Tarea 1: Red Neuronal

Paula Ríos

6 de Septiembre

1. Introducción

El objetivo de esta tarea es implementar una red neuronal, testear su comportamiento y probarla para la realización de predicciones o regresiones con un dataset real. La red neuronal cuenta con *layers* (*hidden* o output) y estos *layers* con cierta cantidad de neuronas.

2. Clasificación de Dígitos

Una vez implementada la red neuronal, esta se puede utilizar para realizar predicciones. Para poner en prueba esto se selecciona un dataset y se realizan experimentos para determinar cómo afectan las distintas variables al comportamiento del clasificador.

2.1. Dataset Utilizado

El dataset seleccionado (1) consiste en datos para reconocimiento óptico de dígitos escritos a mano. Los datos fueron previamente procesados de forma tal que los mapas de bits 32x32 se dividen en bloques no superpuestos de 4x4 y el número de píxeles se cuentan en cada bloque. Esto genera una matriz de entrada de 8x8 donde cada elemento es un entero en el rango 0-16. Con esto cada input consiste un 64 números enteros entre 0 y 16, y las posibles clases corresponden a dígitos, por lo que serían números entre 0 y 9.

El dataset está dividido en dos, uno para *training* y otro para *testing*, con 3823 y 1797 datos cada uno.

2.2. Características de la Red Neuronal

El clasificador consiste en una red neuronal con 64 inputs y 10 outputs, uno por cada clase, donde el resultado final vendría ser el índice del output con el mayor valor. La red tiene las siguientes características:

- Pesos iniciales con valores aleatorios entre -1 y +1.
- *Bias* inicial 0.5
- *Learning rate* entre 0.1 y 0.5

2.3. Configuración de la Red Neuronal

Para optimizar lo más posible el clasificador, se experimenta usando distintas configuraciones de red, probando con distintas cantidades de *hidden layers* y distintas cantidades de neuronas por *layer*. Los resultados obtenidos son los siguientes:

# hidden layers	# neuronas layer 1	# neuronas layer 2	# neuronas layer 3	Tasa de éxito
1	20			0.919
1	30			0.926
1	35			0.927
2	40	20		0.922
2	35	35		0.931
2	35	30		0.923
3	35	35	35	0.9265

Donde la configuración destacada correspondería a la de mejor desempeño. Todos las pruebas fueron realizadas con *learning rate* 0.5.

2.4. Learning Rate

Además de experimentar con la configuración, se prueba con distintos *learning rates* para ver como esto afecta el desempeño del clasificador. Los resultados obtenidos fueron los siguientes:

Tasa de éxito	Learning rate
0.9460	0.6
0.9449	0.5
0.9482	0.4
0.9471	0.3
0.9454	0.2
0.9387	0.1

2.5. Tiempo de procesamiento

Con la configuración de mejor desempeño (2 *hidden layers* de 35 neuronas), un *learning rate* de 0.4 y 5 *epochs* el tiempo que se demora en realizar el proceso de entrenamiento y testeo de los datos es de alrededor de 270 segundos, obteniendo una tasa de éxito de 0.948.

2.6. Efecto del orden de los datos

Además, se experimenta cambiando el orden de los datos de entrada aleatoriamente, con lo cual se obtienen las siguientes tasas de éxito para 3 ejecuciones (con la configuración descrita en la sección anterior):

Tasa de éxito	Tiempo de ejecución
0.9577	279.29
0.9437	260.62
0.9315	287.82

De esto se concluye que el orden de los datos, aunque puede llegar a favorecer el desempeño del clasificador, éste es bastante aleatorio.

3. Referencias

1. https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

4. Anexo

Cómo ejecutar

Para usar el clasificador se debe correr el archivo llamado OptDigitsClassifier.py, el cual realiza un entrenamiento con el dataset de *training* con 5 *epochs* y un testeo con el dataset correspondiente y printea su tasa de éxito y tiempo de ejecución.