Online Submodular Maximization via Adaptive Thresholds

Zhengchen Yang[1] and **Jiping Zheng**[1,2]
[1]College of Computer Science & Technology, Nanjing University of Aeronautics & Astronautics, Nanjing, China
[2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{yangzc, jzh}@nuaa.edu.cn

Abstract

Submodular function maximization has been studied extensively in recent years due to its numerous applications in machine learning and artificial intelligence. We study a natural online variant of this problem on massive streaming data in which elements arrive one-by-one and the algorithm has to maintain a solution under cardinality constraint, i.e., k. Upon arrival of an element, the algorithm to maximize a monotone submodular function has to decide whether to accept the element and may replace a previously chosen element. Existing algorithms cannot simultaneously achieve optimal performance in terms of competitive ratio, memory complexity and running time. Also, the algorithm with best competitive ratio performs poorly in practice. In this paper, we propose a new algorithm ONLINEADAPTIVE with optimal performance by exploiting adaptive thresholds to decide the acceptance of arriving elements by replacement. We prove that the competitive ratio of ONLINEADAPTIVE is at least $1/4$, and the ratio is about 0.2959 when $k \geq 4$ and approaches 0.3178 when k tends to infinity. In addition, ONLINEADAPTIVE only needs $O(k)$ memory and just performs one oracle per element. Experiments on diverse datasets confirm that ONLINEADAPTIVE outperforms existing algorithms in both quality and efficiency.

1 Introduction

Submodularity exhibits the diminishing return property of set functions arising in numerous machine learning and artificial intelligence applications, such as data summarization [Mirzasoleiman *et al.*, 2016; Kumari and Bilmes, 2021], active learning [Wei *et al.*, 2015], feature selection [Schlegel *et al.*, 2017], influence maximization [Becker *et al.*, 2022], and user recommendation [Ashkan *et al.*, 2015]. A set function $f : 2^V \to \mathbb{R}_+$ with a ground set V is submodular if $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$ for all sets $A \subseteq B \subseteq V$ and every element $e \in V \backslash B$. The submodular function f is monotone if for all $A \subseteq B$ we have $f(A) \leq f(B)$. As a natural extension, online submodular maximization on monotone functions plays an important role on ubiquitous streaming data in real-world applications. Consider an example of a soccer team manager recruiting k players for the 2026 World Cup. When getting an additional application from a new potential player, the manager needs to make decision whether to recruit the player. When the number of the players in his team exceeds k, e.g., $k = 11$, a replacement occurs for satisfying the cardinality constraint k. The goal of the manager is to maximize the quality of the team which is indeed a submodular function of the chosen players [Buchbinder *et al.*, 2015]. As another example, in online or reinforcement learning, to avoid the inefficiency of kernel prototype selection for multiple passes over the dataset and maintaining a large number of parallel solutions, the prototypes are selected in the online manner [Schlegel *et al.*, 2017]. Specifically, when a prototype from an infinite or even uncountable space of observations is given, the algorithm needs to immediately decide whether to include the prototype. The objective is to maximize the coverage time to achieve the optimal solution of the chosen kernel prototypes. Also, similar scenarios can be found in many applications. In this paper, we consider the online monotone submodular maximization problem with replacement subject to cardinality constraint k. An immediate decision is made for an arriving element and the discarded elements will never be considered, and the accepted elements may be replaced by oncoming elements when better results can be achieved.

Currently, the representative streaming submodular maximization algorithms, such as STREAMGREEDY [Gomes and Krause, 2010], SIEVE-STREAMING [Badanidiyuru *et al.*, 2014] and SIEVE-STREAMING++ [Kazemi *et al.*, 2019], SALSA [Norouzi-Fard *et al.*, 2018], QUICKSTREAM [Kuhnle, 2021], and THREESIEVES [Buschjäger *et al.*, 2021] *cannot* meet the requirements of the above applications since these algorithms need to cache some elements for later evaluation. Our considered problem is also different from the classical submodular secretary problem [Hajiaghayi *et al.*, 2004; Bateni *et al.*, 2013] for the accepted elements may be replaced by oncoming elements when the algorithm achieves better solutions. Existing online submodular maximization algorithms in the online setting with replacement cannot simultaneously perform well in the three aspects, i.e., competitive ratio, memory complexity and running time. INDEPENDENTSETIMPROVEMENT [Chakrabarti and Kale, 2015], STREAMINGGREEDY [Chekuri *et al.*, 2015] and PREEMP-

TION [Buchbinder *et al.*, 2015; Buchbinder *et al.*, 2019] do not achieve the best competitive ratio and they all have the competitive ratio $1/4$. Moreover, STREAMINGGREEDY and PREEMPTION need $O(k)$ oracles (function calls) for each element thus leading to more time consumption. The current state-of-the-art theoretical result for online submodular maximization with replacement is achieved by the FREEDISPOSAL algorithm [Chan *et al.*, 2017; Chan *et al.*, 2018]. However, the memory complexity of FREEDISPOSAL is $O(n)$ which is intolerable for large-scale datasets in practice.

Our contributions. We design a novel algorithm ONLINEADAPTIVE simultaneously providing the best competitive ratio, memory complexity and running time for the online monotone submodular maximization problem with replacement. More precisely, we consider the optimization problem

$$\max_{S \subseteq V, |S|=k} f(S),$$

where V is a possibly infinite ground set where the elements of $V = \{v_1, v_2, ..., v_n\}$ are revealed one-by-one and $S \subseteq V$ is a solution set which is initialized to an empty set and the revealed elements are added to S according to the elaborately designed thresholds. At any timestamp t $(1 \leq t \leq n)$, S_t is the feasible solution set after element v_t is revealed. When $|S| < k$, the elements are selected into S by the thresholds. For the revealed element v_t when $|S| = k$, a replacement may occur to get a better result set. We assume that f is given as an evaluation oracle: when we specify $S \subseteq V$, the oracle returns the value of $f(S)$.

