枢纽客流量BP神经网络预测模型

1. 研究意义：

智慧交通是智慧城市的重要组成部分，提高公交车站点客流量预测的准确度是智慧公交的研究内容之一。大型现代化综合交通枢纽站汇集多种交通方式并伴有大规模客流集散、换乘等特点，采用科学的方法研究大型综合交通枢纽站的各个交通方式的客流量预测对公交运营部门的快速决策和综合管理提供及时准确的数据参考具有现实意义。

本算法模型有效的解决了交通枢纽各个交通方式短期客流量的预测问题， 对未来的小时客流量和5分钟客流量进行短期预测，能够有效的预测客流量爆发点和未来各个时段客流量的走势，为交通枢纽的运力分配策略提供数据支持。模型充分考虑了节假日、天气变化等因素对客流量的影响，能够提高对于特殊节日和天气影响下的客流量预测精度。

模型首先分析影响客流波动的因素（节假日、天气因素以及时间序列等），然后通过对比各类预测方法的优缺点，设计改进的BP神经网络算法进行短期客流量预测。

2.预期目标和现阶段的成果：

预测模型将提供对交通枢纽地铁、公交、火车等交通方式的客流量预测，实现客流量爆发预警功能，为枢纽的运力调配提供数据支持。同时，客流量的预测和分析，能够为优化枢纽的空间布置和人力分配提供政策建议。

目前，基于Keras深度学习库搭建的BP神经网络预测模型目前已完成对上海虹桥地铁进出站的客流量预测，小时客流量预测精度可以达到10%以下。能够有效的实现未来一周的客流量预测。未来通过对预测算法输入特征值的优化以及对部分特殊情况的规则提取和分析（比如特殊的节假日情况，突发事件等），算法的精度将得到进一步提高。

未来将打造一个完整的交通枢纽客流量预测系统，为交通枢纽的运力决策和综合管理优化提供数据支持和政策建议。该系统可以支持交通枢纽站的地铁、公交、火车等交通方式的客流量预测，届时需要拿到公交刷卡记录和火车站的进出站闸机历史csv数据。

3.模型功能：

模型能够有效预测未来一周枢纽内各个交通方式的客流量大小，提前为枢纽运力分配提供策略。模式采用有监督的机器学习方式，在未来随着训练（历史）数据的不断积累，能够实现自我调整和参数优化，不断提高预测的精度。

根据给定的输入参数： INPUT[日期(eg:20180901)，时段(eg:8)，节假日情况（周末or其他节假日), 天气情况（降雨），最高温度，风力]

输出未来客流量的预测值： OUTPUT[Passenger_flow / h]

4. 网络结构：

网络采用7维的特征值向量作为输入参数，两层64节点的隐含层，隐含层采用“relu”激活函数，输出层是1维的客流量数值。下面是网络的结构：

INPUT[x1,x2...,x7] -> Hiddden layer1[64 Nodes, 'relu' activation function] -> Hidden layer2[64 Nodes,'relu] -> OUTPUT[Passenger_flow]

5.情景分析-上海虹桥地跌进出站客流分析：

本章节采用上海虹桥实际的地铁进出站客流数据进行建模和验证，分析算法模型对与客流量预测的可行性。本章详细的介绍了模型的搭建流程，包括运行环境的检查， 输入数据的处理分析以及预测模型的训练和结果预测。 模型构建的目录如下：

5.1. 环境检查
5.1.1 Python版本检查
5.1.2 基础包导入检查
5.1.3 检查安装版本

5.2. 客流量数据预处理和特征分析
5.2.1 读取数据
5.2.2 提取闸机数据，转换为小时的客流量数据
绘制 客流量 / 小时 统计图
绘制 节假日 / 日期 统计图

5.3. 搭建神经网络预测模型
5.3.1 网络输入数据标准化
5.3.2 搭建网络框架
5.3.3 训练神经网络
5.3.4 预测结果

5.1 环境检查

5.1.1 Python版本检查

python 3 on anaconda

```
In [14]: !python3 --version
```

```
Python 3.7.0
```

Anaconda 环境配置（mac） https://www.jianshu.com/p/b9eac8419c8d 需要的环境参数：export PATH=/Users/zhangfujiang/anaconda3/bin:$PATH， 配置环境变量：conda env create -f /Users/zhangfujiang/virtual_platform_mac.yml

