UNIVERSIDAD POLITÉCNICA DE MADRID

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES

MÁSTER EN AUTOMÁTICA Y ROBÓTICA

RECONOCIMIENTOS DE DÍGITOS ESCRITOS A MANO MEDIANTE DIFERENTES TÉCNICAS DE MACHINE LEARNING

ASIGNATURA:

INTELIGENCIA ARTIFICIAL APLICADA

GRUPO 18

AUTORES:

Álvaro Benito Oliva	M20159
Germán Andrés Di Fonzo Caturegli	M20037
Juan José Jurado Camino	M20039

Madrid a 21 de junio de 2021

TABLA DE CONTENIDOS

1 INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO

Machine Learning es un subcampo de las ciencias de la computación y una rama de la Inteligencia Artificial que tiene como objetivo crear sistemas (máquinas) capaces de aprender de manera automática. Esto último se refiere a que el desempeño de la máquina mejora con la experiencia de forma totalmente autónoma, sin intervención humana. Para ello, los investigadores de Machine Learning desarrollan algoritmos y heurísticas que trabajan con millones de datos y generan programas de computadoras, sin tener que escribir estos últimos explícitamente. El Machine Learning está muy relacionado con el reconocimiento de patrones de una multitud de muestras para predecir comportamientos futuros y clasificar datos.

El objetivo principal de este trabajo consiste en realizar una investigación que permita conocer las principales ventajas y desventajas de las diferentes técnicas de Machine Learning explicadas durante el curso de la asignatura Inteligencia Artificial Aplicada para el reconocimiento de dígitos escritos a mano. Para ello, es necesario cumplir con los siguientes objetivos específicos:

- Realizar un preprocesamiento de las características de los dígitos para reducir la dimensionalidad de los datos de entrada mediante un análisis de componentes principales (PCA) y, de este modo, optimizar los algoritmos de clasificación.
- Entrenar diferentes algoritmos de Machine Learning para la clasificación de dígitos escritos a mano: k-NN, clasificador bayesiano, redes neuronales artificiales supervisadas (MLP) y mapas autoorganizados (SOM).
- Utilizar el discriminante lineal de Fisher para la reducción de la dimensionalidad de datos de entrada como alternativa al PCA.
- Emplear el clasificador no supervisado K-means para la clasificación de los dígitos de cada clase en varias subclases, con el objetivo de mejorar los resultados obtenidos.
- Combinar un red neuronal Autoencoder junto con el PCA para la reducción de la dimensionalidad de los datos de entrada y comparar los resultados obtenidos con los que se consigue cuando solo se emplea un PCA para dicha tarea.
- Construir un mapa visual utilizando redes neuronales de tipo SOM donde cada patrón de imagen se encuentre en una posición de cada neurona del mapa 2D.

MNIST es una base de datos de dígitos escritos a mano representados en imágenes de 28x28 píxeles en una escala de grises. Cada píxel es una característica utilizada para definir el dígito que representa la imagen. Este proyecto, hace uso de una parte de esta base de datos, que solo contiene números con diferentes topologías desde el 0 al 9. El tamaño de la base de datos utilizada es de 1.000 ejemplos por tipo de número, lo que hace un total de 10.000 datos.

Para el entrenamiento de los distintos modelos se ha decidido dividir la base de datos en 8.000 muestras de entrenamiento y 2.000 muestras de test con el fin de obtener una buena validación.

2 PREPROCESAMIENTO DE DATOS

Cuando se trabaja con inteligencia artificial, esta precisa de gran cantidad de datos para obtener su máximo partido y debido a esto se almacenan cantidades inmensas de información. Sin embargo, procesar toda esta información supone un costo de recursos muy alto e incluso mucha de esa información no es relevante para la aplicación u objetivo requerido.

El preprocesado o procesado de características es muy importante para obtener resultados de forma eficiente y rápida. Uno de los procesos que se puede realizar durante el preprocesado es la normalización de estos datos. Su finalidad es cambiar los valores del conjunto de datos para usar una escala común, sin distorsionar las diferencias en los intervalos de valores ni perder información. También algunos algoritmos necesitan que los datos que se proporcionan se encuentren correctamente normalizados. Existen diferentes técnicas para normalizar los datos como, dividir los datos entre un máximo y un mínimo empíricos, el método logístico o el método logarítmico. En este proyecto se hace uso del método denominado puntuación estática descrito por la siguiente función:

$$x_{in} = \frac{x_i - mean(x)}{\sigma(x)}$$

Este proceso centra los datos normalizados en el cero teniendo en cuenta la desviación típica del conjunto de datos para aportar invarianza en la escala. La elección de este método es debido a que funciona correctamente cuando los datos distribuidos semejantes a una normal, como es este caso.

Otra herramienta comúnmente utilizada en el preprocesado de datos es la reducción de la dimensionalidad, que consiste en reducir el número de variables a tener en cuenta en una colección de datos. Para entender este proceso, se puede pensar en la gran cantidad de pixeles negros invariantes que se encuentran en todas las imágenes de este proyecto. Estos píxeles se mantienen negros en todas las imágenes, por lo que son aportan información para diferenciar los números entre sí.

Otras razones por las que se recurre a la reducción de dimensionalidad son:

- No siempre el mejor modelo es el que más variables tiene en cuenta. De esta forma se consigue reducir el sobreajuste de un modelo.
- Se mejora el rendimiento computacional, traduciéndose en un ahorro en coste y tiempo.
- Se reduce la complejidad y con ello una compresión más sencilla de los resultados.

A continuación, se comentan los métodos de reducción de dimensionalidad utilizados en este proyecto y su implementación.

2.1 REDUCCIÓN DE DIMENSIONALIDAD MEDIANTE EL MÉTODO DE ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

El método de análisis de componentes principales es un método de transformación lineal, no supervisado. Este proceso permite identificar patrones en los datos basándose únicamente en su correlación. Consiste en encontrar las direcciones de máxima variación de los datos y proyectarlos en un subespacio más reducido que el original. La matriz de proyección está definida por los vectores de las direcciones de máximo cambio, mientras que los autovalores asociados a esos vectores definen la variación de los datos en esa dirección.

Según la dimensión del subespacio sobre el que se quieran proyectar los datos se cogen un número de vectores determinados. Para formar un subespacio de dimensión tres, hacen falta tres vectores sobre los que se proyecten los datos.

Al ser no supervisado, no se puede obtener un valor de precisión a priori. Sin embargo, como los autovalores definen la varianza sobre la dirección de sus vectores se puede obtener un valor del error teórico sumando los autovalores de las direcciones de proyección no escogidas. De esta forma se puede aproximar la cantidad de información que se está perdiendo al proyectar sobre las direcciones escogidas. Es necesario añadir que el PCA es el método óptimo para la reducción de dimensionalidad de forma lineal.

Para aplicar este proceso de reducción de dimensionalidad en el proyecto, se ha utilizado la función "*processpca*" de Matlab. Con el objetivo de buscar una dimensionalidad óptima para los datos de MNIST se ha realizado la Figura 1 que muestra cómo evoluciona el error cuadrático medio de reconstrucción de las imágenes, según el número de características utilizadas para reducir su dimensionalidad previamente.

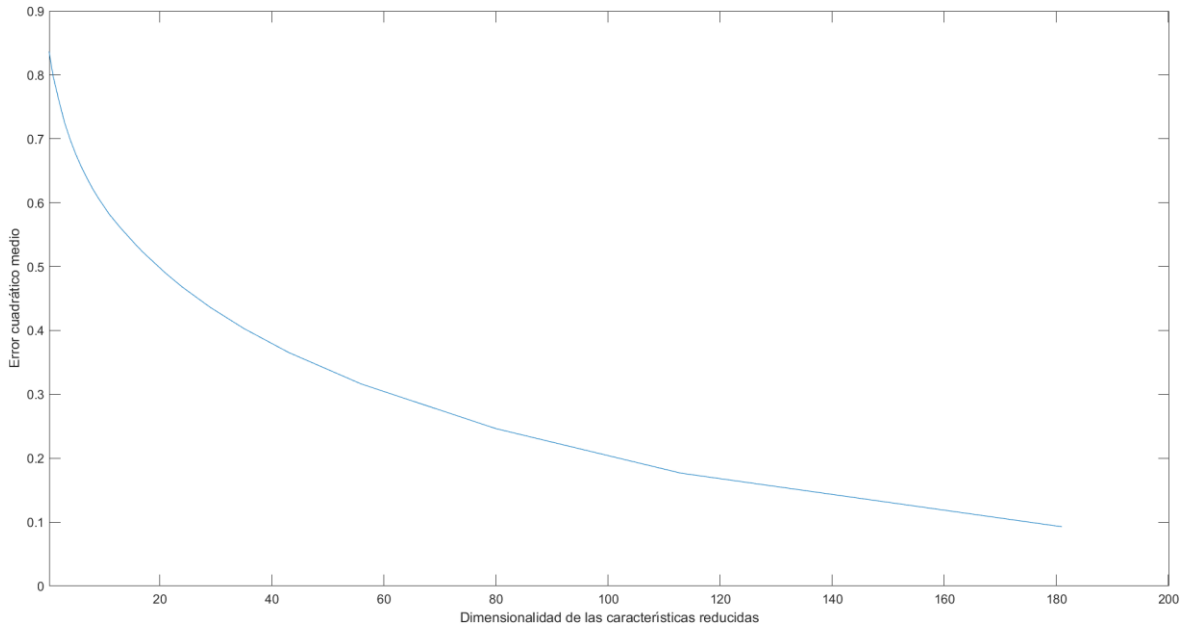

Figura 1.Evolución del MSE en función del número de características reducidas.

Como se puede apreciar, el valor de MSE se reduce exponencialmente con la dimensionalidad. Teniendo en cuenta que cuantas más características se utilicen, más lento funcionará el algoritmo, se determina que el número de características utilizadas óptimo debe encontrarse en el rango entre las 40 – 60 características. Para afinar aún más la elección de la dimensionalidad de las características reducidas, se han comprobado con diferentes valores de dimensionalidad para diferentes algoritmos de clasificación clásicos, (bayesiano y k-nn). Finalmente se decide utilizar un valor de dimensionalidad entorno las 50 características.

Figura 2. Reconstrucción con PCA 5, PCA 50, PCA 150 respectivamente.

Como se puede apreciar en la Figura 2, realizando una reducción de dimensionalidad se pierde información. El objetivo de la reducción de la dimensionalidad consiste en buscar un compromiso entre recursos para estudiar los datos, y el valor de la información que se pierde. Si se hace una reducción de dimensionalidad de 784 a 5 se pierde la noción de que es un 6, pero si se reduce de 784 a 50 se mantiene esta noción. Poder estudiar parámetros con un 6% de información supone agilizar el proceso de aprendizaje enormemente. Además, el hecho de alimentar los diferentes algoritmos de clasificación con datos mucho más flexibles dota de una mayor generalización al modelo obtenido. Con esto se consigue validar el uso del método del PCA para reducir la dimensionalidad.

2.2 REDUCCIÓN DE DIMENSIONALIDAD MEDIANTE AUTO-ENCODER

Los dos procesos anteriores son métodos lineales para la reducción de la dimensionalidad. El auto-encoder hace uso de las herramientas que ofrecen las redes neuronales para crear una representación de menor dimensión de los datos de forma no lineal. El concepto básico de este método consiste en entrenar una red neuronal que realice una compresión de los datos (codificador) y entrenar también otra red neuronal que realice la tarea de descompresión obteniéndose así de nuevo la representación de los datos inicial. Este método nace de la problemática del aprendizaje no supervisado, ya que no necesita de unos datos etiquetados, el objetivo es obtener en la salida, lo mismo que la entrada.

A la hora de diseñar un auto-encoder, se imponen en la capa oculta el tamaño de las dimensiones a las que se quieren reducir los datos. Es necesario escoger, entre otros parámetros, la función de activación para las neuronas que actuarán de encoder y las neuronas que actuarán de decoder. Si se escogiese una función de activación lineal se estaría repitiendo el proceso de PCA ya que es el método óptimo para la reducción de dimensionalidad lineal.

A continuación, se muestra una comparación de una reducción de dimensionalidad desde 784 a 51 únicamente por medio del PCA y otra forma por medio de una PCA de 784 a 184 (para facilitar el aprendizaje de las dos redes neuronales) y una posterior reducción de 184 a 51 con un auto-encoder. Mientras que mediante únicamente un PCA se obtiene un MSE de 0,3412 en la reconstrucción, cuando se realiza una primera etapa de PCA y posteriormente una de auto-encoder, se consigue un valor de MSE de 0,3627. Como se puede deducir, aplicar únicamente el PCA aporta más información correcta con el mismo número de características. Esto significa que la relación que tienen las características entre sí tiende a ser lineal y por tanto el PCA, que es el mejor método lineal, obtiene un mejor resultado por medio del auto-encoder (no lineal).

La estructura del auto-encoder diseñado es la que se puede observar en la Figura 3.

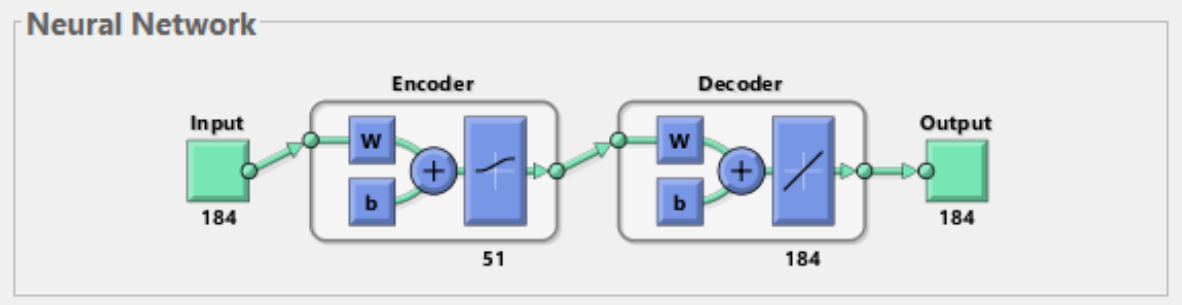

Figura 3. Estructura auto-encoder

En la Figura 4 se muestran la imagen original junto a las reconstrucciones realizadas por los dos métodos comentados anteriormente. En las reconstrucciones no se observa gran diferencia a simple vista, aunque se puede notar menos cantidad de ruido en la generada por el PCA.

