JUNIO 2025

INTELIGENCIA ARTIFICIAL APLICADA

RECONOCIMIENTO DE NÚMEROS MANUSCRITOS

DAVID LABORDA IZQUIERDO
MARTÍN LORING BUENO
DIEGO RAMÍREZ FUENTE

INDUSTRIALES
ETSII | UPM

CAR

ÍNDICE

1. Introducción

El objetivo de este trabajo es explorar el uso de distintas técnicas de inteligencia artificial para resolver un problema de clasificación de números manuscritos extraídos de la base de datos MNIST. Cada uno de dichos números se representa mediante un vector de 784 componentes que, al recolocarse formando una matriz de 28×28, representan las intensidades de los píxeles de una imagen. Para llevar a cabo la tarea, se cuenta con un conjunto de 10000 de estos números (etiquetados), que pueden emplearse para entrenar y poner a prueba los distintos modelos propuestos. La eficacia de las soluciones propuestas será puesta a prueba con un conjunto de datos de test a los que no se tendrá acceso hasta el final del trabajo.

El reparto de roles de los miembros participantes en este trabajo se muestra en el cuadro 1.1. El coordinador será responsable de organizar el trabajo de los otros dos miembros, verificar la calidad general de los resultados, incluida la presentación, y de escribir el documento. El científico será responsable de la faceta científica del trabajo, las fórmulas y el trabajo matemático. El técnico se encargará de implementar las soluciones propuestas por el científico en MATLAB.

Roles	Estudiantes
Coordinador	Martín Loring Bueno
Científico	David Laborda Izquierdo
Técnico	Diego Ramírez Fuente

Tabla 1.1: Reparto de roles de los miembros del grupo.

2. Preprocesamiento

La primera tarea con los datos será realizar un preprocesamiento, que en este caso se tratará de la normalización de los datos y la reducción de la dimensionalidad. La normalización puede evitar que unas variables dominen sobre otras, mejorar la convergencia o evitar errores numéricos; mientras que reducir la dimensionalidad de los datos permitirá desarrollar algoritmos más veloces y ligeros en memoria.

2.1. Principal component analysis (Tarea 1)

La primera técnica propuesta para llevar a cabo esta reducción es el PCA. Se ha probado esta técnica usando distinto número de dimensiones del subespacio sobre el que se proyectan los datos, calculándose en cada caso el MSE, definido por la siguiente ecuación, donde N es el número de datos, x_i las coordenadas del dato i proyectado y $\hat{x}_i$ las coordenadas del dato i originales:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

En la figura 2.1 se muestra el MSE obtenido al reducir la dimensionalidad de las imágenes en mayor o menor medida. Como puede observarse, dicho error tiende a cero a medida que se aumenta la dimensión del subespacio sobre el que se proyecta. No resulta sencillo señalar en esta gráfica un punto en el que el MSE sea tolerable. Para tener una idea intuitiva del efecto de aplicar esta técnica sobre los datos, se ha reconstruido la proyección de uno de los números

en el espacio original de dimensión 784 y se ha representado en la figura 2.2. En esta imagen, puede apreciarse cómo el número manuscrito se vuelve más reconocible a medida que aumenta la dimensión. Aunque se trata de una afirmación abierta a debate, es razonable decir que el dígito es reconocible proyectando sobre un subespacio de entre 10 y 50 dimensiones. La dimensionalidad exacta del subespacio más óptimo puede depender del método de clasificación escogido. Con el fin de tener una idea preliminar al respecto, se ha calculado la precisión de distintos clasificadores en función de dicha dimensionalidad usando un conjunto de datos reservados para testing. Los resultados se muestran en la figura 2.3. En base a ellos, podemos elegir con más criterio el número de dimensiones sobre el que proyectar los datos en función de cómo pensemos clasificarlos. Los detalles sobre los parámetros y resultados de cada clasificador se discutirán en la sección 3.

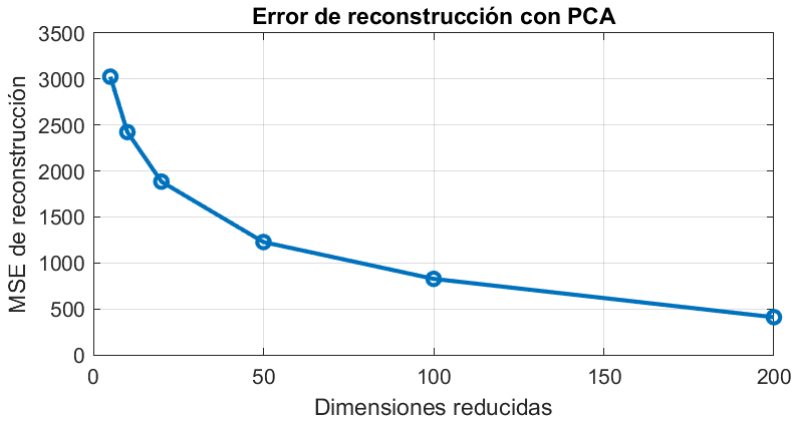

Figura 2.1: MSE de los datos proyectados sobre distintos subespacios de dimensión reducida mediante la técnica de PCA.

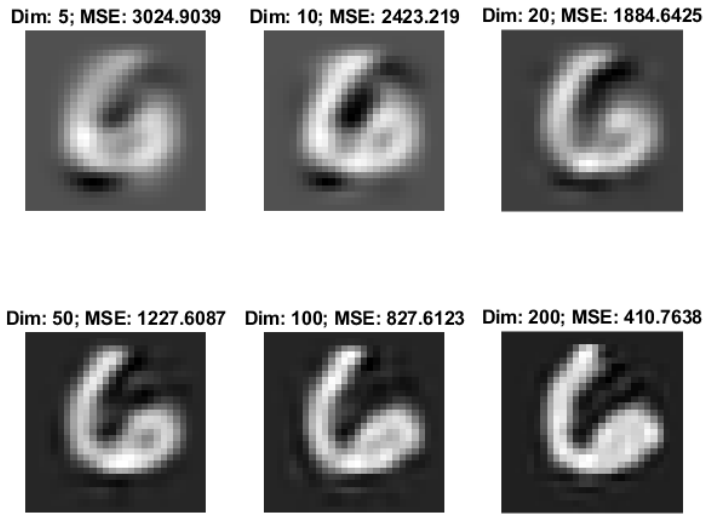

Figura 2.2: Imagen de ejemplo de un número reconstruido tras haber sido proyectado sobre distintos subespacios de dimensión reducida mediante la técnica de PCA.

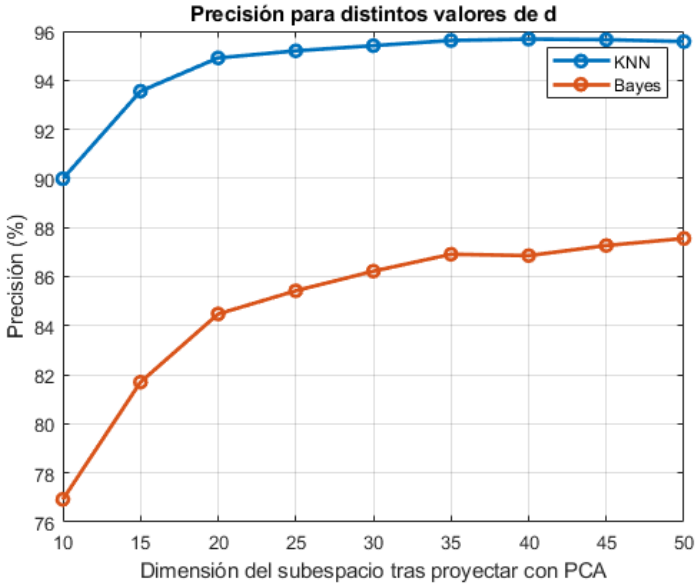

Figura 2.3: Precisión de los clasificadores tras reducir la dimensionalidad de los datos usando PCA en función de la dimensión del espacio sobre el que se proyecta.

2.2. Clustering y discriminante de Fisher (Tareas 7 y 8)

Otro método que puede emplearse para reducir la dimensionalidad del problema es el LDA. A favor de este método se puede decir que es más apto para una tarea de aprendizaje automático supervisada (como es la que estamos tratando). Sin embargo, un problema fundamental que tiene es que siempre proyecta sobre un subespacio de dimensión igual al número de clases - 1. Para el problema aquí tratado, reducir la dimensionalidad desde 784 hasta 9 puede resultar insostenible.

