

**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 18 de junio de 2021

TABLA DE CONTENIDOS

[1 INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO 3](#_Toc75085848)

[2 PROCESAMIENTO DE DATOS 4](#_Toc75085849)

[2.1 ANÁLISIS DE COMPONENTES PRINCIPALES (PCA) 4](#_Toc75085850)

[2.2 DISCRIMINANTE DE FISHER 4](#_Toc75085851)

[2.3 REDUCCIÓN DE DIMENSIONALIDAD MEDIANTE PCA Y AUTO-ENCODER 4](#_Toc75085852)

[3 CLASIFICADOR K-NN 5](#_Toc75085853)

[4 CLASIFICADOR BAYESIANO 6](#_Toc75085854)

[5 PERCEPTRÓN MULTICAPA (MLP) 7](#_Toc75085855)

[6 MAPA AUTOORGANIZADO (SOM) 8](#_Toc75085856)

[7 MEJORA DE LAS TÉCNICAS UTILIZADAS 9](#_Toc75085857)

[8 COMPARACIÓN DE LAS TÉCNICAS UTILIZADAS 10](#_Toc75085858)

[9 CONCLUSIONES FINALES 11](#_Toc75085859)

[10 REPARTO DE ROLES 11](#_Toc75085860)

[ANEXO. CÓDIGO IMPLEMENTADO EN MATLAB 12](#_Toc75085861)

[I. PROCESAMIENTO DE CARACTERÍSTICAS 12](#_Toc75085862)

[II. CLASIFICADOR K-NN 12](#_Toc75085863)

[III. CLASIFICADOR BAYESIANO 12](#_Toc75085864)

[IV. PERCEPTRÓN MULTICAPA (MLP) 12](#_Toc75085865)

[V. MAPA AUTOORGANIZADO (SOM) 12](#_Toc75085866)

# 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.

# PREPROCESAMIENTO DE DATOS

Explicar que se ha decidido entrenar con 8000 datos todos los clasificadores y testear con las 2000 restantes.

Explicar para qué se utiliza la reducción de dimensionalidad de los datos de entrada. Comentar por encima que se va a utilizar primero PCA, después Fisher y después una combinación entre PCA y auto-encoder. Después, en cada aparcado, comentar en qué consiste cada uno de los métodos de reducción por encima y explicar cómo se ha aplicado en este trabajo.

## ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

## DISCRIMINANTE DE FISHER

## REDUCCIÓN DE DIMENSIONALIDAD MEDIANTE PCA Y AUTO-ENCODER

Este apartado yo creo que mejor poner al final en el apartado de MEJORAS DE LAS TÉCNICAS UTILIZADAS.

# 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:

|  |  |  |
| --- | --- | --- |
|  | Train | Test |
| Normalizado | 96.03 | 91.50 |
| Sin normalizar | 97.53 | 94.15 |

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, el discriminante de Fisher y el autoencoder, obteniéndose los resultados de la TABLA.

|  |  |  |
| --- | --- | --- |
|  | Train | Test |
| PCA | 97.53 | 94.15 |
| Fisher |  |  |
| Autoencoder | 96.39 | 92.55 |

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.

**Decir cual se ha escogido**

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

|  |  |  |
| --- | --- | --- |
| k vecinos | Train | 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 |

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

Tabla

Descripción generada automáticamente

# CLASIFICADOR BAYESIANO

Explicar en qué se basa el clasificador bayesiano (muy breve). Comentar cómo se ha utilizado para este trabajo y qué cambios se han ido haciendo para ir mejorando el algoritmo.

# 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 \_\_ se muestra la estructura general de una red neuronal MLP.

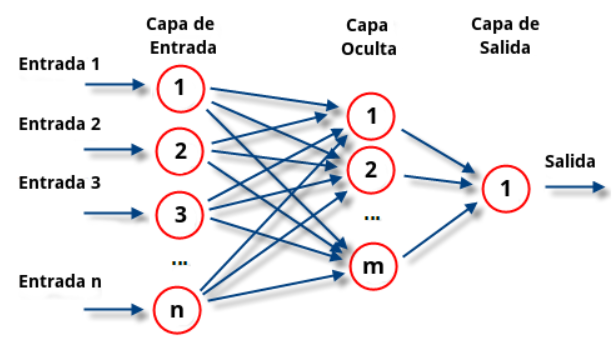


Figura 1. Estructura general de una red neuronal MLP.

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 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 \_\_. 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 1. 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 oculta, 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 \_\_, 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.

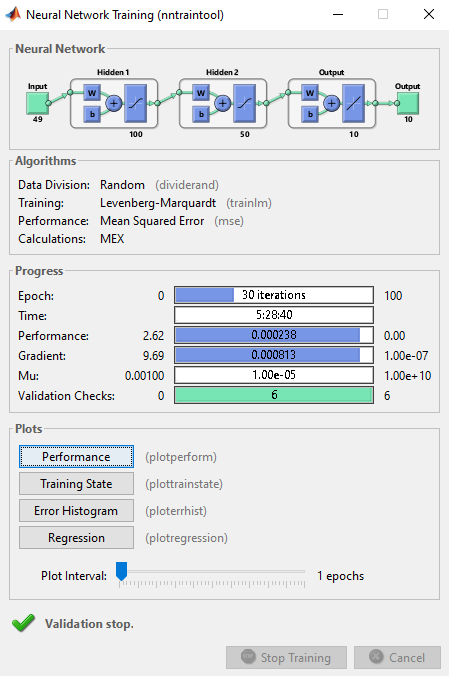


Figura 2. Entrenamiento de la red neuronal MLP definitiva.

Por último, en la Figura \_ 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 3. Matriz de confusión obtenida para la clasificación de dígitos con la red neuronal MLP.

# 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**.

|  |  |  |
| --- | --- | --- |
|  | Train | Test |
| Imagen Completa | 85.55 | 83.60 |
| PCA | 87.75 | 87.40 |

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 mxn, 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 axb (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.

A continuación se mostrarán una serie de pruebas que han sido realizadas durante el entrenamiento del SOM:

|  |  |  |  |
| --- | --- | --- | --- |
| Dimensión | Épocas | Acierto train | 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 | 93.41 | 92.70 |
| 30x30 | 300 | 94.45 | 92.65 |
| 35x35 | 400 | 95.73 | 93.85 |

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** 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.

Imagen que contiene Código QR

Descripción generada automáticamenteImagen que contiene Gráfico

Descripción generada automáticamente

La matriz de confusión obtenida con el clasificador SOM se puede observar en la **FIGURA**

Tabla

Descripción generada automáticamente

Cabe destacar que las imágenes obtenidas cuando no se aplica la PCA presentan menos ruido a pesar de que la precisión al clasificar es peor, como se muestra en la **FIGURA**.

Patrón de fondo

Descripción generada automáticamente con confianza baja Gráfico de rectángulos

Descripción generada automáticamente con confianza media

# MEJORA DE LAS TÉCNICAS UTILIZADAS

Intentar mejorar los resultados obtenidos hasta ahora implementando primero el K-means para generan 3 subclases por cada clase (en total 30 clases).

Comentar la combinación entre Autoencoder y Fisher para la mejora de resultados.

# COMPARACIÓN DE LAS TÉCNICAS UTILIZADAS

Comentar las ventajas y desventajas de cada uno de los métodos empleados estableciendo comparaciones sobre todo fijándose en las matrices de confusión. Utilizar también tablas. También podría venir bien hacer una tabla de Excel donde se comparen las precisiones y los recall de cada clase.

IMPORTANTE CONTESTAR A ESTAR PREGUNTAS GENERALES:

* Como de buenos son los resultados obtenidos para cada técnica? (comparar matrices de confusión)
* Cuáles son las ventajas relevantes de cada algoritmo con respecto de los otros?
* Cómo de rápido son nuestros algoritmos propuestos?

¡REALIZAR UNA COMPARATIVA DEL TIEMPO COMPUTACIONAL EMPLEADO PARA CLASIFICAR!

# CONCLUSIONES FINALES

IMPORTANTE CONTESTAR ESTAS PREGUNTAS

PRE-PROCESAMIENTO:

* La normalización mejora los resultados en este caso? Sí
* Cual es un buen numero para la dimensión de los datos de entrada obtenido con la PCA? Entre 48-50
* Es bueno el discriminante de Fisher en este caso?
* Es bueno el autoencador + pca en este caso?

CLASIFICADORES CLASICOS:

* Cual es un buen valor para k en el k-nn?
* Se pueden obtener mejores resultados utilizando mas de una función normal de densidad para cada clase?

NEURAL CLASSIFIERS:

* Cual es un buen numero de capas ocultas en el MLP? 2 capas ocultas mejor que 1
* Cual es un buen numero de neuronas para el MLP? 150 neuronas
* Cual es la importancia del algoritmo de entrenamiento del MLP? Levenberg-Marquardt más exacto (mejor precisión) pero más lento que los demás.
* Cual es un buna dimensionalidad del SOM?
* Cual es un bue numero de neuronas para el SOM?

USO DE DATOS DISPONIBLES:

* Cual es un buen numero de muestras para entrenar y testear? 8000 y 2000 respectivamente
* Cual es un buen número de épocas para MLP y SOM? MLP no necesita más de 35-40 pero se han utilizado 100 por asegurar que el algoritmo termine por alcanzar error mínimo
* Cómo es la importancia del orden de las muestras utilizado en el entrenamiento de MLP y SOM? Muy importante, al utilizar las muestras de entrenamiento ordenadas por clases se obtiene mayor número de errores y no se obtiene ni si quiera mejora en cuanto al tiempo de entrenamiento.

# 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 |  | 100% |
| Germán Andrés Di Fonzo Caturegli |  | 100% |
| Juan José Jurado Camino |  | 100% |

Tabla . Reparto de roles.

# ANEXO. CÓDIGO IMPLEMENTADO EN MATLAB

## PROCESAMIENTO DE CARACTERÍSTICAS

## CLASIFICADOR K-NN

## CLASIFICADOR BAYESIANO

## PERCEPTRÓN MULTICAPA (MLP)

## MAPA AUTOORGANIZADO (SOM)