



**DEPARTAMENTO  
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

# Trabajo Práctico 1

“Si nos organizamos aprobamos todos...”

Metodos numericos  
Primer Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Gastón Zanitti	058/10	gzanitti@gmail.com
Ricardo Colombo	156/08	ricardogcolombo@gmail.com
Dan Zajdband	144/10	Dan.zajdband@gmail.com
Franco Negri	893/13	franconegri200@gmail.com
Alejandro Albertini	924/12	ale.dc@hotmail.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

<b>1. Introduccion</b>	<b>3</b>
<b>2. Desarrollo</b>	<b>4</b>
2.1. Algoritmo de kNN . . . . .	4
2.2. Optimizacion mediante Análisis de componentes principales . . . . .	4
2.3. Cross-validation . . . . .	4
<b>3. Análisis</b>	<b>5</b>
3.1. KNN . . . . .	5
3.1.1. Cantidad de vecinos . . . . .	5
3.1.2. Mejoras en kNN . . . . .	5
3.2. PCA . . . . .	5
3.2.1. Lambda inicial . . . . .	8
3.2.2. Algo mas que no me acuerde . . . . .	8
<b>4. Resultados</b>	<b>9</b>
4.1. . . . .	9
<b>5. Conclusiones</b>	<b>10</b>
<b>6. Apendice</b>	<b>11</b>

# 1. Introduccion

INTRO!!

## 2. Desarrollo

### 2.1. Algoritmo de kNN

Como primera aproximación para la resolución del problema de OCR, implementamos el algoritmo de *Kvecinosmascercanos* (o *kNN* por sus siglas en ingles). Este algoritmo consiste basicamente en la idea de que entradas parecidas, a partir de una metrica definida en la implementacion (que para este caso podria variar desde, por ejemplo, desde la cantidad de puntos arriba de cierto valor para contabilizar la cantidad de valores *negros* y *blancos* hasta la norma de cada vector digito) presentaran características definidas y al ser ubicadas sobre un plano se agruparan de acuerdo a estas. Luego, para clasificar un nuevo objeto, basta con ubicarlo dentro de este plano y promediar la etiqueta de los  $k$  vecinos mas cercanos para obtener una clasificación. Sin embargo, y mas alla de las mejoras que puedan realizarse sobre los datos en crudo, este algoritmo es muy sensible a la variabilidad de los datos. Un conjunto de datos con un poco de dispersion entre las distintas clases de clasificacion, hace empeorar rapidamente los resultados. Analizaremos en la segunda etapa del trabajo practico, una forma de solucionar este inconveniente mediante el análisis de componentes principales.

//ACA UN PSEUDOCODIGO?

### 2.2. Optimizacion mediante Análisis de componentes principales

En esta segunda parte, utilizaremos una tecnica conocida como *analisisdecomponentesprincipales* como una forma de optimizar los resultados de la primera etapa. El analisis de componentes principales (o PCA) consiste basicamente en conseguir una descomposicion de los datos en sus matrices ortogonales de valores principales para obtener una transformacion lineal que resuma la información mas relevante de cada imagen, descartando aquellos valores que no aportan datos y resultan redundantes. Para realizar este procedimiento tomamos la matriz de covarianza como una forma de expresar la relacion de dependencia intrinseca entre cada variable. A partir de esta información y mediante el metodo de la potencia, obtenemos un vector  $P$  que, al multiplicarlo por nuestros valores originales, realiza el cambio de coordenadas minimizando la covarianza.

//ACA UN PSEUDOCODIGO?

### 2.3. Cross-validation

### 3. Análisis

#### 3.1. KNN

Análisis de KNN

Ahora vamos a analizar el algoritmo KNN (k vecinos mas cercanos) para distintos valores de k, fijando un valor de  $\alpha$ , para ver como varia la efectividad (cantidad de aciertos) del algoritmo. Vamos a probar el algoritmo KNN para los siguientes valores:

$\alpha$ : 10 y k: 5, 50, 250.

Tambien, vamos a ejecutar para los valores anteriores, 5 pruebas iguales para cada uno, ya que el algoritmo varia la cantidad de aciertos segun que base de datos analice y no siempre da lo mismo.

Este algoritmo lo que hace es por cada imagen que queremos averiguar a que digito pertenece, vectorizarla, restarla a cada uno de los vectores imagen y calcular la norma2 para saber en cuanto difieren con cada una de las imagenes. Todos esos resultados los vamos metiendo en una cola de prioridad que los ordena de menor a mayor, segun la diferencias entre la imagen que quiero averiguar a que digito pertenece y todas las imagenes de la base de datos.

Luego lo que hacemos es agarrar los k primeros de la cola de prioridad y verificar a que digito se corresponden para luego saber cual es el digito que recibio mas 'votos' y ver si se produjo un acierto o no. Por lo tanto, a mayor cantidad de vecinos (o sea, k) menor va a ser la cantidad de aciertos, ya que voy a empezar a mirar los elementos de menor prioridad de la cola, eso significa, que voy a empezar a contar las imagenes que mas difieren y eso me puede hacer que las chances de acertar el digito correcto disminuyan.

El algoritmo KNN es muy efectivo ya que tiene aproximadamente entre 85 % y 90 % de aciertos. Pero su 'deficit' es que es muy lento a comparacion del algoritmo PCA.

##### 3.1.1. Cantidad de vecinos

Mover el k y presentar un analisis

##### 3.1.2. Mejoras en kNN

Cortar el dataset en x valor y comparar para ver si mejora

#### 3.2. PCA

Análisis de PCA Ahora vamos a analizar el algoritmo del PCA. Vamos a probar el algoritmo para distintas medidas de k y  $\alpha$ , que van a ser:

k: cantidad de vecinos a considerar en el algoritmo kNN.

$\alpha$ : a la cantidad de componentes principales a tomar.

Vamos a probar el algoritmo para los siguientes valores:

$\alpha$ : 10 y k: 5, 20.

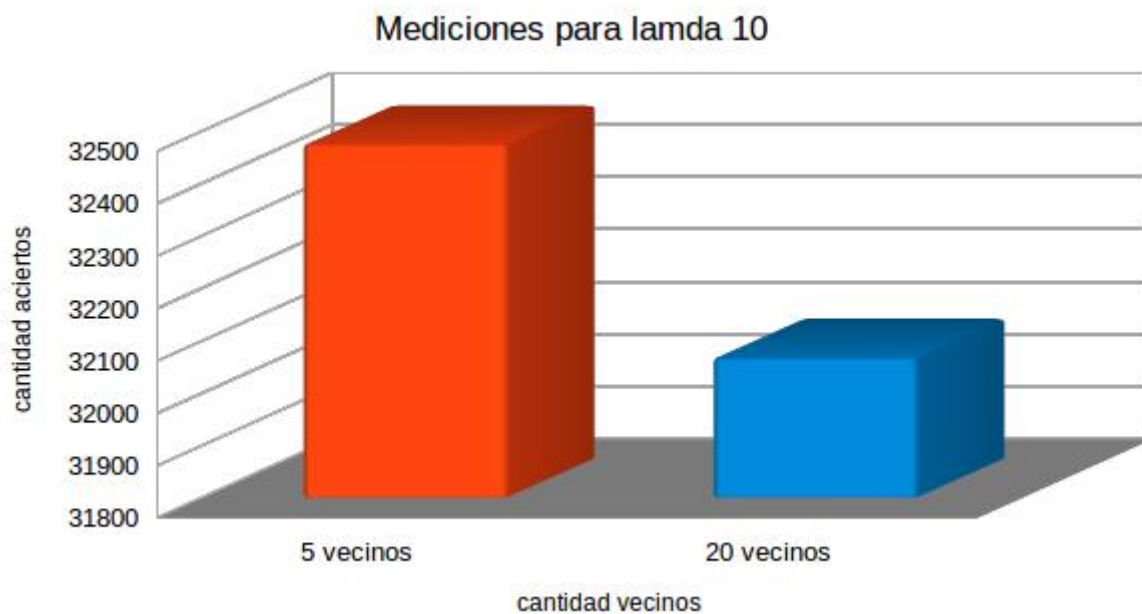
$\alpha$ : 200 y k: 5, 20.

