COLOMBIA POTENCIA DE LA VIDA

TIC

Actividad 4

Ejercicio práctico implementar una red neuronal para regresión utilizando Keras

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Pasos a seguir:

- **Importar bibliotecas:**

Importa las bibliotecas necesarias, incluyendo Keras para crear el modelo de red neuronal, scikit-learn para cargar el conjunto de datos, dividirlo en conjuntos de entrenamiento, validación y prueba, y estandarizar los datos, pandas para manipular datos tabulares y matplotlib.pyplot para trazar gráficos.

```
import keras
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import pandas as pd
import matplotlib.pyplot as plt
```

- **Cargar el conjunto de datos:**

Carga el conjunto de datos de viviendas de California utilizando la función fetch_california_housing() de scikit-learn.

- **Crear conjuntos de datos de entrenamiento, validación y prueba:**

Divide el conjunto de datos en conjuntos de entrenamiento, validación y prueba utilizando la función train_test_split() de scikit-learn.

- **Escalar los datos:**

Estandariza los datos para que tengan una media de cero y una desviación estándar de uno utilizando la clase StandardScaler() de scikit-learn.

- **Crear el modelo de red neuronal:**

Utiliza el modelo secuencial de Keras para construir una red neuronal con una capa densa de 30 neuronas con activación ReLU como función de activación, seguida de una capa de salida con una sola neurona (ya que estamos realizando una regresión).

- **Compilar el modelo:**

Compila el modelo especificando la función de pérdida (mean_squared_error para regresión), el optimizador (sgd para descenso de gradiente estocástico) y las métricas que se utilizarán para evaluar el rendimiento del modelo (root_mean_squared_error y mean_absolute_percentage_error).

- **Entrenar el modelo:**

Entrena el modelo utilizando los datos de entrenamiento durante 10 épocas, especificando también los datos de validación.

- **Graficar el historial de entrenamiento:**

Utiliza pandas para convertir el historial de entrenamiento del modelo en un DataFrame y traza las curvas de pérdida y métricas durante el entrenamiento y la validación.

- **Evaluar el modelo:**

Evalúa el rendimiento del modelo en los conjuntos de entrenamiento, validación y prueba utilizando el método evaluate().
Este código crea, entrena, evalúa y visualiza el rendimiento de una red neuronal para predecir los precios de las viviendas en California.

Solución del ejercicio

Cargar el conjunto de datos:

```
# Cargar el conjunto de datos
housing = fetch_california_housing()
```

Crear conjuntos de datos de entrenamiento, validación y prueba:

```
# Crear conjuntos de datos de entrenamiento, validación y
prueba
X_train_full, X_test, y_train_full, y_test =
train_test_split(housing.data, housing.target)
X_train, X_valid, y_train, y_valid =
train_test_split(X_train_full, y_train_full)

print('X_train:',X_train.shape)
print('X_test:',X_test.shape)
print('X_valid:',X_valid.shape)
print('y_train:',y_train.shape)
print('y_test:',y_test.shape)
print('y_valid:',y_valid.shape)
```

Escalar los datos:

```
# Escalar los datos
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_valid = scaler.transform(X_valid)
```

Crear el modelo de red neuronal:

```
# Crear el modelo de red neuronal:
model = keras.models.Sequential([
    keras.layers.Dense(30, activation='relu',
input_shape=X_train.shape[1:]),
    keras.layers.Dense(1)
    ])
```

Compilar el modelo:

```
# Compilar el modelo:
model.compile(loss='mean_squared_error',
              optimizer='sgd',
              metrics=[keras.metrics.RootMeanSquaredError(),
                       'mean_absolute_percentage_error'])
```

Entrenar el modelo:

```
# Entrenar el modelo
history = model.fit(X_train, y_train,
                    epochs=10,
                    validation_data=(X_valid, y_valid))
```

Graficar el historial de entrenamiento:

```
# Graficar el historial de entrenamiento:
dfHistory = pd.DataFrame(history.history)
dfHistory[['loss', 'val_loss', 'root_mean_squared_error',
'val_root_mean_squared_error']].plot(grid=True)
dfHistory[['mean_absolute_percentage_error',
'val_mean_absolute_percentage_error']].plot(grid=True)
```

Evaluar el modelo:

```
# Evaluar el modelo en el conjunto de entrenamiento
loss, RMSE, MAPE = model.evaluate(X_train, y_train)
print("Evaluar el modelo en el conjunto de entrenamiento")
print("loss train:", loss)
print("root_mean_squared_error train:", RMSE)
print("mean_absolute_percentage_error train:", MAPE)

# Evaluar el modelo en el conjunto de validacion
loss, RMSE, MAPE = model.evaluate(X_valid, y_valid)
print("Evaluar el modelo en el conjunto de validacion")
print("loss valid:", loss)
print("root_mean_squared_error valid:", RMSE)
print("mean_absolute_percentage_error valid:", MAPE)

# Evaluar el modelo en el conjunto de prueba
loss, RMSE, MAPE = model.evaluate(X_test, y_test)
print("Evaluar el modelo en el conjunto de prueba")
print("loss test:", loss)
print("root_mean_squared_error test:", RMSE)
print("mean_absolute_percentage_error test:", MAPE)
```

Preguntas de comprensión

01 ¿Qué conjunto de datos se utiliza en este código y qué problema de aprendizaje automático se aborda?

02 ¿Qué hace la función train_test_split() de scikit-learn y por qué se utiliza en este código?

03 ¿Por qué es importante estandarizar los datos antes de entrenar un modelo de redes neuronales?

04 ¿Qué arquitectura de red neuronal se utiliza en este código y cuántas capas tiene?

05 ¿Qué función de activación se utiliza en la capa oculta de la red neuronal y por qué se elige esa función?

06 ¿Qué función de pérdida se utiliza para compilar el modelo y qué métricas se utilizan para evaluar su rendimiento?

07 ¿Cuántas épocas se utilizan para entrenar el modelo y por qué se elige ese número?

Ejercicios de exploración

1. Modifica el número de neuronas en la capa oculta de la red neuronal y observa cómo afecta el rendimiento del modelo.
2. Experimenta con diferentes optimizadores, como "adam" o "rmsprop", y observa cómo afectan el rendimiento del modelo.
3. Agrega una capa de regularización, como Dropout, a la red neuronal y observa cómo afecta el rendimiento del modelo.
4. Cambia la función de pérdida a "mean_absolute_error" y observa cómo afecta el rendimiento del modelo.
5. Entrena el modelo con un número diferente de épocas y observa cómo afecta el rendimiento del modelo.

Ejercicios de exploración

6. Elimina la estandarización de los datos y observa cómo afecta el rendimiento del modelo.
7. Experimenta con diferentes arquitecturas de red neuronal, como agregar capas adicionales o cambiar el número de neuronas en cada capa, y observa cómo afectan el rendimiento del modelo.
8. Intenta resolver el problema utilizando otro conjunto de datos y compara los resultados con los obtenidos en el conjunto de datos de viviendas de California.

Estas preguntas y ejercicios ayudarán a profundizar tu comprensión del código y a explorar diferentes aspectos del entrenamiento de redes neuronales.

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira