Solving sequence problems using LSTM with PyTorch

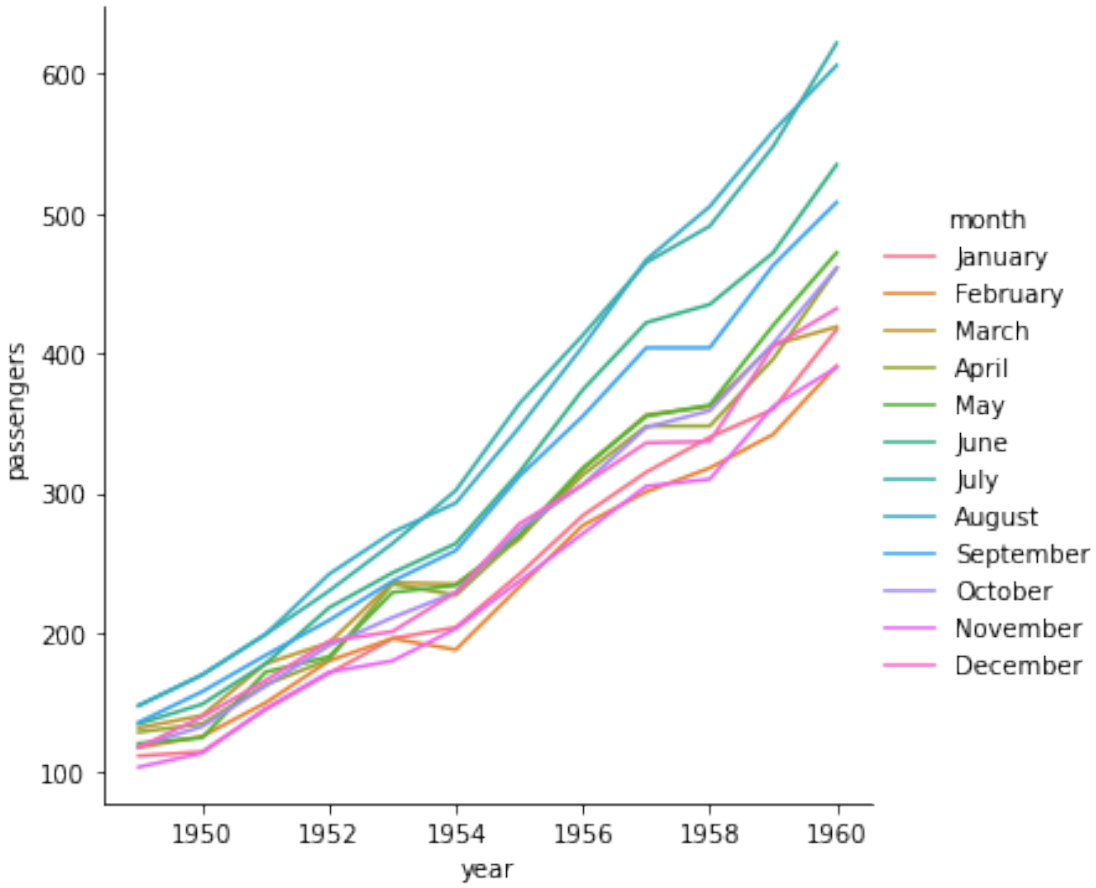
August 11, 2021

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

url = 'https://raw.githubusercontent.com/tvelichkovt/PyTorch/master/flights.csv'
df = pd.read_csv(url)