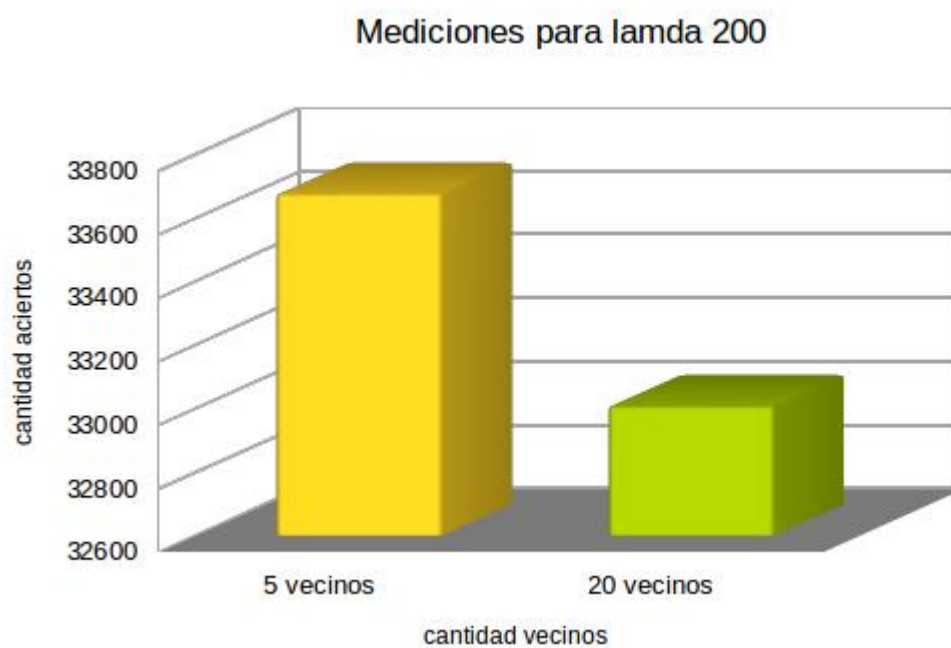
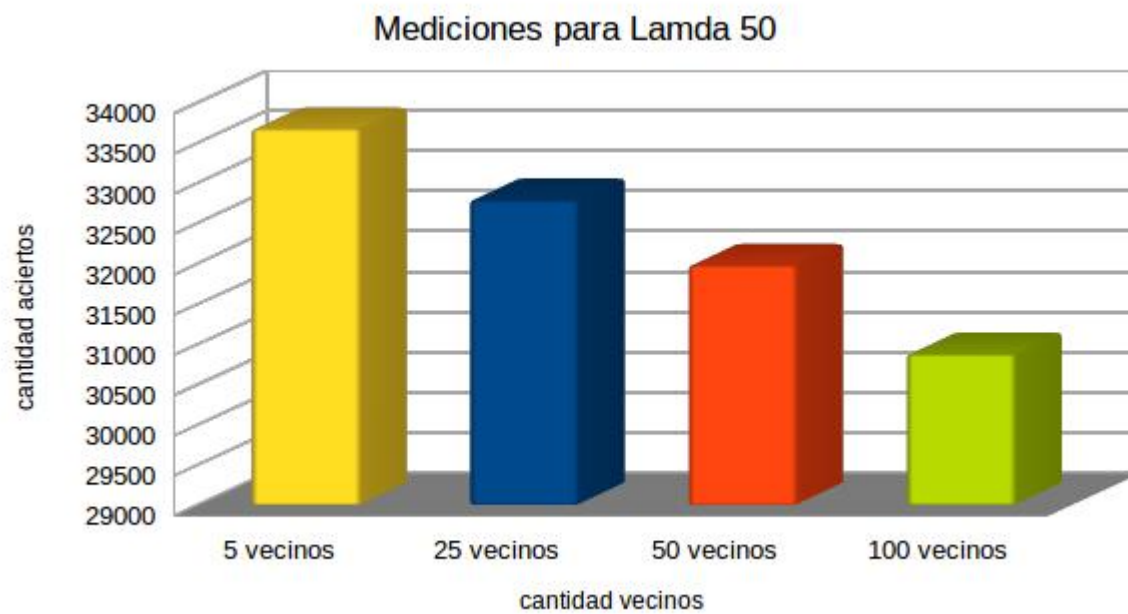
$\alpha$ : 700 y k: 5, 20.

