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

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. Introduccion

INTRO!!

2. Desarrollo

2.1. Algortimo 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 caracteristicas 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 incoveniente 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 dependecia 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

Analisis de KNN

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

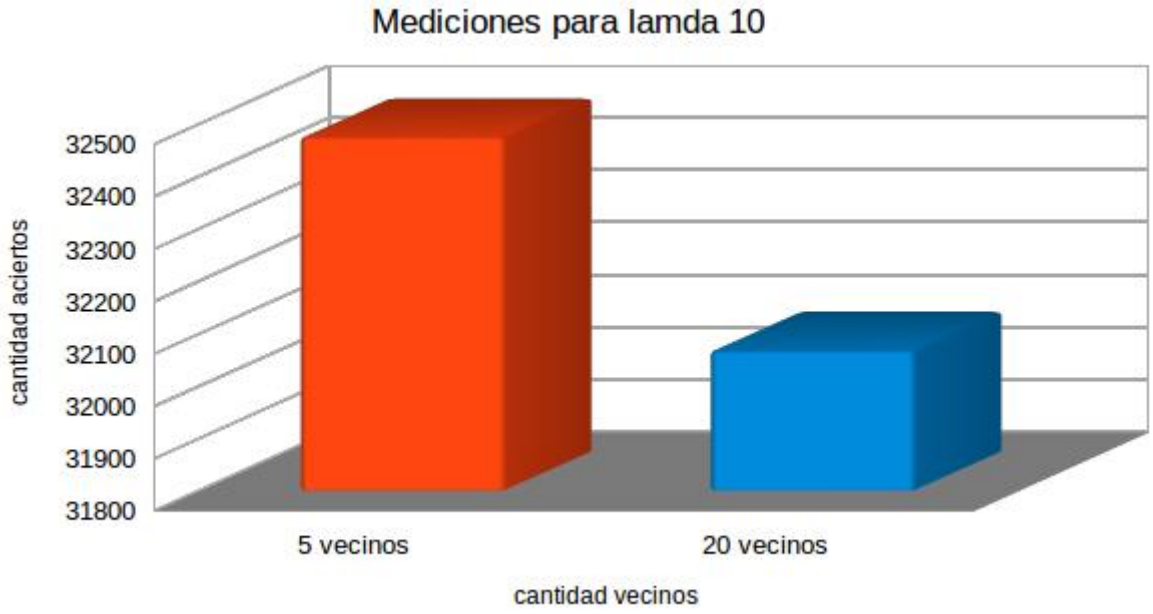
3.2. PCA

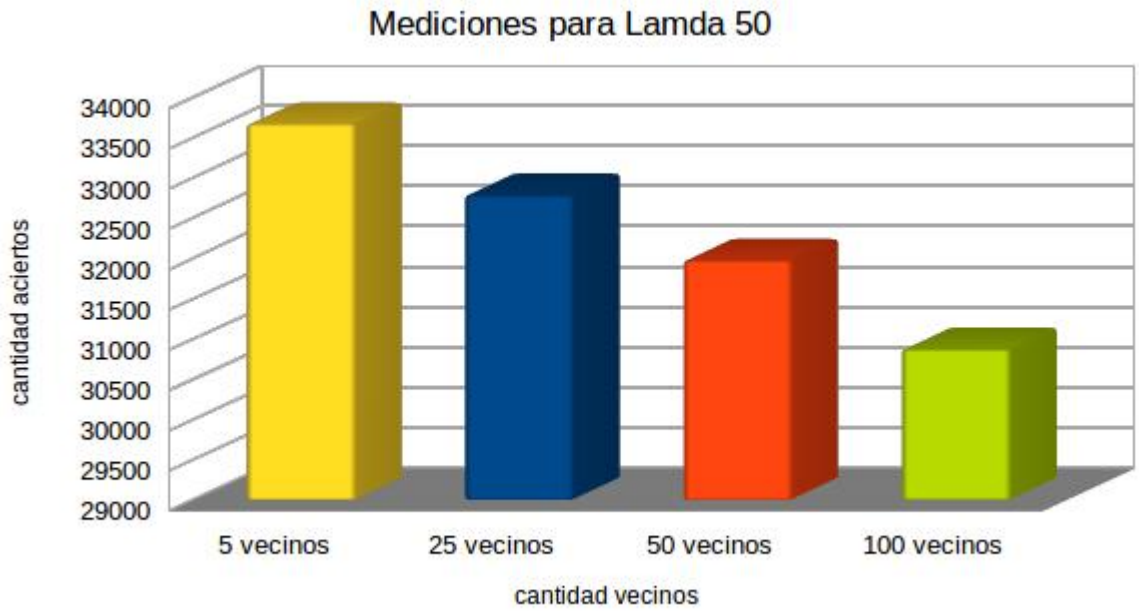
Analisis de PCA Ahora vamos a analizar el algoritmo del PCA. Vamos a probar el algoritmo para distintas medidas de k y α, que van a ser: k: cantidad de vecinos a considerar en el algoritmo kNN. α: a la cantidad de componentes principales a tomar.

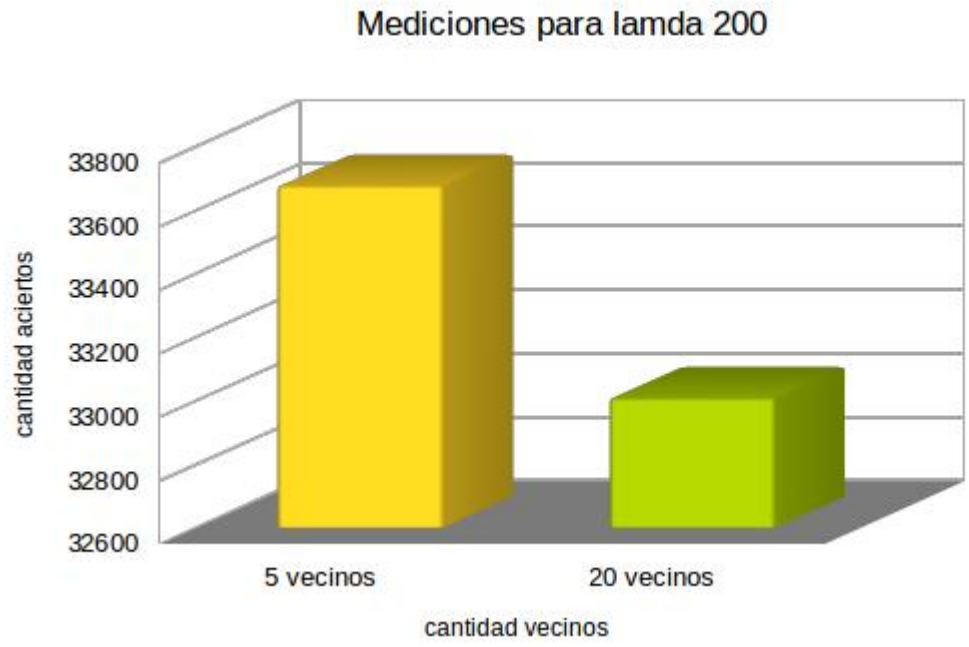
Vamos a probar el algoritmo para los siguientes valores: k: α:

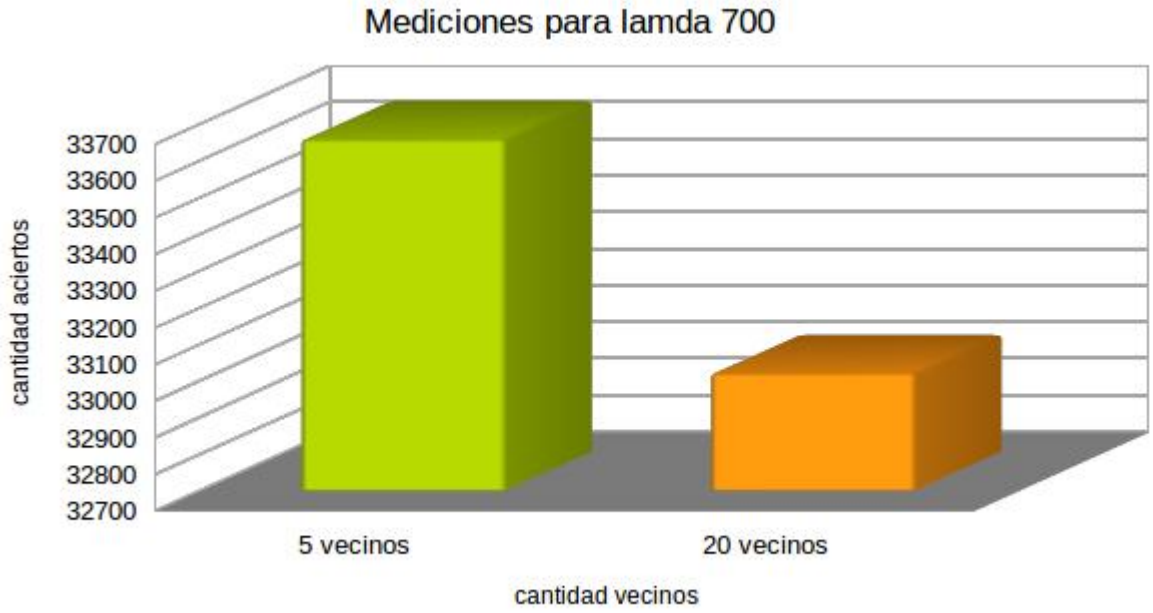
Lo que vamos a probar es fijando un valor de lamda, para que cantidad de vecinos vamos a tener la mayor cantidad de aciertos, y asi, maximizar la cantidad de aciertos. Nosotros despues de aplicar el algoritmo pca, aplicamos el knn y armamos una cola de prioridad para los resultados de aplicar el algoritmo knn. Lo que se hace es agarrar dos imagenes, restarlas y aplicarle la norma2 para saber en cuanto difieren una imagen y la otra. El la cola de prioridad, estan adelante los valores mas chicos, o sea, las imagenes del test que mas cerca de coincidir estan con respecto a la imagen de la base de datos. Por lo tanto, si elegimos mas 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 mas aciertos 2) Que sea malicioso ya que a mayor cantidad de pruebas vamos a obtener peores datos, o sea, vamos a mirar las imagenes que menos coiciden con la imagen de prueba de la base de datos.

La conclusion 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 segun menos diferencias haya entre la foto obtenida y las fotos de la base de datos. Por lo tanto, cuanto mas vecinos exploro, menos cantidad de aciertos obtengo. Eso lo puedo ver a traves de las siguientes mediciones realizadas para distintos k vecinos y fijando un valor de lamda.


Mediciones para lamda 10
cantidad aciertos
32500
32400
32300
32200
32100
32000
31900
31800
5 vecinos
20 vecinos
cantidad vecinos


Mediciones para Lamda 50
cantidad aciertos
34000
33500
33000
32500
32000
31500
31000
30500
30000
29500
29000
5 vecinos
25 vecinos
50 vecinos
100 vecinos
cantidad vecinos





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

5.1. Aca tiramos tiros

6. Apendice