Máster Universitario en Ingeniería Informática

de la Universidad de Granada

Práctica de redes neuronales

Reconocimiento óptimo de caracteres MNIST

Inteligencia Computacional (IC)

Autora

Marina Jun Carranza Sánchez

Índice

Índice de figuras

Índice de cuadros

1. Introducción

Este se enfoca en

1.1. Herramientas y entorno

Se ha optado por utilizar bibliotecas que implementan dichos algoritmos y están disponibles públicamente.

Se ha utilizado Google Colab.

import numpy as np import matplotlib.pyplot as plt from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Dense from tensorflow.keras.utils import to_categorical from sklearn.model_selection import train_test_split

Antes de nada, se han definido dos funciones: *cargar_imagenes()* y *cargar_etiquetas()*.

La primera se encarga de abrir el fichero train-images.idx3-ubyte

2. Red neuronal simple

Red neuronal simple, con una capa de entrada y una capa de salida de tipo softmax: 7.8 % de error sobre el conjunto de prueba y 5.6 % de error sobre el conjunto de entrenamiento (tiempo de entrenamiento necesario: sobre un minuto usando una implementación en un lenguaje interpretado como Matlab).

2.1. Fundamentos teóricos

tfhtfhtr

2.2. Implementación

2.3. Análisis de resultados

3. Red neuronal multicapa

Red neuronal multicapa, con una capa oculta de 256 unidades logísticas y una capa de salida de tipo softmax: 3.0 % de error sobre el conjunto de prueba y 0.0 % de error sobre el conjunto de entrenamiento (tiempo de entrenamiento necesario: unos cuatro minutos usando una implementación en un lenguaje interpretado como Matlab).

3.1. Fundamentos teóricos

3.2. Implementación

3.3. Análisis de resultados

4. Red neuronal convolutiva

Red neuronal convolutiva entrenada con gradiente descendente estocástico: 2.7 % de error sobre el conjunto de prueba (tiempo de entrenamiento necesario: unos 13 minutos usando una implementación en un lenguaje interpretado como Matlab).

4.1. Fundamentos teóricos

4.2. Implementación

4.3. Análisis de resultados

5. Deep learning

Deep learning usando pre-entrenamiento de autoencoders para extraer características de las imágenes usando técnicas no supervisadas y una red neuronal simple con una capa de tipo softmax: del 1.8 % al 2.2 % de error sobre el conjunto de prueba (tiempo de entrenamiento necesario: unos veinte minutos usando una implementación en un lenguaje interpretado como Matlab).

5.1. Fundamentos teóricos

5.2. Implementación

5.3. Análisis de resultados

6. Red neuronal combinada mejorada

6.1. Fundamentos teóricos

6.2. Implementación

1. Dividir train set en validation set
2. Data augmentation
3. Convolutiva
4. EarlyStopping
5. ReduceLROnPlateau

6.3. Análisis de resultados

7. Conclusiones y trabajos futuros

En conclusión, este

8. Bibliografía