Our ONLINEADAPTIVE algorithm achieves a competitive ratio about 0.2959 when $k \geq 4$ by using adaptive thresholds for replacement and the competitive ratio approaches 0.3178 when k tends to infinity which are the same to the current state-of-the-art results. Besides this, ONLINEADAPTIVE only needs $O(k)$ memory by building a relationship to the set of all previously accepted elements. Moreover, for each revealed element, ONLINEADAPTIVE only needs $O(1)$ oracles.

We also conduct experiments on diverse datasets and the experimental results show that the quality, i.e., submodular function value of our ONLINEADAPTIVE algorithm is comparable to other algorithms. Furthermore, ONLINEADAPTIVE runs faster than existing algorithms and the number of oracles is much less than those of the other algorithms.

2 Related Work

After the seminal work [Nemhauser *et al.*, 1978] exhibits nice properties of submodular functions which are apt to obtain near-optimal solutions by a simple greedy algorithm GREEDY with $1-1/e$ approximation ratio, these decades witness the flourish of the study on submodular function maximization [Krause and Golovin, 2014; Bilmes, 2023]. Following we only review the literature of the monotone submodular function maximization over streaming data. Note that the studied problem of [Streeter *et al.*, 2009; Si-Salem *et al.*, 2024] with the same name focuses to reveal the submodular functions and is with different process and objective. Thus, it is completely different from ours.

To deal with the massive streaming data, STREAMGREEDY [Gomes and Krause, 2010] extends the GREEDY algorithm by simply accepting the first k elements and continuing to accept the elements if the improvement of replacing an element in the current solution set surpasses a fixed threshold. STREAMGREEDY achieves a $1/2 - \epsilon$ approximation only if multiple passes over the streaming data are allowed, where ϵ is related to the number of passes and some specified parameters. The first proper streaming algorithm with the same theoretical guarantee is SIEVE-STREAMING [Badanidiyuru *et al.*, 2014] which maintains a number of parallel solutions using different sizes of "sieves" and at arbitrary time period the solution with the maximum function value is returned as the result set. To make SIEVE-STREAMING more efficient or practical, several enhancements are provided. The SIEVE-STREAMING++ algorithm [Kazemi *et al.*, 2019] reduces the number of sieves by offering a better lower bound for the function value. Moreover, SIEVE-STREAMING++ only requires $O(k/\epsilon)$ memory instead of $O(k \log k/\epsilon)$ of SIEVE-STREAMING. The SALSA algorithm [Norouzi-Fard *et al.*, 2018] also achieves $1/2 - \epsilon$ approximation but only needs $O(k \log k)$ memory by the assumption that the elements in data streams arrive in a random order. The QUICKSTREAM [Kuhnle, 2021] and THREESIEVES [Buschjäger *et al.*, 2021] algorithms aim at decreasing the number of oracles to improve efficiency. QUICKSTREAM evaluates the function f every c elements and achieves a $1/(4c) - \epsilon$ approximation while THREESIEVES only accepts informative elements in data streams leading to $O(1)$ oracles per element and $O(k)$ memory, and the approximation further improves to $(1-\epsilon)(1-1/e)$ in high probability. Since these streaming algorithms need to maintain parallel solutions, i.e., the elements are cached without immediate decision, they are *not* suitable for the online setting.

To meet the requirements of the online submodular maximization, some of the single-pass streaming algorithms are indeed applicable to the online scenarios, such as STREAMINGGREEDY [Chekuri *et al.*, 2015] and INDEPENDENTSETIMPROVEMENT [Chakrabarti and Kale, 2015]. They share the same structure that the first k elements are directly accepted and replacement occurs when the marginal gain of an arriving element doubles the minimum marginal gain among the elements in the solution set. They both achieve a competitive ratio of $1/4$ and $O(k)$ memory complexity by recording the marginal gains, INDEPENDENTSETIMPROVEMENT only conducts one oracle per element instead of $O(k)$ oracles for each element of the STREAMINGGREEDY algorithm. The PREEMPTION algorithm [Buchbinder *et al.*, 2015; Buchbinder *et al.*, 2019] is explicitly designed for the submodular function maximization in the online setting. PREEMPTION also accepts the first k elements and continues to accept the elements if the improvement of replacing an element in the current solution surpasses the average value of the elements in it. PREEMPTION achieves the same competitive ratio and memory complexity as STREAMINGGREEDY and INDEPENDENTSETIMPROVEMENT but its theoretical analysis is much simpler than those. The FREEDISPOSAL algorithm [Chan *et al.*, 2017; Chan *et al.*, 2018] achieves the current state-of-the-art competitive ratio by delicately designing the acceptance conditions with the help of an auxiliary set storing all ever accepted elements. FREEDISPOSAL has a competitive ratio at least 0.2959, and the ratio approaches 0.3178 when k tends to infinity at the cost of $O(n)$ memory to store the auxiliary set. For larger-scale streaming data, the memory consumption is intolerable which makes FREEDISPOSAL inapplicable in real-world applications. By the analysis of the existing algorithms for online monotone submodular function maximization, none of the algorithms can simultaneously achieve the optimal competitive ratio, acceptable memory complexity and the fewest number of oracles which usually results in the least running time.

3 Preliminaries

In this section, we present some definitions and observations that are helpful in our analysis.

Let $[e]$ denote the timestamp when element e arrives. We first provide the definitions of marginal gain and incremental value of e where $e \in S_{t-1}$, i.e., $[e] < t$.

Definition 1. *The marginal gain of* e *over its solution set* $S_{[e]-1}$ *is defined as* $\Delta_f(e|S_{[e]-1}) = f(S_{[e]-1} \cup \{e\}) - f(S_{[e]-1})$.

Definition 2. *The incremental value of* e *over its solution set* $S_{[e]-1}$ *and current solution set* S_t *is defined as* $\Delta_f(e|S_{[e]-1} \cap S_{t-1}) = f(S_{[e]-1} \cap S_{t-1} \cup \{e\}) - f(S_{[e]-1} \cap S_{t-1})$.