print(df.head())
```

```
   year     month  passengers
0  1949   January         112
1  1949  February         118
2  1949     March         132
3  1949     April         129
4  1949       May         121
```

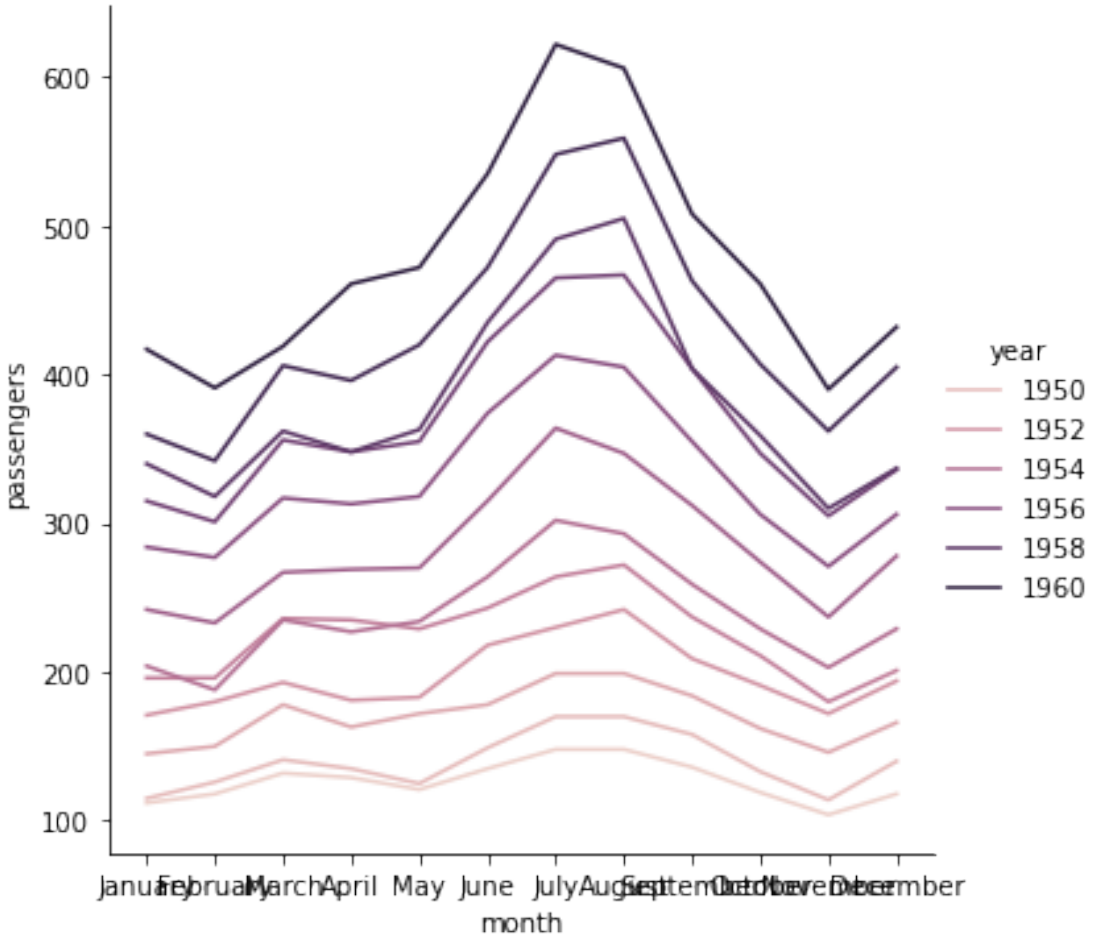
[2]:
```
sns.relplot(data=df, x="year", y="passengers", hue="month", kind="line")
```

[2]:
```
<seaborn.axisgrid.FacetGrid at 0x25688342dc8>
```



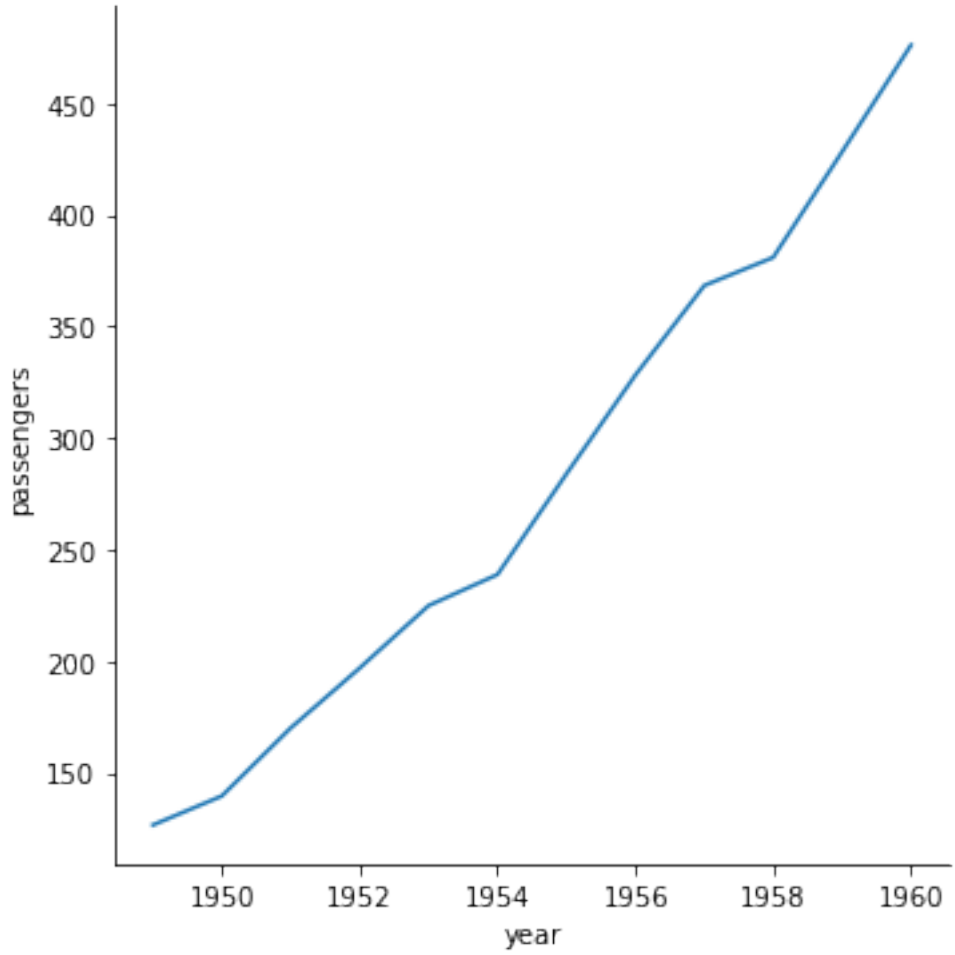
[3]:
```
sns.relplot(data=df, x="month", y="passengers", hue="year", kind="line")
```

[3]: <seaborn.axisgrid.FacetGrid at 0x2568827c408>



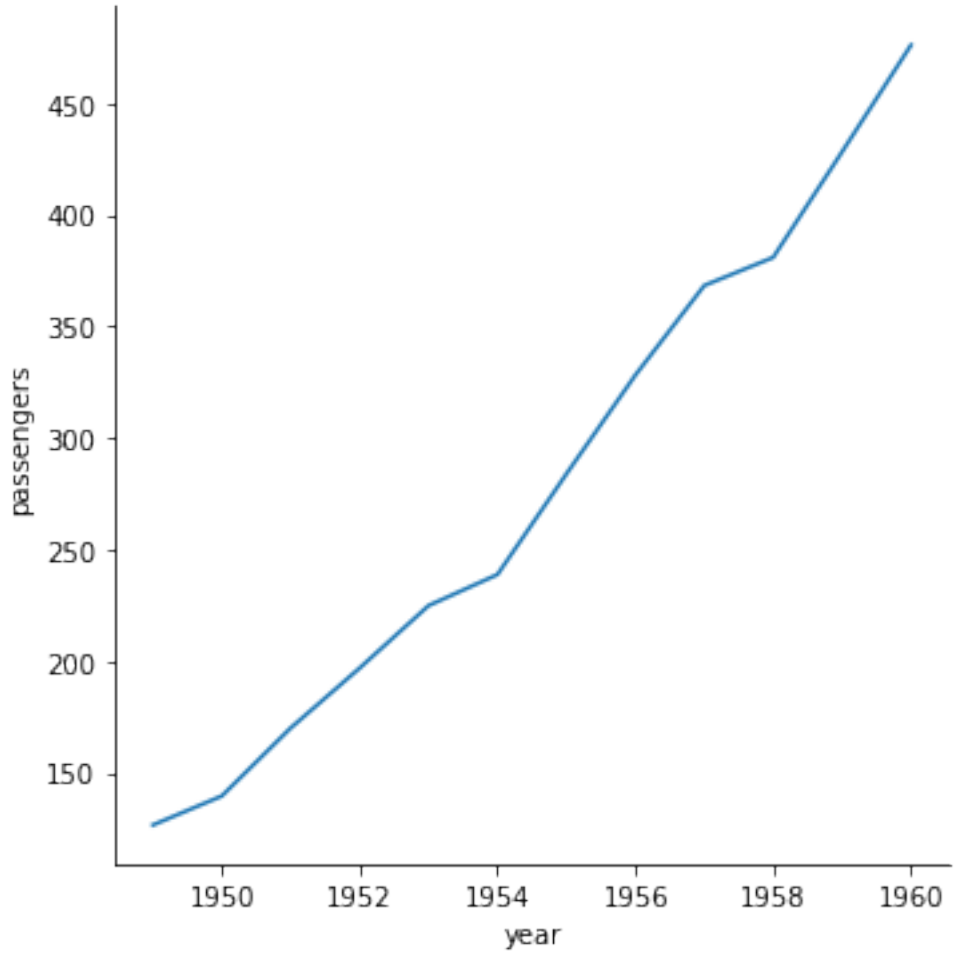
[4]:
```
flights_avg = df.groupby("year").mean()
sns.relplot(data=flights_avg, x="year", y="passengers", kind="line")
```

[4]: <seaborn.axisgrid.FacetGrid at 0x2568bc8f2c8>



[5]:
```
year = flights_avg.index
passengers = flights_avg["passengers"]
sns.relplot(x=year, y=passengers, kind="line")
```

[5]: <seaborn.axisgrid.FacetGrid at 0x2568bd27608>



[6]:
```
import torch
import torch.nn as nn

