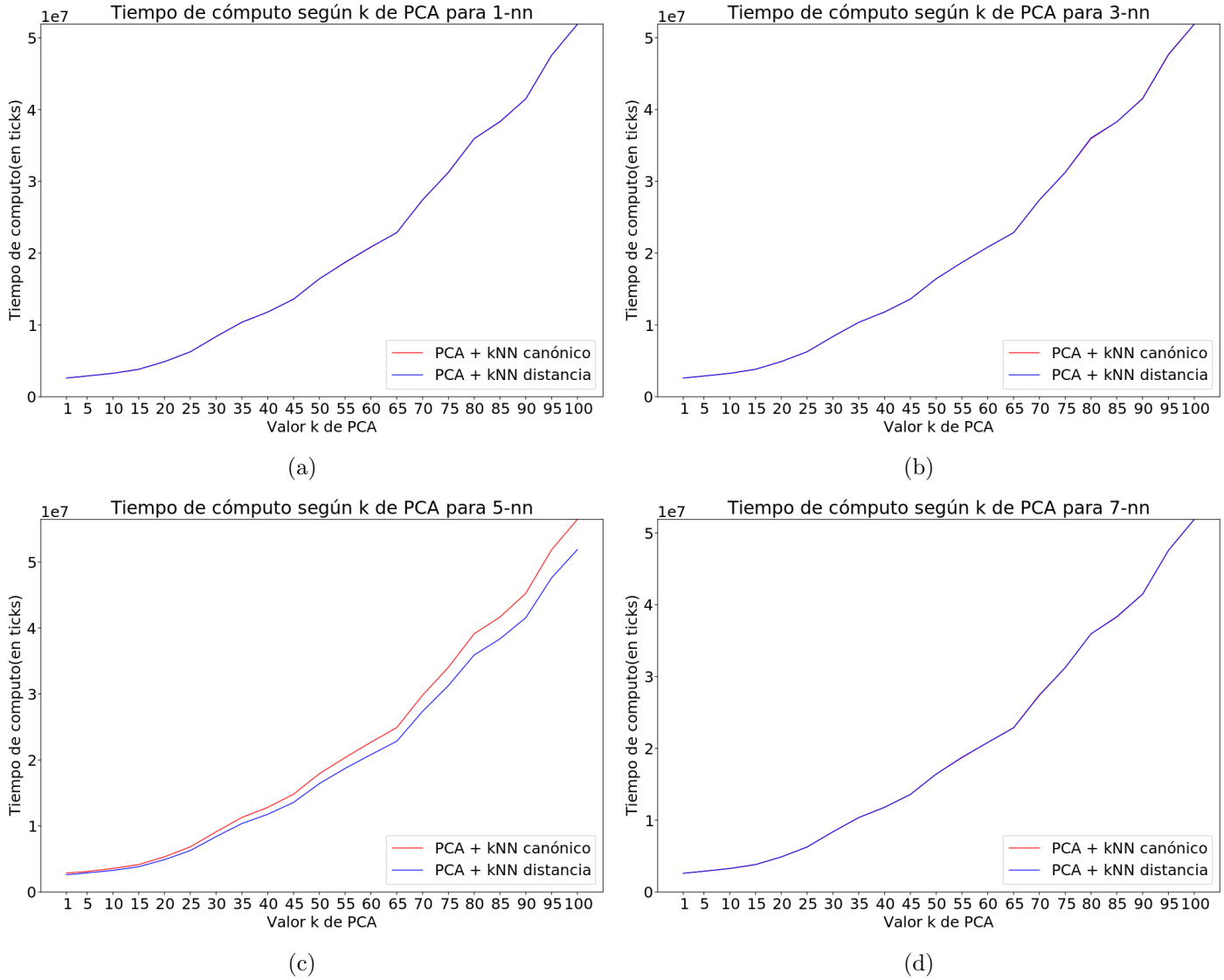
En este experimento queremos analizar si el sistema puede saber si una imagen desconocida es un rostro o no. Para esto, armamos un conjunto de imágenes variado y vemos si el sistema los identifica correctamente, usando como base de entrenamiento todas las imágenes disponibles.

4. Resultados

4.1. Experimento 1

Mostramos a continuación los resultados del experimento de tiempo de cómputo usando cada variante de Knn. Se probó con valores de k de PCA empezando desde 1, y luego desde 5 a 100 en intervalos de 5. También se probo con los siguientes valores de kNN:1,3,5,7. Elegimos números impares de k de kNN para que haya menos chances de empate y no tomamos valores superiores ya que , en las pruebas que hicimos, estos no aportaban información nueva.

Al realizar el experimento, vimos que probando con valores de k de PCA mayores a 100 las medidas no variaban mucho, por lo que elegimos mostrar los valores hasta 100.



(a) (b)

(c) (d)