Una solución posible es agrupar los datos de cada clase formando subclases. Esto puede llevarse a cabo utilizando el algoritmo k-means. Al aumentar el número de clases, aumenta la dimensión del subespacio sobre el que se proyectan los datos. Como punto de partida, podemos dividir todas las clases en la misma cantidad de subclases y ver cómo se comportan los clasificadores en este caso. Para tener una primera idea del número de subclases que puede ser conveniente generar, se ha calculado la precisión de los clasificadores KNN y bayesiano sobre un conjunto de datos aislados para testing. Los resultados se muestran en la figura 2.4. En función de estos, se tiene una idea de cuántas subclases resulta conveniente generar dependiendo del algoritmo a utilizar. Sin embargo, esto es solo un punto de partida, y en la sección 3, veremos cómo afinar más la elección utilizando las matrices de confusión.

También se probó ajustar el número de subclases individualmente por dígito, observando cómo cambiaba la precisión del clasificador al modificar un dígito concreto y mantener fijos los demás. Inicialmente, se asignaron más subclases a los dígitos con mayor confusión, lo que mejoró ligeramente la precisión con datos normalizados. Sin embargo, al usar datos no normalizados (que ofrecían mejor rendimiento general) esta estrategia dejó de ser efectiva. Las correlaciones entre el número de subclases y la precisión del algoritmo resultaron demasiado complejas, por lo que un enfoque global resultó más adecuado.

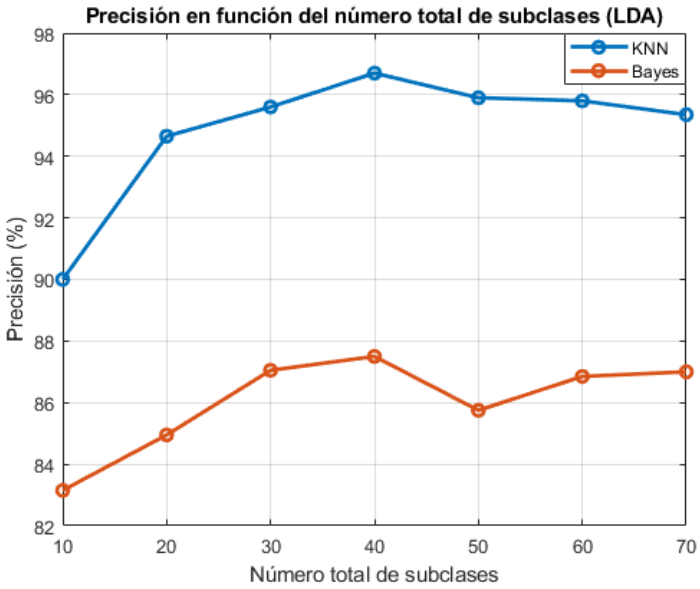

Figura 2.4: Precisión de los clasificadores tras reducir la dimensionalidad de los datos usando LDA habiendo clusterizado los datos en distinto número de subclases.

2.3. Autoencoder (Tarea 9)

Además de las técnicas lineales como PCA o LDA, se probó el autoencoder como método no lineal para reducir la dimensionalidad. Este tipo de técnicas permiten modelar relaciones no lineales entre las variables de entrada, y pueden ofrecer representaciones más eficientes para tareas de clasificación.

Para reducir la carga computacional y facilitar el entrenamiento del autoencoder, se utilizó previamente una reducción de dimensionalidad lineal mediante PCA, fijando el número de componentes en 50. Esta elección se justifica con los resultados presentados en la sección 2.1, donde se observa que a partir de esta dimensión ya se retiene una buena parte de la información relevante. También se probaron valores cercanos como 30, 40 y 60, obteniendo peores resultados en precisión de clasificación o mayor error de reconstrucción.

En la Figura 2.5 se ven reconstrucciones hechas con PCA seguido de un autoencoder, que se comparan con las de la Figura 2.2, donde solo se usó PCA. Aunque las diferencias no son grandes, el autoencoder consigue una mejora visual ligera, sobre todo en los casos con menor compresión. Aun así, esa mejora es leve y no siempre compensa el mayor tiempo de entrenamiento y la complejidad añadida.

El autoencoder fue entrenado para distintas dimensiones de su salida comprimida, evaluando tanto el MSE de reconstrucción como la precisión de clasificación con los modelos k-NN y Bayes. En la Figura 2.6 se muestran estos resultados. Se observa que la menor tasa de error de reconstrucción se alcanza en torno a 30-40 dimensiones latentes. En cuanto a la clasificación, el clasificador k-NN obtiene una precisión máxima del 94,38 % para AE de dimensión 30. Por otro lado, el clasificador bayesiano presenta un comportamiento mucho peor, con una precisión máxima de 82,42 % para AE de dimensión 40.

Este bajo rendimiento de Bayes se debe a que las representaciones generadas por el autoencoder no siguen una distribución gaussiana, lo cual contradice la hipótesis fundamental de este clasificador. Para solucionarlo, fue necesario utilizar una versión basada en *kernel density estimation*,

mucho más costosa en términos de tiempo de entrenamiento y clasificación. Este problema, junto con que los resultados no mejoran en precisión respecto a usar únicamente PCA, hace que el uso del autoencoder no sea recomendable para este conjunto de datos concreto. Cabe destacar que el autoencoder, al ser una red MLP, requiere un tiempo de entrenamiento considerable (3.5 s en este caso). Aunque este tiempo no afecta directamente al de clasificación, sí incrementa el coste de preprocesamiento, sin aportar mejoras significativas en precisión que lo justifiquen.

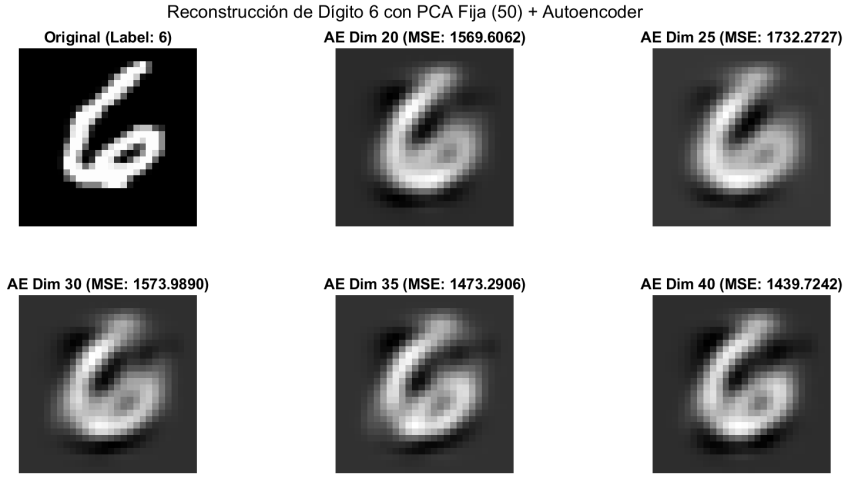

Figura 2.5: Reconstrucción de un dígito tras aplicar PCA y luego un autoencoder con distintas dimensiones.

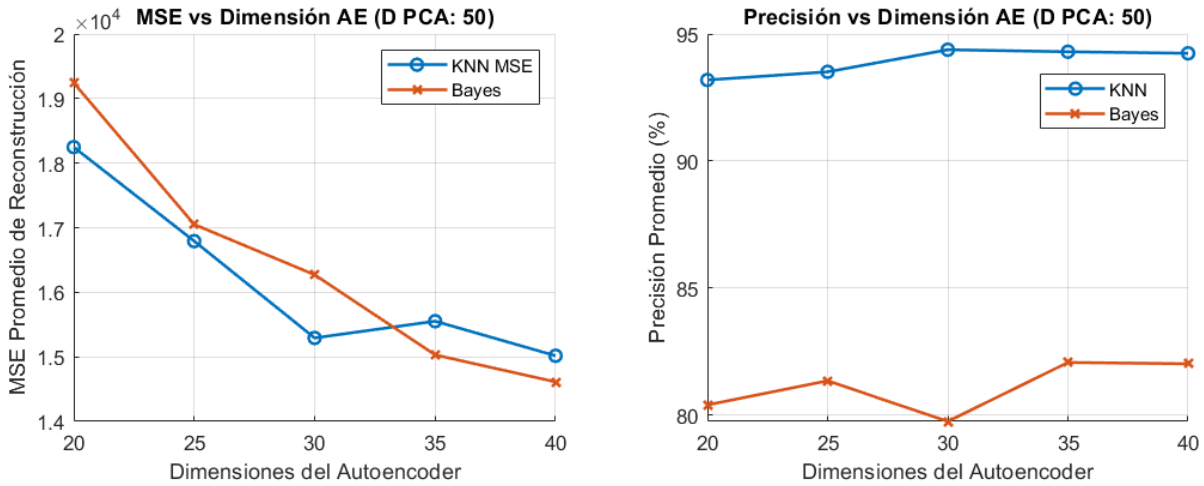

Figura 2.6: Comparativa de MSE (izquierda) y precisión de clasificación (derecha) para diferentes dimensiones del autoencoder, tras aplicar PCA con 50 componentes con k-NN y con Bayes.

