Simple Ordered Statistic CFAR Algorithm

MIT LL Group 102

September 5, 2018

Consider the following input:

- A mask, $M(k,l)$, for selecting elements from an array, where k and l are positive integers, and M is defined as an array of Boolean values represented as 0 and 1 : $M = \{s_1, ..., s_k, t_{2l-1}, s_1, ..., s_k\}$, where $s_i = 1$ for any i from 1 to k and $t_j = 0$ for any j from 1 to $2l - 1$.

  For example, $M(2,1) = \{1,1,0,1,1\},\ M(3,2) = \{1,1,1,0,0,0,1,1,1\}$

- An array A of n numeric values: $A = \{a_1, ..., a_n\}$
- A real value p between 0 and 1

The output of the algorithm is an array B with the same size as A and containing a subset (with repetitions allowed) of the original array A. The algorithm is described as follows:

> The mask is applied successively, centered on each element of the input array, A, in order to select the elements corresponding to the 1s. If elements could not be selected due to boundary conditions (i.e., where the mask M runs off the edges of A or does not cover certain elements of A, in other words, does not fully overlap A) they will be ignored.

More formally, if a_i is an element of A, the set of selected elements will be:

$$S_i = \left\{a_m \mid i > l,\ m \geq \max(1, i-l-k+1),\ m \leq i-l\right\} \cup \left\{a_m \mid i \leq n-l,\ m \geq i+l,\ m \leq \min(n, i+l+k-1)\right\}$$

For instance, assume:

$$A = \{a_1, a_2, a_3, a_4, a_5\},\ M(2,1) = \{1,1,0,1,1\},\ \text{and } p = \frac{3}{4}$$

With these inputs, the algorithm proceeds as follows:

1. For each element a_i to which the mask is applied, the following subsets of elements are selected:
   - $a_1 : S_1 = \{s_{1,1}, s_{1,2}\} = \{a_2, a_3\}$
   - $a_2 : S_2 = \{s_{2,1}, s_{2,2}, s_{2,3}\} = \{a_1, a_3, a_4\}$
   - $a_3 : S_3 = \{s_{3,1}, s_{3,2}, s_{3,3}, s_{3,4}\} = \{a_1, a_2, a_4, a_5\}$
   - $a_4 : S_4 = \{s_{4,1}, s_{4,2}, s_{4,3}\} = \{a_2, a_3, a_5\}$
   - $a_5 : S_5 = \{s_{5,1}, s_{5,2}\} = \{a_3, a_4\}$
2. For each of these selected sets, S_i, where $i = \{1, ..., n\}$, choose the element $s_{i,j}$ such that $\lfloor p * \text{length}(S_i) \rfloor$ elements from the set are no greater than $s_{i,j}$, and the rest are no less than $s_{i,j}$.
3. Set element B_i in output array B equal to the chosen $s_{i,j}$.

For example, assume that the elements in array A are sorted in increasing order. The output of the algorithm is then array $B = \{a_2, a_3, a_4, a_3, a_3\}$.

Task: Devise an efficient implementation of the algorithm and populate the `run()` method within the provided software package. The software package will verify the correct implementation using known input and output pairs. *Hint: copying memory is expensive, as is sorting.*