

# Inteligencia Computacional

## Práctica 1. Redes Neuronales.

---

### Índice

<b>Introducción.</b>	<b>2</b>
<b>Red neuronal simple.</b>	<b>2</b>
Perceptrón	3
<b>Red neuronal multicapa.</b>	<b>6</b>
Una capa oculta	7
Momento	8
Número de neuronas en las capas ocultas	9
Dropout	11
Dos capas ocultas	13
<b>Conclusiones.</b>	<b>14</b>

## Introducción.

El problema que se pretende abordar con esta práctica consiste en entrenar varios modelos predictivos con el objetivo de reconocer los nueve dígitos existentes con la menor tasa de error sobre el conjunto de prueba. Para ello se comenzarán con las topologías más sencillas, las cuales como veremos en este documento, proporcionarán una tasa de error más elevada, y posteriormente aumentaremos la complejidad de su estructura con el fin de reducir dicho error y así obtener un modelo competitivo.

Tanto para el entrenamiento como para la validación de las redes neuronales dispondremos de dos conjuntos de dígitos, uno para cada fase correspondiente de 60.000 y 10.000 imágenes, respectivamente. Mediante estos datos se han entrenado y validado los siguientes tipos de modelos predictivos junto a los que aparecerán los diversos experimentos que se han realizado en particular.

## Red neuronal simple.

La estructura de este modelo consiste en una capa de entrada en la que se reciben los píxeles de cada una de las imágenes y una entrada adicional a 1 para incluir el sesgo en los pesos, el cual se inicializará a 0, y así la propia red será la que aprenda este parámetro. El resto de los pesos han sido inicializados aleatoriamente en el intervalo  $[-0.1, 0.1]$ , de modo que haya tanto pesos negativos, que no estimulen a las neuronas, así como pesos positivos que sean capaces de excitarlas para que envíen la información a la siguiente capa. En este caso, esta será la capa de salida, la cual estará formada por diez neuronas, que se corresponden a los diez dígitos que intentamos reconocer. En ambos casos se trata de capas completamente conectadas, es decir, cada neurona de la capa de salida recibirá todas las salidas de la capa de entrada, que en realidad son las propias entradas puesto que esta primera capa no dispone de pesos.

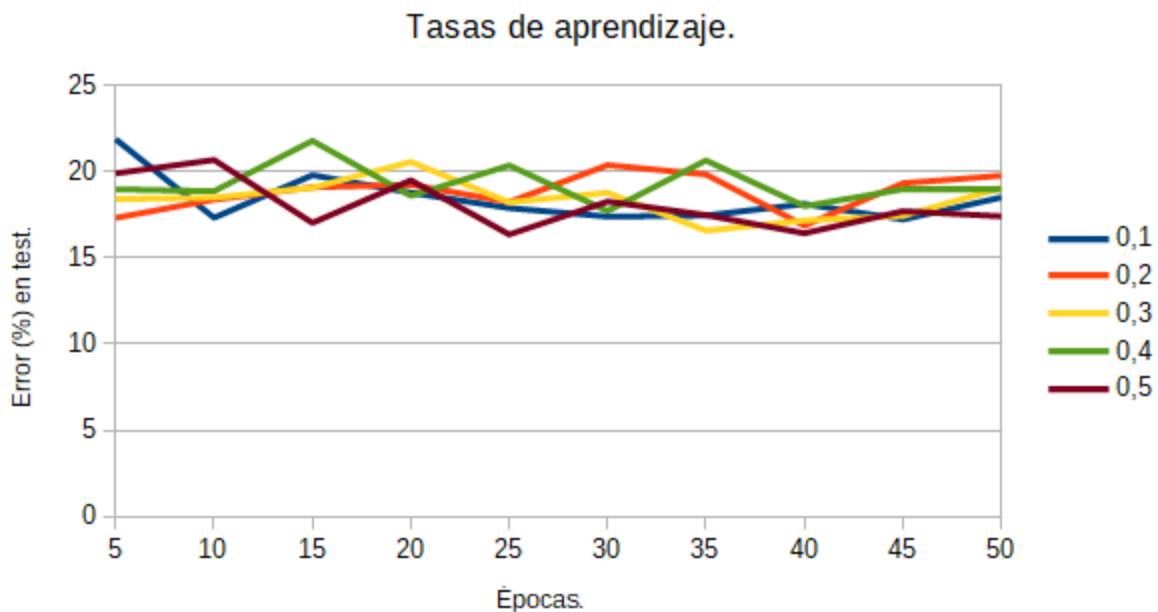
## Perceptrón

Como algoritmo de entrenamiento utilizaremos el **Perceptrón lineal**, con el que se aplicará una función de activación que determinará si una neurona de la capa de salida se ha activado o no. Para ello se realizará una sumatoria del producto de sus pesos por sus entradas y si esta es positiva, entonces la neurona se activa. Si por el contrario es menor que cero la neurona no se activa puesto que no recibe suficiente estimulación.

Para medir la tasa de error obtenida por este modelo aplicaremos el siguiente método:

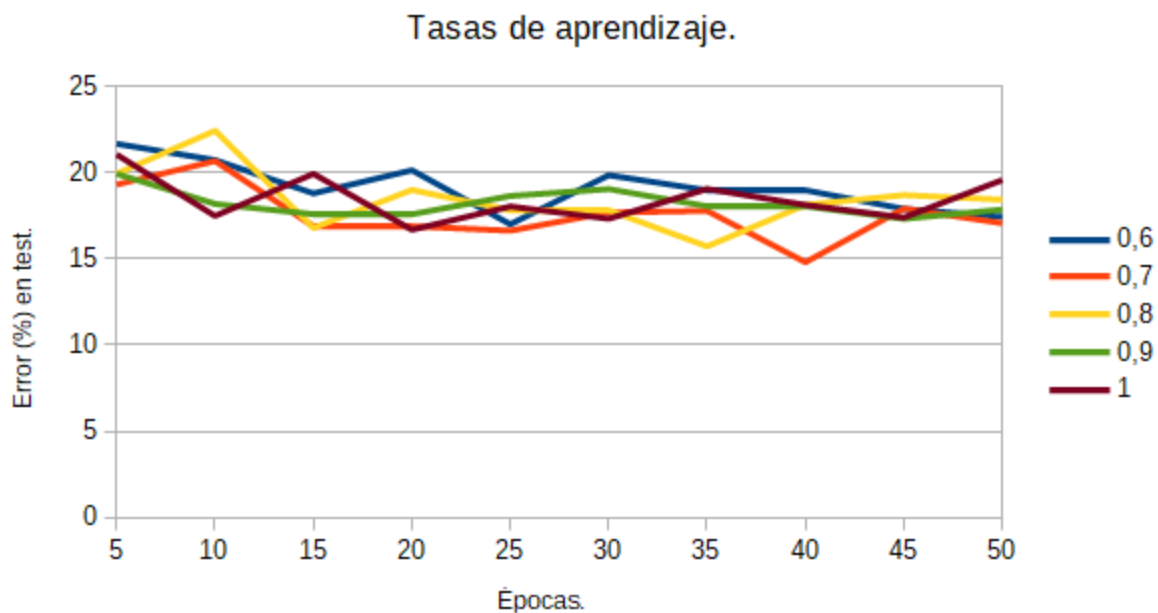
- Si la neurona que representa al dígito actual se activa, entonces no se modifican sus pesos.
- Si una neurona se activa pero no representa al dígito actual entonces se modifican sus pesos restándole las entradas para disminuir su estimulación y que no se active con un dígito que no representa.
- Si la neurona que representa al dígito actual no se activa, entonces se modifican sus pesos sumándole las entradas para proporcionarle una mayor estimulación y que así se active.

Una vez se han evaluado todas las neuronas de la capa de salida para una imagen determinada, se aumenta el error si la neurona que representa al dígito reflejado en la imagen no se ha activado. Siguiendo este proceso he desarrollado dos experimentos. El primero consiste en determinar la **mejor tasa de aprendizaje** para este modelo en concreto. Para ello se ha entrenado un modelo por cada tasa de aprendizaje durante 50 épocas y cada cinco se ha evaluado la red actual con el conjunto de prueba. A continuación se muestran los resultados para las seis primeras tasas de aprendizaje.



Tal y como podemos observar las tasas de errores de todas son bastante similares por lo tanto los seis modelos han adquirido un grado de generalización bastante parecido. No obstante, en esta primera tanda competitiva la tasa de aprendizaje de 0.5 es la que proporciona un menor error en prueba. En la segunda fase de competición se muestran el resto de tasas de aprendizaje desde 0.6 hasta 1.0. De nuevo podemos comprobar su similaridad en relación a la tasa de error

sobre el conjunto de prueba así como su aprendizaje siendo, en este caso, 0.7 la mejor tasa de aprendizaje.

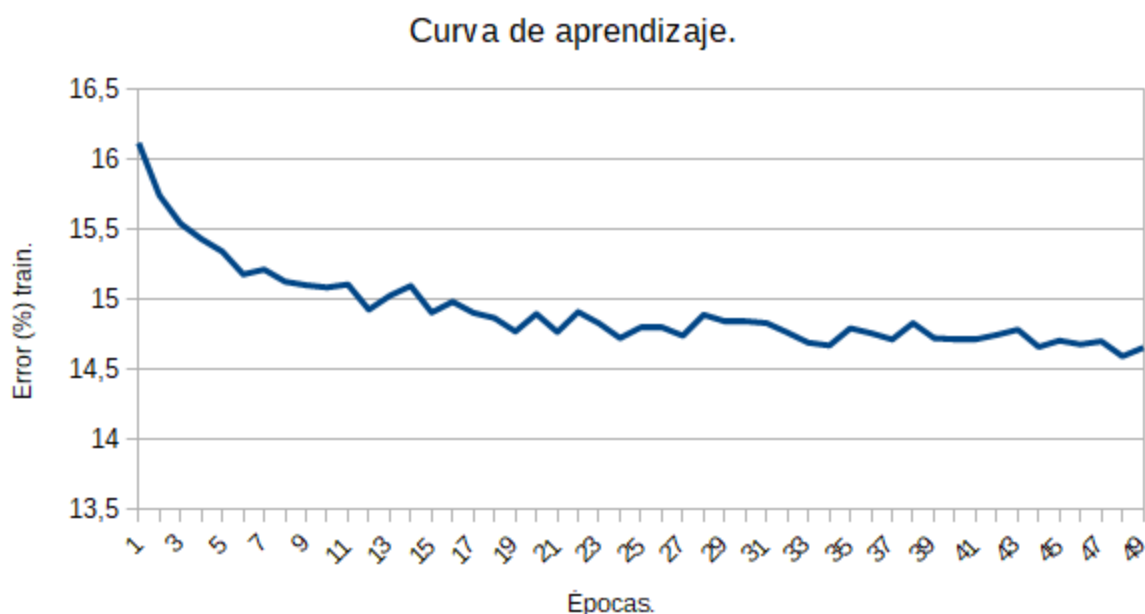


A continuación se muestra una tabla que resume las tasas de error sobre el conjunto de prueba obtenidas por cada modelo que se ha entrenado con una tasa de aprendizaje diferente.

