基于 LSTM 模型的电力负荷预测研究

一、研究背景及意义

电力被广泛应用于我们生活的各个领域，包括工业、农业和商业和通信技术等，是现代工业发展的基础和人民生产生活不可或缺的重要条件。近年来，随着我国经济的迅猛发展以及产业变革的持续推进，全社会的用电需求量将普遍增加，并且随着“碳达峰、碳中和”进程的不断加快，国家将进一步推进能源转型的变革任务。而电能由于其清洁、高效、实用、便捷的特点，被作为能源低碳转型的关键选择，能源转型期间，电力的供需平衡压力将进一步增大，风能、太阳能等新能源的快速发展也使得电力负荷的波动性变大，因此电力系统的稳定供电将面临前所未有的挑战与风险。

但电力的生产和电力系统的运行是一个比较复杂的过程，其特殊性主要在于同时性和整体性，电能的生产过程、传送过程和使用过程几乎是同步进行的，即电力系统中的发电、输电和用电三大部分是一个不可分割的整体。并且电力能源不同于其他能源，其存储难度较大，具有即产即消的特点，这就要求电力系统的供给和需求能够保持实时的动态平衡。如果电力部门的供电量过大，会造成能源的浪费，但若供电量不能满足当地居民和企业的需求，则会严重影响人们的正常生活，阻碍社会生产发展，甚至导致整个电力系统的瘫痪。因此，为全方位保障我国民生及工业用电的可靠供应和电力系统的稳定运行，电力负荷预测应运而生。

现阶段的电力负荷预测研究按照时间尺度主要划分为短期预测、中期预测和长期预测，其中短期负荷预测的任务为预测未来若干时刻或未来几日的电力负荷，主要用于安全实时地对数据分析，以进行自动发电控制；中期负荷预测主要是根据历史同月数据的外推和年度规律的时间序列对未来一个月或几个月的电力负荷进行预测，主要用于合理安排日开停机计划和发电计划，协调电能调度，以满足市场需求；长期预测则主要针对国家政策调整、电力行业、人口及经济类等宏观因素对整个国家未来一年或几年内的电力负荷进行预测，以提供电源、电网规划的基础数据，确定年度检修计划以及国家电网系统的各环节规划。

二、研究思路及相关理论概述

（一）总体研究思路

考虑到时间序列分析法和回归分析法等比较传统的预测方法无法捕捉复杂

的负荷变化趋势，所以本案例采用 LSTM（长短期记忆神经网络）来对某地区的电力负荷进行短期预测，总体研究思路如下：

首先，采集某地区的历史电力负荷数据，考虑到电力负荷的变化与温度、日期等因素也有密切的联系，所以还要完成温度、湿度、降水量等气象因素数据以及日期类型数据的收集，然后对初始数据进行缺失值填充、异常值检测与修正等预处理工作，避免影响后续模型的预测精度。

其次，将原始数据集划分为训练集和测试集，在训练集上训练神经网络，然后在测试集上通过训练好的 LSTM 模型进行预测。

最后，计算该模型的预测误差，并作出预测值与实际值的对比图，以可视化地展示模型预测效果。

（二）相关理论知识

LSTM 是 RNN（循环神经网络）的变体，可以解决一般的循环神经网络在训练长序列过程中出现的梯度消失和梯度爆炸问题。它主要通过引入遗忘门f_t、输入门i_t和输出门o_t三个门结构来解决长期依赖的问题，三个门的具体公式如下：

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3)$$

式中σ代表 *sigmoid* 激活函数，W代表权重，h_{t-1}代表上一时刻的输出，x_t代表当前时刻的输入，b 代表偏置参数。

其中遗忘门f_t决定了上一时刻元胞状态C_{t-1}中哪些信息需要被遗忘，输入门i_t决定了候选状态$\widetilde{C_t}$需要保存和更新哪些信息，输出门o_t则决定了哪些信息可以被输出（李盖等，2023）。各个单元状态的具体公式如下：

$$\widetilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (4)$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \quad (5)$$

$$h_t = o_t * tanh(C_t) \quad (6)$$

式中$\widetilde{C_t}$代表当前时刻的候选状态，$tanh$代表 $tanh$ 激活函数，C_{t-1}代表上一时刻的元胞状态，h_t代表当前时刻的输出。

LSTM 通过引入门控机制来控制传输状态，筛选和更新重要信息，丢弃不重要的信息，以学习和记忆数据中的长期依赖关系，其总体网络结构图如图 1 所示。因此，相较于传统的循环神经网络，LSTM 在处理长序列时有更优异的表现，可以更好的应用于电力负荷时间序列预测。