Figura 1: Gráficos de tiempo para imágenes de tamaño 92x112

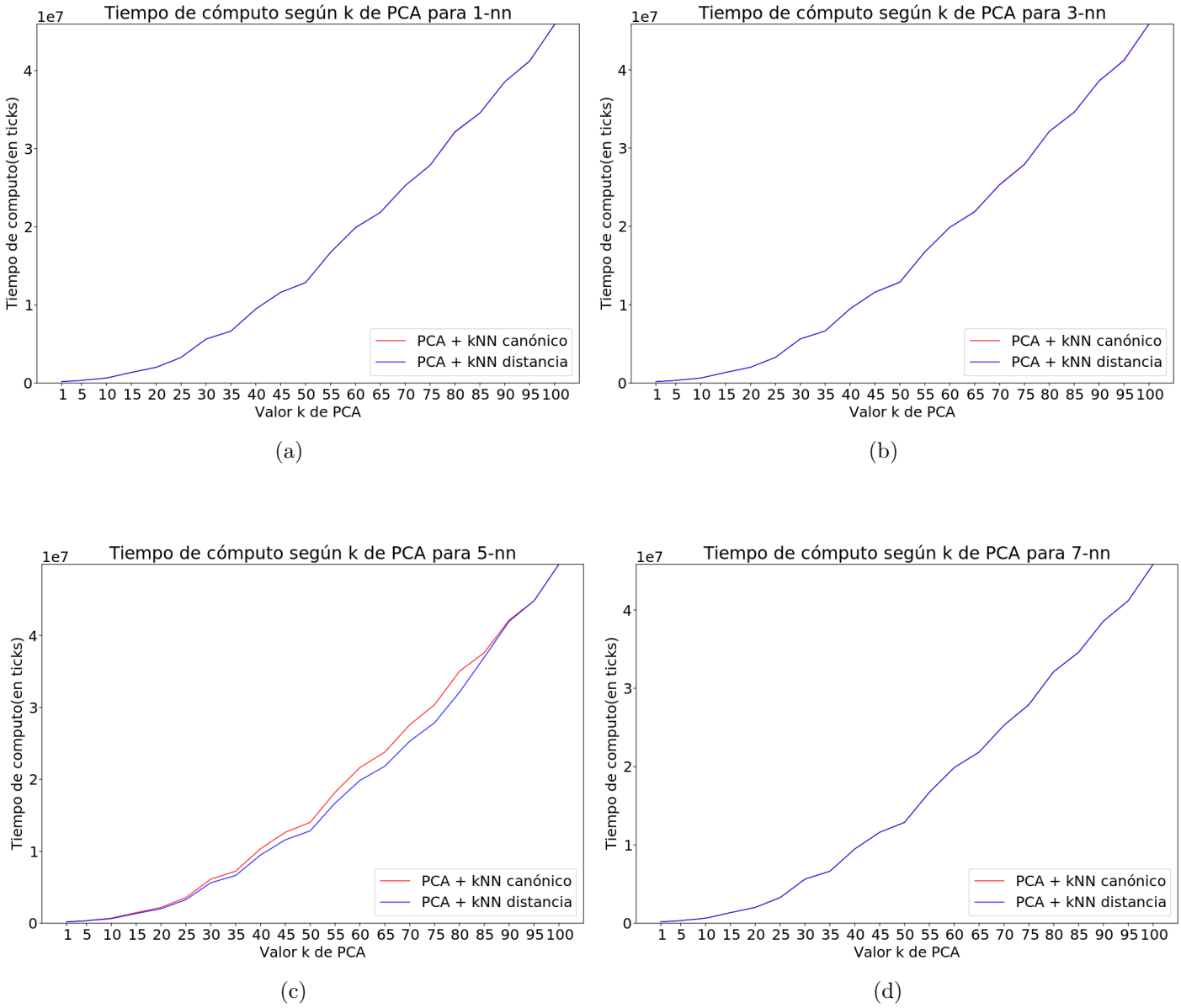


Figura 2: Gráficos de tiempo para imágenes de tamaño 23x28

Observamos en estos gráficos que a medida que aumenta el valor k de PCA, aumenta el tiempo de cómputo y que el tamaño de las imágenes también influye, las de mayor tamaño tardan más. Notamos además que la variante de kNN utilizada por lo general no influye en el tiempo de cómputo.

A continuación mostramos los resultados del experimento de memoria. Por un lado medimos el uso de memoria del sistema(en Mb) calculando los autovectores de XX^t y por el otro el uso cuando calculamos los autovectores de X^tX. Para simplificar el texto de los gráficos, denominamos al primer caso *con truco* y al segundo *sin truco*. Sólo se muestra un gráfico para el caso de imágenes de tamaño reducido ya que, para el caso de imágenes de tamaño original, la diferencia era tan grande que el gráfico propuesto no era claro. En su lugar, se presenta una tabla con las medidas de resumen para ese caso particular.

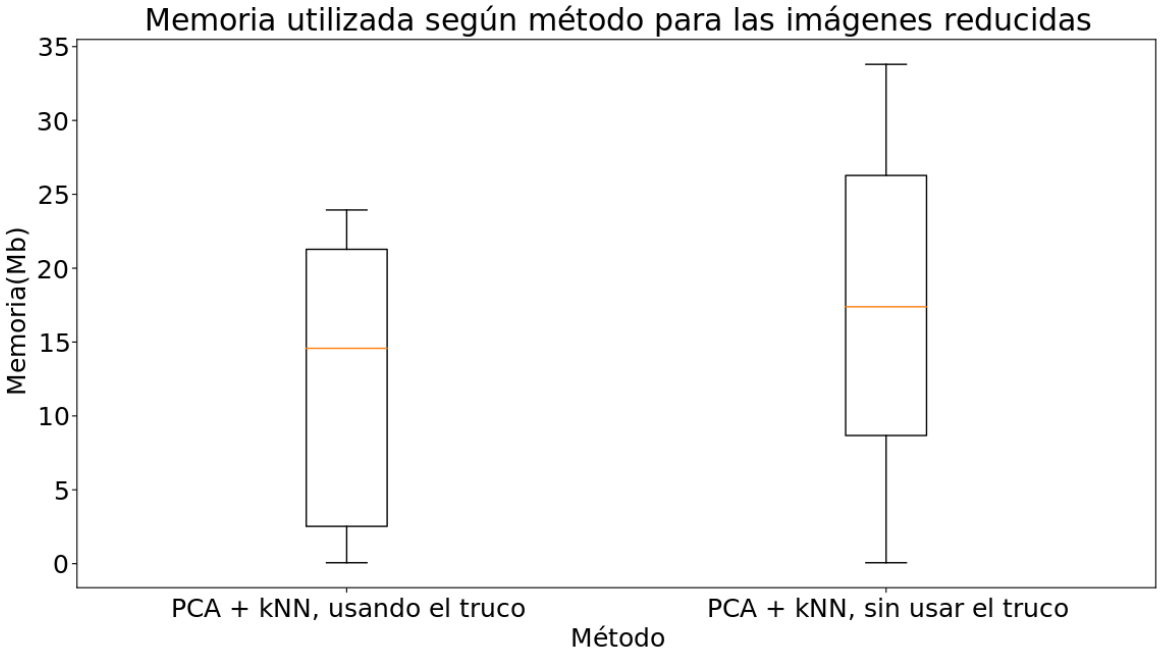


Figura 3: Consumo de memoria(en Mb) según método utilizado, para el conjunto de imágenes de tamaño reducido.

Método	Con Truco	Sin Truco
Media	33.72	1807.42
Desvío estándar	40.06	415.13
Mínimo	0.06	908.53
Primer cuartil	2.26	1721.26
Mediana	14.37	1723.75
Tercer cuartil	72.59	1727.36
Máximo	128.33	3354.43

Figura 4: Consumo de memoria(en Mb) según método utilizado, para el conjunto de imágenes de tamaño original.