5.1.2 基础包导入检查

In [2]:

```
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
```

在终端运行 pip install tensorflow and keras 源： -i https://pypi.tuna.tsinghua.edu.cn/simple

In [450]:

```
import tensorflow
import keras
print('tensorflow:', tensorflow.__version__)
print('keras:', keras.__version__)
!python3 --version
```

```
tensorflow: 1.14.0
keras: 2.2.4
Python 3.7.3
```

5.1.3 检查安装版本

In [5]:

```
import numpy
print('numpy:', numpy.__version__)
import scipy
print("scipy:", scipy.__version__)
import matplotlib as plt
print('matplotlib:', matplotlib.__version__)
```

```
numpy: 1.16.4
scipy: 1.3.0
matplotlib: 3.1.0
```

5.2 客流量数据预处理和特征分析

In [1]:

```
from keras.datasets import boston_housing
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import RMSprop
import numpy as np
import csv
import os
import random
import pandas as pd
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

In [39]:

```
# 把日期hash计算，转换为int值，日期月近，int值越大 eg: '20190801' -> 20190801
# date 指标： 反映时间的变化
def transfer_time(str):
    data = str.split(' ')
    if len(data) == 1:
        date = data[0]
        hour = 0
        #pdb.set_trace()
    else:
        date = data[0]
        hour = data[1]
    try:
        temp_date = date.split('/')
        if len(temp_date[1]) == 1:
            temp_date[1] = '0' + temp_date[1]
        if len(temp_date[2]) == 1:
            temp_date[2] = '0' + temp_date[2]
        date = int(temp_date[0] + temp_date[1] + temp_date[2])
        temp_hour = hour.split(':')
        hour = (float(temp_hour[0]) * 100 + float(temp_hour[1]))/100.0
    # return 日期和时段
    except:
        print(date)
        hour = 0
    return date, hour
```

5.2.1 读取数据

In [3]:

```
# 读取data
# (train_data, train_target), (test_data, test_target) 获取
"""
csv_data 存储了客流量的数据
csv_data2 存储了日期-天气数据
csv_data3 存储了日期-节假日数据
"""
csv_data = pd.read_csv('地铁客流量.csv')
csv_data2 = pd.read_csv('weather_result.csv')
csv_data3 = pd.read_csv('holiday1.csv')
```

In [4]:

```
# 看是否存在缺失值
csv_data.info()
csv_data2.info()
csv_data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 320117 entries, 0 to 320116
Data columns (total 9 columns):
ID                  320117 non-null int64
W_C_AREA_CODE       320117 non-null object
TJ_TIME             320117 non-null object
IN_TOTAL            320117 non-null int64
OUT_TOTAL           320117 non-null int64
REMARK              320117 non-null int64
W_M_WAYKIND_NAME    320117 non-null object
Unnamed: 7          0 non-null float64
Amount              320117 non-null int64
dtypes: float64(1), int64(5), object(3)
memory usage: 22.0+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 573 entries, 0 to 572
Data columns (total 7 columns):
日期      573 non-null int64
周数      573 non-null int64
最高温度    573 non-null int64
最低温度    573 non-null int64
天气      573 non-null int64
风向      573 non-null int64
风力      573 non-null int64
dtypes: int64(7)
memory usage: 31.4 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 587 entries, 0 to 586
Data columns (total 2 columns):
日期    587 non-null int64
类型    587 non-null int64
dtypes: int64(2)
memory usage: 9.2 KB
```

In [5]:

```
# 将数据整理成date -> data 的字典
dict_weather = {}
dict_week = {}
dict_high_temp = {}
dict_low_temp = {}
dict_wind = {}
dict_holiday = {}
# 读取前N条数据
N = 300000
# dataFrame 取前N条
csv_batch_data = csv_data.head(N) #tail
csv_batch_data2 = csv_data2
csv_batch_data3 = csv_data3
# values dataFrame->array  Transfer array data to dictionary:
# data[0] - > date
for data in csv_batch_data2.values:
    dict_weather[data[0]] = data[4]
    dict_week[data[0]] = data[1]
    dict_high_temp[data[0]] = data[2]
    dict_low_temp[data[0]] = data[3]
    dict_wind[data[0]] = data[6]
for data in csv_batch_data3.values:
    dict_holiday[data[0]] = data[1]
print('csv_data shape:', csv_batch_data.shape)
print('csv_data2 shape:', csv_batch_data2.shape)
print('csv_data3 shape:', csv_batch_data3.shape)
# train_batch_data = csv_batch_data[list(range(3, 6))]  取3:5列
# dataFrame 取行
train_batch_data = csv_batch_data[1:]
#  按照机器类型，将数据分为两组
# 314684 row 之后只有machine1 数据
machine1_data = []
machine2_data = []
# data 为list:  ID 0 , W_C_AREA_CODE 1, Time 2, In_total 3, Out_total 4.  DataFrame.values -> array
for data in train_batch_data.values:
    if data[1] == '05-19-10':
        # list of array -> [ID 0 , W_C_AREA_CODE 1, Time 2, In_total 3, Out_total 4]
        machine1_data.append(data)
    elif data[1] == '04-17-03':
        machine2_data.append(data)
    else:
        print("machine type error!")
# Passenger flow =  In_total + Out_total
print('machine1_data shape:', len(machine1_data))
print('machine2_data shape:', len(machine2_data))
```

```
csv_data shape: (300000, 9)
csv_data2 shape: (573, 7)
csv_data3 shape: (587, 2)
machine1_data shape: 147952
machine2_data shape: 152047
```

In [6]:

```python
# debuger:
import pdb
```

5.2.2 提取闸机数据，转换为小时的客流量数据

In [7]:

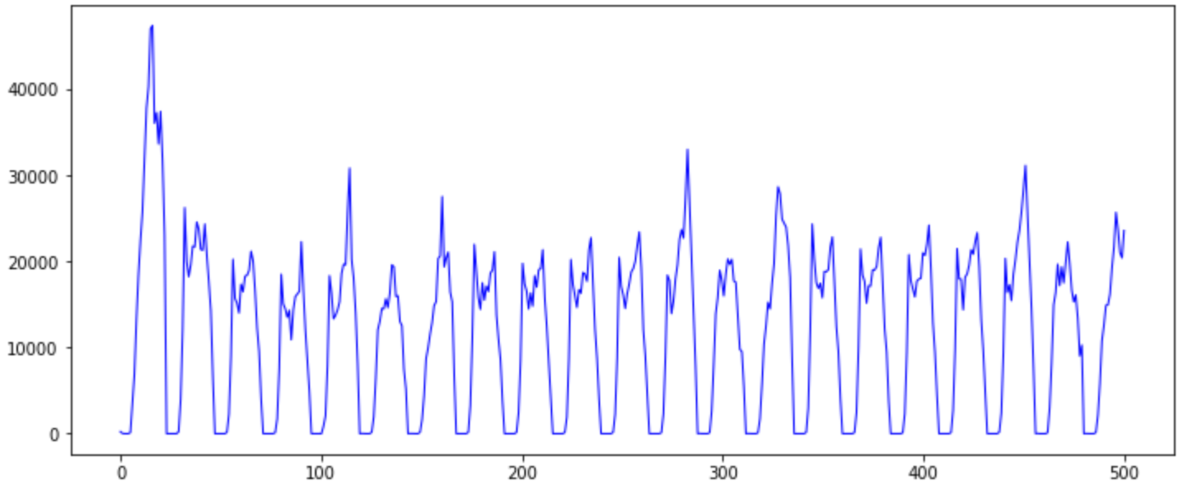
```python
"""
统计小时划分的客流量数据
"""
# [Time 2, In_total 3, Out_total 4] -> machine_data: [Date 0, hour 1, In_total 3, Out_total 4]
# data_set and target_set 按照日期-时段为key进行划分
# 小时统计算法：记录第一个点record = record(h0)，遍历，if date-hour not changed, update flow, else: dict[date-h_pre] = flow - record, update record = flow.
data_set1 = {}
data_set2 = {}
target_set1 = {}
target_set2 = {}
# 每到下一天，要清 0
record = 0
flow = 0
flag_start = True
date_start = True
pre_x = []
pre_date = 0
for i in range(len(machine1_data)):
    x = []
    machine1_data[i][0], machine1_data[i][1] = transfer_time(machine1_data[i][2])
    # shouldn't contain passenger flow in machine2
#     machine1_data[i][3] += machine2_data[i][3]
#     machine1_data[i][4] += machine2_data[i][4]
    in_total = machine1_data[i][3]
    out_total = machine1_data[i][4]
    date = machine1_data[i][0]
    hour = machine1_data[i][1]
    if flag_start:
        key_start = str(machine1_data[0][0])+str(int(machine1_data[0][1]))
        flag_start = False
    if date_start:
        pre_date = date
        date_start = False
    weather = -1
    week = -1
    holiday = -1
    high_temp = -1
    low_temp = -1
    wind = -1
    # if date = data2[j][0]  改为dict,  hash 查询
    if date in dict_weather:
        weather = dict_weather[date]
    if date in dict_week:
        week = dict_week[date]
    if date in dict_high_temp:
        high_temp = dict_high_temp[date]
    if date in dict_low_temp:
        low_temp = dict_low_temp[date]
    if date in dict_wind:
        wind = dict_wind[date]
    if date in dict_holiday:
        holiday = dict_holiday[date]
   # x1: date week hour --> predict amount
   # holiday variable doesn't exists:
    x.append(date)
    x.append(int(hour))
    x.append(week)
    x.append(weather)
    x.append(holiday)
    x.append(high_temp)
    x.append(low_temp)
    key = str(date)+str(int(hour))
    if key == key_start:
        flow = (in_total + out_total)
    # 换下一个小时:
    else:
        # 换到下一天, flow还是上一天的流量总和
        if date != pre_date:
            flow = in_total + out_total
            #pdb.set_trace()
            record = 0 # 上一次的累计客流量
            pre_date = date
            target_set1[key_start] = flow - record
            data_set1[key_start] = pre_x
        else:
            target_set1[key_start] = flow - record
            data_set1[key_start] = pre_x
            record = flow
        key_start = key
    pre_x = x
#print(data_set1)
#print(target_set1)
# print(machine1_data) : [Date 0, hour 1, In_total 3, Out_total 4]
```

In [8]:

```
# 输出一天Passenger Flow / h 实际值   --> machine 1
fig= plt.figure(figsize=(12,5))
y = []
x_data = list(target_set1.keys())
x = x_data[0:500]
for key in x:
    y.append(target_set1[key])
y = np.array(y)
x_data = np.linspace(0, 500, len(y))
#x_data = np.array(x_data)
plt.plot(x_data,y,'b-',lw=1)
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x63b695940>]
```



In [40]:

```
"""
machine2 的数据读取
"""
# 每到下一天，要清 0
record = 0
flow = 0
flag_start = True
date_start = True
pre_x = []
pre_date = 0
#  统计不合格的数据： only date no hours
count_illegal = 0
for i in range(len(machine2_data)):
    x = []
    machine2_data[i][0], machine2_data[i][1] = transfer_time(machine2_data[i][2])
    # pdb.set_trace()
    # shouldn't contain passenger flow in machine2
#     machine1_data[i][3] += machine2_data[i][3]
#     machine1_data[i][4] += machine2_data[i][4]
    in_total = machine2_data[i][3]
    out_total = machine2_data[i][4]
    date = machine2_data[i][0]
    hour = machine2_data[i][1]
    if flag_start:
        key_start = str(machine1_data[0][0])+str(int(machine1_data[0][1]))
        flag_start = False
    if date_start:
        pre_date = date
        date_start = False
    key = str(date)+str(int(hour))
    if key == key_start:
        flow = (in_total + out_total)
    # 换下一个小时：
    else:
        # 换到下一天，flow还是上一天的流量总和
        if date != pre_date:
            flow = in_total + out_total
            #pdb.set_trace()
            record = 0 # 上一次的累计客流量
            pre_date = date
            target_set2[key_start] = flow - record
        else:
            target_set2[key_start] = flow - record
            record = flow
        key_start = key
    pre_x = x
#print(data_set1)
#print(target_set1)
# print(machine1_data) : [Date 0, hour 1, In_total 3, Out_total 4]
```

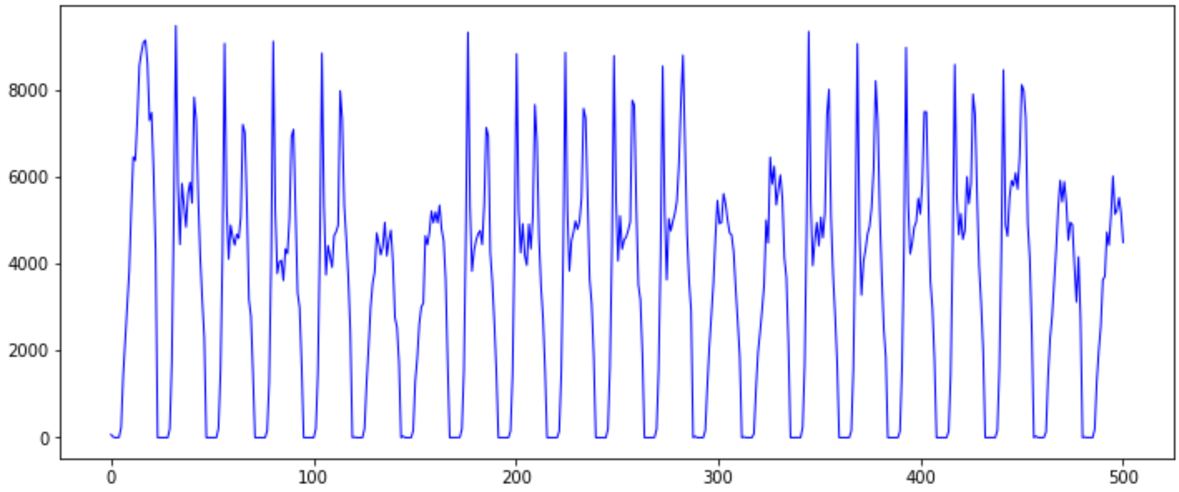
```
20180317
20180814
20181020
20190124
20190214
```

In [41]:

```
# 输出一天Passenger Flow / h 实际值  --> machine 2
fig= plt.figure(figsize=(12,5))
y = []
x_data = list(target_set1.keys())
x = x_data[0:500]
for key in x:
    y.append(target_set2[key])
y = np.array(y)
x_data = np.linspace(0, 500, len(y))
#x_data = np.array(x_data)
plt.plot(x_data,y,'b-',lw=1)
```

Out[41]:

```
[<matplotlib.lines.Line2D at 0x63e635160>]
```



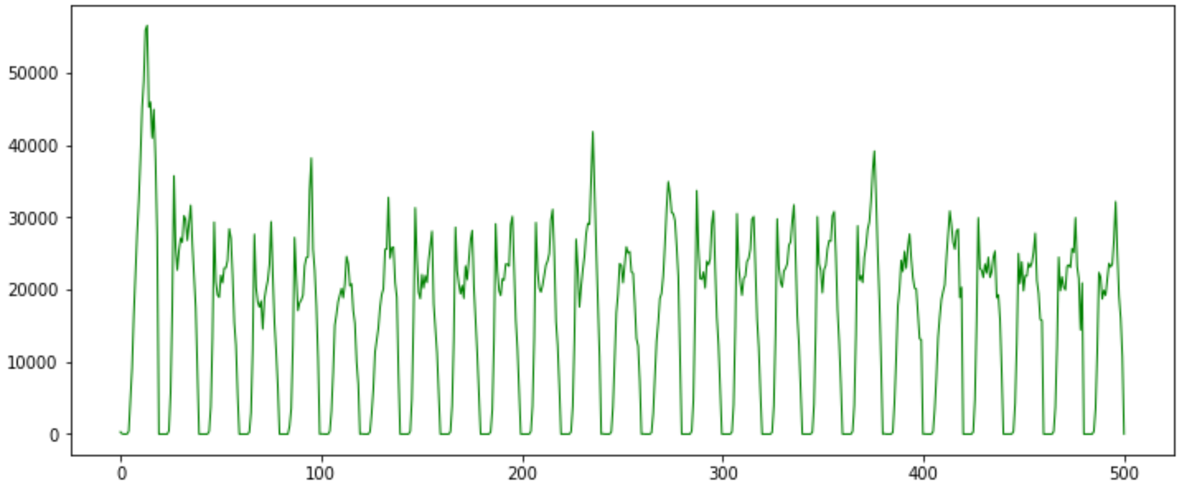
In [43]:

```
# Traversal data_set1 and Add Passenger-flow in target_set1 and tar
get_set2 with same date-hour
# Use data_set1 and target_set as Input and Output raw data
target_set = {}
for key in target_set1:
    if key in target_set2:
        target_set[key] = target_set1[key] + target_set2[key]
print(len(target_set))
```

```
12407
```

```
In [44]: """
         Plot hour_flow figure:
         """
         # 输出一天Passenger Flow / h 实际值  --> machine 2
         fig= plt.figure(figsize=(12,5))
         y = []
         x_data = list(target_set.keys())
         x = x_data[0:600]
         for key in x:
             y.append(target_set[key])
         y = np.array(y)
         x_data = np.linspace(0, 500, len(y))
         #x_data = np.array(x_data)
         plt.plot(x_data,y,'g-',lw=1)
```

```
Out[44]: [<matplotlib.lines.Line2D at 0x63e771f60>]
```

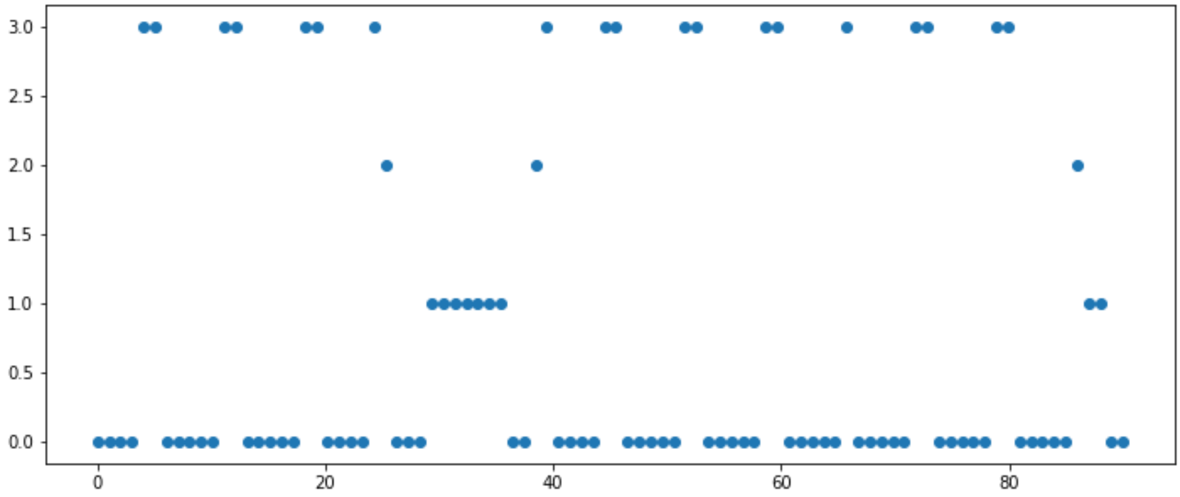


In [14]:

```
"""
Plot dict_holiday figure: 节假日情况分布
"""
# 输出一天Passenger Flow / h 实际值  --> machine 2
fig= plt.figure(figsize=(12,5))
y = []
x_data = list(dict_holiday.keys())
x = x_data[0:90]
for key in x:
    y.append(dict_holiday[key])
y = np.array(y)
x_data = np.linspace(0, 90, len(y))
#x_data = np.array(x_data)
plt.scatter(x_data,y)
```

Out[14]:

