廣東工業大學

《机器学习》课程项目报告

Predict future Sales

学　　院　信息工程学院

专　　业　电子信息

学　　号　2112003209

学生姓名　林泽帆

同组学生　李俊毅

硕士导师　黄永伟

2021 年 01 月

一、项目需求和数据集

这个挑战是 Coursera 课程“如何赢得数据科学竞赛”的最终项目。

使用具有挑战性的时间序列数据集，包括每日销售数据，由俄罗斯最大的软件公司之一 1C 公司提供。

我们要求你预测下个月所有产品和店铺的总销售额。训练集中给出了如下的范例，这是我截取数据集中的一部分数据

date	date_block_num	shop_id	item_id	item_price	item_cnt_day
02.01.2013	0	59	22154	999	1
03.01.2013	0	25	2552	899	1
05.01.2013	0	25	2552	899	-1
06.01.2013	0	25	2554	1709.05	1

图 1-训练数据集表

表格中的变量所代表的含义，项目要求也一并给出

ID- 代表测试集中的（商店，商品）元组的 ID

shop_id-商店的唯一标识符

item_id-产品的唯一标识符

item_cnt_day-销售的产品数量。您正在预测此量度的每月金额

item_price-商品的当前价格

日期 -格式为 dd / mm / yyyy 的日期

date_block_num-连续的月份号，为方便起见。2013 年 1 月为 0,2013 年 2 月为 1，...，2015 年 10 月为 33

二、数据处理

1. 读取数据集。

主要应用 pandas 库读取数据。

2. 数据可视化

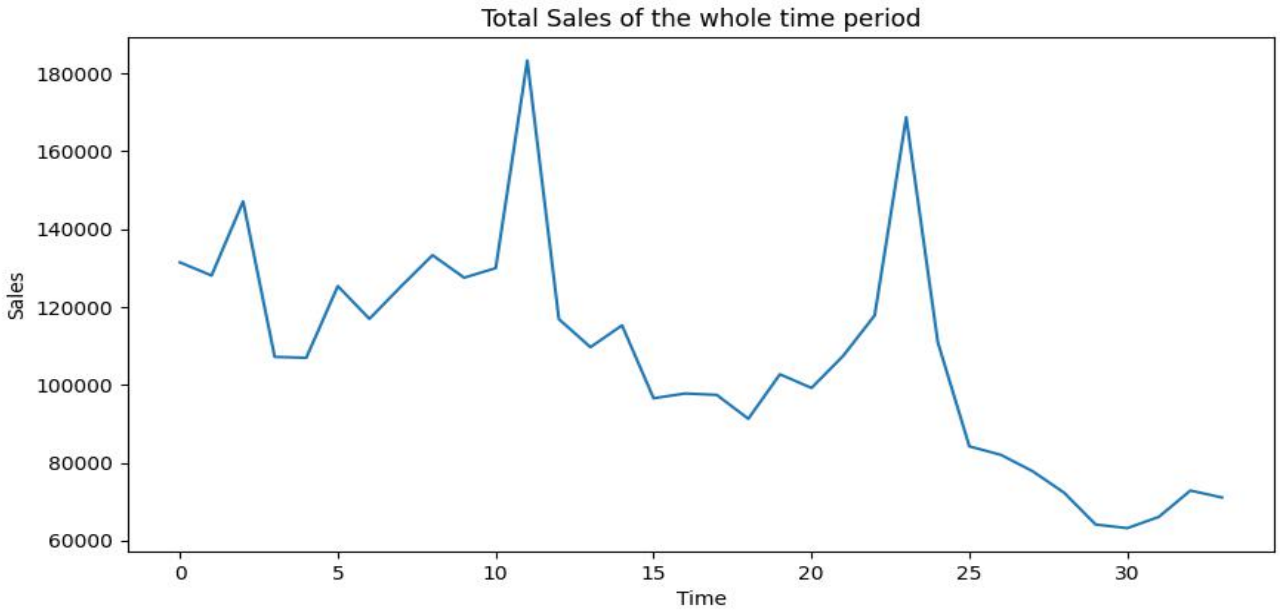


图 2-月份与总销售量的折线图

三、数据优化

1. 数据滤波（见附录）

数据集中会有一些和实际值相差很大的值，为避免出现过拟合，我们可以把这些离群值滤除掉；和测试集中的表格进行整合，删除测试集中没有的元素；然后生成数据透视表。

四、模型

1.Sequential 序贯模型:

序贯模型是函数式模型的简略版，为最简单的线性、从头到尾的结构顺序，不分叉，是多个网络层的线性堆叠

（1）指定输入数据的尺寸(shape)