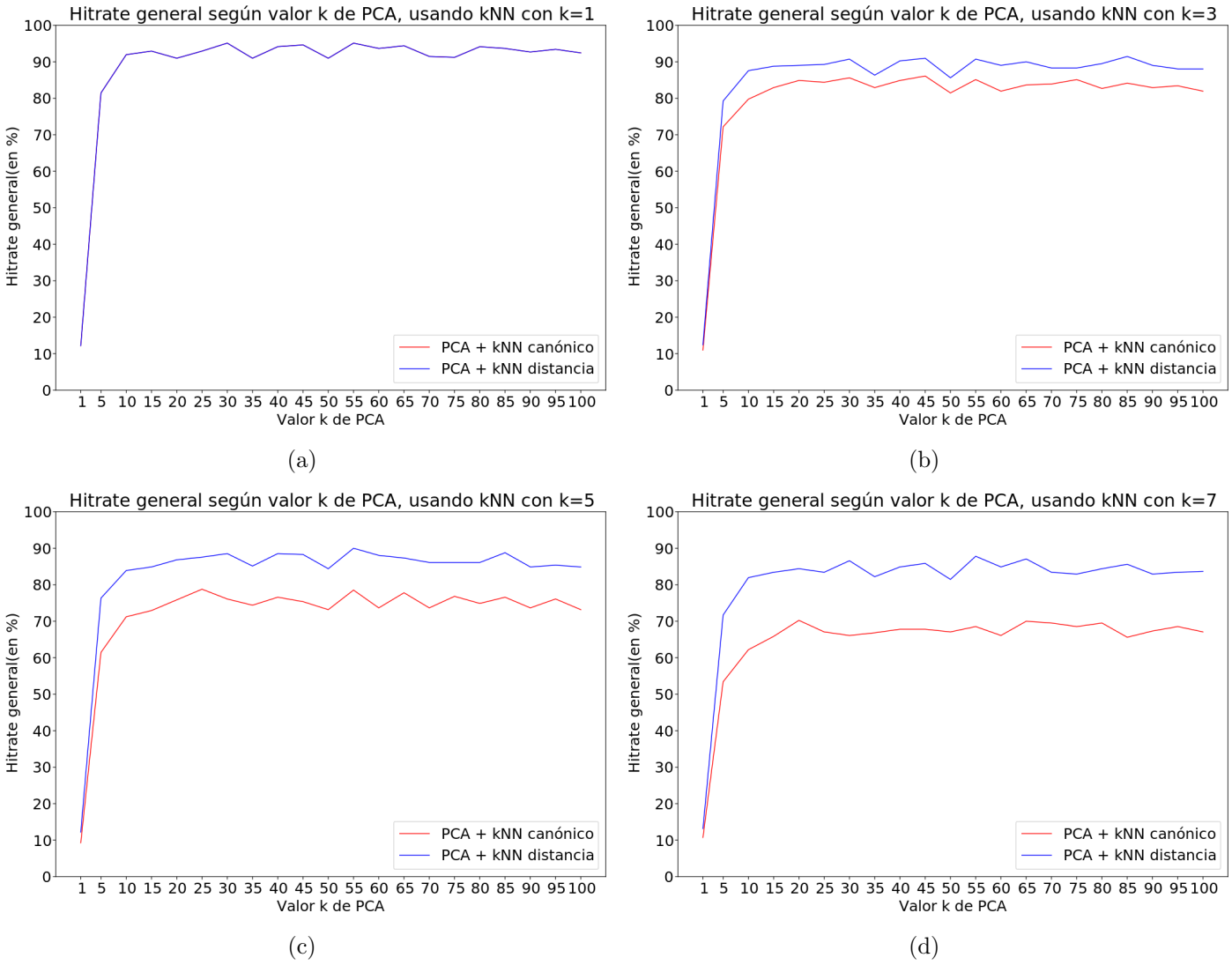
En el primer gráfico se puede observar que, cuando el tamaño de las imágenes no es muy grande comparado con la cantidad de muestras, el consumo de memoria es similar utilizando ambos métodos. En este caso las medidas sin usar el truco resultan mayores ya que el tamaño de la imágenes es $23 \times 28 = 644$, el cual es mayor que la cantidad de muestras en total, 410.

En la tabla, vemos que en las medidas de resumen entre ambos métodos hay diferencias de más de un orden de magnitud. El caso más relevante es que el máximo usando el truco es alrededor de 7 veces menor que el mínimo sin usar el truco. Se ve entonces que al usar las imágenes de tamaño original, consumimos mucha más memoria cuando no usamos el truco que cuando sí. Creemos que se debe a que el tamaño de las imágenes de este caso es $92 \times 112 = 10304$.

Concluyendo, cuando el tamaño de las imágenes no es un número mucho más grande que la cantidad de muestras, se pueden utilizar cualquiera de los dos métodos. Pero cuando es mucho más grande es mucho más eficiente calcular los autovectores de XX^t.

4.2. Experimento 2

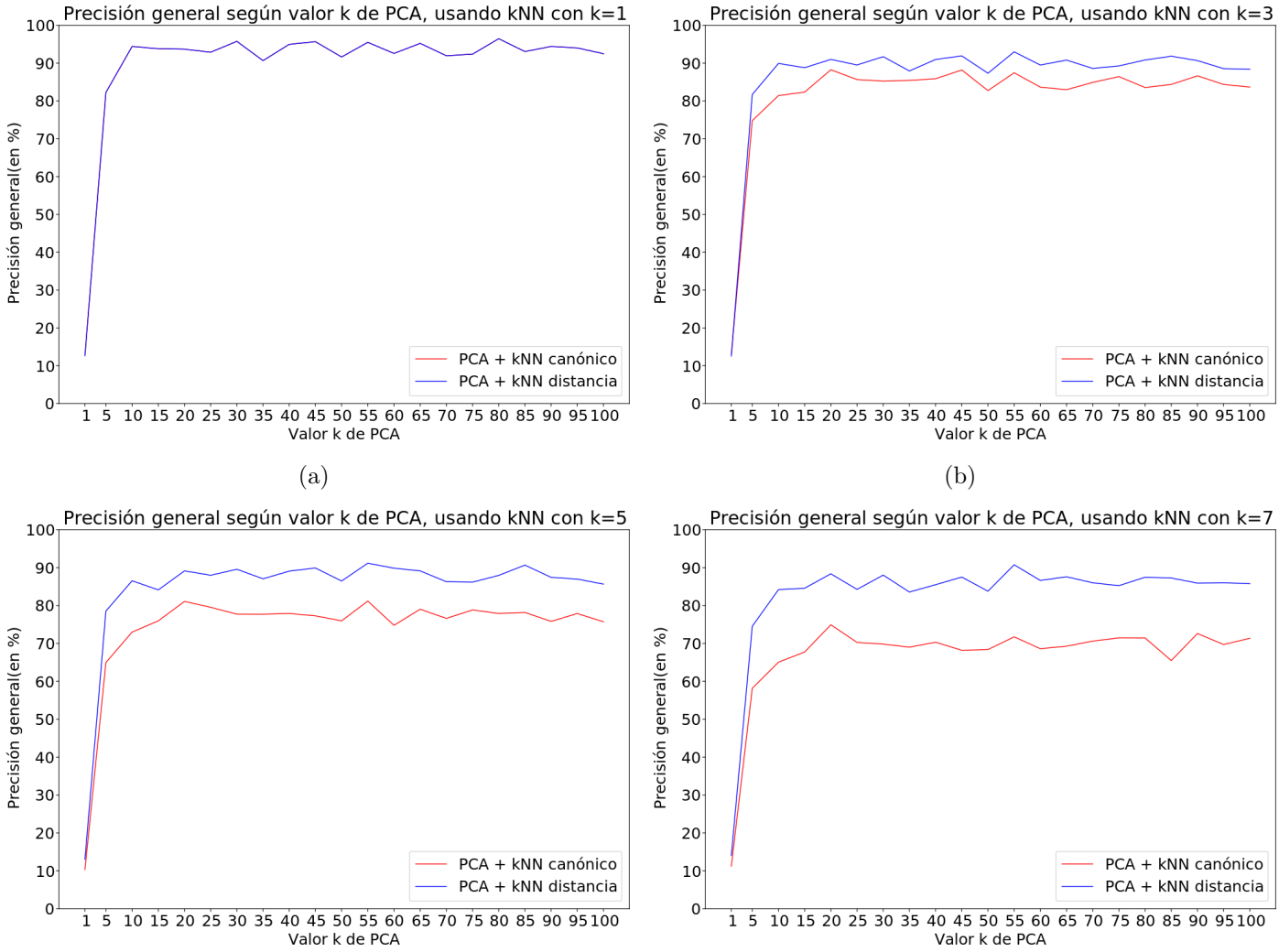
Mostramos a continuación los resultados del experimento 2, separados según métrica y tamaño de imagen.



(a) (b)

(c) (d)

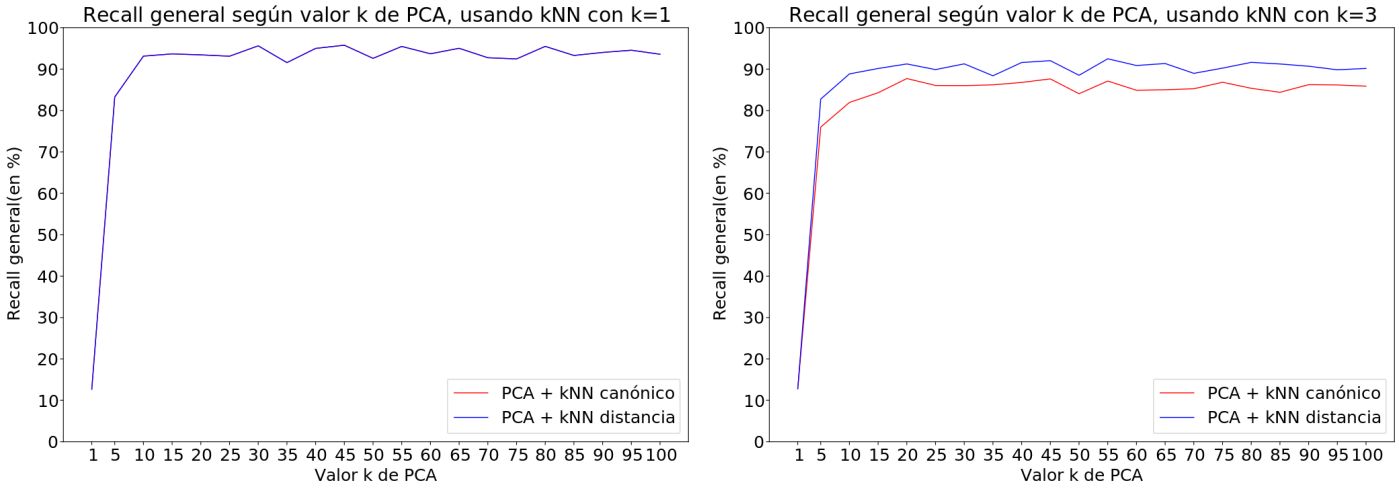
Figura 5: Gráficos de precisión - Hit Rate - para imágenes de tamaño 92x112



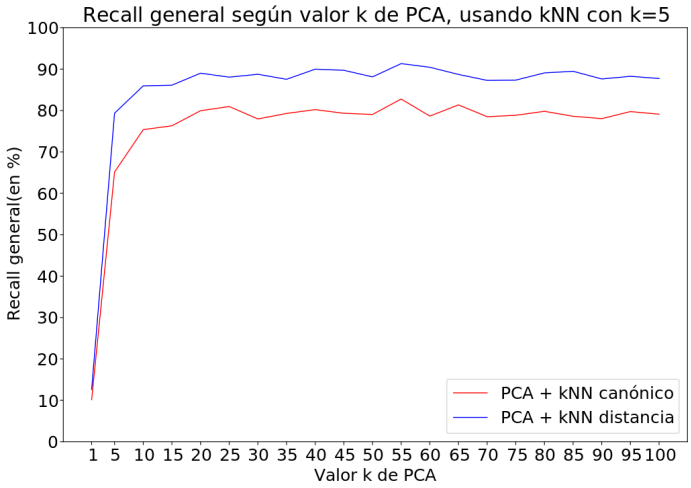
(a) (b)

(c) (d)

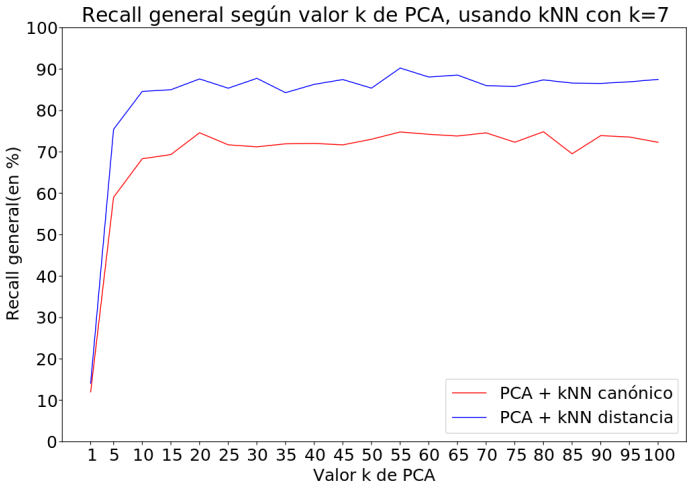
Figura 6: Gráficos de precisión - Precision - para imágenes de tamaño 92x112



(a) (b)

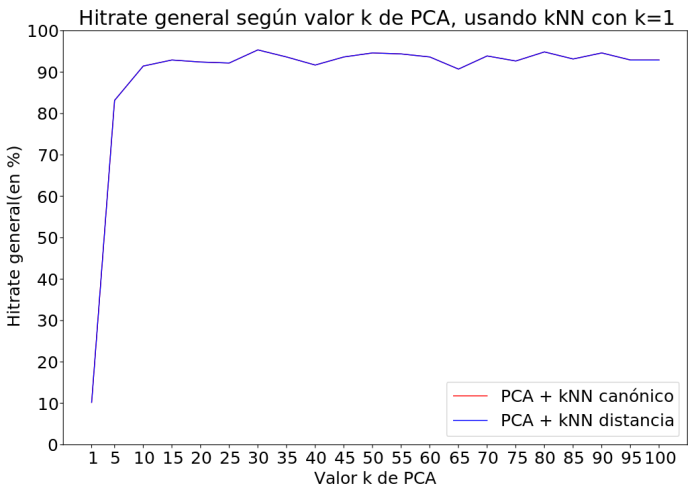


(c)