Let $m(e)$ and $c_t(e)$ denote the marginal gain and the incremental value of e respectively, and we have $c_t(e) \geq m(e)$ due to submodularity of f. To give a lower bound of $f(S_t)$, we use the trivial definition that $f(\varnothing) \geq 0$ and accumulate the marginal gains of elements in S_t, then we naturally obtain:

Lemma 1. *For a timestamp* $t \in \{1, 2, \ldots, n\}$*, the solution set is* S_t*, and its value is at least*

$$f(S_t) \geq \sum_{e \in S_t} m(e).$$

To achieve the optimal competitive ratio for the algorithm, we observe that it is related to the cardinality constraint, i.e., k. Moreover, we only consider $k \geq 4$ because the trivial algorithm that only selects the singleton with the largest value achieves a $1/k$ competitive ratio. For a fixed k value, we define three constants: $\zeta := \log_2 \log_{1.2} k$, η denoting the positive root of the equation $(1+x)^{(k+1)} = kx + x + 2$ and $\beta_0 := \frac{1+k\eta}{(1+\eta)^k - 1}$. Next, based on these we define three parameters α_t, β_t and τ_t varying along with t similar to [Chan *et al.*, 2017; Ene and Nguyen, 2022]: $\alpha_t := \exp((\frac{|S_t|}{k})^\zeta \cdot \log r) \cdot \eta$, $\beta_t := \frac{1+k\alpha_t}{(1+\alpha_t)^k - 1}$, and $\tau_t := \sum_{i=1}^{|S_t|}[(1+\alpha_t)^{i-1} \cdot m(e_i)]$, where r is an adjusting parameter and $m(e_i)$ is the i-th largest marginal gain in $\{m(e) : e \in S_t\}$. After these definitions, we observe that our defined threshold τ_t satisfies monotonicity property.

Lemma 2. τ_t *is monotone non-decreasing with timestamp* t*, i.e., for the timestamp* $t \in \{1, 2, \ldots, n\}$*, we have* $\tau_{t-1} \leq \tau_t$.

Lemma 2 apparently holds due to non-decreasing of α_t and $m(e_i)$.

Algorithm 1: The ONLINEADAPTIVE algorithm

Input: Cardinality constraint k, parameters β_0, ζ, η, r.
Output: The solution set S_t at timestamp t.

1 $S_0 \leftarrow \varnothing$, $\tau_0 \leftarrow 0$.
2 **foreach** *arriving element* v_t **do**
3     $m(v_t) = \Delta_f(v_t|S_{t-1})$.
4   **if** $m(v_t) \geq \frac{\beta_{t-1}}{k} \cdot \tau_{t-1}$ **then**
5   **if** $|S_{t-1}| < k$ **then**
6             $S_t \leftarrow S_{t-1} \cup \{v_t\}$.
7   **else**
8   Let $v'_t = \arg\min_{e \in S_{t-1}} m(e)$.
9             $S_t \leftarrow S_{t-1} \cup \{v_t\} \setminus \{v'_t\}$.
10         $\alpha_t \leftarrow \exp((\frac{|S_t|}{k})^\zeta \cdot \log r) \cdot \eta$.
11         $\beta_t \leftarrow \frac{1+k\alpha_t}{(1+\alpha_t)^k - 1}$.
12         $\tau_t \leftarrow \sum_{i=1}^{|S_t|}[(1+\alpha_t)^{i-1} \cdot m(e_i)]$, where $m(e_i)$ is the i-th largest marginal gain in $\{m(e) : e \in S_t\}$.
13   **return** S_t

---

4 The ONLINEADAPTIVE Algorithm

In this section, we present our ONLINEADAPTIVE algorithm for maximizing monotone submodular functions in the online setting. We first show the advantages of our adaptive thresholding strategy. Then ONLINEADAPTIVE is provided based on the constants and parameters defined before.

Replacement condition. For the arriving element v_t, our adaptive thresholding strategy can be simply expressed as

$$m(v_t) \geq \frac{\beta_{t-1}}{k} \cdot \tau_{t-1}.$$

By expanding the marginal gain $m(v_t)$ and the threshold τ_{t-1} as defined, we obtain the general form of the replacement condition:

$$\Delta_f(v_t|S_{t-1}) \geq \frac{\beta_{t-1}}{k} \sum_{i=1}^{|S_{t-1}|} [(1+\alpha_{t-1})^{i-1} \cdot \Delta_f(e_i|S_{t-1})],$$

where e_i is with the i-th largest marginal gain in $\{m(e) : e \in S_{t-1}\}$. The thresholds based on the delicate constants and parameters pave the way to achieving the state-of-the-art competitive ratio.

The algorithm. Algorithm 1 shows the details of our proposed ONLINEADAPTIVE algorithm. Different from previous online algorithms such as STREAMINGGREEDY, INDEPENDENTSETIMPROVEMENT and PREEMPTION, the first k elements are not directly accepted but need to satisfy the replacement condition (Lines 4-6) which guarantees the accepted elements have enough contributions when $|S_t| < k$. In our ONLINEADAPTIVE algorithm, the element in S_{t-1} with least contribution is replaced when $|S_t| = k$ (Lines 8-9). After that, the parameters α_t, β_t and τ_t are updated along with the solution set S_t (Lines 10-12) and α_t and β_t remain unchanged when $|S_t| = k$. Note that α_t is adjusted by an additional parameter r ($r > 1$) for $m(e_i)$ decreases with the increase of i and a larger weight denoted as $(1+\alpha_t)^{i-1}$ (Line 12) to a smaller $m(e_i)$ means we do not neglect the accepted elements with small marginal gains. With the help of this adjusting parameter r, the quality of the solution set will be improved which is also verified in the experiments.

Advantages. In contrast to existing algorithms for online submodular maximization, our adaptive thresholding strategy in Algorithm 1 simultaneously has three advantages: a) improving the competitive ratio. Compared to the algorithms STREAMINGGREEDY, INDEPENDENTSETIMPROVEMENT and PREEMPTION, the adaptivity of the thresholds along with t is apt to achieve a higher competitive ratio than $1/4$. b) memory- and computation-efficiency. Reexamining the constants and parameters used in our adaptive strategy, we find they are only related to the user-specified cardinality constraint k and the elements in the solution set, instead of the set of ever accepted elements as FREEDISPOSAL did. We know that for FREEDISPOSAL, which is with best competitive ratio, the size of the set of ever accepted elements is $O(n)$ in the worst case. Thus, compared to FREEDISPOSAL, our proposed thresholding strategy is much more efficient in memory and time consumption. c) less number of oracles. Since we record the marginal gains of e as to the solution set S_{t-1} for later reuse, many oracles are avoided which is more efficient especially when the function calls are time-consuming.

5 Theoretical Analysis

In this section, we provide the competitive ratio of our ONLINEADAPTIVE algorithm as well as the memory and computation complexities.

As provided in Lemma 2, τ_t is monotone non-decreasing. More precisely, we quantify the difference between the two adjacent thresholds. It is obvious that when $|S_{t-1}| < k$, $\tau_t \geq \tau_{t-1} + m(v_t)$. When $|S_{t-1}| = k$, we know that α_t remains unchanged and $\alpha_{t-1} = \alpha_t = r\eta$. The difference between τ_{t-1} and τ_t is quantified by Lemma 3.

Lemma 3. *When $|S_{t-1}| = k$, let τ_{t-1} and τ_t be two adjacent thresholds, we have*

$$(a)\ \ \tau_t \geq \tau_{t-1} + m(v_t) - (1+r\eta)^{k-1} m(v'_t)$$

and

$$(b)\ \ \tau_t \leq (1+r\eta)\tau_{t-1} + m(v_t) - (1+r\eta)^{k} m(v'_t)$$

where v_t is the arriving element replacing v'_t and $v'_t = \arg\min_{e\in S_{t-1}} m(e)$.

Proof. When the replacement occurs, i.e., v_t replaces v'_t when $m(v_t)$ exceeds the threshold, the set of the marginal gains changes. Let $m(v_t)$ be the p-th largest marginal gain in $\{m(e) : e \in S_t\}$. After the replacement, assume the elements in τ_{t-1} and τ_t are ordered by the marginal gains. We observe that the first p-1 elements in τ_t remain unchanged as to τ_{t-1}. The p-th element has changed to $(1+r\eta)^{p-1}m(v_t)$ and the i-th element in τ_t ($p+1 \leq i \leq k$) is the (i-1)-th element in τ_{t-1} multiplied by the coefficient $1+r\eta$. Thus, we have

$$\tau_t - \tau_{t-1} = (1+r\eta)^{p-1}m(v_t) - (1+r\eta)^{k-1}m(v'_t) + \sum_{i=p+1}^{k} [(1+r\eta)^{i-1} - (1+r\eta)^{i-2}] m(e_j). \quad (1)$$

Since $1 + r\eta > 1$ and $m(e_i) \geq 0$, Inequality (a) obviously holds. Further, we know that $m(v_t) \leq m(e_i)$ when $1 \leq i \leq p$, we have

$$\begin{aligned}
&\sum_{i=p+1}^{k} ((1+r\eta)^{i-2}) m(e_i) \\
&= \tau_{t-1} - (1+r\eta)^{k-1}m(v'_t) - \sum_{i=1}^{p-1}((1+r\eta)^{i-1})m(e_i) \\
&\leq \tau_{t-1} - (1+r\eta)^{k-1}m(v'_t) - \sum_{i=1}^{p-1}((1+r\eta)^{i-1})m(v_t) \\
&= \tau_{t-1} - (1+r\eta)^{k-1}m(v'_t) + \frac{1-(1+r\eta)^{p-1}}{r\eta} m(v_t).
\end{aligned} \quad (2)$$

By Equations 1 and 2, Inequality (b) holds. □

Next, we show $\beta\tau$ is also monotone non-decreasing.

Lemma 4. *Let τ_{t-1} and τ_t be two adjacent thresholds and β_{t-1}, β_t be the corresponding parameters. We have $\beta_{t-1}\tau_{t-1} \leq \beta_t\tau_t$, i.e., $\beta\tau$ is monotone non-decreasing. Moreover, if $|S_{t-1}| < k$, we have*

$$\beta_t\tau_t - \beta_{t-1}\tau_{t-1} \geq \beta_t\tau_{t-1}.$$

Proof. When $|S_{t-1}| = k$, $\beta_{t-1}\tau_{t-1} \leq \beta_t\tau_t$ trivially holds since $\beta_{t-1} = \beta_t$ and $\tau_{t-1} \leq \tau_t$ (Lemma 2). When $|S_{t-1}| < k$, by Algorithm 1 we have

$$\begin{aligned}
\beta_t\tau_t - \beta_{t-1}\tau_{t-1} &\geq \beta_t(\tau_{t-1} + m(v_t)) - \beta_{t-1}\tau_{t-1} \\
&\geq \beta_t(\tau_{t-1} + \frac{\beta_{t-1}}{k}\tau_{t-1}) - \beta_{t-1}\tau_{t-1} \geq \beta_t\tau_{t-1}.
\end{aligned}$$

□

Then, let S^*_t and A_t be the optimal solution set and the set of all the ever accepted elements respectively at timestamp t. We obtain the upper bound of S^*_t.

Lemma 5. *An upper bound of the optimal solution S^*_t is*

$$f(S^*_t) \leq \sum_{e\in A_t} m(e) + \beta_t\tau_t.$$

Proof. By monotonicity and submodularity of f and $S_t \subseteq A_t$, we have

$$\begin{aligned}
f(S^*_t) &\leq f(S^*_t \cup A_t) \\
&\leq f(A_t) + \sum_{e\in S^*\setminus A_t} \Delta_f(e|A_{[e]-1}) \\
&= \sum_{e\in A_t} \Delta_f(e|A_{[e]-1}) + \sum_{e\in S^*_t\setminus A_t} \Delta_f(e|A_{[e]-1}) \\
&\leq \sum_{e\in A_t} \Delta_f(e|S_{[e]-1}) + \sum_{e\in S^*_t\setminus A_t} \Delta_f(e|S_{[e]-1}) \\
&= \sum_{e\in A_t} m(e) + \sum_{e\in S^*_t\setminus A_t} m(e).
\end{aligned}$$

Since the elements in $S_t^* \setminus A_t$ are all rejected by the algorithm and by Lemma 4, we further obtain

$$f(S_t^*) \leq \sum_{e\in A_t} m(e) + \sum_{e\in S_t^*\setminus A_t} \frac{\beta_{[e]-1}\tau_{[e]-1}}{k} \leq \sum_{e\in A_t} m(e) + \beta_t\tau_t.$$

□

We are now ready to prove the competitive ratio of ONLINEADAPTIVE.

