# Reconocimiento de letras y dígitos mediante el Análisis de las Componentes Principales (PCA)

Memoria Técnica

Nicolás Chareca
chareca.165075

7 de noviembre de 2025

## Índice

# 1. Introducción

En esta memoria se documentará el proceso de clasificación de imágenes a través del algoritmo de PCA (Principal Component Analysis).

Buscamos predecir a partir de un dataset de entrenamiento y sus clases reales (aprendizaje supervisado), cuál es la clase correspondiente de 1 o más ejemplos de prueba.

El funcionamiento de este algoritmo consiste en el prototipado de una imagen genérica para cada clase distinta, y buscar el mínimo MSE *(Mean Squared Error)* de cada prototipo con la imagen en cuestión, eligiendo así como clase predicha la clase correspondiente al prototipo cuya MSE con la imagen es mínima.

Se presentarán gráficos que ilustran el rendimiento del clasificador en función de diferentes parámetros de entrada, permitiendo un análisis visual de su comportamiento.

# 2. Explicación del problema

En este caso específico, contamos con dos datasets:
`[t10k-images-idx3-ubyte, t10k-labels-idx1-ubyte, train-images-idx3-ubyte, train-labels-idx1-ubyte]` y
`[emnist-letters.mat]`, en los cuales se encuentran imágenes de dimensión 28x28 píxeles. El primer dataset está formado por dígitos del 0 al 9, y en el segundo, los dígitos del 0 a 9 y el abecedario inglés completo en mayúscula y minúscula.

Estos dígitos y letras están escritos a mano, lo que causa imperfecciones en su escritura, por lo que a partir de un entrenamiento buscamos predecir de qué dígito (o letra) se trata cada ejemplo de prueba.

Las pruebas específicas que se realizaron son:

1. **Clasificación binaria** entre dígitos del primer dataset (80 ejemplos train, 20 ejemplos test)
   - Clases 1 y 7
   - Clases 2 y 7
   - Clases 4 y 9
2. **Clasificación entre dígitos** del primer dataset para **todas las clases** [0-9]
   - 80 ejemplos train, 20 ejemplos test
   - 160 ejemplos train, 40 ejemplos test
3. **Clasificación entre letras** minúsculas del segundo dataset para **todas las clases** [1-26]
   - 1144 ejemplos (1040 train, 104 test)
   - 2288 ejemplos (2080 train, 208 test)
   - 5720 ejemplos (5200 train, 520 test)
   - 11440 ejemplos (10400 train, 1040 test)

# 3. Resultados

Los resultados obtenidos para las diferentes clasificaciones realizadas son los siguientes:

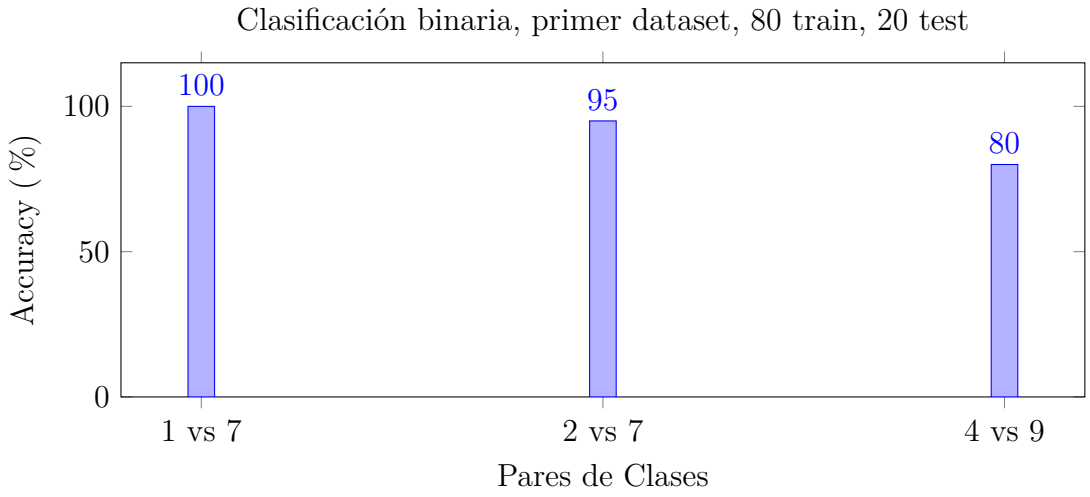


Figura 1: Accuracy de las clasificaciones binarias para clases "difíciles".