（2）编译在训练模型之前，我们需要配置学习过程，这是通过 compile 方法完成的，他接收三个参数：

优化器 optimizer：adam

损失函数 loss：MSE 最小均方差函数。

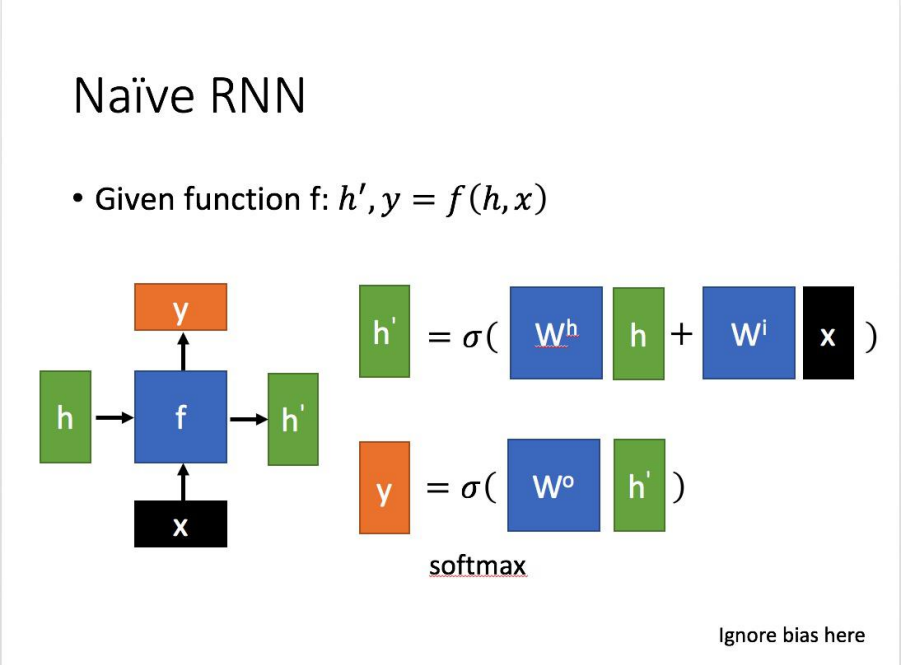
评估标准 metrics(精度值）：对于任何分类问题，你都希望将其设置为 metrics = ['mean_squared_error']

（3）Keras 模型在输入数据和标签的 Numpy 矩阵上进行训练。为了训练一个模型，你通常会使用 fit 函数

（4）训练完的模型放入测试集生成测试结果，通常回使用 predict 函数

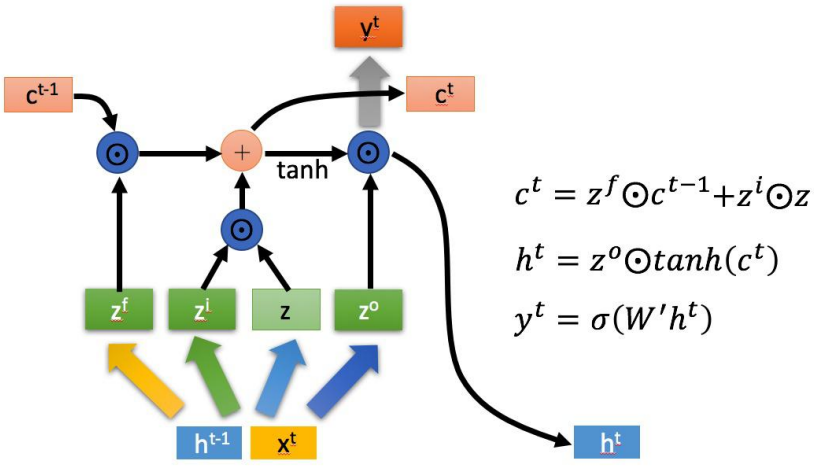
2.普通 RNN

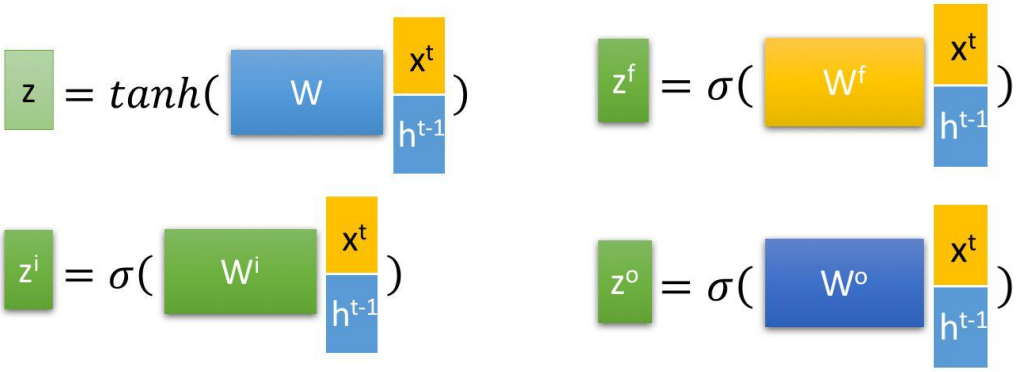
先简单介绍一下一般的 RNN。



3. LSTM

长短期记忆（Long short-term memory, LSTM）是一种特殊的 RNN，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。简单来说，就是相比普通的 RNN，LSTM 能够在更长的序列中有更好的表现。





