Prática7_respostas

August 27, 2020

1 Prática 7

Aprendizado Dinâmico

por Cibele Russo (ICMC/USP - São Carlos SP)

MBA em Ciências de Dados

Nesta prática vamos considerar redes dinâmicas para modelar a temperatura global dos dados em globaltemp.

1.Faça a leitura das bibliotecas.

[1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
```

2. Leia os dados do arquivo globaltemp.csv.

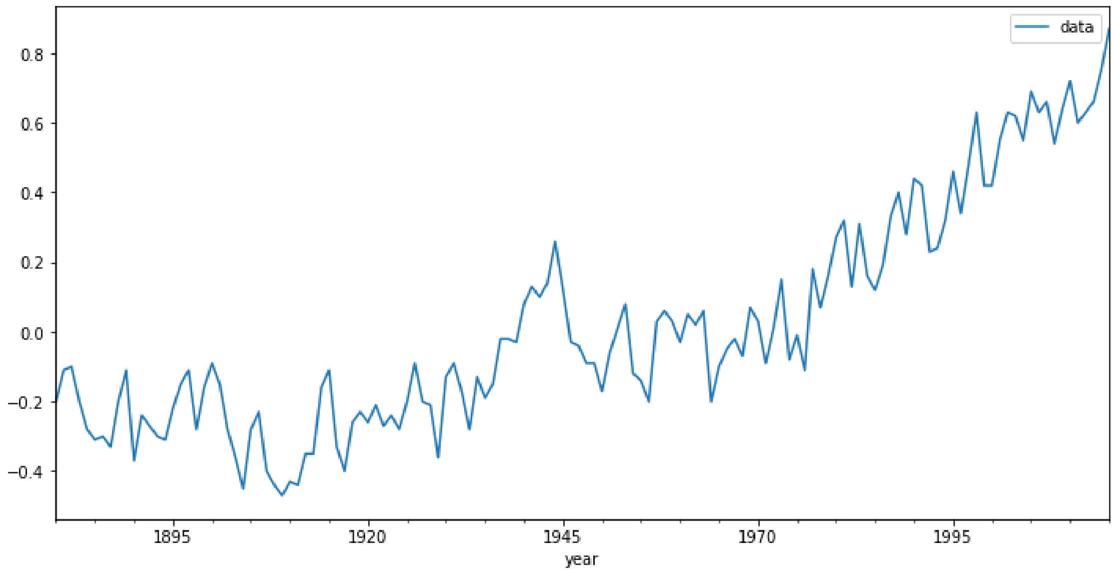
[2]:
```
# Temperatura global

# Diferenças na média de temperatura global.
# Fonte: https://github.com/mjuez/pytsdatasets/

pkgdir = '/home/cibele/CibelePython/AprendizadoDinamico/Data'


df = pd.read_csv(f'{pkgdir}/globaltemp.csv', index_col=0,
                              parse_dates=True)
df.index = df.index
df.plot(figsize=(12,6))
```

[2]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fedb08a0690>
```



[3]:
```
df.head()
```

[3]:
```
             data
year
1880-01-01  -0.20
1881-01-01  -0.11
1882-01-01  -0.10
1883-01-01  -0.20
1884-01-01  -0.28
```

1.1

3. Divida a base em treino e teste, deixando 14 dias para a previsão.

[4]:
```
len(df)
```

[4]: 136

[5]:
```
len(df)-14
```

[5]: 122

[6]:
```
train = df.iloc[:122]
test = df.iloc[122:]
```

[7]:
```
train
```

[7]:
```
            data
year
1880-01-01 -0.20
1881-01-01 -0.11
1882-01-01 -0.10
1883-01-01 -0.20
1884-01-01 -0.28
...          ...
1997-01-01  0.48
1998-01-01  0.63
1999-01-01  0.42
2000-01-01  0.42
2001-01-01  0.55

[122 rows x 1 columns]
```

4. Padronize os dados para a modelagem.

[8]:
```python
from sklearn.preprocessing import MinMaxScaler
```

[9]:
```python
scaler = MinMaxScaler()
```

[10]:
```python
scaler.fit(train)
```

[10]:
```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

[11]:
```python
scaled_train = scaler.transform(train)
scaled_test = scaler.transform(test)
```

5. Considere o gerador de séries temporais, com variados valores para os parâmetros length e batch_size.

[15]:
```python
from keras.preprocessing.sequence import TimeseriesGenerator
```

[16]:
```python
# defina o gerador
n_input = 2
n_features = 1
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input,
 ↪batch_size=1)
```

[17]:
```python
len(scaled_train)
```

[17]:
```
122
```

[18]:
```python
len(generator) # n_input = 2
```

[18]:
```
120
```

```
[19]: # Qual é a aparência do primeiro lote?
X,y = generator[0]
```

```
[20]: print(f'Dado o array: \n{X.flatten()}');
print(f'Previsão: \n {y}');
```

```
Dado o array:
[0.24545455 0.32727273]
Previsão:
 [[0.33636364]]
```

6. Carregue as bibliotecas do keras para as redes dinâmicas.