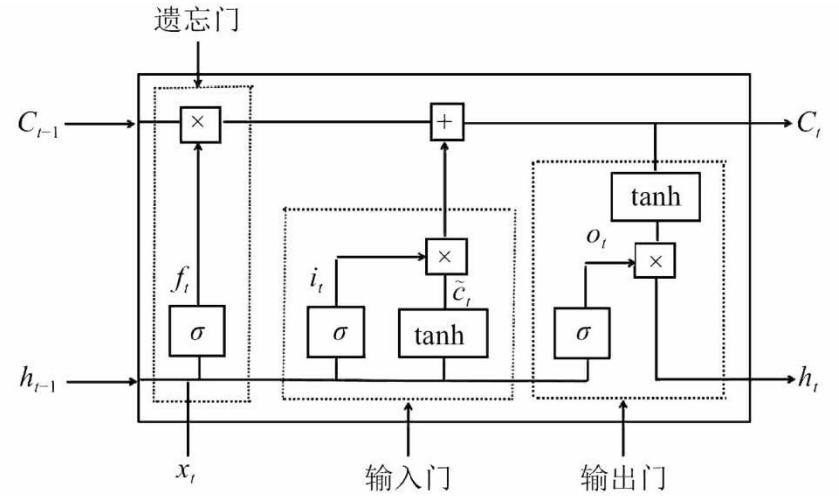


图 1　LSTM 网络结构图

（三）模型评价指标

为了使模型的评估效果更精确、更具说服性，本案例将采用 MAPE（平均绝对百分比误差）、MAE（平均绝对误差）、RMSE（均方根误差）和 SSE（误差平方和）四个指标对模型预测效果进行评价，计算公式如下：

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{y_t - \hat{y}_t}{y_t}\right| * 100\% \quad (7)$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t| \quad (8)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2} \quad (9)$$

$$SSE = \sum_{t=1}^{N}(y_t - \hat{y}_t)^2 \quad (10)$$

式中 N 代表预测值总数，y_t代表实际值，$\hat{y}_t$代表预测值。

三、实验数据集介绍

（一）电力负荷周期性分析

一个地区电力负荷的变化在年度、月度、日度方面均会呈现出明显的周期性，为研究数据的趋势走向提供了重要参考依据。本文选取的数据为“第九届电工数学建模竞赛”提供的数据集，包括 2012-2014 年的历史电力负荷序列及每日最高温度、每日最低温度、每日平均温度、每日相对湿度和每日降水量五种气象因素数据，其中电力负荷每 15 分钟采样一次，一天共 96 条负荷数据，另外还采集了相应的日期类型数据（周内=-1、周末=0、节假日=1），下面将以该数据集为依据从多方面分析电力负荷的周期性。

图 2 展示了某地区 2013 年与 2014 年的年度负荷曲线图，该地区 2013 年与 2014 年的电力负荷曲线走势基本一致，但 2014 年电力负荷值比 2013 年总体稍高，表明 2014 年的社会经济水平和人民生活水平较 2013 年有所提升。从年度负荷曲线整体来看，其波峰出现在 7 月，由于夏季温度较高，无论是居民还是企业都有大量的制冷需求，导致每年的电力负荷都在夏季明显增高。而 2013 年与 2014 年的波谷出现时间虽然并不一致，但均在农历的春节期间，说明春节假期，大量工人停工返乡，工厂设备几乎全部停止工作，导致电力负荷急剧下降。春秋两季由于气温宜人，电力负荷无剧烈波动，整体走向趋于平稳。

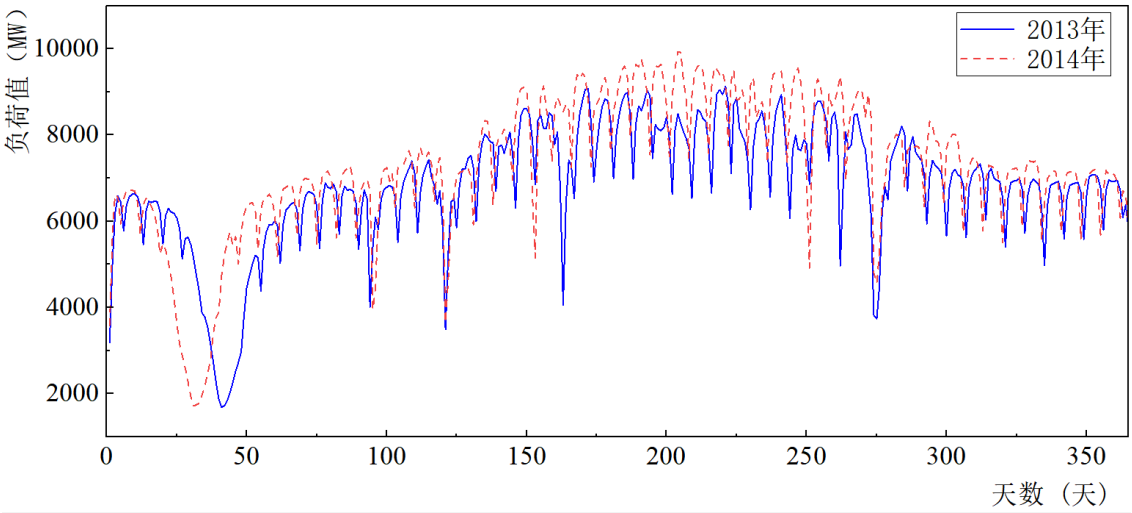


图 2　某地区年度电力负荷曲线对比图

图 3 展示了该地区 2014 年 3 月的负荷曲线图，从月度曲线可以看出每周的电力负荷曲线呈现出相同的变化趋势，即周内的负荷值较高，而周末负荷大幅降低，这是因为工厂和企业的员工周内上班、周末休息，导致电力负荷在每周之间