LSTM 内部主要有三个阶段：

1. 忘记阶段。这个阶段主要是对上一个节点传进来的输入进行选择性忘记。简单来说就是会 “忘记不重要的，记住重要的”。

具体来说是通过计算得到的 z^f（f 表示 forget）来作为忘记门控，来控制上一个状态的 c^{t-1} 哪些需要留哪些需要忘。

2. 选择记忆阶段。这个阶段将这个阶段的输入有选择性地进行“记忆”。主要是会对输入 x^t 进行选择记忆。哪些重要则着重记录下来，哪些不重要，则少记一些。当前的输入内容由前面计算得到的 z 表示。而选择的门控信号则是由 z^i（i 代表 information）来进行控制。

将上面两步得到的结果相加，即可得到传输给下一个状态的 c^t。也就是上图中的第一个公式。

3. 输出阶段。这个阶段将决定哪些将会被当成当前状态的输出。主要是通过 z^0 来进行控制的。并且还对上一阶段得到的 c^0 进行了放缩（通过一个 tanh 激活函数进行变化）。

与普通 RNN 类似，输出 y^t 往往最终也是通过 h^t 变化得到。

五、附录（部分重要代码）

1.对滤波后的数据进行整理

```
# groupby(['A','B'])['C'].sum(),取出['A','B']组成新表格对 C 求和，
# 然后 reset_index():原先第一列前面重新加索引得到新表格（这里会保留原先的一列）
# Aggregate to monthly level the sales 按月计算销售额
monthly_sales_train = train.groupby(["date_block_num", "shop_id", "item_id"])[
    "date_block_num", "date", "item_price", "item_cnt_day"].agg(
    {"date_block_num": 'mean', "date": ["min", 'max'], "item_price": "mean",
"item_cnt_day": "sum"})

print(monthly_sales_train.head(5))
# print('\n')
```

			date_block_num	date		item_price	item_cnt_day
			mean	min	max	mean	sum
date_block_num	shop_id	item_id					
0	2	33	0	05.01.2013	05.01.2013	499.0	1.0
		482	0	16.01.2013	16.01.2013	3300.0	1.0
		491	0	09.01.2013	09.01.2013	600.0	1.0
		839	0	22.01.2013	22.01.2013	3300.0	1.0
		1007	0	11.01.2013	25.01.2013	449.0	3.0

```
# 将 item_cnt_day 列表化并重新进行索引
sales_data_flat = monthly_sales_train.item_cnt_day.apply(list).reset_index()
# 只保留有效的测试练数据（测试集所有列留下，on 是两个数据集中共有的列。 how='left' 左连接，将左边表格留下，把右边表格的列接上，右边在左边没有的值显示 NAN）
sales_data_flat = pd.merge(test, sales_data_flat, on=['item_id', 'shop_id'], how='left')
# na 值补上 0，inplace=True 代表在原对象上进行修改
sales_data_flat.fillna(0, inplace=True)
# 删除['shop_id', 'item_id']列
```

```python
sales_data_flat.drop(['shop_id', 'item_id'], inplace=True, axis=1)
# ID date_block_num sum
print(sales_data_flat.head(5))
# print('\n')
```

	ID	date_block_num	sum
0	0	20.0	1.0
1	0	22.0	1.0
2	0	23.0	2.0
3	0	24.0	2.0
4	0	28.0	1.0

```python
# 创建数据透视表.
# 行=每个商店+物品代码
# 列将按时间顺序输出
# 'ID'是测试集中某一商店卖的某一产品
pivoted_sales = sales_data_flat.pivot_table(index='ID', columns='date_block_num',
fill_value=0, aggfunc='sum')
print(pivoted_sales.head(10))
print('\n')


# 保留除最后一列以外的所有列,
# 最后一列作为训练集的输出
X_train = np.expand_dims(pivoted_sales.values[:, :-1], axis=2)
y_train = pivoted_sales.values[:, -1:]
# 删除第一列
X_test = np.expand_dims(pivoted_sales.values[:, 1:], axis=2)
```

sum

date_block_num / ID	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

date_block_num / ID	17.0	18.0	19.0	20.0	21.0	22.0	23.0	24.0	25.0	26.0	27.0	28.0	29.0	30.0	31.0	32.0	33.0
0	0	0	0	1	0	1	2	2	0	0	0	1	1	1	3	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	3	2	0	1	3	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	2	0	0	3	1	0	0	0	3	0	0	0	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2
7	0	0	0	0	0	1	5	0	0	1	0	0	1	0	0	0	0
8	0	0	0	0	0	5	28	9	2	3	2	2	4	3	2	3	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2.训练部分

```
sales_model = Sequential()
sales_model.add(LSTM(units=64, input_shape=(33, 1)))#
sales_model.add(Dropout(0.5))#正则化
sales_model.add(Dense(1))

sales_model.compile(loss='mse', optimizer='adam', metrics=['mean_squared_error'])
sales_model.summary()

sales_model.fit(X_train, y_train, batch_size=4096, epochs=100)

submission_output = sales_model.predict(X_test)  # 把预测出来的模型放入测试集进行训练
```