2.4. Normalización de los datos

Se ha realizado un experimento para comprobar si estos datos son candidatos a una normalización que ayude a dar mejores resultados. Para ello, se ha comprobado el resultado del clasificador knn y bayesiano utilizados en esta entrega con los datos normalizados y no normalizados, así como comprobando cómo la reducción de dimensiones PCA afecta a ambos. En la Figura 2.9 se puede comprobar que los datos normalizados tienen mejores resultados, y que existe un número de dimensiones reducidas óptimas en torno a las 30-50. Se utilizarán, por tanto, los datos no normalizados, ya que se obtienen mejores resultados en ambos clasificadores.

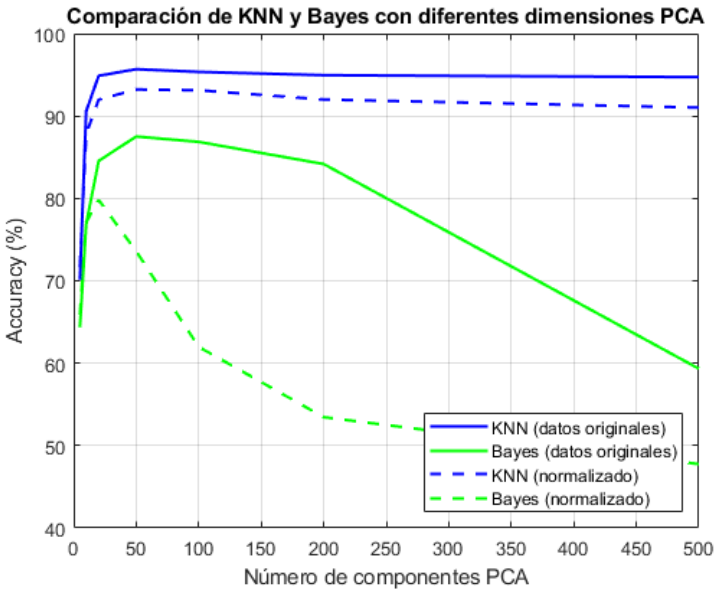

Figura 2.7: Precisión de los clasificadores con los datos normalizados y no normalizados, así como reducción de dimensionalidad de los datos usando PCA

2.5. Comparación de los tamaños del conjunto de entrenamiento y testing

Si bien no se trata de un preprocesamiento, se añade en esta sección la elección del tamaño de entrenamiento y testing antes de comenzar con la sección de clasificadores. Aunque en Inteligencia Artificial se suele utilizar un porcentaje de 80 % para el entrenamiento y 20 % para el testing, en la Figura 2.10 se observa cómo para estos datos se obtienen mejores resultados con una división de 90 % y 10 %, por lo que esta será la utilizada.

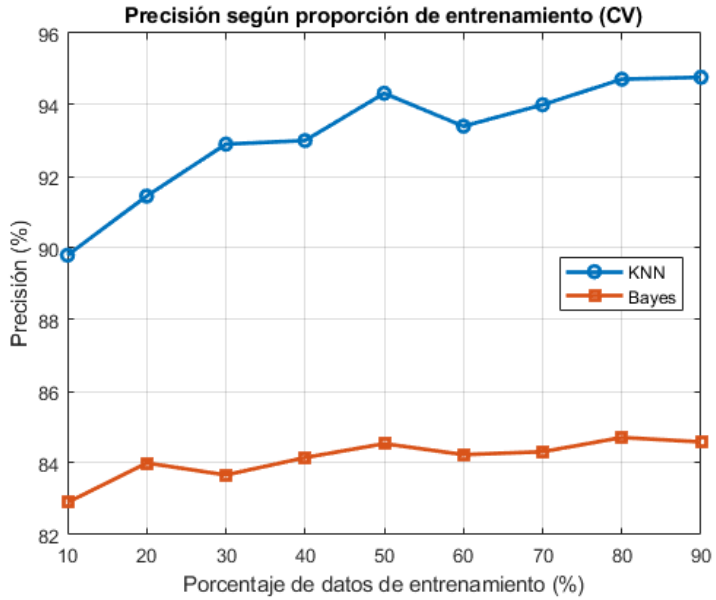

Figura 2.8: Precisión de los clasificadores con distintos tamaños de los conjuntos de entrenamiento y testing

2.6. Normalización de los datos

Se ha realizado un experimento para comprobar si estos datos son candidatos a una normalización que ayude a dar mejores resultados. Para ello, se ha comprobado el resultado del clasificador knn y bayesiano utilizados en esta entrega con los datos normalizados y no normalizados, así como comprobando cómo la reducción de dimensiones PCA afecta a ambos. En la Figura 2.9 se puede comprobar que los datos normalizados tienen mejores resultados, y que existe un número de dimensiones reducidas óptimas en torno a las 30-50. Se utilizarán, por tanto, los datos no normalizados, ya que se obtienen mejores resultados en ambos clasificadores.

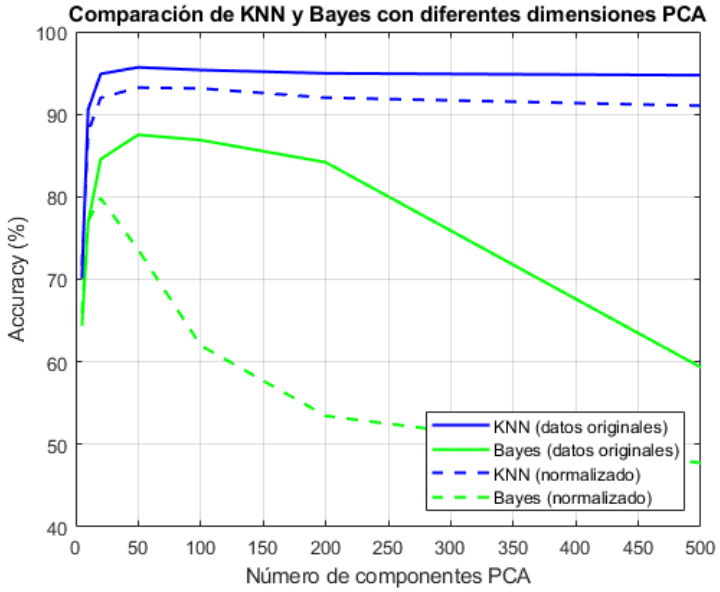

Figura 2.9: Precisión de los clasificadores con los datos normalizados y no normalizados, así como reducción de dimensionalidad de los datos usando PCA

2.7. Comparación de los tamaños del conjunto de entrenamiento y testing

Si bien no se trata de un preprocesamiento, se añade en esta sección la elección del tamaño de entrenamiento y testing antes de comenzar con la sección de clasificadores. Aunque en Inteligencia Artificial se suele utilizar un porcentaje de 80 % para el entrenamiento y 20 % para el testing, en la Figura 2.10 se observa cómo para estos datos se obtienen mejores resultados con una división de 90 % y 10 %, por lo que esta será la utilizada.

3. Clasificadores

En esta sección discutiremos y compararemos los resultados obtenidos por distintos clasificadores. Para evaluar cada uno de los métodos de forma fiable, se ha seguido un procedimiento de validación cruzada, separando en cada iteración los datos originales en un 90 % de datos de entrenamiento y un 10 % de testing. Para poder reproducir los resultados, la división de los números se ha hecho usando funciones aleatorias con una semilla.

Cada clasificador aquí presentado ha sido entrenado desde cero con los datos de entrenamiento y luego su precisión ha sido evaluada sobre los de testing. Esto se ha repetido diez veces dividiendo

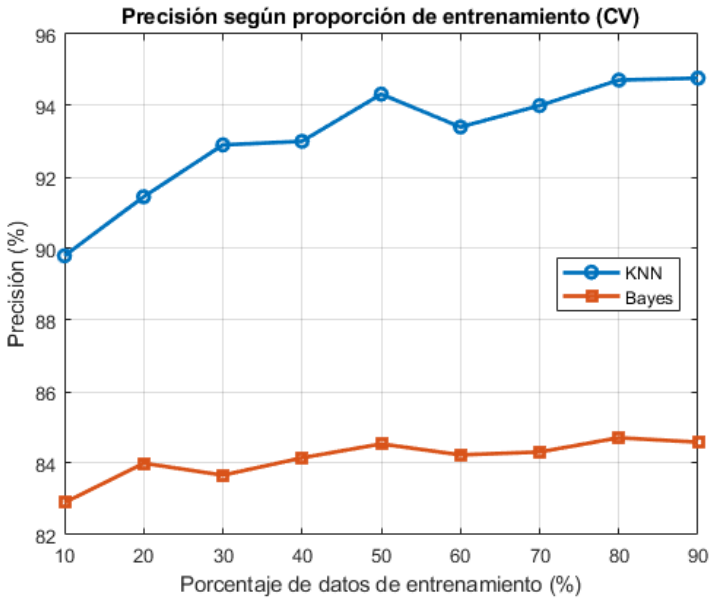

Figura 2.10: Precisión de los clasificadores con distintos tamaños de los conjuntos de entrenamiento y testing

de forma distinta los datos en cada iteración y se han promediado los resultados. Es de esperar que, al utilizar todos los datos de nuevo de cara a la prueba final del trabajo, los resultados puedan mejorarse ligeramente. Se espera que las conclusiones y predicciones extraídas de este informe se ajusten bien a los resultados que se obtendrán al evaluar los modelos con los datos de prueba al final de la asignatura.

3.1. k-NN (Tarea 2)

Para utilizar el clasificador k-NN, antes se ha reducido la dimensionalidad del espacio utilizando, en primer lugar, PCA. En base a los datos mostrados en la figura 2.3, se ha decidido proyectar sobre un espacio de dimensión 20. Para escoger el valor del parámetro k, se han probado distintas opciones entre 1 y 11. La precisión obtenida en cada caso se muestra en la figura 3.1. Como puede verse, la elección óptima es $k = 5$. Con esta elección de parámetros, se ha alcanzado una precisión del 94.85 %. La matriz de confusión obtenida en la fase de evaluación se muestra en la figura 3.2. Como puede verse, se producen más falsos positivos con el 9 que con las demás clases.

Se pasa ahora a aplicar k-NN tras reducir la dimensionalidad del problema utilizando LDA. En base a los resultados mostrados en la figura 2.4, se ha escogido dividir cada clase en 4 subclases antes de proyectar, tal y como se detalla en la sección 2.2. La k elegida se ha mantenido en 5. La precisión alcanzada por el modelo ha sido de 95.70 % y la matriz de confusión se muestra en la figura 3.3. Como puede verse, en este caso no ha habido una mejora significativa con respecto al PCA (aunque sí que se ha conseguido reducir los falsos positivos de la clase 9), mientras que la dimensionalidad de los datos se ha duplicado (recordemos que en PCA el subespacio tenía dimensión 20 y ahora tiene 39). Por este motivo, no parece adecuado utilizar LDA junto con un clasificador k-NN.

Por último, se han utilizado ambos métodos, haciendo una reducción de dimensiones con PCA previa al LDA. La salida del PCA se ha dado sobre un subespacio de dimensión 50. Posteriormente, se ha hecho clustering separando cada clase en 3, y se ha llevado a cabo el LDA, dando lugar a un subespacio final de trabajo de dimensión 29. Con esto, y manteniendo los datos iguales

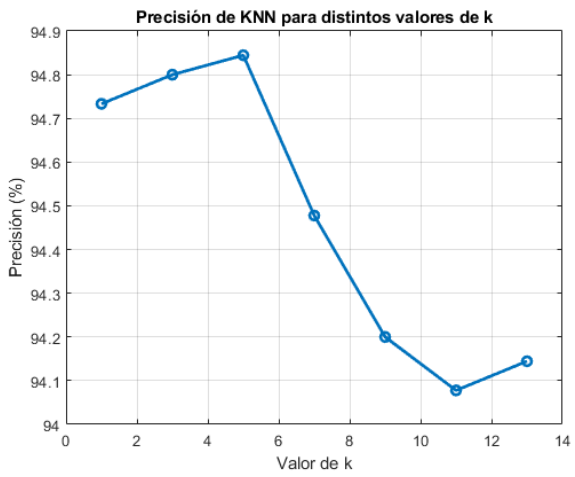

Figura 3.1: Precisión del algoritmo k-NN en función del parámetro k.

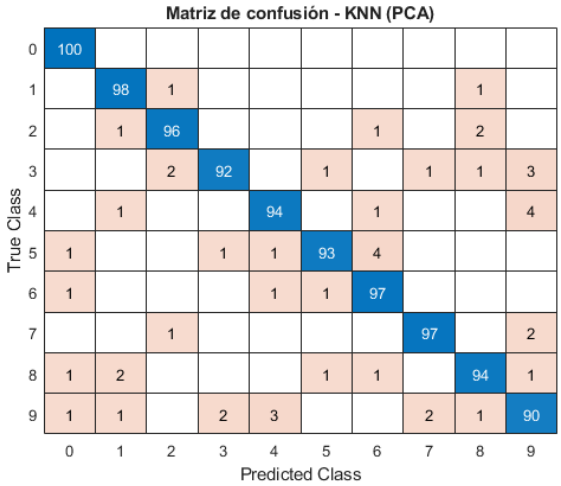

Figura 3.2: Matriz de confusión del clasificador k-NN tras PCA.

que en el PCA y LDA anteriores, se obtiene una precisión de 95.58 % y la matriz de confusión de la Figura 3.4.

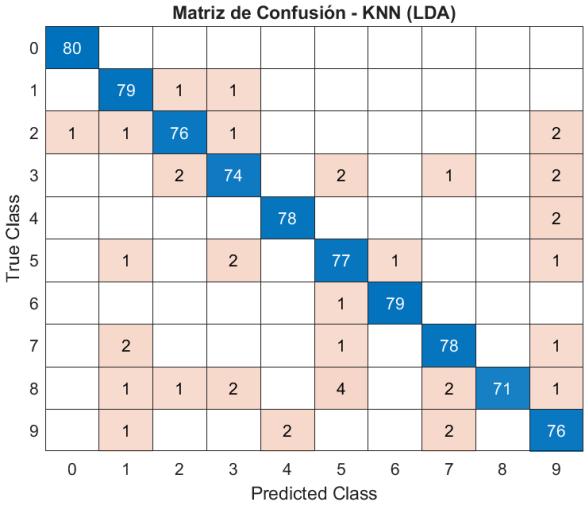

Figura 3.3: Matriz de confusión del clasificador k-NN tras haber reducido la dimensionalidad de los datos usando LDA.

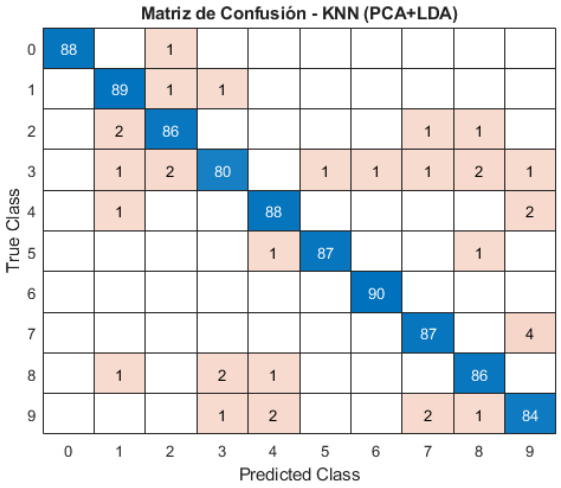

Figura 3.4: Matriz de confusión del clasificador k-NN tras haber reducido la dimensionalidad de los datos usando PCA y LDA.

