

Redes neuronales

Departamento de Computación - FCEyN - UBA

Trabajo práctico 1  
Redes directas multicapa

Matías Battocchia

23 de julio de 2018

# Introducción

El objetivo de este trabajo es —dados problemas del mundo real— implementar y experimentar soluciones a los mismos, utilizando redes neuronales artificiales bajo el paradigma del aprendizaje supervisado, construyendo modelos de redes neuronales *feedforward* multicapa que tengan una capacidad aceptable de generalización.

El código del trabajo es accesible desde el repositorio <https://github.com/matiasbattocchia/redes-neuronales>.

## Optimización

Elegir una **tasa de aprendizaje** apropiada para el descenso por gradiente puede ser difícil. Una tasa demasiado pequeña lleva a una convergencia lenta, mientras que una tasa demasiado grande puede impedir la convergencia y causar que la función objetivo fluctúe alrededor del mínimo e incluso diverja.

La tasa de aprendizaje puede ser *templada* durante el entrenamiento (*annealing*), es decir ser reducida de manera predefinida, en función de la época o cuando la disminución del error entre épocas cae por debajo de cierto umbral. Estas planificaciones, de todas formas, tienen que ser definidas de antemano y por lo tanto no se adaptan a las características de los datos.

La superficie de la función objetivo que es no convexa y presenta una multiplicidad de mínimos locales, podría pensarse como una pelota de golf. La optimización de los parámetros lleva a descender hacia alguna de las cuencas de la pelota. La superficie suele ser marcadamente irregular por lo que esperaríamos encontrar dentro de una cuenca, cuencas más pequeñas. Una tasa de aprendizaje alta ayudaría a entrar a una cuenca rápidamente, a

su vez pasaría por alto a las cuencas más pequeñas, por lo que llegado cierto momento el aprendizaje se estanca. El templado reduce la tasa de aprendizaje para que el aprendizaje continúe.

Implementé un templado exponencial decreciente. Reduce la tasa de aprendizaje en un 63 % cada determinada cantidad de épocas (coeficiente de adaptación).

La tasa de aprendizaje es un hiperparámetro que se aplica en la actualización de todos los parámetros, es decir, tiene un efecto general. El **momento** es una técnica simple para darle un tratamiento particular a cada parámetro. Hace que la optimización se comporte como una pelota que rueda cuesta abajo, ganando cada vez más velocidad en su camino. El momento incrementa para las dimensiones cuyos gradientes apuntan siempre en el mismo sentido y disminuye en las dimensiones con gradientes que alternan su sentido frecuentemente, lo que equivale a moverse con mayor celeridad en las dimensiones con pendientes suaves y a amortiguar las oscilaciones en las dimensiones con perfil de canaleta.

Implementé el momento de Nesterov (NAG). Dejé el coeficiente de momento en 0,9 por ser un valor recomendado.

Por otro lado hay técnicas (RMSprop, Adadelta, Adam, etc.) que adaptan la tasa de aprendizaje individualmente para cada parámetro. Son especialmente útiles cuando los atributos de los datos tienen distintas frecuencias (datos esparsos): podríamos no querer actualizarlos a todos en la misma extensión sino realizar mayores actualizaciones en aquellos que ocurren menos. La tasa de aprendizaje deja de ser un hiperparámetro y es reemplazado por otros que suelen ser mejor comportados.

Por completitud menciono los métodos de segundo orden (métodos de Newton y cuasi-Newton) que son una alternativa al descenso por gradiente aunque computacionalmente más costosos. Al utilizar información sobre la curvatura de la función objetivo logran una convergencia más directa.

## Funciones de costo

Implementé las siguientes funciones: error cuadrático medio (ECM), error absoluto medio (EAM), entropía cruzada. Esta última es especialmente útil para las tareas de clasificación multiclase. ECM y ACM son similares; ECM penaliza más los errores grandes ya que el residuo está elevado al cuadrado, ACM presenta un gradiente más robusto sobre todo cuando los residuos son menores a la unidad.

## Regularización

A la función de costo le puede añadir algunos términos con restricciones sobre los parámetros para evitar el sobreajuste.

Implementé la regularización  $L_2$ . Penaliza a los parámetros con valores altos haciendo que los pesos en la red sean comparables y por lo tanto utilice todas sus conexiones más o menos por igual.

La regularización  $L_1$ , que menciono aunque no fue implementada, lleva a parte de los pesos a la anulación, por lo que la red termina utilizando un subconjunto de sus conexiones.

$L_2$  fomenta la igualdad entre atributos,  $L_1$  favorece la selección de atributos.

*Early stopping* no fue implementado. Salva los parámetros cada cierta cantidad de épocas. Si el error de validación comienza a aumentar en vez de descender es porque el modelo está sobreajustando y es momento de detener el entrenamiento. El modelo utiliza los últimos parámetros salvados.

## Inicialización de los pesos

Los pesos fueron inicializados tomando valores de una distribución gaussiana con varianza de  $2/n$ , donde  $n$  es la cantidad de conexiones entrantes a la neurona (inicialización He et al.).

Cuando los pesos iniciales no fueron los correctos (problema hasta que corregí la varianza de la distribución) experimenté problemas tanto de *overflow*

como de *underflow*.

## Descenso por gradiente

El tamaño del lote puede ser del conjunto entero (lote, *batch*), de una muestra (estocástico), o un compromiso entre ambos extremos (mini-lote, *mini-batch*).

El descenso por lote realiza cálculos redundantes cuando el conjunto de datos es grande porque calcula gradientes para ejemplos similares antes de cada actualización de parámetros. El descenso estocástico se deshace de esta redundancia al realizar una actualización a la vez. Es más rápido y se necesitan menos épocas. Realiza actualizaciones frecuentes con una varianza alta que causa que la función objetivo fluctúe intensamente. Cuando los lotes son pequeños la varianza de las actualizaciones se reduce, lo que conlleva una convergencia más estable.

El orden de las muestras se barajó en cada época antes de separarlas por lotes.

# Ejercicio 1: Diagnóstico de cáncer de mamas

El conjunto de datos contiene 10 atributos y 1 variable objetivo con 2 categorías, por tanto se trata de una tarea de clasificación binaria. Verifiqué que las clases estuviesen balanceadas; lo estaban (B 215, M 195). Particioné el conjunto en entrenamiento (80 %) y validación (20 %). No usar conjunto de prueba fue una elección.

Como métrica para evaluar el desempeño de los modelos elegí la efectividad (*accuracy*).

Para obtener un puntaje de referencia contra el cual comparar posteriores experimentaciones entrené un estimador **gradient boosting**. Como el algoritmo está basado en árboles de decisión los datos no requirieron preprocesamiento. Usé la clase `GRADIENTBOOSTINGCLASSIFIER` con hiperparámetros máxima profundidad 5, número de estimadores 300, resto de los valores por defecto. Obtuve una efectividad (validación) de 0,85.

Para la redes neuronales utilicé el siguiente preprocesamiento: los atributos fueron estandarizados, la variable objetivo (tipo *string*) fue codificada. Cuidé de entrenar la estandarización solo con el conjunto de entrenamiento.

En este ejercicio me enfoqué en explorar hiperparámetros relacionados a la optimización de la función objetivo, dejando constante la estructura de la red. Las redes estuvieron compuestas por una capa de entrada de 10 unidades, una capa oculta con 32 unidades, una capa de salida con una única unidad. Todas las redes fueron inicializadas con los mismos pesos, corrieron la misma cantidad de épocas (1000) y recibieron las muestras de entrenamiento en el mismo orden aleatorio.

Luego de pruebas preliminares entrené una red *base* con las características función de activación logística, función de costo error cuadrático medio, descenso por gradiente estocástico, tasa de aprendizaje 0,1. El resto de las redes fueron variantes de esta, en las que modifiqué solo un hiperparámetro por vez para poder observar comparativamente su efecto. La red base tuvo una efectividad (validación) de 0,90, superando al estimador de referencia.

Elegí la función logística porque es beneficioso hacer coincidir la imagen de la función de activación de la última capa con el rango de la variable objetivo (valores entre 0 y 1). Para evaluar el desempeño de las redes, las predicciones fueron binarizadas utilizando un umbral de 0,5.

En la figura 1 se encuentran las evoluciones de los errores de entrenamiento y validación de las redes entrenadas. La primera subfigura corresponde a la red base.

Los hiperparámetros del resto de las redes entrenadas fueron: Tasa de aprendizaje 0,9 (aprendizaje alto) y 0,01 (aprendizaje bajo). Coeficiente de momento 0,9. Coeficiente de regularización 0,0003. Coeficiente de adaptación 1000. Tamaños de lote conjunto entero (batch) y 8 muestras (mini-batch). Función de costo error absoluto medio.

El mejor estimador fue el que utilizó regularización, alcanzando una efectividad (validación) 0,95; ver subfigura inferior central. Su matriz de confusión resultó:

Clase/Predicción	Maligno	Benigno
Maligno	40	4
Benigno	0	38



Figura 1: Evolución de los errores de entrenamiento y de validación de las redes entrenadas en el ejercicio 1. Cada red es una variante de la denominada red base en un único hiperparámetro. La estructura 10:32:1 se mantuvo en todos los casos.



## Ejercicio 2: Eficiencia energética

El conjunto de datos de este ejercicio tiene 8 atributos y 2 objetivos; se trata de un problema de regresión bivariada. Corroboré que las variables objetivo tuvieran media y varianza comparables para que ninguna dominase sobre la otra. Particioné el conjunto en entrenamiento (80 %) y validación (20 %).

Preprocesé los datos de dos maneras. Los atributos numéricos fueron estandarizados, los atributos categóricos fueron codificados (*one-hot encoding*). Como resultado de este último proceso la cantidad de atributos ascendió a 16.

Escogí como métrica el coeficiente de determinación ( $R^2$ ), adecuado para tareas de regresión. Este coeficiente va de 0 a 1, siendo 1 el mejor escenario posible—el modelo ajusta perfectamente a los datos.

Al igual que en el ejercicio 1 se utilizó un estimador de referencia. En este caso **cuadrados mínimos con regularización  $L_2$**  (clase `RIDGE REGRESSION`) configurado con  $\alpha = 0,001$  y resto de hiperparámetros por defecto. Obtuve un  $R^2$  (validación) de 0,91.

En este ejercicio dejé fijos varios hiperparámetros en valores que, experimentando, encontré satisfactorios. Tasa de aprendizaje  $1e^{-5}$ , coeficiente de momento 0,9, tamaño de lote 4 muestras, función de activación tangente hiperbólica, función de activación de la última capa identidad, función de costo error cuadrático medio.

Para tareas de regresión es importante que la función de activación de la capa de salida cubra el rango  $(-\infty, \infty)$ . En cambio no había una razón especial por la cual usar tangente hiperbólica en el resto de las unidades. Tiene un funcionamiento similar a la función logística ya que ambas son

curvas sigmoideas, con la diferencia de poseer un rango de  $(-1, 1)$ .

Exploré estructuras de redes directas variando la cantidades de capas ocultas y de unidades. La capa de entrada quedó de 16 unidades y la de salida, en 2. La duración del entrenamiento dependió de la estructura de la red. La figura 3 muestra la evolución de los errores de las redes entrenadas.

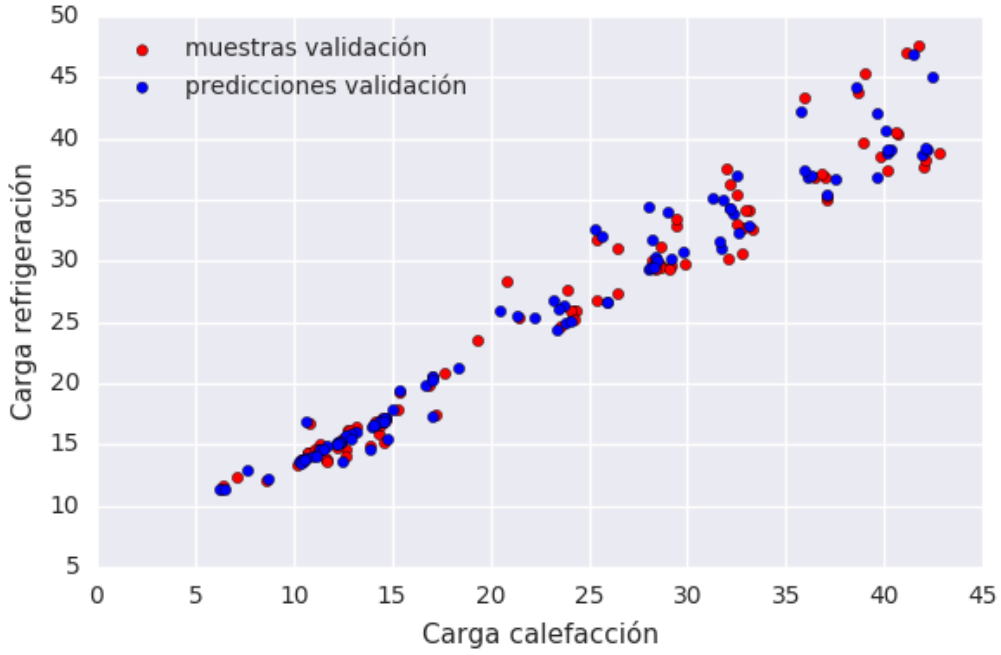


Figura 2: Distribución de las variables objetivo de la red 16:8:8:2.

Haciendo uso de la *navaja de Occam*, frente a un problema con multiplicidad de soluciones plausibles, la hipótesis con menos asumpciones tiende a ser la correcta. Llevado a los estimadores estadísticos, entre modelos con similar desempeño, es preferible el que tenga menos parámetros, ya que tiende a generalizar mejor.

El estimador elegido, 16:8:8:2, tuvo una efectividad (validación) de 0,9948 y en comparación con otros de puntaje similar (por ejemplo la red 16:16:16:2 con 0,9962), menos parámetros (298 contra 578); subfigura inferior derecha.

Observo que añadir capas permite alcanzar resultados comparables con menos parámetros. Entiendo que las sucesivas capas elaboran una representación de la capa anterior, logrando atributos más significativos.

