COLOMBIA POTENCIA DE LA VIDA

TIC

Actividad 3

Ejercicio práctico implementar una red neuronal para clasificación utilizando Keras

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Ejercicio práctico implementar una red neuronal para clasificación utilizando Keras

Pasos a seguir:

- Importación de bibliotecas

Importamos las bibliotecas necesarias. keras es la API de alto nivel para construir y entrenar modelos de redes neuronales. make_classification es una función de scikit-learn para generar un conjunto de datos de clasificación sintético y train_test_split es una función para dividir los datos en conjuntos de entrenamiento y prueba.

```
import keras
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
```

- Generamos un conjunto de datos de clasificación sintético utilizando make_classification de scikit-learn.

Creamos un conjunto de datos de clasificación sintética con 100 muestras, 5 características y 2 clases.

Ejercicio práctico implementar una red neuronal para clasificación utilizando Keras

- Dividimos los datos en conjuntos de entrenamiento y prueba con train_test_split.

Dividimos los datos en conjuntos de entrenamiento y prueba. Aquí, el 20% de los datos se reservan para el conjunto de pruebas.

- Convertimos las etiquetas a codificación one-hot utilizando to_categorical de Keras.

Convertimos las etiquetas a codificación one-hot utilizando to_categorical de Keras. Esto convierte las etiquetas de clases en vectores binarios donde la posición de la clase activa es 1 y todas las demás son 0.

- Creamos un modelo secuencial de Keras con una capa oculta de 10 neuronas y activación ReLU, y una capa de salida con 2 neuronas (una para cada clase) y activación softmax.

Creamos una instancia de un modelo secuencial de Keras.
Añadimos una capa oculta con 10 neuronas y activación ReLU.
Añadimos una capa de salida con 2 neuronas (una para cada clase) y activación softmax.

Ejercicio práctico implementar una red neuronal para clasificación utilizando Keras

- Compilamos el modelo utilizando el optimizador 'adam' y la función de pérdida 'categorical_crossentropy'.

Compilamos el modelo utilizando el optimizador Adam, la función de pérdida de entropía cruzada categórica y métricas de precisión.

- Entrenamos el modelo durante 10 épocas en lotes de tamaño 32, utilizando los datos de entrenamiento y validación.

Entrenamos el modelo durante 10 épocas con un tamaño de lote de 32. Utilizamos los conjuntos de validación durante el entrenamiento para monitorear el rendimiento del modelo.

Ejercicio práctico implementar una red neuronal para clasificación utilizando Keras

- Evaluamos el modelo en el conjunto de prueba y mostramos la pérdida y la precisión obtenidas.

Evaluamos el modelo en el conjunto de prueba y obtenemos la pérdida y la precisión.

Este es un ejercicio básico de cómo implementar una red neuronal para clasificación utilizando Keras. Se pueden ajustar muchos otros parámetros y arquitecturas de red para adaptarse a las necesidades específicas del problema.

Solución del ejercicio

Generación de datos de clasificación sintéticos:

```
# Generar un conjunto de datos de clasificación sintético
X, y = make_classification(n_samples=100, n_features=5,
n_classes=2)
```

División de los datos:

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
```

Codificación one-hot de las etiquetas:

```
# Convertir las etiquetas a codificación one-hot
y_train_categorical = keras.utils.to_categorical(y_train)
y_test_categorical = keras.utils.to_categorical(y_test)
```

Solución del ejercicio

Creación del modelo:

```
# Crear una instancia de un modelo secuencial de Keras
model = keras.models.Sequential()
```

Añadir capas al modelo:

```
# Añadir una capa oculta con 10 neuronas y activación ReLU
model.add(keras.layers.Dense(10,
input_shape=(X_train.shape[1],), activation='relu'))
# Añadir una capa de salida con 2 neuronas (una para cada
clase) y activación softmax
model.add(keras.layers.Dense(2, activation='softmax'))
```

Solución del ejercicio

Compilación del modelo:

```
# Compilar el modelo
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
```

Entrenamiento del modelo:

```
# Entrenar el modelo
model.fit(X_train, y_train_categorical, epochs=10,
batch_size=32, validation_data=(X_test, y_test_categorical))
```

Evaluación del modelo:

```
# Evaluar el modelo en el conjunto de prueba
loss, accuracy = model.evaluate(X_test, y_test_categorical)
```

Solución del ejercicio

Impresión de resultados:

```
print("Pérdida en el conjunto de prueba:", loss)
print("Precisión en el conjunto de prueba:", accuracy)
```

Al ejecutar el código se espera más o menos lo siguiente, pero cada que se ejecute se van a obtener valores un poco diferentes.

```
Epoch 1/10
3/3 [==============================] - 1s 79ms/step - loss: 0.5500 - accuracy: 0.7500 - val_loss: 0.4971 - val_accuracy: 0.8500
Epoch 2/10
3/3 [==============================] - 0s 12ms/step - loss: 0.5365 - accuracy: 0.7625 - val_loss: 0.4826 - val_accuracy: 0.8500
Epoch 3/10
3/3 [==============================] - 0s 12ms/step - loss: 0.5239 - accuracy: 0.7625 - val_loss: 0.4706 - val_accuracy: 0.8500
Epoch 4/10
3/3 [==============================] - 0s 13ms/step - loss: 0.5113 - accuracy: 0.7750 - val_loss: 0.4577 - val_accuracy: 0.8500
Epoch 5/10
3/3 [==============================] - 0s 12ms/step - loss: 0.4999 - accuracy: 0.7750 - val_loss: 0.4465 - val_accuracy: 0.8500
Epoch 6/10
3/3 [==============================] - 0s 12ms/step - loss: 0.4888 - accuracy: 0.7875 - val_loss: 0.4364 - val_accuracy: 0.8500
Epoch 7/10
3/3 [==============================] - 0s 11ms/step - loss: 0.4779 - accuracy: 0.8000 - val_loss: 0.4270 - val_accuracy: 0.8500
Epoch 8/10
3/3 [==============================] - 0s 12ms/step - loss: 0.4679 - accuracy: 0.8000 - val_loss: 0.4182 - val_accuracy: 0.8500
Epoch 9/10
3/3 [==============================] - 0s 12ms/step - loss: 0.4581 - accuracy: 0.8000 - val_loss: 0.4101 - val_accuracy: 0.8500
Epoch 10/10
3/3 [==============================] - 0s 12ms/step - loss: 0.4486 - accuracy: 0.8125 - val_loss: 0.4029 - val_accuracy: 0.8500
1/1 [==============================] - 0s 18ms/step - loss: 0.4029 - accuracy: 0.8500
Pérdida en el conjunto de prueba: 0.40287724137306213
Precisión en el conjunto de prueba: 0.8500000238418579
```

Solución del ejercicio

En este punto se puede pedir a los estudiantes que realicen algún cambio en el código para ver qué efecto tiene.

Dentro de los cambio se puede plantear:

- Incrementa el número de épocas
- Cambia el tamaño del batch
- Cambia la funcion de activacion
- Añadir más neuronas a las capas existentes.
- Añadir más capas

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira