# 高级量化交易技术

闫涛
科技有限公司
北京 2021.05.08
{yt7589}@qq.com

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

# 第零篇深-度学习

# 第 1 章行情数据处理

Abstract

在本章中我们将通过 AKshare 库，获取 A 股分钟级行情数据，并将其进行预处理，变为深度学习可用的数据集。

## 1 行情数据处理概述

### 1.1 获取原始行情数据

我们首先通过 apps.fmts.ds.akshare_data_source.AkshareDataSource 获取原始的行情数据，-将其保存到 csv 文件中。如果存在该 csv 文件，则直接从该文件中读出数据并返回。数据格式为：

```
1 ......
2 ['2021-08-17 14:55:00', 30.339999999999996, 30.339999999999996,
     30.339999999999996, 30.339999999999996, 200.0]
3 ['2021-08-17 14:55:01', 30.339999999999996, 30.339999999999996,
     30.339999999999996, 30.339999999999996, 200.0]
4 ......
```

Listing 1: 行情数据格式

### 1.2 行情数据预处理

#### 1.2.1 价格折线图

我们以收盘价为例，收盘价的折线图绘制程序如下所示：

```
1     class OhlcvProcessor(object):
2     # 价格折线图模式
3     PCM_DATETIME = 1
4     PCM_TICK = 2
5
6     @staticmethod
7     def draw_close_price_curve(stock_symbol: str, mode=1) -> None:
8         '''
9         绘制收盘价折线图, 横轴为时间, 纵轴为收盘价
10        '''
11        data = AkshareDataSource.get_minute_bars(stock_symbol=
    stock_symbol)
12        x = [v[0] for v in data[0:1000]]
13        y = [v[4] for v in data[0:1000]]
14        if mode == OhlcvProcessor.PCM_DATETIME:
15            OhlcvProcessor._draw_date_price_curve(x, y)
16        else:
17            OhlcvProcessor._draw_tick_price_curve(y)
18
19    def _draw_date_price_curve(x: List, y: List) -> None:
```

```
20          x = [datetime.datetime.strptime(di, '%Y-%m-%d %H:%M:%S')
    for di in x]
21          fig, axes = plt.subplots(1, 1, figsize=(8, 4))
22          plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中
    文标签
23          plt.rcParams['axes.unicode_minus'] = False #用来正常显示负
    号
24          # 最大化绘图窗口
25          figmanager = plt.get_current_fig_manager()
26          figmanager.window.state('zoomed')     #最大化
27          # 绘制收盘价格折线图
28          axes.plot_date(x, np.array(y), '-', label='Net Worth')
29          # 设置横轴时间显示格式
30          axes.xaxis.set_major_formatter(DateFormatter('%Y-%m-%d %H
    :%M:%S'))
31          plt.gcf().autofmt_xdate()
32          # 显示图像
33          plt.show()
34
35      def _draw_tick_price_curve(y: List) -> None:
36          x = range(len(y))
37          fig, axes = plt.subplots(1, 1, figsize=(8, 4))
38          plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中
    文标签
39          plt.rcParams['axes.unicode_minus'] = False #用来正常显示负
    号
40          # 最大化绘图窗口
41          figmanager = plt.get_current_fig_manager()
42          figmanager.window.state('zoomed')     #最大化
43          # 绘制收盘价格折线图
44          plt.title('收盘价折线图')
45          axes.set_xlabel('时间刻度')
46          axes.set_ylabel('收盘价')
47          axes.plot(x, np.array(y), '-', label='Net Worth')
48          plt.show()
```

Listing 2: 收盘价折线图

代码解读如下所示：

- 第 3、4 行：定义收盘价曲线绘制方式，一种是横轴为时间，另一种横轴为行情序号；
- 第 6~10 行：定义收盘价绘制方法，参数为股票代码和绘制模式，缺省值为横轴为时间（以分钟为单位），这种模式的缺点是从上一日收盘到下一日开盘有较大的时间间隔；
- 第 11 行：获取分钟线行情数据，格式为：[[dateteime, open, high, low, close, volume]]；
- 第 19 行：以横轴为行情时间值绘制收盘价曲线；

- 第 20 行：将时间变为'2021-08-21 12:56:00' 格式的列表；
- 第 21 行：设置显示图形；
- 第 22 行：设置字体使 matplotlib 可以正确显示汉字；
- 第 23 行：使 matplotlib 可以显示负号；
- 第 24~26 行：使 matplotlib 绘图窗口最大化；
- 第 27、28 行：绘制收盘价时间曲线；
- 第 29~31 行：设置横坐标轴时间显示格式为'2021-08-21 12:56:00'，并自动调整为 45 度角倾斜，以节省显示空间；

