A Note on Efficient Updates for Order Statisitcs on Moving Windows.

September 4, 2021

1 Background

Efficiently calculating sequential updates to basic summary statistics is important when an algorithm or calculation traverses a data set in some way. For example, when a sliding window is used to process a large data set or analyse a stream of sensor data in real time. Importantly, for real time streamed data, a guaranteed maximum latency is a common requirement and, as such, it is the worst case complexity of the update that is significant. Consequently, in what follows, all measures of complexity are worst case unless otherwise stated.

When windowing is done using a linear data structure with $\mathcal{O}(1)$ push and pop operations, such as a deque or circular buffer, the mean and variance can both be calculated in $\mathcal{O}(1)$ time. In contrast, order statistics, such as quantiles, require algorithms with $\mathcal{O}(n)$ complexity. This situation can be improved by employing self balancing Binary Search Trees (BSTs), which support $\mathcal{O}(\log n)$ insertion, removal, and bisection. The resulting update complexity for quantiles is then $\mathcal{O}(\log n)$, although the linear data structure has to be retained to maintain the removal priority of the elements during an update.

Typically, when determining n quantiles, $n + 1$ BSTs are required since indexing for a BST is an $\mathcal{O}(n)$ operation. This situation can be simplified by using an Ordered Statistics Tree (OST) which provides an $\mathcal{O}(\log n)$ indexing operation [cite algorithms book here]. When this is done, a single tree can be used to dynamically determine an arbitary number of quantiles. Furthermore, using an OST also allows the rank of an arbitary element to be determined in $\mathcal{O}(\log n)$ time which, in principle, admits the efficient calculation of a wide range of useful rank based statistics.

This article demonstrates how the combination of a linear sequence and an OST can be synchronised to provide a new Abstract Data Type (ADT). The resulting ADT, referred to here as an Ordered Statistics Window (OSW), supports operations suitable for the efficient sequential update of order and rank based statistics on a sliding window. With view to motivating the application of OSWs in practice, two algorithms which employ them are developed and examined in detail.

2 An Orderded Statistics Window ADT

In the context of efficiently determining order statistics on a moving window the following operations are pertinent.

update($\mathcal{W}$,x) Adds x to the window $\mathcal{W}$ and, if the size of the resulting window is greater than a pre-specified maximum, removes the oldest entry y from $\mathcal{W}$. It returns the ordered pair (i, y) where i is the rank of x in $\mathcal{W}$. In the case of no element being removed $y =$ none.

bisect($\mathcal{W}$,x) Returns an ordered pair (i, y) where i is the rank that x would have if $\mathcal{W}$ were updated with x, and y is the smallest element in the window such that $x \leq y$. If no such y exists then $y =$ none.

select($\mathcal{W}$,i) Returns the value of the element with rank i if such an element exists or none otherwise.

Henceforth, a data structure supporting the above operations will be refered to as an Ordered Statistics Window (OSW). Table 2 shows the worst case computational complexity for an OSW when implemented using some common data structures in isolation. Assume now that an algorithm $\mathcal{A}$ on an OSW $\mathcal{W}$ will

	update$(\mathcal{W}, x)$	bisect$(\mathcal{W}, x)$	select$(\mathcal{W}, x)$
array	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
deque	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
red-black tree	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$
ordered statistics tree	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

Table 1: Computational complexity of an Ordered Statistics Window when implemented using some common data structures.

require and update and at least one other operation, then the table shows the least worst case complexity of $\mathcal{A}$ to be $\mathcal{O}(n)$. On the other hand, it also suggests the possibility of a composite data structure consisting of a deque $\mathcal{D}$ and an OST $\mathcal{T}$ that could reduce the best worst case complexity of $\mathcal{A}$ to $\mathcal{O}(\log n)$.

Let an OSW consist of a deque $\mathcal{W}.\mathcal{D}$, an OST $\mathcal{W}.\mathcal{T}$, and an positive integer $\mathcal{W}.n$ indicating the maximum size $\#\mathcal{W}$ of $\mathcal{W}$.

3 Median Absolute Deviation

The Median Absolute Devitaion (MAD) is a robust measure of scale defined as

$$\text{mad}(X) = \text{med}(|x_i - \widetilde{X}|) \tag{1}$$

where the outer median is taken over all $x_i \in X$.

Algorithm 1 Median Absolute Deviation of a Sorted Container T via Bisection

function MAD(T)
  if $\#T = 1$ **then**
    return 0
  function BISECT($T, (a, b), (c, d)$)
    $\widetilde{T} \leftarrow \frac{1}{2}(T_{\lceil \#T/2 \rceil - 1} + T_{\lfloor \#T/2 \rfloor})$
    if $b - a < 2$ **then**
      if $\#T \% 2 = 0$ **then**
        return $\frac{1}{2}(\max(\widetilde{T} - T_b, T_c - \widetilde{T}) + \min(\widetilde{T} - T_a, T_b - \widetilde{T}))$
      else
        return $\min(\max(\widetilde{T} - T_b, T_c - \widetilde{T}), \min(T_a - \widetilde{T}, T_b - \widetilde{T}))$
    if $4\widetilde{T} > T_a + T_b + T_c + T_d$ **then**
      return BISECT$(T, (a + \left\lfloor \frac{b-a}{2} \right\rfloor, b), (c + \left\lfloor \frac{d-c}{2} \right\rfloor, d))$
    else
      return BISECT$(T, (a, a + \left\lceil \frac{b-a}{2} \right\rceil), (c, c + \left\lceil \frac{d-c}{2} \right\rceil))$
  return BISECT$(T, (0, \lfloor \#T/2 \rfloor - 1), (\lceil \#T/2 \rceil, \#T - 1))$

3.1 Algorithm

4 Two Sample Cramér-von Mises test.

5 Walsh's Outlier Test

6 Dean and Dixon Test

7 Pearson and Hartley Test

8 Nalimov Test

9 Kruskal-Wallis Test