En la figura 1 se demuestra el comportamiento del clasificador binario para el primer dataset, sobre dígitos, contando con 80 ejemplos del conjunto de entrenamiento y 20 ejemplos del conjunto de test. Estas pruebas se realizaron para las clases indicadas por el enunciado de la práctica: (1 y 7), (2 y 7) y (4 y 9).

Se planteó en el enunciado esta pregunta:

> *¿Cómo clasifica el algoritmo realizando la distinción entre las clases "más conflictivas"? ¿Por qué consideras que clasifica bien o mal?*

Analizando el comportamiento del clasificador, es notable que la clasificación entre 4 y 9 es decente, aun siendo números fácilmente confundibles (al estar escritos a mano) debido a su similitud tipográfica. Estos son los resultados de comprobar qué sucedía si utilizaba más ejemplos para comparar estas clases, en vez de solo 80 y 20.

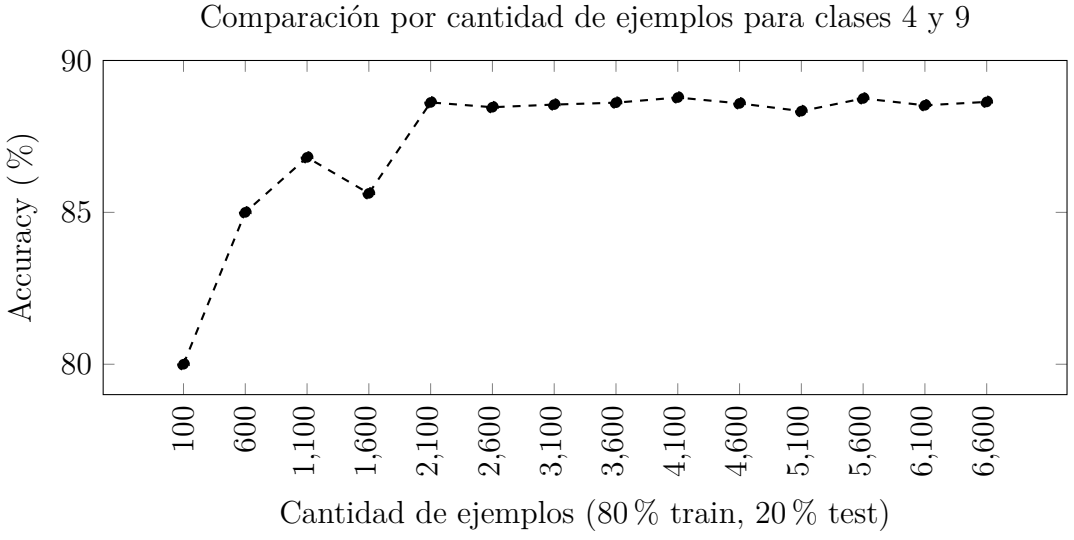


Figura 2: Accuracy de las clasificaciones binarias para las clases 4 y 9.

Como se visualiza en la figura 2, aunque el rendimiento del clasificador era bueno, al aumentar el número de ejemplos, este pasa a aprender mejor las características que diferencian a un 4 de un 9 al estar escrito a mano. Seguramente, por ejemplo, ha aprendido que el 4 no cuenta con la unión superior con la que sí cuenta el 9 .

