Reconocimiento de patrones con redes neuronales

# Clasificación

Instrucciones. Plantea un contexto de clasificación. Indica tipo de objetos de clasificar, características y clases posibles.

Contexto: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Objetos: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Características: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Clases: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Perceptrón

Instrucciones. Resuelve el ejercicio de clasificación de quásares. Haz dos épocas de entrenamiento.

Época 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g(X,W) | y=f(g(X,W)) | ¿Actualización? (sí/no) | w0 | w1 | w2 |
| g(-1,0,0),(0.5,0.4,1)=-0.5 | 0 | d=1, actualizar | 0.3 | 0.4 | 1 |
| g(-1,0,1),(0.3,0.4,1)=0.7 | 1 | d=1, no se actualiza | 0.3 | 0.4 | 1 |
| g(-1,1,1),(0.3,0.4,1)=1.1 | 1 | d=0, actualizar | 0.5 | 0.2 | 0.8 |
|  |  |  |  |  |  |

Época 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g(X,W) | y=f(g(X,W)) | ¿Actualización? (sí/no) | w0 | w1 | w2 |
| g(-1,0,0),(0.5,0.2,0.8)=-0.5 | 0 | d=1, actualizar | 0.3 | 0.2 | 0.8 |
| g(-1,0,1),(0.3,0.2,0.8)=0.5 | 1 | d=1,no actualizar | 0.3 | 0.2 | 0.8 |
| g(-1,1,1),(0.3,0.2,0.8)=0.7 | 1 | d=0,actualizar | 0.5 | 0 | 0.6 |
|  |  |  |  |  |  |

# Conjunto de entrenamiento

Instrucciones. Plantea un conjunto de entrenamiento, donde los objetos tengan al menos tres características (puede ser el mismo contexto que habías propuesto para el ejercicio de clasificación).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objeto | Característica 1 | Característica 2 | Característica 3 | Clase |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Redes neuronales profundas

Instrucciones. Resuelve el ejercicio de clasificación de dígitos con redes por convolución utilizando la librería Keras.

1. Instala Python (de preferencia versión 3.7) y la librería Keras.
2. Corre el código de ejemplo para el reconocimiento de dígitos con el conjunto de datos MNIST.
3. Reporta con una captura de pantalla el resultado que obtuviste.
4. Indica ***en tus propias palabras*** para qué sirve la instrucción

model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=input\_shape))

1. Indica ***en tus propias palabras*** para qué sirve la instrucción

model.add(MaxPooling2D(pool\_size=(2, 2)))

1. Indica ***en tus propias palabras*** para qué sirve la instrucción

model.add(Dropout(0.25))

1. Indica ***en tus propias palabras*** para qué sirve la instrucción

model.add(Flatten())

1. Indica ***en tus propias palabras*** para qué sirve la instrucción

model.add(Dense(128, activation='relu'))

1. Indica ***en tus propias palabras*** para qué sirve la instrucción

model.add(Dense(num\_classes, activation='softmax'))

1. Modifica la arquitectura de la red (capas) y reporta tus resultados. ¿Fueron mejores o peores los resultados y por qué crees que fue así?