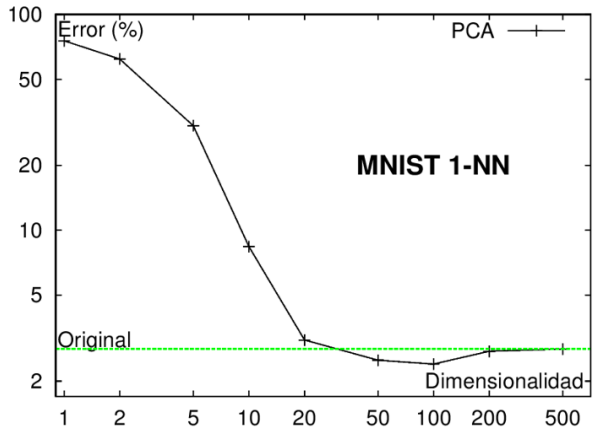
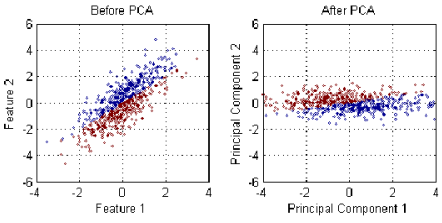
Ejercicio PCA: A continuación tenemos la gráfica obtenida con el pca+knn-exp.m, utilizando para ello la distancia L2 implementada en el archivo L2dist.m del ejercicio 3 del boletín, usada a su vez por el knn.m para calcular los vecinos más cercanos y el pca.m para reducir la dimensionalidad. En dicho ejercicio se nos planteó hacer un estudio sobre los distintos números de k a utilizar en nuestro k-medias con tal de encontrar la mejor combinación.

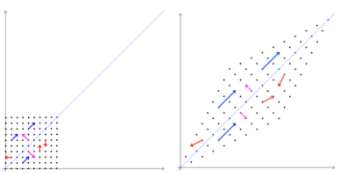


Por lo tanto, a lo largo del experimento usamos el dataset de dígitos manuscritos MNIST. Para pca+knn-exp.m cogimos los datos de entrenamiento de MNIST y los separamos en el subconjunto de entrenamiento y validación con una proporción de 90% para training_data y 10% para el validation_data. A continuación hicimos dos pruebas separadas, por una parte ejecutamos el K-NN con un vecino más cercano con dimensionalidad original de los datos, sin usar el PCA y usando la distancia L2 sobre en conjunto de validación, para ver cual seria el error con los datos originales sin redimensionarlos, el cual resultado nos dio 2.8167%. Dicho resultado se representa en la tabla con la línea verde. Seguidamente, aplicamos PCA a nuestro conjunto de entrenamiento y calculamos el K-NN para todos los k del siguiente conjunto [1, 2, 5, 10, 20, 50, 100, 200, 500]. Así pues, obtenemos la tabla de la izquierda, en la que podemos observar los distintos valores de k, y por otro lado en la grafica1 se muestran los resultados ,obsérvese que el eje x representa la dimensionalidad (k vecinos usados) y el eje y el error en porcentaje en los datos de validación.

	PCA + L2
Original (no PCA)	2,81667
ks= 1	75,466
ks= 2	62,15
ks= 5	30,483
ks= 10	8,4
ks= 20	3,1
ks= 50	2,5
ks= 100	2,4
ks= 200	2,75
ks= 500	2,8167
TEST DATA (PCA)	2,84
TEST DATA (no PCA)	3,09

Ahora bien, visto como sacamos estos resultados, procedamos a su análisis. Observando la gráfica con el PCA aplicado, vemos que cuantos mas vecinos cojamos para el calculo, menor error conseguimos mientras nos vamos acercando a un punto donde se estabiliza este error, que sería el error mínimo, el cual al final conforme usaremos más vecinos, convergerá al error original. Por lo tanto, en nuestro caso en concreto, el K óptimo seria el 100, ya que conseguimos un error sobre los datos de validación de 2.4%. Sabemos que el PCA reduce la dimensionalidad del espacio de características al restringir la atención a aquellas direcciones a lo largo de las cuales la dispersión de la nube de datos es mayor, por otro lado, nos interesa elegir el eigenvector correspondiente al mayor eigenvalue de la matriz de covarianzas. Así pues, como resultado de aplicar PCA sobre nuestro conjunto de datos, tendremos que el vector principal estará representando la máxima dispersión y, por lo tanto al usar k pequeños, nos arriesgamos a que en caso de tener las características de todos los datos parecidas, pues tendremos más posibilidades de fallos, ya que la varianza de por medio es mayor y pueden haber datos de otras clases que sean más cercanas al nuestro, recordemos que proyectamos la matriz de pesos sobre nuestros datos, con lo que después de esta proyección nuestros datos podrías acabar representándose como el dibujo de la izquierda. Resumiendo, al proyectar la matriz de pesos sobre nuestros datos, sus direcciones no cambian, sin embargo sus magnitudes si, por ello las distancias entre los puntos pueden variar y, si antes para k = 1, se calculaba una clase, después de proyectar la matriz de pesos obtenida por el PCA sobre los