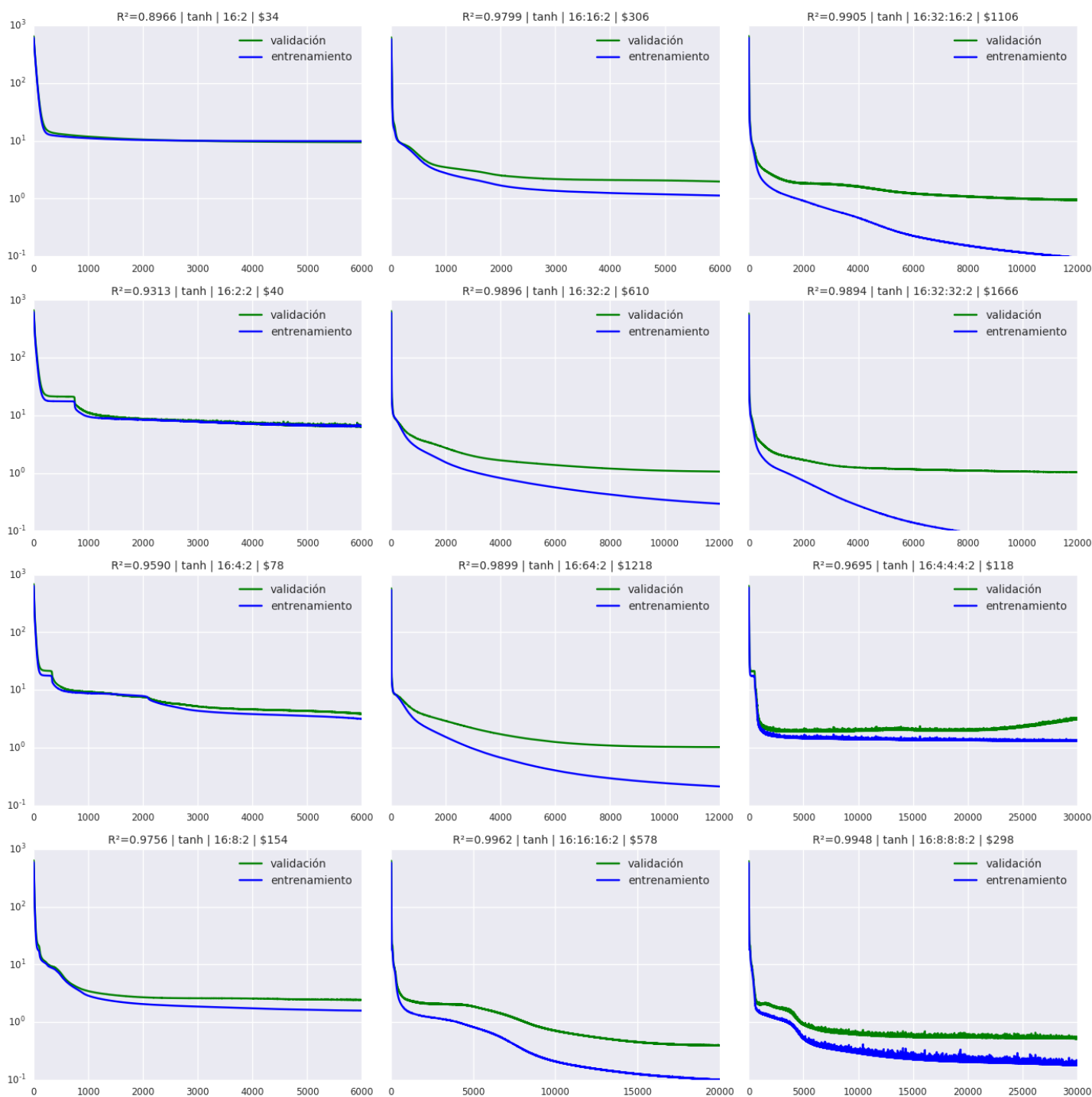


Figura 3: Evolución de los errores de entrenamiento y de validación de las redes entrenadas en el ejercicio 2. Cada red tiene una configuración distinta de capas ocultas y unidades. Salvo la duración del entrenamiento, el resto de los hiperparámetros fueron los mismos para todas las redes.

## Bonus: MNIST

El conjunto de datos MNIST consiste en 70.000 imágenes en escala de grises de dígitos del cero al nueve escritos a mano; viene particionado en conjuntos de entrenamiento (60.000) y prueba (10.000). Cada imagen tiene 28 píxeles de alto y 28 píxeles de ancho, un total de 784 píxeles. Cada píxel tiene un valor entero entre 0 (blanco) y 255 (negro). Conforman los atributos de los datos. La variable objetivo es el dígito de la imagen. Se trata de un problema de clasificación multiclase.

Como preprocesamiento, escalé el valor de los píxeles entre 0 y 1 (estandarizar hubiera arruinado la esparcidad de los datos). Codifiqué la variable objetivo (*one-hot encoding*).

Entrené una red con capas de 784 (entrada), 800 (oculta), 10 (salida) unidades durante 200 épocas usando lotes de 500 muestras. Funciones de activación ReLU y *softmax* en la última capa, función de costo entropía cruzada, tasa de aprendizaje 0,01, coeficiente de momento 0,9, coeficiente de regularización 0,0001. Conseguí una efectividad (prueba) de 0,9825. La evolución de los errores se muestra en la figura 4. La matriz de confusión fue:

Clase/Predicción	0	1	2	3	4	5	6	7	8	9
0	969	0	1	1	1	1	2	1	3	1
1	0	1126	2	1	0	0	2	1	3	0
2	4	1	1010	1	2	0	2	7	5	0
3	0	0	2	991	0	5	0	4	3	5
4	1	0	3	1	965	0	2	1	1	8
5	3	0	0	6	1	873	4	1	3	1
6	5	3	0	1	5	2	942	0	0	0
7	0	4	6	2	0	0	0	1010	3	3
8	2	0	1	6	5	1	2	2	953	2
9	2	3	0	3	7	1	1	3	3	986