呈现出显著的周期性规律。

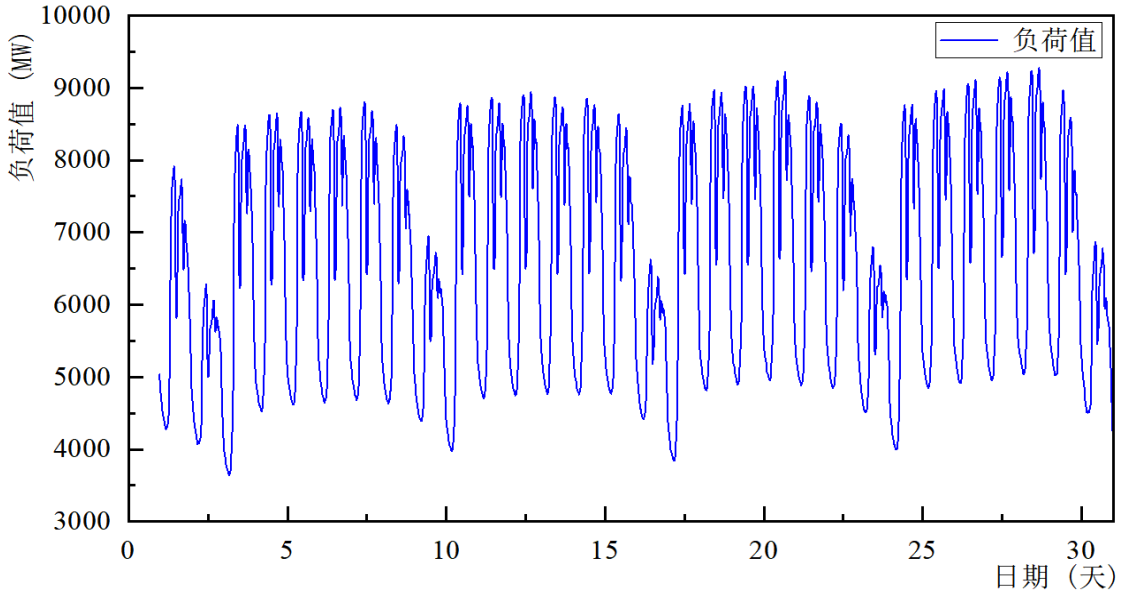


图 3 某地区月度负荷曲线图

日负荷曲线图如图 4 所示，图中选取了 3 天作为示例，其中 2014 年 2 月 28 日和 7 月 1 日均为周内，2014 年 3 月 1 日为周六，但三条日负荷曲线的趋势一致，说明每天的负荷变化规律在不同季节和不同日期类型都是类似的，因此日负荷变化的周期性对任意日期都是适用的。全天的负荷最低值位于早上六点左右，最高值则出现在中午 11 点和下午 17 点，在上午 8 点和下午 14 点，由于企业员工开始上班工作，负荷值急剧增大，在上下午工作时间平稳上升，而在中午 12 点和晚上 18 点明显下降，到晚上 8 点左右出现一个较小的晚高峰，之后便成逐渐下降的趋势。

