Proyecto PER – Entrega 1

Elias Urios Alacreu

1.Ejercicios obligatorios

PCA se halla implementado en el fichero pca.m y su funcionamiento es auto explicativo tal y como se puede ver por los comentarios presentes en la función. Por otro lado, los resultados de la visualización de los 5 primeros vectores propios son:

Con pca+knn-exp.m y pca+knn-eva.m ocurre lo mismo, el código resulta autoexplicativo. Con pca+knn-exp.m y distintos valores de k, vemos que obtenemos la menor tasa de error al proyectar con k = 100 (en este caso, el error de clasificación es el mínimo entre los distintos valores de k y es menor error que con el original). Los resultados resultan lógicos, a menor proyección los datos resultantes presentan una tasa de error mayor, conforme vamos aumentando las dimensiones en las que proyectamos vemos que el error disminuye hasta llegar a un mínimo (k=100). Si aumentamos ahí vemos que el error vuelve a aumentar pero que nos acercamos a una cota impuesta por el error original.

Error original = 2,82 %

Error proyectando a 100 dimensiones = 2,4 %

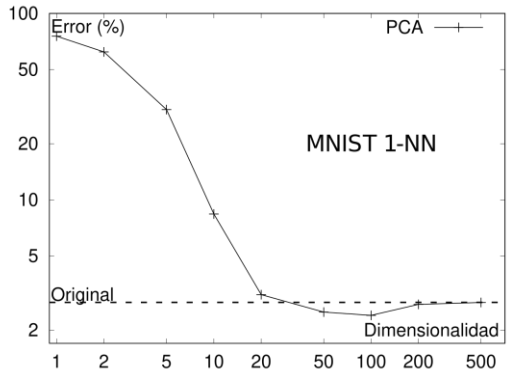

Figura 1. Tasa de error obtenida con reconstrucción PCA y usando knn para clasificar con distancia L2

Ahora, viendo los resultados tras llevar a cabo la ejecución de pca+knn-eva.m y con k=100 con el conjunto de entrenamiento y los datos a clasificar, vemos que el error con el uso de la distancia euclidiana a través de KNN coincide con los resultados expuestos en la web del MNIST (3,09% de error) y, tras hacer la proyección en 100 dimensiones y usando el mismo método de clasificación, el error proporcionado por nuestro programa es de 2,84 %. No podemos encontrar este error en la web del MNIST puesto que nadie se ha dedicado a realizar la implementación a través de nuestros métodos (uso de pca + knn), sin embargo, podemos asumir que se trata de un valor estable viendo el rango de valores presente en la web del MNIST.

2.Ejercicios opcionales

Se han hecho dos ejercicios opcionales: distancias L0, L1 y L3 y la implementación de la distancia Euclídea ponderada (**Mahalanobis diagonal por clase)**. Para ello hemos modificado el fichero pca+knn-exp.m y pca+knn-eva.m para incluir un parámetro de entrada **p.** Así mismo cuando ambas funciones llaman a knn y queremos calcular distancias, existe una función intermedia **GetDistance** que actúa como un switch para que se calculen las distancias con la p correspondiente (0,1,2 o 3). En los ficheros L0dist, L1dist, L2dist y L3dist están implementado los algoritmos, siendo L2 la que está implementada de forma eficiente y los 3 restantes de forma más rudimentaria. La implementación de los algoritmos poco eficientes el método a seguir ha sido coger la fórmula al pie de la letra y realizar las operaciones (pero quitando las raíces para hacer el cálculo de forma más rápida, puesto que tenemos que saber cuál es la distancia exacta sabemos que a mayor valor del número dentro de la raíz mayor será la solución):

Están todas las distancias implementadas, pero como el tiempo de ejecución de cada una a realizar es demasiado largo, se ha decidido únicamente realizar las pruebas con L3 para comparar nuestros resultados con los del MNIST.

Para distintos valores de k obtenemos la siguiente gráfica con el error:

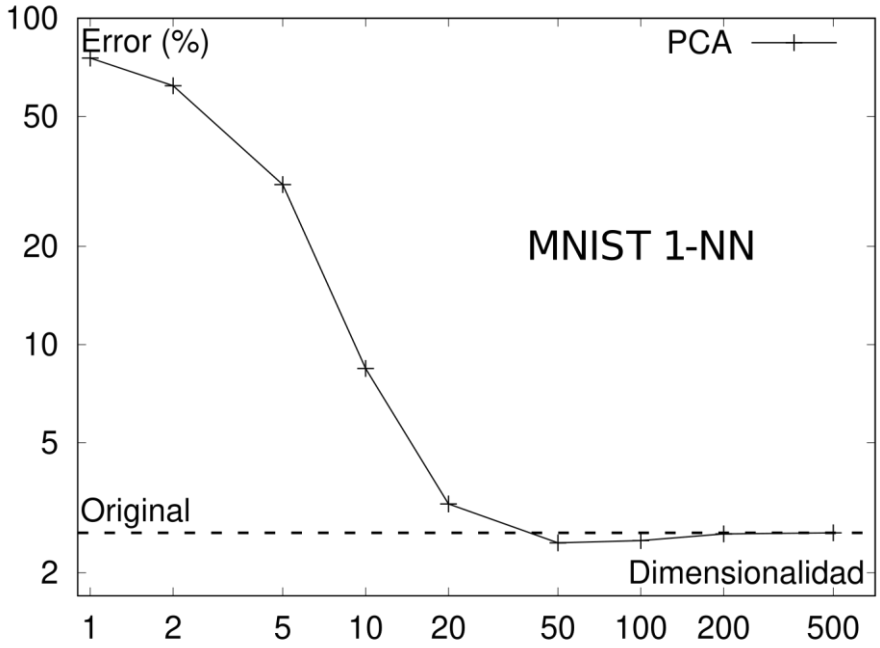

Figura 2. tasa de error en función de las dimensiones proyectadas con PCA+ knn con distancia l3

Vemos que la mejor proyección se da cuando k = 50, donde obtenemos una tasa de error equivalente a un 2,47 % aproximadamente.

Ahora pasamos a realizar el experimento sobre el conjunto de entrenamiento y las muestras oficiales, primero aplicamos knn y vemos que nuestro resultado coincide con el que hallamos en la web del MNIST (tasa de error del 2,83%), después de aplicar la técnica PCA y knn la tasa de error obtenida es de un 2,71%, un resultado bastante correcto.

Para calcular la distancia Euclídea ponderada, primero debemos calcular la matriz de varianza de cada clase en una matriz W, en cada fila i almacenaremos la variación de cada vector y luego al hacer las operaciones correspondientes de distancia L2 simplemente hay que dividir por la inversa de la matriz de varianza. Para más detalles ver los comentarios de los correspondientes archivos, pero básicamente se ha implementado la fórmula de forma convencional.