Traffic/Sales Forecasting with LSTM

Diana Akolzina

May 26, 2023

1 Introduction

This Python code is made to forecast traffic by leveraging the Long Short-Term Memory (LSTM) Neural Network, a form of recurrent neural network known for its ability to remember long-term dependencies in sequence datawhich is suitable for stationary and seasonal data.

2 Prerequisites

- pandas
- numpy
- sklearn
- keras
- matplotlib

Could be installed using pip

```
pip install pandas numpy sklearn keras matplotlib
```

3 Data

The program expects a CSV file with the following columns:

- DateTime: Date and time of the data points
- Predicted feature: sales, traffic or similar

The DateTime should be in a format recognized by pandas (yyyy-mm-dd hh:mm:ss) and will be resampled to a daily frequency. The ideal format is ZIP for the datetime parsing.

4 Code Explanation

The code first loads and preprocesses the data:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Load the data
data = pd.read_csv('traffic.csv')
data['DateTime'] = pd.to_datetime(data['DateTime'])
data = data.set_index('DateTime')

# Resample data to daily frequency, aggregating by sum
data = data.resample('D').sum()

# Scale the data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)
```

Then the LSTM model is defined, compiled, and fitted:

```
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Create the LSTM model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=True))
model.add(LSTM(25, return_sequences=False))
model.add(Dense(1))

# Compile and fit the model
model.compile(optimizer='adam', loss=loss)
history = model.fit(X_train, y_train, batch_size=batch_size, epochs=epoch,
    validation_data=(X_test, y_test))
```

Lastly, the program predicts the next 31 days of traffic and visualizes the results, the forecast period can be manually changed, given there is enough data:

```
import matplotlib.pyplot as plt

# Predict the next 31 days of traffic
forecast_period = 31
input_data = scaled_data[-step:]  # get the last 'step' days of the data
input_data = input_data.reshape((1, step, 1))  # reshape to fit the model input shape

forecast = []
for _ in range(forecast_period):
    prediction = model.predict(input_data)
    forecast.append(prediction[0, 0])

    # Use the prediction to update the input_data, discarding the oldest value
    input_data = np.roll(input_data, -1)
    input_data[0, -1, 0] = prediction

# Inverse transform the forecast data
forecast = scaler.inverse_transform(np.array(forecast).reshape(-1, 1))

# Plot the forecast
plt.figure(figsize=(14, 6))
plt.plot(date, forecast, color='r')
plt.title('Forecasted traffic for the next 31 days')
plt.xlabel('Day')
plt.ylabel('Traffic')
plt.legend(['Forecast'], loc='upper right')
plt.show()
```

5 Conclusion

The data is meticulously prepared and fine-tuned to enhance the precision of the model's predictions, specifically considering the time-dependent characteristics of the data. The time series information is processed through a variety of LSTM-based models, each having distinct parameters. Parameters like batch size, loss function, and the number of epochs can be tweaked to optimize the model's performance. Alongside the model's evaluation and forecasting, plots of the loss function's behavior over successive epochs are provided to visualize the model's learning process.