```
<matplotlib.collections.PathCollection at 0x63d8cf080>
```



In [ ]:

```
# 两个machine的数据不对称，先放弃machine2， 对machine1的数据进行分析
```

标准化：$X_t = \frac{X-mean}{std}$ 计算客流量累加值为/1 hour 的时段值

5.3 搭建神经网络预测模型

5.3.1 网络输入数据标准化

In [48]:

```
# data_set  为1h 流量  data_set1 --> target_set 按照日期对映
# dict_values -> obj -> list -> array
x1 = list(data_set1.values())
y1 = list(target_set1.values())
# pdb.set_trace()
data = np.array(x1)
target = np.array(y1)
# pdb.set_trace()
mean = data.mean(axis=0)
# pdb.set_trace()
std = data.std(axis = 0)
# 数组中的每一个element都会减mean
data = (data - mean) / std
# target_set = (target_set - mean_y) / std_y
```

In [49]:

```
print(len(target))
```

```
12619
```

In [50]:

```
# 两个machine的数据不对称，先放弃machine2， 对machine1的数据进行分析
print(data.shape)
print(target.shape)
print(len(target_set1))
print(len(target_set2))
```

```
(12619, 7)
(12619,)
12619
12757
```

5.3.2 搭建网络框架

In [51]:

```
def build_model():
    model = Sequential()
    # 先将维度放大， 在高维空间中抽取特征
    # Input: week, hour, date, weather
    model.add(Dense(64, input_shape = (7,),activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    # 输出偏差矩阵 correspond to train_target
    model.compile(optimizer = RMSprop(), loss="mse", metrics=['mae'
])
    return model
```

交叉检验，将数据分成n个集合 1 个验证， n-1个训练集 fold n = 5 / 10 求的模型的平均得分 防止一次训练产生过拟合 评价模型综合的情况 构造交叉检验的模型

In [52]:

```
num_epochs = 5000
all_scores = []
```

In [53]:

```
train_size = int(len(target_set1) * 0.9)
train_data = data[ : train_size]
#print(train_data)
train_target = target[ : train_size]
test_data = data[train_size : ]
test_target = target[train_size : ]
```

5.3.3 训练神经网络

In [ ]:

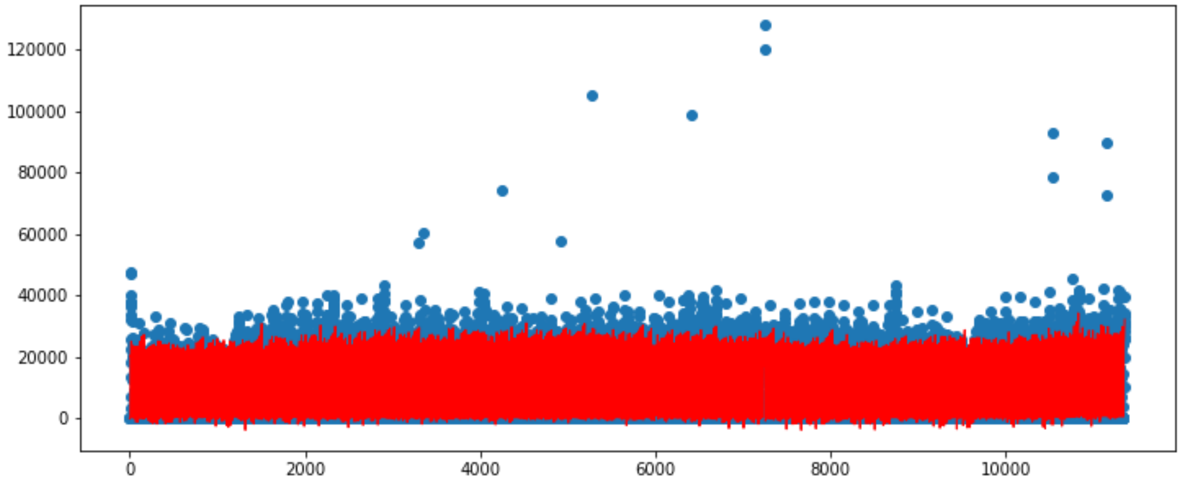
```
model = build_model()
# batch_size 训练进入的数据大小为1,  verbose = 0 不输出训练数据
model.fit(train_data, train_target, epochs = num_epochs, batch_size = 4000 , verbose = 2)
val_mse, val_mae = model.evaluate(test_data, test_target, verbose = 0)
all_scores.append(val_mae)
print("MSE:", val_mse,"MAE:",val_mae)
```

In [58]:

