Journal of Machine Learning Research 18 (2018) 1-39 Submitted 8/2016; Revised 1/2018; Published 6/2018

Interactive Algorithms: Pool, Stream and Precognitive Stream

Sivan Sabato SABATOS@CS.BGU.AC.IL
Tom Hess TOMHE@POST.BGU.AC.IL
Department of Computer Science
Ben-Gurion University of the Negev
Beer Sheva 8410501, Israel.

Editor: Csaba Szepesvari

Abstract

We consider interactive algorithms in the pool-based setting, and in the stream-based setting. Interactive algorithms observe suggested elements (representing actions or queries), and interactively select some of them and receive responses. Pool-based algorithms can select elements at any order, while stream-based algorithms observe elements in sequence, and can only select elements immediately after observing them. We further consider an intermediate setting, which we term *precognitive stream*, in which the algorithm knows in advance the identity of all the elements in the sequence, but can select them only in the order of their appearance. For all settings, we assume that the suggested elements are generated independently from some source distribution, and ask what is the stream size required for emulating a pool algorithm with a given pool size, in the stream-based setting and in the precognitive stream setting. We provide algorithms and matching lower bounds for general pool algorithms, and for utility-based pool algorithms. We further derive nearly matching upper and lower bounds on the gap between the two settings for the special case of active learning for binary classification.

Keywords: interactive algorithms, active learning, pool-based, stream-based

1. Introduction

Interactive algorithms are algorithms which are presented with input in the form of suggested elements (representing actions or queries), and iteratively select elements, getting a response for each selected element. The reward of the algorithm, which is application-specific, is a function of the final set of selected elements along with their responses. Interactive algorithms are used in many application domains, including, for instance, active learning (McCallum and Nigam, 1998), interactive sensor placement (Golovin and Krause, 2011), summarization (Singla et al., 2016) and promotion in social networks (Guillory and Bilmes, 2010). As a specific motivating example, consider an application in which elements represent web users, and the algorithm should select up to q users to present with a free promotional item. For each selected user, the response is the observed behavior of the user after having received the promotion, such as the next link that the user clicked on. The final reward of the algorithm depends on the total amount of promotional impact it obtained, as measured by some function of the set of selected users and their observed responses. Note that the algorithm can use responses from previous selected users when deciding on the next user to select.

We consider two main interaction settings for interactive algorithms: The *pool-based* setting and the *stream-based* setting. In the pool-based setting, the entire set of suggested elements is provided in advance to the algorithm, which can then select any of the elements at any order. For

instance, in the web promotion example, there might be a set of users who use the website for an extended period of time, and any of them can be approached with a promotion. In the stream-based setting, elements are presented to the algorithm in sequence, and the algorithm must decide immediately after observing an element, whether to select it or not. In the web promotion example, this is consistent with a setting where users access the website for single-page sessions, and so any promotion must be decided on immediately when the user is observed. Note that in the stream-based setting considered in this work, the only direct restriction is on the timing of selecting elements. We do not place restrictions on storage space or other resources.

The stream-based setting is in general weaker than the pool-based setting. Nonetheless, it is important and useful: In many real-life scenarios, it is not possible to postpone selection of elements, for instance due to storage and retrieval constraints, or because of timing constraints. This is especially pertinent when the data stream is real-time in nature, such as in streaming document classification (Bouguelia et al., 2013), in spam filtering (Chu et al., 2011), in web streams such as Twitter (Smailović et al., 2014), in video surveillance (Loy et al., 2012) and with active sensors (Krishnamurthy, 2002).

In this work, our goal is to study the relationship between these two important settings: the pool-based setting and the stream-based setting. Both of these settings have been widely studied in many contexts. In active learning, both the pool-based and the stream-based setting have been studied in classic works (Cohn et al., 1994; Lewis and Gale, 1994). Works that address mainly the stream-based setting include, for instance, Balcan et al. (2009); Hanneke (2011); Dasgupta (2012); Balcan and Long (2013); Sabato and Munos (2014). Some theoretical results hold for both the stream-based and the pool-based settings (e.g., Balcan and Long, 2013; Hanneke and Yang, 2015). Several near-optimal algorithms have been developed for the pool-based setting (Dasgupta, 2005; Golovin and Krause, 2011; Golovin et al., 2010b; Hanneke, 2007; Sabato et al., 2013; Gonen et al., 2013; Cuong et al., 2014). The pool-based setting is also heavily studied in various active learning applications (e.g., Tong and Koller, 2002; Tong and Chang, 2001; Mitra et al., 2004; Gosselin and Cord, 2008; Cebron and Berthold, 2009; Guo et al., 2013). General interactive algorithms have also been studied in both a pool-based setting (e.g., Golovin and Krause, 2011; Guillory and Bilmes, 2010; Deshpande et al., 2014) and in stream-based settings (e.g., Demaine et al., 2014; Arlotto et al., 2016; Streeter and Golovin, 2009; Golovin et al., 2010a).

To facilitate this study, we introduce a third setting, which we term the *precognitive stream-based* setting. This is an intermediate setting, which is weaker than the pool-based setting but stronger than the stream-based setting. In the precognitive setting, as in the standard stream-based setting, there is a sequence of elements and the algorithm must make a decision immediately after the item is presented to it. However, unlike the standard stream-based setting, in the precognitive setting the algorithm knows in advance the identity of items that will be presented in the future. This intermediate setting is of interest, since it has only one of the disadvantages of the standard stream-based setting: the requirement that elements are selected in order of their presentation in the sequence. On the other hand, it does not have the second disadvantage: the lack of knowledge of future possible items. Thus, studying this setting allows distinguishing the effect of the two issues on the performance of stream-based algorithms.

To study the relationship between the pool-based setting and the stream-based setting, as well as its precognitive variant, we assume that in all settings the suggested elements, along with their hidden responses, are drawn i.i.d. from some unknown source distribution. We then ask under what conditions, and at what cost, can a stream-based algorithm obtain the same output distribution as

a given black-box pool algorithm. Such an exact emulation is advantageous, as it allows direct application of methods and results developed for the pool-based setting, in the stream-based setting. Especially, if a pool-based algorithm succeeds in practice, but its analysis is unknown or limited, exact emulation guarantees that this success is reproduced in the streaming setting as well.

For discrete source distributions, any pool-based algorithm can be emulated in a stream-based setting, simply by waiting long enough, until the desired element shows up again. The challenge for stream-based interactive algorithms is thus to achieve the same output distribution as a pool-based algorithm, while observing as few suggested elements as possible. Clearly, there are many cases in which it is desirable to require less suggested elements, as this could save resources such as time, money, and communication. In active learning as well, while examples are usually assumed cheap, they are not usually completely free in all respects.

We study emulation of pool-based algorithms in two regimes. First, we consider the fully general case, of emulating some unrestricted pool algorithm. We provide a stream algorithm that can emulate any given black-box pool algorithm, and uses a uniformly bounded expected number of observed elements. The bound on the expected number of observed elements is exponential in the number of selected elements. We further prove a lower bound which indicates that this exponential dependence is necessary. We also study the precognitive stream-based setting and show the following: On the one hand, it can require a significantly smaller number of observed elements than the standard stream-based setting. On the other hand, its worst-case performance is also exponential in the number of selected elements, similarly to the stream-based setting. We conclude that while knowing the sequence in advance can be helpful, it does not improve the general performance of stream-based emulation.

Second, we consider *utility-based* interactive algorithm for the pool setting. We provide a stream algorithm that emulates such pool algorithms, using repeated careful applications of solutions of the well known "Secretary Problem" (Dynkin, 1963; Gilbert and Mosteller, 1966; Ferguson, 1989). The expected number of observed elements for this algorithm is only linear in the number of selected elements. In this case too we prove matching lower bounds. These results hold also for the precognitive stream setting. Our analysis shows a tradeoff between the number of observed elements and the number of selected elements, in cases where the stream algorithm is allowed to select extra elements over what the pool-algorithm selects.

Finally, we consider the special case of active learning for binary classification. We give nearly-matching upper and lower bounds for stream emulation in this setting. From the lower bound, we conclude that even in this well-studied setting, there are cases in which there exists a significant gap between the best pool-based algorithm and the best stream-based algorithm. This result generalizes a previous observation of Gonen et al. (2013) on the sub-optimality of CAL (Cohn et al., 1994), the classical stream-based active learning algorithm, compared to pool algorithms.

This paper is structured as follows: In Section 2 formal definitions and notations are provided. Section 3 discusses natural but suboptimal solutions. Section 4 considers emulating general pool algorithms, and Section 5 addresses the case of utility-based pool algorithms. In Section 6 we study active learning for binary classification. We conclude in Section 7.

2. Definitions

For a predicate p, denote by $\mathbb{I}[p]$ the indicator function which is 1 if p holds and zero otherwise. For an integer k, denote $[k] := \{1, \ldots, k\}$ and $[k]_0 = \{0, \ldots, k\}$. For a sequence S, $S(i)$ is the

i'th member of the sequence. Denote concatenation of sequences by $\circ$. Denote by Π_k the set of permutations over $[k]$.

For A, B which are both sequences, or one is a set and one a sequence, we use $A =_\pi B$ and $A \subseteq_\pi B$ to denote equality or inclusion on the unordered sets of elements in B and in A. We sometimes omit the π when no other interpretation is possible.

Let $\mathcal{X}$ be a measurable domain of elements, and let $\mathcal{Y}$ be a measurable domain of responses. A pool-based (or just pool) interactive algorithm $\mathcal{A}_p$ receives as input an integer $q \leq m$, and a pool of elements $(x_1, \ldots, x_m) \in \mathcal{X}^m$. We assume that for each x_i there is a response $y_i \in \mathcal{Y}$, which is initially hidden from $\mathcal{A}_p$. Denote $S = ((x_i, y_i))_{i \in [m]}$. We will assume throughout this work that S is drawn i.i.d. from a distribution over $\mathcal{X} \times \mathcal{Y}$. For a given S, S_X denotes the pool $(x_1, \ldots, x_m)$. At each round, $\mathcal{A}_p$ selects one of the elements i_t that have not been selected yet, and receives its response y_{i_t}. After q rounds, $\mathcal{A}_p$ terminates. Its output is the set $\{(x_{i_1}, y_{i_1}), \ldots, (x_{i_q}, y_{i_q})\}$. For a pool algorithm $\mathcal{A}_p$, denote by $\mathsf{sel}_p(S, t)$ the element that $\mathcal{A}_p$ selects at round t, if S is the pool it interacts with. $\mathsf{sel}_p(S, t)$, which can be random, can depend on S_X and on y_{i_k} for $k < t$. Denote by $\mathsf{sel}_p(S, [t])$ the sequence of elements selected by $\mathcal{A}_p$ in the first t rounds. $\mathsf{pairs}_p(S, t)$ and $\mathsf{pairs}_p(S, [t])$ similarly denote the selected elements along with their responses. The final output of $\mathcal{A}_p$ is the set of pairs in the sequence $\mathsf{pairs}_p(S, [q])$. We assume that $S \mapsto \mathsf{pairs}_p(S, [q])$ is measurable.

We assume for simplicity that the pool algorithm is permutation invariant. That is, for any $S, S' \subseteq (\mathcal{X} \times \mathcal{Y})^m$, if S' is a permutation of S then $\mathsf{sel}_p(S, [q]) = \mathsf{sel}_p(S', [q])$, or if $\mathcal{A}_p$ is randomized then the output distributions are the same. Since the pool S is drawn i.i.d. this does not lose generality.

A stream-based (or just stream) interactive algorithm $\mathcal{A}_s$ receives as input an integer q. We assume an infinite stream $S \subseteq (\mathcal{X} \times \mathcal{Y})^\infty$, where $S(t) = (x_t, y_t)$. We will assume that this stream is also an i.i.d. sample from a distribution over $\mathcal{X} \times \mathcal{Y}$. At iteration t, $\mathcal{A}_s$ observes x_t, and may select one of the following actions:

- Do nothing
- Select x_t and observe y_t
- Terminate.

At termination, the algorithm outputs a subset of size q of the set of pairs (x_t, y_t) it observed. Denote by $\mathsf{sel}_s(S, t)$ the t'th element that $\mathcal{A}_s$ selects and is also in the output set. Denote by $\mathsf{sel}_s(S, [t])$ the sequence of first t elements that $\mathcal{A}_s$ selects and are also in the output set. Use pairs_s to denote these elements along with their responses. The output of $\mathcal{A}_s$ when interacting with S is the set of the pairs in the sequence $\mathsf{pairs}_s(S, [q])$. We assume $S \mapsto \mathsf{pairs}_s(S, [q])$ is measurable. The total number of elements selected by $\mathcal{A}_s$ when interacting with S (including discarded elements) is denoted $N_{\mathsf{sel}}(\mathcal{A}_s, S, q)$. The number of iterations (observed elements) until $\mathcal{A}_s$ terminates is denoted $N_{\mathsf{iter}}(\mathcal{A}_s, S, q)$.

We would like to have stream algorithms that emulate pool algorithms, under the assumption that both the pool and the stream are drawn from the same distribution. We define an equivalence between a stream algorithm and a pool algorithm as follows.

Definition 1 *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ and let q be an integer. Let $S \sim \mathcal{D}^m, S' \sim \mathcal{D}^\infty$. A pool algorithm $\mathcal{A}_p$ and a stream algorithm $\mathcal{A}_s$ are* $(q, \mathcal{D})$-equivalent*, if the total-variation distance between the distributions of $\mathsf{pairs}_p(S, [q])$ and $\mathsf{pairs}_s(S', [q])$ is zero.*

In the standard stream setting defined above, the algorithm can only make decisions based on data observed in the past, that is its decision at iteration t can depend on $x_1, \ldots, x_t$, as well as the responses for the elements it selected until iteration $t-1$. Thus, the stream setting is harder than the pool setting in two separate aspects:

- It must make decisions without knowing what elements will show up after the current iteration; and
- It must select elements in the order of their appearance in the stream.

We introduce an additional setting, which we term *precognitive stream*. A precognitive stream algorithm follows the same protocol as the standard stream algorithm defined above, but we assume that it knows the elements in the entire stream, including those in future iterations. Formally, its decisions in iteration t for an input stream $S \subseteq (\mathcal{X} \times \mathcal{Y})^\infty$ may depend on the full (infinite) sequence elements S_X.

Denote by $\mathcal{D}_X$ the marginal of $\mathcal{D}$ on $\mathcal{X}$. For a given distribution $\mathcal{D}_X$ over $\mathcal{X}$, let $\textsc{ds}(\mathcal{D}_X)$ be the set of distributions over $\mathcal{X} \times \mathcal{Y}$ such that their marginal over $\mathcal{X}$ is equal to $\mathcal{D}_X$. Below, unless specified otherwise, we assume that the probability under $\mathcal{D}_X$ of observing any single $x \in \mathcal{X}$ is zero. This does not lose generality, since if this is not the case, $\mathcal{D}_X$ can be replaced by the distribution $\mathcal{D}_X \times \mathrm{Unif}[0,1]$, with the interactive algorithms ignoring the second element in the pair.

In some of the proofs below we use the following lemma, proved in Appendix A. This lemma captures a relationship between the expected number of observed Bernoulli trials and the probability of success in the last trial.

Lemma 2 *Let $\alpha \in (0,1), p \in (0, \alpha^4/2)$. Let $X_1, X_2, \ldots$ be independent Bernoulli random variables with $\mathbb{P}[X_i = 1] \leq p$. Let I be a random integer, which can be dependent on the entire sequence $X_1, X_2, \ldots$. Suppose that $\mathbb{P}[X_I = 1] \geq \alpha$. Then $\mathbb{E}[I] \geq \frac{\alpha^2}{4p}$.*

3. Simple Equivalent Stream Algorithms

Algorithm 1 Algorithm $\mathcal{A}_{\text{wait}}$

In the first m iterations, observe $x_1, \ldots, x_m$ and do nothing.
$S \leftarrow ((x_1, \star), \ldots, (x_m, \star))$
$j \leftarrow 1$, $t \leftarrow m+1$
repeat
  In iteration t, observe element x_t
  if $x_t = \mathsf{sel}_p(S, j)$ **then**
    Select x_t and observe y_t
    $S(i) \leftarrow (x_t, y_t)$
    $j \leftarrow j + 1$
  end if
  $t \leftarrow t + 1$
until $j = q + 1$
Return the set of all the pairs (x, y) in S with $y \neq \star$.

Emulating a pool algorithm in a streaming setting can be naively done using two extremely simple approaches, which we discuss below. However, each of these approaches is wasteful, in either the number of iterations it requires, or the number of selections it makes.

For the first approach, let $\mathcal{A}_p$ be a pool algorithm. For any discrete distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, and any q, it is easy to define a stream algorithm which is $(q, \mathcal{D})$-equivalent to $\mathcal{A}_p$. Let "$\star$" be some value not in $\mathcal{Y}$, and define $\mathcal{A}_{\text{wait}}$ as in Alg. 1. $\mathcal{A}_{\text{wait}}$ first draws a pool of size m, and then replicates the selections that a pool algorithm would have done for this pool, by each time waiting until the requested element is observed again in the stream. This stream algorithm is clearly $(q, \mathcal{D})$ equivalent to $\mathcal{A}_p$ for any discrete distribution $\mathcal{D}$, and it has $N_{\mathsf{sel}}(\mathcal{A}_{\text{wait}}, S', q) = q$ for all $S' \in (\mathcal{X} \times \mathcal{Y})^\infty$. However, $\mathbb{E}_{S' \sim \mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}_{\text{wait}}, S', q)]$ is not uniformly bounded, even for the class of discrete distributions. For instance, if the probability of observing any $x \in \mathcal{X}$ is some small p, then every selection that the algorithm makes would require observing $1/p$ elements in expectation.

In the second approach, one can simply select the first m elements observed by the stream, as done in the stream algorithm $\mathcal{A}_{\text{nowait}}$ defined in Alg. 2. This algorithm is also $(q, \mathcal{D})$ equivalent to $\mathcal{A}_p$, and it requires the smallest possible number of observed elements, since $N_{\mathsf{iter}}(\mathcal{A}_{\text{wait}}, S', q) = m$ for all $S' \in (\mathcal{X} \times \mathcal{Y})^\infty$, exactly the same as for the pool algorithm. However, in this case the number of selections is large, since $N_{\mathsf{sel}}(\mathcal{A}_{\text{nowait}}, S', q) = m > q$, regardless of q. This trivial algorithm is thus again unsatisfying.