Figura 4. Imagen original, reconstrucción mediante auto-encoder, reconstrucción mediante PCA.

El uso de auto-encoder suele utilizarse junto con un *softmax* para obtener un sistema clasificador. En este proyecto se contempla como una alternativa rápida y eficaz a los métodos de clasificación utilizados en este proyecto.

3 CLASIFICADOR K-NN

El algoritmo k-nn se basa en clasificar un dato de entrada en el grupo correspondiente de acuerdo con los k vecinos que tenga más cercanos en el espacio de entrada. Para ello, debe calcular las distancias con los otros elementos y escoger los vecinos más cercanos.

Para entrenar el clasificador k-nn, lo más importante es la obtención de unos buenos datos a partir de una reducción dimensional y la correcta selección del número de vecinos para evitar el problema del *overfitting* y lograr una buena precisión en los datos de test.

En primer lugar se ha estudiado la diferencia entre el uso de los datos normalizados y sin normalizar, con el fin de utilizar aquellos que proporcionen mejores resultados de clasificación. Para los mismos datos de entrenamiento, se han obtenidos los siguientes valores de clasificación de la Tabla 1:

	Entrenamiento	Test
Normalizado	96.03	91.50
Sin normalizar	97.53	94.15

Tabla 1. Errores para datos normalizados y sin normalizar del clasificador k-nn.

Para la obtención de estos resultados se ha utilizado la reducción de dimensionalidad con PCA y un modelo con 3 vecinos, concluyendo que es mejor utilizar los datos sin normalizar.

A continuación se han evaluado los valores obtenidos al usar distintas técnicas de reducción de la dimensionalidad. Las tres técnicas que se han evaluado son la PCA y el auto-encoder, obteniéndose los resultados de la Tabla 2.

	Acierto Entrenamiento (%)	Acierto Test (%)
PCA	97.53	94.15
Auto-encoder	96.39	92.55

Tabla 2. Errores para distintas técnicas de reducción de dimensionalidad.

Para cada una de ellas, el objetivo ha sido reducir la dimensionalidad iterativamente hasta obtener el mejor resultado de clasificación, ya que no todas las técnicas obtienen los mejores resultados para la misma reducción de dimensionalidad. Finalmente se ha decidido utilizar la PCA para este propósito por su mejor comportamiento con el clasificador.

A continuación, se han realizado varias pruebas para comprobar el número de vecinos más adecuados para el clasificador. Los resultados obtenidos para los distintos valores de vecinos “k” se recogen en la Tabla 3.

k vecinos	Acierto Entrenamiento (%)	Acierto Test (%)
1	100	94.50
2	97.50	93.75
3	97.33	95.55
4	96.96	94.90
5	96.65	93.95
6	96.05	94.60

Tabla 3. Errores para distinto número de vecinos.

A continuación se muestra en la Figura 5 la matriz de confusión obtenida con el mejor clasificador de k-nn que se ha logrado.

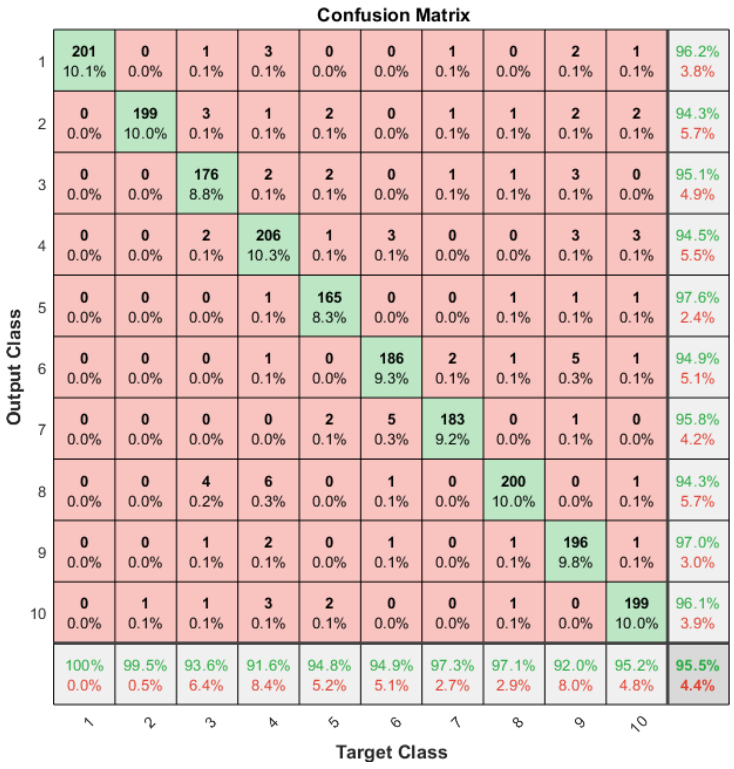

Figura 5 Matriz de confusión obtenida para la clasificación de dígitos con el clasificador k-nn.

4 CLASIFICADOR BAYESIANO

El clasificador bayesiano es un clasificador probabilístico fundamentado en el teorema de Bayes, se utiliza como clasificador supervisado. El funcionamiento es el de utilizar la probabilidad de Bayes que defina la probabilidad de pertenecer un dato a una clase determinada, se le puede añadir una matriz de pesos para que den más importancia a unas clases u a otras, y finalmente se obtiene una función de costes. El objetivo del método es minimizar esta función de costes.

Este método es óptimo cuando buscamos el menor error de discriminación, cuando la distribución de los datos es normal y cuando el número de datos es suficientemente grande.

Al igual que los otros métodos, el clasificador bayesiano se comporta mejor cuando los datos no se encuentran normalizados llegándose a obtener los valores de precisión de la Tabla 4. Para esta comprobación se ha utilizado la mejor PCA para cada uno.

	Acierto Entrenamiento (%)	Acierto Test (%)
Normalizado	74.91	74.55
Sin normalizar	88.45	87.75

Tabla 4. Errores para datos normalizados y sin normalizar del clasificador bayesiano.

En cuanto al método para reducir la dimensionalidad, los resultados obtenidos para el auto-encoder ofrecen un mayor valor de precisión con el clasificador bayesiano, como se muestra en la

	Acierto Entrenamiento (%)	Acierto Test (%)
PCA	88.45	87.75
Auto-encoder	89.10	88.45

Tabla 5. Errores para distintas técnicas de reducción de dimensionalidad.

La matriz de confusión obtenida al aplicar la reducción dimensional con un auto-encoder de dimensión de salida 160 se muestra en la Figura 6.

