# 案例：224 航班流量预测

## 一、实验目的

（1）掌握LSTM循环神经网络的基本原理；

（2）掌握实现航班流量预测的基本思想；

（3）掌握图表可视化的基本方法；

（4）熟悉在jupyter notebook上代码的运行；

（5）通过上机编写、调试程序，培养数据处理算法的设计能力。

## **实验内容**

随着我国民航事业的迅猛发展，空中交通流量骤然增加，航班流量骤然增加，航班流量是民航空中管制得重要数据，是评定民航空管发展水平得关键指标。对非常态得高峰航班流量进行预测，不仅有助于提高繁忙机场得应急管理能力，而且还能为繁忙机场制定应急预案提供科学依据。

该实验中我们用深度学习模型对航班10年的月流量进行建模，并对客流量进行预测，帮助我们进行旅游出行的规划。

## 实验原理

**1.LSTM**

长短期记忆网络是循环神经网络的一个变体，可以有效地解决简单循环神经网络的梯度爆炸或消失问题。LSTM网络主要改进在以下两个方面

1. 新的内部状态：LSTM网络引入一个新的内部状态 专门进行线性的循环信息传递，同时（非线性地）输出信息给隐藏层的外部状态。内部状态通过下面公式计算：

 （公式 1）

 （公式 2）

其中为三个门来控制信息传递的路径；为向量元素乘积；为上一时刻的记忆单元；是通过非线性函数得到的候选状态：



在每个时刻t，LSTM网络的内部状态记录了到当前时刻为止的历史信息。

（2）门控机制：在数字电路中，门为一个二值变量{0, 1}，0代表关闭状态，不许任何信息通过；1代表开放状态，允许所有信息通过。LSTM 网络引入门控机制来控制信息传递的路径。三个“门”分别为输入门、遗忘门和输出门．这三个门的作用为：

1.遗忘门控制上一个时刻的内部状态需要遗忘多少信息。

2.输入门控制当前时刻的候选状态有多少信息需要保存。

3.输出门控制当前时刻的内部状态有多少信息需要输出给外部状态。

当时，记忆单元将历史信息清空，并将候选状态向量写入。但此时记忆单元依然和上一时刻的历史信息相关．当时，记忆单元将复制上一时刻的内容，不写入新的信息。

LSTM网络中的“门”是一种“软”门，取值在 (0, 1)之间，表示以一定的比例允许信息通过．三个门的计算方式为：

 （公式 3）

其中为Logistic函数，其输出区间为(0,1)，为当前时刻的输入，为上一时刻的外部状态。下图给出了 LSTM 网络的循环单元结构，其计算过程为：

1. 首先利用上一时刻的外部状态和当前时刻的输入，计算出三个门，以及候选状态；
2. 结合遗忘门和输入门来更新记忆单元；
3. 结合输出门，将内部状态的信息传递给外部状态。

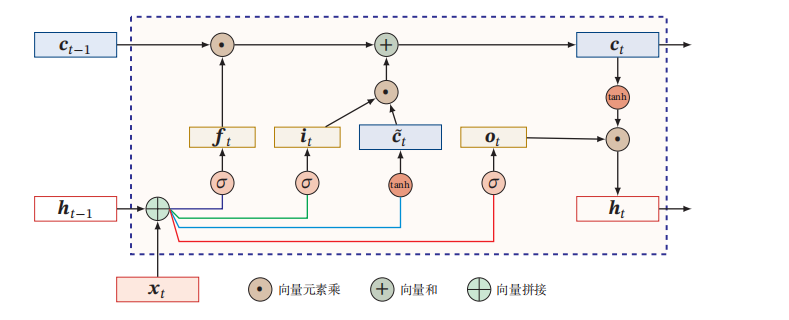


图 1 LSTM网络的循环单元结构示意图

循环神经网络中的隐状态h存储了历史信息，可以看作一种记忆。在简单循环网络中，隐状态每个时刻都会被重写，因此可以看作一种短期记忆。在神经网络中，长期记忆可以看作网络参数，隐含了从训练数据中学到的经验，其更新周期要远远慢于短期记忆．而在 LSTM 网络中，记忆单元c可以在某个时刻捕捉到某个关键信息，并有能力将此关键信息保存一定的时间间隔．记忆单元c中保存信息的生命周期要长于短期记忆h，但又远远短于长期记忆， 长短期记忆是指长的 因此称为长短期记忆。

## **四、实验步骤**

（1）导入航班客流量预测所需得包。

（2）导入航班流量数据。

（3）数据预处理。

（4）划分训练集和测试集。

（5）搭建LSTM模型并训练。

（6）预测结果及可视化。

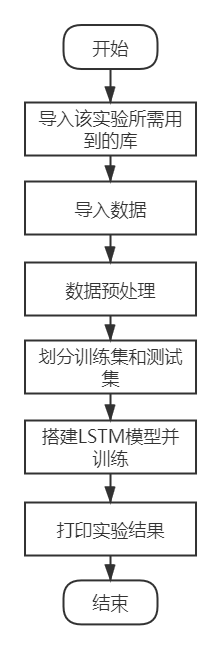


图 2 航班客流量预测流程图

## **五、实验代码及运行结果**

导入航班客流量预测所需得包。

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import torch

from torch import nn

from torch.autograd import Variable

%matplotlib inline

读取并可视化数据。

data\_csv = pd.read\_csv('./data/flights.csv', usecols=[2])

plt.plot(data\_csv)