Algorithm 2 Algorithm $\mathcal{A}_{\text{nowait}}$

input Pool size m, Black-box pool algorithm $\mathcal{A}_p$.

1: In each iteration $t \in [m]$, select x_t and observe y_t.
2: Return the pairs in $\mathsf{pairs}_p(S, q)$.

In the next section we consider the general pool emulation problem, and show that it is possible to have a uniform upper bound on the number of iterations, regardless of the distribution, and without selecting more than q elements. In addition, in Section 5, we show for utility-based pool algorithms, that there is a tradeoff between the number of additional selected elements and the expected number of observed elements. This tradeoff is evident also in the two simple algorithms shown above.

4. General Pool Algorithms

In this section we consider emulating general pool algorithms. In Section 4.1 we present a stream algorithm which can emulate any pool-based algorithm, using only black-box access to the pool algorithm. We prove a distribution-free upper bound on its expected number of iterations. In Section 4.2 we show that the number of iterations required by the proposed algorithm cannot be significantly improved. In Section 4.3, we study the precognitive stream setting and compare its guarantees to the standard stream setting.

4.1 Stream Emulation for General Pool Algorithms

The stream algorithm $\mathcal{A}_{\text{gen}}$, listed in Alg. 3, emulates any pool based algorithm $\mathcal{A}_p$ using only black-box access to $\mathcal{A}_p$. The algorithm emulates a general pool algorithm by making sure that in each iteration, its probability of selecting an element is identical to the conditional probability of the pool algorithm selecting the same element, conditioned on the history of elements and responses

selected and observed so far. This is achieved by repeatedly drawing the remaining part of the pool, and keeping it only if it is consistent with the elements that were already selected. The algorithm further uses the partial pool draw only if the element to be selected happens to have been observed last.

Algorithm 3 Algorithm $\mathcal{A}_{\text{gen}}$

input Original pool size m, budget $q < m$, black-box pool algorithm $\mathcal{A}_p$.

$S_0 \leftarrow ()$
for $i = 1 : q$ **do**
  repeat
    Draw $m - i + 1$ elements, denote them $\bar{x}_{i,i}, \ldots, \bar{x}_{i,m}$.
    $S'_i \leftarrow ((\bar{x}_{i,i}, \star), \ldots, (\bar{x}_{i,m}, \star))$.
  until $\mathsf{pairs}_p(S_{i-1} \circ S'_i, [i-1]) =_\pi S_{i-1}$ and $\mathsf{sel}_p(S_{i-1} \circ S'_i, i) = \bar{x}_{i,m}$.
  Select $\bar{x}_{i,m}$, get the response $\bar{y}_{i,m}$.
  $S_i \leftarrow S_{i-1} \circ ((\bar{x}_{i,m}, \bar{y}_{i,m}))$.
end for
Output S_q.

Below we show that $\mathcal{A}_{\text{gen}}$ improves over the two stream algorithms presented above, in that it selects exactly q elements, and has a uniform upper bound on the expected number of iterations, for any source distribution. First, we prove that $\mathcal{A}_{\text{gen}}$ indeed emulates any pool-based algorithm.

Theorem 3 *For any pool algorithm $\mathcal{A}_p$, any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, any integer m and $q \leq m$, $\mathcal{A}_s := \mathcal{A}_{gen}(\mathcal{A}_p)$ is $(q, \mathcal{D})$-equivalent to $\mathcal{A}_p$.*

Proof For simplicity of presentation, we prove the result for discrete distributions. The proof for continuous distribution is analogous. Consider the probability space defined by the infinite sequence $S' \sim \mathcal{D}^\infty$ which generates the input to the stream algorithm, and an independent sequence $S \sim \mathcal{D}^m$ which is the input to the pool algorithm. For $z_1, \ldots, z_q \in \mathcal{X} \times \mathcal{Y}$, denote $Z_j = \{z_1, \ldots, z_j\}$. We have, for every $i \in [q]$,

$$\mathbb{P}[\mathsf{pairs}_p(S, [i]) =_\pi Z_i] = \sum_{j=1}^{i} \mathbb{P}[(\mathsf{pairs}_p(S, i) = z_j) \wedge (\mathsf{pairs}_p(S, [i-1]) =_\pi Z_i \setminus \{z_j\})]$$

$$= \sum_{j=1}^{i} \mathbb{P}[\mathsf{pairs}_p(S, i) = z_j \mid \mathsf{pairs}_p(S, [i-1]) =_\pi Z_i \setminus \{z_j\}] \cdot \mathbb{P}[\mathsf{pairs}_p(S, [i-1]) =_\pi Z_i \setminus \{z_j\}].$$

The same holds for $\mathsf{pairs}_s(S', \cdot)$. To show the equivalence it thus suffices to show that for all $z_1, \ldots, z_q \in \mathcal{X} \times \mathcal{Y}$, $i \in [q]$,

$$\mathbb{P}[\mathsf{pairs}_s(S', i) = z_i \mid \mathsf{pairs}_s(S', [i-1]) =_\pi Z_{i-1}] = \mathbb{P}[\mathsf{pairs}_p(S, i) = z_i \mid \mathsf{pairs}_p(S, [i-1]) =_\pi Z_{i-1}].$$

From the definition of $\mathcal{A}_s$ we have

$$\begin{aligned}
&\mathbb{P}[\mathsf{pairs}_s(S', i) = z_i \mid \mathsf{pairs}_s(S', [i-1]) =_\pi Z_{i-1}] \\
&= \mathbb{P}[\mathsf{pairs}_p(S_{i-1} \circ S'_i, i) = z_i \mid S_{i-1} =_\pi Z_{i-1} \wedge \mathsf{pairs}_p(S_{i-1} \circ S'_i, [i-1]) =_\pi Z_{i-1}] \\
&= \mathbb{P}[\mathsf{pairs}_p(S, i) = z_i \mid \mathsf{pairs}_p(S, [i-1]) =_\pi Z_{i-1}].
\end{aligned}$$

The last equality follows since $\mathcal{A}_p$ is permutation invariant and never selects the same index twice. This proves the equivalence. ■

The next theorem provides an upper bound on the expected number of elements observed by $\mathcal{A}_{\text{gen}}$. Unlike $\mathcal{A}_{\text{wait}}$, this upper bound holds uniformly for all source distributions.

Theorem 4 *For any pool algorithm $\mathcal{A}_p$, any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, any integer m and $q \leq m$, if $\mathcal{A}_s := \mathcal{A}_{gen}(\mathcal{A}_p)$, $N_{\mathsf{sel}}(\mathcal{A}_s, S, q) = q$ for any $S \in (\mathcal{X} \times \mathcal{Y})^\infty$, and*

$$\mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}_s, S, q)] \leq m^2 \left(\frac{em}{q-1}\right)^{q-1}.$$

Proof First, clearly $N_{\mathsf{sel}}(\mathcal{A}_s, S, q) = q$ for any $S \sim \mathcal{D}^\infty$. We now prove the upper bound on the expected number of iterations of $\mathcal{A}_s$. Let $S \sim \mathcal{D}^m$. For $i \geq 1$, $z_1, \ldots, z_{i-1} \in \mathcal{X}$, denote $Z_j = \{z_1, \ldots, z_j\}$, and let

$$p_i(z_1, \ldots, z_i) := \mathbb{P}[\mathsf{sel}_p(S, [i]) =_\pi Z_i \mid Z_i \subseteq_\pi S_X].$$

Suppose that $(S_{i-1})_X =_\pi Z_{i-1}$. The expected number of times that steps 3 to 6 are repeated for index i is the inverse of the probability that the condition in 6 holds. This condition, in our notation, is that $\mathsf{sel}_p(S_{i-1} \circ S'_i, [i-1]) =_\pi Z_{i-1}$ and $\mathsf{sel}_p(S_{i-1} \circ S'_i, i) = \bar{x}_{i,m}$. We have, from the permutation invariance of $\mathcal{A}_p$,

$$\mathbb{P}[\mathsf{sel}_p(S_{i-1} \circ S'_i, [i-1]) =_\pi Z_{i-1} \mid (S_{i-1})_X =_\pi Z_{i-1}] = p_{i-1}(z_1, \ldots, z_{i-1}).$$

In addition, for every draw of S'_i,

$$\mathbb{P}[\mathsf{sel}_p(S_{i-1} \circ S'_i, i) = \bar{x}_{i,m} \mid \mathsf{sel}_p(S_{i-1} \circ S'_i, [i-1]) =_\pi Z_{i-1} \wedge (S_{i-1})_X =_\pi Z_{i-1}] = \frac{1}{m-i+1}.$$

This is since under the conditional, one of the elements in S'_i must be selected by $\mathcal{A}_p$ in round i. Therefore, the probability that the condition in step 6 holds is $p_{i-1}(z_1, \ldots, z_{i-1})/(m-i+1)$. The expected number of times that steps 3 to 6 are repeated for index i is the inverse of that, and in each round $m-i+1$ elements are observed. Therefore the expected number of elements observed until selection i is made conditioned on $z_1, \ldots, z_{i-1}$ is $(m-i+1)^2/p_{i-1}(z_1, \ldots, z_{i-1})$. The unconditional expected number of elements observed until selection i is $(m-i+1)^2 \cdot \mathbb{E}[1/p_{i-1}(\mathsf{sel}_s(S', [i-1]))]$. For a set of indices J, denote $S|_J = \{S(j) \mid j \in J\}$. For simplicity of presentation we give the following derivation for discrete distributions, the proof for continuous distributions is analogous.

$$\begin{aligned}
\mathbb{E}[1/p_i(\mathsf{sel}_s(S', [i])] &= \mathbb{E}[1/p_i(\mathsf{sel}_p(S, [i])] \\
&= \sum_{\{z_1,\ldots,z_i\}\subseteq\mathcal{X}\times\mathcal{Y}} \mathbb{P}[\mathsf{sel}_p(S, [i]) =_\pi Z_i] \cdot \frac{1}{p_i(z_1, \ldots, z_i)} \\
&= \sum_{\{z_1,\ldots,z_i\}\subseteq\mathcal{X}\times\mathcal{Y}} \mathbb{P}[Z_i \subseteq_\pi S_X].
\end{aligned}$$

Hence

$$\begin{aligned}\mathbb{E}[1/p_i(\mathsf{sel}_s(S',[i])] &\leq \sum_{\{z_1,\ldots,z_i\}\subseteq\mathcal{X}\times\mathcal{Y}} \sum_{J\subseteq[m],|J|=i} \mathbb{P}[(S|_J)_X = Z_i] \\ &= \sum_{J\subseteq[m],|J|=i} \sum_{\{z_1,\ldots,z_i\}\subseteq\mathcal{X}\times\mathcal{Y}} \mathbb{P}[(S|_J)_X = Z_i] \\ &= \sum_{J\subseteq[m],|J|=i} 1 = \binom{m}{i}.\end{aligned}$$

It follows that the expected number of elements observed after the $i-1$'th selection and until selection i is at most $(m-i+1)^2\binom{m}{i-1}$. We conclude that

$$\mathbb{E}[N_{\mathsf{iter}}(\mathcal{A}_s,S,q)] \leq \sum_{i=0}^{q-1}(m-i)^2\binom{m}{i} \leq m^2\left(\frac{em}{q-1}\right)^{q-1}.$$

This completes the proof. ■

From the existence of $\mathcal{A}_{\text{gen}}$ we can conclude that the pool-based and the stream-based setting are essentially equivalent, up to the number of observed elements. However, the expected number of observed elements is exponential in q. In the next sections we show that this exponential dependence cannot be avoided when emulating general pool algorithms in a stream setting.

4.2 A Lower Bound for the Standard Stream Setting

The upper bound in Theorem 4 is exponential in q. We next show that this dependence cannot be eliminated in the general case for a standard stream algorithm. This result indicates that $\mathcal{A}_{\text{gen}}$ is close to optimal in terms of expected number of iterations. The lower bound holds for a standard stream algorithm, which does not know the identity of future elements in the stream. We consider precognitive stream algorithms in Section 4.3.

The lower bound is proved using a construction in which some of the elements represent base elements, and some represent permutations over base elements. The pool algorithm selects permutation elements that are consistent with the base elements it selected. Since the ranking of elements depends on the elements in the input pool, the same permutation can be consistent with different selections in different pools. The following lemma, which is used the proof of the lower bound, shows that nonetheless, once certain base elements have been selected, the set of permutations that are likely consistent with them is considerably limited. The lemma is also later used in a lower bound for the precognitive stream setting.

Lemma 5 *Let $t \leq t' \leq l$ be integers such that $l \geq 2t$. Let $Z = z_1,\ldots,z_t$ a set of values in $[0,1]$. Let X be l random values sampled i.i.d. from the uniform distribution on $[0,1]$. Denote $X = x_1,\ldots,x_l$ where $x_1 \leq x_2 \leq \ldots x_l$. Let $A_\sigma(Z,X) \subseteq \Pi_l$ be the set of permutations σ such that $\{x_{\sigma(1)},\ldots,x_{\sigma(t')}\} \supseteq \{z_1,\ldots,z_t\}$. For any $d \leq l-t$, there exists a set of permutations $\Phi(Z) \subseteq \Pi_l$, such that*

$$|\Phi(Z)|/|\Pi_l| \leq (4dt'/l)^t,$$

and

$$\mathbb{P}[A_\sigma(Z, X) \subseteq \Phi(Z) \mid Z \subseteq X] \geq 1 - 2t \exp(-2d^2/(l-t)),$$

Where the probability is over the randomness of X. Moreover, let $L(\sigma) := \cup_{Z:\sigma\in\Phi(Z)}\Phi(Z)$, then for all $\sigma \in \Pi_l$,

$$|L(\sigma)|/|\Pi_l| \leq (8dt'/l)^{2t}.$$

Proof Fix $d \leq l - t$. Denote the expected number of elements that are smaller than z_i in X, conditioned on $Z \subseteq X$, by $n_i := (l-t)z_i + \sum_{j=1}^{t} \mathbb{I}[z_j < z_i]$, and let

$$\Phi(Z) := \{\sigma \in \Pi_l \mid \exists f : [t] \to [t'] \text{ s.t. } f \text{ is one-to-one and } \forall i \in [t], |\sigma(f(i)) - n_i| < d\}. \quad (1)$$

These are the permutations such that the first t elements according to the permutation are mapped from elements with ranks in $[n_i - d, n_i + d]$. It is easy to see that

$$|\Phi(Z)|/|\Pi_l| \leq (t')^t \frac{(2d)^t}{\prod_{i=0}^{t-1}(l-i)} \leq \left(\frac{4dt'}{l}\right)^t,$$

where the last inequality follows since $l \geq 2t$. This proves the first part of the lemma. We now show the second part of the lemma. For $x \in X$, let $r(x)$ be its rank in X. By Hoeffding's inequality, for any $i \leq t$,

$$\mathbb{P}[|r(z_i) - n_i| \geq d \mid Z \subseteq X] \leq 2 \exp(-2d^2/(l-t)).$$

Therefore,

$$\mathbb{P}[\forall i \leq t, |r(z_i) - n_i| < d \mid Z \subseteq X] \geq 1 - 2t \exp(-2d^2/(l-t)).$$

For any $\sigma \in A_\sigma$ we have that for each $i \leq t$, $z_i = x_{\sigma(j)}$ for some $j \leq t'$, that is $r(z_i) = \sigma(j)$. Moreover, different values of i are mapped to different values of j. Therefore there is some one-to-one function $f : [t] \to [t']$ such that for all $i \leq t$, $z_i = x_{\sigma(f(i))}$. Conditioned on the event $\forall i \leq t, |r(z_i) - n_i| < d$, it follows that $\forall i \leq t, |\sigma(f(i)) - n_i| < d$. Thus $\sigma \in \Phi(Z)$, which proves the second claim of the lemma.

To see the last claim, observe that if $\sigma, \sigma' \in \phi(Z)$ for some Z, then there are functions $f, g : [t] \to [t']$ such that for $i \in [t]$, $|\sigma(f(i)) - \sigma'(g(i))| < 2d$. Therefore,

$$L(\sigma) \subseteq \{\sigma' \in \Pi_l \mid \exists f, g : [t] \to [t'], \forall i \in [t], |\sigma(f(i)) - \sigma'(g(i))| < 2d\}.$$

The bound on the size of $L(\sigma)$ in the claim directly follows, similarly to the bound on $|\Phi(Z)|$. ■

The lower bound for the standard stream setting is provided below. It shows that for some pool algorithm, any equivalent stream algorithm has an expected number of observed elements which is exponential in q. The lower bound is worst-case over all source distributions $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where the pool and the stream, as above, are drawn i.i.d. from the distribution. The proof involves constructing a pool-based algorithm in which the last selected element is significantly constrained by the identity of the previously selected elements, using the permutation construction. This type of constraint is not an issue in a pool setting, since the algorithm has advance knowledge of all the available elements. In a streaming setting, however, this requires a possibly long wait to obtain the matching last element. Because the stream algorithm is allowed to select elements in a different order than the pool algorithm, additional care is taken to make sure that in this construction, it is not possible to circumvent the problem this way.

Theorem 6 *There is an integer q_0, such that for $q \geq q_0$ and $m \geq 16q^2 \log(4q)+1$, there exist a pool algorithm $\mathcal{A}_p$ and a marginal $\mathcal{D}_X$, such that any stream algorithm $\mathcal{A}_s$ which is $(q, \mathcal{D})$ equivalent to $\mathcal{A}_p$ for all $\mathcal{D} \in \text{DS}(\mathcal{D}_X)$, and selects only q elements, has*

$$\exists \mathcal{D} \in \text{DS}(\mathcal{D}_X), \mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}_s, S, q)] \geq \frac{1}{1{,}000}\left(\frac{m-1}{8q^2\log(4q)}\right)^{\frac{q-1}{2}}.$$

Proof Let the domain of elements be $\mathcal{X} = [0, 2)$ and assume responses in $\mathcal{Y} = \{0, 1\}$. Denote $\texttt{base} := [0, 1]$ and $\texttt{perm} := (1, 2)$. Call a pool S_X in which exactly one element in the pool is in $\texttt{perm}$ and the rest are in $\texttt{base}$ a "good pool".