	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
5	21,88	17,32	18,41	19,03	19,9	21,67	19,31	19,94	19,93	21,05
10	17,33	18,38	18,5	18,87	20,68	20,73	20,68	22,43	18,19	17,49
15	19,8	19,09	19,07	21,79	17,03	18,8	16,9	16,8	17,6	19,93
20	18,78	19,25	20,57	18,6	19,5	20,14	16,88	19	17,58	16,69
25	17,89	18,24	18,22	20,36	16,36	17,01	16,64	17,82	18,65	18,04
30	17,4	20,38	18,78	17,72	18,27	19,85	17,69	17,87	19,05	17,33
35	17,46	19,84	16,58	20,65	17,48	19,02	17,79	15,73	18	19,06
40	18,13	16,87	17,17	18	16,43	18,93	14,81	18,14	18,02	18,13
45	17,23	19,33	17,44	19	17,71	17,9	17,92	18,7	17,33	17,4


50	18,51	19,77	19,05	19,02	17,41	17,44	17,07	18,43	17,86	19,57
----	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

De entre todos los resultados podemos comprobar que la menor tasa de error sobre el conjunto de prueba es de **14,81%**, la cual ha sido obtenida en la época 40 y con el modelo cuya tasa de aprendizaje es de 0.7. No obstante, las diferencias entre las tasas mínimas de error obtenidas con cada modelo no son sumamente considerables puesto que este tipo de red neuronal es bastante simple y por tanto, no dispone de diversos parámetros configurables con los que seguir reduciendo el error y mejorando el aprendizaje. Esta afirmación se puede ver reflejada en la siguiente captura donde se muestra la **curva de aprendizaje** para el modelo anterior que ha conseguido la menor tasa de error en prueba. Si observamos dicha curva podemos comprobar que comienza a oscilar a partir de la época 13, lo que significa que el algoritmo no es capaz de seguir aprendiendo y en consecuencia no le es posible seguir reduciendo la tasa de error sobre el conjunto de entrenamiento.



## Red neuronal multicapa.

Este modelo predictivo también dispone de una capa de entrada, con el mismo número de neuronas que en el modelo anterior, es decir, 784 por los píxeles de cada imagen de entrada más una adicional para incluir el sesgo en los pesos. El proceso de inicialización de estos parámetros es el mismo que en el caso anterior. Asimismo, también dispone de una capa de salida con la misma estructura que en el modelo anterior, es decir, se compone de diez neuronas



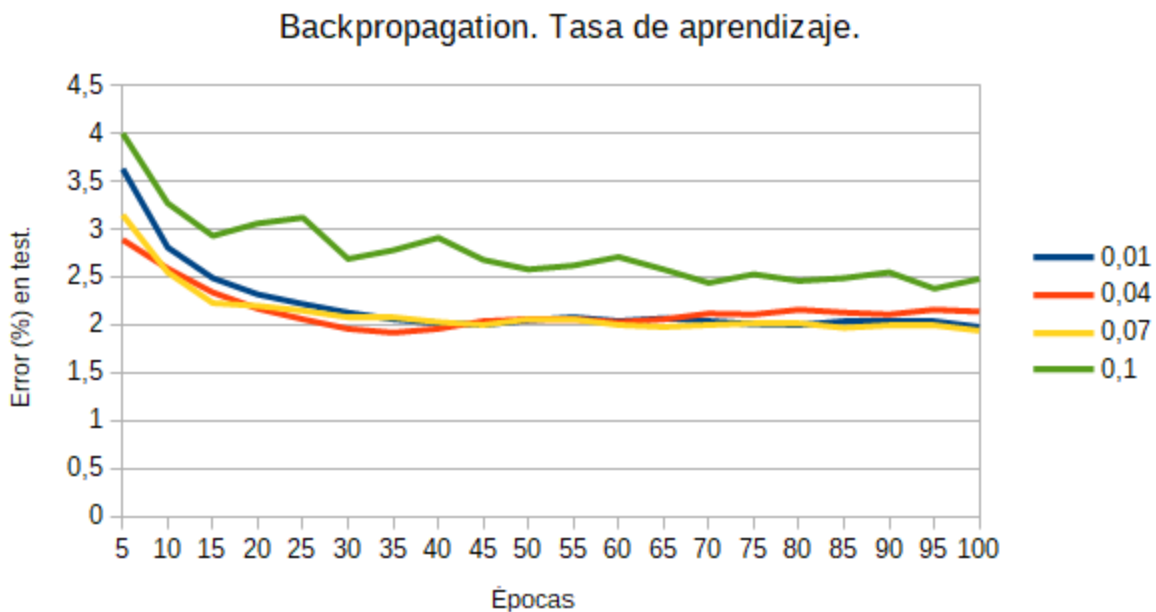
representando, cada una, un dígito diferente. Sin embargo en esta topología se pueden incluir una o varias capas ocultas cuyas neuronas se encargarán de aprender características de las imágenes con el objetivo de mejorar la capacidad de generalización de la red neuronal.

