The Frequent Items Problem in Online Streaming under Various Performance Measures

Joan Boyar, Kim S. Larsen, and Abyayananda Maiti
University of Southern Denmark, Odense, Denmark
{joan,kslarsen,abyaym}@imada.sdu.dk

Abstract. In this paper, we strengthen the competitive analysis results obtained for a fundamental online streaming problem, the Frequent Items Problem. Additionally, we contribute with a more detailed analysis of this problem, using alternative performance measures, supplementing the insight gained from competitive analysis. The results also contribute to the general study of performance measures for online algorithms. It has long been known that competitive analysis suffers from drawbacks in certain situations, and many alternative measures have been proposed. However, more systematic comparative studies of performance measures have been initiated recently, and we continue this work, using competitive analysis, relative interval analysis, and relative worst order analysis on the Frequent Items Problem.

1 Introduction

The analysis of problems and algorithms for streaming applications, treating them as online problems, was started in [2]. In online streaming, the items must be processed one at a time by the algorithm, making some irrevocable decision with each item. A fixed amount of resources is assumed. In the frequent items problem [12], an algorithm must store an item, or more generally a number of items, in a buffer, and the objective is to store the items appearing most frequently in the entire stream. This problem has been studied in [15]. In addition to probabilistic considerations, they analyzed deterministic algorithms using competitive analysis. We analyze the frequent items problem using relative interval analysis [14] and relative worst order analysis [4]. In addition, we tighten the competitive analysis [17,16] results from [15].

It has been known since the start of the area that competitive analysis does not always give good results [17] and many alternatives have been proposed. However, as a general rule, these alternatives have been fairly problem specific and most have only been compared to competitive analysis. A more comprehensive study of a larger number of performance measures on the same problem

* Supported in part by the Danish Council for Independent Research. Part of this work was done while the authors were visiting the University of Waterloo.
scenarios was initiated in [8] and this line of work has been continued in [9,6,7]. With this in mind, we would like to produce complete and tight results, and for that reason, we focus on a fairly simple combinatorial problem and on simple algorithms for its solution, incorporating greediness and adaptability trade-offs to a varying extent.

Finally, we formalize a notion of competitive function, as opposed to competitive ratio, in a manner which allows us to focus on the constant in front of the high order term. These ideas are also used to generalize relative worst order analysis.

Most proofs have been omitted due to space restrictions. These can be found in the full version of the paper [10].

2 Preliminaries

This is a streaming problem, but as usual in online algorithms we use the term sequence or input sequence to refer to a stream. We denote an input sequence by $I = a_1, a_2, \ldots, a_n$, where the items $a_i$ are from some universe $\mathcal{U}$, assumed to be much larger than $n$. We may refer to the index also as the time step. We consider online algorithms, which means that items are given one by one.

We consider the simplest possible frequent items problem: An algorithm has a buffer with space for one item. When processing an item, the algorithm can either discard the item or replace the item in the buffer by the item being processed. The objective is to keep the most frequently occurring items in the buffer, where frequency is measured over the entire input, i.e., when an algorithm must make a decision, the quality of the decision also depends on items not yet revealed to the algorithm. We define this objective function formally:

Given an online algorithm $\mathcal{A}$ for this problem, we let $s_t^A$ denote the item in the buffer at time step $t$. We may omit the superscript when it is clear from the context which algorithm we discuss.

Given an input sequence $I$ and an item $a \in \mathcal{U}$, the frequency of the item is defined as $f_I(a) = \frac{n_I(a)}{n}$, where $n_I(a) = |\{i \mid a_i = a\}|$ is the number of occurrences of $a$ in $I$. The objective is to maximize the aggregate frequency [15], defined by $F_A(I) = \sum_{t=1}^{n} f_I(s_t^A)$, i.e., the sum of the frequencies of the items stored in the buffer over the time.

We compare the quality of the achieved aggregate frequencies of three different deterministic online algorithms from [15]: the naive algorithm (Nai), the eager algorithm (Eag), and the majority algorithm (Maj). All three are practical streaming algorithms, being simple and using very little extra space.

**Definition 1. [Nai]** Nai buffers every item as it arrives, i.e., $s_t^{Nai} = a_t$ for all $t = 1, 2, \ldots, n$.

The algorithm EAG switches mode upon detecting a repeated item, an item which occurs in two consecutive time steps.