We now define a pool algorithm as follows. On bad pools, $\mathcal{A}_p$ always selects only elements in $\texttt{base}$ or only elements in $\texttt{perm}$. In a good pool, denote for simplicity the single element from $\texttt{perm}$ by z_m, and the elements from $\texttt{base}$ by $z_1, \ldots, z_{m-1}$, where $z_{i-1} < z_i$ for $i \in [m-1]$. Define a mapping $\psi : \texttt{perm} \to \Pi_{m-1}$, such that if z_m is uniform over $\texttt{perm}$, then for $\psi(z_m)$ all permutations in the range are equally likely. Let $\sigma = \psi(z_m)$. On a good pool, $\mathcal{A}_p$ behaves as follows: The first $q-1$ elements that $\mathcal{A}_p$ selects are $z_{1+\sigma(1)}, \ldots, z_{1+\sigma(q-1)}$. The last element that it selects is z_m if the response for all previous elements was 0, and z_1 otherwise.

Define the marginal $\mathcal{D}_X$ over $\mathcal{X}$ such that for $X \sim \mathcal{D}_X$, $\mathbb{P}[X \in \texttt{base}] = 1 - 1/m$, $\mathbb{P}[X \in \texttt{perm}] = 1/m$, and in each range $\texttt{base}, \texttt{perm}$, X is uniform. The probability of a good pool under $\mathcal{D} \in \text{DS}(\mathcal{D}_X)$ is $(1-1/m)^{m-1} \geq 1/e^2 =: c_1$. We now show a lower bound on the expected number of iterations of a stream algorithm which is $(q, \mathcal{D})$-equivalent to any $\mathcal{D} \in \text{DS}(\mathcal{D}_X)$. Let $\mathcal{D}_0$ be the distribution over $\mathcal{X} \times \mathcal{Y}$ such that for $(X, Y) \sim \mathcal{D}_0$, $X \sim \mathcal{D}_X$ and $Y = 0$ with probability 1. Let $S \sim \mathcal{D}_0^m$ be the input to $\mathcal{A}_p$.

We apply Lemma 5 with $t = t' = q-1, l = m-1, d = \sqrt{(m-q)\log(4q)/2}$. By this lemma, for any set $Z = \{z_1, \ldots, z_{q-1}\} \subseteq \texttt{base}$, there exists a set of permutations $\Phi(Z) \subseteq \Pi_{m-1}$ such that

$$|\Phi(Z)|/|\Pi_{m-1}| \leq \left(\frac{8q^2(m-q)\log(4q)}{(m-1)^2}\right)^{(q-1)/2} \leq \left(\frac{8q^2\log(4q)}{m-1}\right)^{\frac{q-1}{2}}, \tag{2}$$

and also

$$\mathbb{P}[\psi(z_m) \in \Phi(Z) \mid \mathsf{sel}_p(S, [q-1]) =_\pi Z \wedge S \text{ is good}] \geq 1 - 2(q-1)\exp(-\log(4q)) \geq \frac{1}{2}. \tag{3}$$

The theorem follows from the following two claims:

1. When $\mathcal{A}_s$ emulates a good pool, it selects an element from $\texttt{perm}$ only after selecting $q-1$ elements from $\texttt{base}$.
2. Therefore, when $\mathcal{A}_s$ emulates a good pool, the expected number of observed elements until selecting the last element is lower bounded, and so the overall expected number is lower bounded.

We first prove claim 1. Consider a stream algorithm which is $(q, \mathcal{D})$-equivalent to $\mathcal{A}_p$ for any $\mathcal{D} \in \text{DS}(\mathcal{D}_X)$. Consider runs of $\mathcal{A}_s$ with input $S' \sim \mathcal{D}_0^\infty$. Denote by E_g the event that the output of $\mathcal{A}_s$ is equal to a possible output of $\mathcal{A}_p$ on a good pool with $S \sim \mathcal{D}_0^m$. Then $\mathbb{P}[E_g] \geq c_1$. Claim 1 is that

$$\mathbb{P}[\mathsf{sel}_s(S', [q-1]) \subseteq_\pi \texttt{base} \mid E_g] = 1. \tag{4}$$

In other words, when simulating a good pool, the elements in `base` are all selected before the element in `perm`.

To show claim 1, note that by the definition of $\mathcal{A}_p$, for any source distribution over $\mathcal{X} \times \mathcal{Y}$, if $\mathcal{A}_p$ outputs a set with elements both in `base` and in `perm`, then there is exactly one element in `perm` in the output, and all the responses in the output for elements in `base` are 0 with probability 1.

Now, suppose that $\mathbb{P}[\mathsf{sel}_s(S', [q-1]) \subseteq_\pi \texttt{base} \mid E_g] < 1$. Then $\mathbb{P}[\mathsf{sel}_s(S', q) \in \texttt{base} \mid E_g] > 0$, since there can be only one element in `perm` in the output of a good pool. But, consider running $\mathcal{A}_s$ with a source distribution $\mathcal{D}' \in \textsc{ds}(\mathcal{D}_X)$ such that for $(X, Y) \sim \mathcal{D}'$, $X \sim \mathcal{D}_X$ and $\mathbb{P}_{\mathcal{D}'}[Y = 0 | X = x] = \frac{1}{2}$ for all x. There is a positive probability that in the first $q-1$ selected elements all the responses are 0, just as for $\mathcal{D}_0$. Therefore, also for $S'' \sim \mathcal{D}'^{\infty}$, $\mathbb{P}[\mathsf{sel}_s(S'', q) \in \texttt{base} \mid E_g] > 0$. But then there is a positive probability that the response for the last element, which is in `base`, is 1, contradicting the $(q, \mathcal{D}')$-equivalence of the pool and $\mathcal{A}_s$. This proves claim 1.

We now show claim 2, which completes the proof. From claim 1 in Eq. (4), we conclude that $\mathbb{P}[\mathsf{sel}_s(S', q) \in \texttt{perm} \mid E_g] = 1$. Therefore, from Eq. (3), for any $Z \subseteq \texttt{base}$ with $|Z| = q - 1$,

$$\mathbb{P}[\psi(\mathsf{sel}_s(S', q)) \in \Phi(\mathsf{sel}_s(S', [q-1])) \mid E_g] \geq 1/2.$$

Therefore

$$\mathbb{P}[\psi(\mathsf{sel}_s(S', q)) \in \Phi(\mathsf{sel}_s(S', [q-1]))] \geq \mathbb{P}[E_g]/2 \geq c_1/2.$$

Now, let $X_i \sim \mathcal{D}_X$ be the i'th element observed after selecting the first $q-1$ elements, and let $B_i = \mathbb{I}[\psi(X_i) \in \Phi(Z)]$, where Z is the set of $q-1$ selected elements. B_i are independent Bernoulli random variables, each with a probability of success at most p, where by Eq. (2)

$$p \leq \frac{|\Phi(Z)|}{|\Pi_{m-1}|} \leq \left(\frac{8q^2 \log(4q)}{m-1} \right)^{\frac{q-1}{2}}.$$

Let I be the number of elements that $\mathcal{A}_s$ observes after selecting Z, until selecting element q. We have $\mathbb{P}[B_I = 1] \geq c_1/2$. From the assumption in the theorem statement that $m \geq 16q^2 \log(4q) + 1$, we have that for sufficiently large q, $p \leq c_1^4/32$. By Lemma 2, it follows that $\mathbb{E}[I] \geq c_1^2/(16p) \geq (1000p)^{-1}$. Hence

$$\mathbb{E}[I] \geq \frac{1}{1000} \left(\frac{m-1}{8q^2 \log(4q)} \right)^{\frac{q-1}{2}}.$$

Since $\mathbb{E}[N_{\mathsf{iter}}(\mathcal{A}, S, q)] \geq \mathbb{E}[I]$, this completes claim 2 and finalizes the proof. ■

The lower bound, as well as the upper bound in Theorem 4, both show an exponential dependence on q. However, the exponent in the lower bound is about half that of the upper bound. Some of this gap might be an artifact of the fact that in the lower bound, only a fixed marginal distribution is considered. Closing this gap remains an open problem, which we leave for future work.

4.3 Emulation with a Precognitive Stream Algorithm

We next consider the precognitive setting. In the precognitive setting, the stream algorithm has the advantage that it can plan ahead, since it knows in advance the identity of elements that will be available for selection from the entire stream. This breaks the construction used in the lower bound of Theorem 6, thus showing that in some cases there is a significant gap between standard stream algorithms and precognitive stream algorithms. We prove this in the following theorem.

Theorem 7 *There is an integer q_0, such that for $q \geq q_0$ and $m \geq 16q^2\log(4q)+1$, there exist a pool algorithm $\mathcal{A}_p$ and a marginal $\mathcal{D}_X$, such that any stream algorithm $\mathcal{A}_s$ which is $(q, \mathcal{D})$ equivalent to $\mathcal{A}_p$ for all $\mathcal{D} \in \mathrm{DS}(\mathcal{D}_X)$, and selects only q elements, requires an expected number of iterations of $\Omega\left(\left(\frac{m}{q^2\log(q)}\right)^{q/2}\right)$, while there exists a precognitive stream algorithm that satisfies the same equivalence, and requires only m^2 iterations in expectation.*

Proof Consider the same construction as in the proof of Theorem 6: the same $\mathcal{D}_X$, and the same behavior of the pool algorithm on "good pools", defined as in that proof. Assume further, consistently with the definition of $\mathcal{A}_p$ in that proof, that on bad pools, the pool algorithm checks if there are more `base` element or more `perm` elements in the pool, and then selects the first q elements in the pool that belong to the set with the larger number (in case of a tie, `base` is selected). Denote $\texttt{base}, \texttt{perm}, \psi$ as in the proof of Theorem 6.

From Theorem 6 we have that the expected number of iterations of a standard stream algorithm on this problem is $\Omega\left(\left(\frac{m}{q^2\log(q)}\right)^{q/2}\right)$.

Now, consider the following precognitive stream algorithm: The algorithm checks, before the start of the iterations, whether the first m stream elements constitute a good pool. If they do not, it checks whether there are more `base` or more `perm` elements in the first m elements. Then it selects the first q elements in the stream that belong to the majority set (tie-breaking with preference to `base` as in the pool algorithm). Thus, for bad pools, the expected number of iterations by the precognitive stream algorithm is at most m.

Now, if the first m elements in the stream constitute a good pool, then the precognitive stream algorithm finds an integer t such that x_t is the smallest element from `base` among the previous $m-1$ `base` elements, using the following procedure:

1. Let $t_0 \leftarrow 1$; $t_1 \leftarrow m$.
2. Let A be the set of $m-1$ elements from `base` in $x_{t_0}, \ldots, x_{t_1}$.
3. If x_{t_1} is the minimum of A, set $t \leftarrow t_1$ and terminate.
4. Otherwise, set $t_0 \leftarrow t_1 + 1$, and $t_1 \leftarrow$ the smallest integer such that $x_{t_0}, \ldots, x_{t_1}$ has exactly $m-1$ elements from `base`. Go to 2

After setting t, the algorithm identifies the earliest iteration $t' > t$ such that $x_{t'} \in \texttt{perm}$. The algorithm then ranks the $m-1$ elements from `base` that immediately precede x_t, denoting them $z_2 < \ldots < z_{m-1}$, and denotes $z_1 := x_t$.

The algorithm then selects $z_{1+\sigma(1)}, \ldots, z_{1+\sigma(q-1)}$ in the order of their appearance in the stream, where $\sigma := \psi(x_{t'})$. Then, if all of the responses to these elements were 0, it selects $x_{t'}$. Otherwise, it selects z_1.

It is easy to see that this algorithm is equivalent to the pool algorithm defined above. Also, the expected number of iterations of this algorithm is

$$\mathbb{E}[t'] = \mathbb{E}[t' - t] + \mathbb{E}[t].$$

Now, $\mathbb{E}[t'-t]$ is the expected wait time until observing an element from `perm`. Since for $X \sim \mathcal{D}_X$, $\mathbb{P}[X \in \texttt{perm}] = 1/m$, we have $\mathbb{E}[t'-t] = m$. In addition, by Wald's identity, $\mathbb{E}[t]$ is equal to the expected number of rounds of the procedure for setting t above, which is $m-1$, times the expected size

of $t_1 - t_0 + 1$ in each round, which is $\mathbb{P}[X \in \texttt{base}] \cdot (m-1)$. Thus, $\mathbb{E}[t] = (m-1)^3/m \leq (m-1)m$. It follows that the expected number of iterations of the precognitive stream algorithm is $\mathbb{E}[t'] \leq m^2$. ■

This result shows that the precognitive setting has a significant advantage in some cases. Nonetheless, we show a lower bound, indicating that its worst-case performance cannot be much better than the worst-case performance of a standard stream algorithm. The proof of the lower bound shares some techniques with the proof of Theorem 6, especially the use of permutations. It additionally relies on the observation that while the stream algorithm knows in advance what elements will be available for selection and when, it is still constrained to selecting the elements in the order that they are provided by the stream. It also must keep to a certain order of selections to emulate the pool algorithm, since the pool algorithm makes decisions based on previously observed responses. Thus, in the proof of the lower bound, we construct a pool algorithm that might select two permutation elements, but the order in which they would be selected depends on the responses to selected base elements. As a result, the precognitive algorithm cannot tell in advance in which order the two elements must be selected. The probability of both orders appearing early enough in the stream is low, so that even when the whole stream is taken into account, an exponential dependence cannot be avoided, though this dependence is weaker than the one shown in Theorem 6 for standard stream algorithms.

The proof of the lower bound employs the following lemma, which states that after partially observing two sequences of fair coin tosses, there is a positive probability that both sequences share the majority outcome, as well as a positive probability that their majority outcome is different. In the lower bound below, this lemma is used to show that the precognitive algorithm must commit to a small set of permutations before finding out in which order they must be selected.

Lemma 8 *Consider n i.i.d. tosses of two fair coins, where n is an odd number. Denote the tosses of the first coin by $A_1, \ldots, A_n \in \{0,1\}$, and the second by $B_1, \ldots, B_n \in \{0,1\}$. Let $K \in \{n/2, \ldots, n\}$ be a stopping time for the sequence $B_1, \ldots, B_n$, which can depend also on $\bar{A} := (A_1, \ldots, A_{n/2})$. Denote $\bar{B}(K) := (B_1, \ldots, B_K)$.*

There exists some integer n_0 and constants $c = 0.68$, $c' = 0.15$ such that for all odd $n > n_0$, with a probability at least c over the values of $\bar{A}, K, \bar{B}(K)$,

$$\mathbb{P}\Big[\mathbb{I}[\sum_{i=1}^{n} A_i > n/2] = \mathbb{I}[\sum_{i=1}^{n} B_i > n/2] \;\Big|\; \bar{A}, K, \bar{B}(K)\Big] \in [c', 1-c']$$

Proof Denote for brevity $A_j^l := \sum_{i=j}^{l} A_i$ and similarly for B_j^l. First note that

$$\begin{aligned}
&\mathbb{P}\left[\mathbb{I}[A_1^n > n/2] = \mathbb{I}[B_1^n > n/2] \;\middle|\; K, \bar{A}, \bar{B}(K)\right] \\
&= \mathbb{P}[B_1^n > n/2 \mid K, \bar{B}(K)] \cdot \mathbb{P}[A_1^n > n/2 \mid \bar{A}] + \mathbb{P}[B_1^n \leq n/2 \mid K, \bar{B}(K)] \cdot \mathbb{P}[A_1^n \leq n/2 \mid \bar{A}]
\end{aligned}$$

Thus it suffices to prove that with a probability at least c over the values of $\bar{A}$, $\mathbb{P}[A_1^n > n/2 \mid \bar{A}] \in [c', 1-c']$, which implies the same for $\mathbb{P}[A_1^n \leq n/2 \mid \bar{A}]$, leading to the claim of the lemma.

We first consider the limit case $n \to \infty$. In this case $A_1^{n/2} \sim N(n/4, \sqrt{n/8})$. Thus, with a probability of $\mathrm{erf}(1/\sqrt{2}) > 0.68 =: c$,

$$A_1^{n/2} \in [n/4 - \sqrt{n/8}, n/4 + \sqrt{n/8}].$$

If this is the case, then

$$A^n_{n/2+1} < n/4 - \sqrt{n/8} \quad \Longrightarrow \quad A^n_1 < n/2, \text{ and}$$
$$A^n_{n/2+1} > n/4 + \sqrt{n/8} \quad \Longrightarrow \quad A^n_1 > n/2.$$

Since $A^n_{n/2+1} \sim N(n/4, \sqrt{n/8})$, each of the above events occurs with a probability at least $\frac{1}{2} - \mathrm{erf}(1/\sqrt{2})/2 > 0.15 =: c'$. Thus, with a probability at least c over the values of $\bar{A}$,

$$\mathbb{P}[A^n_1 > n/2 \mid \bar{A}], \mathbb{P}[A^n_1 \leq n/2 \mid \bar{A}] \in [c', 1-c'].$$

The same holds for any large enough finite value of n. The statement of the lemma directly follows. ■

The lower bound for the precognitive setting is given below. It is exponential in q, like the lower bound for the standard stream setting in Theorem 6, however the dependence is slightly weaker.

Theorem 9 *There is an integer q_0 such that for any $q \geq q_0$ and $m \geq 2q+1$ there exist a pool algorithm $\mathcal{A}_p$ and a distribution $\mathcal{D}$, such that any precognitive stream algorithm $\mathcal{A}_s$ which is $(q, \mathcal{D})$ equivalent to $\mathcal{A}_p$ and selects only q elements, has*

$$\mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}_s, S, q)] \geq \frac{1}{600}\left(\frac{m}{8q^2\log(2q)}\right)^{q/8}.$$

Proof Assume for simplicity that 4 divides q and m. Let the domain of elements be $\mathcal{X} = [0,4) \cup \{\star\}$ and assume responses in $\mathcal{Y} = \{0,1\}$. Denote $\texttt{base1} := [0,1]$, $\texttt{base2} := [2,3]$, $\texttt{perm1} := (1,2)$ and $\texttt{perm2} := (3,4)$. Denote $\texttt{base} = \texttt{base1} \cup \texttt{base2}$ and $\texttt{perm} = \texttt{perm1} \cup \texttt{perm2}$.