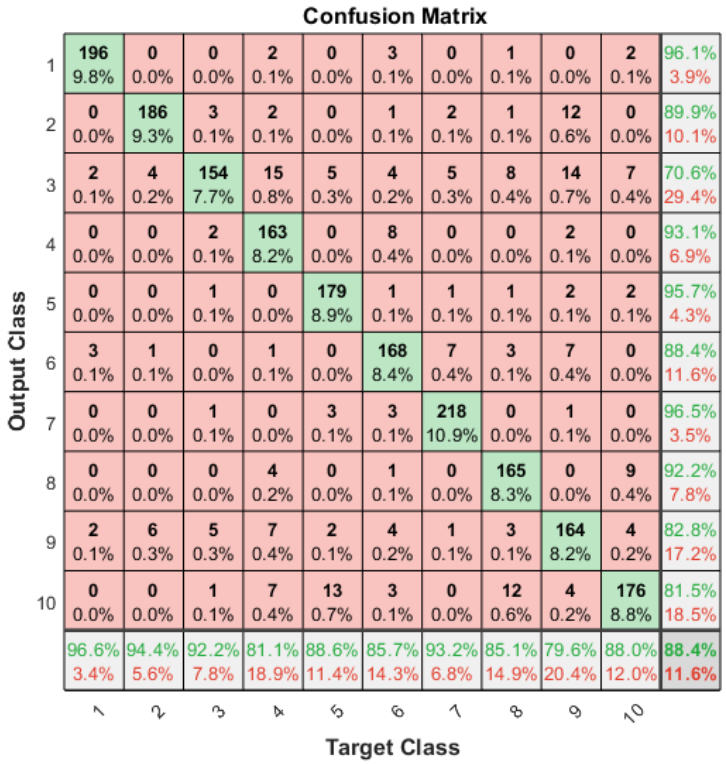

Figura 6. Matriz de confusión obtenida para la clasificación de dígitos con el clasificador bayesiano.

5 PERCEPTRÓN MULTICAPA (MLP)

En este apartado se explica brevemente en qué consiste el perceptrón multicapa (*Multilayer Perceptron* – MLP –) y cómo se ha utilizado para realizar la clasificación de los dígitos escritos a mano.

El perceptrón multicapa es una red neuronal artificial (*Artificial Neural Network* – ANN –) que está formada por múltiples capas, de manera que tiene la capacidad de resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón simple.

En un perceptrón multicapa, se distinguen tres tipos de capas: capa de entrada, capas ocultas y capa de salida. Cada una de estas capas puede estar formada por una o varias neuronas. La capa de entrada está constituida por las neuronas que introducen los patrones de entrada a la red neuronal y en las cuales no se produce procesamiento. Por otro lado, las capas ocultas son aquellas que están compuestas por neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores. Finalmente, la capa de salida contiene las neuronas cuyos valores de salida se corresponden con las salidas de toda la red. En el caso de este trabajo, las salidas de la red se corresponden con la clasificación de los dígitos entre las diez clases que hay. En la Figura 7 se muestra la estructura general de una red neuronal MLP.

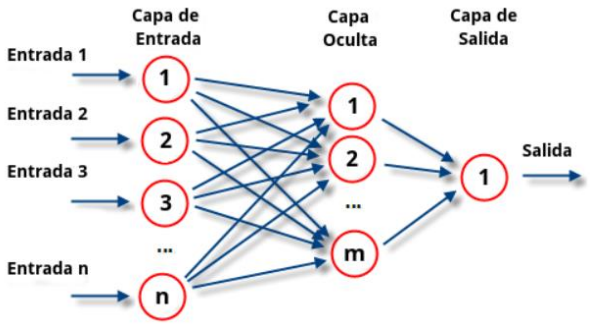

Figura 7. Estructura general de una red neuronal MLP.

El entrenamiento de este tipo de red neuronal es supervisado, es decir, se conoce la clase a la que pertenece cada uno de los datos de entrenamiento. El algoritmo utilizado para su entrenamiento se conoce con el nombre de *back-propagation* (propagación hacia atrás). Este algoritmo consiste en inicializar los pesos de la red de forma aleatoria y después se van introduciendo los diferentes datos de entrada varias veces (varias épocas) para que se vayan actualizando los valores de los pesos de la neurona de manera que las salidas de la red coincidan lo máximo posible con las salidas deseadas, es decir, se intenta minimizar el error cuadrático medio entre el valor real de salida de la real y el deseado para cada muestra de entrenamiento.

El algoritmo de *back-propagation* puede terminar de tres maneras posibles: llegando a un mínimo absoluto (cuando el error cuadrático medio disminuye por debajo de un determinado umbral); llegando a un mínimo local (cuando la derivada del error está por debajo de un umbral específico); o cuando se alcanza el número de épocas establecido.

En este trabajo, para entrenar la red neuronal MLP se han modificado tanto los valores iniciales de la red, como pueden ser el algoritmo empleado para realizar los cálculos o el número de épocas, así como el número de muestras de entrenamiento y el número de capas y neuronas totales de la red.

Para establecer el número de muestras empleadas en el entrenamiento, se han realizado pruebas con 6000, 8000 y 9000 muestras de las 10000 totales de las que se disponen. Obviamente, se ha

demostrado que cuanto mayor era el número de muestras con las que se entrenaba la red, el error de test disminuía. Sin embargo, entre las pruebas realizadas con 8000 y 9000 muestras no había prácticamente mejora de una a otra. Por lo tanto, se ha decidido utilizar 8000 muestras en vez de 9000 para tener suficientes muestras para el testear la red posteriormente, ya que 1000 muestras solo para test parece ser un número escaso de muestras.

En cuanto a los valores iniciales de la red para el entrenamiento, se ha optado por escoger valores iniciales aleatorios y realizar varios entrenamientos y testeos con los mismos datos de manera que se obtenga la media del número de errores de clasificación. Esto permite reducir la dependencia de la aleatoriedad de los valores iniciales en los resultados obtenidos. Por otro lado, en cuanto al algoritmo de cálculo, se ha decidido utilizar finalmente el de Levenberg-Marquardt, ya que era el que más precisión de clasificación proporcionaba, aunque sí que es cierto que también era el más lento de todos. En cuanto al número de épocas elegido, éste ha sido igual a 100 épocas, aunque este valor inicial ha afectado en realidad a la pruebas realizadas por que siempre el algoritmo convergía antes de llegar a las 30 épocas.

Para clasificar los dígitos mediante una red neuronal MLP, se ha decidido que la capa de salida disponga de 10 neuronas, de manera que cada una de estas neuronas se corresponde con una de las 10 clases. Como consecuencia, la neurona que se corresponde con la clase más probable del dato de test a clasificar tendrá como salida un valor igual a 1, mientras que las restantes 9 neuronas proporcionarán una salida con valor 0. Es decir, solo se activa aquella neurona correspondiente a la clase en la que se clasificará la muestra de entrada a la red.

Finalmente, se necesita elegir el número de capas ocultas de la red y él número de neuronas de cada una de ellas. Para ello, se han ido probando con diversos valores para cada uno de estos dos parámetros, tal y como se muestra en la Tabla 6. Cabe destacar que para estas pruebas se ha utilizado siempre el mismo número de muestras de entrenamiento y de test (8000 y 2000 respectivamente) y el algoritmo de Levenberg-Marquardt.

N.º Capas ocultas	N.º Neuronas por capa oculta	N.º Errores de clasificación (test)	Porcentaje de acierto
1	10	257	87.2 %
1	30	180	91.0 %
1	50	126	93.7 %
1	100	122	93.9 %
2	50 - 50	115	94.3 %
2	100 - 50	95	95.3 %

Tabla 6. Comparativa entre errores de clasificación y número de capas y neuronas de la red MLP.

Como se puede observar en la tabla anterior, utilizar dos capas ocultas en vez de una con el mismo número de neuronas en total ha supuesto una cierta mejora en cuanto al porcentaje de acierto de clasificación. Además, también se ha podido comprobar que al ir aumentando el número total de neuronas en las capas ocultas, mejora el rendimiento de la red, disminuyendo el error de test. Sin embargo, esto ocurre hasta cierto punto, ya que si se sigue aumentando el número total de neuronas llegaría un momento en el cual el error de test empezaría a aumentar por *overfitting*. Esto se debe a que, al utilizar muchos parámetros, la red aprendería a clasificar muy bien las muestras de entrenamiento y perdería capacidad de generalización, por lo que no sería muy buena clasificando datos distintos a los de entrenamiento. Por este motivo, se ha decidido no seguir aumentando más el número de neuronas de la red, ya que con dos capas de 100 y 50 neuronas, respectivamente, se ha

conseguido un buen porcentaje de acierto (95.3%). Además, cuando se intentaba entrenar con más neuronas, Matlab varias veces se bloqueaba y si no, el entrenamiento tardaba más de 10 horas.

En la Figura 8, se muestra la estructura final de la red neuronal MLP que se ha desarrollado donde también se puede observar los algoritmos con los que se ha entrenado. Para el entrenamiento, se recuerda que se han utilizado 8000 muestras a las que previamente se les ha realizado una reducción de dimensionalidad con una PCA. La dimensión de los datos de entrada de la red es 48, en vez de 784 que es el tamaño original de los dígitos.

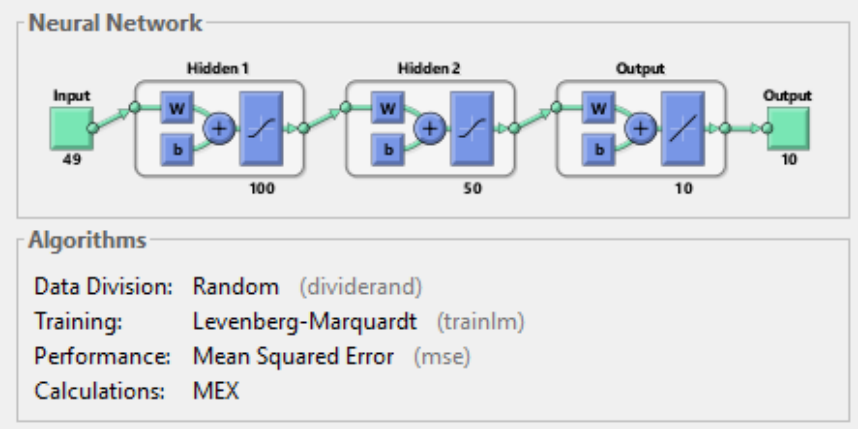

Figura 8. Entrenamiento de la red neuronal MLP definitiva.

Por último, en la Figura 9 se presenta la matriz de confusión obtenida al clasificar las 2000 muestras que se reservaron para testear la red neuronal. Como se puede observar en dicha matriz, se ha obtenido un porcentaje de acierto del 95.3 %, con tan solo 95 errores de clasificación.

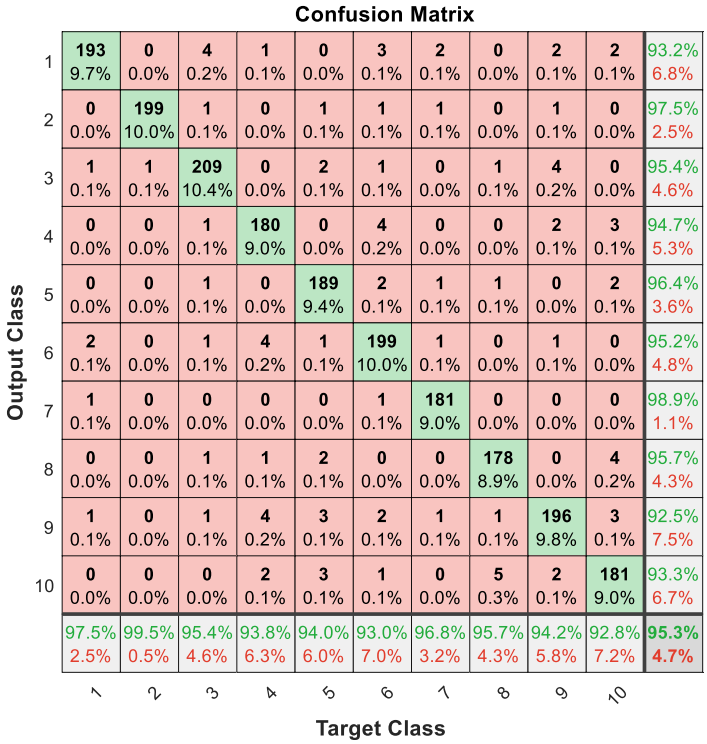

Figura 9. Matriz de confusión obtenida para la clasificación de dígitos con la red neuronal MLP.

6 MAPA AUTOORGANIZADO (SOM)

Una de las opciones que se plantea para la clasificación de los números es el uso de una red neuronal de tipo SOM (*Self organizing map*), un tipo de red no supervisada que genera un mapa de neuronas cuya activación determinará la clase a la que pertenece la imagen de entrada.

El objetivo de la red será obtener una serie de grupos o clusters y asignarles el valor de la clase a la que pertenecen a partir de las etiquetas conocidas previamente. Se tendrá que asegurar que las neuronas que se activan con cada número se encuentran próximas entre sí para asegurar el principio de vecindad de las redes SOM (muestras cercanas en el espacio de entrada activan neuronas cercanas en el espacio de salida).

En primer lugar se analizará si utilizar la imagen completa o la reducida dimensionalmente con la PCA. Para ello, se entrenarán varios modelos con los mismos datos y se comprobará la precisión de cada uno de ellos. Además, también se tendrá en cuenta la velocidad de entrenamiento, que será mucho más reducida para los datos provenientes de la PCA.

Para decidir el tipo de entrada a la red, se han utilizado dos mapas de 20x20 neuronas a la salida durante 250 épocas, obteniéndose los datos de precisión mostrados en la Tabla 7.

	Acierto Entrenamiento (%)	Acierto Test (%)
Imagen Completa	85.55	83.60
PCA	87.75	87.40

Tabla 7. Errores del SOM 20x20 con la imagen original y con PCA.