```
[21]: from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

7. Defina os lotes pra o processo iterativo.

```
[22]: # Vamos redefinir lotes de tamanho 21 para o procedimento iterativo
# Veja mais informações sobre o tamanho do lote http://deeplearningbook.com.br/
 ↪o-efeito-do-batch-size-no-treinamento-de-redes-neurais-artificiais/

n_input = 7
n_features = 1
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input,␣
 ↪batch_size=1)
```

8. Defina o modelo. Ele pode ter uma camada LSTM e uma camada Dense. Teste alternativas.

```
[23]: # Defina o modelo
model = Sequential()
model.add(LSTM(100, activation='relu', input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

```
[24]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100)               40800
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 40,901
Trainable params: 40,901
```

```
Non-trainable params: 0
_________________________________________________________________
```

9. Faça o ajuste do modelo e observe a função de perda.

[25]:
```python
# Ajuste do modelo

model.fit_generator(generator,epochs=100)
```

```
WARNING:tensorflow:From <ipython-input-25-5e7daf52724a>:3: Model.fit_generator
(from tensorflow.python.keras.engine.training) is deprecated and will be removed
in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/100
115/115 [==============================] - 1s 6ms/step - loss: 0.0438
Epoch 2/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0132
Epoch 3/100
115/115 [==============================] - 0s 4ms/step - loss: 0.0125
Epoch 4/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0132
Epoch 5/100
115/115 [==============================] - 0s 4ms/step - loss: 0.0123
Epoch 6/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0115
Epoch 7/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0112
Epoch 8/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0130
Epoch 9/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0117
Epoch 10/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0112
Epoch 11/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0115
Epoch 12/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0100
Epoch 13/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0114
Epoch 14/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0106
Epoch 15/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0100
Epoch 16/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0107
Epoch 17/100
115/115 [==============================] - 1s 5ms/step - loss: 0.0101
```

```
Epoch 18/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0102
Epoch 19/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0102
Epoch 20/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0101
Epoch 21/100
115/115 [==============================] - 0s 4ms/step - loss: 0.0102
Epoch 22/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0102
Epoch 23/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0104
Epoch 24/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0099
Epoch 25/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0098
Epoch 26/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0097
Epoch 27/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0102
Epoch 28/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0092
Epoch 29/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0104
Epoch 30/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0096
Epoch 31/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0098
Epoch 32/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0091
Epoch 33/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0086
Epoch 34/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0089
Epoch 35/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0094
Epoch 36/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0091
Epoch 37/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0091
Epoch 38/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0089
Epoch 39/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0090
Epoch 40/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0090
Epoch 41/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0087
```

```
Epoch 42/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0086
Epoch 43/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0093
Epoch 44/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0089
Epoch 45/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0094
Epoch 46/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0091
Epoch 47/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0088
Epoch 48/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0087
Epoch 49/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0090
Epoch 50/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0092
Epoch 51/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0101
Epoch 52/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0086
Epoch 53/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0087
Epoch 54/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0086
Epoch 55/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
Epoch 56/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0091
Epoch 57/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0088
Epoch 58/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0089
Epoch 59/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0088
Epoch 60/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0091
Epoch 61/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
Epoch 62/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 63/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0086
Epoch 64/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0083
Epoch 65/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
```

```
Epoch 66/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0084
Epoch 67/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
Epoch 68/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0087
Epoch 69/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0086
Epoch 70/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0083
Epoch 71/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0083
Epoch 72/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0088
Epoch 73/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0091
Epoch 74/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0090
Epoch 75/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0086
Epoch 76/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0090
Epoch 77/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0081
Epoch 78/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
Epoch 79/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
Epoch 80/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0086
Epoch 81/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0081
Epoch 82/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
Epoch 83/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0088
Epoch 84/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0088
Epoch 85/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
Epoch 86/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 87/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0085
Epoch 88/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0084
Epoch 89/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0078
```