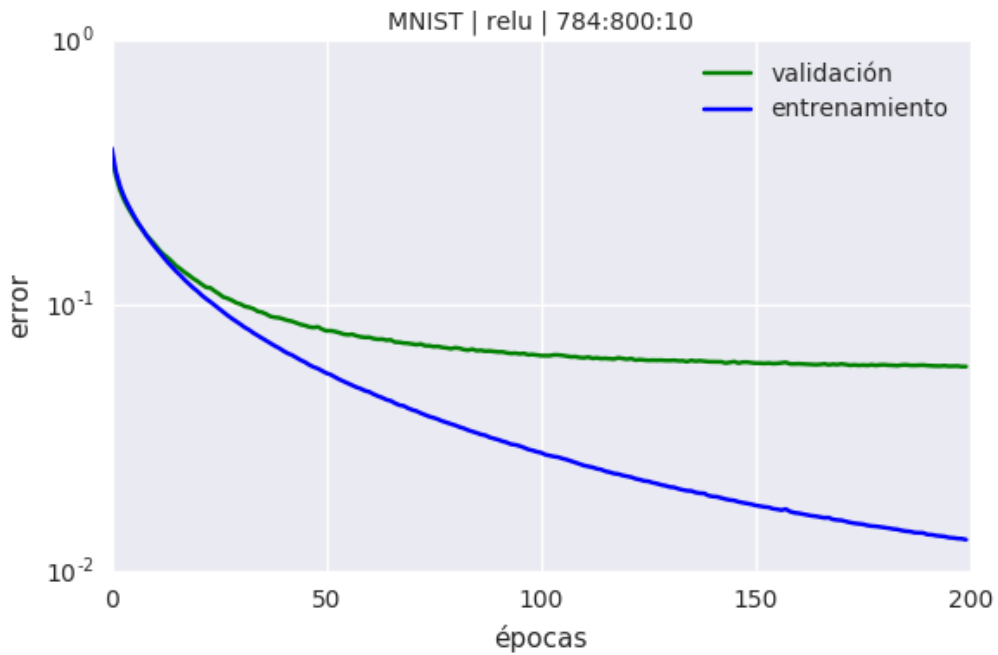


Figura 4: Evolución de los errores de entrenamiento y de validación del estimador utilizado para el conjunto de imágenes de MNIST.

Respecto a algunas elecciones, ReLU es una función de activación que como ventaja no sufre de *gradientes desvanecientes* como sí sucede con las funciones sigmoideas, por lo tanto es útil cuando las capas ocultas son numerosas (no fue el caso). Como contrapartida es una función que no está

acotada (puede explotar) y por otro lado puede quedarse enclavada en cero (puede morir).

Softmax es una función que normaliza la salida de la red y la convierte en un vector de probabilidades, es adecuada para las tareas de clasificación de múltiples clases. A diferencia de otras funciones de activación, opera simultáneamente sobre todas las unidades de una capa. La entropía cruzada es una función de costo que penaliza los errores de manera correcta cuando se modela una distribución multinomial. Cuando las clases son dos, se puede demostrar que una red de dos salidas, softmax y entropía cruzada es equivalente a la red del ejercicio 1: una salida, logística y error cuadrático medio (regresión logística).

En este ejercicio prácticamente no hubo experimentación, recurrí a la siguiente práctica que se desprende de ejercicios anteriores y dio buenos resultados: Usar una red con suficiente capacidad para sobreajustar los datos y aplicar regularización como paliativo. Así asegurarnos de usar un modelo justo, ni demasiado simple —cuando se podría llegar más lejos— ni demasiado complejo. En otras palabras, lograr que el error de entrenamiento baje lo más posible e intentar que el error de validación lo acompañe.































Predicción: 0 Clase: 6 	Predicción: 6 Clase: 0 	Predicción: 3 Clase: 4 	Predicción: 4 Clase: 8 	Predicción: 4 Clase: 6 
Predicción: 5 Clase: 3 	Predicción: 2 Clase: 1 	Predicción: 4 Clase: 8 	Predicción: 7 Clase: 2 	Predicción: 8 Clase: 3 
Predicción: 0 Clase: 9 	Predicción: 7 Clase: 2 	Predicción: 2 Clase: 4 	Predicción: 4 Clase: 2 	Predicción: 5 Clase: 3 
Predicción: 0 Clase: 0 	Predicción: 7 Clase: 7 	Predicción: 9 Clase: 9 	Predicción: 5 Clase: 5 	Predicción: 4 Clase: 4 
Predicción: 7 Clase: 7 	Predicción: 2 Clase: 2 	Predicción: 6 Clase: 6 	Predicción: 8 Clase: 8 	Predicción: 3 Clase: 3 
Predicción: 9 Clase: 9 	Predicción: 6 Clase: 6 	Predicción: 0 Clase: 0 	Predicción: 8 Clase: 8 	Predicción: 4 Clase: 4 

Figura 5: Imágenes del conjunto MNIST clasificadas por el estimador del ejercicio. Las tres filas superiores corresponden a muestras mal clasificadas y las tres inferiores a muestras correctamente clasificadas.