3.2. Clasificador Bayesiano (Tarea 3)

Se comienza aplicando el clasificador bayesiano tras reducir la dimensionalidad mediante PCA. Siguiendo la misma configuración empleada para k-NN, se ha proyectado el conjunto de datos sobre un subespacio de dimensión 20. Con esta configuración, la precisión obtenida fue del 86.76 %. La matriz de confusión correspondiente se muestra en la Figura 3.5. Como puede observarse, los errores más frecuentes se presentan entre las clases 3 y 5, así como entre las clases 4 y 9. A pesar de estos errores, se presenta una alta tasa de acierto.

Posteriormente, se ha aplicado LDA utilizando 4 subclases por clase, conforme a la configuración óptima determinada en la sección 2.2. La precisión alcanzada en este caso fue del 86.70 %, ligeramente inferior a la obtenida con PCA. La matriz de confusión se muestra en la Figura 3.6.

Se observa una mejora en la clasificación de ciertas clases como el 7 y el 9, pero también un incremento de errores en clases como el 3 y el 8. En general, el rendimiento es similar, aunque ligeramente inferior en precisión global.

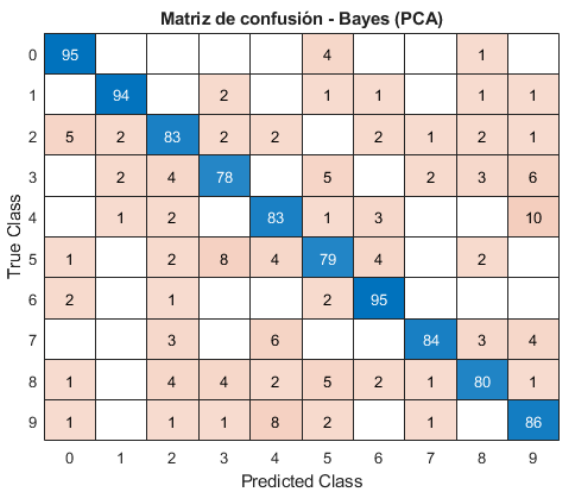

Figura 3.5: Matriz de confusión para Bayes con PCA (precisión 86.76 %)

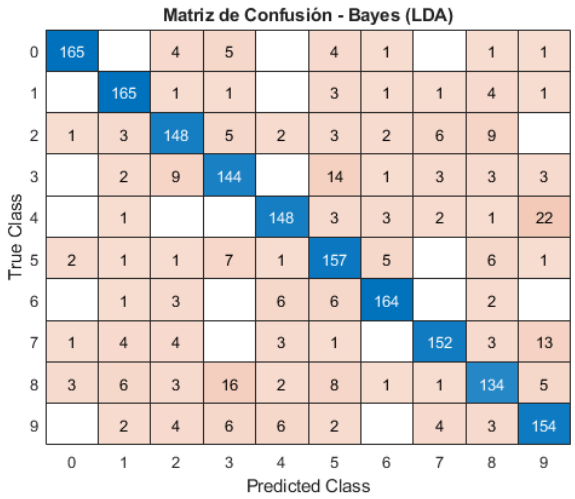

Figura 3.6: Matriz de confusión para Bayes con LDA (precisión 86.70 %)

Finalmente, se ha probado una combinación secuencial de PCA y LDA. En este caso, primero se ha reducido a 50 dimensiones mediante PCA y, posteriormente, se ha aplicado LDA tras dividir cada clase en 3 subclases. Esta configuración ha permitido alcanzar una precisión del 86.76 %, igualando el rendimiento obtenido con PCA por sí solo. La matriz de confusión se muestra en la Figura 3.7. Aunque no se ha logrado una mejora en la precisión total, sí se ha observado una distribución de errores más equilibrada. Las confusiones tienden a estar menos concentradas en pares específicos de clases, lo que sugiere un comportamiento más robusto y generalizado del clasificador.

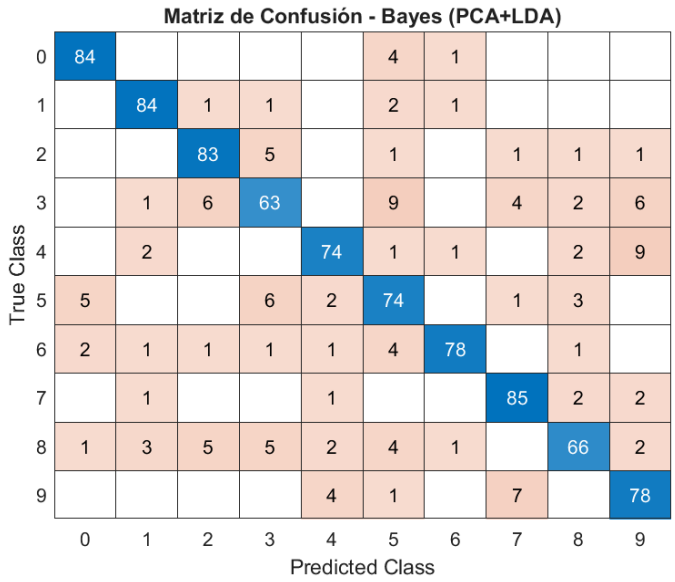

Figura 3.7: Matriz de confusión para Bayes con PCA+LDA (precisión 86.76 %)

3.3. MLP (Tarea 4)

Se estudia en esta sección la posibilidad de usar un perceptrón multicapa en la tarea de clasificación. Para ello se propone realizar previamente utilizar una reducción de dimensionalidad mediante PCA. También puede ser de interés normalizar los datos de entrada para que la intensidad de los bits se encuentre entre 0 y 1. Para elegir la dimensión del espacio reducido y decidir si aplicar o no la normalización, se han probado distintas redes con 1 sola capa intermedia, trabajando con distinta dimensionalidad del espacio y con datos normalizados y sin normalizar. Los resultados se muestran en la figura 3.8.

Además, para elegir la mejor arquitectura posible y el número adecuado de épocas de entrenamiento, se han probado distintos modelos y se ha representado la precisión obtenida por cada uno en función del número de épocas. Los resultados se muestran en la figura 3.9

Tal y como se hizo en apartados anteriores, se ha probado también a reducir la dimensionalidad empleando LDA y una combinación de PCA y LDA. Tras llevar a cabo un análisis similar al aquí mostrado para el PCA, se ha concluido que no hay diferencias significativas en cuanto a precisión entre los tres métodos.

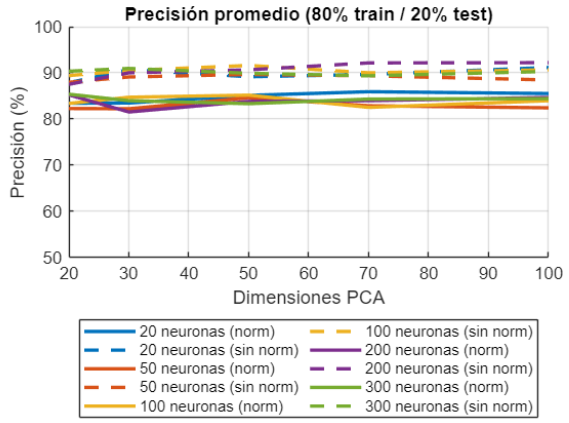

Figura 3.8: Precisión de distintos MLP en función de la dimensión del espacio reducido tras aplicar PCA normalizando y sin normalizar los datos.

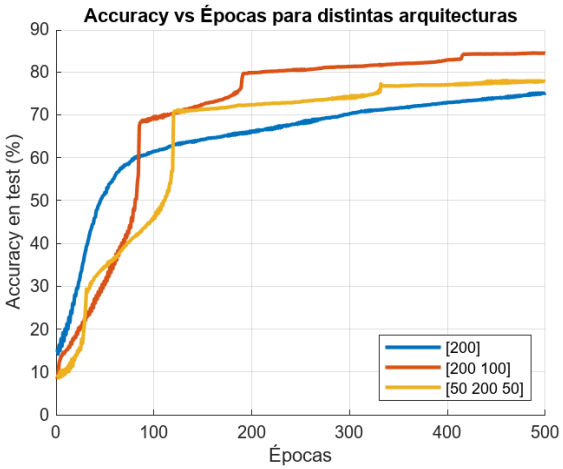

Figura 3.9: Precisión de tres redes neuronales MLP de distintas arquitecturas (número de neuronas por capa en la leyenda) en función de las épocas de entrenamiento.

