Curso de IA

Capítulo 8. Cuestionario

**Para estudiantes**

ⓒ2023 SAMSUNG. Todos los derechos reservados.

La Oficina de Ciudadanía Corporativa de Samsung Electronics posee los derechos de autor de este documento.

Este documento es una propiedad literaria protegida por la ley de derechos de autor, por lo que está prohibida su reimpresión y reproducción sin permiso.

Para utilizar este documento fuera del plan de estudios de Samsung Innovation Campus, debe recibir el consentimiento por escrito del titular de los derechos de autor..

1. ¿Cuál es el problema que se plantea cuando el número de nodos es pequeño al analizar redes neuronales artificiales?
2. Si el número de nodos es pequeño, el número de operaciones a procesar es pequeño, por lo que se realiza rápidamente.
3. No se pueden crear límites de decisión complejos al realizar un modelo de análisis.
4. No se puede realizar el algoritmo de retropropagación que ajusta pesos y umbrales.
5. El número de nodos no tiene nada que ver con el modelo de análisis.
6. ¿Cuál es el concepto que se describe a continuación?

|  |
| --- |
| ***Matriz multidimensional con 3 componentes: Rango, Forma, Tipo*** |

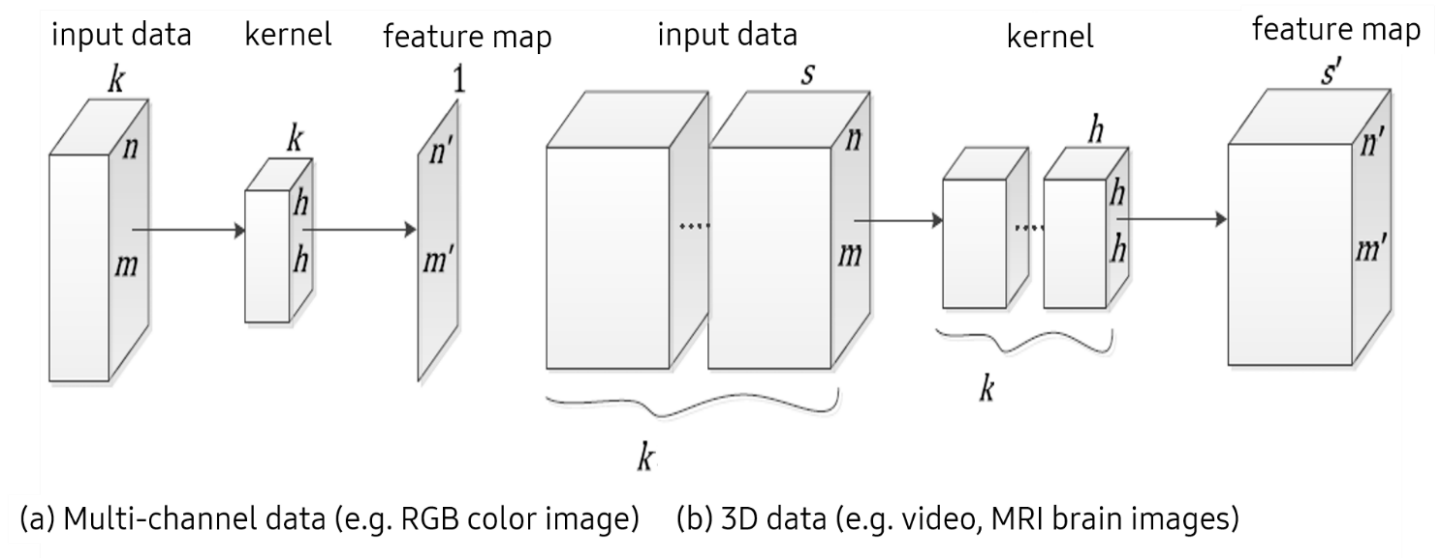
Se esta haciendo referencia al concepto de **Tensor.**

1. ¿Qué ocurre si el error de verificación aumenta constantemente cuando graficamos el error de verificación para cada época utilizando el método de descenso de pendiente por lotes? Además, ¿cómo se puede resolver este problema?