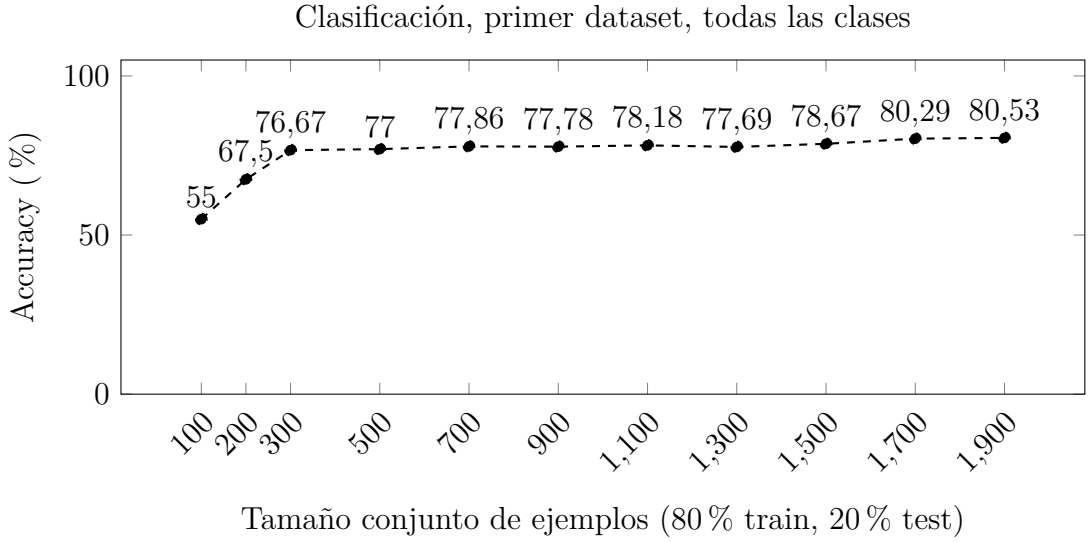


Figura 3: Accuracy de las clasificaciones para todas las clases [0-9] 1er dataset.

En la figura 3 se demuestra el comportamiento del clasificador para el primer dataset, sobre dígitos, contando con una diferente variedad de ejemplos, entre 100 y 1900 como total, siempre manteniendo la proporción 80 % conjunto de entrenamiento, 20 % conjunto de test. Estas pruebas se realizaron para las clases [0-9], es decir, para todos los números del dataset.

Se puede concluir que, para un menor número de ejemplos de cada clase, el clasificador tiene un peor rendimiento, ya que el entrenamiento por el que pasa se realiza con una menor cantidad de datos. Sin embargo, a medida que el número avanza, el clasificador mejora su clasificación a menor ritmo. Esto se puede deber a que aún contando con más ejemplos, el clasificador ya cuenta con una gran cantidad de ellos, por lo que las características que se aprendan sobre los nuevos ejemplos no afectarán tanto al rendimiento del clasificador.

Otra forma de explicarlo es que entre 100 y 300 se realiza un aumento del 200 % de datos, mientras que entre 1500 y 1700 el aumento es de 13.33 %.

En el enunciado se mencionó esta pregunta:

> *¿Qué ocurre cuando tiene que clasificar 10 clases? ¿Cambia algo el algoritmo cuando tenemos el doble de datos de entrenamiento?*

Lo que sucede en primera instancia es que el porcentaje de aciertos baja considerablemente a la hora de compararlo con clasificaciones entre 2 clases. El algoritmo sigue siendo exactamente el mismo, ya que está programado para utilizar cualquier cantidad de clases y de ejemplos.