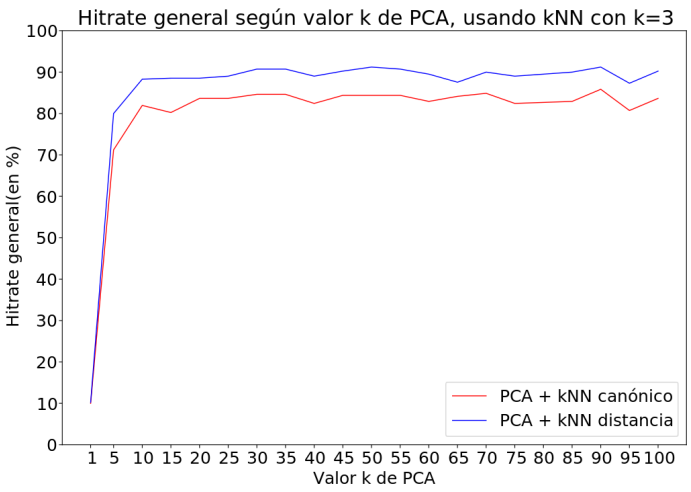


(d)

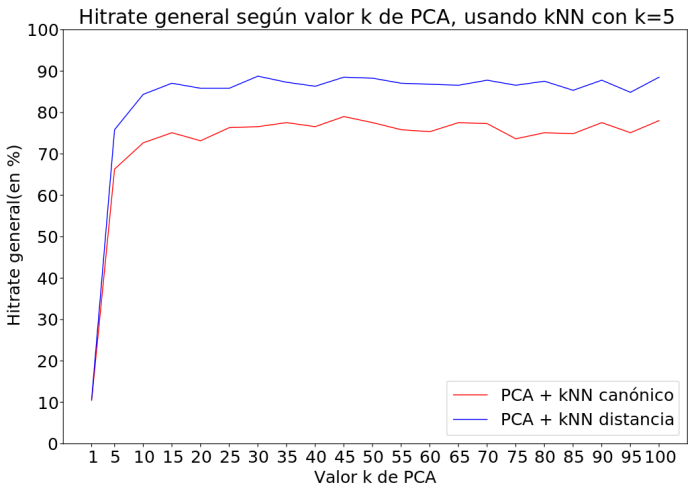
Figura 7: Gráficos de precisión - Recall - para imágenes de tamaño 92x112



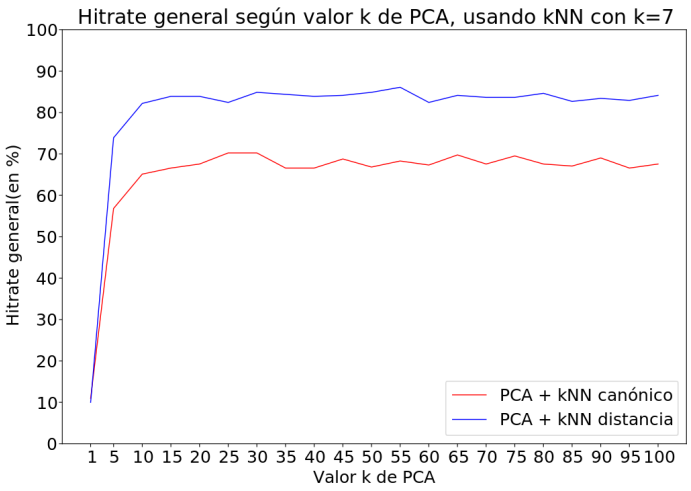
(a)



(b)

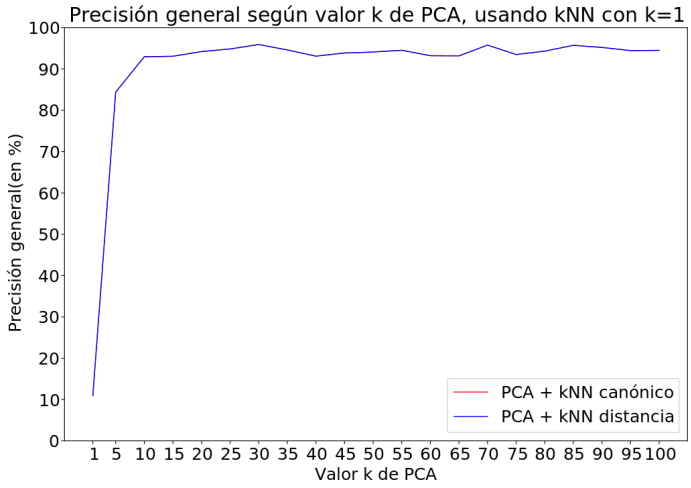


(c)

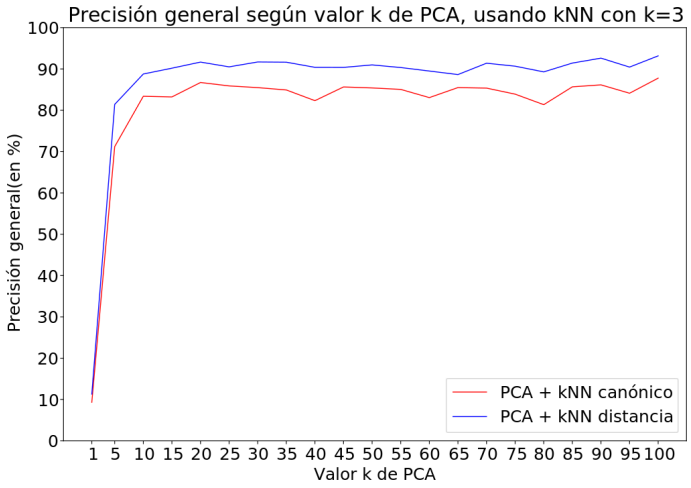


(d)

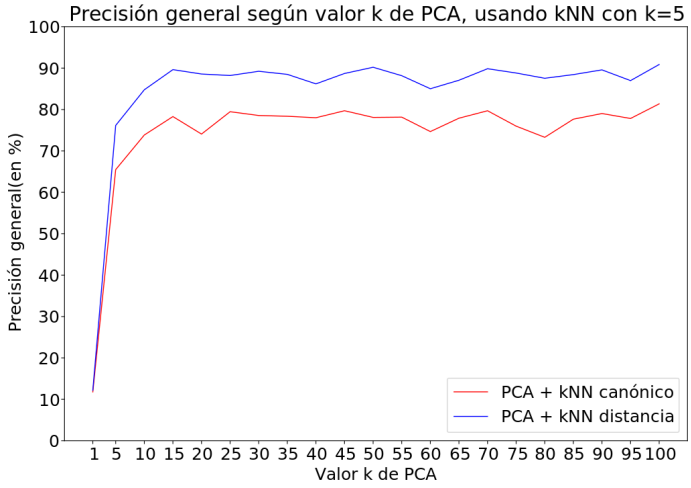
Figura 8: Gráficos de precisión - Hit Rate - para imágenes de tamaño 23x28