Cuando hay un aumento constante del error de verificación, esto indica overfitting en el modelo, lo que significa que el modelo aprende demasiado bien los datos de entrenamiento, pero este aprendizaje no le sirve para datos futuros o nunca antes vistos lo que lo vuelve poco confiable.

Este problema puede resolverse aplicando técnicas como la regularización, el uso de dropout, detener el entrenamiento a tiempo para evitar un sobreajuste aún mayor, ampliar el conjunto de datos o, en su caso, reducir la complejidad de la red para mejorar su capacidad de generalización ante datos futuros.

1. La figura de la página siguiente son datos con una estructura tridimensional.
2. Presente las ecuaciones de convolución para (a).
3. Presente las ecuaciones de convolución para (b).



Datos de entrada

Datos de entrada

Datos multicanal (eje. imagen a color RGB) (b) Datos 3D (eje. video, imágenes del cerebro MRI)

Mapa de funciones

Mapa de funciones

1. Aplicar la función softmax cuando la salida de la red neuronal es y escribir el resultado.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Forme un modelo de predicción de series temporales basado en RNN con referencia al siguiente código.

|  |
| --- |
| importar pandas como pd  importar numpy como np  desde sklearn.model\_selection importar train\_test\_split  desde sklearn.preprocessing importar MinMaxScaler  importar tensorflow como tf  importar matplotlib.pyplot como plt  # read data# Leer datos  df = pd.read\_csv('data\_boston.csv', header='infer', encoding='latin1')  df = df[['PRICE']]  # scale input & X, y  scaler = MinMaxScaler()  ts\_scaled = scaler.fit\_transform(df)  # scale  ts\_scaled\_2 = ts\_scaled.reshape(1, -1, 1)  # training parameters  batch\_size = 1  n\_epochs = 1000  learn\_rate = 0.0001  # model  model = tf.keras.Sequential()  model.add(tf.keras.Input(shape=(None, 1)))  # completar el modelo basado en la rnn y la capacitación  ***EL CODIGO COMPLETO LO MANDE EN***  ***PDF EN FORMATO JUPYTER NOTEBOOK*** |

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

import tensorflow as tf

import matplotlib.pyplot as plt

*#? Lectura de datos*

df = pd.read\_csv(r'./data\_boston.csv',

                header='infer',

                encoding='latin1')

df = df[['PRICE']]

*#? Escalar los datos*

scaler = MinMaxScaler()

scaleddata = scaler.fit\_transform(df)

*#? Crear datos de entrada para RNN*

def crear\_secuencias(data, lensec=5):

    """Creacion de datos de entrada

    data = datos a introducir

    lensec = secuencia de datos por defecto 5"""

    X, y = [], []

    for i in range (len(data) - lensec):

        X.append(data[i:i+lensec])

        y.append(data[i+lensec])

    return np.array(X), np.array(y)

lensecuencias = 5

X, y = crear\_secuencias(scaleddata)

*#? Dividir datos en entrenamiento y prueba*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

                                                    test\_size=0.2,

                                                    shuffle=False)

*#? Parámetros de entrenamiento*

batch\_size = 1

n\_epochs = 1\_000

learn\_rate = 0.0001

*#? Definicion del modelo RNN*

model = tf.keras.Sequential()

model.add(tf.keras.layers.SimpleRNN(50, activation='tanh', return\_sequences=False, input\_shape=(lensecuencias, 1)))

model.add(tf.keras.layers.Dense(1))

*#? Compilar modelo*

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=learn\_rate),

            loss='mse')

*#? Entrenar modelo*

history = model.fit(X\_train, y\_train,

                    epochs=n\_epochs,

                    batch\_size=batch\_size,

                    validation\_data=(X\_test, y\_test),

                    verbose=1)

*#? Graficar pérdida*

plt.plot(history.history['loss'], label='Pérdida entrenamiento')

plt.plot(history.history['val\_loss'], label='Pérdida validación')

plt.legend()

plt.show()

*# Predicción final para verificar funcionamiento*

predictions = model.predict(X\_test)

predictions = scaler.inverse\_transform(predictions)