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

[7]:
```
flight_data = df
flight_data.head()
```

[7]:

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129

```
4  1949       May          121
```

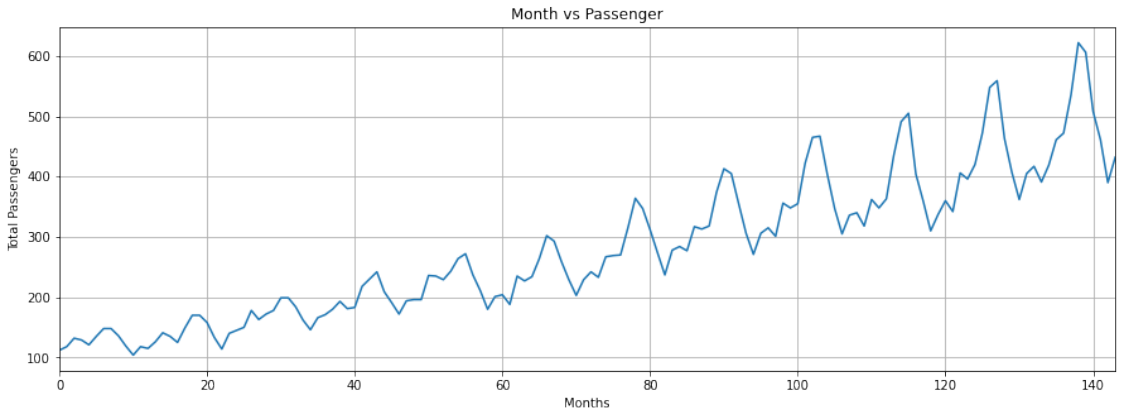
[8]:
```
flight_data.shape
```

[8]:
```
(144, 3)
```

[9]:
```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 15
fig_size[1] = 5
plt.rcParams["figure.figsize"] = fig_size
```