valores, nos podría dar otra clase distinta. Por lo tanto, dicha situación la podemos observar en la grafica de cuando aplicamos el PCA, para K = 1, en el K-NN original el error es pequeño, pero en el del PCA es muy grande, pero luego para un K considerable, los errores convergen en lo mismo, pero por el camino, tenemos un k donde el PCA no da un error menor, por esto nos renta utilizarlo. Por otro lado, el K-NN con un vecino más cercano, no tendría este problema, con lo que su error sería menor.
Una vez hemos comprobado el error sobre los datos de validación, que en nuestro caso es con K = 100 para la distancia L2, tenemos que comprobarlo ahora en los datos de test. Para ello usaremos el siguiente ejecutable que es el pca+knn-eva.m, al cual le pasaremos todo el training_data y el test_data de MNIST, sobre el cual probaremos los parámetros elegidos en el experiment. Por lo tanto, nuestro esperimento del pca+knn-eva constará de dos partes, una en la que calcularemos el error usando PCA y otro donde no usaremos PCA para ver si de verdad los parámetros elegidos mejoran la predicción o no.

	PCA + L2
Original (no PCA)	2,81667
TEST DATA (PCA)	2,84
TEST DATA (no PCA)	3,09

Al ejecutar nuestro K-NN sobre los datos de testeo, hemos podido observar que, por una parte, el error obtenido sobre los datos de validación no se va mucho del error obtenido en los datos de testeo, eso quiere decir que hemos logrado elegir unos buenos parámetros. También es interesante apuntar, de que el error en el test es mayor, es debido a que hemos entrenado en un conjunto de muestras muy pequeño, y al utilizar el modelo en un dataset de test mucho más grande, hay más cambios y combinaciones de distancias en las que podemos fallar. Por otra parte, obsérvese que al aplicar PCA mejoramos un 0.25% respecto a no aplicar PCA, sigue siendo un valor muy pequeño, ya que tenemos que tener en cuenta que seguimos ejecutando sobre datos de testeo, por lo que a la hora de elegir entre modelos, teniendo en cuenta algún margen, podríamos ya compararlo, pero a la hora de aplicar el modelo en producción, el error podría subir más, dichas comparaciones solo son orientativas.

K-Nearest Neighbors			
K-nearest-neighbors, Euclidean (L2)	none	5.0	LeCun et al. 1998
K-nearest-neighbors, Euclidean (L2)	none	3.09	Kenneth Wilder, U. Chicago

Por último, vamos a comprobar nuestro resultado del test con la página oficial de MNIST, observaremos el clasificador ‘K-nearest-neighbors, Euclidean(L2)’ por ser el mismo clasificador que nosotros usamos. Así pues, en primer lugar vemos que el clasificador de Jenneth Wilder tiene el mismo error que nosotros en el test, 3.09% sin aplicar PCA. Con lo que podríamos dar por supuesto que nuestro clasificador no es del todo malo. Por otro lado, si nos fijamos en el clasificador hecho por LeCun, podemos ver que tiene 2% de error más que nosotros, por lo tanto, a la hora de elegir uno u otro clasificador, nos quedaríamos con el nuestro, ya que hasta suponiendo algún margen de error para clasificar muestras nuevas, 2% es una diferencia ya considerable. Sin duda alguna, para clasificar dígitos nuestro modelo es mucho mejor que el de LeCun.

K-nearest-neighbors, Euclidean (L2)	deskewing	2.4	LeCun et al. 1998
K-nearest-neighbors, Euclidean (L2)	deskewing, noise removal, blurring	1.80	Kenneth Wilder, U. Chicago

Así pues, una vez comprobada la parte por defecto del clasificador sin utilizar ningún preprocesado, vamos a comprobar si la implementación con PCA es tan buena como lo creíamos o no. Recordemos que el error que conseguimos utilizando PCA es de 2.84% sobre los datos de test, así pues, fijándonos en el clasificador donde solo usa enderezado, vemos que es casi un 0.5% mejor que nuestro modelo, por lo tanto, es una diferencia de error relativamente pequeña, con lo que podríamos quedarnos con ese modelo, pero antes deberíamos ver si nos rentaría desde la parte de computabilidad y complejidad, ya que si nos mejora solo un 0.5% pero pasaremos el doble de tiempo ejecutando el algoritmo por ejemplo, no nos sale rentable el cambio, ya que un 0.5% de error es muy pequeño. Por otro lado, la implementación de 1.8% de error de Keneth Wilder, es 1.04% mejor que nuestro modelo, otra vez estamos en las mismas, la diferencia de error ya es considerable, ya nos podríamos plantear cambiar de modelo o mejorar el nuestro en caso de ponerlo en producción, pero aún así, antes de nada, sería recomendable hacer un estudio de costes y complejidad de los algoritmos para decidir si de verdad sale rentable dicho cambio o no.