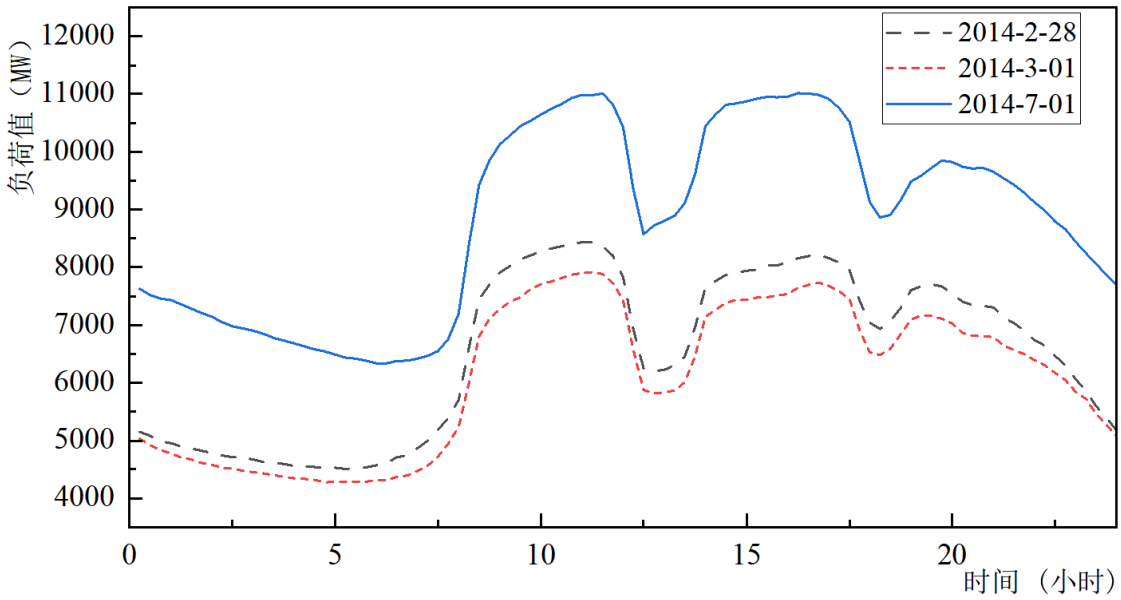


图 4　某地区日负荷曲线图

（二）气象因素及日期对电力负荷的影响分析

一个城市的电力负荷不仅会受国家经济发展的影响，还会受到其他多种因素的影响，尤其是温度、湿度、降水量等气象因素和日期类型的影响。

1. 温度对电力负荷的影响

温度对电力负荷的影响最为明显和直观，如图 5 所示，从全年来看，夏季的温度和电力负荷的增减趋势一致，呈现正相关性，而在冬季，温度和电力负荷则呈现负相关性，这是因为当夏季温度较高时，空调、冰箱等制冷电器用量较大，电力负荷也随之出现波峰，而在冬季居民又有较高的取暖需求，所以温度越低，电力负荷越大。

除此以外，气温变化不仅会导致制冷或供暖用电量增加，还会影响工农业生产和交通运输，进而导致电力负荷的变化，比如寒冷天气会导致一些工业设备无法正常运行，需要通过额外的电力设备进行升温来维持其正常运转，使得电力负荷增大。

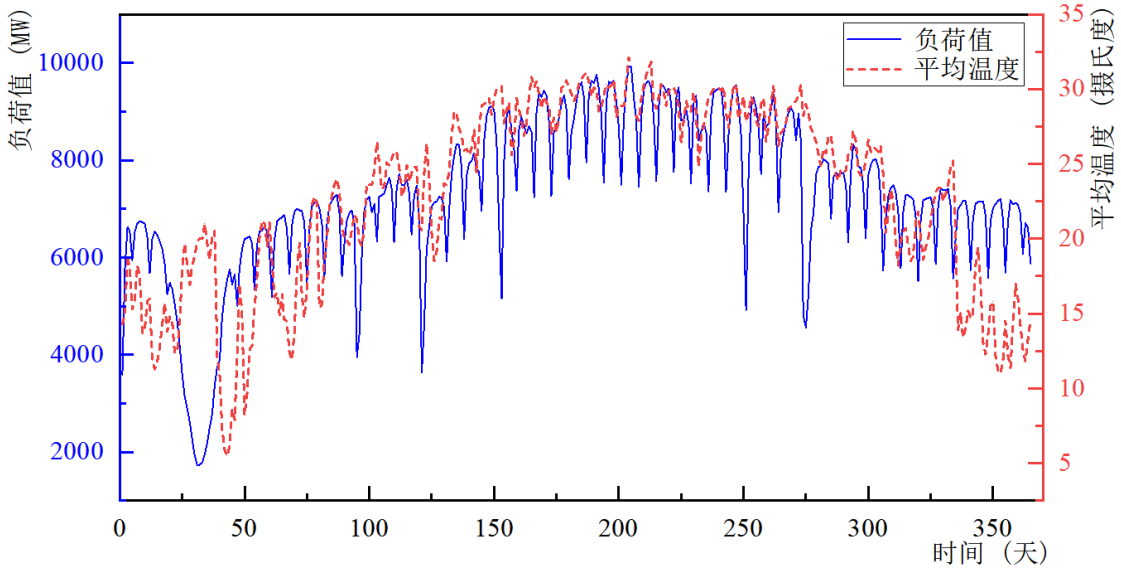


图 5　温度对电力负荷的影响曲线图

2. 湿度对电力负荷的影响

湿度对电力负荷也有着不容忽视的影响，观察 30 天内湿度对电力负荷的影响曲线图，如图 6 所示，可以看出相对湿度与电力负荷之间呈现出比较明显的负相关性，这是由于湿度会影响人们对环境的舒适度，当湿度比较低时，人们可能会使用加湿器来保证日常日常生活需求，从而导致用电负荷增加。