```
Epoch 90/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0087
Epoch 91/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0080
Epoch 92/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 93/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 94/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 95/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 96/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 97/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0080
Epoch 98/100
115/115 [==============================] - 0s 4ms/step - loss: 0.0082
Epoch 99/100
115/115 [==============================] - 0s 4ms/step - loss: 0.0080
Epoch 100/100
115/115 [==============================] - 0s 3ms/step - loss: 0.0079
```

[25]:

```
<tensorflow.python.keras.callbacks.History at 0x7fedadcfad90>
```

[26]:

```python
model.history.history.keys()
```

[26]:

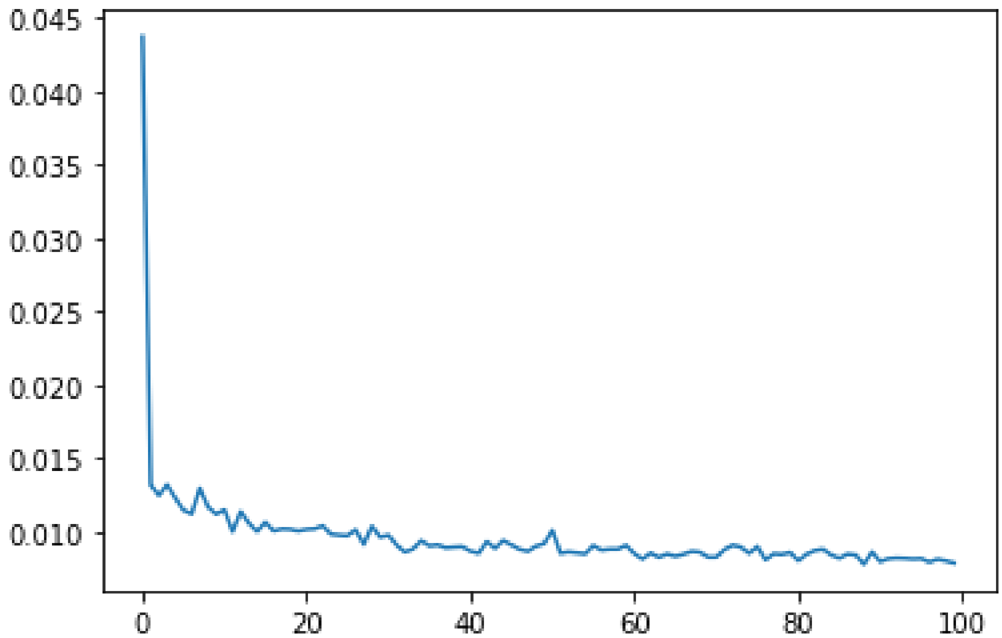
```
dict_keys(['loss'])
```

[27]:

```python
loss_per_epoch = model.history.history['loss']
plt.plot(range(len(loss_per_epoch)),loss_per_epoch)
```

[27]:

```
[<matplotlib.lines.Line2D at 0x7fed60436a50>]
```



10. Faça a previsão.

[28]:
```
# Vejamos passo a passo como é feita a previsão, a princípio para a próxima␣
↪observação usando o tamanho do lote igual a 7

first_eval_batch = scaled_train[-7:]
```

[29]:
```
first_eval_batch
```

[29]:
```
array([[0.84545455],
       [0.73636364],
       [0.86363636],
       [1.        ],
       [0.80909091],
       [0.80909091],
       [0.92727273]])
```

[30]:
```
# Agora vamos considerar as previsões para as próximas 21 observações e comparar␣
↪com a base de teste

test_predictions = []

first_eval_batch = scaled_train[-n_input:]
current_batch = first_eval_batch.reshape((1, n_input, n_features))

for i in range(len(test)):
```

```
    # obter a previsão de tempo 1 antecipadamente ([0] é para pegar apenas o␣
↪número em vez de [array])
    current_pred = model.predict(current_batch)[0]

    # predição
    test_predictions.append(current_pred)

    # atualize a rodada para agora incluir a previsão e descartar o primeiro␣
↪valor
    current_batch = np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
```

[31]:
```
test_predictions
```

[31]:
```
[array([0.8536765], dtype=float32),
 array([0.76443297], dtype=float32),
 array([0.8243903], dtype=float32),
 array([0.945121], dtype=float32),
 array([0.8264261], dtype=float32),
 array([0.8049556], dtype=float32),
 array([0.88539714], dtype=float32),
 array([0.84559166], dtype=float32),
 array([0.7754138], dtype=float32),
 array([0.7949036], dtype=float32),
 array([0.8879841], dtype=float32),
 array([0.81870073], dtype=float32),
 array([0.79119396], dtype=float32),
 array([0.8409509], dtype=float32)]
```

[32]:
```
scaled_test
```

[32]:
```
array([[1.        ],
       [0.99090909],
       [0.92727273],
       [1.05454545],
       [1.        ],
       [1.02727273],
       [0.91818182],
       [1.00909091],
       [1.08181818],
       [0.97272727],
       [1.        ],
       [1.02727273],
       [1.10909091],
       [1.21818182]])
```

11. Retorne da padronização.

[33]:
```
true_predictions = scaler.inverse_transform(test_predictions)
```

[34]:
```
true_predictions
```

[34]:
```
array([[0.46904415],
       [0.37087626],
       [0.43682932],
       [0.56963309],
       [0.4390687 ],
       [0.41545116],
       [0.50393685],
       [0.46015083],
       [0.38295519],
       [0.40439393],
       [0.50678251],
       [0.4305708 ],
       [0.40031336],
       [0.455046  ]])
```

[35]:
```
# Possivelmente encontraremos warnings aqui
test['Predictions'] = true_predictions
```

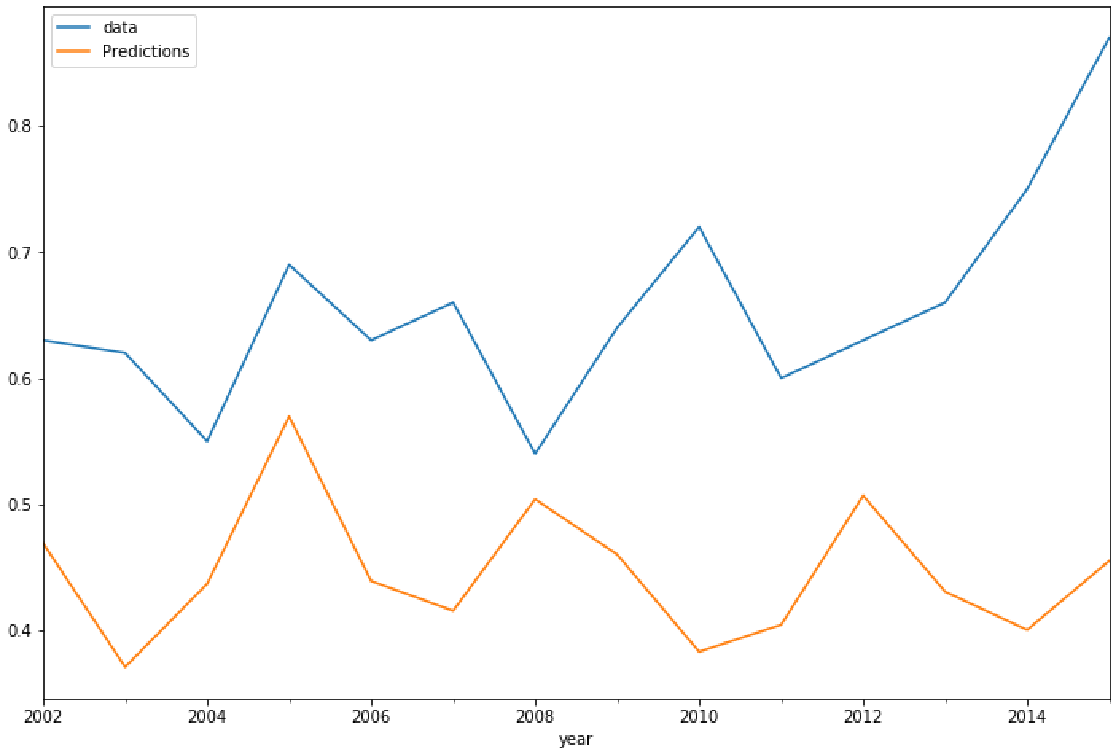
```
/home/cibele/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

12. Visualize os resultados, comparando as previsões com a base de teste.

[36]:
```
test.plot(figsize=(12,8))
```

[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed60282c10>
```


data
Predictions
0.8
0.7
0.6
0.5
0.4
2002
2004
2006
2008
2010
2012
2014
year