LSTM_FEIGCUBICA

June 24, 2024

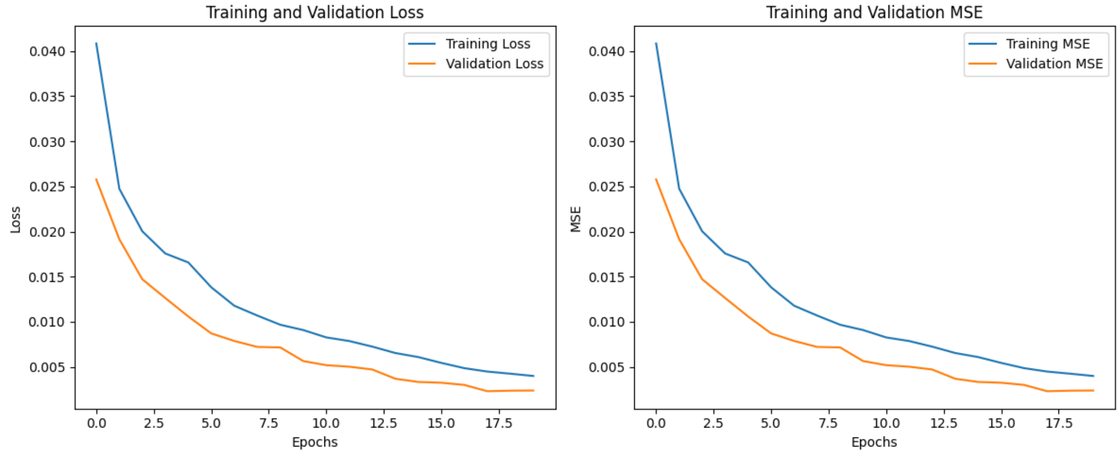
1 PREDICCIÓN Valores X Feigenbaum Exponencial

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.metrics import MeanSquaredError

def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        v = X.iloc[i:(i + time_steps)].values
        Xs.append(v)
        ys.append(y.iloc[i + time_steps])
    return np.array(Xs), np.array(ys)

data = pd.read_csv('datosFeigenbaumExponencial.csv')

scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)

time_steps = 3
X, y = create_dataset(pd.DataFrame(data_scaled), pd.DataFrame(data_scaled),␣
↪time_steps)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=0)

model = Sequential([
    LSTM(100, activation='relu', input_shape=(X_train.shape[1], X_train.
↪shape[2]), return_sequences=True),
    Dropout(0.2),
    LSTM(50, activation='relu'),
    Dropout(0.2),
```

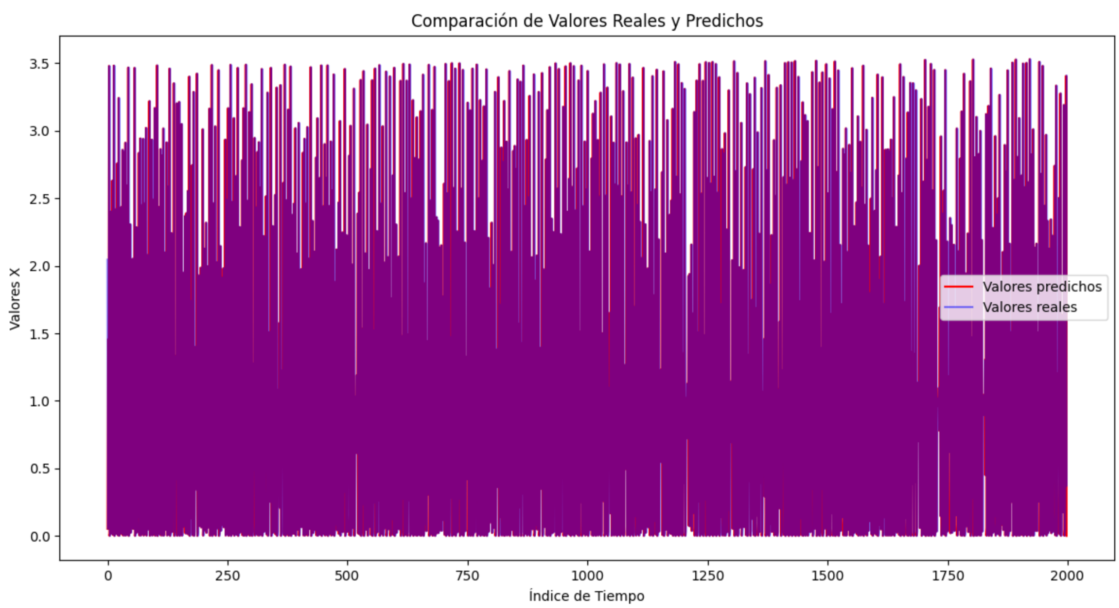
```
    Dense(2)
])

model.compile(optimizer='adam', loss='mean_squared_error',␣
 ↪metrics=[MeanSquaredError()])

history = model.fit(X_train, y_train, epochs=20, batch_size=32,␣
 ↪validation_split=0.2)

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['mean_squared_error'], label='Training MSE')
plt.plot(history.history['val_mean_squared_error'], label='Validation MSE')
plt.title('Training and Validation MSE')
plt.xlabel('Epochs')
plt.ylabel('MSE')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Epoch 1/20
460/460 [==============================] - 9s 10ms/step - loss: 0.0409 -
mean_squared_error: 0.0409 - val_loss: 0.0258 - val_mean_squared_error: 0.0258
Epoch 2/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0248 -
mean_squared_error: 0.0248 - val_loss: 0.0191 - val_mean_squared_error: 0.0191
Epoch 3/20
460/460 [==============================] - 6s 12ms/step - loss: 0.0200 -
mean_squared_error: 0.0200 - val_loss: 0.0147 - val_mean_squared_error: 0.0147
Epoch 4/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0176 -
mean_squared_error: 0.0176 - val_loss: 0.0126 - val_mean_squared_error: 0.0126
Epoch 5/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0166 -
mean_squared_error: 0.0166 - val_loss: 0.0106 - val_mean_squared_error: 0.0106
Epoch 6/20
460/460 [==============================] - 6s 12ms/step - loss: 0.0138 -
mean_squared_error: 0.0138 - val_loss: 0.0087 - val_mean_squared_error: 0.0087
```

```
Epoch 7/20
460/460 [==============================] - 4s 10ms/step - loss: 0.0118 -
mean_squared_error: 0.0118 - val_loss: 0.0079 - val_mean_squared_error: 0.0079
Epoch 8/20
460/460 [==============================] - 5s 11ms/step - loss: 0.0107 -
mean_squared_error: 0.0107 - val_loss: 0.0072 - val_mean_squared_error: 0.0072
Epoch 9/20
460/460 [==============================] - 6s 13ms/step - loss: 0.0097 -
mean_squared_error: 0.0097 - val_loss: 0.0071 - val_mean_squared_error: 0.0071
Epoch 10/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0091 -
mean_squared_error: 0.0091 - val_loss: 0.0056 - val_mean_squared_error: 0.0056
Epoch 11/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0083 -
mean_squared_error: 0.0083 - val_loss: 0.0052 - val_mean_squared_error: 0.0052
Epoch 12/20
460/460 [==============================] - 5s 11ms/step - loss: 0.0079 -
mean_squared_error: 0.0079 - val_loss: 0.0050 - val_mean_squared_error: 0.0050
Epoch 13/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0072 -
mean_squared_error: 0.0072 - val_loss: 0.0047 - val_mean_squared_error: 0.0047
Epoch 14/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0065 -
mean_squared_error: 0.0065 - val_loss: 0.0037 - val_mean_squared_error: 0.0037
Epoch 15/20
460/460 [==============================] - 6s 12ms/step - loss: 0.0061 -
mean_squared_error: 0.0061 - val_loss: 0.0033 - val_mean_squared_error: 0.0033
Epoch 16/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0054 -
mean_squared_error: 0.0054 - val_loss: 0.0032 - val_mean_squared_error: 0.0032
Epoch 17/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0048 -
mean_squared_error: 0.0048 - val_loss: 0.0030 - val_mean_squared_error: 0.0030
Epoch 18/20
460/460 [==============================] - 6s 12ms/step - loss: 0.0045 -
mean_squared_error: 0.0045 - val_loss: 0.0023 - val_mean_squared_error: 0.0023
Epoch 19/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0042 -
mean_squared_error: 0.0042 - val_loss: 0.0024 - val_mean_squared_error: 0.0024
Epoch 20/20
460/460 [==============================] - 5s 10ms/step - loss: 0.0040 -
mean_squared_error: 0.0040 - val_loss: 0.0024 - val_mean_squared_error: 0.0024
```



[ ]:
```
plt.figure(figsize=(14, 7))
plt.plot(predicted_values, label='Valores predichos', color='red')
plt.plot(actual_values, label='Valores reales', color='blue', alpha=0.5)
plt.title('Comparación de Valores Reales y Predichos')
plt.xlabel('Índice de Tiempo')
plt.ylabel('Valores X')
plt.legend()
plt.show()
```



2 ANEXOS

Los códigos utilizados se pueden encontrar en cualquiera de estos enlaces:

Google Drive: https://drive.google.com/drive/folders/1q-YwDGhG-700djaiBptVDcwc_OYP1dE0?usp=sharing

GitHub: https://github.com/RodrigoG13/Time-Series.git