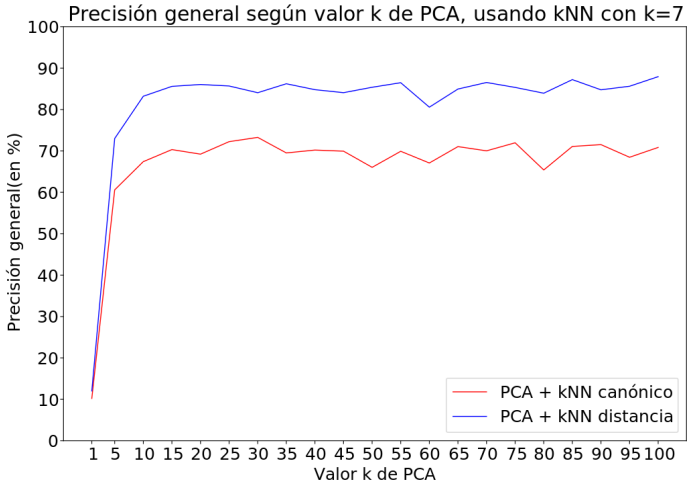
(a)



(b)

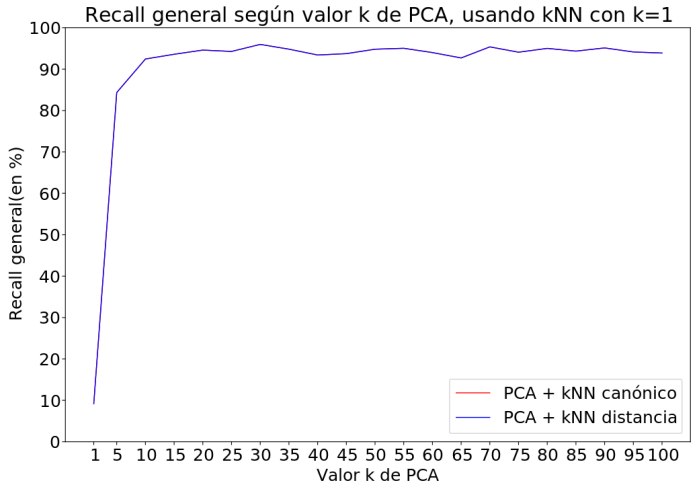


(c)

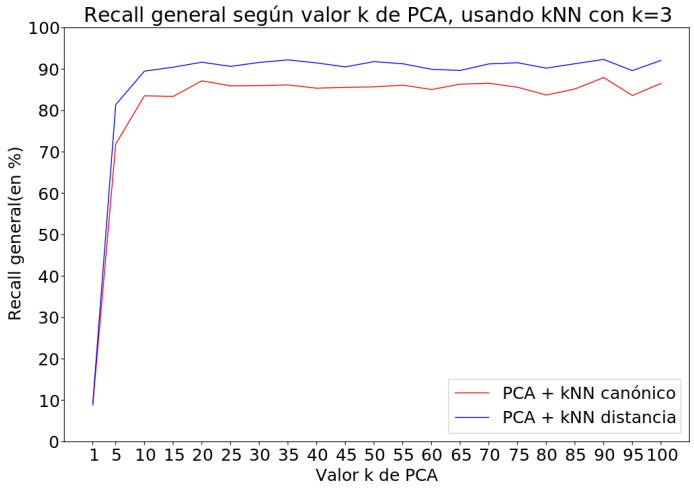


(d)

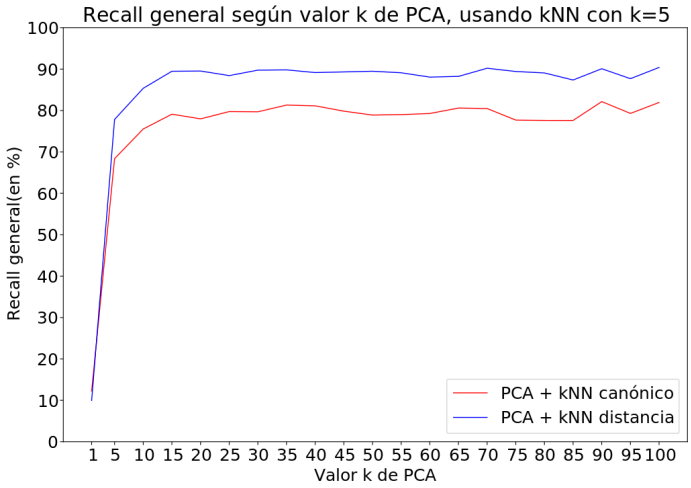
Figura 9: Gráficos de precisión - Precision - para imágenes de tamaño 23x28



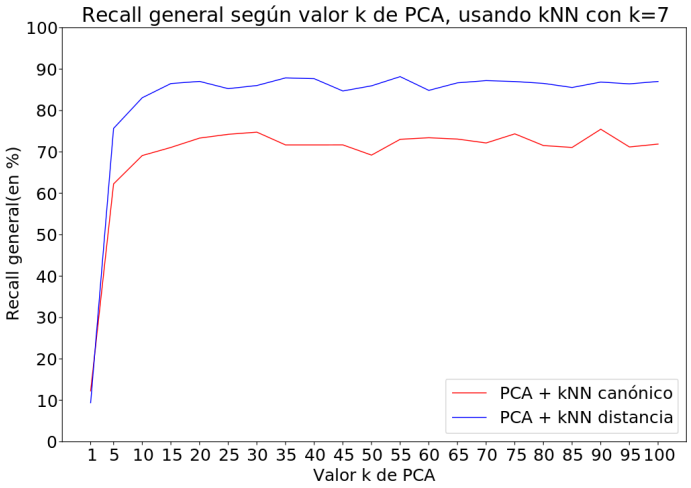
(a)



(b)



(c)



(d)

Figura 10: Gráficos de precisión - Recall - para imágenes de tamaño 23x28

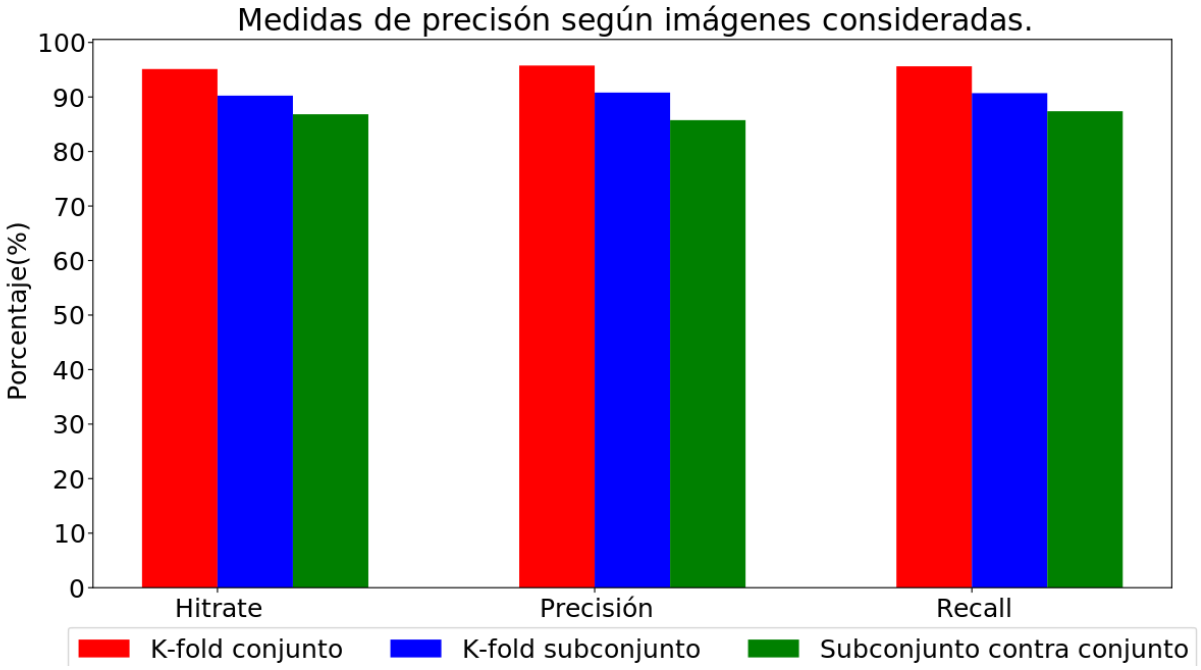
Observamos en los gráficos que, para todas las métricas, cuando k de kNN es 1, ambas variantes de kNN se comportan de la misma manera. Para los demás valores de k, notamos que la variante kNN distancia tiene mejores medidas que kNN canónica y esta diferencia se acentúa a medida que aumenta k de kNN. Además vemos que los valores de las métricas aumentan junto al k de PCA hasta que éste alcanza cierto valor, a partir del cual tienden a oscilar.