Observando los resultados obtenidos, se partirá de las imágenes provenientes de la PCA con el fin de reducir el tiempo de entrenamiento y de mejorar la precisión. A continuación se deberá decidir la dimensión de la red SOM, que tendrá una relevancia muy importante en la calidad del clasificador. Una dimensionalidad muy reducida daría lugar a una clasificación muy pobre ya que algunas neuronas se activarían con varios números muchas veces, mientras que una dimensionalidad elevada presentaría el problema de tener neuronas que nunca se activen además de tener un tiempo de entrenamiento elevado.

El último parámetro que se deberá determinar es el número de épocas para entrenar a la red, ya que en las redes SOM, un número demasiado elevado no dará lugar a cambios en los pesos.

Procedimiento para el entrenamiento de la red

1. Reducción de la dimensionalidad de los datos a partir de la PCA y normalización de los mismos.
2. Definición de los parámetros de la red (épocas y dimensión)
3. Entrenamiento del SOM con los 8000 datos de entrenamiento
4. Obtención del vector de neuronas que se activan para los datos de entrenamiento *classes_train*.
5. Obtención de la matriz que determina para cada neurona, cuántas veces se ha activado con cada etiqueta (números del 0 al 9 del *dataset*). A esta matriz se le llamará *SOM_Classes* y tendrá una dimensionalidad de $m_x n$, donde "m" es el número de neuronas del SOM y "n" es el número de etiquetas del *dataset*.
6. Cálculo de la matriz que determina la etiqueta del *dataset* correspondiente para cada neurona del SOM. A esta matriz se le denominará *SOM_Matrix* y tendrá una dimensión de $a_x b$ (donde "a" es el número de neuronas por columna y "b" es el número de neuronas por fila del mapa).

Para la obtención de esta matriz se han tenido en cuenta dos casos singulares. El primero de ellos es en el que una de las neuronas no se active con ninguna entrada, en cuyo caso se le asignará el valor de la neurona más cercana para cumplir la condición de vecindad. El segundo caso es que una neurona se active el mismo número de veces con dos entradas distintas, para lo cual se escogerá si existiera un valor de alguna neurona vecina.

7. Una vez obtenida *SOM_Matrix*, se comprobarán las neuronas que se activan con el *dataset* de test y se les asignará la etiqueta correspondiente a la neurona que se active.

Es destacable que los dos casos mencionados en el apartado 6 se pueden solucionar de forma visual observando la imagen que viene dada por los pesos de la neurona afectada.

Para estudiar la dimensionalidad óptima de la red neuronal, se han realizado varias pruebas con distinta dimensionalidad, obteniéndose para cada una de ellas el porcentaje de acierto con los datos de entrenamiento y con los datos de test. Además, se ha estudiado la influencia de las épocas de entrenamiento para varias de ellas para buscar el punto óptimo al tratarse de un sistema de aprendizaje no supervisado.

Por simplicidad del post-procesado de los resultados obtenidos, se ha trabajado con mapas SOM 2D con las mismas filas que columnas. Además, los datos de entrenamiento se han incorporado a la red de forma aleatoria y no de forma consecutiva por clases para mejorar la calidad de la red.

En la Tabla 8 se muestran varias pruebas que han sido realizadas durante el entrenamiento del SOM:

Dimensión	Épocas	Acierto Entrenamiento (%)	Acierto Test (%)
6x6	200	77.50	78.05
10x10	200	86.71	85.65
15x15	100	84.29	84.10
15x15	200	89.94	88.85
15x15	300	90.41	89.30
15x15	400	90.30	89.05
20x20	100	86.68	85.70
20x20	200	92.18	89.95
20x20	300	92.50	91.30
25x25	300	93.70	91.95
25x25	400	93.51	92.65
30x30	300	94.45	92.65
35x35	400	95.73	93.85

Tabla 8. Errores de clasificación de la red SOM para distinta dimensión y épocas.

Observando los datos, se puede comprobar que el número de épocas óptimo para el entrenamiento de la red ronda las 300, obteniéndose resultados similares con entrenamientos con más épocas tanto para entrenamiento como para test y resultados peores para un menor número de épocas.

A partir de cierta dimensionalidad, el error de test apenas mejora y comienzan a aparecer neuronas que no se activan con los datos de entrenamiento, lo cual puede causar problemas posteriormente a la hora de asignarles etiquetas. Por ello, se ha decidido utilizar la red de 25x25, que aporta unos buenos resultados para los datos de test sin presentar muchas neuronas que no se activan.

En la Figura 10 se muestra un mapa de la red SOM en la cual se dibujan las imágenes definidas por los pesos de cada una de las neuronas. Como se puede observar, debido a la PCA realizada, las imágenes mostradas presentan cierto ruido en los bordes, aunque los resultados que se obtienen son buenos.

Además una mejor visualización, se ha realizado un mapa con colores donde se pueden visualizar los agrupamientos de las distintas clases, con algunas neuronas aisladas de las demás.

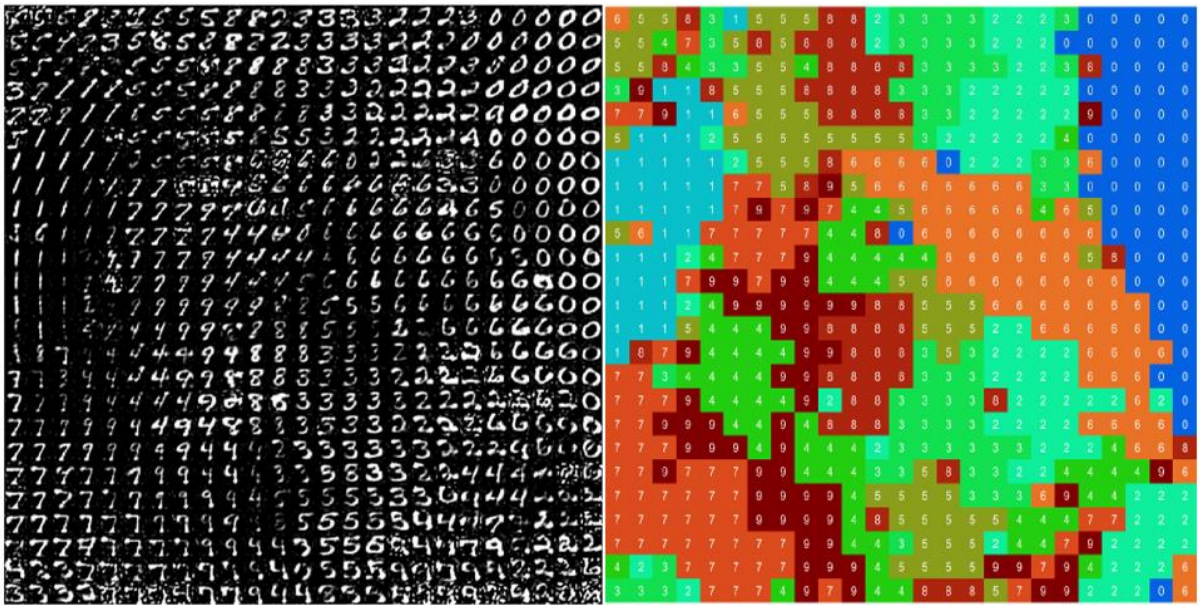

Figura 10.Mapa de los pesos de las neuronas (izquierda) y de las clases de las neuronas (derecha) para el SOM25x25.