En base a la información de la figura 3.8, se ha decidido no normalizar los datos (ya que, sistemáticamente, se obtiene una mayor precisión al no hacerlo). Y se ha concluido que la mejor dimensión del subespacio de trabajo es 70 (es cierto que hay redes que funcionan mejor con menos, pero el mejor rendimiento obtenido lo alcanza la red de 200 neuronas en este espacio). En base a la información de la figura 3.9, se ha decidido que la mejor opción es implementar un MLP de dos capas intermedias en vez de solo 1, con 200 y 100 neuronas cada una. También se ha observado que lo ideal es entrenarlo en torno a 450 épocas. A partir de este punto los resultados no mejoran significativamente y el tiempo de entrenamiento sigue aumentando.

Con estas especificaciones, se ha entrenado el MLP final, utilizando validación cruzada con 5 divisiones de los datos para garantizar la robustez de los resultados. La precisión promedio final obtenida ha sido del 95.07 %. En la figura 3.10 se muestra la matriz de confusión de dicho modelo.

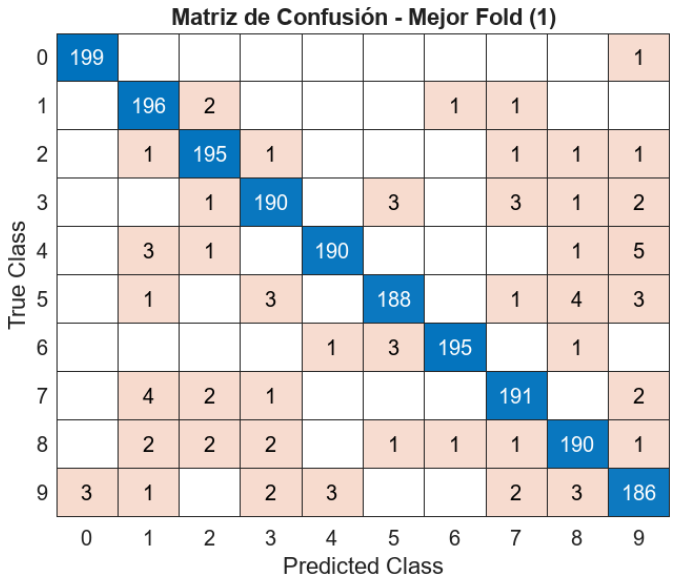

Figura 3.10: Matriz de confusión de un MLP de dos capas con 200 y 100 neuronas respectivamente, entrenado durante 450 épocas tras reducir la dimensionalidad de los datos hasta 70 usando PCA (precisión 95.07 %).

3.4. SOM (Tareas 5 y 10)

El objetivo original de las SOMs (self organizing maps) no es realmente ser utilizadas como clasificadores, sino mapear los datos desde un espacio de dimensión elevada a uno bidimensional. Sin embargo, podemos construir una red SOM utilizando los datos proporcionados y asignando a cada neurona una etiqueta en base al dato más próximo a ella que encontremos en el espacio original. Para clasificar un nuevo dato, puede utilizarse la etiqueta de la neurona más cercana. Con este método puede construirse un clasificador que proporciona un rendimiento aceptable.

Para construir el clasificador, debemos primero reducir la dimensionalidad del espacio con alguno de los métodos ya vistos y después elegir un número de neuronas adecuado para la red. Para encontrar el mejor método de reducción de la dimensionalidad y el mejor número posible de neuronas, se han llevado a cabo una serie de experimentos cuyos resultados se muestran en la figura 3.11. Lo que se ha hecho es dividir los datos proporcionados en un set de entrenamiento (80 %) y otro de test (20 %). Los datos de entrenamiento han sido proyectados sobre subespacios de dimensionalidad reducida utilizando cada uno de los métodos propuestos (PCA, LDA, PCA+LDA) y en cada caso se han probado múltiples tamaños de la red SOM para ver con cuál puede obtenerse una mayor precisión al clasificar los datos de test.

Lo primero que puede concluirse en base a estos resultados es que el PCA resulta más conveniente en este caso. También parece que, usando este método, la precisión del clasificador crece asintóticamente hasta un límite desconocido que ronda el 90 %. Dado que la red es una malla cuadrada, el número de neuronas crece cuadráticamente. Esto puede suponer un problema porque hace que los tiempos de entrenamiento aumenten demasiado. Con una malla de 20×20 neuronas, el balance entre precisión y tiempos de entrenamiento se ha considerado adecuado.

Para determinar cuál es la mejor dimensión del subespacio sobre el que proyectar con PCA se ha hecho el experimento cuyos resultados se muestran en la figura 3.12. Como puede verse, a partir de 25 dimensiones, las diferencias no son significativas (pero los tiempos de entrenamiento sí). Por este motivo, se reducirá la dimensión únicamente a 25.

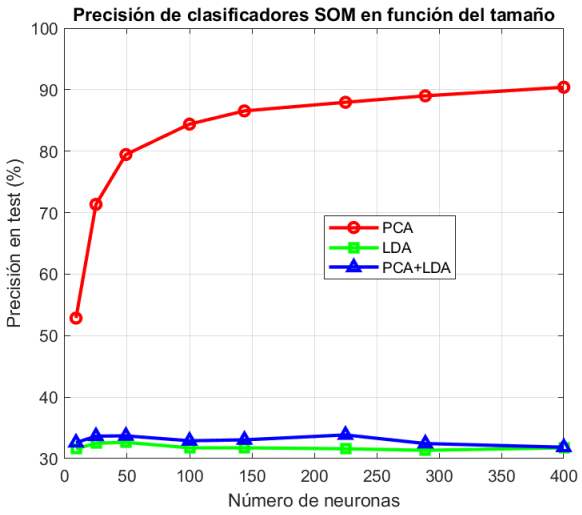

Figura 3.11: Precisión del clasificador SOM en función del número de neuronas de la red tras reducir la dimensionalidad del espacio aplicando PCA (dimensión del espacio reducido: 150), LDA y PCA+LDA.

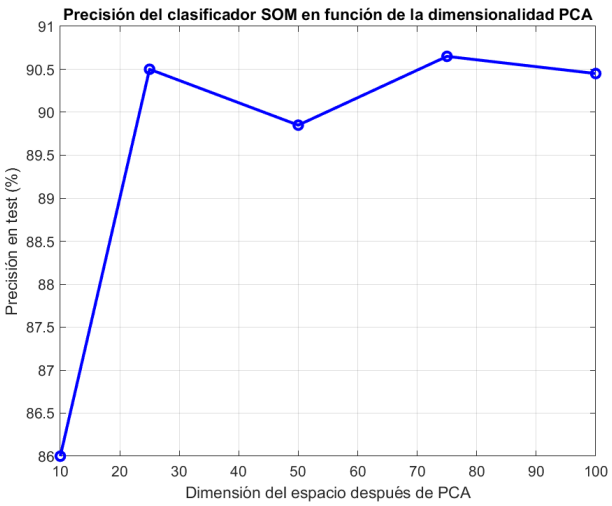

Figura 3.12: Precisión del clasificador SOM de 20×20 neuronas en función de la dimensión del subespacio sobre el que se proyecten previamente los datos usando PCA.

Con las especificaciones mencionadas anteriormente, finalmente se ha construido un clasificador capaz de alcanzar un 90.6 % de precisión en la clasificación de los datos de test. La matriz de confusión de dicho clasificador se muestra en la figura 3.13.

También resulta interesante visualizar la distribución de los datos en las proximidades de la red (en el espacio de salida). Con este objetivo se ha construido la figura 3.14 (tarea 10), en la que se muestran las imágenes asociadas a los datos que han servido para etiquetar cada una de las neuronas de la red. Esto da una idea intuitiva de cómo las neuronas se han agrupado en torno a los números en el espacio de dimensión 25 y de cómo ligeras deformaciones en los dígitos manuscritos (trazos más rectos, gruesos, diagonales, etc.) posicionan el dato en un punto distinto de dicho espacio.