Theorem 1. *Algorithm 1 is $\frac{1}{\rho_{k,r}}$-competitive, where*

$$\rho_{k,r} = 1 + kr\eta + \frac{1+kr\eta}{(1+r\eta)^k - 1}.$$

Proof. We first consider the case that $|S_{t-1}| < k$ and $A_t = S_t$. We have

$$\begin{aligned} f(S_t^*) &\leq \sum_{e\in A_t} m(e) + \beta_t\tau_t = \sum_{e\in S_t} m(e) + \beta_t\tau_t \\ &= \sum_{i=1}^{|S_t|} m(e_i) + \beta_t \sum_{i=1}^{|S_t|} [(1+\alpha_t)^{i-1} \cdot m(e_i)] \\ &\leq (1 + \beta_t(1+\alpha_t)^{k-1}) \sum_{i=1}^{|S_t|} m(e_i). \end{aligned}$$

By Lemma 1 and monotonicity of α_t, we obtain

$$\begin{aligned} f(S_t^*) &\leq (1 + \frac{1+k\alpha_t}{(1+\alpha_t)^k - 1}(1+\alpha_t)^{k-1}) f(S_t) \\ &\leq (1 + k\alpha_t + \frac{1+k\alpha_t}{(1+\alpha_t)^k - 1}) f(S_t) \\ &\leq (1 + kr\eta + \frac{1+kr\eta}{(1+r\eta)^k - 1}) f(S_t). \end{aligned}$$

Next, we consider the case $|S_{t-1}| = k$. For simplicity, let $U_t = \sum_{e\in A_t} m(e) + \beta_t\tau_t$ denote the upper bound of $f(S_t^*)$ and $L_t = \sum_{e\in S_t} m(e)$ denote the lower bound of $f(S_t)$. Since we have proved $f(S_t^*) \leq U_t \leq \rho_{k,r}L_t \leq \rho_{k,r}f(S_t)$ for the case: $|S_{t-1}| < k$, it is enough to prove the case: $|S_{t-1}| = k$ by induction. If ONLINEADAPTIVE rejects v_t, we have $U_t = U_{t-1}$ and $L_t = L_{t-1}$, then we obtain $f(S_t^*) \leq U_t = U_{t-1} \leq \rho_{k,r}L_{t-1} = \rho_{k,r}L_t \leq \rho_{k,r}f(S_t)$. If the algorithm accepts v_t, we have

$$\begin{aligned} f(S_t^*) &\leq U_t = U_{t-1} + m(v_t) + \beta_t\tau_t - \beta_{t-1}\tau_{t-1} \\ &= U_{t-1} + m(v_t) + \beta_t(\tau_t - \tau_{t-1}) \\ &\leq U_{t-1} + (1+\beta_t)m(v_t) + \beta_t r\eta\tau_{t-1} - \beta_t(1+r\eta)^k m(v_t') \\ &\leq U_{t-1} + (1+\beta_t)m(v_t) + kr\eta m(v_t) - \beta_t(1+r\eta)^k m(v_t') \\ &= U_{t-1} + \rho_{k,r}m(v_t) - \rho_{k,r}m(v_t') \\ &\leq \rho_{k,r}L_{t-1} + \rho_{k,r}m(v_t) - \rho_{k,r}m(v_t') \\ &= \rho_{k,r}(L_{t-1} + m(v_t) - m(v_t')) = \rho_{k,r}L_t \leq \rho_{k,r}f(S_t). \end{aligned}$$

Here, the second and third inequalities hold by Lemma 3 and Algorithm 1 respectively, and the fourth inequality is induced from the case $|S_{t-1}| < k$. Overall, the competitive ratio of our ONLINEADAPTIVE algorithm is at least $\frac{1}{\rho_{k,r}}$. □

Corollary 1. *For $r = 1$, the competitive ratio of* ONLINEADAPTIVE *is at least $\frac{1}{4}$. Moreover, the competitive ratio is at least 0.2959 when $k \geq 4$, and 0.3178 when k approaches infinity.*

Proof. Due to the proof of Theorem 1, $\rho_{k,r}$ is set to the maximum of $g(\eta) = 1 + k\eta + \frac{1+k\eta}{(1+\eta)^k - 1}$ where η is the positive root of the equation $(1+x)^{(k+1)} = kx + x + 2$. When $k = 1$, we have $\eta = 1$, and the competitive ratio is at least $\frac{1}{\rho_{1,1}} = \frac{1}{4}$. When $k = 4$, we have $\eta \approx 0.2756$, so the competitive ratio is at least $\frac{1}{\rho_{4,1}} \approx \frac{1}{3.3784} \approx 0.2959$. When k approaches infinity, we can easily have $\rho_{k,r} \geq g(\frac{\eta}{k})$. Thus, we estimate the maximum of $g(\frac{\eta}{k}) = 1 + \eta + \frac{1+\eta}{(1+\frac{\eta}{k})^k - 1}$. When k approaches infinity, we have $(1+\frac{\eta}{k})^k \approx e^\eta$, $g(\frac{\eta}{k}) = 1 + \eta + \frac{1+\eta}{e^\eta - 1}$ and $g(\frac{\eta}{k})$ reaches its maximum 3.1461 at $\eta \approx 1.1461$. Hence, when k approaches infinity, the competitive ratio is at least $\frac{1}{3.1461} \approx 0.3178$. □

Finally, we provide the memory complexity and the number of oracles of ONLINEADAPTIVE.