Notamos además que las mejores medidas de precisión se obtienen cuando k de kNN es 1, para ambos tamaños de imágenes. En cuanto al mejor valor k de PCA, notamos que con $k = 30$ y $k = 80$ por lo general se obtienen las mejores medidas. Sin embargo, como vimos en el experimento 1, el tiempo de cómputo cuando $k = 80$ es mayor que cuando $k = 30$, y como la diferencia entre las medidas de ambos k es despreciable, elegimos $k = 30$ como el valor de k que mejores resultados nos dio.

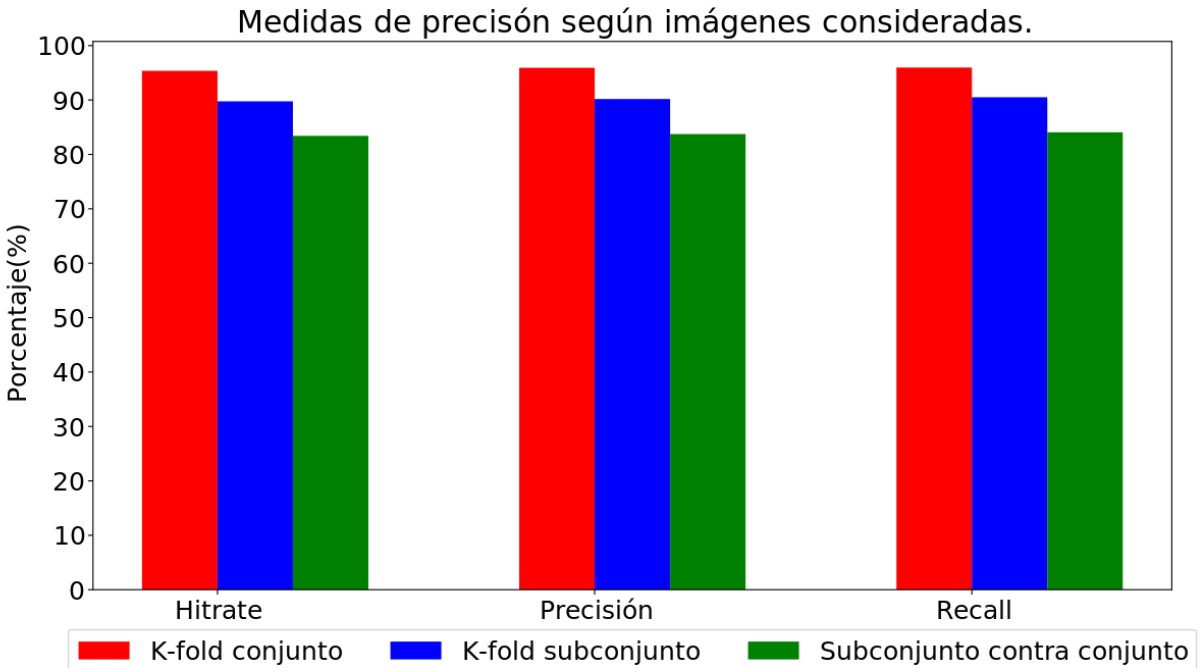
En conclusión, hallamos que la mejor combinación de parámetros es k de PCA igual a 30 y k de kNN igual a 1 (usando cualquiera de las dos variantes de kNN) ya que obtenemos las mejores medidas de precisión y el k de PCA no es muy alto.

4.3. Experimento 3

Mostramos a continuación los resultados del experimento 3. En este contexto, **K-fold conjunto** significa que se realizo K-fold crossvalidation sobre el conjunto de imágenes sin alterar (igual al experimento 2, K-fold), **K-fold subconjunto** que se realizó crossvalidation sólo sobre el subconjunto de imágenes elegido y **Subconjunto contra Conjunto** es el caso donde se usan las imágenes del subconjunto elegido para entrenar pero se las testea contra imágenes del conjunto entero.



(a)



(b)

Figura 11: Medidas de precisión según el conjunto de imágenes considerado. En (a) utilizamos como conjunto a las imágenes de tamaño original, en (b) a las de tamaño reducido.

Se observa en los gráficos que, sin importar que tipos de imágenes usamos, al entrenar al sistema sólo con las primeras 5 imágenes de cada sujeto obtenemos medidas de precisión peores que utilizando en conjunto de imágenes entero. Además vemos que en el caso de crossvalidation sobre el subconjunto elegido las métricas dan resultados un poco más bajos que cuando utilizamos todo el subconjunto de imágenes.

Si bien no son mucho peores, a lo sumo un 10 % peores, en el contexto de este trabajo, que es el de

un sistema de reconocimiento de rostros para uso de seguridad, consideramos que el uso de una menor cantidad de imágenes de entrenamiento (y por ende, de memoria) no justifica la pérdida de precisión.

Concluimos entonces que tener una menor cantidad de imágenes de entrenamiento da métricas sólo un poco peores a cambio de una menor cantidad de memoria utilizada y un menor tiempo de cómputo. Sin embargo, en el contexto en el que se utilizará el sistema, no lo consideramos aceptable.

4.4. Experimento 4 : Detección de rostros

Se evaluaron 28 imágenes en donde sólo 16 pertenecían a rostros y las demás no. Cada imagen tiene dos versiones: una versión de 92×112 pixeles y otra de 23×28. Usamos un valor que consideramos suficiente para el parámetro k de PCA igual a 30, donde este parámetro determina la cantidad de autovectores de PCA que mejor describen la distribución de datos. Obtuvimos este valor k a partir de los experimentos anteriores y lo evaluamos en este experimento.