图 1: 以时间为横轴的收盘价折线图

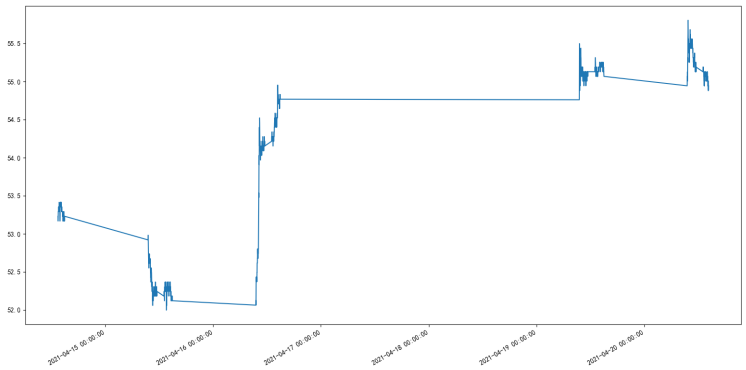
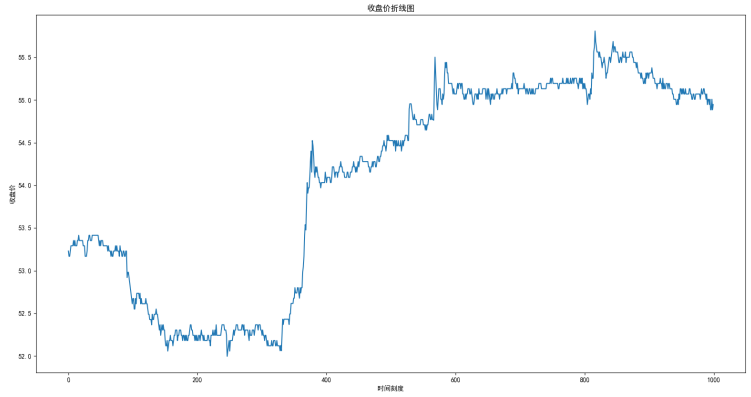


图 2: 以序号为横轴的收盘价折线图




如1所示，图中每天收盘到第二天开盘间没有行情数据，所以图形不太好看出规律，而图2则可以较好的反映价格的变化规律，因此我们在通常情况下，选择图2的形式。

### 1.2.2 对数差分序列

我们都知道，原始的行情数据，不具备平稳性，即无法通过历史数据来预测未来，而对数差分序列则具有平稳性，可以用来进行预测。如下所示：

```
1     @staticmethod
2     def gen_1d_log_diff_norm(stock_symbol, items):
3         '''
```

```
4         从原始行情数据，求出一阶对数收益率log(day2)-log(day1)，然
    后求出每列均值和标准差，利用
5         (x-mu)/std进行标准化，分别保存原始信息和归整后信息
6         参数:
7             stock_symbol 股票编号
8             items 由AkshareDataSource.get_minute_bars方法获取到
9         '''
10        datas = np.array([x[1:] for x in items])
11        log_ds = np.log(datas)
12        log_diff = np.diff(log_ds, n=1, axis=0)
13        log_diff_mu = np.mean(log_diff, axis=0)
14        log_diff_std = np.std(log_diff, axis=0)
15        ld_ds = (log_diff - log_diff_mu) / log_diff_std
16        # 保存原始信息
17        raw_file = './apps/fmts/data/{0}_1m_raw.txt'.format(
    stock_symbol)
18        with open(raw_file, 'w', encoding='utf-8') as fd:
19            for item in items[1:]:
20                fd.write('{0},{1},{2},{3},{4},{5}\n'.format(item
    [0], item[1],
21                                                            item[2], item[3], item[4], item[5]))
22        # 保存规整化后数据
23        ld_file = './apps/fmts/data/{0}_1m_ld.csv'.format(
    stock_symbol)
24        np.savetxt(ld_file, ld_ds)
25
26# 测试程序
27    def test_gen_1d_log_diff_norm_001(self):
28        stock_symbol = 'sh600260'
29        items = AkshareDataSource.get_minute_bars(stock_symbol=
    stock_symbol)
30        OhlcvProcessor.gen_1d_log_diff_norm(stock_symbol, items)
```

Listing 3: 收盘价折线图

代码解读如下所示：

- 第 2 行：items 由 AkshareDataSource.get_minute_bars 方法获取到，格式为 [..., ['2021-08-19 15:00:00', 1.1, 1.5, 1.0, 1.2, 1000], ...]；
- 第 10 行：把 items 中的条目，去除掉日期列后，生成 ndarray；
- 第 11 行：对所有元素取对数，以自然数 e 为底，np.log2 是以 2 底，np.log10 是以 10 为底；
- 第 12 行：取一阶差分，其中 n=1 代表是一阶差分，即后面一个元素减前面一个元素，out[i] = x[i+1]-x[i]，因为 axis=0，所以 i 代表行；
- 第 13 行：求出行方向的均值；
- 第 14 行：求出行方向的标准差；

- 第 15 行：进行归一化：$\hat{x} = \frac{x-\mu}{\sigma}$；
- 第 18 21 行：保存原始的行情信息，因为取了一阶差分，所以去掉了第 1 行；
- 第 23、24 行：保存一阶差分规整化后的数据；

### 1.3 数据集支撑数据

在每一个时间点，我们向前看 window_size 个时间点，缺省是 10 个，然后再加上当前时间点的数据：开盘、最高、最低、收盘、交易量，所以共有 $10 \times 5 + 5 = 55$ 个数据，我们的算法会根据这 55 维向量，判断出市场未来是上涨、下跌和震荡，进而根据当前持仓情况，采取不同的操作。我们先来看数据的生成：

## 2 最后

# 3 附录 X