La matriz de confusión obtenida con el clasificador SOM se puede observar en la Figura 11.

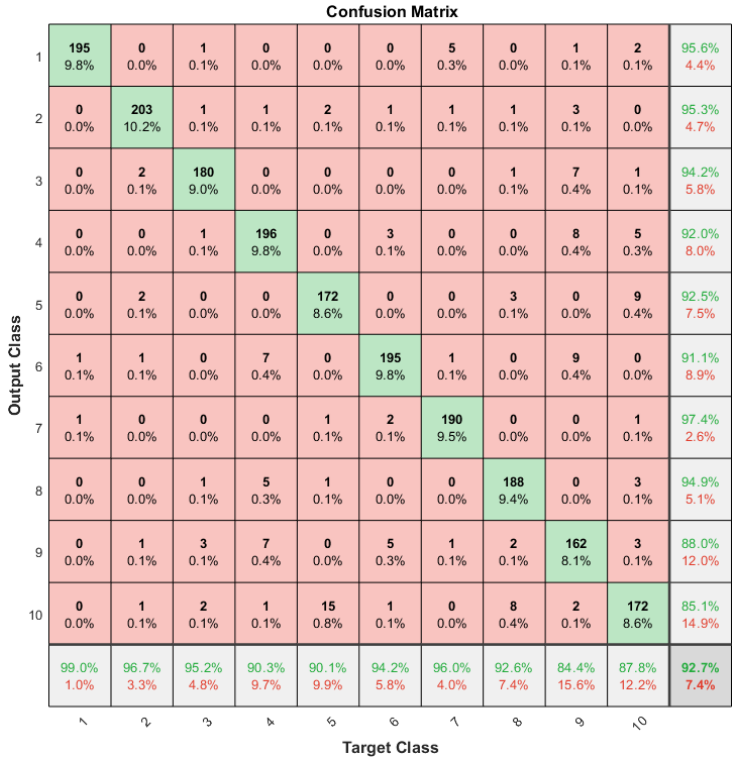

Figura 11. Matriz de confusión obtenida para la clasificación de dígitos con la red neuronal SOM.

7 ALTERNATIVAS ESTUDIADAS

Como se ha comentado anteriormente, en este proyecto se propone una alternativa mucho más rápida de entrenar que los demás métodos que contienen neuronas. Este método hace uso de dos capas de auto-encoders y una capa *softmax* para la clasificación.

El proceso ha sido el siguiente:

1. Se entrena un auto-encoder para reducir la dimensionalidad de los datos. Se decide obtener 100 características en esta primera etapa.
2. Se entrena un segundo auto-encoder para reducir aún más la dimensionalidad desde la dimensionalidad de 100 reducida anteriormente hasta conseguir comprimir la información a 50 características.
3. Después se unen los dos encoder entrenados y se pasan las características por una capa *softmax* para clasificar los datos. Tras este paso se consigue una precisión del 84% con la estructura mostrada en la Figura 12.

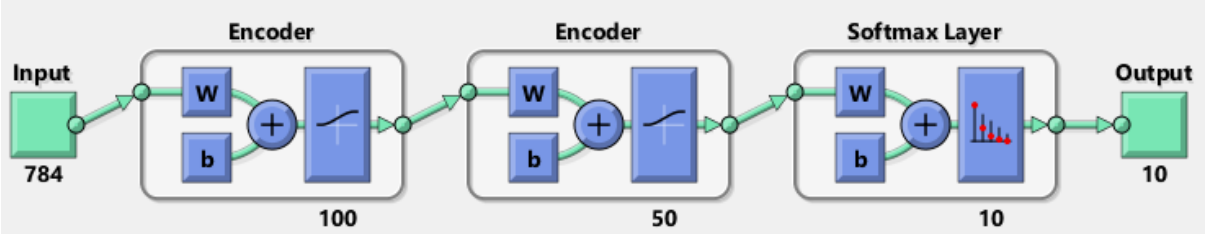

Figura 12. Estructura de ambos auto-encoders y unca capa de softmax.

4. Por último, al ser estos módulos formados por redes neuronales, se puede aplicar el proceso de *back-propagation* de forma que se ajusten mejor los pesos de las neuronas. Tras este proceso se consigue una precisión del 96% con la matriz de confusión de la Figura 13.

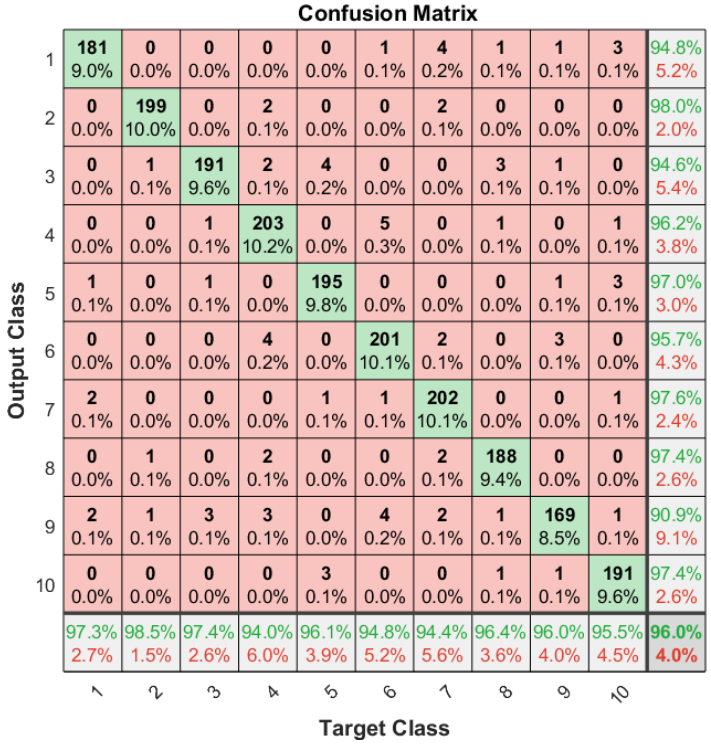

Figura 13. Matriz de confusión obtenida para la clasificación de dígitos con la alternativa propuesta.

8 COMPARACIÓN DE LAS TÉCNICAS UTILIZADAS

Para realizar una comparativa entre los distintos modelos que se han entrenado para la clasificación de dígitos, se ha generado la Tabla 9, en la cual se representan los porcentajes de acierto, tiempos de clasificación y tiempos de entrenamiento.