We define the pool algorithm $\mathcal{A}_p$ as follows. Call a pool S_X a "good pool" if it includes at least $m/4$ elements from each of $\texttt{base1}$ and $\texttt{base2}$, and at least one element from each of $\texttt{perm1}, \texttt{perm2}, \star$. In a good pool S_X, the pool algorithm $\mathcal{A}_p$ behaves as follows. For $i \in \{1,2\}$, let $X_i \subseteq S_X \cap \texttt{base}(i)$ be a subset of size $m/4$, selected uniformly at random from $S_X \cap \texttt{base}(i)$. Let $x_{p(i)}$ be a random element from $S_X \cap \texttt{perm}(i)$. Let $\sigma_1 = \psi(x_{p(1)} - 1)$, $\sigma_2 = \psi(x_{p(2)} - 3)$. For $j \in [m/4]$, let x^i_j be the j-th largest value in X_i. Define a mapping $\psi : (0,1) \to \Pi_{m/4}$, such that if Z is uniform over $(0,1)$, then for $\psi(Z)$ all permutations in the range are equally likely.

The first $q-2$ elements that $\mathcal{A}_p$ selects are $\bar{X}_i := x^i_{\sigma_i(1)}, \ldots, x^i_{\sigma_i(q/2-1)}$, for each of $i \in \{1,2\}$. The last 2 elements that $\mathcal{A}_p$ selects are determined as follows: Let $r_i \in \{0,1\}$ be the value of the majority of responses received for the elements in $\bar{X}_i$. Denote $s = 1 + \mathbb{I}[r_1 = r_2]$ and $\bar{s} = 1 + \mathbb{I}[r_1 \neq r_2]$. The second-to-last element that $\mathcal{A}_p$ selects is $x_{p(s)}$. The last element that $\mathcal{A}_p$ selects is $x_{p(\bar{s})}$ if the response for $x_{p(s)}$ is 0, and $\star$ otherwise. See illustration in Figure 1.

Define the distribution $\mathcal{D}$ such that its marginal $\mathcal{D}_X$ over $\mathcal{X}$ satisfies, for $X \sim \mathcal{D}_X$, $\mathbb{P}[X \in \texttt{base1}] = \mathbb{P}[X \in \texttt{base2}] = 1/3$, $\mathbb{P}[X \in \texttt{perm1}] = \mathbb{P}[X \in \texttt{perm2}] = \mathbb{P}[X = \star] = 1/9$, and in each range $\texttt{base1}, \texttt{base2}, \texttt{perm1}, \texttt{perm2}$, X is conditionally uniform. In addition, for $(X,Y) \sim \mathcal{D}$, $\mathbb{P}[Y = 0 | X = x] = \frac{1}{2}$ for all $x \in \mathcal{X}$. Assume a large enough m such that the probability of a good pool under $\mathcal{D}_X$ is at least $1-\epsilon$ for some very small $\epsilon > 0$. We set $\epsilon = 0$ below for simplicity, noting that any small enough choice will lead to the same lower bound due to rounding down of constants. Thus we assume all pools are good pools.

Consider running the $(q, \mathcal{D})$-equivalent algorithm $\mathcal{A}_s$ with the source distribution $\mathcal{D}$. We derive a lower bound on $\mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}_s, S, q)]$ via the following sequence of arguments.

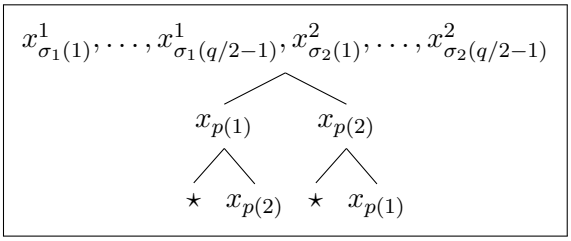

Figure 1: A tree describing the possible selections made by the pool algorithm on a good pool in the proof of Theorem 9.

1. We show that $\mathcal{A}_s$ selects at least $q/4$ elements from each of $\texttt{base1}$ and $\texttt{base2}$ before it selects any element from $\texttt{perm}$.

2. Using the previous claim, we show that there is a positive probability over runs of $\mathcal{A}_s$ on $S \sim D^\infty$ that there is an iteration in the run such that the following hold:

   - Before this iteration at least $q/4$ elements from each of $\texttt{base1}$ and $\texttt{base2}$ have been selected.
   - Conditioned on the run until this iteration, and on the entire S_X, the probability that at some time after this iteration an element from $\texttt{perm1}$ is selected, and an element from $\texttt{perm2}$ is selected at some time after that, is larger than a constant.
   - The same property holds for the reverse order: selecting first an element from $\texttt{perm2}$ and then an element from $\texttt{perm1}$.

3. We conclude that the expected number of iterations of $\mathcal{A}_s$ depends on the probability to have, in a single stream S_X, two elements that are both mapped to permutations in a single small subset. We then show that this implies a lower bound on the number of iterations of $\mathcal{A}_s$.

We start by showing the claim in item 1. Consider a run of $\mathcal{A}_s$ with input $S \sim \mathcal{D}^\infty$. Since the output must be equivalent to an output of $\mathcal{A}_p$ on a good pool, this implies that for $i \in \{1, 2\}$, an element from $\texttt{perm}(i)$ is selected if and only if $s = i$ (where s is defined as done above for the pool algorithm) or the response for an element selected from $\texttt{perm}(\bar{s})$ is 0. Denote the iteration in which an element from $\texttt{perm}$ is first selected by I. Suppose for contradiction that before iteration I, less than $q/4$ elements from either $\texttt{base1}$ or $\texttt{base2}$ have been selected. Thus, for at least one of $\texttt{base1}, \texttt{base2}$, less than half of the elements to be selected from this set during this run are selected until iteration I. It follows that at iteration I the majority of the responses for at least one of the sets $\bar{X}_i$ is yet unknown. Therefore the value of s is also unknown and there is a positive probability that it will be each of $1, 2$. There is also a positive probability that, if an additional element from $\texttt{perm}$ is selected later, its response will be 1. Thus, there is a positive probability that the output of this run does not correspond to any output of $\mathcal{A}_p$, contradicting the equivalence of $\mathcal{A}_s$ and $\mathcal{A}_p$. This proves the claim in item 1.

We now prove the claim in item 2. Let T be the smallest integer such that until iteration T (non-inclusive), at least $q/4$ elements from each of $\texttt{base1}$ and $\texttt{base2}$ have been selected, and no

element from $\texttt{perm}$ has been selected yet. T exists by item 1 above. An element from $\texttt{perm}(s)$, where s depends on the majority of responses in each of $\bar{X}_i$, must be selected before an element from $\texttt{perm}(\bar{s})$ is selected (if at all). Consider the distribution of s, conditioned on the run of $\mathcal{A}_s$ until iteration T and on S_X. We have $s = 2$ if the majority of responses on $\bar{X}_1$ is equal to the majority of responses on $\bar{X}_2$. Out of these responses, at least $q/4 - 1$ of at least one of $\bar{X}_1, \bar{X}_2$, are yet unknown at iteration T. Thus, by Lemma 8, for a large enough q we have that for any S_X, there is a probability of at least $c = 0.68$ over the responses until iteration T that

$$\mathbb{P}[s = 2 \mid \text{the responses until iteration } T, S_X] \in [c', 1 - c'],$$

where $c' = 0.15$. The same holds for $s = 1$.

To finalize the proof of the claim in item 2, observe that if the response for the element selected from $\texttt{perm}(s)$ is 0, then an element from $\texttt{perm}(\bar{s})$ will also be selected later. This occurs with a probability of $\frac{1}{2}$. Consider the event $B_1 :=$ "an element from $\texttt{perm}(1)$ is selected at some point in the run and an element from $\texttt{perm}(2)$ is selected some time later". Let B_2 be the symmetric event. We conclude that with a probability of at least c,

$$\forall i \in \{1, 2\}, \quad \mathbb{P}[B_i \mid \text{the responses until iteration } T, S_X] \geq c'/2. \tag{5}$$

This proves the claim in item 2. Denote the event that Eq. (5) holds by E_p.

We now turn to the third and last part of the proof. Denote by Z_i, for $i \in \{1, 2\}$, the first $q/4$ elements selected by $\mathcal{A}_s$ from $\texttt{base}(i)$. We apply Lemma 5 with $t = q/4, t' = q/2 - 1, l = m/4, d = \sqrt{m \log(q/c)/8}$, to obtain that if the pool algorithm has $Z_i \subseteq \bar{X}_i$ for $i \in \{1, 2\}$, then there is a probability of at least $1 - \frac{q}{2} \exp(-8d^2/(m-q)) \geq 1 - c/2$ that the element that $\mathcal{A}_p$ selects from $\texttt{perm}(i)$ for $i \in \{1, 2\}$ (if it exists) is in a given set of permutations $\Phi(Z_i)$ that satisfies[1]

$$\frac{|\Phi(Z_i)|}{|\Pi_{m/4}|} \leq \left(\frac{8dq}{m}\right)^{q/4}.$$

Thus, this holds also for $\mathcal{A}_s$. Since this holds with probability $1 - c/2$, and Eq. (5) holds with a probability at least c, there is a probability at least $c/2$ that both events hold simultaneously. That is, denoting the event $G_1 :=$ "an element from $\phi(Z_1)$ is selected at some point in the run and an element from $\phi(Z_2)$ is selected some time later" and symmetrically for G_2, we have that with a probability at least $c/2$ over the responses until iteration T and S_X,

$$\forall i \in \{1, 2\}, \quad \mathbb{P}[G_i \mid \text{the responses until iteration } T, S_X] \geq c'/2.$$

Denote the event that this holds E_G. Abbreviate $N_{\mathsf{iter}} := N_{\mathsf{iter}}(\mathcal{A}_s, S, q)$. It follows that

$$\begin{aligned}\mathbb{E}[N_{\mathsf{iter}} \mid S_X, E_G] &\geq c'/2 \cdot \left(\mathbb{E}_{Z_1,Z_2}\left[\mathbb{E}[N_{\mathsf{iter}} \mid G_1, E_G, S_X, Z_1, Z_2] + \mathbb{E}[N_{\mathsf{iter}} \mid G_2, E_G, S_X, Z_1, Z_2]\right]\right) \\ &\geq c'/2 \min_{Z_1,Z_2} \left(\mathbb{E}[N_{\mathsf{iter}} \mid G_1, E_G, S_X, Z_1, Z_2] + \mathbb{E}[N_{\mathsf{iter}} \mid G_2, E_G, S_X, Z_1, Z_2]\right)\end{aligned}$$

Letting $S_X = x_1, x_2, \ldots$, denote by $N_1 \equiv N_1(S_X, Z_1, Z_2)$ the length of the shortest prefix of $x_T, x_{T+1}, \ldots$ that includes an element from $\Phi(Z_1)$ and at some later point an element from

1. We abuse notation and let Φ denote a mapping from $\texttt{base}(i)$ to $\texttt{perm}(i)$ for both of $i \in \{1, 2\}$, using the obvious isomorfisms.

$\Phi(Z_2)$, and by $N_2 \equiv N_2(S_X, Z_1, Z_2)$ the length of the shortest prefix that includes the reverse order. Clearly, the number of iterations under G_i is at least N_i. Therefore

$$\mathbb{E}[N_{\mathsf{iter}} \mid S_X, E_G] \geq c'/2 \min_{Z_1,Z_2}(N_1(S_X, Z_1, Z_2) + N_2(S_X, Z_1, Z_2)).$$

Let $N \equiv N(S_X, Z_1, Z_2) := \max(N_1(S_X, Z_1, Z_2), N_2(S_X, Z_1, Z_2))$. Observe that in the subsequence $x_T, \ldots, x_{T+N}$, there is an element from $\Phi(Z_1)$ followed by one from $\Phi(Z_2)$ some time later, and also the reverse order. It follows there are at least two elements from $\Phi(Z_i)$ for at least one of $i \in \{1, 2\}$ in this subsequence. Let $N' \equiv N'(S_X, Z_1, Z_2)$ be the smallest integer such that $x_T, \ldots, x_{T+N'}$ includes at least one of the two options. Then $N' \leq N$, and so

$$\mathbb{E}[N_{\mathsf{iter}} \mid S_X, E_G] \geq c'/2 \min_{Z_1,Z_2}(N'(S_X, Z_1, Z_2)).$$

Taking expectation over S_X conditioned on E_G, it follows that

$$\mathbb{E}[N_{\mathsf{iter}} \mid E_G] \geq c'/2 \cdot \mathbb{E}_{S_X}[\min_{Z_1,Z_2}(N'(S_X, Z_1, Z_2)) \mid E_G].$$

therefore, since $\mathbb{P}[E_G] \geq c/2$,

$$\mathbb{E}[N_{\mathsf{iter}}] \geq c' \cdot c/4 \cdot \mathbb{E}_{S_X}[\min_{Z_1,Z_2}(N'(S_X, Z_1, Z_2)) \mid \mathbb{E}_G]. \tag{6}$$

We now lower-bound the RHS of this inequality. Let K be an integer. We have

$$\mathbb{E}_{S_X}[\min_{Z_1,Z_2} N' \mid \mathbb{E}_G] \geq K(\mathbb{P}[\min_{Z_1,Z_2} N' \geq K] - (1 - \mathbb{P}[E_G])) \geq K(c/2 - \mathbb{P}[\min_{Z_1,Z_2} N' < K]). \tag{7}$$

We have left to upper-bound $\mathbb{P}[\min_{Z_1,Z_2} N' < K]$ for a non-trivial value of K. Denoting $S_X = x_1, x_2, \ldots$, we have

$$\begin{aligned}\mathbb{P}[\min_{Z_1,Z_2} N' < K] &= \mathbb{P}[\exists i \in \{1,2\}, Z_i \subseteq \texttt{base}(i), j < j' < K, \text{ s.t. } |Z_i| = q/4, x_j, x_{j'} \in \Phi(Z_i)] \\ &\leq 2\mathbb{P}[\exists Z_1 \subseteq \texttt{base1}, j < j' < K, \text{ s.t. } |Z_1| = q/4, x_j, x_{j'} \in \Phi(Z_1)].\end{aligned}$$

The last inequality follows from a union bound and the symmetry between the cases $i = 1$, $i = 2$. For $x \in \texttt{perm1}$, denote $L(x) := \cup_{Z \subseteq \texttt{base1}:|Z|=q/4, x \in \Phi(Z))} \Phi(Z)$. For $x \notin \texttt{perm1}$, let $L(x) := \emptyset$. We have

$$\begin{aligned}&\mathbb{P}[Z_1 \subseteq \texttt{base1}, j < j' < K, \text{ s.t. } |Z_i| = q/4, x_j, x_{j'} \in \Phi(Z_1)] \\ &\leq \mathbb{P}[\exists j < j' < K \text{ s.t. } x_{j'} \in L(x_j)] \\ &\leq K^2 \max_{x \in \texttt{perm1}} \mathbb{P}_{X \sim \mathcal{D}_X}[X \in L(x)] \\ &\leq K^2 \max_{x \in \texttt{perm1}} |L(x)|/|\Pi_{m/4}| \\ &\leq K^2 (8dq/m)^{q/2},\end{aligned}$$

where the last inequality follows from the last part of Lemma 5. Substituting for the value of d, it follows that

$$\mathbb{P}[N < K] \leq K^2 \left(8q^2 \log(q/c)/m\right)^{q/4}.$$

Setting $K = \sqrt{c}/2 \cdot \left(\frac{m}{8q^2 \log(q/c)}\right)^{q/8}$, we get that $\mathbb{P}[N < K] \leq c/4$, therefore by Eq. (6) and Eq. (7),

$$\mathbb{E}[N_{\mathsf{iter}}] \geq c' \cdot c/4 \cdot K(c/4) \geq c' \cdot c^{5/2}/32 \cdot \left(\frac{m}{8q^2 \log(q/c)}\right)^{q/8}.$$

Substituting $c' = 0.15$, $c = 0.68$ and rounding downward, we get the statement of the lemma. ■

The lower bounds above indicate that to improve the dependence on q, one must consider a more restricted class of pool algorithms. This is the topic of the next section.

5. Utility-Based Pool Algorithms

In Section 4 we provided an algorithm with a uniform guarantee on the expected number of iterations. However, this guarantee was exponential q, and as our lower bounds show, this dependence cannot be removed for a general pool algorithm. We now consider a more restricted class of pool algorithms, which we term *utility-based* pool algorithms, and show that it allows stream emulation with an expected number of iterations linear in q. Utility-based pool algorithms are defined in Section 5.1. In Section 5.2 an algorithm that emulates utility-based pool algorithms in a streaming setting is proposed, and it is shown that for this algorithm, the expected number of iterations is at most linear in q, in contrast to the exponential dependence required in the general case. In Section 5.3 two lower bounds are proved, showing that a quasi-linear dependence of the number of iterations on q cannot be avoided when emulating utility-based pool algorithms.

5.1 Defining Utility-Based Algorithms

A common approach for designing pool-based interactive algorithms is to define a utility function that scores each element depending on the history of selected elements and their responses so far (e.g., Seung et al., 1992; Lewis and Gale, 1994; Tong and Koller, 2002; Guo and Greiner, 2007; Golovin et al., 2010b; Guillory and Bilmes, 2010; Golovin and Krause, 2011; Gonen et al., 2013; Cuong et al., 2014). In each round, the algorithm selects the element that maximizes the current utility function. For example, the score can estimate the marginal benefit of selecting an element based on the current information on the source distribution, as gleaned from the previous elements and their responses. We consider black-box emulation for this class of pool-based algorithms.

Formally, a utility-based interactive pool algorithm is defined by a utility function $\mathcal{U}$, of the form $\mathcal{U} : \cup_{n=0}^{\infty} \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^n \to \mathbb{R}_{>0}$. The value of $\mathcal{U}(x, S_{t-1})$ represents the score of element x given history S_{t-1}. The pool algorithm selects, in each round, the element that is assigned the maximal score by the utility function given the history. We assume for simplicity that there are no ties in $\mathcal{U}$ and that all its outputs are positive. The general form of a utility-based interactive pool algorithm for $\mathcal{U}$, denoted $\mathcal{A}_p^{\mathcal{U}}$, is given in Alg. 4.

Our goal when emulating a pool algorithm is not to maximize $\mathcal{U}$ on the selected elements, but to exactly emulate the behavior of the pool algorithm $\mathcal{A}_p^{\mathcal{U}}$. This is because we do not assume any specific relationship between the value of the utility function and the reward of the algorithm. For instance, the utility-based pool algorithm might be empirically successful, while its analysis is not fully understood (e.g. Tong and Koller, 2002).

Algorithm 4 $\mathcal{A}_p^{\mathcal{U}}$

input Elements $x_1, \ldots, x_m$, budget $q < m$.

$S_0 \leftarrow ()$
$M_0 \leftarrow [m]$
for $t = 1 : q$ **do**
  $i_t \leftarrow \text{argmax}_{j \in M_{t-1}} \mathcal{U}(x_j, S_{t-1})$.
  Select x_{i_t}, get y_{i_t}.
  $S_t \leftarrow S_{t-1} \circ (x_{i_t}, y_{i_t})$.
  $M_t \leftarrow M_{t-1} \setminus \{i_t\}$.
end for
Output the set of all pairs in S_q.

5.2 Stream Emulation for Utility-Based Pool Algorithms

We propose a stream algorithm that emulates a utility-based pool algorithm using a black-box solution to the well-known *secretary problem* (Dynkin, 1963; Gilbert and Mosteller, 1966; Ferguson, 1989). We first present this classical riddle and discuss its possible solutions in Section 5.2.1. The stream algorithm and its analysis are presented in Section 5.2.2.

5.2.1 THE SECRETARY PROBLEM WITH A BI-CRITERION

In the classical formulation of the secretary problem, an algorithm sequentially observes a stream of n different real numbers, and selects a single number. The goal of the algorithm is to select the unique maximal number out of the n numbers, but it can only select a number immediately after it is observed, before observing the rest of the numbers. It is assumed that the n numbers in the stream are unknown and selected by an adversary, but their order of appearance is uniformly random.

The classical goal is to select the maximal number with a maximal probability, where n is known to the algorithm. This task can be optimally solved by a simple deterministic algorithm, achieving a success probability which approaches $1/e$ in the limit of $n \to \infty$ (Dynkin, 1963; Gilbert and Mosteller, 1966; Ferguson, 1989). The optimal algorithm observes the first $t(n)$ numbers, and then selects the next observed number which is at least as large as the maximal in the first $t(n)$. The limit of $t(n)/n$ for $n \to \infty$ is $1/e$.

In the next section we show how any secretary problem strategy can be used to emulate a utility-based pool algorithm in a streaming setting. However, while the classical formulation of the secretary problem is concerned only with the probability of success, in our context an additional criterion is important. Thus for a given strategy, we consider the following two criteria:

1. The probability of success—the probability that the strategy selects the maximal element out of n, assuming a random ordering of the input sequence. This is the single criterion of the classical secretary problem.

2. The *success/select ratio*—the conditional probability that a number is maximal, given that the strategy selected it.

For the second criterion to be meaningful, we allow the strategy to decide not to select any number. For instance, in the classical optimal solution to the secretary problem, the strategy should not select

any number if after the first $t(n)$ numbers, all the other numbers are smaller than the maximal in the first $t(n)$. As we see in Section 5.2.2, we would optimally like the strategy to have both a high success probability and a high success/select ratio. This is because a high success probability leads to a lower number of observed elements N_{iter} required by the stream algorithm, while a high success/select ratio leads to a lower number of selections of elements N_{sel}.

The classical solution maximizes the success probability, but does not optimize for the success/select ratio. Another extreme can be found in the following strategy, which sets the success/select ratio at one, at the expense of the success probability: Wait for the last number, and then select it only if it is the maximal out of all the observed items. This strategy never selects a non-maximal number, hence its success/select ratio is one, however its success probability is only $1/n$. Denote this strategy SecPr[last].

In general, consider strategies of the following form: For a given $M \geq 1$, observe the first $M-1$ numbers. Then select the first number which is larger than the maximal in the first $M-1$ numbers. If no such number is observed, avoid selecting a number. Denote this strategy SecPr$[M]$. Denote by $p_s(M)$ the success probability of this strategy, and by $\mathrm{sr}(M)$ its success/select ratio. The analysis of $p_s(M)$ (see Ferguson, 1989) gives $p_s(1) = 1/n$, and for $M > 1$,

$$p_s(M) = \frac{M-1}{n}(H_{n-1} - H_{M-1}),$$

where H_i is the i'th harmonic number $H_i := \sum_{j=1}^{i} \frac{1}{i}$.

To calculate $\mathrm{sr}(M)$, note that the event that some number is selected occurs exactly when the maximal number is in the last $n-(M-1)$ observed numbers. Therefore the probability of selecting a number is $1 - \frac{M-1}{n}$, and so

$$\mathrm{sr}(M) = \frac{p_s(M)}{1 - \frac{M-1}{n}}.$$

Setting $\alpha := \frac{M-1}{n}$ and considering large n, we have $H_{n-1} - H_{M-1} \approx \ln(\frac{n-1}{M-1}) \approx \ln(1/\alpha)$. Therefore

$$p_s(M) \approx \alpha \log(1/\alpha), \quad \text{and} \quad \mathrm{sr}(M) \approx \frac{\alpha \log(1/\alpha)}{1-\alpha}.$$

For $\alpha \in (0,1)$, $p_s(M)$ is concave with a single maximum at $\alpha = 1/e$, while $\mathrm{sr}(M)$ is monotonic increasing. Since we wish for both criteria to be large, the Pareto frontier includes only and all the solutions with $\alpha \geq 1/e$. See Figure 2 and Figure 3 for the trade-off between $p_s(M)$ and $\mathrm{sr}(M)$ as α changes between $1/e$ and 1.

In the implementation of the stream algorithm shown in the following section, the secretary problem strategy is executed multiple times, for sequences of different lengths. Given any secretary problem strategy for length m, we apply it for any length up to m, using the procedure in Alg. 5.

When running SecPrVar, the subset I should be drawn uniformly at random from all possible subsets. This way, clearly the success probability and the success/select ratio for SecPrVar, for any sequence of positive numbers of length $m' \leq m$, are at least as high as those for SecPr with a sequence of length m.

We also consider stream emulation using precognitive stream algorithms. In this case, the secretary problem can easily be perfectly solved: it may be assumed that the algorithm knows in advance all the numbers in the sequence and thus it can select the maximal number with probability 1. We

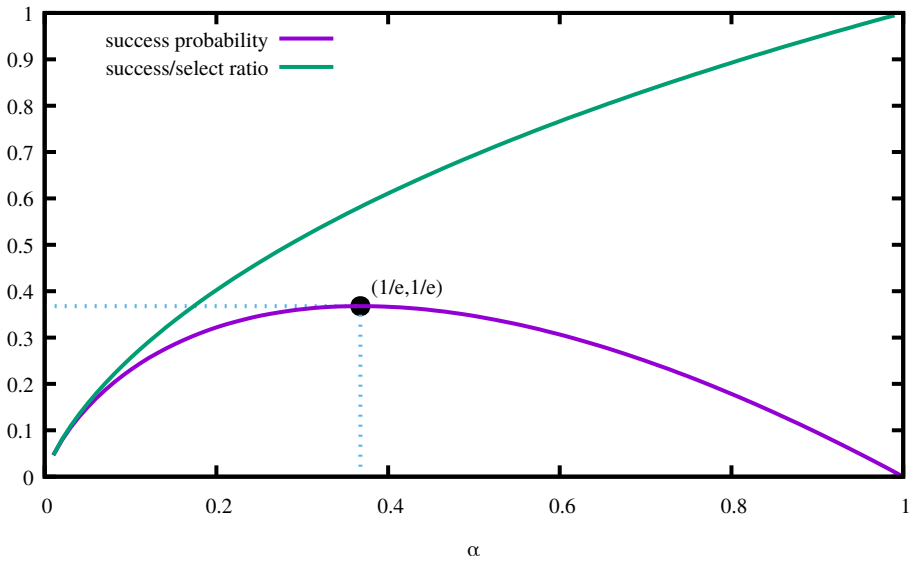

Figure 2: The success probability and the success/select ratio for large n, as a function of α.

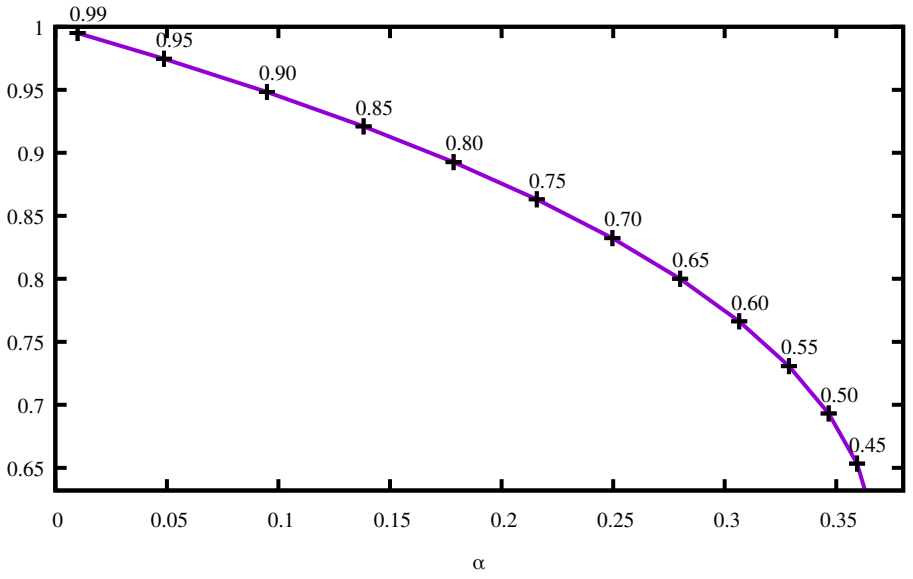

Figure 3: The trade-off between the success probability and the success/select ratio for large n, in the Pareto frontier of α values, for strategies of the form SecPr$[\alpha n]$. On the Pareto frontier α ranges between $1/e$ (largest success probability) and 1 (smallest success probability). The labels on the points indicate the value of α.

Algorithm 5 SecPrVar(m, m', I,SecPr): Secretary problem for sequences of variable length

input Integers $m, m' \leq m$, $I \subseteq [m]$, a secretary problem strategy SecPr for sequence length m.

$R \leftarrow ()$
for $i = 1 : m$ **do**
  if $i \notin I$ **then**
    $R \leftarrow R \circ (0)$ # Add a dummy zero to sequence
  else
    Get next number r from input
    $R \leftarrow R \circ (r)$
    if r should be selected according to SecPr on R **then**
      Select r and terminate.
    end if
  end if
end for

will denote this strategy $SecPr[\text{precog}]$. It has success probability and success/select ratio both equal to 1.

We next present and analyze the stream emulation algorithm for utility-based pool algorithms, which uses a secretary problem strategy via SecPrVar.

5.2.2 THE STREAM EMULATION ALGORITHM

We propose the stream emulation algorithm $\mathcal{A}_s^{\mathcal{U}}$, listed in Alg. 6. This algorithm repeatedly applies a secretary problem strategy to decide which elements to select. Because the secretary problem strategy might fail to select the maximal number, repeated applications of the strategy might be necessary. This trial-and-error approach means that $\mathcal{A}_s^{\mathcal{U}}$ might select more than q elements. However, the expected number of selected elements is a constant factor over q. The trade-off between the number of iterations and number of selected elements is controlled by the probability of success and the success/select ratio of the selected strategy SecPr, which is provided to $\mathcal{A}_s^{\mathcal{U}}$ as input.

$\mathcal{A}_s^{\mathcal{U}}$ is equivalent to $\mathcal{A}_p^{\mathcal{U}}$, as Theorem 10 below shows. In order to guarantee this equivalence, $\mathcal{A}_s^{\mathcal{U}}$ never selects an element that could not have been in a pool in which the previous elements were selected. This is achieved by discarding such elements in each round: The set $\mathcal{X}_i$ is the set of elements that are allowed in round i, and is defined to include only elements that could not have been selected by the pool algorithm before round i. To bound the expected number of iterations, we show in Theorem 11 that the probability mass of $\mathcal{X}_i$ can be controlled in expectation, which leads to a bound on the expected number of discarded elements.

Theorem 10 *For any utility function $\mathcal{U}$, any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, any integer m and $q \leq m$, $\mathcal{A}_s^{\mathcal{U}}$ is $(q, \mathcal{D})$-equivalent to $\mathcal{A}_p^{\mathcal{U}}$.*

Proof Consider the probability space defined by $S \sim \mathcal{D}^m$ and $S' \sim \mathcal{D}^\infty$, where S, S' are independent. We prove the equivalence by showing that for any $j \in [q]$ and $L_j = ((x_{i,k_i}, y_{i,k_i}))_{i \in [j]}$ that could have been selected by the pool algorithm,

$$d\mathbb{P}[\mathsf{pairs}_p(S, j+1) \mid \mathsf{pairs}_p(S, [j]) = L_j] = d\mathbb{P}[\mathsf{pairs}_s(S', j+1) \mid \mathsf{pairs}_s(S', [j]) = L_j].$$

Algorithm 6 $\mathcal{A}_s^{\mathcal{U}}$ Stream-emulation for utility-based pool algorithms

input Integers m, $q \leq m$, a secretary problem strategy SecPr for sequences of length m.

$L_0 \leftarrow ()$
$\mathcal{X}_1 = \mathcal{X}$
for $i = 1 : q$ **do**
  repeat
    Draw a random subset $I \subseteq [m]$ of size m'
    for $j = 1 : m - i + 1$ **do**
      Repeatedly draw elements from $\mathcal{D}_X$, until drawing an element in $\mathcal{X}_i$.
      Denote it $x_{i,j}$, and let $r_{i,j} \leftarrow \mathcal{U}(x_{i,j}, L_{i-1})$.
      if SecPrVar$(m, m - i + 1, I$,SecPr$)$ would select the last element in the sequence prefix $r_{i,1}, \ldots, r_{i,j}$ **then**
        $k \leftarrow j$
        Select $x_{i,k}$, get its response $y_{i,k}$.
      end if
    end for
  until $r_{i,k} = \max\{r_{i,1}, \ldots, r_{i,m-i+1}\}$.
  $k_i \leftarrow k$
  $L_i \leftarrow L_{i-1} \circ (x_{i,k_i}, y_{i,k_i})$.
  $\mathcal{X}_{i+1} \leftarrow \{x \in \mathcal{X}_i \mid \mathcal{U}(x, L_{i-1}) < \mathcal{U}(x_{i,k_i}, L_{i-1})\}$
end for
Output the set of pairs in L_q.

For a given L_j, denote by $\mathcal{D}_{j+1}$ the distribution generated by drawing $(X, Y) \sim \mathcal{D}$ conditioned on $X \in \mathcal{X}_{j+1}$, where $\mathcal{X}_{j+1}$ depends on L_j. Denote by $\mathcal{G}$ all the finite sequences of pairs such that when the optimal secretary problem solution is applied to the sequence, it succeeds. That is, the optimal value under the score $(x, y) \to \mathcal{U}(x, L_j)$ is indeed selected. From the definition of $\mathcal{A}_{\mathcal{U}}^s$, we have

$$d\mathbb{P}[\mathsf{pairs}_s(S', j+1) \mid \mathsf{pairs}_s(S', [j]) = L_j] = d\mathbb{P}_{\bar{S} \sim \mathcal{D}_{j+1}^{m-j}}[\operatorname*{argmax}_{(x,y) \in \bar{S}} \mathcal{U}(x, L_j) \mid \bar{S} \in \mathcal{G}].$$

For a given sequence $\bar{S} = ((\bar{x}_i, y_i))_{i \in [m-j]}$, let $\sigma(\bar{S}) : [m-j] \to [m-j]$ be a permutation such that for all $i \leq m - j$, $\bar{x}_{\sigma(i)} \leq \bar{x}_{\sigma(i+1)}$. The success of the optimal secretary problem algorithm depends only on the ordering of ranks in its input sequence, hence there is a set of permutations $\mathcal{G}'$ such that $\bar{S} \in \mathcal{G}$ if and only if $\sigma(\bar{S}) \in \mathcal{G}'$. Now, $\operatorname{argmax}_{(x,y) \in \bar{S}} \mathcal{U}(x, L_j)$ depends only on the identity of pairs in $\bar{S}$, while $\sigma(\bar{S})$ depends only on their order. Since the elements in $\bar{S}$ are i.i.d., these two properties are independent. Therefore

$$d\mathbb{P}_{\bar{S} \sim \mathcal{D}_{j+1}^{m-j}}[\operatorname*{argmax}_{(x,y) \in \bar{S}} \mathcal{U}(x, L_j) \mid \bar{S} \in \mathcal{G}] = d\mathbb{P}_{\bar{S} \sim \mathcal{D}_{j+1}^{m-j}}[\operatorname*{argmax}_{(x,y) \in \bar{S}} \mathcal{U}(x, L_j)].$$

Therefore

$$
\begin{aligned}
&d\mathbb{P}[\mathsf{pairs}_s(S', j+1) \mid \mathsf{pairs}_s(S', [j]) = L_j] \\
&= d\mathbb{P}_{\bar{S}\sim\mathcal{D}^{m-j}_{j+1}}[\operatorname*{argmax}_{(x,y)\in\bar{S}} \mathcal{U}(x, L_j)] \\
&= d\mathbb{P}_{\hat{S}\sim\mathcal{D}^{m-j}}[\operatorname*{argmax}_{(x,y)\in\hat{S}} \mathcal{U}(x, L_j) \mid \hat{S} \subseteq (\mathcal{X}_{j+1} \times \mathcal{Y})^{m-j}] \\
&= d\mathbb{P}_{\hat{S}\sim\mathcal{D}^{m-j}}[\operatorname*{argmax}_{(x,y)\in\hat{S}} \mathcal{U}(x, L_j) \mid \forall (x,y) \in \hat{S}, i \in [j],\ \mathcal{U}(x, L_{i-1}) < \mathcal{U}(x_{i,k_i}, L_{i-1})] \\
&= d\mathbb{P}_{\hat{S}\sim\mathcal{D}^{m-j}}[\operatorname*{argmax}_{(x,y)\in\hat{S}} \mathcal{U}(x, L_j) \mid \mathsf{pairs}_p(L_j \circ \hat{S}, [j]) = L_j] \\
&= d\mathbb{P}_{S\sim\mathcal{D}^{m}}[\operatorname*{argmax}_{(x,y)\in S\setminus L_j} \mathcal{U}(x, L_j) \mid \mathsf{pairs}_p(S, [j]) = L_j] \\
&= d\mathbb{P}_{S\sim\mathcal{D}^{m}}[\mathsf{pairs}_p(S)(j+1) \mid \mathsf{pairs}_p(S, [j]) = L_j].
\end{aligned}
$$

Here L_i is the prefix of length i of L_j. Since this equality holds for all $j \in [q-1]$, $d\mathbb{P}[\mathsf{pairs}_s(S', [q])] = d\mathbb{P}[\mathsf{pairs}_p(S, [q])]$. ■

The following theorem gives the expected number of selected elements and the expected number of observed elements used by $\mathcal{A}^{\mathcal{U}}_s$. Both of these values depend on the properties of the secretary problem strategy SecPr that $\mathcal{A}^{\mathcal{U}}_s$ receives as input.

Theorem 11 *Suppose that $\mathcal{A}^{\mathcal{U}}_s$ is run with $m, q \leq m$ and a secretary problem strategy SecPr for sequences of size m with success probability p_s and success/select ratio* sr. *Then, for any utility function $\mathcal{U}$ and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$*

$$\mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{sel}}(\mathcal{A}^{\mathcal{U}}_s, S, q)] = q/\mathrm{sr}$$

and

$$\mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}^{\mathcal{U}}_s, S, q)] = mq/p_s.$$

To observe the implications of these upper bounds, consider for instance the classical secretary problem strategy SecPr$[t(n)]$, as defined in Section 5.2.1, which has $p_s \approx 1/e$ and $\mathrm{sr} \approx \frac{1}{e(1-1/e)}$ for large m. It follows that for $m \to \infty$, the expected number of selected elements by $\mathcal{A}^{\mathcal{U}}_s$ is eq, and the expected number of observed elements is eqm. Other choices of parameters for the secretary problem lead to other points on the Pareto frontier in Figure 3. All the points on the Pareto frontier give a constant factor over q for the expected number of selected elements, and a factor linear in q over m for the expected number of iterations. To get an algorithm that selects exactly q elements, one can use the strategy SecPr[last], which gives $p_s = 1/m$ and $\mathrm{sr} = 1$, leading to an expected number of iterations of qm^2. In the precognitive setting, one can use $SecPr$[precog], which gives an expected number of selections exactly q, and an expected number of iterations mq.

Proof [of Theorem 11] Call a full run of the loop starting at step 6 an attempt for the i'th element. In each attempt for the i'th element, $m - i + 1$ elements from $\mathcal{X}_i$ are observed. The expected number of element selection attempts for each successful element selection is $\frac{1}{\mathrm{sr}}$. Thus,

$$\mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{sel}}(\mathcal{A}^{\mathcal{U}}_s, S, q)] = \frac{1}{\mathrm{sr}} \cdot q.$$

For the second part of the statement, we need to bound $\mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}_s^{\mathcal{U}}, S, q)]$. The total expected number of attempts of running the secretary problem for each successful selection of an element is $\frac{1}{p_s}$, and the expected number of elements from $\mathcal{X}_i$ observed in each attempt is $m-i+1$. Thus the expected number of elements from $\mathcal{X}_i$ observed until x_i is selected is $\frac{1}{p_s}\cdot(m-i+1)$. However, drawing a single element from $\mathcal{X}_i$ might require several draws from $\mathcal{X}$. To bound the total expected number of observed elements, we now consider the definition of $\mathcal{X}_i$.

Denote by f_i the utility function $\mathcal{U}(\cdot, L_{i-1})$. Let $x_i := x_{i,k_i}$, be the i'th element added to L_i. Then $\mathcal{X}_i = \{x \in \mathcal{X}_{i-1} \mid f_{i-1}(x) \leq f_{i-1}(x_{i-1})\}$. Consider the probability space defined by the input to the stream algorithm $S \sim \mathcal{D}^\infty$, and let $Z_i, Z_i' \sim \mathcal{D}_X$ for $i \in [q]$ such that these random variables and S are all independent. Denote

$$p(\alpha, i) := \mathbb{P}[f_i(Z_i) \leq \alpha \mid Z_i \in \mathcal{X}_i].$$

$p(\alpha, i)$ is a random variable since $\mathcal{X}_i$ depends on S. Let $U_i := p(f_i(Z_i'), i)$. Since we assume no ties in $\mathcal{U}$, and no single x has a positive probability in $\mathcal{D}_X$, then conditioned on $\mathcal{X}_i$, U_i is distributed uniformly in $[0,1]$. Hence $U_1, \ldots, U_q$ are statistically independent.

For $i > 1$, define the random variable $M_i := p(f_{i-1}(x_{i-1}), i-1)$. Then $M_i = \mathbb{P}[\mathcal{X}_i]/\mathbb{P}[\mathcal{X}_{i-1}]$. The expected number of elements that need to be drawn from $\mathcal{D}$ to get a single element from $\mathcal{X}_i$ is $1/\mathbb{P}[\mathcal{X}_i] = (\prod_{j=1}^i M_j)^{-1}$. Therefore,

$$\mathbb{E}[N_{\mathsf{iter}}(\mathcal{A}_s^{\mathcal{U}}, S, q) \mid M_2, \ldots, M_q] = \sum_{i=1}^q \frac{p_s^{-1}\cdot(m-i+1)}{\prod_{j=1}^i M_j}.$$

The element x_i maximizes the function $x \mapsto f_i(x)$ over $m-i+1$ independent draws of elements x from $\mathcal{D}_X$ conditioned on $x \in \mathcal{X}_i$, hence it also maximizes $x \mapsto p(f_i(x), i)$. Therefore, for $i > 1$, M_i is the maximum of $m-i+2$ independent copies of U_i, hence $\mathbb{P}[M_i \leq p] = p^{m-i+2}$. Hence

$$d\mathbb{P}[M_2, \ldots, M_q](p_2, \ldots, p_q)/dp_2 \cdot \ldots \cdot dp_q = \prod_{i=2}^q d\mathbb{P}[M_i \leq p_i]/dp_i = \prod_{i=2}^q (m-i+2)p_i^{m-i+1}.$$

We have

$$\begin{aligned}
\mathbb{E}[N_{\mathsf{iter}}(\mathcal{A}_s^{\mathcal{U}}, S, q)] &= \int_{M_2=0}^1 \cdots \int_{M_q=0}^1 \mathbb{E}[N_{\mathsf{iter}}(\mathcal{A}_s^{\mathcal{U}}, S, q) \mid M_1, \ldots, M_q] d\mathbb{P}[M_1, \ldots, M_q] \\
&= p_s^{-1} \int_{M_2=0}^1 \cdots \int_{M_q=0}^1 \sum_{i=1}^q \frac{m-i+1}{\prod_{j=1}^i M_j} \prod_{l=2}^q (m-l+2) M_l^{m-l+1} dM_l \\
&= p_s^{-1} \sum_{i=1}^q (m-i+1) \int_{M_2=0}^1 \cdots \int_{M_q=0}^1 \prod_{l=2}^i (m-l+2) M_l^{m-l} dM_l \\
&\qquad \cdot \prod_{l=i+1}^q (m-l+2) M_l^{m-l+1} dM_l,
\end{aligned}$$

Therefore

$$\begin{aligned}\mathbb{E}[N_{\mathsf{iter}}(\mathcal{A}_s^{\mathcal{U}}, S, q)] &= p_s^{-1} \sum_{i=1}^{q} (m-i+1) \prod_{l=2}^{i} \frac{m-l+2}{m-l+1} \\ &= p_s^{-1} \sum_{i=1}^{q} (m-i+1) \cdot \frac{m}{m-i+1} = mq/p_s.\end{aligned}$$

This concludes the proof. ■

5.3 Lower Bounds

The following lower bounds show that the number of iterations required by any stream emulator for utility-based algorithms must be at least quasi-linear in q. We provide two results. The first result considers a stream algorithm that selects exactly q elements, and it allows q to be large with respect to m. The second result considers stream algorithms that are allowed to select more than q elements. In this case, the lower bound is smaller, but it is reduced only linearly in the number of selections. It follows that a constant factor increase in the number of selections cannot overcome the quasi-linear dependence on q in the number of iterations. The proofs of the lower bounds are based on constructing a utility function which in effect allows only one set of selected elements for a given distribution, and forces the stream algorithm to select them in the same order as the pool algorithm. The first lower bound considers stream emulation with exactly q selections. The bound holds for both standard streaming algorithms and precognitive streaming algorithms.

Theorem 12 *For any integer m, $q \leq m/2$, there exists a utility-based pool algorithm, and a marginal $\mathcal{D}_X$, such that any stream algorithm $\mathcal{A}_s$ which is $(q, \mathcal{D})$ equivalent to the pool algorithm for all $\mathcal{D} \in \mathrm{DS}(\mathcal{D}_X)$, and selects only q elements, has*

$$\exists \mathcal{D} \in \mathrm{DS}(\mathcal{D}_X), \mathbb{E}_{S \sim \mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}_s, S, q)] \geq \frac{q}{32} \left\lfloor \frac{m}{2\log(4q)} \right\rfloor .$$

The result holds even if $\mathcal{A}_s$ is a precognitive stream algorithm.

Proof Let $n = \left\lfloor \frac{m}{2\log(4q)} \right\rfloor$, and let $\mathcal{D}_X$ be a uniform distribution over $\mathcal{X} = \{a_i \mid i \in [n]\}$. Assume $\mathcal{Y} = \{0, 1\}$. A pool of size m then includes all elements in $A := \{a_i \mid i \in [2q-1]\}$ with a probability of at least $\alpha \geq 1 - (2q-1)\exp(-m/n) \geq 1 - \frac{1}{4q}$.

Consider a utility function $\mathcal{U}$ such that given a history of the form $((a_1, y_1), \ldots, (a_t, y_t))$ for $t \in [q-1]$, assigns a maximal score in $\mathcal{X}$ to a_{t+1} if $\forall i \leq t, y_i = 0$, and a maximal score to a_{q+t} if $\exists i \leq t, y_i = 1$. In addition, if the history is $((a_1, y_1), \ldots, (a_t, y_t), (a_{q+t}, y_{t+1}), \ldots, (a_{q+t'}, y_{t'+1}))$ for some $t \in [q-1], t' \in [q-2]$, then $\mathcal{U}$ assigns a maximal score to $a_{q+t'+1}$. Then, in a pool that includes all elements $a_1, \ldots, a_{2q-1}$, the pool algorithm based on $\mathcal{U}$ behaves as follows: In every round, if all selected elements so far received the response 0, it selects at round t the element a_t. Otherwise, it selects the element a_{q+t} and then continues to select $a_{q+t+1}, \ldots, a_{2q-1}$.

Let $\mathcal{D}_0$ be a distribution in which the response is deterministically zero. If the distribution is $\mathcal{D}_0$, $\mathcal{A}_s$ selects $Z_0 = \{a_1, \ldots, a_q\}$ with a probability at least α. Denote $\mathcal{D}_t$ for $t \in [q]$, in which the

response is deterministically zero for $X \in \{a_1, \ldots, a_q\} \setminus \{a_t\}$ and 1 for a_t. For this distribution, the algorithm must select the elements in $Z_t = \{a_1, \ldots, a_t, a_{q+t}, \ldots, a_{2q-1}\}$ with a probability at least α.

Consider the probability space defined by the input sequence $S \sim \mathcal{D}_0^\infty$ and the randomness of $\mathcal{A}_s$. Let E_0 be the event in this space that $\mathcal{A}_s$ selects $\{a_1, \ldots, a_q\}$, and let $\bar{E}_0$ be the event in this space that $\mathcal{A}_s$ selects $(a_1 \ldots, a_q)$ in order. We have $\mathbb{P}[E_0] \geq \alpha$. Denote $\beta = \mathbb{P}[\bar{E}_0]$. We show a lower bound on β.

Let T be a random variable in the same probability space, such that $T = 0$ if E_0 does not hold. If E_0 does hold, then T is the smallest round in which the algorithm selects some $a_{t'}$, for $t' > T$, or $T = 0$ if no such round exists. If $\mathcal{A}_s$ selects $\{a_1, \ldots, a_q\}$ but not in order, then $T \in [q]$. Therefore, $\mathbb{P}[T \in [q]] \geq \alpha - \beta$, hence there exists some $t^* \in [q]$ such that $\mathbb{P}[T = t^*] \geq (\alpha - \beta)/q$. Now, consider the distribution $\mathcal{D}_{t^*}$. Define a sequence of pairs $\gamma(S)$ such that S and $\gamma(S)$ have the same elements in the same order, and the responses in $\gamma(S)$ are determined by $\mathcal{D}_{t^*}$ instead of by $\mathcal{D}_0$. Clearly, $\gamma(S)$ is distributed according to $\mathcal{D}_{t^*}^\infty$. Consider a run of the algorithm on S in which E_0 holds, and a parallel run (with the same random bits) on $\gamma(S)$. The algorithm selects the same elements for both sequences until the T'th selection, inclusive. By the definition of T, the T'th selection is some element in $\{a_{T+1}, \ldots, a_q\}$. If $T = t^*$, then an element not in Z_{t^*} is selected in round T. But this same element would be selected by $\mathcal{A}_s$ in the parallel run on $\gamma(S)$. Since under $\gamma(S) \sim \mathcal{D}_{t^*}^\infty$, $\mathcal{A}_s$ selects exactly the set Z_{t^*} with a probability of at least α, and so we have $\mathbb{P}[T = t^*] \leq 1 - \alpha$. This holds also for the precognitive stream algorithm since it is required from the emulation of the pool algorithm. It follows that $(\alpha - \beta)/q \leq \mathbb{P}[T = t^*] \leq 1 - \alpha$. Hence $\beta \geq \frac{1}{2}$.

Let W_i be the number of elements that $\mathcal{A}_s$ observes after selecting element $i-1$, until observing the next element. Let $X_1, X_2, \ldots$ be the sequence of elements observed after selecting the first $i-1$ elements, and for integers j, let $B_j = \mathbb{I}[X_j = a_i]$. $B_1, B_2, \ldots$ are independent Bernoulli random variables with $\mathbb{P}[B_j = 1] = 1/n$, and $\mathbb{P}[B_{W_i} = 1] \geq \mathbb{P}[\bar{E}_0] = \beta \geq \frac{1}{2}$. By Lemma 2, if $\frac{1}{n} \leq \frac{1}{32}$, $\mathbb{E}[W_i] \geq \frac{n}{16}$. It follows that the expected number of iterations for making q selections is at least $\frac{qn}{16}$ if $n \geq 32$. Since it is also at least q, a lower bound of $\frac{qn}{32} = \frac{q}{32}\left\lfloor \frac{m}{2\log(4q)} \right\rfloor$ holds for all n. This analysis holds also for the precognitive stream algorithm, since it also must select the same set of elements in the given order. ■

The second lower bound considers stream emulation with possibly more than q selections. This lower bound also holds for both standard and precognitive stream algorithms.

Theorem 13 *For any integers m, q such that $m \geq 2^{q+1}\log(2q)$ there exists a utility-based pool algorithm, and a marginal $\mathcal{D}_X$, such that for any $\beta \geq 1$, any stream algorithm $\mathcal{A}_s$ which is $(q, \mathcal{D})$ equivalent to the pool algorithm for all $\mathcal{D} \in \text{DS}(\mathcal{D}_X)$, and selects at most βq elements, has*

$$\exists \mathcal{D} \in \text{DS}(\mathcal{D}_X), \mathbb{E}_{S\sim\mathcal{D}^\infty}[N_{\mathsf{iter}}(\mathcal{A}_s, S, q)] \geq \frac{q}{256\beta} \cdot \left\lfloor \frac{m}{\log(2q)} \right\rfloor .$$

This result holds also for precognitive stream algorithms.

Proof Let $n = \left\lfloor \frac{m}{\log(2q)} \right\rfloor$, and let $\mathcal{D}_X$ be a uniform distribution over $\mathcal{X} = \{a_v \mid v \in \{0,1\}^t, t \in [q-1]_0\} \cup \mathcal{X}'$, where $\mathcal{X}'$ includes arbitrary elements so that $|\mathcal{X}| = n$. Note that $\mathcal{X}$ includes $a_{()}$, which stands for a zero-length vector. Assume $\mathcal{Y} = \{0, 1\}$. Let $v \in \{0,1\}^q$ be a binary vector of length

q, and let $v(1:t)$ be its prefix of length $t \in [q-1]_0$. Consider distributions $\mathcal{D}_{v,f}$ for $v \in \{0,1\}^q$ and $f : \cup_{t\in[q-1]_0}\{0,1\}^t \to \{0,1\}$, such that $\mathcal{D}_v$ has a uniform marginal over $\mathcal{X}$, and for every $t \in [q-1]_0$, $\mathbb{P}[Y = v(t+1) \mid X = a_{v(1:t)}] = 1$. In other words, the label of any element which is a prefix of v is the next bit in v. In addition, for any other element a_b, $\mathbb{P}[Y = 1 \mid X = a_b] = f(a_b)$. Denote the elements representing prefixes of v by $A_v := \{a_{v(1:t)} \mid t \in [q-1]_0\}$.

Assume a utility function $\mathcal{U}$ such that for any $b \in \{0,1\}^t$ of length $t < q$, if the last element selected so far was a_b and its response was y, the function assigns the maximal score to $a_{b\circ y}$. Under this utility function, if the distribution is $\mathcal{D}_{v,f}$ and $A_v \subseteq_\pi S_X$, then the pool algorithm selects exactly the elements in A_v. This happens with probability at least $1 - q\exp(-m/n) \geq \frac{1}{2}$.

Let $\mathcal{A}_s$ be a stream-based algorithm which is equivalent to the pool algorithm $\mathcal{A}_p^{\mathcal{U}}$. Let V be a random variable drawn uniformly at random from $\{0,1\}^q$, and let F be a random variable drawn uniformly over all functions $f : \cup_{t\in[q-1]_0}\{0,1\}^t \to \{0,1\}$. Consider the probability space defined by $V, F, S \sim \mathcal{D}_{V,F}^{\infty}$, and the run of $\mathcal{A}_s$ on S. Let I_t be a random variable that is equal to 1 if the t'th selection of $\mathcal{A}_s$ is an element from A_V that has not been selected in previous rounds. Since $\mathcal{A}_s$ is equivalent to $\mathcal{A}_p^{\mathcal{U}}$, then $\mathbb{P}[\sum_{t=1}^{\beta q} I_t = q] \geq \frac{1}{2}$. By the reverse Markov's inequality, there are at least $q/4$ rounds t such that $\mathbb{E}[I_t] \geq \frac{1}{4\beta}$. This holds also for a precognitive stream algorithm due to its emulation of the pool algorithm.

Consider one such round t. Let $Z = \mathsf{pairs}_s(S, [t-1])$. We have $\frac{1}{4\beta} \leq \mathbb{E}[I_t] = \mathbb{E}_Z[\mathbb{E}[I_t \mid Z]]$. By the reverse Markov's inequality, with a probability of at least $\frac{1}{8\beta}$, the value of Z is some z that satisfies $\mathbb{E}[I_t \mid Z = z] \geq \frac{1}{8\beta}$. Let z such that $\mathbb{E}[I_t \mid Z = z] \geq \frac{1}{8\beta}$. Denote by B_t the (random) index of the element a_{B_t} selected in round t. By the reverse Markov's inequality, the probability that $B_t = b$ for some b such that $\mathbb{P}[I_t = 1 \mid Z = z, B_t = b] \geq \frac{1}{16\beta}$ is at least $\frac{1}{16\beta}$. In other words,

$$P(b,t) := \mathbb{P}[a_b \in A_V \mid Z = z, B_t = b] \geq \frac{1}{16\beta}.$$

We now upper bound the number of vectors b such that this inequality holds and they do not already appear in Z. Call such vectors "admissible".

Let Z', z' be the prefixes of length $t-2$ of Z, z respectively, and denote $P(b, t-1) := \mathbb{P}[a_b \in A_V \mid Z' = z']$. Let (w, y) be the last pair in z, and assume without loss of generality that (w, y) is absent from z'. The relationship between $P(b, t-1)$ and $P(b, t)$ can be described based on the relationships between w, y, b, by distinguishing into cases.

1. b is a (weak) prefix of w.
2. $w \circ y$ is a (weak) prefix of b.
3. $w \circ (1-y)$ is a (weak) prefix of b.
4. Neither of the cases above hold. In this case, b and w are incompatible.

In cases 1 and 4, the distribution of y is uniform on $\{0,1\}$ conditioned on b being a prefix of V. It is uniform also conditioned on b not being a prefix of V. Hence, in these cases conditioning on (w, y) does not affect the probability that b is a prefix of V, hence $P(b,t) = P(b,t-1)$. In case 3, $P(b,t) = 0$. We are left with case 2, in which $w \circ y$ is a (weak) prefix of b. In this case we have that $P(b,t) = 2P(b,t-1)$. By induction, we can conclude that if $P(b,t) \neq 0$, then $P(b,t) = 2^{-|b|} \cdot 2^{N(b)}$, where $N(b)$ is the number of pairs in Z of the form (w, y) where $w \circ y$ is a prefix of b.

To bound the number of admissible vectors, let us show a one-to-one mapping between such vectors and binary vectors with a bounded length. For any such b, we have $N(b)-|b| = \log_2(P(b,t)) \geq \log_2(\frac{1}{16\beta})$. Denote $k = \lceil \log_2(16\beta) \rceil$. Then $k \geq |b| - N(b)$. b can be encoded as a vector of length at most k as follows: Go over the bits in b from first to last. For each bit i in b, copy it to the output vector in the next available location only if the prefix $b(1:(i-1))$ does not match some w such that $(w,y) \in Z$. Clearly, $N(b)$ bits are skipped this way, so that the total number of bits in the output is no more than k. Moreover, the mapping is one-to-one, since the only possible ambiguity in decoding is how many bits to decode; but whenever $(w,y) \in Z$, b cannot be equal to w, since w has appeared in Z. Therefore the decoding should stop only when it is not possible to infer more bits. It follows that the number of admissible vectors is at most $2^{k+1} - 1 \leq 64\beta$.

Therefore, the probability of observing such an element in each iteration after round $t-1$ is at most $64\beta/|\mathcal{X}|$. Since there are $q/4$ such iterations, the expected total number of iterations is at least $\frac{qn}{256\beta} = \frac{q}{256\beta} \cdot \left\lfloor \frac{m}{\log(2q)} \right\rfloor$. This result holds also for a precognitive stream algorithm, since it also must select admissible vectors due to the same analysis. ■

6. Active Learning for Binary Classification

Lastly, we consider a special case of interactive algorithms, active learning for binary classification. Recent works provide for this setting relatively tight label complexity bounds, that hold for both the stream-based and the pool-based settings. In Balcan and Long (2013), tight upper and lower bounds for active learning of homogeneous linear separators under isotropic log-concave distributions are provided. The bounds hold for both the stream-based and the pool-based setting, and assume the same budget of unlabeled examples. In Hanneke and Yang (2015), tight minimax label complexity bounds for active learning are provided for several classes of distributions. These bounds also hold for both the stream-based and the pool-based setting. No explicit restriction is placed on the number of unlabeled examples.

The upper bound in Theorem 11 for utility-based pool algorithms can be applied to several pool-based active-learning algorithms which use a utility function (e.g., Golovin and Krause, 2011; Gonen et al., 2013; Cuong et al., 2014). The upper bound shows that when the label budget q is relatively small, the gap between the stream and the pool settings is not significant. For instance, consider an active learning problem in which a utility-based pool active learner achieves a label complexity close to the information-theoretic lower bound for the realizable setting (Kulkarni et al., 1993), so that $q \in \Theta(\log(1/\epsilon))$. The passive learning sample complexity in the realizable setting is at most $m \in \Theta(1/\epsilon)$. Therefore, a stream-based active learner with the same properties needs at most $O(\log(1/\epsilon)/\epsilon)$ unlabeled examples. Therefore, in this case the difference between the pool-based setting and the stream-based setting can be seen as negligible.

Here, we study the general question: what is the possible gap between pool-based and stream based active learning? We study this question in the realizable setting, where the distribution is consistent with some hypothesis in a given finite hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. We give an upper bound and a lower bound on this gap, which hold for a large class of pool-based algorithms.

First, we show the upper bound. This upper bound is derived by showing that any pool-based active learner for the realizable setting with a finite hypothesis class can be approximated by a utility-based pool algorithm. This in turn implies, using Theorem 11, that a stream algorithm with

similar guarantees can emulate the pool-based algorithm with a bounded approximation factor on the number of iterations and queries.

To show that any pool-based active learner can be approximated by a utility-based pool algorithm, we use the machinery of submodular function maximization for active learning. The challenge is that a successful pool-based active learner might not fully identify the true hypothesis consistent with the distribution, since it suffices that it identifies a hypothesis which is ϵ-close to the true one. Thus, we define a submodular function that allows taking this into account. Our construction combines ideas from Guillory and Bilmes (2010); Golovin et al. (2010b); Dasgupta (2006).

Let $f : \cup_{i=0}^{\infty}(\mathcal{X} \times \mathcal{Y})^n \to \mathbb{N}$ be an objective function which maps any set of labeled examples to a non-negative value. For a given pool of examples, let OPT be the smallest number of queries that are required (in the worst-case over all possible labelings of the pool that are consistent with $\mathcal{H}$) to obtain an S such that $f(S) = Q$ for some fixed integer Q. Guillory and Bilmes (2010) show that if f is monotone non-decreasing and submodular, then there exists a utility-based active learning algorithm which requires at most $O(\log(Q|\mathcal{H}|)) \cdot \mathrm{OPT}$ queries in the worst case to obtain an S such that $f(S) = Q$.

We define a function f which will allow proving the existence of a utility-based pool active learner in our setting. For a given set of labeled examples $S \subseteq \mathcal{X} \times \mathcal{Y}$, the version space induced by S on $\mathcal{H}$ is defined by $\mathrm{VS}(S) := \{h \in \mathcal{H} \mid \forall (x, y) \in S, h(x) = y\}$. Define a fixed set of pairs of hypotheses $E \subseteq \mathcal{H} \times \mathcal{H}$. Let $E(S) := (\mathrm{VS}(S) \times \mathrm{VS}(S)) \cap E$. This is the set of pairs from E such that none of the hypotheses in the pair are disqualified by the labels in S.

Given E, define the objective function f by

$$f(S) := |E| - |E(S)|.$$

Let $Q = |E|$. The following lemma shows that the objective-function requirements above hold for this definition of f.

Lemma 14 *For any $E \subseteq \mathcal{H} \times \mathcal{H}$, f as defined above is monotone and submodular.*

Proof Monotonicity is trivial here since if $S \subseteq S'$ then $|E(S)| \leq |E(S')|$. f is submodular (see e.g., Fujishige 2005) if and only if for any S, S' such that $S \subseteq S'$, and any $(x, y) \notin S'$,

$$f(S \cup \{(x, y)\}) - f(S) \geq f(S' \cup \{(x, y)\}) - f(S'). \tag{8}$$

Denote $S_+ := S \cup \{(x, y)\}$ for brevity, and similarly for S'_+. It is easy to see that $\mathrm{VS}(S) \supseteq \mathrm{VS}(S')$. In addition, $\mathrm{VS}(S) \setminus \mathrm{VS}(S_+) \supseteq \mathrm{VS}(S') \setminus \mathrm{VS}(S'_+)$, since any $h \in \mathrm{VS}(S') \setminus \mathrm{VS}(S'_+)$ is consistent with all the pairs in S' but not with (x, y), implying that $h \in \mathrm{VS}(S) \setminus \mathrm{VS}(S_+)$.

It follows that

$$\mathrm{VS}(S) \times \mathrm{VS}(S) \setminus (\mathrm{VS}(S_+) \times \mathrm{VS}(S_+)) \supseteq \mathrm{VS}(S') \times \mathrm{VS}(S') \setminus (\mathrm{VS}(S'_+) \times \mathrm{VS}(S'_+)).$$

Therefore,

$$(\mathrm{VS}(S) \times \mathrm{VS}(S) \cap E) \setminus ((\mathrm{VS}(S_+) \times \mathrm{VS}(S_+)) \cap E) \supseteq (\mathrm{VS}(S') \times \mathrm{VS}(S') \cap E) \setminus ((\mathrm{VS}(S'_+) \times \mathrm{VS}(S'_+)) \cap E).$$

It follows that $|E(S)| - |E(S_+)| \geq |E(S')| - |E(S'_+)|$, which implies Eq. (8). ■

We use this construction in the proof of the upper bound on the gap between pool-based and stream-based active learning, which we presently state and prove.

A somewhat technical issue is that the results of Guillory and Bilmes (2010) hold only under the assumption that whenever the optimal pool-based algorithm succeeds, it knows that it has succeeded. In other words, it has "proof" that its answer is ϵ-good. This is known as a *self-certifying* algorithm Golovin et al. (2010b). Our results thus apply to pool-based active learning algorithm that have this property as well. This property is related, though not completely equivalent, to the concept of "verifiable active learning algorithms" studied in Balcan et al. (2010). In that work it is shown that non-verifiable active learning algorithms sometime require fewer labels than verifiable active learning algorithms. Characterizing the gap between pool-based and stream-based algorithms for non-verifiable active learning algorithms is an open question which we leave for future work.

For a fixed distribution $\mathcal{D}_X$ over $\mathcal{X}$, denote $\Delta(h, h') := \mathbb{P}[h(X) \neq h'(X)]$.

Theorem 15 *Let $\mathcal{X}$ be a finite instance domain and let $\mathcal{Y}$ be a finite label domain. Let $\mathcal{D}_X$ be a marginal distribution over $\mathcal{X}$. Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class. Suppose that there is a pool-based active learning algorithm such that for any $\mathcal{D} \in \text{DS}(\mathcal{D}_X)$ which is consistent with some $h^* \in \mathcal{H}$, with a probability at least $1 - \delta$ over i.i.d. pools of size m, it outputs $\hat{h} \in \mathcal{H}$ such that $\Delta(\hat{h}, h^*) < \epsilon$ for $(X, Y) \sim \mathcal{D}$, using q label queries. We also assume that the pool algorithm is self-certifying: the algorithm also outputs an indicator $I \in \{0, 1\}$ where $\mathbb{P}[\Delta(\hat{h}, h^*) < \epsilon \mid I = 1] = 1$, and $\mathbb{P}[I = 1] \geq 1 - \delta$.*

Under these assumptions, there exists a stream-based active learner that with a probability of at least $1 - |\mathcal{H}|\delta$ outputs a hypothesis $\hat{h}$ such that $\Delta(\hat{h}, h^*) \leq 2\epsilon$. The expected number of queries requested by the stream-based active learner is at most $O(\log(|\mathcal{H}|)) \cdot q$, and its expected number of iterations is at most $O(\log(|\mathcal{H}|)) \cdot mq$.

Proof Define $E := \{(h, h') \mid \Delta(h, h') \geq 2\epsilon\}$. Denote an ϵ-ball around a hypothesis $h \in \mathcal{H}$ by

$$B(h, \epsilon) := \{h' \in \mathcal{H} \mid \Delta(h, h') < \epsilon].$$

Fix a $\mathcal{D} \in \text{DS}(\mathcal{D}_X)$ which is consistent with some $h^* \in \mathcal{H}$. Any pool algorithm with the guarantees assumed by the theorem statement, outputs an $I = 1$ with a probability at least $1 - \delta$, and has $\mathbb{P}[\hat{h} \in B(h^*, \epsilon) \mid I = 1] = 1$. Thus, for any pool with $I = 1$, $h^* \in B(\hat{h}, \epsilon)$. Let S be the set of labeled examples that the pool algorithm obtained during its entire run. Since any of the hypotheses in $\text{VS}(S)$ could be the true h^*, it follows that for any $h \in \text{VS}(S)$, $\Delta(\hat{h}, h) < \epsilon$, hence for any $h, h' \in \text{VS}(S)$, $\Delta(h, h') \leq 2\epsilon$. Therefore, $(h, h') \notin E$. It follows that $E(S) = \emptyset$.

There are $|\mathcal{H}|$ possible distributions $\mathcal{D} \in \text{DS}(\mathcal{D}_X)$ which are consistent with some $h^* \in \mathcal{H}$. It follows that with a probability of $1 - |\mathcal{H}|\delta$ over pools, at the end of the run $E(S) = \emptyset$. For any such pool, we have that q queries suffice to certifiably obtain $f(S) = Q$, thus $\text{OPT} \leq q$.

We now apply the results cited above from Guillory and Bilmes (2010), which imply that there exists a utility-based pool algorithm that obtains $f(S) = Q$ with at most $O(\log(|\mathcal{H}|)q)$ queries. By Theorem 11, we conclude that there exists a stream algorithm which requires at most $O(\log(|\mathcal{H}|)) \cdot q$ queries and $O(\log(|\mathcal{H}|)) \cdot mq$ iterations to obtain $f(S) = Q$.

Since $f(S) = Q$ at the end of the run of the stream algorithm, then $E(S) = \emptyset$, which implies that for any (h, h') such that $\Delta(h, h') \geq 2\epsilon$, at least one of h, h' is not in $\text{VS}(S)$. Therefore, the diameter of $\text{VS}(S)$ is at most 2ϵ. The stream algorithm may therefore return any $\hat{h} \in \text{VS}(S)$, and it holds that $\Delta(\hat{h}, h^*) \leq 2\epsilon$. ■

Next, we provide a lower bound, showing that in the general case $\tilde{\Omega}(mq)$ iterations are required for emulating pool-based active learning in a stream setting. The lower bound, which holds also for the precognitive setting, employs the following example and is presented in Theorem 16.

Example 1 *For given integers m and $q \leq m$, and $T \leq q$, define $\mathcal{X} = \{a_{k,j} \mid k \in [q], j \in \{0, \ldots, 2^{\min(k,T)-1} - 1\}\} \cup \mathcal{X}'$, where $\mathcal{X}'$ includes arbitrary elements so that $|\mathcal{X}| = n$, for some $n \geq q2^T/2$. Define the following hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$.*

$$\mathcal{H} := \{h_i \mid i \in \{0, \ldots, 2^q - 1\}\}, \text{ where } h_i(a_{k,j}) = \begin{cases} \mathbb{I}[i \bmod 2^k = j] & k \leq T, \\ \mathbb{I}[\lfloor i/2^{T-k} \rfloor \bmod 2^T = j]. & k > T. \end{cases} \tag{9}$$

Essentially, for $k \leq T$, $h_i(a_{k,j}) = 1$ if the k least significant bits in the binary expansion of i are equal to the binary expansion of j to T bits. For $k \geq T$, $h_i(a_{k,j}) = 1$ if T consecutive bits in i, starting from bit $T - k$, are equal to the binary expansion of j.

Theorem 16 *Let $q \geq 22$ and $m \geq 8 \log(2q) q^2$ be integers. Consider Example 1 with m, q, setting $T = \lceil \log_2(q) \rceil$ and $n = \lfloor m/7 \log(2q) \rfloor$. Consider $\mathcal{H}$ as defined in Eq. (9). There exist $\delta, \epsilon \in (0, 1)$ such that there is a pool-based active learning algorithm that uses a pool of m unlabeled examples and q labels, such that for any distribution $\mathcal{D}$ which is consistent with some $h^* \in \mathcal{H}$ and has a uniform marginal over $\mathcal{X}$, with a probability of at least $1 - \delta$, $\mathbb{P}[\hat{h}(X) \neq h^*(X)] \leq \epsilon$. On the other hand, for $q > 22$, any stream-based active learning algorithm with the same guarantee, including a precognitive stream algorithm, requires at least $\frac{q}{512} \left\lfloor \frac{m}{7 \log(2q)} \right\rfloor$ unlabeled examples in expectation.*

Proof Let $\mathcal{D}_X$ be uniform over $\mathcal{X}$. Let E be the event that $\mathcal{X} \not\subseteq_\pi S_X$, and define $\delta := \mathbb{P}_{S \sim \mathcal{D}_X^m}[E]$. Define $\epsilon = 1/n$, so that $\mathbb{P}[\hat{h}(X) \neq h^*(X)] < \epsilon$ if and only if $\hat{h} = h^*$. Let i^* such that $h^* = h_{i^*}$.

First, a pool-based algorithm can achieve the required accuracy as follows: Let $j_t := i^* \bmod 2^t$ for $t \leq T$, and $j_t := \lfloor i^*/2^{T-t} \rfloor \bmod 2^T$ for $t \geq T$. If E holds, then t'th element selected by the pool algorithm is $a_{t,j}$, where j is obtained as follows: If $t \leq T$, $j = j_{t-1}$. If $t > T$, $j = \lfloor j_{t-1}/2 \rfloor$. In round 1, $j = 0$ and the selected element is $a_{1,0}$. Inductively, in this strategy the algorithm finds the t'th least significant bit in the binary expansion of i^* in round t, thus it can use j_{t-1} to set j for round t. Under E, after q labels i^* is identified exactly. This happens with a probability of $1 - \delta$ for any $\mathcal{D}$ with the uniform marginal $\mathcal{D}_X$.

Now, let $\mathcal{D}_h$ be a distribution with a uniform marginal over $\mathcal{X}$ with labels consistent with $h \in \mathcal{H}$. Consider a stream-based algorithm $\mathcal{A}_s$ which is equivalent to the pool algorithm. Denote its output by $\bar{h}$ and its input by $S \sim \mathcal{D}_{h^*}^\infty$. Let I be a random variable drawn uniformly at random from $\{0, \ldots, 2^q - 1\}$. Let $H = h_I$ be a hypothesis chosen uniformly at random from $\mathcal{H}$. Consider the probability space defined by $I, S \sim \mathcal{D}_H^\infty$, and the run of $\mathcal{A}_s$ on S. Let $(Z_1, Y_1), \ldots, (Z_q, Y_q)$ be the examples that $\mathcal{A}_s$ receives and the labels it gets, in order. Let $Y = (Y_1, \ldots, Y_q)$. Let $\alpha = \mathbb{P}[Z_1 = a_{1,0} \mid S_X]$. If $Z_1 = a_{1,0}$, then $\mathbb{P}[Y_1 = 0 \mid S_X] = \frac{1}{2}$. If $Z_1 \neq a_{1,0}$, then $\mathbb{P}[Y_1 = 0 \mid S_X] \geq 3/4$. Let $\mathbb{H}$ be the base-2 entropy, and $\mathbb{H}_b$ be the binary entropy. Then $\mathbb{H}_b(Y_1 \mid S_X) = \mathbb{H}_b((\alpha + 1)/4)$, and so

$$\begin{aligned} \mathbb{H}(H \mid Y, S_X) &= \mathbb{H}(H, Y \mid S_X) - \mathbb{H}(Y_1 \mid S_X) - \mathbb{H}(Y_1, \ldots, Y_q \mid Y_1, S_X) \\ &\geq q - \mathbb{H}_b((\alpha + 1)/4) - (q - 1) \\ &= 1 - \mathbb{H}_b((\alpha + 1)/4). \end{aligned}$$

From the Taylor expansion of the binary entropy around $1/2$, $\mathbb{H}_b(p) \leq 1 - (1-2p)^2/2$, therefore $\mathbb{H}(H \mid Y, S_X) \geq (1-\alpha)^2/8$. Since $\mathcal{A}_s$ is equivalent to $\mathcal{A}_p^{\mathcal{U}}$, we have $\mathbb{P}[\bar{h} \neq H] \leq \delta$, hence $\mathbb{P}_{S_X}[\mathbb{P}[\bar{h} \neq H \mid S_X] \leq 2\delta] \geq \frac{1}{2}$. By Fano's inequality, for any S_X such that $\mathbb{P}[\bar{h} \neq H \mid S_X] \leq 2\delta$,

$$(1-\alpha)^2/8 \leq \mathbb{H}(H \mid Y, S_X) \leq \mathbb{H}_b(2\delta) + 2\delta q \leq 2\delta(\log_2(\frac{1}{2\delta}) + 2 + q).$$

Where the last inequality follows from $\mathbb{H}_b(p) \leq p\log_2(1/p) + 2p$. From the definition of δ, we have $\delta \leq |\mathcal{X}| \exp(-m/n)$. Setting $T = \lceil \log_2(q) \rceil$, and noting that $|\mathcal{X}| \leq q2^T/2 \leq q^2$ and so $m \geq n\log(128q^3|\mathcal{X}|)$, we have $\delta \leq \frac{1}{128q^3}$. Therefore, for $q \geq 22$, $1-\alpha \leq \frac{1}{2q}$. It follows that $\mathbb{P}_{S_X}[\mathbb{P}[Z_1 \neq a_{1,0} \mid S_X] \leq \frac{1}{2q}] \geq 1/2$.

Now, the same argument holds for any round t conditioned on $I \bmod 2^t = 0$ and $Z_1 = a_{1,0}, \ldots, Z_t = a_{t,0}$, since in this case after t labels, the algorithm has $q-t$ queries left, and needs to select from $\mathcal{H}'$, which is equivalent to $\mathcal{H}$, with $q-t$ instead of q. Moreover, $\mathbb{P}[\bar{h} = H \mid I \bmod 2^t = 0] \leq 1-\delta$ as well, since this holds for every H individually. We conclude that for every $t \leq q$, with a probability at least $\frac{1}{2}$ over S_X,

$$\mathbb{P}[Z_t \neq a_{t,0} \mid S_X, H = h_0] \leq \frac{1}{2q}.$$

It follows that with a probability at least $\frac{1}{2}$ over S_X, $\mathbb{P}[Z_1 = a_{1,0}, \ldots, Z_q = a_{q,0} \mid S_X, H = h_0] \geq 1/2$. Hence $\mathbb{P}[Z_1 = a_{1,0}, \ldots, Z_q = a_{q,0} \mid H = h_0] \geq 1/4$.

Finally, suppose $H = h_0$. Let W_t be the number of elements that $\mathcal{A}_s$ observes after selecting element $t-1$, until observing the next element. Let $X_j \sim \mathcal{D}_X$ be the j'th element observed after selecting the first $t-1$ elements, and let $B_j = \mathbb{I}[X_j = a_{t,0}]$. B_j are independent Bernoulli random variables with $\mathbb{P}[B_j = 1] = 1/n$, and $\mathbb{P}[B_{W_t} = 1] \geq \mathbb{P}[E] = \beta \geq \frac{1}{4}$. By Lemma 2, if $\frac{1}{n} \leq \frac{1}{512}$, then $\mathbb{E}[W_t] \geq \frac{n}{64}$. It follows that the expected number of iterations over q selections is at least $\frac{qn}{64}$ if $n \geq 512$. Since it is also at least q, a lower bound of $\frac{qn}{512}$ holds for all n. ■

This lower bound shows that a gap between the stream-based and the pool-based settings exists not only for general interactive algorithms, but also specifically for active learning for binary classification. The gap is the most significant when $q = \tilde{\Theta}(\sqrt{m})$, in which case the stream algorithm requires $\tilde{\Omega}(m^{3/2})$ unlabeled examples, compared to only m for the pool algorithm. It has been previously observed (Gonen et al., 2013) that in some cases, the ALuMA algorithm, which is a pool-based active learning algorithm for halfspaces, is superior to the classical stream-based algorithm CAL (Cohn et al., 1994). Theorem 16 shows that this is not a limitation specifically of CAL, but of any stream-based active learning algorithm. On the other hand, Theorem 15 shows that this gap cannot be very large, at least for self-certifying active learning algorithms.

7. Conclusions

In this work we studied the relationship between the stream-based and the pool-based interactive settings, by designing algorithms that emulate pool-based behavior in a stream-based setting, and proving upper and lower bounds on the stream sizes required for such emulation. Our results concern mostly the case where the budget of the stream algorithm is similar or identical to that of the pool algorithm. We expect that as the budget grows, there should be a smooth improvement in

the expected stream length, which should approach m as the budget approaches m. It is an open problem to quantify this tradeoff in the various settings we considered.

Acknowledgments

This work was supported by the Israel Science Foundation (grant No. 555/15).

Appendix A. Proof of Lemma 2

The proof of Lemma 2, defined in Section 2, is provided below.

Proof [of Lemma 2] $\mathbb{E}[I]$ is minimized under the constraint when $\mathbb{P}[X_i = 1] = p$. Therefore assume this equality holds. Let W be the random variable whose value is the smallest integer such that $X_W = 1$. Let T be the largest integer such that $\mathbb{P}[W \leq T] \leq \alpha$.

The expectation of I is subject to $\mathbb{P}[X_I = 1] \geq \alpha$ is smallest if I is defined as follows: $\mathbb{P}[I = W \mid W \leq T] = 1$, $\mathbb{P}[I = W \mid W = T + 1] = \alpha - \mathbb{P}[W \leq T]$, and in all other cases, $I = 0$. Therefore,

$$\mathbb{E}[I] \geq \mathbb{E}[W \cdot \mathbb{I}[W \leq T]].$$

We have

$$\begin{aligned}\frac{1}{p} = \mathbb{E}[W] &= \mathbb{E}[W \cdot \mathbb{I}[W \leq T]] + \mathbb{E}[W \cdot \mathbb{I}[W > T]] \\ &= \mathbb{E}[W \cdot \mathbb{I}[W \leq T]] + (\frac{1}{p} + T)(1-p)^T.\end{aligned}$$

Therefore

$$\mathbb{E}[I] \geq \mathbb{E}[W \cdot \mathbb{I}[W \leq T]] = \frac{1}{p} - (\frac{1}{p} + T)(1-p)^T.$$

From the definition of T, T is the largest integer such that $1 - (1-p)^T \leq \alpha$. Hence $1 - \alpha \leq \exp(-pT)$, so $T \leq \frac{\log(1/(1-\alpha))}{p}$. In addition, since $1 - (1-p)^{T+1} \geq \alpha$, we have $(1-p)^T \leq (1-\alpha)/(1-p)$. Therefore

$$\mathbb{E}[I] \geq \frac{1}{p} - \left(\frac{1}{p} + \frac{\log(1/(1-\alpha))}{p}\right)\frac{1-\alpha}{1-p} = \frac{1}{p}\left(1 - \frac{(1-\alpha)(1+\log(1/(1-\alpha)))}{1-p}\right).$$

To show that $\mathbb{E}[I] \geq \alpha^2/(4p)$, for $p \leq \alpha^4/2$ and $\alpha \in (0,1)$, we show that

$$1 - \frac{(1-\alpha)(1+\log(1/(1-\alpha)))}{1-p} \geq \alpha^2/4. \tag{10}$$

It suffices to show that

$$1 - \alpha^2/4 - \frac{(1-\alpha)(1+\log(1/(1-\alpha)))}{1-\alpha^4/2} \geq 0.$$

Multiplying the LHS by $1 - \alpha^4/2$, we have

$$\begin{aligned}A(\alpha) &:= (1-\alpha^2/4)(1-\alpha^4/2) - (1-\alpha)(1+\log(1/(1-\alpha))) \\ &= \alpha - \alpha^2/4 - \alpha^4/2 + \alpha^6/8 - (1-\alpha)\log(1/(1-\alpha)).\end{aligned}$$

By the Taylor expansion of the natural logarithm around $1-\alpha$,

$$(1-\alpha)\log(1/(1-\alpha)) = \sum_{n=1}^{\infty}\frac{\alpha^n}{n} - \sum_{n=1}^{\infty}\frac{\alpha^{n+1}}{n} = \alpha + \sum_{n=2}^{\infty}\alpha^n(\frac{1}{n} - \frac{1}{n-1}).$$

Therefore

$$\begin{aligned} A(\alpha) &= \alpha - \alpha^2/4 - \alpha^4/2 + \alpha^6/8 - \alpha + \sum_{n=2}^{\infty}\alpha^n(\frac{1}{n-1} - \frac{1}{n}) \\ &= \sum_{n=2}^{\infty}\alpha^n(\frac{1}{n-1} - \frac{1}{n}) - \alpha^2/4 - \alpha^4/2 + \alpha^6/8 \\ &= \sum_{n=3}^{\infty}\alpha^n(\frac{1}{n-1} - \frac{1}{n}) + (\alpha^2/4 - \alpha^4/2) + \alpha^6/4 \geq 0. \end{aligned}$$

It follows that $A(\alpha)/(1-\alpha^4/2) \geq 0$, implying Eq. (10), and concluding that $\mathbb{E}[I] \geq \alpha^2/(4p)$. ■

References

Alessandro Arlotto, Elchanan Mossel, and J Michael Steele. Quickest online selection of an increasing subsequence of specified size. *Random Structures & Algorithms*, 49(2):235–252, 2016.

M.-F. Balcan and Phil Long. Active and passive learning of linear separators under log-concave distributions. In *Proceedings of the Twenty-Sixth Annual Conference on Computational Learning Theory (COLT)*, pages 288–316, 2013.

M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1):78–89, 2009.

Maria-Florina Balcan, Steve Hanneke, and Jennifer Wortman Vaughan. The true sample complexity of active learning. *Machine Learning*, 80(2-3):111–139, 2010.

M. Bouguelia, Y. Belaid, and A. Belaïd. A stream-based semi-supervised active learning approach for document classification. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 611–615. IEEE, 2013.

N. Cebron and M. R. Berthold. Active learning for object classification: from exploration to exploitation. *Data Mining and Knowledge Discovery*, 18(2):283–299, 2009.

W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng. Unbiased online active learning in data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 195–203. ACM, 2011.

D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994.

N.V. Cuong, W.S. Lee, and N. Ye. Near-optimal adaptive pool-based active learning with general loss. In *30th conference on Uncertainty in Artificial Intelligence (UAI)*, pages 122–131, 2014.

S. Dasgupta. Analysis of a greedy active learning strategy. *Advances in Neural Information Processing Systems 18 (NIPS)*, 17:337–344, 2005.

S. Dasgupta. Coarse sample complexity bounds for active learning. *Advances in Neural Information Processing Systems 19 (NIPS)*, 18:235, 2006.

Sanjoy Dasgupta. Consistency of nearest neighbor classification under selective sampling. In *Proceedings of the 25th Annual Conference on Learning Theory (COLT)*, pages 18.1–18.15, 2012.

E. D. Demaine, P. Indyk, S. Mahabadi, and A. Vakilian. On streaming and communication complexity of the set cover problem. In *Distributed Computing*, pages 484–498. Springer, 2014.

A. Deshpande, L. Hellerstein, and D. Kletenik. Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1453–1467. SIAM, 2014.

E. B. Dynkin. The optimum choice of the instant for stopping a markov process. In *Soviet Math. Dokl*, volume 4, pages 627–629, 1963.

T. S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4(3):282–289, 1989.

Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

J. P. Gilbert and F. Mosteller. Recognizing the maximum of a sequence. *Journal of the American Statistical Association*, 61(313):35–73, 1966.

D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

D. Golovin, M. Faulkner, and A. Krause. Online distributed sensor selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 220–231. ACM, 2010a.

D. Golovin, A. Krause, and D. Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems 23 (NIPS)*, pages 766–774, 2010b.

A. Gonen, S. Sabato, and S. Shalev-Shwartz. Efficient active learning of halfspaces: an aggressive approach. *Journal of Machine Learning Research*, 14:2487–2519, 2013.

P. H. Gosselin and M. Cord. Active learning methods for interactive image retrieval. *IEEE Transactions on Image Processing*, 17(7):1200–1211, 2008.

A. Guillory and J. A. Bilmes. Interactive submodular set cover. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 415–422, 2010.

Y. Guo and R. Greiner. Optimistic active-learning using mutual information. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 823–829, 2007.

Y. Guo, I. Silins, U. Stenius, and A. Korhonen. Active learning-based information structure analysis of full scientific articles and two applications for biomedical literature review. *Bioinformatics*, 29 (11):1440–1447, 2013.

S. Hanneke. Teaching dimension and the complexity of active learning. In *Proceedings of the Twentieth Annual Conference on Computational Learning Theory (COLT)*, 2007.

S. Hanneke. Rates of convergence in active learning. *The Annals of Statistics*, 39(1):333–361, 2011.

S. Hanneke and L. Yang. Minimax analysis of active learning. *Journal of Machine Learning Research*, 16:3487–3602, 2015.

V. Krishnamurthy. Algorithms for optimal scheduling and management of hidden markov model sensors. *IEEE Transactions on Signal Processing*, 50(6):1382–1397, 2002.

S. R. Kulkarni, S. K. Mitter, and J. N. Tsitsiklis. Active learning using arbitrary binary valued queries. *Machine Learning*, 11(1):23–35, 1993.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.

C. C. Loy, T. M. Hospedales, T. Xiang, and S. Gong. Stream-based joint exploration-exploitation active learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1560–1567, June 2012.

A. K. McCallum and K. Nigam. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 350–358, 1998.

P. Mitra, C.A. Murthy, and S. K. Pal. A probabilistic active support vector learning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):413–418, 2004.

S. Sabato and R. Munos. Active regression by stratification. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 469–477, 2014.

S. Sabato, A D Sarwate, and N Srebro. Auditing: Active learning with outcome-dependent query costs. In *Advances in Neural Information Processing Systems 26*, pages 512–520, 2013.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on computational learning theory*, pages 287–294. ACM, 1992.

A. Singla, S. Tschiatschek, and A. Krause. Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization. In *Conference on Artificial Intelligence (AAAI)*, 2016.

J. Smailović, M. Grčar, N. Lavraŏ, and M. Žnidaržič. Stream-based active learning for sentiment analysis in the financial domain. *Information Sciences*, 285(0):181 – 203, 2014.

M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 1577–1584, 2009.

S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the Ninth ACM International Conference on Multimedia*, MULTIMEDIA '01, pages 107–118. ACM, 2001.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research (JMLR)*, 2:45–66, 2002.