Otra diferencia que caracteriza a este modelo reside en la función de activación, ya que esta no se corresponde con la función lineal que se aplicó en el anterior modelo. En este caso se hará uso de la **función sigmoideal** y con ella la neurona que se activa para cada imagen será aquella que tenga el valor más alto.

Del mismo modo que no se utiliza la función lineal tampoco se podrá aplicar el algoritmo del Perceptrón. La razón principal reside en que, a priori, no conocemos las características que las neuronas de la capa oculta deben aprender. Por ello deberemos guiarnos por el error que generan cada una de las neuronas de la última capa tras calcular sus respectivas salidas utilizando todas las capas del modelo. Para ello se implementará el algoritmo **backpropagation**, a través del cual se propaga el error desde la capa de salida hasta la última capa oculta, puesto que como se comentó anteriormente, la capa de entrada no dispone de pesos. En mi caso he utilizado **aprendizaje online** puesto que considero que es una buena técnica ya que actualiza los pesos de las neuronas tras cada imagen de entrenamiento, lo que provoca muchas actualizaciones de los pesos para fomentar un aprendizaje constante y más rápido..

## Una capa oculta

En esta sección se han realizado tres experimentos iniciales con una red neuronal multicapa con una capa de entrada, **una capa oculta con 256 neuronas** y una capa de salida, todas completamente conectadas. El primer experimento consiste en obtener la tasa de aprendizaje con la que este tipo de modelo presente la menor tasa de errores sobre el conjunto de prueba. Para ello se han evaluado **cuatro tasas de aprendizaje** en el intervalo [0.01, 0.1] durante **100 épocas**.

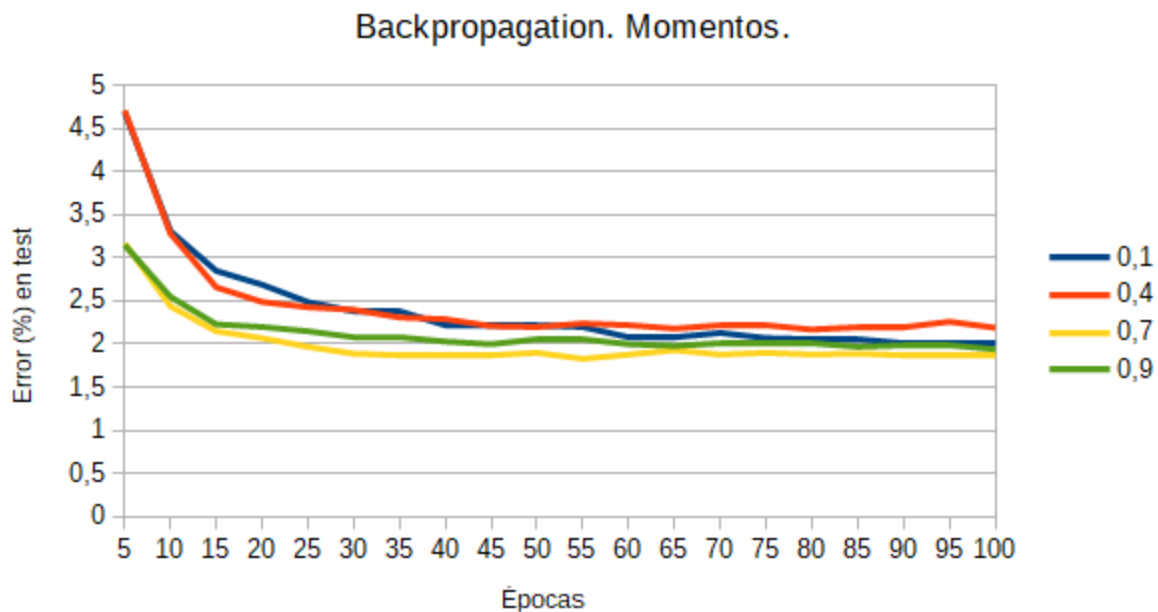


Tal y como podemos observar el modelo con una tasa de 0.1 es el que proporciona un error (%) sobre prueba mayor. Esto es debido a que las tasas de aprendizaje grandes provocan que el algoritmo realice un aprendizaje demasiado rápido y por tanto no sea capaz de encontrar el mínimo. Sin embargo un modelo con una tasa de aprendizaje demasiado pequeña necesitará más épocas para aprender y obtener una mejor capacidad de generalización. Por lo tanto, tal y como se puede comprobar en el experimento, el modelo con una **tasa de aprendizaje de 0.07** es el que obtiene una tasa de error menor sobre prueba, en concreto de **1,94%**. Por lo tanto utilizaremos esta tasa de aprendizaje para los sucesivos experimentos.