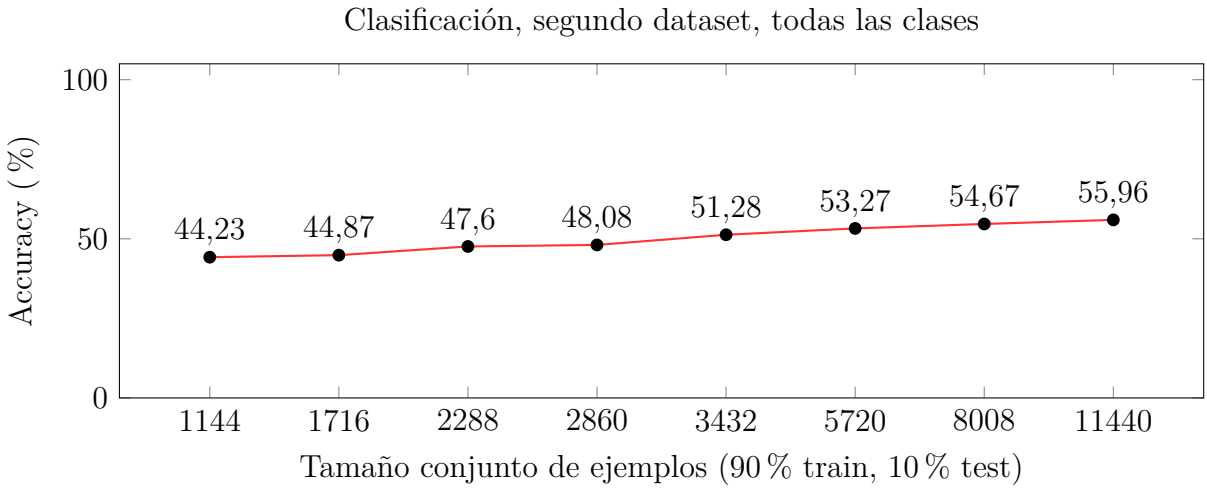


Figura 4: Accuracy de las clasificaciones para todas las clases [1-26] 2do dataset.

La Figura 4 muestra el rendimiento del clasificador en el dataset de letras minúsculas, usando entre 1144 y 11440 ejemplos totales, con una división del 90 % para entrenamiento y 10 % para test. Estas pruebas se realizaron para todas las clases [1-26], es decir, para todas las letras del alfabeto inglés (sin la ñ).

Se puede concluir que, para un menor número de ejemplos de cada clase, el clasificador tiene un peor rendimiento, ya que el entrenamiento por el que pasa se realiza con una menor cantidad de datos totales.

En el enunciado se preguntó:

> *¿Qué ocurre cuando tratamos de clasificar 26 clases con un entrenamiento realizado con 1000 ejemplos? ¿Por qué crees que funciona de esa manera? ¿Has probado a realizar un entrenamiento con más ejemplos (2000, 5000, 10000, etc.)? Si lo has probado, comenta qué es lo que ocurre.*

Cuando tratamos de clasificar 26 clases con un entrenamiento realizado con $\approx 1000$ ejemplos (1144), el clasificador tiene un valor de accuracy mucho más bajo que para menos clases y la misma cantidad de ejemplos. Creo que funciona de esta manera, ya que para más clases hay que aprender muchas más características por cada una de las clases, por lo que el clasificador no se especializa en la diferenciación entre 2 clases, sino que aumenta su alcance de predicción disminuyendo su accuracy score. Al entrenar con muchos más datos, como primer efecto notable, el código toma bastante más tiempo en ejecutar debido a su complejidad computacional, y también aumenta considerablemente el porcentaje de aciertos entre 1144 y 11440 ($55,96\,\% - 44,23\,\% = +11,73\,\%$).

# 4. Conclusión final

Estos resultados nos dan a comprender que el rendimiento del algoritmo de PCA está directamente ligado a la complejidad del problema. El algoritmo, para este dataset, ofrece resultados excelentes en tareas de clasificación binaria con clases con características más distinguibles (como "1" vs "7"), pero su rendimiento decrece al comparar clases más conflictivas (como "4" vs "9"). Esta caída de accuracy se potencia cuando el problema es escalado a más clases, con la clasificación de 10 dígitos (MNIST) y la de 26 letras (EMNIST). En estos casos, el accuracy score inicial es considerablemente más bajo, ya que el clasificador no se especializa en diferenciar pocas clases, sino que debe aprender las características de un conjunto mucho mayor.

Se demostró que al aumentar el volumen de datos de entrenamiento, el accuracy mejora en todos los escenarios. El mayor impacto siempre se muestra cuando el aumento de datos de entrenamiento entre una iteracion y otra representa un alto % del conjunto de entrenamiento de la primera iteracion sobre la segunda. Aún así, en el clasificado de las 26 clases de letras minúsculas del problema, aumentar hasta 10 veces (1144 vs 11440) el tamaño del conjunto de entrenamiento no se logró superar el 56 % de accuracy. Esto, sumado el problema del coste computacional que conlleva tratar tantos datos de entrenamiento, sugiere que el método de prototipado del PCA no es suficiente para modelar correctamente problemas con muchas clases y altas dimensiones de datos, al menos con esta cantidad de datos de entrenamiento.