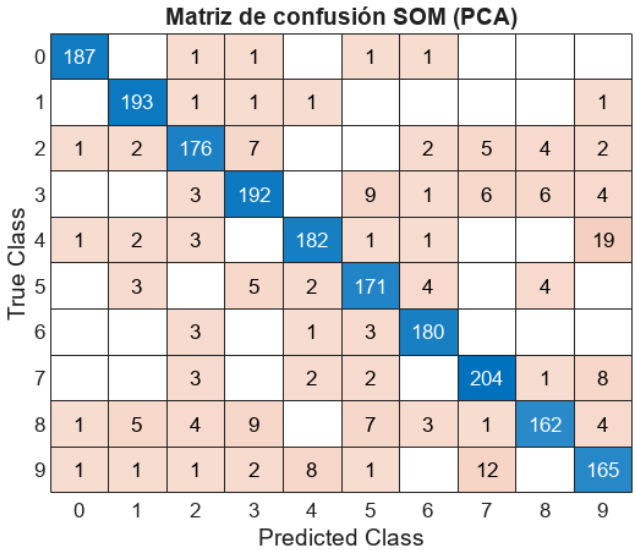

Figura 3.13: Matriz de confusión de un clasificador basado en una red SOM de 20×20 neuronas entrenado durante 200 épocas tras reducir la dimensionalidad de los datos hasta 25 usando PCA (precisión 90.60 %).

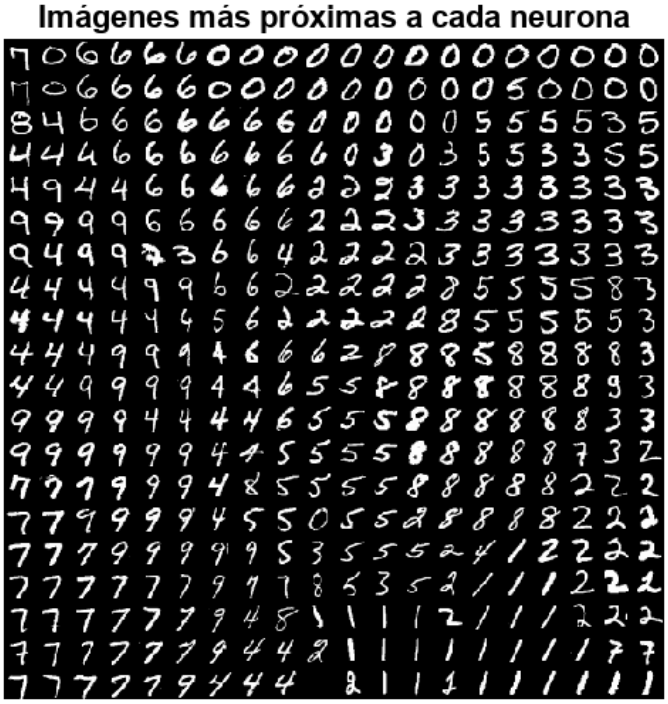

Figura 3.14: Grid 20×20 que muestra las imágenes más próximas a cada una de las neuronas de la red SOM en el espacio de salida (tarea 10). Estas son las imágenes que se emplean para asignar etiquetas a dichas neuronas, permitiendo así usar la red SOM como clasificador (aunque no sea su propósito original).

3.5. Red profunda (DL network) (Tarea 6)

Antes del entrenamiento, las imágenes se reconstruyeron a tamaño 28×28 y se normalizaron dividiendo por 255 para que sus valores estuviesen en el rango [0,1]. Las etiquetas fueron convertidas a variables categóricas con 10 clases. El conjunto completo de datos se dividió en un 80 % para entrenamiento y un 20 % para prueba, y sobre el conjunto de entrenamiento se aplicó *data augmentation* consistente en pequeñas rotaciones y desplazamientos aleatorios, con el objetivo de mejorar la generalización del modelo.

Se ha implementado una red neuronal convolucional profunda (CNN) con tres capas convolucionales, normalización por lotes, funciones de activación `ReLU`, `max pooling`, y capas densas con `dropout` y regularización L2. El objetivo ha sido maximizar la precisión de clasificación sobre los dígitos manuscritos, optimizando los hiperparámetros mediante una búsqueda sistemática.

Para evaluar cada configuración de forma robusta, se ha realizado validación cruzada con múltiples particiones de los datos (cross-validation), entrenando la red desde cero en cada fold y calculando la precisión y los tiempos promedio. En particular, se utilizaron 5 folds.

Se han analizado los efectos de distintos valores para la **tasa de aprendizaje** (`LearningRate`), el **número de épocas**, el **tamaño de lote** (`MiniBatchSize`) y la cantidad de filtros en la primera capa convolucional. Asimismo, se ha comparado el rendimiento entre optimizadores (Adam y SGDM), aplicando estrategias de *learning rate drop*.

Los principales resultados se resumen en la Tabla 3.1. Las observaciones clave fueron las siguientes:

- Una tasa de aprendizaje excesivamente baja (`LR = 0.0001`) condujo a menor precisión, debido a una convergencia muy lenta.
- Aumentar el número de épocas mejora la precisión, aunque a costa de un tiempo de entrenamiento significativamente mayor.
- Los lotes pequeños (64 muestras) aportan mayor precisión, pero incrementan el coste temporal.
- La mejor configuración en cuanto a equilibrio entre precisión y eficiencia fue obtenida con el optimizador Adam y una tasa de aprendizaje de 0.0007 durante 20 épocas. Aplicando validación cruzada con 5 particiones, se alcanzó una precisión promedio del **98.67 %** (desviación estándar: 0.31 %), con un tiempo medio de entrenamiento de 334.01 segundos por fold y un tiempo de clasificación de 0.574 segundos.

Tabla 3.1: Comparativa de configuraciones representativas de CNN (media tras validación cruzada)

Optimizador	**LR**	**Épocas**	**Precisión (%)**	**T. Training (s)**	**T. Test (s)**
Adam	0.0001	15	94.45	180.6	0.69
Adam	0.0005	15	97.50	183.2	0.56
Adam	0.0007	10	97.75	128.6	0.40
SGDM	0.005	25	98.52	726.54	0.76
Adam	**0.0007**	**20**	**98.67**	**334.0**	**0.57**

La Figura 3.15 muestra el progreso del entrenamiento para la mejor red, donde puede observarse cómo la precisión se estabiliza en valores cercanos al máximo alcanzado y el error de validación desciende progresivamente.

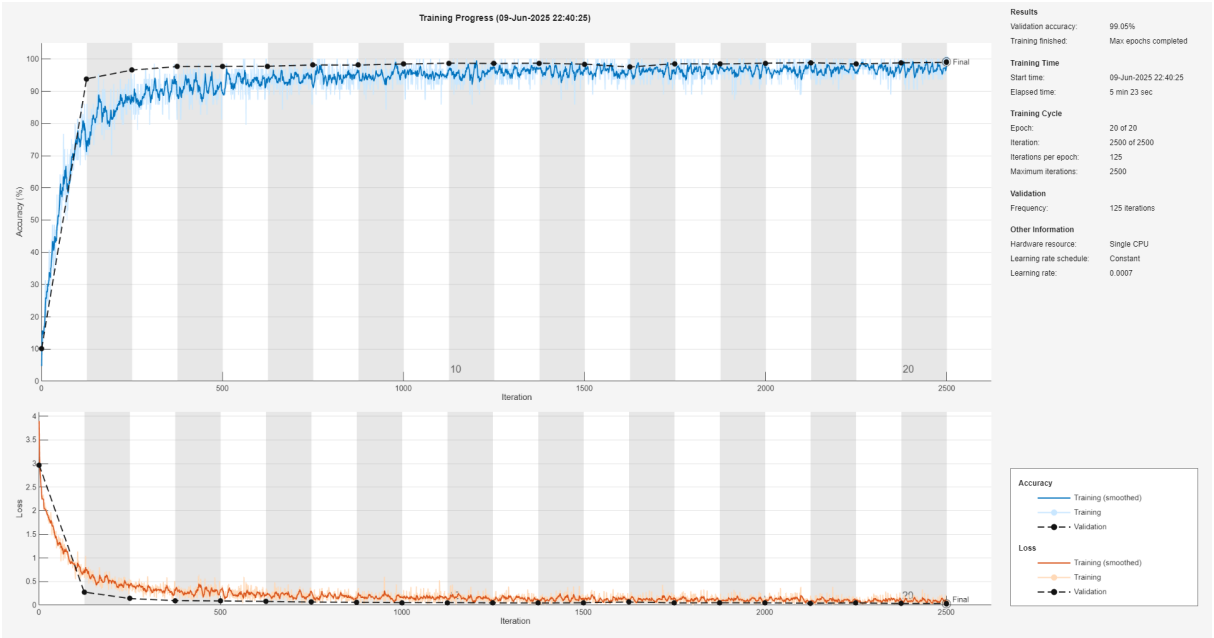

Figura 3.15: Progreso del entrenamiento de la mejor CNN (Adam, LR = 0.0007, 20 épocas)

El perceptrón multicapa (MLP) alcanzó una precisión promedio del 95.07 %, frente al 98.67 % logrado por la CNN. Esta mejora refleja la capacidad superior de las redes convolucionales para extraer patrones espaciales complejos. Aunque el coste computacional es mayor, el rendimiento obtenido justifica su uso en tareas de clasificación exigentes.