## Momento

Con el segundo experimento se pretende conseguir disminuir la tasa de aprendizaje anterior introduciendo, para ello, una de las posibles técnicas de optimización: el **momento**. Este parámetro ayuda a suavizar la actualización de los valores de los pesos con el objetivo de minimizar las oscilaciones que puede realizar el algoritmo durante el aprendizaje. Cuanto más elevado sea, más reducimos la velocidad de aprendizaje provocando una modificación de los pesos más sutil. Si bien para aprovechar el potencial que ofrece este parámetro su valor debe ser variable, siendo menor al comienzo del aprendizaje para acelerar su velocidad y aumentarlo progresivamente conforme nos vamos acercando al mínimo, en mi caso va a ser un **parámetro estático**.

En base a lo anterior, en este experimento he analizado la tasa de error sobre el conjunto de prueba con **cuatro momentos** diferentes y una topología similar al caso anterior, incorporando la tasa de aprendizaje que mejor resultado ha obtenido.



Tal y como podemos observar, con un momento más pequeño se obtienen tasas de error sobre el conjunto de prueba mayores que con momentos más grandes puesto que estos reducen más la velocidad de aprendizaje con la ventaja de, a su vez, poder explorar el terreno actual de manera más detenida en busca del mínimo. Sin embargo un momento demasiado alto puede frenar demasiado la velocidad del algoritmo ralentizando su aprendizaje e impidiendo obtener una mejor tasa de error en un mismo número de épocas que un momento un poco más bajo. Es por ello por lo que el **momento 0.7** es el que mejores resultados proporciona puesto que obtiene una tasa de error sobre prueba de un **1,83% en la época 55** y, por ello, es el que se utilizará en los siguientes experimentos.

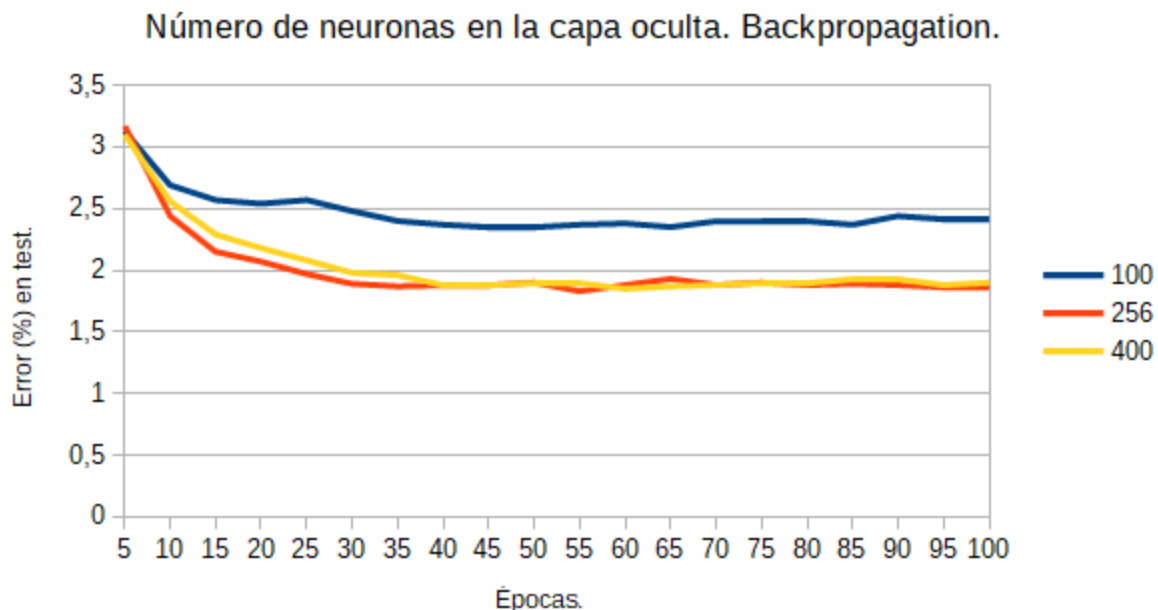
### Número de neuronas en las capas ocultas

Si bien las neuronas ocultas tienen como objetivo obtener características de la imagen para ser capaces de identificar patrones que ayuden a identificar, en este problema, el dígito en cuestión, podemos pensar que cuantas más neuronas compongan la capa oculta, mayor capacidad de extracción de características. Este hecho introduce el tercer experimento en el que se van a



entrenar tres modelos, cada uno de los cuales dispondrá de un número distinto de neuronas en su capa oculta, pero en todos se utilizarán los dos parámetros anteriores que han proporcionado los mejores resultados en los dos experimentos previos.

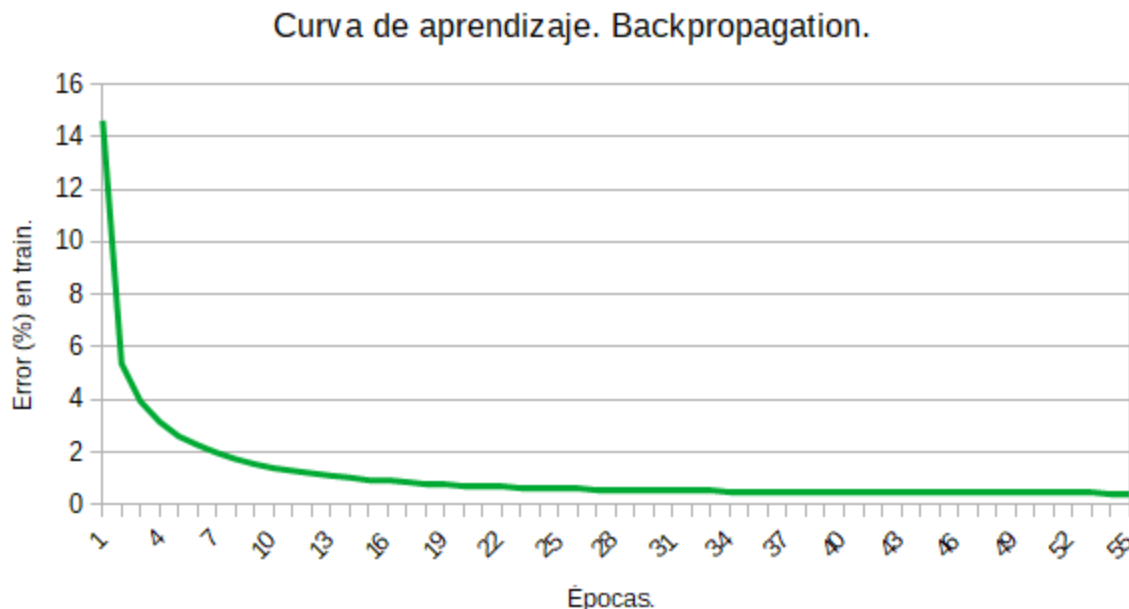
El primer modelo cuenta con **100 neuronas ocultas**, el segundo dispondrá de **256 neuronas** y el último estará compuesto por **400 neuronas** en su única capa oculta.



Tal y como podemos comprobar en el gráfico una sola capa oculta con 100 neuronas no es capaz de extraer suficientes características de la imagen como para proporcionar una tasa de error por debajo del 2%. No obstante, si bien con 400 neuronas se obtiene una tasa de error sobre el conjunto de prueba de 1.85% no supone una diferencia considerable con respecto a una capa oculta con 256 neuronas puesto que proporciona un error dos centésimas más bajo, el coste computacional que supone entrenar un modelo de estas características es mucho menor que una red neuronal con 400 neuronas en la capa oculta. Por tanto, como conclusión podemos afirmar que no por añadir más neuronas a la capa oculta esta va a ser capaz de extraer un mayor número de características que ayuden a proporcionar una tasa de error menor. Así, en el siguiente experimento se establecerá una red neuronal multicapa cuya capa oculta se compone de **256 neuronas**.

En resumen, podemos determinar que la mejor configuración para una red neuronal multicapa, con una capa oculta, consiste en establecer la tasa de aprendizaje a 0.07, un momento de 0.7 y 256 neuronas ocultas. Asimismo también se ha comprobado que la menor tasa de error sobre el

conjunto de prueba se obtiene en la época 55, por lo tanto la **curva de aprendizaje** que a continuación se muestra será la de una red multicapa cuyos parámetros han sido inicializados con estos valores y que ha sido entrenada durante 55 épocas.



Tal y como se puede observar comienza con una tasa de error de entrenamiento de algo más de un 14% y continúa descendiendo rápidamente. La razón de ello es que en la primera época los pesos son inicializados aleatoriamente y por lo tanto no disponen de unos valores ajustados para realizar una buena predicción. No obstante, conforme se va desarrollando el aprendizaje la tasa de error sobre entrenamiento se reduce drásticamente puesto que en cada época los valores de los pesos se actualizan de forma constante y por lo tanto la modificación de sus respectivos valores es muy acusada. Sin embargo, a partir de la época 28 la velocidad se reduce bastante debido a que a partir de este instante la modificación de los pesos no es tan significativa como en épocas anteriores, es decir, sus valores están más cerca de los óptimos y por lo tanto a partir de ahora solo surgen pequeños cambios que permiten reducir un poco la tasa de error sobre entrenamiento. Al final de las 55 épocas esta tasa es de **0.435%**.

### Dropout

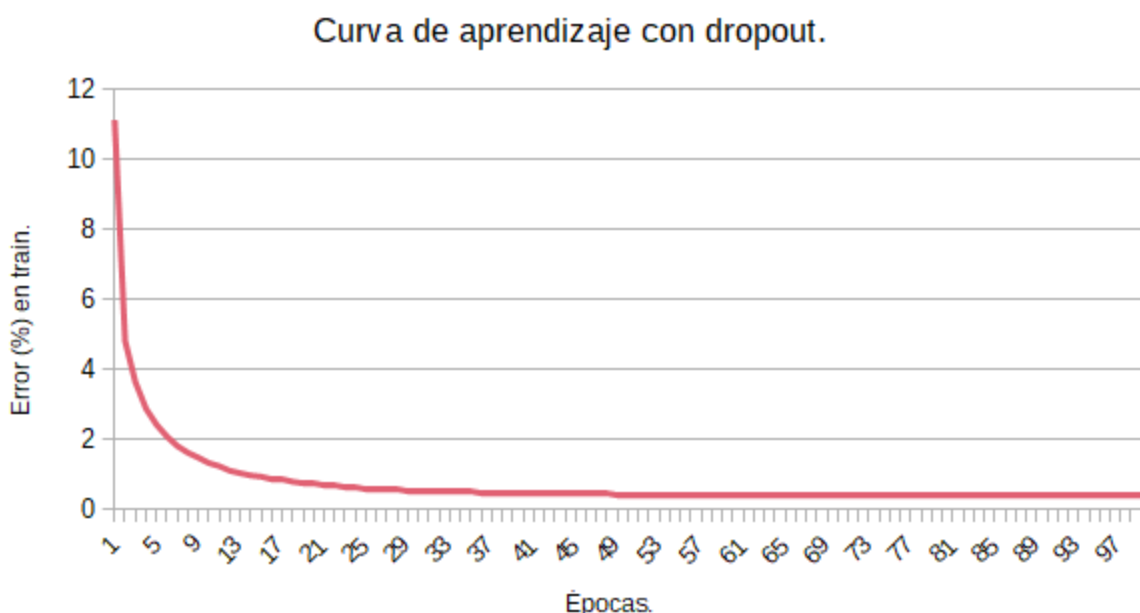
Con el objetivo de continuar reduciendo la tasa de error sobre el conjunto de prueba procedo a incluir una técnica de regularización conocida como *dropout*. Su objetivo consiste en forzar a que cada neurona contribuya con su propia salida sin estar influida por el resultado obtenido de las neuronas que se encuentren a su alrededor. Para ello se desactivan la mitad de las neuronas

de la capa oculta y se entrena el modelo predictivo como en los casos anteriores, solo que esta cualidad se deberá tener en cuenta tanto en el proceso de *forward* como en el de *backward*.

Una variante de esta técnica conocida como ***inverted dropout*** consiste en componer una máscara para desactivar la mitad de las neuronas de la capa oculta pero con la particularidad de incluir el escalado en cada una de las probabilidades con el fin de poder utilizar el modelo entrenado como lo venimos haciendo hasta ahora.

En el siguiente experimento se ha aplicado esta técnica de manera que he desactivado la mitad de las neuronas de la capa oculta de forma alternativa, es decir, las neuronas pares son las que no participarán en toda la fase de entrenamiento. Si bien es aconsejable cambiar las neuronas que están activas y las que no de forma dinámica para cada muestra, en mi caso se establecerá de forma estática por lo que para todas las imágenes las neuronas de la capa oculta que sean pares estarán desactivadas. Luego, en el proceso correspondiente a propagar el error hacia atrás he utilizado la siguiente fórmula  $mask_i \cdot \frac{1}{1-p}$  que se explica en esta [fuente](#). De este modo el error solo se propaga hacia las neuronas activas que han contribuido a la salida de la imagen actual.

A continuación se muestra la **curva de aprendizaje** asociada a entrenar una red neuronal multicapa, con la configuración anterior añadiendo esta técnica durante **100 épocas**.



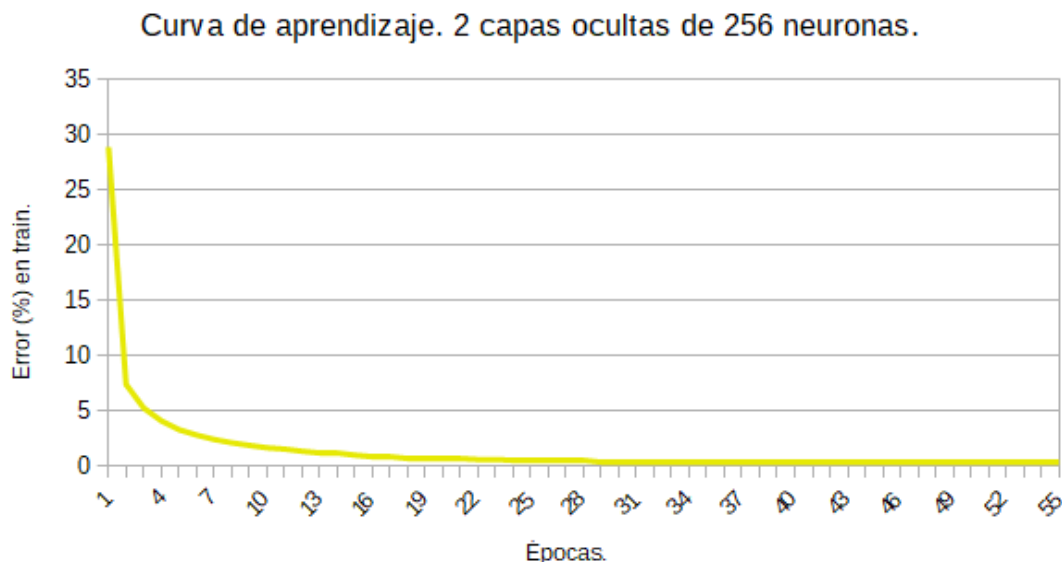
Tal y como se puede comprobar el proceso de aprendizaje para este modelo es muy similar al del caso anterior, es decir, sin incluir *dropout*. Pese a comenzar con un error menor, la tasa de

error sobre el conjunto de prueba es mayor aún habiendo entrenado esta red con el doble de épocas que la anterior, puesto que con este modelo obtenemos un error de **2.41%**. Por lo tanto, en mi caso, el incluir esta técnica de regularización no ha mejorado la capacidad de generalización del modelo básico multicapa.