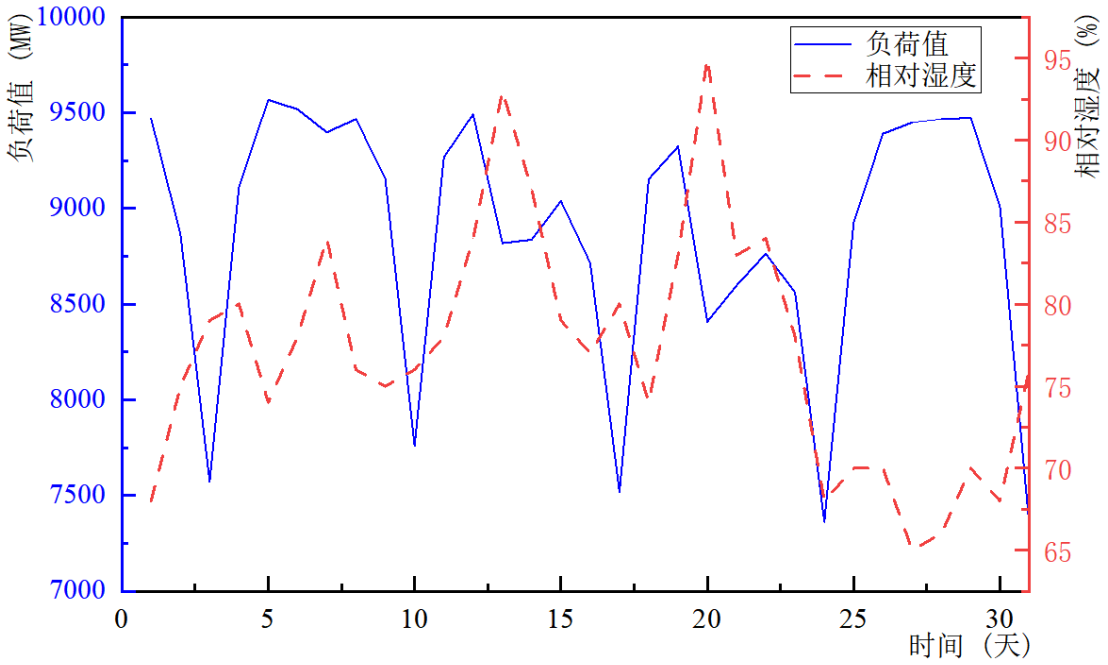


图 6　湿度对电力负荷的影响曲线图

3. 降水量对电力负荷的影响

降水量会通过影响温度、湿度，从而间接影响到电力负荷。如图 7 所示，当降水量较大时，温度降低，使得负荷值暂时下降，而夏季短期强降雨的影响则会更大，它会导致负荷值骤然下降到一个很低的水平，但在冬季，降雪会使温度降低，从而导致电力负荷增加，因此，在预测短期电力负荷时，降水量也将作为一个必不可少的自变量因素。

