Tarea Perceptron

Sebastián Martínez Santos 176357
Pablo Barranco Soto 151528

Octubre 18, 2020

1 Considera un problema de clasificación binaria, en donde queremos separar imágenes de los dígitos 5 y 9. ¿Qué etiquetas usarías y en qué formato?

Usaríamos etiquetas binarias. Sin perdida de generalidad podríamos asignar 0 a lo 5 y 1 a los 9 para hacer mas sencilla la clasificación.

2 ¿Por qué es conveniente usar la activación sigmoide en la capa de salida de una red neuronal para clasificación binaria?

La forma de la funcion sigmoide es la siguiente:

$$\sigma(z) \equiv \frac{1}{1+e^{-z}} \tag{1}$$

Con $z = \vec{w} * \vec{x} + b$ siendo w los pesos, x el input de la neurona y b el bias.
La motivación de usar esta función pueden ser varias.

- Al usar esta función podemos mapear funciones de $\mathbb{R} \to [0, 1]$, y asignar de esta forma una probabilidad a un input de pertenecer o no pertenecer a cierta clase.
- Usar esta función en la capa final nos daría un $\hat{y} \in [0, 1]$, con el cual podríamos hacer clasificación binaria.
- Cuando se utiliza esta función, al hacer backpropagation tiene un buen comportamiento a diferencia de un perceptron normal. En otras palabras los cambios son mas suaves sobre los pesos de la red (también para los bias).
- Adicionalmente es conveniente usar la activación sigmoide en las capas interiores para resolver problemas no lineales.

3 Investiga qué forma tiene la función softmax, y explica por qué es conveniente en la activación de salida de una red neuronal para clasificación multi-clase.

La forma de la función softmax es la siguiente:

$$\sigma(\vec{z}) \equiv \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2}$$

Las ventajas de usar esta función, sigue un poco la misma lógica que la función *sigmoide*, pero para hacer clasificaciones de mas de una clase. Esta función recibe valores de $\mathbb{R}^n \to \mathbb{R}^n$. Al usar este vector en la capa final obtenemos una clasificación, en donde el resultado de $\sigma(\vec{z}) = \hat{y}$ es un vector, en donde las entradas y_i contienen la probabilidad de que el input $\vec{x}$ pertenezca a la i-esima clase. En donde se cumple que

- $\hat{y}_i \in [0, 1]$
- $\sum_{i=1}^{n} \hat{y}_i = 1$

4 Investiga qué forma tiene la función ReLU, y explica en qué tipo de capas de redes neuronales se usa comúnmente.

La forma de la función ReLu es la siguiente:

$$f(x) = \max\{0, x\} \tag{3}$$

En la practica es común usar este tipo de funciones en las capas interiores de las redes neuronales. Se usa sobre todo en redes neuronales con muchas capas,ya que al usar la función sigmoide cuando el gradiente del error es muy pequeño, muchas veces no se corrigen las primeras capas de la neurona pues llega muy diluido el cambio al hacer *back propagation*. Es por eso que es conveniente sustituir por funciones *ReLu*.

5 Usa entre 1 y 5 capas (tú decide el número de nodos por capa), y trata de maximizar el desempeño de clasificación para el problema de 10 clases. En una tabla, reporta los hiperparámetros de tus 5 mejores modelos y su desempeño de clasificación.

Creamos 5 diferentes modelos. Cada uno con diferentes números de capas internas dentro de la red neuronal. El numero de nodos que seleccionamos fue tratando de hacer espaciado entre el input 784 y el output 10, dependiendo el numero de capas que tendrá el modelo.

Los resultados obtuvidos se muestran en la siguiente tabla.

Layers	Train	Validation
1	0.899450	0.9067
2	0.932250	0.9329
3	0.947267	0.9459
4	0.959150	0.9522
5	0.966050	0.9594

Figure 1: Accuracys Table

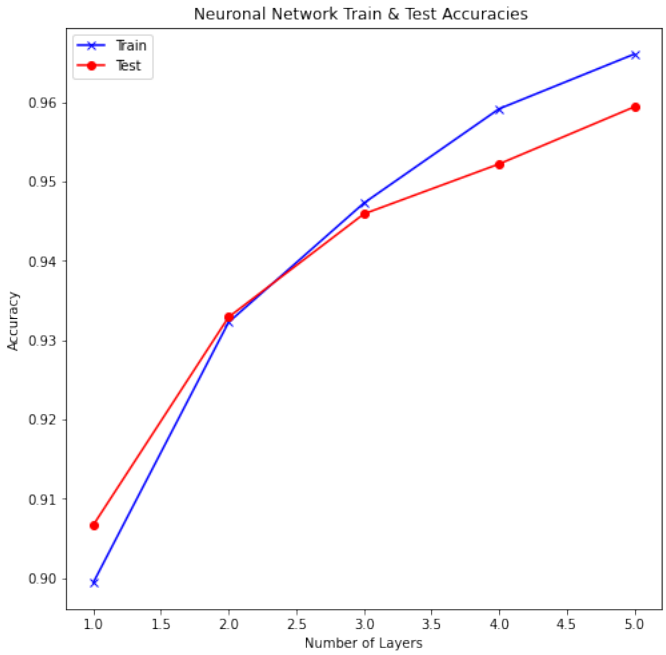

Figure 2: Accuracys Graph

Podemos ver una tendencia positiva. Conforme aumentamos las capas de la neurona obtenemos mejores resultados de precisión. Al ver estos resultados podemos ver que al aumentar las capas la variación entre precisiones no es tan grande, ya que desde que usamos pocas capas tenemos un resultado decente.