Clasificador	Porcentaje de acierto	Tiempo de clasificación (ms)	Tiempo de entrenamiento
k-NN	95.55	149.692	14.3 ms
Bayesiano	88.45	38.018	31.9 ms
MLP	95.30	13.233	5h 28 min 40 s
SOM	92.65	69.719	4 min 59 s
Autoencoder+Softmax	96.00	37.470	4 min 47 s

Tabla 9. Tabla comparativa de los distintos clasificadores

El clasificador k-nn presenta la ventaja de ofrecer unos resultados muy buenos para la clasificación de los dígitos (95.55%) a partir de un modelo muy sencillo de entrenar. Sin embargo, es el que tiene un mayor tiempo de ejecución debido al cálculo de las distancias que debe realizar.

El mapa SOM ofrece unos resultados de clasificación muy buenos teniendo en cuenta que se trata de una red no supervisada. El tiempo de ejecución del modelo no es ni rápido ni lento, mientras que el tiempo de entrenamiento es alto para redes con dimensionalidad elevada. Por otro lado, una de las grandes desventajas que presenta el SOM es la necesidad de llevar a cabo una asignación de las etiquetas a posteriori, lo cual dificulta el entrenamiento del SOM.

En cuanto a las ventajas que ofrece la red neuronal MLP frente a los demás clasificadores, según se expone en la Tabla 9, se llega a la conclusión de que es de las técnicas de Machine Learning con más precisión para la clasificación de dígitos escritos a mano (95.3%), junto con el clasificador k-nn. Además, el perceptrón multicapa se caracteriza por ser el más rápido en realizar la clasificación con respecto a los demás clasificador (13.23 ms), utilizando el mismo número de muestras de test. El único inconveniente que tiene la red neuronal MLP es que el coste computacional es muy alto durante el entrenamiento, sobre todo si se emplea el algoritmo de Levenberg-Marquardt, por lo que el tiempo en que se tarda en entrenar la red es bastante alto (proporcional al número de neuronas utilizadas). Para el caso de una red MLP con 2 capas y 150 neuronas, se ha tardado aproximadamente 5 horas y 30 minutos para conseguir el modelo. Sin embargo, cabe destacar que el tiempo de entrenamiento depende de las características del procesador en el que se realice.

Teniendo en cuenta los resultados de la Tabla 9 se puede determinar que la alternativa propuesta ha cumplido con los objetivos de precisión requeridos. El propósito de la alternativa era reducir considerablemente el tiempo de entrenamiento manteniendo un importante porcentaje de precisión. Finalmente, además de obtener un tiempo para su entrenamiento de 5 minutos frente al tiempo de aprendizaje de los algoritmos SOM o MLP, ha superado el valor de precisión de test que ofrecen estos algoritmos.

9 CONCLUSIONES

En cuanto a la etapa de pre-procesamiento de los datos de entrada, se ha comprobado que aplicar una normalización mejora los resultados de reconstrucción de la imagen pero empeora los resultados de clasificación en el caso del k-nn, el bayesiano y la red SOM. Por otro lado, la mejor dimensión de los datos de entrada obtenida con el PCA se encuentra entre los 30 y 50 en función del clasificador que se implemente. Además, analizando la combinación de un auto-encoder con una PCA para reducir la dimensionalidad, se ha comprobado que los resultados obtenidos son peores que con una PCA simplemente. Es importante mencionar que no siempre la reducción de dimensionalidad con PCA aporta mejores resultados, ya que el clasificador bayesiano presenta mejor comportamiento con la reducción del auto-encoder.

Al estudiar el rendimiento de los clasificadores clásicos, se ha llegado a la conclusión de que un buen valor de vecinos para el modelo k-nn se encuentra entre 1 y 5, obteniéndose valores de precisión muy similares entre sí. Por otro lado, los resultados obtenidos para la clasificación de los dígitos muestran que el clasificador bayesiano es el que peor funciona. Esto puede deberse a que no haya una gran cantidad de datos y estos no sigan una distribución normal.

Atendiendo al clasificador MLP, se ha podido comprobar que utilizar dos capas ocultas en vez de una mejora considerablemente los resultados de clasificación, aunque aumenta el tiempo de entrenamiento de la red. En cuanto al número de neuronas del MLP, se ha observado que, al ir aumentando el número de neuronas, el error de test disminuía hasta cierto punto, a partir del cual empezaba a aumentar. Un buen número de neuronas que se ha obtenido es 150, divididas en dos capas (100 + 50). Finalmente se concluye que el algoritmo de Levenberg-Marquardt es el más preciso a cambio de ser más costoso computacionalmente que los demás. El número de épocas para el caso del MLP ha sido de 100, pero no es muy relevante ya que el algoritmo de entrenamiento siempre acababa convergiendo al alcanzar el error mínimo establecido antes de las 30 épocas.

Por otra parte, el clasificador SOM se ha estudiado directamente con una dimensionalidad 2D debido a que se pierde menos información que con un mapa 1D al hacer la reducción de dimensionalidad. La precisión de clasificación está estrechamente relacionada con el número de neuronas, obteniéndose mejores resultados con mapas más grandes (hasta 35x35). Sin embargo, la existencia de neuronas que nunca se activan durante el entrenamiento puede dar lugar a fallos en la clasificación de nuevas muestras si el mapa es demasiado grande. Por lo tanto, se ha decidido que un buen número de neuronas para el SOM es de 25x25. A la hora de escoger el número de épocas, se ha llevado a cabo un proceso iterativo en el cual se han analizado la precisión de varios modelos entrenados con distinto número de épocas, llegando a la conclusión de que 300-400 épocas dan unos buenos resultados.

En este proyecto se ha mostrado que el clasificador que hace uso de los auto-encoders y la capa softmax logra reducir considerablemente el tiempo de aprendizaje a la vez que se consigue clasificar con mayor precisión que los demás métodos utilizados.

Para organizar los datos de entrada, se ha decidido dividirlos en 8000 datos de entrenamiento y 2000 datos de test con el fin de lograr un buen entrenamiento de los distintos modelos y tener un número considerable de datos de test para analizar los resultados. Adicionalmente se decidió dividir los datos de entrenamiento y de test de forma aleatoria para distintos entrenamientos con el fin de asegurar que los modelos entrenados presentaban un funcionamiento correcto. Finalmente se comprobó el comportamiento de las redes MLP y SOM al introducir los datos ordenados secuencialmente por clases, obteniéndose unos resultados peores en este caso.

10 REPARTO DE ROLES

A continuación, se muestra una tabla con el reparto de roles que se ha realizado para la consecución de este trabajo. Aunque el trabajo se haya dividido en tres roles, todos los integrantes del grupo han participado tanto en la programación de los algoritmos, como en la investigación científica y en la elaboración de la memoria. La tabla simplemente muestra quiénes han sido los principales responsables de cada una de estas tres áreas para conseguir con los objetivos previstos. Por último, cabe destacar que el tiempo y esfuerzo que le ha dedicado cada miembro del grupo a la elaboración de este trabajo han sido totalmente equitativos.

Responsable	Rol	Nota
Álvaro Benito Oliva	Técnico	100%
Germán Andrés Di Fonzo Caturegli	Coordinador	100%
Juan José Jurado Camino	Científico	100%

Tabla 10. Reparto de roles.