Investigando acerca del por qué de este resultado he encontrado diversas fuentes, como [esta](#), en la que detallan las causas de que al aplicar *dropout* la tasa de error empeore. En el primero de ellos se desaconseja utilizar esta técnica si se aplica en la capa inmediatamente anterior a la de salida, como es mi caso, puesto que la última capa no puede seguir modificando los pesos por lo que calcula las salidas en base a los valores establecidos por la capa oculta con *dropout*. La segunda situación explica que para redes relativamente sencillas o que ya obtienen tasas de error bajas no se obtienen beneficios al aplicar técnicas de regularización. En mi opinión este caso también se puede aplicar a mi situación puesto que es una red multicapa básica, con una sola capa oculta, y como se observó anteriormente ya es capaz de conseguir una tasa de error sobre el conjunto de prueba menor que el 2%.

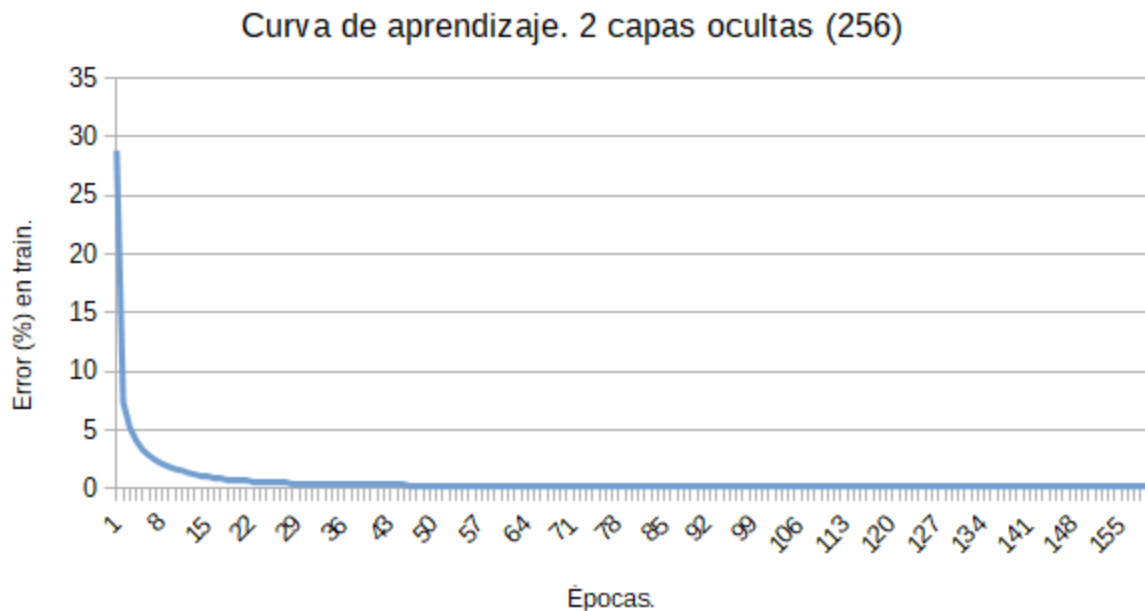
## Dos capas ocultas

En esta última sección entrenaremos un modelo usando la mejor configuración descubierta hasta ahora pero añadiendo una capa oculta más de 256 neuronas. Tras repetir el proceso durante **55 épocas**, la curva de aprendizaje se muestra a continuación.



La tasa de error sobre el conjunto de entrenamiento es de **0.32%**, la cual es menor que la tasa que proporcionaba el mismo modelo con una sola capa oculta (0.435%). Sin embargo, la tasa de

error sobre el conjunto de prueba es mayor, pues se obtiene **2.16%** frente al 1.83% de la red neuronal con una sola capa oculta. Frente a esta desmejoría he aumentado el número de épocas para entrenar este nuevo modelo hasta las **160**, y estos son los resultados obtenidos.




Si bien he aumentado el número de épocas con el objetivo de que la red tuviese más tiempo para adaptar los pesos de las dos capas ocultas, la tasa de error sobre el conjunto de prueba no ha mejorado puesto que se ha obtenido un **1.95%**. Sin embargo la tasa de error sobre el conjunto de entrenamiento sí que ha mejorado hasta reducirse en **0.235%**. En base a estos resultados puedo afirmar que aunque el error sobre las imágenes de entrenamiento se reduzca, no significa que la capacidad de generalización del modelo mejore.

## Conclusiones.

Como conclusiones finales de esta práctica podemos determinar que para obtener un modelo predictivo con una buena capacidad de generalización y que proporcione una razonable tasa de error sobre prueba, es necesario añadir al menos una capa oculta que sea capaz de extraer características de las imágenes que ayuden en el proceso de aprendizaje. Por lo tanto, mediante redes tan sencillas como la primera entrenada, no se consiguen buenos resultados y lo que es aún peor, no existen formas de mejorar su capacidad de predicción respetando la topología.

Asimismo, me hubiese gustado poder terminar y entrenar una **red convolutiva**, puesto que pese a haberla implementado al completo, debido a su enorme complejidad y al estrés por las



entregas de prácticas de otras asignaturas me ha sido imposible encontrar los errores que no me permitían entrenar un modelo de estas características adecuadamente. Esto se debe a que el modelo de CNN que he implementado obtiene como primera tasa de error sobre el conjunto de entrenamiento un **73%**, lo cual indica que debe haber errores en el proceso de generar las salidas utilizando todas las capas (*forward*) y/o en el método de propagación de errores.