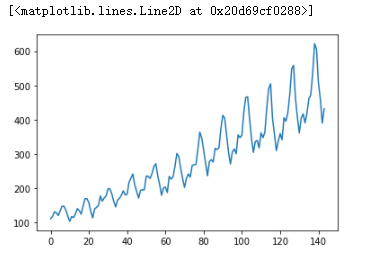


图 3 读取数据可视化

处理数据中的缺失值，转换成float32格式，然后获取最大最小值和间隔数量用于对数据的归一化处理。

# 数据预处理

data\_csv = data\_csv.dropna() # 滤除缺失数据

dataset = data\_csv.values # 获得csv的值

dataset = dataset.astype('float32')

max\_value = np.max(dataset) # 获得最大值

min\_value = np.min(dataset) # 获得最小值

scalar = max\_value - min\_value # 获得间隔数量

dataset = list(map(lambda x: x / scalar, dataset)) # 归一化

将数据划分为输入输出数据，然后按7：3划分训练集和测试集。

def create\_dataset(dataset, look\_back=2):

dataX, dataY = [], []

for i in range(len(dataset) - look\_back):

a = dataset[i:(i + look\_back)]

dataX.append(a)

dataY.append(dataset[i + look\_back])

return np.array(dataX), np.array(dataY)

# 创建好输入输出

data\_X, data\_Y = create\_dataset(dataset)

# 划分训练集和测试集，70% 作为训练集

train\_size = int(len(data\_X) \* 0.7)

test\_size = len(data\_X) - train\_size

train\_X = data\_X[:train\_size]

train\_Y = data\_Y[:train\_size]

test\_X = data\_X[train\_size:]

test\_Y = data\_Y[train\_size:]

转换矩阵格式，然后转换成张量格式。

train\_X = train\_X.reshape(-1, 1, 2)

train\_Y = train\_Y.reshape(-1, 1, 1)

test\_X = test\_X.reshape(-1, 1, 2)

train\_x = torch.from\_numpy(train\_X)

train\_y = torch.from\_numpy(train\_Y)

test\_x = torch.from\_numpy(test\_X)

搭建LSTM模型。

class lstm(nn.Module):

def \_\_init\_\_(self,input\_size=2,hidden\_size=4,output\_size=1,num\_layer=2):

super(lstm,self).\_\_init\_\_()

self.layer1 = nn.LSTM(input\_size,hidden\_size,num\_layer)

self.layer2 = nn.Linear(hidden\_size,output\_size)

def forward(self,x):

x,\_ = self.layer1(x)

s,b,h = x.size()

x = x.view(s\*b,h)

x = self.layer2(x)

x = x.view(s,b,-1)

return x

model = lstm(2, 4,1,2)

损失函数使用MSE均方误差，优化器使用adam优化算法。

criterion = nn.MSELoss()

optimizer = torch.optim.Adam(model.parameters(), lr=1e-2)

训练LSTM模型并每迭代100次打印损失。

# 开始训练

for e in range(1000):

var\_x = Variable(train\_x)

var\_y = Variable(train\_y)

# 前向传播

out = model(var\_x)

loss = criterion(out, var\_y)

# 反向传播

optimizer.zero\_grad()

loss.backward()

optimizer.step()

if (e + 1) % 100 == 0: # 每 100 次输出结果

print('Epoch: {}, Loss: {:.5f}'.format(e + 1, loss.item()))

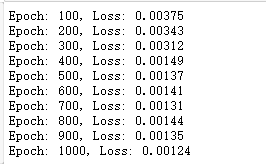


图 4 LSTM模型损失

将模型预测结果以及真实值用折线图表示。

model = model.eval() # 转换成测试模式

data\_X = data\_X.reshape(-1, 1, 2)

data\_X = torch.from\_numpy(data\_X)

var\_data = Variable(data\_X)

pred\_test = model(var\_data) # 测试集的预测结果

# 改变输出的格式

pred\_test = pred\_test.view(-1).data.numpy()

# 画出实际结果和预测的结果

plt.plot(pred\_test, 'r', label='prediction')

plt.plot(dataset, 'b', label='real')

plt.legend(loc='best')

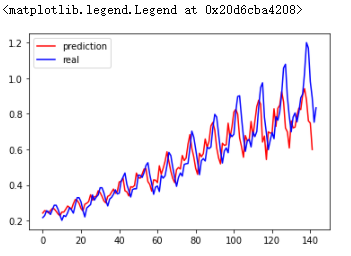


图 5 结果可视化

## **实验总结**

该实验中通过LSTM循环神经网络模型对航班10年的月流量进行建模，并对客流量进行预测，帮助我们进行旅游出行的规划。学生通过该实验应该LSTM循环神经网络基本概念、数据分析的基本方法以及图表可视化的基本使用。