6 Para el mejor modelo obtenido, reporta sus curvas de desempeño respecto a las épocas de entrenamiento. También reporta la matriz de confusión para el set de test.

Nuestro mejor modelo fue aumentando el numero de capas (es decir con 5 capas) de la red neuronal, y usando la función de activación *elu*, la cual es una implementación de la función *relu*. Con la cual se evita el problema de dejar neuronas "muertas" en el proceso de back-propagation.

Reportamos las curvas de desempeño y la matriz de confusión a continuación.

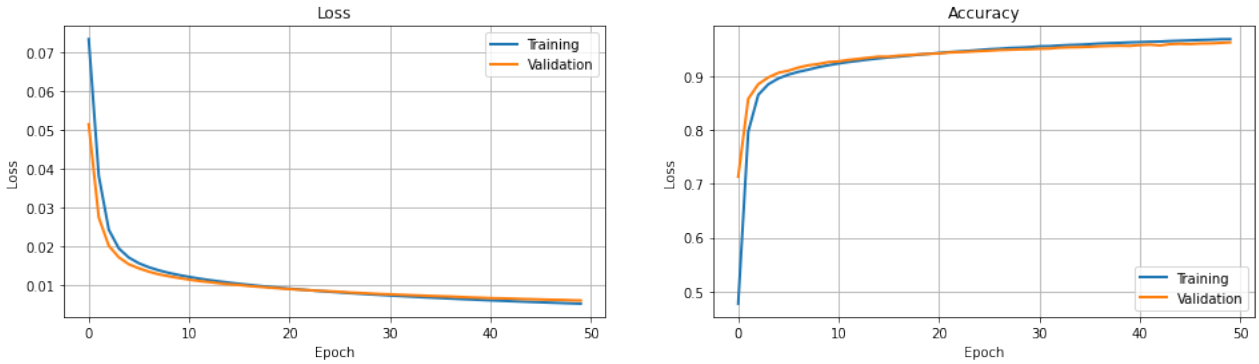

Figure 3: Curvas de desempeño

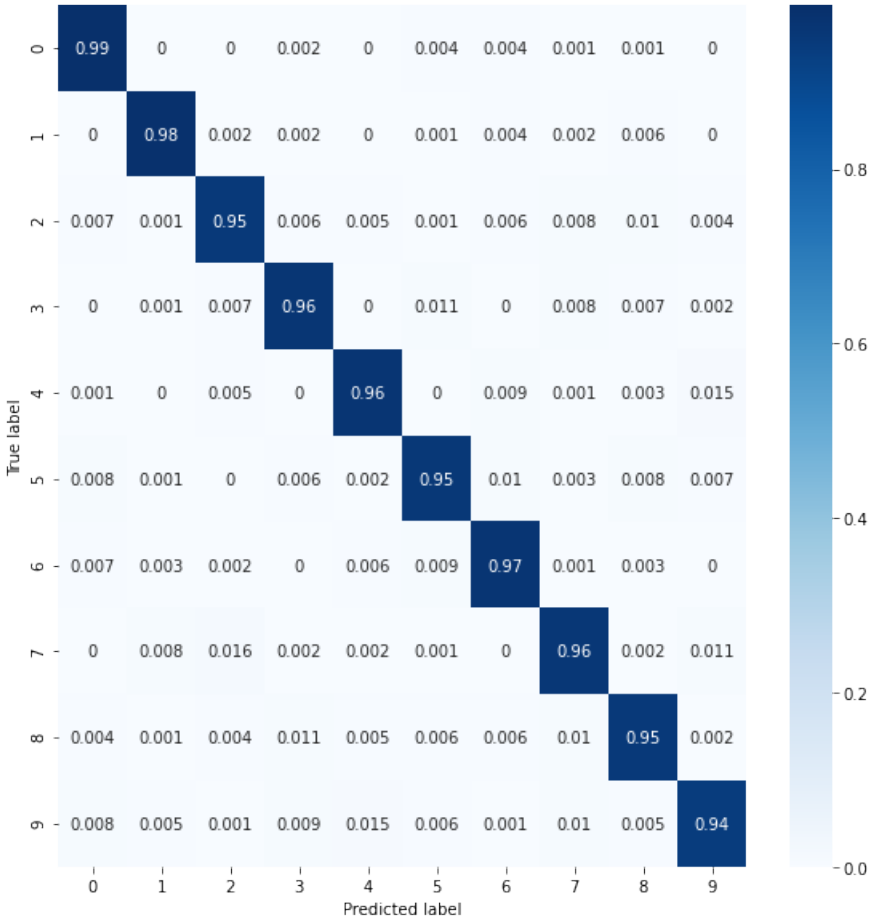

Figure 4: Matriz de Confusión

Obteniendo al final un resultado de 0.968 en entrenamiento y 0.962 de precisión en validación.La cual es considerablemente mayor a las obtenidas con *relu*.