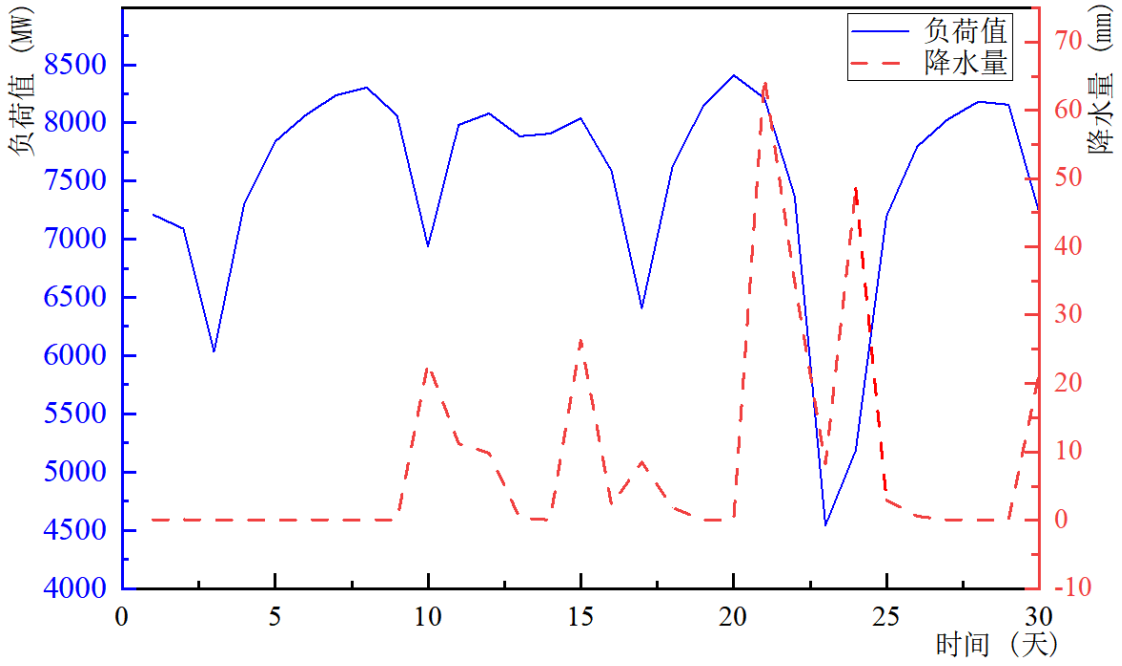


图 7　降水量对电力负荷的影响曲线图

4. 日期类型对电力负荷的影响

除以上气象因素对电力负荷有比较明显的影响以外，日期类型也会影响电力负荷的波动。图 8 展示了普通周内、普通周末、元旦、五一和十一五天的日负荷曲线，可以看到普通周内和周末的日负荷曲线趋势基本一致，但节假日期间，人们的生活习惯发生较大的变化，负荷变化规律也随之变化，呈现出与普通周内和周末完全不同的变化趋势。因此，日期类型（周内=-1、周末=0、节假日=1）会决定预测日的曲线走势，在短期负荷预测中起着不可忽视的作用。

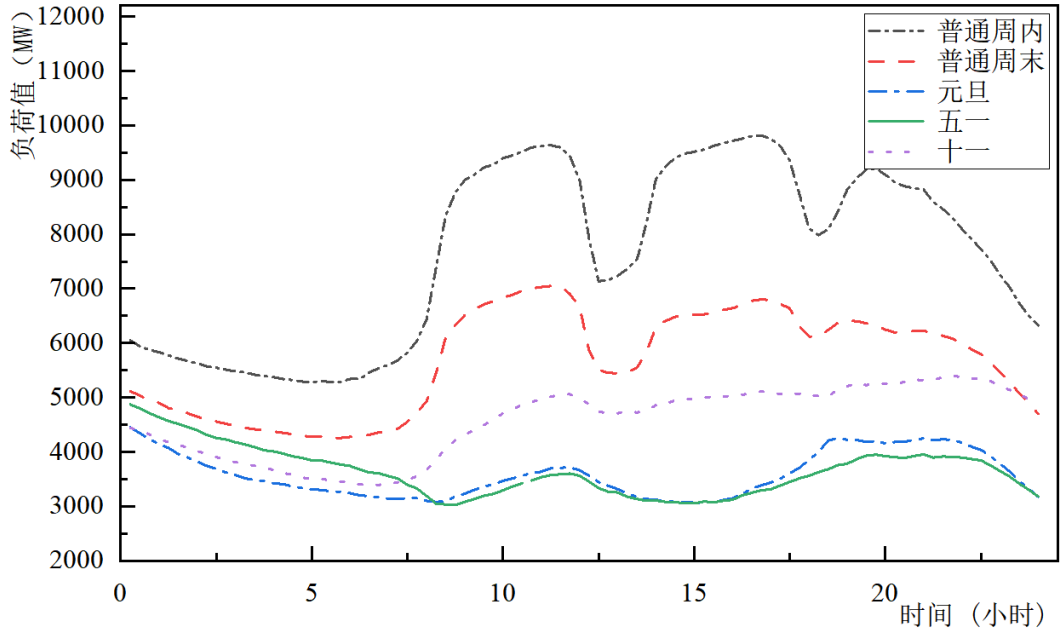


图 8　日期类型对电力负荷的影响曲线

四、实验操作

本案例立足于短期电力负荷预测，将基于历史电力负荷数据及相关影响因素数据，构建 LSTM 模型对未来一天内 96 个时刻的电力负荷进行预测。

（一）导入库（若电脑中没有，可通过 pip install 指令进行安装）

```
import pandas as pd
import numpy as np
from keras import Sequential
from keras.src.layers import Dense, LSTM, Dropout
from numpy import concatenate
from pandas import concat
from sklearn.preprocessing import MinMaxScaler
from scipy.ndimage import gaussian_filter
import matplotlib.pyplot as plt
```

（二）导入数据，并进行数据预处理工作

```
# 读取数据
load_1 = pd.read_csv(r'E:\STUDY\毕设\数据\Area1_Load.csv', encoding='utf-8')
argument_1 = pd.read_csv(r'E:\STUDY\毕设\数据\Area1_Argument.csv',
encoding='utf-8')

# 异常值检测与修正
def outlier_detection(data):
    # 统计行数与列数
    num_rows = data.shape[0]
    num_cols = data.shape[1]
    # 检测并修正异常值
    for i in range(0, num_rows - 1):
        for j in range(2, num_cols - 1):
            if abs(data.iloc[i, j] - data.iloc[i, j - 1]) > 0.05 and abs(
                    data.iloc[i, j] - data.iloc[i, j + 1]) > 0.05:
                data.iloc[i, j] = (data.iloc[i, j - 1] + data.iloc[i, j + 1]) / 2
    return data

# 对历史电力负荷数据进行异常值检测和修正
load_1 = outlier_detection(load_1)

# 把 96 个时刻的负荷数据和当天的天气数据合并，分别存放到 csv 中
load = load_1.iloc[:, 1:]
arg = argument_1.iloc[:, 1:]
for i in range(96):
    load_0 = pd.DataFrame({'Load': load.iloc[:, i], 'HighTemp': arg.iloc[:, 0],
                           'LowTemp': arg.iloc[:, 1], 'AveTemp': arg.iloc[:,
2],
                           'Humidity': arg.iloc[:, 3], 'Rainfall': arg.iloc[:, 4],
                           'DateType': arg.iloc[:, 5]})
    # 为整理后的数据添加时间戳
    load_0.index = pd.date_range(start='2012-01-01', periods=1106, freq='D')
    load_0.to_csv('E:/STUDY/毕设/数据/Load_Arg/Load_%d.csv' % i,
index=True, header=True)
```

（三）构建 LSTM 预测模型，本案例用前 7 天同一时刻的电力负荷、气象及日期类型数据预测后一天对应时刻的负荷值

```
# 将数据缩放到 [-1, 1]之间
def scale(train, test):
    # 构造最小-最大规范化缩放器
    scaler = MinMaxScaler(feature_range=(-1, 1))
```

```
    # 对训练数据进行转换
    train_scaled = scaler.fit_transform(train)
    # 对测试数据进行转换
    test_scaled = scaler.transform(test)
    return scaler, train_scaled, test_scaled

# 将时间序列数据转换为监督型学习数据，即把前 1 天-前 n 天的数据都作为影响因素
# 其中 data 为时间序列数据，n_in 为输入序列长度，n_out 为输出序列长度
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = pd.DataFrame(data)
    cols, names = [], []
    # 输入序列（t-n, ... ,t-1）
    # i: n_in, n_in-1, ..., 1
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j + 1, i)) for j in range(n_vars)]
    # 输出序列，即预测序列（t,t+1, ... ,t+n）
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j + 1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j + 1, i)) for j in range(n_vars)]
    # 合并
    agg = concat(cols, axis=1)
    agg.columns = names
    # 删除具有缺失值 NaN 值的行
    if dropnan:
        agg.dropna(inplace=True)
    return agg

# 初始化保存预测结果的列表
result = []
# 由于电力负荷每天呈现出相同的周期性，所以对于每一个时刻，都是基于前 7 天的历史负荷值、气象数据和日期类型进行预测
# 循环遍历 96 次完成未来一天内 96 个时刻电力负荷的预测
for i in range(96):
    print("现在在预测第%d 个时刻" % i)
# 读取前面保存的电力负荷和天气合并之后的 csv 文件
    load = pd.read_csv('E:/STUDY/实验/数据/Load_Arg/Load_%d.csv' % i,
encoding='utf-8')
    load = load.iloc[:, 1:]
```

```
        # 划分训练集与测试集，可自定义训练集与测试集的比例
        split_rate = 0.8
        train_size = int(load.shape[0] * 0.8)
        train = load.iloc[0:train_size, :]
        test = load.iloc[train_size:, :]
        # 将数据统一映射到[-1,1]的区间内，以提高收敛速度
        scaler, train_scaled, test_scaled = scale(train, test)
        # 用前 7 天的七个特征预测下一天同一时刻的负荷
        # 将训练集转换为监督型学习数据
        train_supervised = series_to_supervised(train_scaled, 7, 1)
        train_X, train_y = train_supervised.iloc[:, 0:49], train_supervised.iloc[:, 49]
        train_X = train_X.values.reshape(train_X.shape[0], 7, 7)
        # 将测试集转换为监督型学习数据
        test_supervised = series_to_supervised(test_scaled, 7, 1)
        test_X, test_y = test_supervised.iloc[:, 0:49], test_supervised.iloc[:, 49]
        test_X = test_X.values.reshape(test_X.shape[0], 7, 7)
        # print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
        # 搭建 LSTM 模型
        model = Sequential()
        model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
        # model.add(Dropout(0.2))
        model.add(Dense(1))
        model.compile(loss='mean_squared_error', optimizer='adam')
        model.fit(train_X, train_y, epochs=50, batch_size=32, verbose=2, shuffle=False)
        # 使用刚刚训练的 LSTM 模型进行预测
        prd = model.predict(test_X)
        # 逆缩放时 shape 应与原来的大小一致，所以需将预测结果和部分测试集数据组合后再逆缩放
        # 将预测的电力负荷值进行逆缩放
        test_X = test_X.reshape((test_X.shape[0] * test_X.shape[1], test_X.shape[2]))
        inv_prd = concatenate((prd, test_X[0:len(prd), 1:]), axis=1)
        inv_prd = scaler.inverse_transform(inv_prd)
        inv_prd = inv_prd[:, 0]
        # 将测试集中的实际电力负荷值进行逆缩放
        test_y = test_y.values.reshape((len(test_y), 1))
        inv_y = concatenate((test_y, test_X[0:len(test_y), 1:]), axis=1)
        inv_y = scaler.inverse_transform(inv_y)
        inv_y = inv_y[:, 0]
        # 将实际值与预测值合并，方便后续查看
        prd_act =np.concatenate((inv_y,inv_prd))
        result.append(prd_act)
# 将预测结果保存到 csv 中
# 共 96 行对应着 96 个时刻，每行前一半的列表示实际值，后一半的列表示对应的预测值
```

```
load_pred = pd.DataFrame(result)
load_pred.to_csv('E:/STUDY/实验/数据/Load_Arg/result.csv', encoding='utf-8')
```

