凯读笔试题目程序文档

李想

2022 年 3 月 13 日

1 程序运行

核心类 `MovingAverage` 位于脚本 `kdma.py` 中。

脚本 `test.py` 中构造了一系列间隔不固定的时间戳和价格序列，用于测试该类。

请直接运行

```
$ python test.py
```

测试代码。

2 Notation

t_i：第 i 个数据点的时间戳

p_i：第 i 个数据点的价格

τ_i：第 i 个数据点与上个数据点的时间间隔

W：窗口长度

$w_{i|t=t_N}$：计算时间戳为 t_N 的数据点到达后的平均价格时，第 i 个数据点的权重

$\bar{p}_i$：第 i 个数据点到达后的平均价格

3 平均价格

3.1 数学表达

为使平均价格应能较好反映股票价格在窗口时间内的“平均水平”，程序中使用每个价格对应时间戳距离最新时间戳的距离为权重，计算平均价格。

记某个时刻共到达 N 条数据，时间戳与价格的数据对为 $\{(t_i, p_i)\}, i = 1, 2, \cdots, N$，则最新到达的数据时间戳为 t_N。窗口时间长度为 W，最新窗口期 $[t_N - W, W]$ 内共有 $K(K \geq 1)$ 条数据，数据对为 $\{(t_i, p_i)\}, i = N - K + 1, \ldots, N$。

在计算平均价格时，时刻 t_i 的权重

$$w_{i|t=t_N} = W - (t_N - t_i) = W + t_i - t_N, \quad i = N - K + 1, \ldots, N \tag{1}$$

时刻 t_N 的平均价格则可以表示为

$$\bar{p}_N = \frac{\sum_{i=N-K+1}^{N} p_i w_{i|t=t_N}}{\sum_{i=N-K+1}^{N} w_{i|t=t_N}} \tag{2}$$

3.2 权重更新

类 MovingAverage 中对权重是这样处理的：在暂时不考虑内存限制的情况（考虑内存限制下的权重更新方法参见5.2）下，在 t_N 时刻，第 i 条数据的权重

$$w_{i|t=t_N} = W + t_i - t_N \tag{3}$$

当 t_{N+1} 时刻的数据到达时，该权重应被更新为

$$w_{i|t=t_{N+1}} = W + t_i - t_{N+1} = W + t_i - t_N - (t_{N+1} - t_N) \tag{4}$$

$$= w_{i|t=t_N} - \tau_{N+1} \tag{5}$$

其中 $\tau_{N+1} = t_{N+1} - t_N$ 为两个时刻的时间间隔。

3.3 鲁棒性

由于相邻数据点之间的间隔不固定，所以在窗口期 $[t_N - W, W]$ 内，可能存在多个数据点，也可能只有一个数据点（即最新的数据 $\{(t_N, p_N)\}$）。式(2)对上述两种情况无差别对待，均可计算出合理的平均价格。

4 不考虑内存限制的算法

当内存没有任何限制时，我们可以将窗口期内任意多的数据点都记录下来，可以精确地计算式(2)所表达的平均价格。具体算法见算法1：

5 考虑内存限制的算法

当内存存在限制时，可能会出现无法将窗口期内所有数据点都记录下来的情况，这时式(2)将无法直接适用。

解决方案是：**当最新数据到达时，在计算最新平均价格之前，若最新窗口期内的数据点数量已经达到内存上限时，找出上一个窗口期内所有数据点中时间间隔最小的两个数据点，将二者的信息“合并”存储，再将最新数据存入，计算最新平均价格**。

5.1 “合并”存储

对于任意两个相邻的数据点 $(t_{i-1}, \tau_{i-1}, p_{i-1}, w_{i-1}), (t_i, \tau_i, p_i, w_i)$，合并后的权重为二者权重之和，价格为二者价格的加权平均，且合并后沿用后一时刻的时间数据（可以理解为把“合并”后的数据存储在后一时刻）

$$t'_i = t_i \tag{6}$$

$$\tau'_i = \tau_{i-1} + \tau_i \tag{7}$$

$$w'_i = w_{i-1} + w_i \tag{8}$$

$$p'_i = \frac{p_{i-1}w_{i-1} + p_i w_i}{w'_i} \tag{9}$$

为什么选择把“合并”后的数据点存储在后一时刻而非前一时刻或中间某时刻，是考虑到若不存储在后一时刻，那么当窗口向后滚动时，很有可能会将这个“合并”的数据点直接排除在新窗口之外，丢失信息导致计算偏差变大。

Algorithm 1 不考虑内存限制的算法

Input:

时间戳与价格的数据对时间序列 $\{(t_i, p_i)\}, i = 1, 2, \ldots, N.$

窗口期长度 W

Initialize:

最新数据的时间戳 $t = 0$

窗口期内数据点的个数 $n = 0$

窗口期内数据点的时间戳列表 $\mathrm{L}_t = [\,]$

窗口期内数据点的时间间隔列表 $\mathrm{L}_\tau = [\,]$

窗口期内数据点的时间间隔之和 $S_\tau = 0$

窗口期内数据点的价格列表 $\mathrm{L}_p = [\,]$

窗口期内数据点的权重列表 $\mathrm{L}_w = [\,]$

for i in $\{1, 2, \ldots, N\}$ **do**
  $\tau = t - t_i,\ t = t_i$
  while $n > 0$ and $S_\tau + \tau > W$ **do**
    按先进先出法弹出 $\mathrm{L}_t, \mathrm{L}_\tau, \mathrm{L}_p, \mathrm{L}_w$ 各列表中的第一个元素，记为 t_0, τ_0, p_0, w_0
    $S_\tau = S_\tau - \tau_0$
    $n = n - 1$
  end while
  将 L_w 中各元素分别减去 τ
  将 t_i, τ, p_i, W 分别存入 $\mathrm{L}_t, \mathrm{L}_\tau, \mathrm{L}_p, \mathrm{L}_w$ 各列表尾部
  $S_\tau = S_\tau + \tau$
  $n = n + 1$
  $\bar{p}_i = \mathrm{sum}(\mathrm{L}_p \cdot \mathrm{L}_w)/\mathrm{sum}(\mathrm{L}_w)$
end for
Output: 平均价格序列 $\{\bar{p}_i\}, i = 1, 2, \ldots, N$

如此，如果这两个数据点的时间间隔足够小，则可以将“合并”后的数据点视为一个正常数据点，用于计算平均价格。

同理，还可以会出现需要“合并”已被“合并”数据点的情况，但只要是两个数据点间隔足够小，近似程度仍会比较高。

5.2 内存限制时权重的更新

3.2部分介绍了在没有内存限制时如何在新数据到达时对权重进行更新。当存在内存限制时，因为要“合并”数据，所以对已经被“合并”的数据，式(4)不再适用。但此式仍可以给我们一些启发：如果某个数据点中的信息只有一个原始数据点，那么更新权重时要减去一个 τ。如果某个“合并”数据点中存有 k 个原始数据点的信息，那么更新权重时只要减去 k 个 τ 就可以了。

按上述思路，在循环计算平均价格过程中还需要记录“合并”次数这一信息。

5.3 算法

考虑有内存限制时的算法见算法2：

Algorithm 2 考虑内存限制的算法

Input:

时间戳与价格的数据对时间序列 $\{(t_i, p_i)\}, i = 1, 2, \ldots, N$.

窗口期长度 W

内存上限 n_B

Initialize:

最新数据的时间戳 $t = 0$

窗口期内数据点的个数 $n = 0$

窗口期内数据点的时间戳列表 $\mathrm{L}_t = [\,]$

窗口期内数据点的时间间隔列表 $\mathrm{L}_\tau = [\,]$

窗口期内数据点的时间间隔之和 $S_\tau = 0$

窗口期内数据点的价格列表 $\mathrm{L}_p = [\,]$

窗口期内数据点的权重列表 $\mathrm{L}_w = [\,]$

窗口期内数据点的“合并”次数列表 $\mathrm{L}_c = [\,]$

for i in $\{1, 2, \ldots, N\}$ **do**
    $\tau = t - t_i,\ t = t_i$
    while $n > 0$ and $S_\tau + \tau > W$ **do**
        按先进先出法弹出 $\mathrm{L}_t, \mathrm{L}_\tau, \mathrm{L}_p, \mathrm{L}_w, \mathrm{L}_c$ 各列表中的第一个元素，记为 $t_0, \tau_0, p_0, w_0, c_0$
        $S_\tau = S_\tau - \tau_0$
        $n = n - 1$
    end while
    if $n = n_B$ **then**
        $j = (\mathrm{argmin}\ \mathrm{L}_\tau) + 1$
        弹出 $\mathrm{L}_t, \mathrm{L}_\tau, \mathrm{L}_p, \mathrm{L}_w, \mathrm{L}_c$ 各列表中的第 j 个元素，记为 $t_j, \tau_j, p_j, w_j, c_j$
        $n = n - 1$
        $\mathrm{L}_t[j-1] = t_j$
        $\mathrm{L}_\tau[j-1] = \mathrm{L}_\tau[j-1] + \tau_j$
        $w' = \mathrm{L}_w[j-1] + w_j$
        $p' = \frac{\mathrm{L}_p[j-1]\mathrm{L}_w[j-1] + p_j w_j}{w'}$
        $\mathrm{L}_w[j-1] = w'$
        $\mathrm{L}_p[j-1] = p'$
        $\mathrm{L}_c[j-1] = \mathrm{L}_c[j-1] + c_j$
    end if
    $\mathrm{L}_w[k] = \mathrm{L}_w[k] - \tau \mathrm{L}_c[k], k = 1, 2, \ldots, i$
    将 $t_i, \tau, p_i, W, 1$ 分别存入 $\mathrm{L}_t, \mathrm{L}_\tau, \mathrm{L}_p, \mathrm{L}_w, \mathrm{L}_c$ 各列表尾部
    $S_\tau = S_\tau + \tau$
    $n = n + 1$
    $\bar{p}_i = \mathrm{sum}(\mathrm{L}_p \cdot \mathrm{L}_w) / \mathrm{sum}(\mathrm{L}_w)$
end for
Output: 平均价格序列 $\{\bar{p}_i\}, i = 1, 2, \ldots, N$