(a) Imágenes numeradas del 1 al 5.

(b) Imágenes numeradas del 6 al 10.

(c) Imágenes numeradas del 11 al 15.

Figura 12: Imágenes utilizadas en el experimento, numeradas de izquierda a derecha.

(a) Imágenes numeradas del 16 al 20.

(b) Imágenes numeradas del 21 al 25.

(c) Imágenes numeradas del 26 al 28.

Figura 13: Imágenes utilizadas en el experimento, numeradas de izquierda a derecha.

El resultado que se muestra en la siguiente tabla es la predicción que hace el sistema clasificador de rostros al pedirle que detecte si una imagen es rostro o no.

N° de imagen	¿Es cara?	¿Reconocido como cara?	
		23x28	92x112
1 (*)	No	Si	Si
2	No	Si	No
3	No	Si	Si
4 (*)	No	No	Si
5 (*)	No	Si	Si
6 (*)	No	Si	Si
7 (*)	No	Si	Si
8 (*)	No	Si	Si
9	No	No	No
10	No	Si	Si
11 (*)	No	No	No
12	Si	Si	Si
13	Si	Si	Si
14	No	No	No
15	Si	Si	Si
16	Si	Si	Si
17	Si	No	Si
18	Si	Si	Si
19	Si	Si	Si
20	Si	Si	Si
21	Si	Si	Si
22	Si	Si	Si
23	Si	No	No
24	Si	No	No
25	Si	Si	Si
26	Si	Si	Si
27	Si	Si	Si
28	Si	Si	Si

Cuadro 1: Resultados de la evaluación de detección de rostros. (*) refiere a imágenes que contienen personas.

La tasa de efectividad observada con imágenes de 23×28 pixeles es $\frac{17}{28} \approx 0.6$, mientras que para imágenes de tamaño 92×112 es $\frac{18}{28} \approx 0.65$.

Muchas de las imágenes etiquetadas como no rostro contenían personas en donde se podía apreciar su rostro y es en estas donde se produce una alta tasa de falsos positivos. Por otra parte las imágenes 3 y 10, que no contienen personas, tienen una distancia al espacio de autocaras, muy por debajo del umbral que elegimos. Es decir que se aceptan como rostros cuando quisiéramos que queden por arriba del umbral (que sean rechazadas). Nos quedamos con este valor de umbral porque es la máxima distancia al espacio de autocaras de un rostro perteneciente a nuestras imágenes de entrenamiento y por lo tanto permite que todos los rostros de las imágenes de entrenamiento sean detectados. También se destaca que las imágenes 23 y 24, que no fueron detectadas, tienen una distancia al espacio de autocaras apenas arriba del umbral.

Restringidos al umbral definido anteriormente, quisimos saber si se mejoraba la tasa de éxito al incrementar el valor del k de PCA y lo intentamos para valores de 30, 40, 50 y 80. Para este experimento nos

quedamos con las muestras de tamaño 92×112 porque es la de más alta tasa de efectividad. Se obtuvieron las mismas predicciones que las mostradas en la tabla 1 pero las distancias variaron, no mucho. Elegimos una imagen y armamos una tabla mostrando el valor de su distancia al espacio de autocaras y el umbral, obtenidos a partir de variar el k.

k de PCA	Distancia	Umbral
30	542**39**15511.341409	1120**54**84945.912378
40	542**45**24861.654340	1120**55**63276.557882
50	542**46**50744.378531	1120**56**00016.996103
80	542**47**42211.286021	1120**56**27140.469442

Cuadro 2: Resultados para la imagen 1.

Se muestra resaltado las principales variaciones entre un resultado y otro. No se aprecia gran diferencia en los distintos valores de umbral pero si hay un salto en la distancia al variar el k de 30 a 40. Sin embargo, la distancia apenas varía al pasar el k de 50 a 80. Recordemos que el resultado para esta imagen cae por debajo del umbral y se acepta como rostro cuando en realidad no lo es. Luego parece que no lograremos extender suficientemente la distancia para caer fuera del umbral al incrementar el k y, de esa manera, rechazar la imagen que no es rostro.

Por la tasa de efectividad observada de tabla 1, la de resultados de detección de rostros, se deduce que el sistema tiene más éxito en detectar rostros cuando las imágenes tienen resolución de 92×112 pixeles que cuando tienen tamaño 23×28, aunque no demasiado. El porcentaje de éxito que obtuvimos en el mejor de los casos fue de un 65 %. Suponemos que se podría incrementar este porcentaje ampliando la cantidad de imágenes de entrenamiento. Por otra parte a partir de lo observado en tabla 2 concluimos que el k elegido, $k = 30$, es aceptable para la tarea de detección de rostros. Es decir que incrementando el k no mejoraremos la tasa de efectividad. Tal vez las imágenes no rostro con distancia al espacio de autocaras apenas por debajo del umbral queden excluidas si se incrementa suficientemente el k de PCA, es decir para un $k \geq 80$. Esto se deduce a partir de lo observado en la tabla de distancias (tabla 2) donde se aprecia mayor incremento de la distancia que del umbral al crecer k. Luego a mayor k podría superar la distancia al umbral y, en particular, la imagen 1 quedaría rechazada como rostro que es lo que buscamos. Queda por responder esto en un futuro experimento.

5. Conclusiones

En este trabajo vimos cómo utilizando métodos de álgebra lineal y de clasificación se puede armar un sistema de reconocimiento de rostros. Mediante experimentos sobre los parámetros de entrada, analizamos las características y el comportamiento del sistema en diferentes contextos.

El sistema cumple satisfactoriamente su propósito de clasificar imágenes de caras en sus correspondientes categorías con una alta precisión, en especial con la combinación de parámetros hallada. Sin embargo, a la hora de determinar si una imagen desconocida es una cara o no, no tiene la precisión necesaria, por lo que no se recomienda utilizar esta funcionalidad en el contexto de seguridad dado.

Además, vimos que utilizar un subconjunto de imágenes para entrenar el sistema no provee la precisión deseada. Este sistema será parte de un sistema de seguridad y por esto debe ser lo más preciso posible.

Luego de haber experimentado y ver sus resultados, nos surgen mas incertidumbres con respecto a ciertos aspectos. Algunos experimentos que se podrían realizar serían:

- Dado que en el experimento 4 las imágenes donde tenían cuerpo y rostro no las reconoció como tal, sería interesante encontrar un sistema que pueda aislar el rostro del cuerpo y luego identificarlos.
- Hallar un método alternativo más preciso para el reconocimiento de caras.
- En los experimentos 3 y 4 se podrían variar más los parámetros.
- Para reconocer un rostro, dentro de los cálculos a realizar, obtenemos la distancia entre la imagen y su proyección, en la cual utilizamos la norma 2. Un experimento posible sería ver que resultados obtenemos al cambiar esta por la norma uno y la norma infinito.

Referencias

[1] Matthew Turk, Alex Pentland, *Eigenfaces for Recognition*, Journal of Cognitive Neuroscience, Massachusetts Institute of Technology, 1991.