```
# 输出一个一天/h的训练值与实际值
fig = plt.figure(figsize=(12,5))
y_pred = model.predict(train_data)
x_data = np.linspace(0,len(train_data),len(y_pred))
plt.scatter(x_data,train_target)
plt.plot(x_data,y_pred,'r-',lw=1)
```

Out[58]:

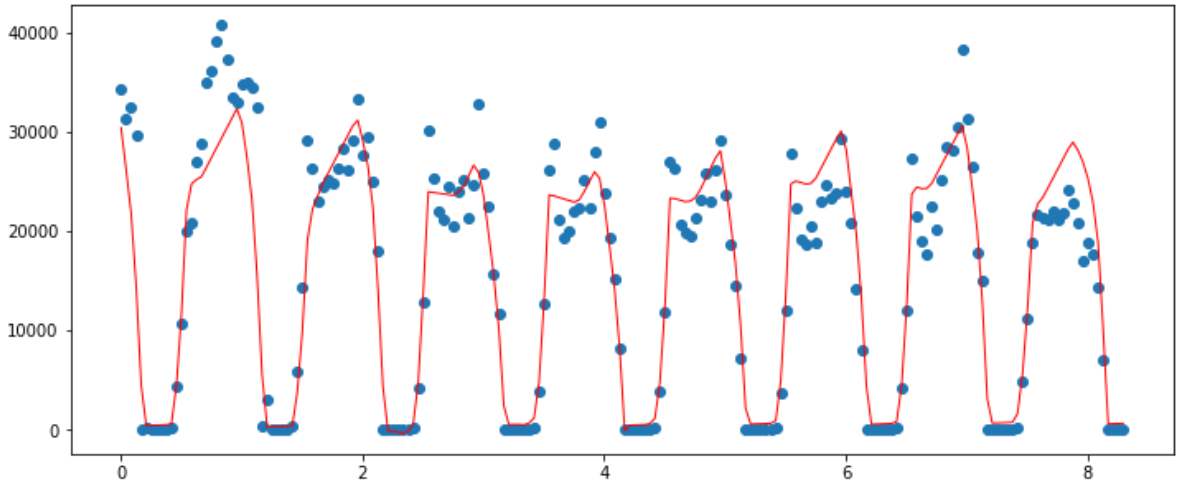
```
[<matplotlib.lines.Line2D at 0x63e6a4f98>]
```



5.3.4 预测结果

In [66]:
```
# 输出一个客流量 / h 的预测值与实际值
fig = plt.figure(figsize=(12, 5))
y_pred = model.predict(test_data)
x_data = np.linspace(0,len(y_pred)/24 ,len(y_pred))
plt.scatter(x_data[0:200],test_target[0:200])
plt.plot(x_data[0:200],y_pred[0:200],'r-',lw=1)
```

Out[66]:
```
[<matplotlib.lines.Line2D at 0x6402d5be0>]
```



6. 结论

通过对上海虹桥地铁进出站的客流量数据分析，发现客流量受到节假日，天气等因素的影响，本模型综合考虑以上影响因素，建立地铁客流量的预测模型。

情景分析结果表明，本预测算法能够有效的应用于地铁进出站的客流量预测情景。将来通过进一步对模型的改进和优化，能够支持公交、火车等交通方式的客流量预测。

届时将打造一个完整的大型交通枢纽客流量预测系统。为交通枢纽的运力决策和综合管理优化提供数据支持和政策建议。