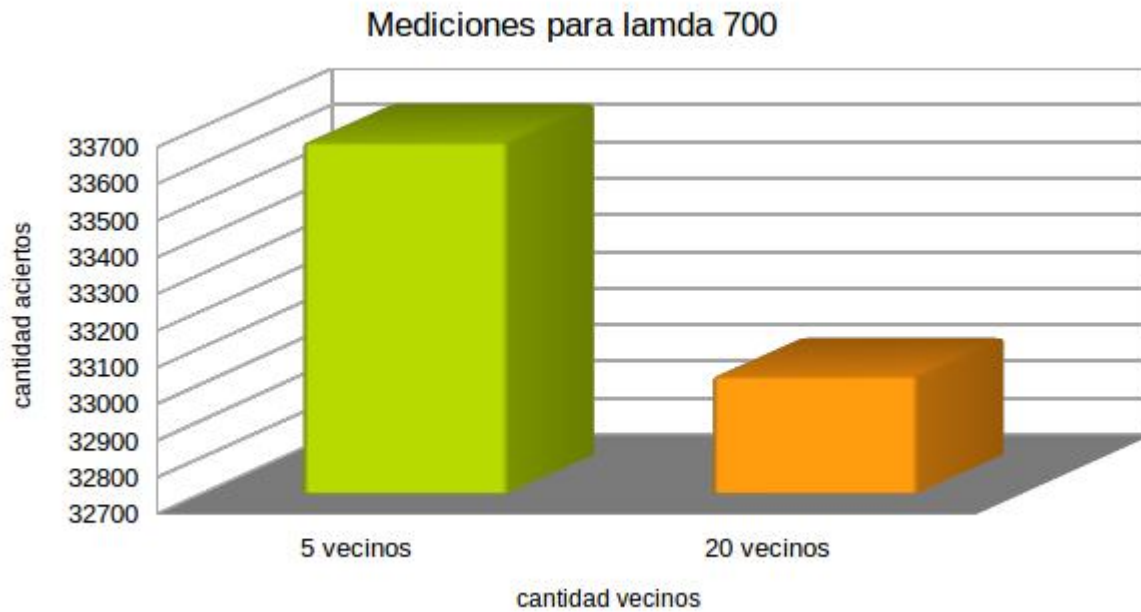
$\alpha$ : 50 y k: 5, 25, 50, 100.

Lo que vamos a probar es fijando un valor de  $\lambda$ , para que cantidad de vecinos vamos a tener la mayor cantidad de aciertos, y así, maximizar la cantidad de aciertos. Nosotros después de aplicar el algoritmo pca, aplicamos el knn y armamos una cola de prioridad para los resultados de aplicar el algoritmo knn. Lo que se hace es agarrar dos imágenes, restarlas y aplicarle la norma2 para saber en cuanto difieren una imagen y la otra. En la cola de prioridad, están adelante los valores más chicos, o sea, las imágenes del test que más cerca de coincidir están con respecto a la imagen de la base de datos. Por lo tanto, si elegimos más cantidad de vecinos, lo que nos puede pasar son dos cosas: 1) Que sea beneficioso ya que a mayor cantidad de pruebas vamos a tener más aciertos 2) Que sea malicioso ya que a mayor cantidad de pruebas vamos a obtener peores datos, o sea, vamos a mirar las imágenes que menos coinciden con la imagen de prueba de la base de datos.

La conclusión que llegamos es que a mayor cantidad de vecinos, el algoritmo empieza a funcionar peor, ya que estamos mirando los vecinos que menos coinciden con la base de datos, porque la cola de prioridad los ordena según menos diferencias haya entre la foto obtenida y las fotos de la base de datos. Por lo tanto, cuanto más vecinos exploro, menos cantidad de aciertos obtengo. Eso lo puedo ver a través de las siguientes mediciones realizadas para distintos  $k$  vecinos y fijando un valor de  $\lambda$ .







Otra conclusion que sacamos es que para  $k=5$  vecinos es la mejor cantidad de vecinos para obtener la mayor cantidad de aciertos. Ahora, ¿No seria mejor agarrar solo el primero de la cola de prioridad, o sea,  $k=1$ ? Lo que podria pasar es que el unico vecino que agarremos, sea el mejor pero no alcance para saber cual es el digito de la imagen, ya que si no se acierta con el unico vecino que elegi, me podria fallar el digito del resultado y eso reduciria muchisimo la cantidad de aciertos.

### 3.2.1. Lambda inicial

### 3.2.2. Algo mas que no me acuerde



## 4. Resultados

### 4.1.

## 5. Conclusiones

- 1) El algoritmo KNN es muy efectivo ya que tiene aproximadamente entre 85 % y 90 % de aciertos. Pero su 'deficit' es que es muy lento a comparacion del algoritmo PCA.
- 2) El algoritmo PCA tiene un pequeño menor porcentaje de efectividad con respecto al algoritmo KNN pero es muchisimo mas rapido en tiempo de ejecucion, por lo que vale la pena perder un poquito de efectividad a cambio de muchisimo menor tiempo de ejecucion del PCA.
- 3) Cuanto mas vecinos exploro, menor es la cantidad de aciertos ya que estoy mirando los vecinos que mas difieren entre si y eso hace que pueda equivocarme a la hora de elegir el digito correcto.

## 6. Apendice