7 Usando tu mejor modelo, ¿Notas algún cambio en el desempeño si usas *categorical crossentropy* como pérdida, en vez de *mse* ? Si sí, ¿cuál es el cambio?

Reportamos las curvas de desempeño y la matriz de confusión a continuación al usar *categorical crossentropy*.

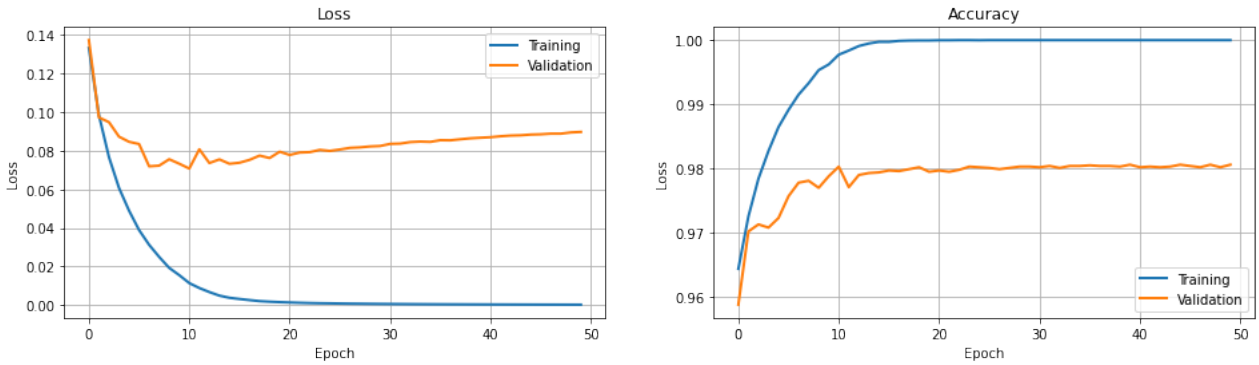

Figure 5: Curvas de desempeño

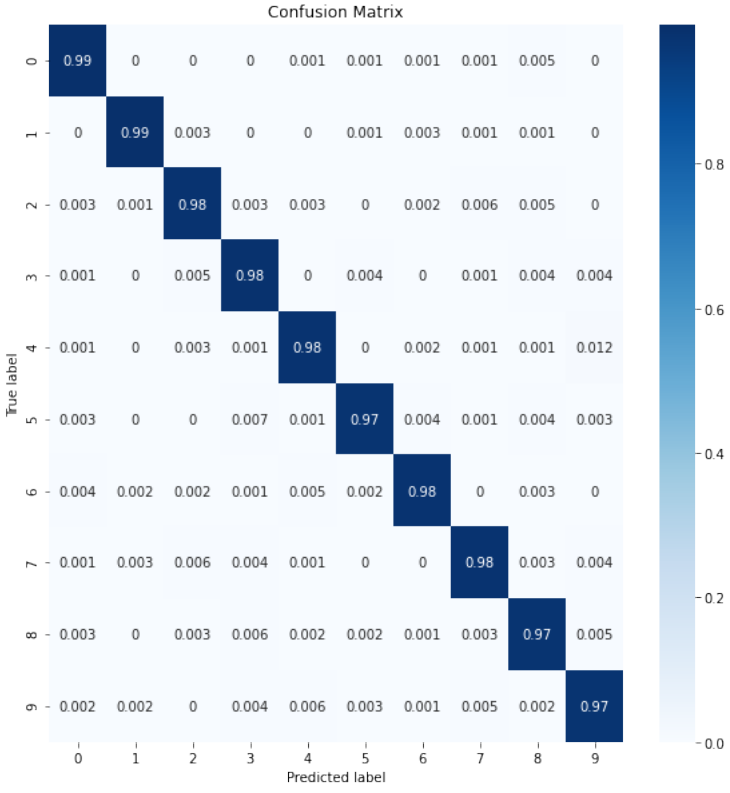

Figure 6: Matriz de Confusión

Al usar la función de perdida *crossentropy* podemos observar que hay un considerable cambio positivo en nuestro mejor modelo obtenido. Esto lo aseguramos pues se obtuvieron 1 precisión en entrenamiento y

0.98 de precisión en validación.

Esto se debe a que pasamos de usar el *mse* a la función *crossentropy* dada por la siguiente ecuación.

$$C = -\frac{1}{n}\sum_{x}[y\ln(a) + (1-y)\ln(1-a)] \tag{4}$$

Donde:

- $z = \sum_j w_j x_j + b$
- $a = \sigma(z)$
- n es el numero total de items en el set de entrenamiento.
- x denota suma es sobre todos los inputs de x
- y es la clasificación deseada.

Esta función es mejor, pues tiene el beneficio, a diferencia del error cuadrático medio, de evitar una lenta tasa de aprendizaje.