4. Conclusiones

La Tabla 4.1 presenta un resumen cuantitativo del rendimiento de los métodos evaluados. Adicionalmente, estos datos pueden visualizarse en las Figuras 4.1 y 4.2.

En lo referente a los métodos elegidos para reducir la dimensionalidad antes de aplicar el clasificador k-NN, la figura 4.1 parece indicar que ninguno supone una mejora significativa con respecto a los demás en términos de precisión. No obstante, como puede observarse en la figura 4.2, los tiempos de entrenamiento son menores al utilizar PCA. Además, a pesar de que en los tres casos principales (PCA, LDA y PCA+LDA) se obtiene una precisión muy similar, la reducción mediante PCA permite trabajar con un subespacio de menor dimensión, como se discutió en la sección 3.1, lo cual es más eficiente en memoria. En cuanto a la opción PCA + autoencoder, aunque introduce una transformación no lineal adicional, no mejora los resultados de precisión y supone un incremento significativo en el tiempo de preprocesado, lo que descarta su uso práctico en este contexto.

El mismo análisis puede hacerse para el clasificador bayesiano. De nuevo, no hay una estrategia de reducción de dimensionalidad que proporcione una ventaja significativa en cuanto a la precisión alcanzada (ver figura 4.1) pero es cierto que el tiempo de ejecución es más eficiente al utilizar PCA+LDA (figura 4.2), seguramente debido a que la dimensión del subespacio final es menor. Esto puede suponer una ventaja en aplicaciones on-line o al ejecutar el software en plataformas con bajos recursos computacionales.

Comparando ambos clasificadores, está claro que k-NN tiene una mayor precisión y tiempos de entrenamiento y ejecución más óptimos (ver figuras 4.1 y 4.2). Esto último es algo inesperado, puesto que el tiempo de test del clasificador bayesiano debería ser inferior. Es posible que esta discrepancia esté relacionada con el funcionamiento interno de las funciones de MATLAB y no con la lógica de los algoritmos en sí, por lo que tal vez las cosas podrían cambiar si se utilizase otro software.

El clasificador MLP ha demostrado ser una alternativa eficaz, superando en precisión a los métodos anteriores. En concreto, el uso de una arquitectura de dos capas (200 y 100 neuronas) entrenada durante 450 épocas ha alcanzado una precisión del 95.07 %. Aunque sus tiempos de entrenamiento son considerablemente mayores, la rapidez en la fase de clasificación y su buena capacidad de generalización lo convierten en una opción muy válida.

En cuanto al uso de SOM, a pesar de no estar diseñadas inicialmente para clasificación, han alcanzado un rendimiento aceptable del 90.60 % con una malla de 20×20 y reducción de dimensionalidad con PCA. No obstante, sus resultados son inferiores a los de otros clasificadores más especializados.

Finalmente, la red neuronal profunda (DLN) ha obtenido los mejores resultados, con una precisión del 98.67 % tras aplicar técnicas de data augmentation y optimización de hiperparámetros. A pesar de requerir un tiempo de entrenamiento significativamente superior, su rendimiento en la clasificación justifica su uso. La capacidad de las CNN para captar patrones espaciales complejos ha demostrado ser muy superior a la de otros modelos como MLP.

Dado que la tarea final no requiere ejecución en tiempo real, la prioridad es maximizar la precisión. Siguiendo este razonamiento, el mejor modelo entrenado ha sido la CNN con optimizador Adam y tasa de aprendizaje de 0.0007, lo que posiciona a las redes profundas (DLN) como la mejor alternativa en este trabajo.

Método	**Precisión (%)**	**T. Entrenamiento (s)**	**T. Clasificación (s)**
PCA + k-NN	94.98	0.004	0.021
LDA + k-NN	95.70	0.004	0.033
PCA + LDA + k-NN	95.58	0.004	0.023
PCA + AE + k-NN	94.38	0.016	0.2
PCA + Bayes	87.20	0.020	0.034
LDA + Bayes	86.70	0.016	0.027
PCA + LDA + Bayes	86.76	0.019	0.023
PCA + AE + Bayes	82.42	1.442	19.782
PCA + MLP	95.07	48.47	0.020
LDA + MLP	94.95	56.19	0.020
PCA + LDA + MLP	94.66	42.61	0.020
PCA + SOM	90.60	48.67	0.030
LDA + SOM	33.50	18.00	0.023
PCA + LDA + SOM	32.92	18.32	0.020
DLN	98.67	334.0	0.57

Tabla 4.1: Comparativa de métodos de clasificación

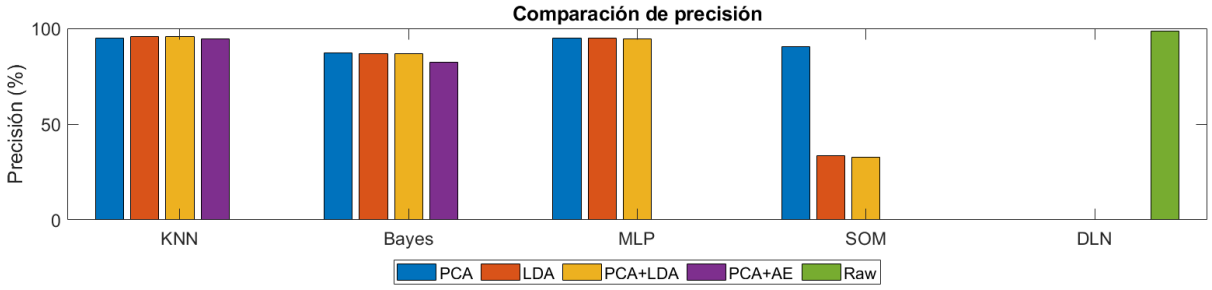

Figura 4.1: Comparación de la precisión de los distintos algoritmos.

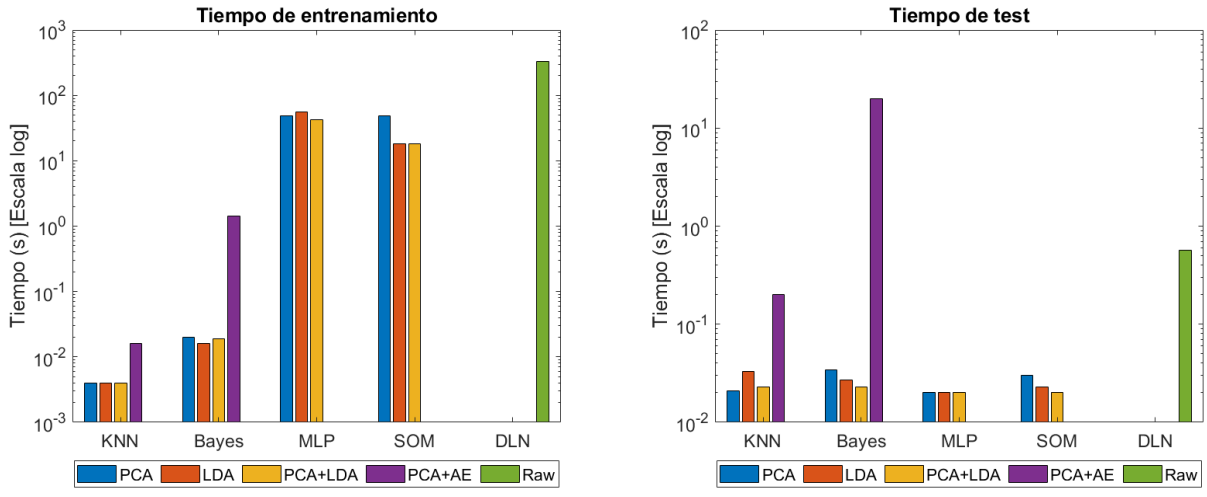

Figura 4.2: Comparación de los tiempos de entrenamiento y ejecución de los distintos algoritmos (escala logarítmica).