Corollary 2. *Algorithm 1 uses $O(k)$ memory and the number of oracles is just one per element.*

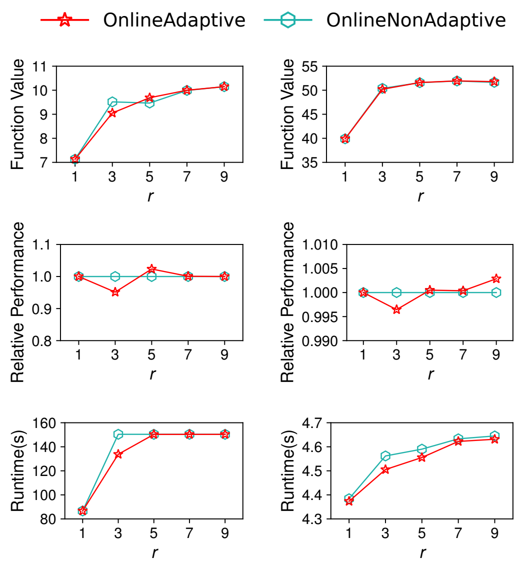
The results can be easily derived from the process of Algorithm 1. Moreover, we maintain the result set by a priority queue where the marginal gains are ordered in the queue.

6 Experimental Evaluation

In this section, we experimentally evaluate our proposed ONLINEADAPTIVE algorithm on three applications corresponding to five real-world datasets, i.e., the first three datasets to the first application and the last two datasets to the second and third applications respectively. We begin by comparing ONLINEADAPTIVE with its non-adaptive version to assess the advantages of exploiting adaptive thresholds, and subsequently compare ONLINEADAPTIVE with five well-established algorithms to evaluate its quality and efficiency. All the experiments were conducted on a machine running Ubuntu 20.04 with an Intel(R)Xeon(R) E3-1225 3.30GHz CPU and 16 GB main memory.

Name	Size	Dim.	Reference
ForestCover	286,048	10	[Liu *et al.*, 2008]
CreditCardFraud	284,807	29	[Pozzolo *et al.*, 2015]
KDDCup99	48,113	79	[Campos *et al.*, 2016]
YouTube	9,010	4	[Kazemi *et al.*, 2019]
Twitter	42,104	-	[Kazemi *et al.*, 2019]

Table 1: Five datasets corresponding to the listed applications.



(a) ForestCover (b) YouTube

Figure 1: Comparison between ONLINEADAPTIVE and its non-adaptive version ONLINENONADAPTIVE for different rs when $k = 30$ on the ForestCover and YouTube datasets. Note that the second row shows the relative performance to ONLINENONADAPTIVE. We omit the results on number of oracles for they are the same.

6.1 Applications and Datasets

We maximize the corresponding submodular functions in the following three applications.

Online Kernel Prototype Selection. In this application, we want to select a set S with k kernel prototypes from each in the three datasets as shown in the top group in Table **??**, and each element e of the datasets is a multi-dimensional representative vector. The goal is to maximize the log-determinant $f(S) = \frac{1}{2}\log\det(I + \kappa M_S)$ using Gaussian kernel $K_G(x, y) = \exp(-\frac{||x-y||_2^2}{2\sigma^2})$. Here, I denotes the identity matrix, $\kappa, \sigma \in \mathbb{R}_+$ are parameters, and M_S denotes the kernel matrix based on all pairs of the elements in S, where $M_S[i][j] = K_G(e_i, e_j)$ for a pair of elements (e_i, e_j). This function is shown to be submodular [Schlegel *et al.*, 2017], and we set $\kappa = 1$ and $\sigma = \frac{1}{2\sqrt{d}}$ in our experiments, where d denotes the dimensionality of the element in each dataset.

Online Video Summarization. In this application, we want to select a set S with k representative frames from the YouTube dataset as shown in Table **??**, and each element e of the dataset is a 4-dimensional representative vector compressed from a frame. The goal is to maximize the log-determinant $f(S) = \log\det(I + \kappa M_S)$ using Laplacian kernel $K_L(x, y) = \exp(-\frac{||x-y||_2}{\sigma})$, where $M_S[i][j] = K_L(e_i, e_j)$. Note that the submodular function and the kernel function used here are different from those used in the first application, and we set $\kappa = 10$ and $\sigma = 1$ in our experiments.

Online Text Summarization. In this application, we want to select a set S with k tweets from the Twitter dataset as shown in Table **??**. Each element e in the ground set V is a tweet with a group of keywords W_e and a number of retweets N_e. The score of a word $w \in W_e$ for a tweet e is defined by $val(w, e) = N_e$ and $val(w, e) = 0$ if $w \notin W_e$. Let $\mathcal{W}$ be the general set including all the words in the ground set V. The goal is to maximize $f(S) = \sum_{w\in\mathcal{W}} \sqrt{\sum_{e\in S} val(w, e)}$ and it is shown to be submodular [Kazemi *et al.*, 2019].

6.2 Baselines and Evaluation Metrics

In the first experiment, we use the non-adaptive version of ONLINEADAPTIVE, called ONLINENONADAPTIVE for comparison. For all the timestamps $t \in \{1, 2, \ldots, n\}$, ONLINENONADAPTIVE sets all α_ts to the constant $r\eta$. Thus, β_ts also remain unchanged. In the second experiment, we use the classical offline algorithm GREEDY and the four above-mentioned online algorithms, INDEPENDENTSETIMPROVEMENT, STREAMINGGREEDY, PREEMPTION, and FREEDISPOSAL for comparison. The GREEDY algorithm is not an online algorithm but with the best solution quality, and we put GREEDY here to show how far the online algorithms are from it. Moreover, since PREEMPTION accepts a parameter c and achieves a competitive ratio $\frac{c}{(c+1)^2}$, we set $c = 1$ to achieve its best ratio $\frac{1}{4}$ in our experiments.

To comprehensively evaluate the performance of the algorithms, we employ the following four metrics:

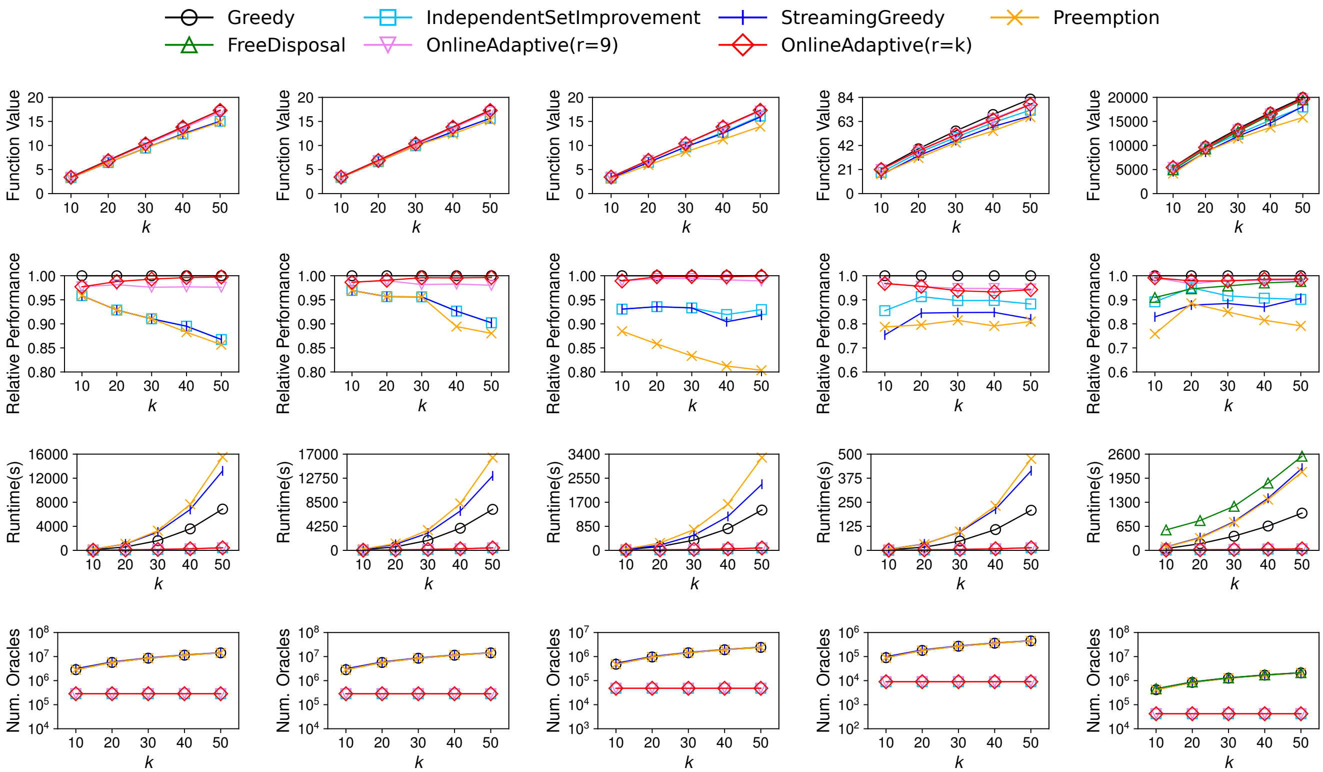
- **Function Value**: The function value of the selected solution set S, i.e., $f(S)$, which intuitively reflects the algorithm's variation trends and its performance in terms of effectiveness.
- **Relative Performance**: The relative performance in terms of function value to the specified algorithm, i.e., ONLINENONADAPTIVE or GREEDY, which facilitates the comparison among the algorithms for effectiveness.
- **Runtime**: The total runtime in seconds, which provides an intuitive reflection of the algorithms' efficiency and feasibility.
- **Number of Oracles**: The number of total oracles, which reflects efficiency and feasibility of the algorithms.

Note that larger values are preferred for the first two metrics while smaller values are better for the last two metrics. In addition, *runtime* cannot be wholly substituted by *number of oracles* since for different applications, the running time of an oracle varies a lot. Our code is publicly available [1].

6.3 Performance of Our Algorithm

To illustrate the advantages of employing the adaptive thresholding strategy, we first compare ONLINEADAPTIVE with its non-adaptive version ONLINENONADAPTIVE by varying $r \in \{1, 3, 5, 7, 9\}$ while fixing the solution size k to 30. Since they have exactly the same number of total oracles, we omit the results with respect to this metric. As shown in Figure 1, their function values all exhibit their minimums at $r = 1$ and increase with r, which once again emphasizes the phenomenon that using theoretically optimal parameter, i.e. $r = 1$, does not result in the best actual performance. The reason is that the thresholds for filtering elements become progressively more suitable to increase $f(S)$ with the growth of r which improves the quality of the solution set.

[1] https://github.com/dcsjzh/OnlineAdaptive



(a) ForestCover (b) CreditCardFraud (c) KDDCup99 (d) YouTube (e) Twitter

Figure 2: Comparison between GREEDY(an offline algorithm), INDEPENDENTSETIMPROVEMENT, STREAMINGGREEDY, PREEMPTION, FREEDISPOSAL, and ONLINEADAPTIVE by varying cardinality constraint k. Note that the second row shows the relative performances of the online algorithms to GREEDY, and since FREEDISPOSAL fails to finish within the specified time, i.e., three days, on the first four datasets, i.e., ForestCover, CreditCardFraud, KDDCup99 and YouTube, we only present its results on the last dataset, i.e., Twitter.

Since ONLINENONADAPTIVE selects at least k elements earlier, it has better solution quality when r is small, e.g., $r = 3$. However, when both of the algorithms have selected k elements, the adaptive thresholding strategy of ONLINEADAPTIVE results in superior quality for the ONLINEADAPTIVE algorithm only accepts the elements above the thresholds and is with a smaller number of replacements. Thus, ONLINEADAPTIVE consumes less time than the ONLINENONADAPTIVE algorithm.

To evaluate the effectiveness and efficiency of ONLINEADAPTIVE, we compare it with five well-established algorithms by varying solution size $k \in \{10, 20, 30, 40, 50\}$. Here, we use ONLINEADAPTIVE with $r = 9$ and ONLINEADAPTIVE with $r = k$, where the former is due to its good performance as shown in Figure 1 and the latter comes from a simple fact that users are not required to specify r. Note that since FREEDISPOSAL uses the set of all ever accepted elements to evaluate each arriving element, it fails to finish within the specified time, i.e., three days, on the first four datasets. Thus, we only present its results on the Twitter dataset. As shown in Figure 2, except the offline algorithm GREEDY, our ONLINEADAPTIVE($r = 9$) and ONLINEADAPTIVE($r = k$) algorithms both exhibit superior performance in terms of function value, runtime and number of oracles. As k increases, it can be observed that ONLINEADAPTIVE($r = 9$), ONLINEADAPTIVE($r = k$) and FREEDISPOSAL maintain relatively good performance compared to GREEDY, while the relative performances of the other algorithms tend to decrease. This is due to the fact that their competitive ratios increase with k. Moreover, ONLINEADAPTIVE($r = k$) performs almost the best among them.

7 Conclusion

In this paper, we investigated the online submodular maximization problem under cardinality constraint. We proposed the ONLINEADAPTIVE algorithm with the adaptive thresholding strategy that not only simultaneously achieved the current state-of-the-art in terms of competitive ratio, memory complexity, running time and number of oracles, but also demonstrated significantly better empirical performance in real-world applications than existing online algorithms. Especially, the competitive ratio of ONLINEADAPTIVE is at least $1/4$, and is about 0.2959 when $k \geq 4$ and approaches 0.3178 as k tends to infinity. Besides, it just uses $O(k)$ memory and performs one oracle per element. An interesting future work is to generalize the ONLINEADAPTIVE framework to weakly submodular functions in the online setting.

References

[Ashkan *et al.*, 2015] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. Optimal greedy diversity for recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1742–1748, 2015.

[Badanidiyuru *et al.*, 2014] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 671–680, 2014.

[Bateni *et al.*, 2013] Mohammad Hossein Bateni, Mohammad Taghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Transactions on Algorithms*, 9(4):32:1–32:23, 2013.

[Becker *et al.*, 2022] Ruben Becker, Gianlorenzo D'Angelo, Sajjad Ghobadi, and Hugo Gilbert. Fairness in influence maximization through randomization. *Journal of Artificial Intelligence Research*, 73:1251–1283, 2022.

[Bilmes, 2023] Jeff Bilmes. Submodular optimization. In Kevin P. Murphy, editor, *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[Buchbinder *et al.*, 2015] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1202–1216, 2015.

[Buchbinder *et al.*, 2019] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. *ACM Transactions on Algorithms*, 15(3):30:1–30:31, 2019.

[Buschjäger *et al.*, 2021] Sebastian Buschjäger, Philipp-Jan Honysz, Lukas Pfahler, and Katharina Morik. Very fast streaming submodular function maximization. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 151–166, 2021.

[Campos *et al.*, 2016] Guilherme Oliveira Campos, Arthur Zimek, Jörg Sander, Ricardo J. G. B. Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E. Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927, 2016.

[Chakrabarti and Kale, 2015] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154:225–247, 2015.

[Chan *et al.*, 2017] T.-H. Hubert Chan, Zhiyi Huang, Shaofeng H.-C. Jiang, Ning Kang, and Zhihao Gavin Tang. Online submodular maximization with free disposal: Randomization beats ¼ for partition matroids. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1204–1223, 2017.

[Chan *et al.*, 2018] T.-H. Hubert Chan, Zhiyi Huang, Shaofeng H.-C. Jiang, Ning Kang, and Zhihao Gavin Tang. Online submodular maximization with free disposal. *ACM Transactions on Algorithms*, 14(4):56:1–56:29, 2018.

[Chekuri *et al.*, 2015] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 318–330, 2015.

[Ene and Nguyen, 2022] Alina Ene and Huy Nguyen. Streaming algorithm for monotone k-submodular maximization with cardinality constraints. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5944–5967, 2022.

[Gomes and Krause, 2010] Ryan Gomes and Andreas Krause. Budgeted nonparametric learning from data streams. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 391–398, 2010.

[Hajiaghayi *et al.*, 2004] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, page 71–80, 2004.

[Kazemi *et al.*, 2019] Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3311–3320, 2019.

[Krause and Golovin, 2014] Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. 2014.

[Kuhnle, 2021] Alan Kuhnle. Quick streaming algorithms for maximization of monotone submodular functions in linear time. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1360–1368, 2021.

[Kumari and Bilmes, 2021] Lilly Kumari and Jeff A. Bilmes. Submodular span, with applications to conditional data summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 12344–12352, 2021.

[Liu *et al.*, 2008] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 413–422, 2008.

[Mirzasoleiman *et al.*, 2016] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1358–1367, 2016.

[Nemhauser *et al.*, 1978] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

[Norouzi-Fard *et al.*, 2018] Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3826–3835, 2018.

[Pozzolo *et al.*, 2015] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 159–166, 2015.

[Schlegel *et al.*, 2017] Matthew Schlegel, Yangchen Pan, Jiecao Chen, and Martha White. Adapting kernel representations online using submodular maximization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3037–3046, 2017.

[Si-Salem *et al.*, 2024] Tareq Si-Salem, Gözde Özcan, Iasonas Nikolaou, Evimaria Terzi, and Stratis Ioannidis. Online submodular maximization via online convex optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 15038–15046, 2024.

[Streeter *et al.*, 2009] Matthew J. Streeter, Daniel Golovin, and Andreas Krause. Online learning of assignments. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1794–1802, 2009.

[Wei *et al.*, 2015] Kai Wei, Rishabh K. Iyer, and Jeff A. Bilmes. Submodularity in data subset selection and active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1954–1963, 2015.