[10]:
```
plt.title('Month vs Passenger')
plt.ylabel('Total Passengers')
plt.xlabel('Months')
plt.grid(True)
plt.autoscale(axis='x',tight=True)
plt.plot(flight_data['passengers'])
```

[10]:
```
[<matplotlib.lines.Line2D at 0x2568f632f88>]
```



[11]:
```
flight_data.columns
```

[11]:
```
Index(['year', 'month', 'passengers'], dtype='object')
```

[12]:
```
all_data = flight_data['passengers'].values.astype(float)
```

[14]:
```
test_data_size = 12

train_data = all_data[:-test_data_size]
test_data = all_data[-test_data_size:]

print(len(train_data))
```

```
print(len(test_data))
```

```
132
12
```

[15]:
```
print(test_data)
```

```
[417. 391. 419. 461. 472. 535. 622. 606. 508. 461. 390. 432.]
```

[17]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(-1, 1))
train_data_normalized = scaler.fit_transform(train_data .reshape(-1, 1))

print(train_data_normalized[:5])
print(train_data_normalized[-5:])
```

```
[[-0.96483516]
 [-0.93846154]
 [-0.87692308]
 [-0.89010989]
 [-0.92527473]]
[[1.        ]
 [0.57802198]
 [0.33186813]
 [0.13406593]
 [0.32307692]]
```

[18]:
```
train_data_normalized = torch.FloatTensor(train_data_normalized).view(-1)
```

[19]:
```
def create_inout_sequences(input_data, tw):
    inout_seq = []
    L = len(input_data)
    for i in range(L-tw):
        train_seq = input_data[i:i+tw]
        train_label = input_data[i+tw:i+tw+1]
        inout_seq.append((train_seq ,train_label))
    return inout_seq
```

[21]:
```
train_window = 12
```

[22]:
```
train_inout_seq = create_inout_sequences(train_data_normalized, train_window)
```

[23]:
```
train_inout_seq[:5]
```

[23]:
```
[(tensor([-0.9648, -0.9385, -0.8769, -0.8901, -0.9253, -0.8637, -0.8066,
-0.8066,
```

```
          -0.8593, -0.9341, -1.0000, -0.9385]),
  tensor([-0.9516])),
 (tensor([-0.9385, -0.8769, -0.8901, -0.9253, -0.8637, -0.8066, -0.8066,
-0.8593,
          -0.9341, -1.0000, -0.9385, -0.9516]),
  tensor([-0.9033])),
 (tensor([-0.8769, -0.8901, -0.9253, -0.8637, -0.8066, -0.8066, -0.8593,
-0.9341,
          -1.0000, -0.9385, -0.9516, -0.9033]),
  tensor([-0.8374])),
 (tensor([-0.8901, -0.9253, -0.8637, -0.8066, -0.8066, -0.8593, -0.9341,
-1.0000,
          -0.9385, -0.9516, -0.9033, -0.8374]),
  tensor([-0.8637])),
 (tensor([-0.9253, -0.8637, -0.8066, -0.8066, -0.8593, -0.9341, -1.0000,
-0.9385,
          -0.9516, -0.9033, -0.8374, -0.8637]),
  tensor([-0.9077]))]
```

[24]:
```python
# Creating LSTM Model

class LSTM(nn.Module):
    def __init__(self, input_size=1, hidden_layer_size=100, output_size=1):
        super().__init__()
        self.hidden_layer_size = hidden_layer_size

        self.lstm = nn.LSTM(input_size, hidden_layer_size)

        self.linear = nn.Linear(hidden_layer_size, output_size)

        self.hidden_cell = (torch.zeros(1,1,self.hidden_layer_size),
                            torch.zeros(1,1,self.hidden_layer_size))

    def forward(self, input_seq):
        lstm_out, self.hidden_cell = self.lstm(input_seq.view(len(input_seq)␣
↪,1, -1), self.hidden_cell)
        predictions = self.linear(lstm_out.view(len(input_seq), -1))
        return predictions[-1]
```

[25]:
```python
model = LSTM()
loss_function = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

[26]:
```python
print(model)
```

```
LSTM(
  (lstm): LSTM(1, 100)
```

```
  (linear): Linear(in_features=100, out_features=1, bias=True)
)
```

[27]:
```
epochs = 150

for i in range(epochs):
    for seq, labels in train_inout_seq:
        optimizer.zero_grad()
        model.hidden_cell = (torch.zeros(1, 1, model.hidden_layer_size),
                             torch.zeros(1, 1, model.hidden_layer_size))

        y_pred = model(seq)

        single_loss = loss_function(y_pred, labels)
        single_loss.backward()
        optimizer.step()

    if i%25 == 1:
        print(f'epoch: {i:3} loss: {single_loss.item():10.8f}')

print(f'epoch: {i:3} loss: {single_loss.item():10.10f}')
```

```
epoch:   1 loss: 0.00147865
epoch:  26 loss: 0.00611455
epoch:  51 loss: 0.00024260
epoch:  76 loss: 0.00021025
epoch: 101 loss: 0.00006307
epoch: 126 loss: 0.00332760
epoch: 149 loss: 0.0005737843
```

[28]:
```
# Making Predictions

fut_pred = 12

test_inputs = train_data_normalized[-train_window:].tolist()
print(test_inputs)
```

```
[0.12527473270893097, 0.04615384712815285, 0.3274725377559662,
0.2835164964199066, 0.3890109956264496, 0.6175824403762817, 0.9516483545303345,
1.0, 0.5780220031738281, 0.33186814188957214, 0.13406594097614288,
0.32307693362236023]
```

[29]:
```
model.eval()

for i in range(fut_pred):
    seq = torch.FloatTensor(test_inputs[-train_window:])
    with torch.no_grad():
        model.hidden = (torch.zeros(1, 1, model.hidden_layer_size),
```

```
                        torch.zeros(1, 1, model.hidden_layer_size))
        test_inputs.append(model(seq).item())
```

[30]:
```
test_inputs[fut_pred:]
```

[30]:
```
[0.6159785389900208,
 0.6011861562728882,
 1.014121174812317,
 1.6501661539077759,
 2.01296329498291,
 2.175931930541992,
 2.275872230529785,
 2.338855743408203,
 2.3798298835754395,
 2.4063427448272705,
 2.423597812652588,
 2.4363346099853516]
```

[31]:
```
actual_predictions = scaler.inverse_transform(np.array(test_inputs[train_window:
 ↪] ).reshape(-1, 1))
print(actual_predictions)
```

```
[[471.63511762]
 [468.26985055]
 [562.21256727]
 [706.91280001]
 [789.44914961]
 [826.5245142 ]
 [849.26093245]
 [863.58968163]
 [872.91129851]
 [878.94297445]
 [882.86850238]
 [885.76612377]]
```

[32]:
```
plt.title('Month vs Passenger')
plt.ylabel('Total Passengers')
plt.grid(True)
plt.autoscale(axis='x', tight=True)

plt.plot(flight_data['passengers'][-train_window:])
plt.plot(x,actual_predictions)
plt.show()x = np.arange(132, 144, 1)
print(x)
```

```
[132 133 134 135 136 137 138 139 140 141 142 143]
```

[33]:
```
plt.title('Month vs Passenger')
plt.ylabel('Total Passengers')
plt.grid(True)
plt.autoscale(axis='x', tight=True)
plt.plot(flight_data['passengers'])
plt.plot(x,actual_predictions)
plt.show()
```