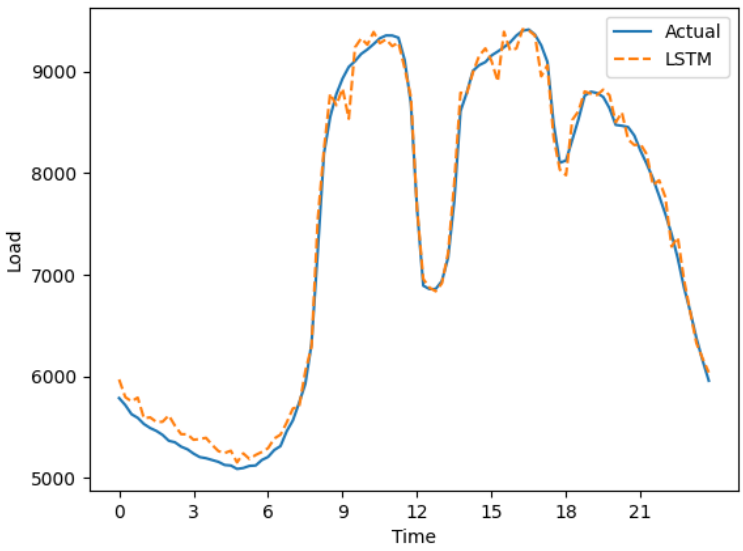
（四）通过计算 MAPE、MAE、RMSE 和 SSE 四个指标，来衡量模型预测效果

```
# 计算平均绝对百分比误差 MAPE
def calculate_mape(actual, pred):
    actual = np.array(actual)
    pred = np.array(pred)
    return np.mean(np.abs((actual - pred) / actual))

# 计算平均绝对误差 MAE
def calculate_mae(actual, pred):
    actual = np.array(actual)
    pred = np.array(pred)
    return np.mean(np.abs(actual - pred))

# 计算均方根误差 RMSE
def calculate_rmse(actual, pred):
    actual = np.array(actual)
    pred = np.array(pred)
    squared_errors = (actual - pred) ** 2
    return np.sqrt(np.mean(squared_errors))

# 计算误差平方和 SSE
def calculate_sse(actual, pred):
    actual = np.array(actual)
    pred = np.array(pred)
    squared_errors = (actual - pred) ** 2
    return np.sum(squared_errors)


# 读取刚才保存预测值与对应实际值的 csv 文件
result = pd.read_csv('E:/STUDY/实验/数据/Load_Arg/result.csv', encoding='utf-8')
# 获取列数,行数不用获取共 96 行
column_count = result.shape[1]
# 获取测试集的预测天数
days = int((column_count-1)/2)
# 初始化各个评价指标
mape = 0
mae = 0
rmse = 0
sse = 0
#分别计算测试集中每天 96 个时刻的误差
```

```
for i in range(1,days+1):
        actual = result.iloc[:,i]
        pred = result.iloc[:,i+days]
        mape += calculate_mape(actual,pred)
        mae += calculate_mae(actual,pred)
        rmse += calculate_rmse(actual,pred)
        sse += calculate_sse(actual,pred)
# 输出整个测试集的平均误差
print("MAPE 为" + str(mape/days) + ",MAE 为" + str(mae/days)
          + ",RMES 为" + str(rmse/days) + ",SSE 为" + str(sse/days))
```

（五）作预测值与实际值的对比图，以可视化呈现预测效果

```
# 随便选取测试集中的一天画实际值与预测值的对比图
# 选择不同的列对应着不同天的数据，测试集每天的预测误差可能有大有小，对比图呈现出的效果可能也会有所不同
actual = result.iloc[:,1]
lstm = result.iloc[:,1+days]
plt.plot(actual, label='Actual')
plt.plot(lstm, label='LSTM', linestyle='--')
# 由于对每个时刻单独预测，相邻时刻之间的数据可能会比较分散，所以使用高斯滤波对预测数据进行平滑处理
lstm = gaussian_filter(lstm, sigma=1)
plt.xlabel('Time')
plt.xticks([0, 12, 24, 36, 48, 60, 72, 84], ['0', '3', '6', '9', '12', '15', '18', '21'])
plt.ylabel('Load')
plt.legend()
plt.show()
```



效果图样例